Simple Linear Classifiers via Discrete Optimization: Learning Certifiably Optimal Scoring Systems for Decision-Making and Risk Assessment

by

Berk Ustun

B.S., Industrial Engineering and Operations Research, U.C. Berkeley (2009)
B.A., Economics, U.C. Berkeley (2009)
S.M., Computation for Design and Optimization, M.I.T. (2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
August 31, 2017

Signature redacted

Certified by....................................
Cynthia Rudin
Associate Professor of Computer Science
Duke University
Thesis Supervisor

Signature redacted

Accepted by................................
Leslie A. Kolodjiezski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Simple Linear Classifiers via Discrete Optimization: Learning Certifiably Optimal Scoring Systems for Decision-Making and Risk Assessment

by

Berk Ustun

Submitted to the Department of Electrical Engineering and Computer Science on August 31, 2017, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering

Abstract

Scoring systems are linear classification models that let users make quick predictions by adding, subtracting, and multiplying a few small numbers. These models are widely used in applications where humans have traditionally made decisions because they are easy to understand and validate. In spite of extensive deployment, many scoring systems are still built using ad hoc approaches that combine statistical techniques, heuristics, and expert judgement. Such approaches impose steep trade-offs with performance, making it difficult for practitioners to build scoring systems that will be used and accepted.

In this dissertation, we present two new machine learning methods to learn scoring systems from data: *Supersparse Linear Integer Models* (SLIM) for decision-making applications; and *Risk-calibrated Supersparse Linear Integer Models* (RiskSLIM) for risk assessment applications. Both SLIM and RiskSLIM solve discrete optimization problems to learn scoring systems that are fully optimized for feature selection, small integer coefficients, and operational constraints. We formulate these problems as integer programming problems and develop specialized algorithms to recover certifiably optimal solutions with an integer programming solver.

We illustrate the benefits of this approach by building scoring systems for real-world problems such as recidivism prediction, sleep apnea screening, ICU seizure prediction, and adult ADHD diagnosis. Our results show that a discrete optimization approach can learn simple models that perform well in comparison to the state-of-the-art, but that are far easier to customize, understand, and validate.

Thesis Supervisor: Cynthia Rudin
Title: Associate Professor of Computer Science
Duke University

Acknowledgments

I would like to thank my advisor, Cynthia Rudin, for introducing me to this class of problems, for working with me as we struggled to solve them, and then for being patient when I wanted to do things a little differently.

I would also like to thank my committee members, Stefanie Jegelka and Leslie Kaelbling, for helping me frame the methods in this dissertation, for identifying new problems, and for fostering a fun environment for machine learning at MIT.

This dissertation also reflects the work of several other talented individuals, namely: Stefano Tracà, who helped with the initial work on SLIM; Jiaming Zeng, who helped with the recidivism prediction application; as well as Brandon Westover, Matt Bianchi, Aaron Struck, Ronald Kessler, and Nancy Sampson, who helped with all the medical applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scoring systems are sparse linear classification models with small integer coefficients. Starting with the work of Burgess (1928), these models have been extensively used for decision-making and risk assessment in domains where humans have traditionally made decisions. They are currently deployed for a large number of applications in medicine (e.g. to predict mortality in the ICU, or diagnose PTSD), criminal justice (e.g. to assess recidivism risk), and finance (e.g. to assess creditworthiness).

The widespread deployment of scoring systems is inherently related to the fact that they are easy to use, understand, and validate. On one hand, sparse linear models with small integer coefficients let users make quick predictions through simple arithmetic, without a computer or a calculator. In medicine, for instance, sparsity and small integer coefficients produce models that physicians can memorize using a mnemonic (see e.g. the HEART score of Six et al., 2008, in Figure 1.3). On the other hand, these qualities also address key limitations in human cognition, such as limits in our ability to handle over four items in working memory (Cowan, 2010), and to track associations between three or more entities (Jennings et al., 1982). In light of these limitations, scoring systems help users understand how multiple input variables are used in the prediction. This allows users to easily validate the model without additional analysis, and provides them with the option to overrule the model in an informed manner when needed.

In spite of extensive deployment over the past century, there has been no standardized approach to build scoring systems. This is partially due to the fact that models have to satisfy *operational constraints* in order to be used and accepted in such domains (see e.g. the requirements for the EDACS score in Figure 1.6). Operational constraints are difficult to address in a systematic manner because they are related to ill-defined model qualities (e.g. usability, understandability, and alignment with domain expertise) that can change significantly across applications. As a result, the majority of scoring systems are still developed *ad hoc*. In some cases, models are built by combining traditional statistical methods with heuristics and expert judgement (e.g. preliminary feature selection by experts, logistic regression, scaling, and rounding). In others, models are hand-crafted by a panel of experts and data is used for validation purposes only (e.g. for the CHADS_2 score for stroke prediction of Gage et al. 2001, and the National Early Warning Score to assess acute illness in the ICU of McGinley and Pearse 2012).

The lack of a standardized approach to create scoring systems has important consequences given their pervasive use in high-stakes applications. Models that are used for applications such as sentencing and credit scoring may not perform as well as they could. Moreover, model development may involve significant time and resources as practitioners have to design entirely new approaches to address operational constraints. A unified approach can improve decisions in these domains by making it easier for practitioners to build models that perform well and that address the constraints needed for deployment.

In this dissertation, we introduce a modern machine learning approach to learn scoring systems from data:

- *Supersparse Linear Integer Models* (SLIM) to create data-driven scoring systems for decision-making;
- *Risk-calibrated Supersparse Linear Integer Models* (RiskSLIM) to create data-driven scoring systems for risk assessment.

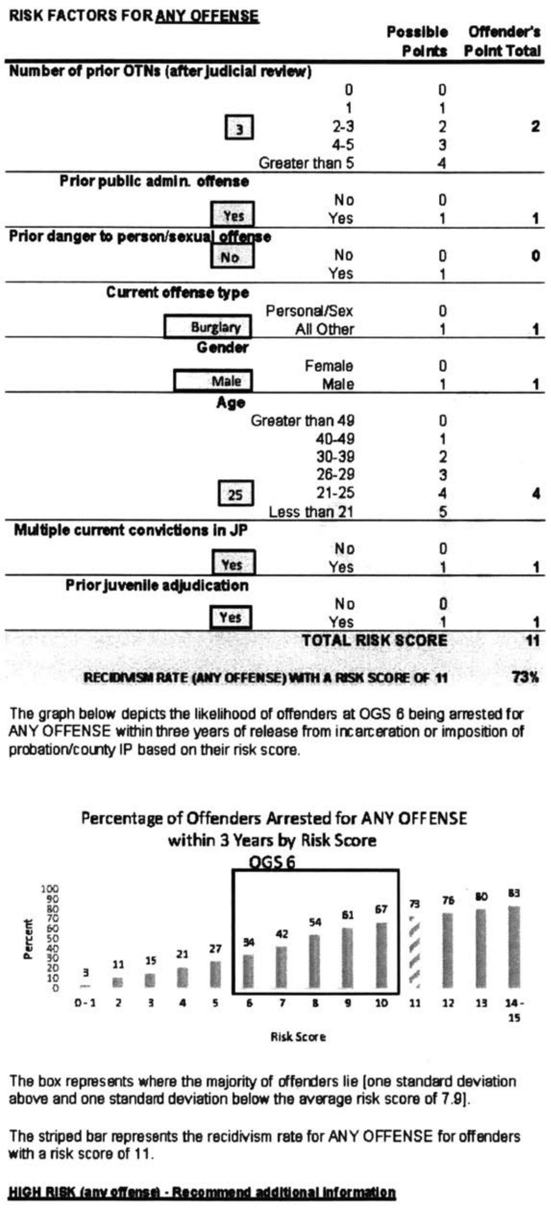
In contrast to traditional approaches, both SLIM and RiskSLIM are designed to learn scoring systems that are fully optimized for feature selection, small integer coefficients, and operational constraints. To this end, they solve discrete problems that optimize and constrain *exact* quantities related to model performance and model form. We formulate these problems as *integer programs* (IP) or *mixed-integer programs* (MIP) and aim to recover certifiably optimal solutions using a MIP solver. This approach requires the solution to computationally difficult optimization problems. However, it has several major benefits in this setting:

1. It does not need to sacrifice accuracy to satisfy constraints on model form as discrete optimization can *directly* optimize, penalize and constrain discrete model qualities such as the number of mistakes (via the 0-1 loss) or the number of features (via the ℓ_0 penalty).

2. It avoids the ad-hoc training process required by current methods. Specifically, practitioners can directly encode a wide range of operational constraints into the IP formulation, and fit scoring systems without post-processing or parameter tuning. Further, they can solve this formulation using a commercial solver, without the need to implement a new algorithm.

3. It can pair models with a certificate of optimality, which be used to make informed choices between models in the presence of real-world constraints.

In light of these benefits, a major goal of this dissertation is to recover certifiably optimal solutions to these problems for the largest possible datasets. Over the past 30 years, developments in computer hardware and integer programming have allowed modern MIP solvers to handle combinatorial optimization problems for a wide range of real-world applications. In our setting, commercial solvers can recover good feasible solutions for small to mid-sized instances when used off-the-shelf. On larger instances, however, an off-the-shelf approach is unable to find good feasible solutions or pair models with a certificate of optimality. To address this, we develop



RISK FACTORS FOR ANY OFFENSE

	Selected	Category	Possible Points	Offender's Point Total
Number of prior OTNs (after judicial review)				
		0	0	
		1	1	
	3	2-3	2	2
		4-5	3	
		Greater than 5	4	
Prior public admin. offense				
		No	0	
	Yes	Yes	1	1
Prior danger to person/sexual offense				
	No	No	0	0
		Yes	1	
Current offense type				
		Personal/Sex	0	
	Burglary	All Other	1	1
Gender				
		Female	0	
	Male	Male	1	1
Age				
		Greater than 49	0	
		40-49	1	
		30-39	2	
		26-29	3	
	25	21-25	4	4
		Less than 21	5	
Multiple current convictions in JP				
		No	0	
	Yes	Yes	1	1
Prior juvenile adjudication				
		No	0	
	Yes	Yes	1	1
			TOTAL RISK SCORE	**11**

RECIDIVISM RATE (ANY OFFENSE) WITH A RISK SCORE OF 11 **73%**

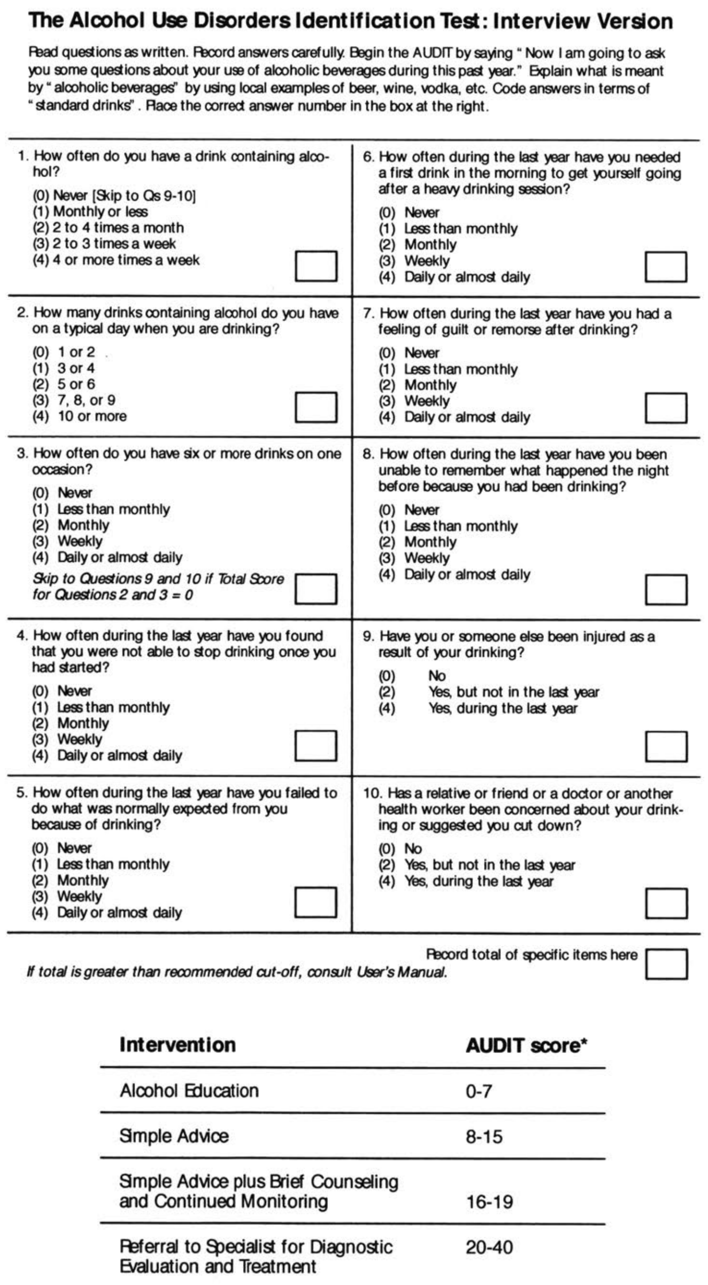
The graph below depicts the likelihood of offenders at OGS 6 being arrested for ANY OFFENSE within three years of release from incarceration or imposition of probation/county IP based on their risk score.

The box represents where the majority of offenders lie [one standard deviation above and one standard deviation below the average risk score of 7.9].

The striped bar represents the recidivism rate for ANY OFFENSE for offenders with a risk score of 11.

HIGH RISK (any offense) - Recommend additional information

Figure 1.1: Risk score developed by the Pennsylvania Commission on Sentencing (Pennsylvania Bulletin, 2017). The creation and validation of this model took over 7 years: model development started in 2010 (Pennsylvania Code, 2010) and a final model was approved for use in 2017 (Pennsylvania Bulletin, 2017).



The Alcohol Use Disorders Identification Test: Interview Version

Read questions as written. Record answers carefully. Begin the AUDIT by saying " Now I am going to ask you some questions about your use of alcoholic beverages during this past year." Explain what is meant by " alcoholic beverages" by using local examples of beer, wine, vodka, etc. Code answers in terms of " standard drinks" . Place the correct answer number in the box at the right.

1. How often do you have a drink containing alcohol?
   (0) Never [Skip to Qs 9-10]
   (1) Monthly or less
   (2) 2 to 4 times a month
   (3) 2 to 3 times a week
   (4) 4 or more times a week

2. How many drinks containing alcohol do you have on a typical day when you are drinking?
   (0) 1 or 2
   (1) 3 or 4
   (2) 5 or 6
   (3) 7, 8, or 9
   (4) 10 or more

3. How often do you have six or more drinks on one occasion?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily
   Skip to Questions 9 and 10 if Total Score for Questions 2 and 3 = 0

4. How often during the last year have you found that you were not able to stop drinking once you had started?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily

5. How often during the last year have you failed to do what was normally expected from you because of drinking?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily

6. How often during the last year have you needed a first drink in the morning to get yourself going after a heavy drinking session?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily

7. How often during the last year have you had a feeling of guilt or remorse after drinking?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily

8. How often during the last year have you been unable to remember what happened the night before because you had been drinking?
   (0) Never
   (1) Less than monthly
   (2) Monthly
   (3) Weekly
   (4) Daily or almost daily

9. Have you or someone else been injured as a result of your drinking?
   (0) No
   (2) Yes, but not in the last year
   (4) Yes, during the last year

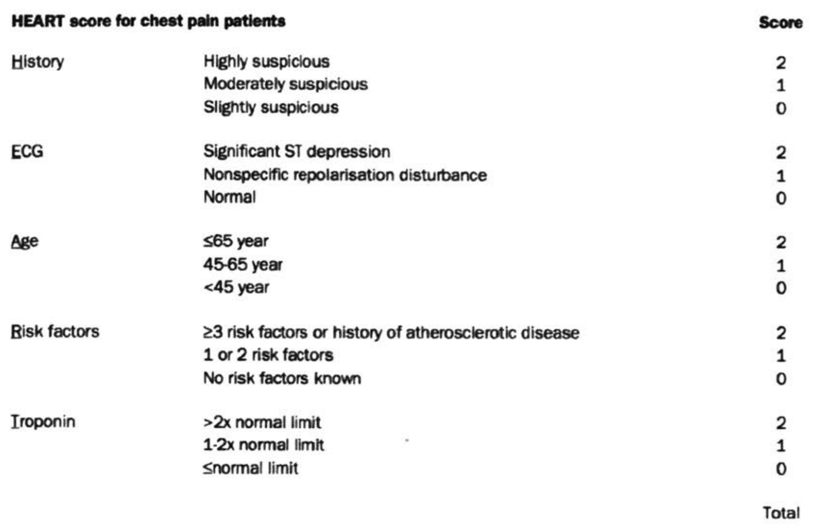
10. Has a relative or friend or a doctor or another health worker been concerned about your drinking or suggested you cut down?
    (0) No
    (2) Yes, but not in the last year
    (4) Yes, during the last year

Record total of specific items here

If total is greater than recommended cut-off, consult User's Manual.

Intervention	**AUDIT score***
Alcohol Education	0-7
Simple Advice	8-15
Simple Advice plus Brief Counseling and Continued Monitoring	16-19
Referral to Specialist for Diagnostic Evaluation and Treatment	20-40

Figure 1.2: AUDIT scoring system for alcohol use disorders (Babor et al., 2001).



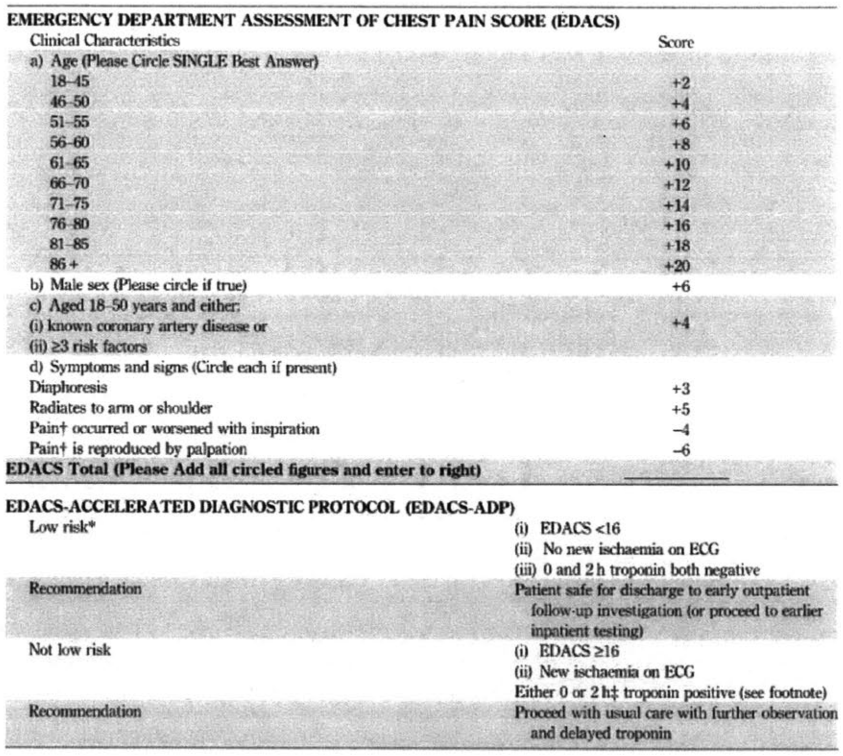
HEART score for chest pain patients		Score
History	Highly suspicious	2
	Moderately suspicious	1
	Slightly suspicious	0
ECG	Significant ST depression	2
	Nonspecific repolarisation disturbance	1
	Normal	0
Age	≤65 year	2
	45-65 year	1
	<45 year	0
Risk factors	≥3 risk factors or history of atherosclerotic disease	2
	1 or 2 risk factors	1
	No risk factors known	0
Troponin	>2x normal limit	2
	1-2x normal limit	1
	≤normal limit	0
		Total

Figure 1.3: HEART score to screen for adverse cardiac events in the emergency room (Six et al., 2008). This model is designed to be used as a mnemonic. Six et al. (2008) recommend discharging patients with a total score ≤ 3, monitoring patients with a total score between 4 to 6, and pursuing aggressive treatment for a total score ≥ 7.

specialized techniques that can be used effectively with modern solvers, such as algorithms to reduce data-related computation, heuristics to produce feasible solutions, and techniques to close the optimality gap.

Our work provides a principled approach to create these simple predictive models that have been used for nearly a century. The broader contributions of this dissertation include:

- We show that simple scoring systems can perform just as well as state-of-the-art machine learning methods.
- We develop new techniques and algorithms to effectively solve an important class of risk minimization problems.
- We demonstrate the value of discrete optimization in terms of improved performance, customization, reduced parameter tuning.
- We present theoretical results on sparse linear models with integer coefficients. These results provide insights as to why scoring systems perform well, why they generalize, and why they are easy to understand.
- We illustrate the value of scoring systems through real-world applications in medicine and criminal justice, where model deployment hinges on the ability to address qualitative constraints such as usability and interpretability.



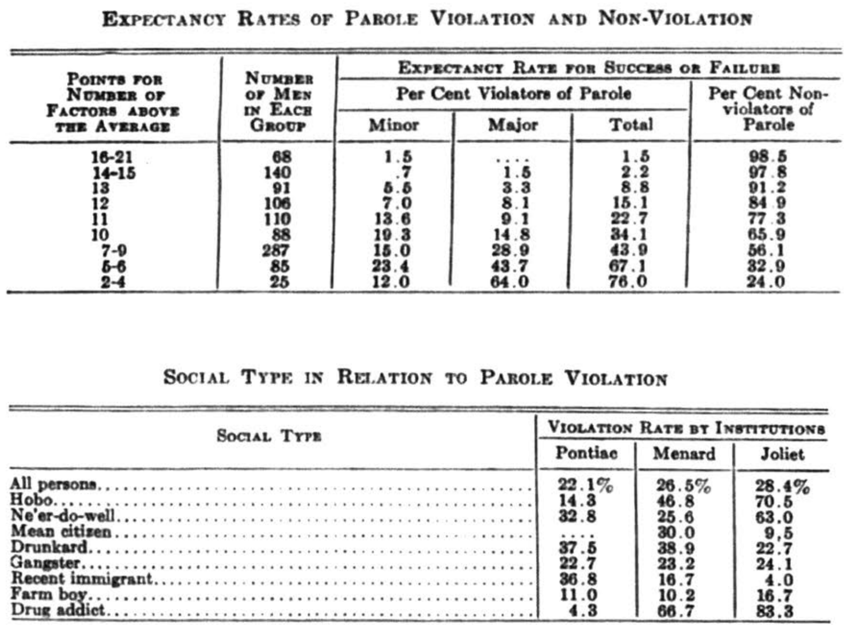
EMERGENCY DEPARTMENT ASSESSMENT OF CHEST PAIN SCORE (EDACS)

Clinical Characteristics	Score
a) Age (Please Circle SINGLE Best Answer)	
18–45	+2
46–50	+4
51–55	+6
56–60	+8
61–65	+10
66–70	+12
71–75	+14
76–80	+16
81–85	+18
86 +	+20
b) Male sex (Please circle if true)	+6
c) Aged 18–50 years and either: (i) known coronary artery disease or (ii) ≥3 risk factors	+4
d) Symptoms and signs (Circle each if present)	
Diaphoresis	+3
Radiates to arm or shoulder	+5
Pain† occurred or worsened with inspiration	−4
Pain† is reproduced by palpation	−6
EDACS Total (Please Add all circled figures and enter to right)	________

EDACS-ACCELERATED DIAGNOSTIC PROTOCOL (EDACS-ADP)

Low risk*	(i) EDACS <16 (ii) No new ischaemia on ECG (iii) 0 and 2 h troponin both negative
Recommendation	Patient safe for discharge to early outpatient follow-up investigation (or proceed to earlier inpatient testing)
Not low risk	(i) EDACS ≥16 (ii) New ischaemia on ECG Either 0 or 2 h‡ troponin positive (see footnote)
Recommendation	Proceed with usual care with further observation and delayed troponin

Figure 1.4: EDACS score to screen for adverse cardiac events in the emergency room (Than et al., 2014). EDACS appears to be a risk assessment tool, but it does not produce risk estimates. Here, the model is effectively being used as a decision-making tool: the corresponding decision rule is to predict that someone is at "high-risk" of an adverse cardiac event if the total score ≥ 16.



Expectancy Rates of Parole Violation and Non-Violation

Points for Number of Factors above the Average	Number of Men in Each Group	Expectancy Rate for Success or Failure			
		Per Cent Violators of Parole			Per Cent Non-violators of Parole
		Minor	Major	Total	
16-21	68	1.5	....	1.5	98.5
14-15	140	.7	1.5	2.2	97.8
13	91	5.5	3.3	8.8	91.2
12	106	7.0	8.1	15.1	84.9
11	110	13.6	9.1	22.7	77.3
10	88	19.3	14.8	34.1	65.9
7-9	287	15.0	28.9	43.9	56.1
5-6	85	23.4	43.7	67.1	32.9
2-4	25	12.0	64.0	76.0	24.0

Social Type in Relation to Parole Violation

Social Type	Violation Rate by Institutions		
	Pontiac	Menard	Joliet
All persons	22.1%	26.5%	28.4%
Hobo	14.3	46.8	70.5
Ne'er-do-well	32.8	25.6	63.0
Mean citizen	....	30.0	9,5
Drunkard	37.5	38.9	22.7
Gangster	22.7	23.2	24.1
Recent immigrant	36.8	16.7	4.0
Farm boy	11.0	10.2	16.7
Drug addict	4.3	66.7	83.3

Figure 1.5: Risk score proposed to predict success on parole by Burgess (1928). The model assigns a *total score* for each prisoner by summing the points from 21 factors (selected by domain expertise). We show the table used to determine risk (top) along with the table used to determine points for the "social type" factor (bottom). Here, a person receives a point if the violation rate for their social type exceeds the "average for all persons"

1.1 Background

In what follows, we discuss related work in applied domains, machine learning, and discrete optimization. We aim to provide background on the use and development of scoring systems and to describe relevant work in machine learning and optimization at a high level. We include a discussion of related work from a technical perspective in later chapters.

1.1.1 Scoring Systems

In Table 1.1, we present a short list of well-known scoring systems used in medicine, criminal justice, and finance. All of these models are developed for prediction problems where the outcome of interest is a binary variable (e.g. $y_i = +1/-1$ if a client default/does not default on a loan). Although this is the case for most models, some scoring systems are developed for prediction problems where the outcome variable is ordinal (e.g. the pain scales of Payen et al., 2001).

The widespread deployment of scoring systems in these domains is related to the fact that sparse linear models are easy to use, understand, and validate. In Table 1.2, we provide a list of statements where authors in each of the previous domains describe the benefits of this format.

Explicitly, sparsity and small integer coefficients affect usability, understandability, and validation as follows:

- *Usability*: Sparse linear models with small integer coefficients let users make quick predictions without a computer or a calculator. In many medical applications, for example, models are designed so that they can be memorized using a mnemonic (see e.g., the HEART score of Six et al., 2008).

- *Understandability*: Scoring systems help overcome well-known deficiencies in human cognition, such as limitations in handling more than 4 cognitive entities in working memory (Cowan, 2010), and limitations in estimating the association between 3+ variables (Jennings et al., 1982). In light of these limitations, sparsity limits the number of items in working memory. Linear models impose a flat structure that allows users to gauge the influence of each variable by comparing coefficients, which is easier when a model uses small integer coefficients Reyna and Brainerd (2007). Lastly, when a scoring system uses binary input variables, the decision rule has a Boolean representation, which can further help users understand interactions between multiple variables (see Section 2.1.3).

- *Potential for Validation*: The ability to understand how the model works allows users to validate its predictions, which is crucial in high-stakes applications such as sentencing (Pennsylvania Bulletin, 2017). In particular, a user can see all of the variables that a scoring system uses, gauge the importance of each variable, and understand how the variables interact to produce the predicted outcome. This allows users to validate the model during deployment and override the prediction in an informed way if needed.

Medicine	
SAPS I, II, III, to assess ICU mortality risk	Moreno et al. (2005)
APACHE I, II, III, to assess ICU mortality risk	Knaus et al. (1981, 1985, 1991)
CHADS_2, to assess stroke risk	Gage et al. (2001)
TIMI, to assess risk of death and ischemic events	Antman et al. (2000)
SIRS, to detect system inflammatory response syndrome	Bone et al. (1992)
CURB-65, to screen for pneumonia	Lim et al. (2003)
HEART, to screen for adverse cardiac events	Six et al. (2008)
EDACS, to screen for adverse cardiac events	Than et al. (2014)
AUDIT, to detect harmful alcohol consumption	Babor et al. (2001)
ASRS, to screen for adult ADHD	Kessler et al. (2005b)
PCL, to screen for PTSD	Weathers et al. (2013)
Criminal Justice	
Ohio Risk Assessment System	Latessa et al. (2009)
Kentucky Pretrial Risk Assessment Instrument	Austin et al. (2010)
Salient Factor Score	Hoffman (1994)
Criminal History Category	U.S. Sentencing Commission (1987)
Offense Gravity Score	Pennsylvania Bulletin (2017)
Finance	
Z-Score, to predict bankruptcy	Altman et al. (2000)
F-Score, to assess the strength of a company's balance sheet	Piotroski (2000)
M-Score, to detect manipulation in reported earnings	Beneish et al. (2013)

Table 1.1: Scoring systems used in medicine, criminal justice, and finance. An extensive list of scoring systems used in medicine can be found at www.mdcalc.com. This list does not include models that are used for credit scoring, which have the same format but are not published due to their proprietary nature (see Finlay, 2012).

Domain	Reference	Quote
Medicine	Than et al. (2014)	*"Ease of use might be facilitated by presenting a rule developed from logistic regression as a score, where the original predictor weights have been converted to integers that are easy to add together... Though less precise than the original regression formula, such presentations are less complex, easier to apply by memory and usable without electronic assistance."*
Criminal Justice	Duwe and Kim (2016)	*"It is commonplace... for fine-tuned regression coefficients to be replaced with a simple-point system... to promote the easy implementation, transparency, and interpretability of risk-assessment instruments."*
Finance	Finlay (2012)	*"presenting a linear model in the form of a scorecard is attractive because it's so easy to explain and use. In particular, the score can be calculated using just addition to add up the relevant points that someone receives"*

Table 1.2: Recent quotes on why sparse linear models with small integer coefficients are used in domains such as medicine, criminal justice, and finance.

1.1.2 Model Development

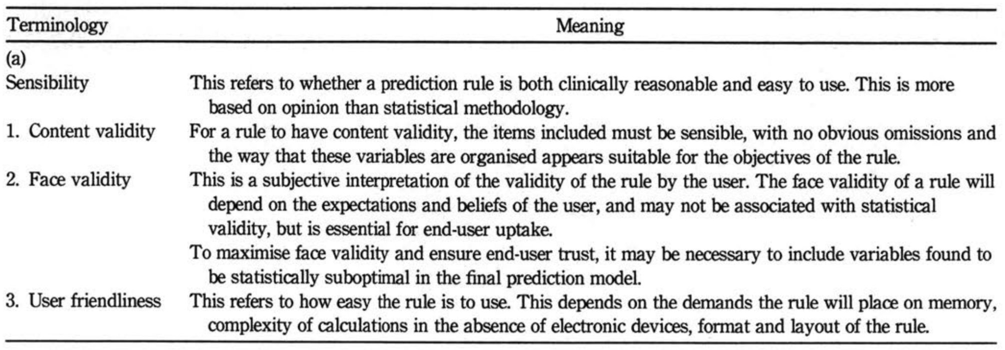
The approaches used to create scoring systems can vary significantly within the same domain (see e.g. the various techniques used in criminal justice described in Gottfredson and Snyder, 2005; Bobko et al., 2007; Duwe and Kim, 2016), and even within similar applications (see e.g. the different approaches used to create models to assess the risk heart-related illness Six et al., 2008; Antman et al., 2000; Than et al., 2014).

A key reason for this is because predictive models in domains such as medicine and criminal justice need to obey additional *operational constraints* to be used and accepted. In some cases, these constraints can be explicitly stated. Reilly and Evans (2006), for example, describe the requirements put forth by physicians when building a model to detect major cardiac complications for patients with chest pain:

> "*Our physicians... insisted that a new left bundle-branch block be considered as an electrocardiographic predictor of acute ischemia. In addition, they argued that patients who are stratified as low risk by the prediction rule could inappropriately include patients presenting with acute pulmonary edema, ongoing ischemic pain despite maximal medical therapy, or unstable angina after recent coronary revascularization (52). They insisted that such emergent clinical presentations be recommended for coronary care unit admission, not telemetry unit admission.*"

In other cases, however, operational constraints may depend on qualities that are difficult to define a priori. Consider for example, the following statement in Than et al. (2014), that describes the importance of sensibility for deployment:

"An important consideration during development is the clinical sensibility of the resulting prediction rule [...] Evaluation of sensibility requires judgment rather than statistical methods. A sensible rule is easy to use, and has content and face validities. Prediction rules are unlikely to be applied in practice if they are not considered sensible by the end-user, even if they are accurate."



Terminology	Meaning
(a)	
Sensibility	This refers to whether a prediction rule is both clinically reasonable and easy to use. This is more based on opinion than statistical methodology.
1. Content validity	For a rule to have content validity, the items included must be sensible, with no obvious omissions and the way that these variables are organised appears suitable for the objectives of the rule.
2. Face validity	This is a subjective interpretation of the validity of the rule by the user. The face validity of a rule will depend on the expectations and beliefs of the user, and may not be associated with statistical validity, but is essential for end-user uptake.
	To maximise face validity and ensure end-user trust, it may be necessary to include variables found to be statistically suboptimal in the final prediction model.
3. User friendliness	This refers to how easy the rule is to use. This depends on the demands the rule will place on memory, complexity of calculations in the absence of electronic devices, format and layout of the rule.

Figure 1.6: Table of qualitative requirements for the EDACS scoring system (see Box 1(a) in Than et al., 2014).

Techniques used in Model Development

Common techniques used in model development include:

- *Heuristic Feature Selection*: Many approaches use heuristic feature selection to reduce the number of variables in the model. Model development pipelines can often involve multiple rounds of feature selection, and may use different heuristics at each stage (e.g. Antman et al. 2000 uses a significance test to remove weak predictors, then uses approximate feature selection via forward stepwise regression).
- *Heuristic Rounding*: Many approaches use rounding heuristics to produce models with integer coefficients. In the simplest case, this involves scaling and rounding the coefficients from a logistic regression model (Goel et al., 2016) or a linear probability model (U.S. Department of Justice, 2005). The SAPS II score (Le Gall et al., 1993), for example, was built in this way ("*the general rule was to multiply the β for each range by 10 and round off to the nearest integer.*")
- *Empirical Risk Assessment*: In risk assessment applications, many approaches determine the coefficients of the model using logistic regression. Although these models are capable of generating a predicted risk estimates for each score through the logit function, the final model typically uses empirical risk estimates determined using an out-of-sample population (Six et al., 2008).
- *Expert Judgement*: A common approach to model development involves having a panel experts build a model by hand, and using data to validate the model after it is built (e.g. for the CHADS$_2$ score for stroke prediction of Gage et al. 2001, and

the National Early Warning Score to assess acute illness in the ICU of McGinley and Pearse 2012). Expert judgement can also be used in data-driven approaches to model development. In developing the EDACS score (Than et al., 2014), for example, the model was learned from data and expert judgement was used to: (i) determine a scaling factor for the coefficients ("*The beta coefficients were multiplied by eight, which was the smallest common multiplication factor possible to obtain a sensible score that used whole numbers and facilitated clinical ease of use.*") (ii) convert a continuous variable into a binary variables ("*Age was the only continuous variable to be included in the final score. It was converted to a categorical variable, using 5-year age bands with increasing increments of +2 points.*")

- *Unit Weighting*: This technique aims to produce a score by adding all variables that are significantly correlated with the outcome of interest. Unit weighting is prevalent in the criminal justice community (see e.g. Bobko et al., 2007; Duwe and Kim, 2016), where it is referred to as the *Burgess* method (as it was first proposed in Burgess, 1928). Unit weighting is often motivated by empirical work showing that linear models with unit weights may perform surprisingly well (see e.g. Einhorn and Hogarth, 1975; Dawes, 1979; Holte, 1993, 2006; Bobko et al., 2007).

Training Pipelines

Many scoring systems are built using complex *training pipelines* that combine traditional statistical techniques, heuristics, and expert judgement. The TIMI Risk Score of Antman et al. (2000), for example, was developed using the following training pipeline:

1. *"A total of 12 baseline characteristics arranged in a dichotomous fashion were screened as candidate predictor variables of risk of developing an end-point event"*
2. *"After each factor was tested independently in a univariate logistic regression model, those that achieved a significance level of P<.20 were [retained]."*
3. *"[The remaining factors]... selected for testing in a multivariate step-wise (backward elimination) logistic regression model. Variables associated with P < .05 were retained in the final model."*
4. *"After development of the multivariate model, the [risk predictions were determined]... for the test cohort using those variables that had been found to be statistically significant predictors of events in the multivariate analysis."*
5. *"The score was then constructed by a simple arithmetic sum of the number of variables present."*

Here, the training pipeline combines well-known statistical techniques. The pipeline is unlikely to produce in a scoring system that attains the best possible performance for several reasons, namely:

- Decisions involving feature selection and rounding are made sequentially (e.g. Steps 1-3 involve feature selection, Step 5 involves rounding).

- The objective function that is optimized at each step differs from the performance metric of the final model (i.e. the calibration error, which measures the reliability of risk estimates).
- Some steps optimize conflicting objective functions (e.g. backward elimination optimizes the error rate, while the final model is fit to optimize the logistic loss).
- Some steps do not fully optimize their own objective function (i.e. backward elimination does not return a globally optimal feature set).
- Some steps depend free parameters that are set without validation (e.g. the threshold significance level to keep each feature).
- The final model was not fit with all of the training data. Here, Steps 4 and 5 use data from a test cohort that may have been useful in improving the fit of the model.
- The final model uses an empirical risk estimate for each score (i.e. the predicted risk for each score simply represents the % of patients in the test cohort with $y_i = +1$).
- The final model uses unit weights.

Reasons for Ad Hoc Development

There are several potential reasons to explain the prevalence of ad hoc approaches in applied domains, and the lack of concern surrounding performance guarantees.

- It is difficult to design methods that can address the wide range of constraints that are required for each application.
- There are few empirical studies that benchmark the performance effects of different approaches (e.g. preliminary feature selection via significance testing vs. backward stepwise regression). It is difficult to design a study given the number of distinct approaches that are used in practice. It is also unlikely that such a study will produce impactful findings given that many approaches incorporate domain expertise (e.g. for feature selection, or rounding).
- Models are primarily developed by researchers in applied domains. In light of this, the methods used to build the model only represent a single stage of the overall model development process. Other important aspects of model development include data collection, prescribing treatment options for each score, and validating the performance of the model on a new population.
- Models are primarily developed for tasks that are traditionally performed by humans (e.g. diagnosing illness, approving loans, granting parole). In these applications, any model may be acceptable so long as it outperforms random guessing (i.e. if the task is currently performed exclusively by humans), or outperforms an existing model.

Consequences of Ad Hoc Development

The lack of a standard methodology to build scoring systems has several important consequences, namely:

- *Suboptimal Performance*: Models can perform poorly compared to models built using modern data-driven methods (see e.g., the performance of CHADS$_2$ in Letham et al. 2015 and "Burgess" scoring systems in Duwe and Kim 2016).
- *No Feasibility Guarantee*: Ad hoc approaches cannot guarantee that they will produce a feasible model in new settings (e.g. under additional operational constraints, on a different dataset, in a different application).
- *No Performance Guarantee*: Ad hoc approaches cannot guarantee that they will produce the best possible model. In an application such as sentencing in the criminal justice, a model must be easy to use but also perform well. In light of the trade-offs between accuracy and usability, practitioners need to provide extensive additional analysis to show that their proposed model does not sacrifice too much performance in order to be usable (see e.g. validation studies for the sentencing tool in Figure 1.1, described in Pennsylvania Commission on Sentencing 2012).
- *No Performance Objective*: The lack of a formally defined problem affects the development of both models and methods. Without a clear performance objective in place, scoring systems are often built using the wrong techniques and evaluated using the wrong metrics. In criminal justice, for instance, tools that are used for decision-making are built using methods for risk assessment (e.g. logistic regression) and evaluated using metrics for ranking (e.g. AUC). In medicine, for example, many physicians are unaware of the correct performance metric for a risk assessment tool (e.g. calibration error), which leads to the proliferation of techniques that affect risk calibration (e.g. scaling before rounding).

1.1.3 Related Work in Machine Learning

Sparse Linear Classification

Although many scoring systems are built using traditional classification methods, modern methods for sparse linear classification are ill-suited to build them because: (i) they need to be paired with a rounding heuristic to produce models with integer coefficients; (ii) they may not include built-in controls to address operational constraints; and (iii) they optimize surrogate measures to ensure scalability.

The use of surrogate measures merits further discussion given the prevalence of methods that optimize surrogate measures. The methods in this dissertation do not optimize surrogate measures for two reasons:

1. Methods that optimize surrogate measures (e.g the hinge loss and the ℓ_1-penalty) do not fit models that attain the best possible trade-off between accuracy and sparsity. Surrogate loss functions, for example, are not robust to outliers (Wu and Liu, 2007; Long and Servedio, 2010). Similarly, ℓ_1-penalization is only guaranteed to recover the correct sparse solution (i.e. the one that minimizes the ℓ_0-norm) under restrictive conditions that may not be satisfied in practice (see Lin et al., 2008, for a discussion).

   The empirical results in this work suggest that the loss in performance due to convex surrogates is substantial in constrained settings (e.g. when fitting highly sparse models with real-valued coefficients, or when these methods are paired with rounding heuristics to produce integer coefficients). This may be true in other settings, as recent methods have aimed to replace surrogate with exact measures, such as methods that optimize the 0-1 loss for accuracy (see e.g. Brooks, 2011; Nguyen and Sanner, 2013), and penalize the ℓ_0 for exact feature selection (see e.g. Neumann et al., 2005; Liu and Wu, 2007).

2. Methods that optimize surrogate measures typically introduce free parameters. In practice, this means that users have to tune the free parameters, and use nested cross-validation (CV) to obtain an unbiased estimate of model performance. In a standard nested CV setup, methods have to be run multiple times (e.g. $110P$ times for a nested CV setup with 10 outer folds and 10 inner folds over a free parameter grid with P total instances). The time required to complete the training process can be substantial if each run does not terminate quickly (e.g. for methods such as ℓ_0-penalized SVM that use optimize both surrogate measures and exact measures). In this case, it is faster to use an approach that does not require parameter tuning.

Linear Classification with Integer Coefficients

There have been some methods proposed for the purpose of building simple linear models with integer coefficients, namely:

- Chevaleyre et al. (2013) aim to fit models for decision-making by solving an integer programming problem that optimizes the hinge loss and rounding coefficients to +1 or -1.
- Carrizosa et al. (2016) aim to fit models for decision-making by solving an integer programming problem that optimizes the hinge loss and restricts coefficients to small values.
- Ertekin and Rudin (2015) propose a Bayesian approach to produce models for risk-assessment. This approach has the benefit in that it produces a posterior distribution on the coefficients.
- Jung et al. (2017) propose a method that combines stepwise forward feature selection, logistic regression, scaling, and rounding. The resulting method produces risk assessment tools with high AUC, but poor risk calibration (see e.g. Chapter 6.4).

The optimization techniques in this dissertation can be used to solve the same optimization problems as the ones considered by these methods. Further, they can also be applied to learn certain classes of boolean classification models (see Chapter 2), such as AND-of-OR models for decision-making (Malioutov and Varshney, 2013) and risk assessment (Wang et al., 2015). In contrast to these methods, the approach in this dissertation can recover a globally optimal solution, provide a certificate of optimality, address operational constraints, and scale seamlessly with the number of samples in the training data when the problems use a convex loss function.

Although there has not been much theoretical work that has explicitly focused on the problem of learning sparse linear classifiers with small integer coefficients, there are two noteworthy results with practical value. In settings where the training data contains only binary variables, the decision-making models are effectively *threshold gates* (Muroga, 1971; Long and Servedio, 2014). We can make use of results in this area to understand regularization due to small integer coefficients, and establish connections between discrete linear classification models and other kinds of rule-based models. We discuss these connections in greater detail in Chapter 2.

Interpretable Machine Learning

Scoring systems are examples of interpretable predictive models. Such models are becoming increasingly important for applications where predictive models output decisions that significantly affect humans, as evidenced by several press surrounding the use of predictive models in healthcare (Mukherjee, 2017) and criminal justice (Starr, 2014; Barry-Jester et al., 2015; Tashea, 2017; Angwin et al., 2016). In some applications, interpretable predictive models may be required by law. The Fair Credit Reporting Act, for instance, requires an explanation when consumers are denied access to credit in the United States. Similarly, upcoming EU regulations that require a "right to an explanation" from algorithmic decision-making tools that affect humans (Goodman and Flaxman, 2016).

Proposed approaches to handle interpretability in machine learning include:

- *Sparsity*: In statistics, sparsity refers to the number of terms in a model and constitutes a standard measure of model complexity (Sommer, 1996; Rüping, 2006). Several approaches aim to handle the interpretability through sparsity, as measured by the number of coefficients in a linear model, the number of nodes in a decision tree, and the number of rules in a rule set (Tibshirani, 1996; Zou and Hastie, 2005; Efron et al., 2004; Hesterberg et al., 2008; Quinlan, 1999; Breslow and Aha, 1997; Kohavi and John, 1997; Guyon and Elisseeff, 2003).
- *Monotonicity*: Several methods have focused on learning models that enforce monotonic relationships between certain input variables and the predicted outcome (Pazzani et al., 2001; Verbeke et al., 2011; Martens et al., 2011; Gupta et al., 2016).
- *Transparency*: A large body of work has aimed to develop *transparent* models, which provide a textual or visual representation of the relationship between input variables and the predicted outcome (as opposed to *black-box* models). Examples of transparent classification models include linear models, decision trees (Quinlan, 1986; Utgoff, 1989; Quinlan, 2014), decision tables (Kohavi, 1995), rule lists (Rivest, 1987; Letham et al., 2013), and rule sets (Lakkaraju et al., 2016).
- *Diagnostics*: These techniques extract rules and prototype examples to illustrate the relationship between input variables and the predicted outcome (Meinshausen et al., 2010; Van Assche and Blockeel, 2007; Fung et al., 2005; Martens et al., 2007; Bien and Tibshirani, 2011; Kim et al., 2015). Many diagnostics are model-agnostic, meaning that they can be used to troubleshoot and generate insights from predictive model. However, they are unable to address issues that they may reveal.

The approach in this dissertation can fit simple transparent models that are sparse, that obey monotonicity constraints, and that do not require additional diagnostics to extract rules.

Handling Interpretability and Qualitative Constraints

The challenges of building interpretable models has been frequently discussed up in the machine learning community for over two decades (see e.g., Pazzani, 2000; Kodratoff, 1994; Freitas, 2014; Doshi-Velez and Kim, 2017). This discussion highlights two major issues that make it difficult to handle interpretability from a methodological perspective:

1. *How to define interpretability?* Pazzani (2000), for example, writes: "[There is a].. conflicting... [and unfounded].. set of claims in the literature as to which [type of model]... is easiest to understand." User studies on interpretability, for instance, often conclude that different predictive models are "most" interpretable in different domains for different reasons (Subramanian et al., 1992; Kohavi and Sommerfield, 1998; Allahyari and Lavesson, 2011; Huysmans et al., 2011). The notion is subjective, domain-dependent, and multifaceted (Kodratoff, 1994; Pazzani, 2000; Freitas, 2014). Models that are interpretable to one audience may not be interpretable for a different audience due to differences in their affinity for certain types of knowledge representation, their exposure to the data, and their domain expertise (see Kodratoff, 1994; Rüping, 2006; Freitas, 2014).

2. *How to strike a balance between performance and interpretability?* It is well-known that there is a trade-off between performance and interpretability. However, the precise trade-off depends on how we define interpretability, whether our method returns a pareto-optimal model, and the difficulty of the learning problem. In practice, this means that the trade-off changes significantly across each application and should be re-evaluated each time. A related problem for practitioners is that the target audience does not wish to explicitly define exact requirements for interpretability, or know how much performance they are willing to sacrifice for interpretability (e.g., Kodratoff 1994, mentions that interpretability is an "ill-defined" concept, which is echoed by Doshi-Velez and Kim 2017). In this case, the audience would rather have the flexibility to choose a model from a set of models that do not violate interpretability related constraints.

The discrete optimization approach in this work provides a practical alternative to address these issues. Instead of adopting a particular definition of interpretability, it provides users with the ability to incorporate their exact requirements in the training process, and recovers a model that optimizes performance under these constraints. Since we fit models from a constrained hypothesis space, these models generalize, so the performance on training data is reflective of the performance on test data. In this way, users can evaluate the impact of their requirements on predictive performance, and navigate these trade-offs in an informed manner.

This approach provides a promising alternative to handle other qualitative constraints, such as *safety* (Amodei et al., 2016) and *fairness* (Kamishima et al., 2011; Dwork et al., 2012; Zafar et al., 2015). Such constraints are similar to interpretability in that they depend on multiple model qualities, change across applications, and impose unclear performance trade-offs. As a result, building models that address these constraints will require an approach that offers the flexibility to deal with different definitions, and optimality guarantees that allow us to make informed comparisons between different models.

1.1.4 Related Work in Optimization

The discrete optimization approach in this dissertation is broadly related to the application of mixed-integer programming in supervised learning. In particular, our work is related to linear classification methods that solve mixed-integer programming problems (see the survey papers of Rubin, 2009; Lee and Wu, 2009; Fan and Chaovalitwongse, 2009).

Early work in this area was primarily focused on the *misclassification minimization* problem, which aims to fit a linear model that optimizes the 0–1 loss (Rubin, 1990; Mangasarian, 1994; Asparoukhov and Stam, 1997). The first attempts at misclassification minimization were able to fit models on small datasets with at most $n = 200$ examples (Joachimsthaler and Stam, 1990; Erenguc and Koehler, 1990). Subsequent work was able to recover solutions for datasets with around $n = 500$ by developing specialized algorithms and heuristics (Soltysik and Yarnold, 1994; Yanev and Balev, 1999; Rubin, 1997; Asparouhov and Rubin, 2004). The size of the datasets used in these early applications may explain why MIP-based approaches have not been widely used by the machine learning community.

Over the last three decades, commercial MIP solvers have been able to solve increasingly large discrete optimization problems (Bixby et al., 2004; Bixby and Rothberg, 2007). This is due to improvements in computer hardware and implementations that incorporate new techniques to improve branch and bound. Accordingly, recent work has been able to solve MIPs to fit classification models for much larger datasets (see e.g. Brooks, 2011; Nguyen and Sanner, 2013, for the misclassification minimization problem), and that handle additional constraints (e.g. feature selection Glen, 1999; Liu and Wu, 2007; Goldberg and Eckstein, 2012; Guan et al., 2009; Nguyen and Franke, 2012; Bertsimas et al., 2015; Sato et al., 2015, 2016).

The results from this stream of work show that MIP-based methods can now be used to fit classification models that perform well on real-world datasets. These models typically correspond to feasible solutions with large optimality gaps (e.g. Brooks, 2011; Carrizosa et al., 2016; Bertsimas et al., 2015). In other words, they do not necessarily reflect the performance of the best models that can be obtained using an exact approach. In light of this, the fact that feasible models outperform state-of-the-art methods suggests that "an approximate solution to the exact problem may outperform an exact solution to the approximate problem" (Lin et al., 2008).

The discrete optimization approach in this dissertation differs from other MIP-based methods in that it aims to not only improve performance, but address operational constraints, and recover a certificate of optimality. Our results show that certifiably optimal solutions to our problems correspond to models that not only have better performance but that also have various other benefits in real-world applications.

1.2 Organization

The remainder of this dissertation is organized as follows.

In Chapter 2, we introduce basic concepts, notation, and themes used in this dissertation. We formally define an optimization problem to learn discrete linear classification models from data. We describe the kinds of models that can be built using this framework and list the various operational constraints that they can address. We discuss key properties of discrete linear classification models, such as generalization, regularization, and complexity. We then describe how the optimization problems are solved by modern MIP solvers, and how we can improve their ability to recover certifiably optimal solutions in our setting.

In Chapter 3, we consider the problem of learning scoring systems for decision-making. We formulate a discrete optimization problem to fit scoring systems that fully optimized for accuracy, sparsity, and small integer coefficients. Our scoring system problem is an integer program (IP), which minimizes the 0-1 loss for accuracy, penalizes the ℓ_0-norm for sparsity, and restricts coefficients to small coprime integers. We refer to the optimal solution to this problem as a *Supersparse Linear Integer Model* (SLIM). We present tight designed IP formulations for SLIM and techniques to improve the ability of commercial MIP solvers to recover a certifiably optimal solution. In addition, we present new theoretical results for scoring systems, including generalization bounds to motivate why these simple models perform well, and discretization bounds that can be used to control the regularization due to small integer coefficients. We end with an extensive set of numerical experiments to benchmark the accuracy and sparsity of SLIM scoring systems against popular classification methods. These results show that our approach can learn models that are accurate and sparse in a matter of minutes.

We illustrate the benefits of our approach through two real-world applications. In Chapter 4, we use SLIM to create a scoring system for sleep apnea screening in collaboration with the Massachusetts General Hospital Sleep Clinic. The results in this chapter show our approach can address operational constraints related to accuracy and model form without parameter tuning, and highlight the performance benefits in solving an exact problem under such constraints. We discuss the importance of our approach from a clinical standpoint, as our model can screen for sleep apnea using information that can be extracted from electronic health records, without the need for patient-reported symptoms. In Chapter 5, we use SLIM to develop decision-making tools for a collection of recidivism prediction problems. Our work addresses several important questions that have been raised by the criminal justice community regarding trade-offs between accuracy, transparency, and interpretability. We show that there may exist simple models for common recidivism prediction problems, and discuss the importance of producing models that can be validated in this domain.

In Chapter 6, we turn to the the problem of learning scoring systems for risk assessment, which we refer to as *risk scores*. We consider a discrete optimization problem to fit risk scores that have good risk calibration and rank accuracy, and are optimized for feature selection and small integer coefficients. The risk score problem is a mixed-integer non-linear problem (MINLP), which minimizes the logistic loss, penalizes the ℓ_0-norm, and restricts coefficients to small integers. We refer to the optimal solution to this problem as a *Risk-Calibrated Supersparse Linear Integer Model* (RiskSLIM). We show that solving the risk score problem using commercial MINLP solvers can be time-consuming even on small instances. Accordingly, we present a new cutting plane algorithm to recover the optimal solution to the risk score problem in a way that scales linearly in the number of samples and can be implemented using a MIP solver. We further improve our algorithm with specialized techniques to generate feasible solutions, narrow the optimality gap, and reduce data-related computation. We illustrate the benefits of this approach through an extensive set of numerical experiments, in which we compare the performance of RiskSLIM models to those built using advanced heuristics.

We use RiskSLIM to create risk assessment tools for two real-world problems. In Chapter 7, we create a customized risk score for ICU seizure prediction. In Chapter 8, we create a risk score for ADHD diagnosis from a self-reported questionnaire. Both applications show how our approach can produce highly usable models by formulating operational constraints.

PREDICT PATIENT HAS OBSTRUCTIVE SLEEP APNEA IF SCORE > 1

1.	*Age $\geq$ 60*	4 points	$\cdots$
2.	*Hypertension*	4 points	$+ \cdots$
3.	*BMI $\geq$ 30*	2 points	$+ \cdots$
4.	*BMI $\geq$ 40*	2 points	$+ \cdots$
5.	*Female*	-6 points	$+ \cdots$
	ADD POINTS FROM ROWS 1 – 5	**SCORE**	$= \cdots$

Figure 1.7: SLIM scoring system for sleep apnea screening, developed in Chapter 4. This model achieves a 10-CV mean test TPR/FPR of 61.4/20.9%.

PREDICT ARREST FOR ANY OFFENSE IF SCORE > 1

1.	*Age at Release 18 to 24*	2 points	$\cdots$
2.	*Prior Arrests $\geq$ 5*	2 points	$+ \cdots$
3.	*Prior Arrest for Misdemeanor*	1 point	$+ \cdots$
4.	*No Prior Arrests*	-1 point	$+ \cdots$
5.	*Age at Release $\geq$ 40*	-1 point	$+ \cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$= \cdots$

Figure 1.8: SLIM scoring system for recidivism prediction, developed in Chapter 5. This model has a test TPR/FPR of 76.6%/44.5%.

1.	*AnyBriefRhythmicDischarge*	2 points	···
2.	*PatternsInclude LPD*	2 points	+ ···
3.	*AnyPriorSeizure*	1 point	+ ···
4.	*Epiletiform or Discharge*	1 point	+ ···
	ADD POINTS FROM ROWS 1–4	**SCORE**	= ···

SCORE	0	1	2	3	4	5	6
SEIZURE RISK	4.7%	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

Figure 1.9: RISKSLIM risk score for ICU seizure prediction, developed in Chapter 7. This model has a 5-CV mean test CAL/AUC of 2.5%/0.801.

	Never	Rarely	Sometimes	Often	Very Often
How often do you have trouble concentrating on what people say to you even when they speak to you directly?	0	4	4	5	5
How often do you leave your seat in meetings or other situations in which you are expected to remain seated?	0	0	1	1	5
How often do you have difficulty unwinding and relaxing when you have time to yourself?	0	4	4	6	6
How often do you find yourself finishing the sentences of the people you talk to before they can finish them themselves?	0	0	2	2	2
How often do you put things off until the last minute?	0	2	2	4	4
How often do you depend on others to keep your life in order and attend to details?	0	2	3	3	3

SCORE	≤13	14	15	16	17	18	≥19
ADHD RISK	<5.0%	11.9%	26.9%	50.0%	73.1%	88.1%	>95.0%

Figure 1.10: RISKSLIM risk score for DSM-5 adult ADHD, developed in Chapter 8. This model has a 10-CV CAL/AUC of 1.5%/0.978.

1.3 Contributions

This dissertation contains material from the following publications.

1. B. Ustun and C. Rudin. Supersparse Linear Integer Models for Optimized Medical Scoring Systems. Machine Learning, 102(3):349–391, 2016b
2. B. Ustun, S. Traca, and C. Rudin. Supersparse Linear Integer Models for Predictive Scoring Systems. In AAAI Late-Breaking Developments, 2013
3. B. Ustun, M. Westover, C. Rudin, and M. T. Bianchi. Clinical Prediction Models for Sleep Apnea: The Importance of Medical History over Symptoms. Journal of Clinical Sleep Medicine, 12(2):161–168, 2016
4. J. Zeng, B. Ustun, and C. Rudin. Interpretable Classification Models for Recidivism Prediction. Journal of the Royal Statistical Society: Series A, 2016
5. B. Ustun and C. Rudin. Learning Optimized Risk Scores for Large-Scale Datasets. arXiv:1610.00168, 2016a
6. B. Ustun and C. Rudin. Optimized Risk Scores. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017
7. A. F. Struck, B. Ustun, A. Rodriguez Ruiz, J. W. Lee, S. LaRoche, L. J. Hirsch, E. J. Gilmore, C. Rudin, and B. M. Westover. A Practical Risk Score for EEG Seizures in Hospitalized Patients. Forthcoming in JAMA Neurology, 2017
8. B. Ustun, L. A. Adler, C. Rudin, S. V. Faraone, T. J. Spencer, P. Berglund, M. J. Gruber, and R. C. Kessler. The World Health Organization Adult Attention-Deficit / Hyperactivity Disorder Self-Report Screening Scale for DSM-5. JAMA Psychiatry, 74(5):520–526, 2017

Software

Optimized Scoring Systems.
http://github.com/ustunb/slim-python
http://github.com/ustunb/slim-matlab

Optimized Risk Scores.
http://github.com/ustunb/risk-slim

Binary Classification Pipeline.
https://github.com/ustunb/classification-pipeline

Chapter 2

Preliminaries

In this chapter, we introduce several important concepts related to scoring systems. We define a general optimization problem that we solve to learn scoring systems from data. Next, we present results related to the performance, generalization, and interpretation of scoring systems. Lastly, we discuss how to solve the discrete optimization problems using a modern mixed integer programming approach.

2.1 Optimization Problem

We start with a dataset of n i.i.d. training examples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)_{i=1}^n\}$ where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^{d+1}$ denotes a vector of features $[1, x_{i,1}, \ldots, x_{i,d}]^\top$ and $y_i \in \mathcal{Y} = \{-1, 1\}$ denotes a class label. We consider linear classification models of the form

$$\hat{y} = \text{sign}(\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle),$$

where $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_d]^\top$ represents a vector of coefficients and λ_0 represents an intercept. We learn the coefficients by solving an optimization problem of the form:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & l(\boldsymbol{\lambda}) + C \cdot \Phi(\boldsymbol{\lambda}) \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{2.1}$$

Here:

- $l(\boldsymbol{\lambda}) : \mathbb{R}^{d+1} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathbb{R}_+$ is a *loss function* that controls the overall fit of the model;
- $\Phi(\boldsymbol{\lambda}) : \mathbb{R}^{d+1} \to \mathbb{R}_+$ is a *penalty function* that induce qualities that are desirable but may be sacrificed for greater accuracy;
- $\mathcal{L} \subset \mathbb{Z}^{d+1}$ is a finite discrete *coefficient set* that encodes hard qualities that must be satisfied;
- $C > 0$ is a *trade-off parameter* that controls the balance between accuracy and soft qualities.

2.1.1 Notation, Terminology and Assumptions

We provide a list of all notation used in later chapters in Table 2.1.

We denote the objective function of the optimization problem in (2.1) as

$$V(\boldsymbol{\lambda}) = l(\boldsymbol{\lambda}) + C \cdot \Phi(\boldsymbol{\lambda}),$$

and an optimal solution as

$$\boldsymbol{\lambda}^* \in \operatorname*{argmin}_{\boldsymbol{\lambda} \in \mathcal{L}} V(\boldsymbol{\lambda}).$$

We bound the optimal value as $V(\boldsymbol{\lambda}^*) \in [V^{\min}, V^{\max}]$. The *optimality gap* refers to the quantity

$$\varepsilon = 1 - \frac{V^{\min}}{V^{\max}}.$$

In practice, the upper bound $V^{\max} = V(\boldsymbol{\lambda}^{\text{best}})$ is set as the objective value of the best feasible solution $\boldsymbol{\lambda}^{\text{best}} \in \mathcal{L}$, and $V^{\min}$ is set using B&B algorithm (see Algorithm 1). Solving the problem to *optimality* means that we have recovered a solution with an optimality gap of $\varepsilon = 0.0\%$. This implies that we have found the best integer feasible solution to (2.1) and paired with a lower bound $V^{\min} = V$.

For clarity of exposition, we make the following assumptions about the training dataset $\mathcal{D}_n$:

(A1) $n^+, n^- \geq 1$ (at least one example from each class);

(A2) for all $i \in I, x_{i,j} \neq 0$ for some $j = 1, \ldots, d$ (no null examples);

(A3) for all $j \in \{1, \ldots, d\}, x_{i,j} \neq 0$ for some $i \in I$ (no null features);

(A4) for all $j, k \in \{1, \ldots, d\}, x_{i,j} \neq x_{i,k}$ for some $i \in I$ (no duplicate features).

2.1.2 Model Classes and Operational Constraints

The optimization problem in (2.1) generalizes a rich class of models based on the choice of loss function, coefficient set, and training data. We provide a list of models in Table 2.2. In this dissertation, we focus on scoring systems (Chapter 3) and risk scores (Chapter 6). The techniques discussed in these chapters are broadly applicable to any problem with the form of (2.1) so long as: (i) the loss function is discrete or convex; (iii) the penalty function is discrete or linear; (iii) the coefficient set is discrete and bounded.

The choice of a loss function and penalty function is an important design decision. In this dissertation, we consider problems that avoid the use of surrogate measures. In the scoring system problem, for example, the objective optimizes the 0-1 loss $l_{01}(\boldsymbol{\lambda})$ and penalizes the ℓ_0-norm. In the risk score problem, the objective optimizes the logistic loss $l_{\log}(\boldsymbol{\lambda})$ and penalizes the ℓ_0-norm.

Given that exact measures result in difficult optimization problems, there are several reasons to use them: (i) optimizing exact measures produce models that

Data	
n	number of examples
d	number of dimensions
$\mathcal{D}_n$	training dataset with n examples
I	index set of all examples
$\mathcal{X}$	feature space
$\mathcal{Y}$	label space
$\boldsymbol{x}_i = (1, x_{i,1}, \ldots, x_{i,d})$	feature vector for example i
y_i	label for example i
n_i	number of examples with features $\boldsymbol{x} = \boldsymbol{x}_i$
I^+, I^-	index set for positive/negative examples
n^+, n^-	number of positive/negative examples
Model	
$\hat{y}_i$	predicted label for example i
$\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \ldots, \lambda_d)$	coefficient vector
λ_0	intercept term
$\mathcal{L}$	set of coefficient vectors
$\mathcal{L}_j$	feasible coefficients for feature j
$\Lambda_j^{\text{min}}, \Lambda_j^{\text{max}}$	min/max coefficient for feature j
$\Lambda^{\text{min}}, \Lambda^{\text{max}}$	min/max coefficient for any feature j
$\mathcal{L}^{\text{trivial}}$	set of coefficient vectors for trivial models
Optimization Problem	
C_0	ℓ_0 penalty parameter
w^+, w^-	misclassification cost for a positive/negative example
$\boldsymbol{\lambda}^*$	optimal set of coefficients
ε	optimality gap
$V(\cdot)$	objective value function
$l(\cdot)$	loss function
$l_{01}(\cdot)$	0–1 loss function
$l_{\log}(\cdot)$	logistic loss function
$[V^{\text{min}}, V^{\text{max}}]$	bounds on optimal objective value $V(\boldsymbol{\lambda}^*)$
$[R^{\text{min}}, R^{\text{max}}]$	bounds on optimal number of non-zero coefficients $\|\boldsymbol{\lambda}^*\|_0$
$[L^{\text{min}}, L^{\text{max}}]$	bounds on optimal value of loss function $l(\boldsymbol{\lambda}^*)$

Table 2.1: Notation.

attain Pareto-optimal trade-offs on the training data; (ii) problems that use exact measures have trade-off parameters that can be set a priori; (iii) problems that use exact measures can directly incorporate operational constraints that involve these quantities in their feasible region. We provide further motivation for these choices in Chapter 3 and Chapter 6, and illustrate their benefits through applications in Chapters 4, 5, 7, and 8.

Model Class	**Optimization Problem**	
Scoring System	$\min_{\boldsymbol{\lambda}} \; l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$	
	s.t. $\boldsymbol{\lambda} \in \mathcal{L}$	
Risk Score	$\min_{\boldsymbol{\lambda}} \; l_{\log}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$	
	s.t. $\boldsymbol{\lambda} \in \mathcal{L}$	
Rule[†] Set	$\min_{\boldsymbol{\lambda}} \; l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$	
	s.t. $\boldsymbol{\lambda} \in \{0,1\}^{d+1}$	
Predictive[†] Checklist	$\min_{\boldsymbol{\lambda}} \; l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$	
	s.t. $\lambda_0 \in \{1,\ldots,d\}$	
	$\lambda_j \in \{0,1\}$	$j = 1,\ldots,d$
Decision[†] List	$\min_{\boldsymbol{\lambda}} \; l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$	
	s.t. $\lambda_j = \sum_{k=1}^{d} 2^k u_{jk}$	$j = 1,\ldots,d$
	$\sum_{k=1}^{d} u_{jk} = 1$	$j = 1,\ldots,d$
	$u_{jk} = \{0,1\}$	$j,k = 1,\ldots,d$

Table 2.2: Types of predictive models that can be represented as linear classifiers with finite integer coefficients. Here we have used the penalty function $\Phi(\boldsymbol{\lambda}) = \|\boldsymbol{\lambda}\|_0$. All of these models can be built by solving optimization problem in (2.1). Model classes marked with [†] require training data that only uses binary input variables.

Model Requirement	Example
Feature Selection	Choose up to 5 features
Group Sparsity	Include either *Male* or *Female*, not both
Optimal Thresholding	Use at most 3 thresholds for *Age*: $\sum_{k=1}^{100} \mathbb{1}\left[Age \leq k\right] \leq 3$
Hierarchical Structure	If *Male* is in model, then include *Hypertension*
Monotonicity	Ensure that the coefficient for *Male* is positive
Side Information	Predict $\hat{y} = +1$ when $Male = $ TRUE and $Hypertension = $ TRUE
Fairness	Limit disparate impact between $i \in A, B$ to 80%: $\frac{\Pr(\hat{y}=+1 \mid i \in A)}{\Pr(\hat{y}=+1 \mid i \in B)} \leq 0.8$
Accuracy	Restrict FPR to 20%

Table 2.3: Operational constraints that can be added to the feasible region of the optimization problem (2.1).

2.1.3 Connections with Linear Threshold Gates

In problems where the training data contains only binary input variables, a linear classification model with finite integer coefficients is a Boolean function. As an example, consider the linear model,

$$\hat{y} = \text{sign}\left(-1 + 2Male - 2Hypertension\right),$$

which can be viewed as the Boolean function,

$$\hat{y} = Male \wedge \neg Hypertension.$$

The relationship between linear classification models and Boolean functions has been studied in early work in circuit design (Muroga, 1971) as well as more recent work in computational learning theory (see e.g., Håstad, 1994; Servedio, 2007). Formally stated, every linear classifier with integer coefficients can be represented as a Boolean function. However, not every Boolean function can be represented as a linear classification model with integer coefficients. The class of Boolean functions that can be represented as linear classification models are called *linear threshold gates.* The theoretical results pertaining to linear threshold gates have several practical benefits in our setting:

- We can use bounds on the size of the largest coefficient for a threshold gate to determine a coefficient set that will guarantee no loss in training accuracy due to the use of finite integer coefficients. In Theorem 2.1, for example, we use a bound on the largest coefficient of a threshold gate from Schmitt (2012) to determine a coefficient set $\mathcal{L}$ that is large enough to guarantee that we do not lose training accuracy due to the use of finite integer coefficients.
- We can extract a Boolean function from any linear classification model with integer coefficients, which can help users understand the interactions between multiple

variables. In practice, users can extract the Boolean function through inspection (see e.g. the sleep apnea model in Chapter 4). Alternatively, the Boolean function can also be extracted by means of an auxiliary technique (see e.g. Fiat and Pechyony, 2004, for an algorithm to extract a minimal height decision tree).

Theorem 2.1 (Coefficient Set for No Loss in Training Accuracy)
Given a training dataset $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$ with binary features $\boldsymbol{x}_i \in \{0,1\}^{d+1}$, let $\boldsymbol{\rho} \in \mathbb{R}^{d+1}$ denote the coefficients of a standard linear classifier.

If the coefficients of a discrete linear classifier are restricted to a finite set of integer values $\boldsymbol{\lambda} \in \mathcal{L}$ such that:

$$\mathcal{L} = \left\{ \boldsymbol{\lambda} \in \mathbb{Z}^{d+1} \;\middle|\; \begin{array}{l} |\lambda_0| \leq \frac{1}{2^{d+1}} d(d+1)^{\frac{d+3}{2}} + \frac{1}{2} \\ |\lambda_j| \leq \frac{1}{2^d}(d+1)^{\frac{d+1}{2}} \text{ for } j = 1, \ldots, d \end{array} \right\}, \tag{2.2}$$

then, the most accurate classifier with finite integer coefficients will attain the same training accuracy as the most accurate classifier with real-valued coefficients:

$$\min_{\boldsymbol{\lambda} \in \mathcal{L}} l_{01}(\boldsymbol{\lambda}) = \min_{\boldsymbol{\rho} \in \mathbb{R}^{d+1}} l_{01}(\boldsymbol{\rho}),$$

That is, there will be no loss in training accuracy in the optimal classifier due to the use of finite integer coefficients.

2.1.4 Generalization

One of the key properties of scoring systems is that they generalize well. This observation is supported by the empirical results in this dissertation, and often mentioned in the literature from applied domains (see e.g. work on the out-of-sample performance of linear models with unit weights in Einhorn and Hogarth, 1975; Wainer, 1976; Dawes, 1979; Bobko et al., 2007).

We can motivate the generalization of these models from a machine learning perspective using ideas from structural risk minimization (Vapnik, 1998). Consider fitting a classifier $f : \mathcal{X} \to \mathcal{Y}$ with data $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$, where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_i \in \mathcal{Y} = \{-1, 1\}$. In Theorem 2.2, we present a well-known uniform generalization guarantee on the predictive accuracy of all functions, $f \in \mathcal{F}$. This guarantee bounds the *true risk*,

$$R^{\text{true}}(f) = \mathbb{E}_{\mathcal{X},\mathcal{Y}} \mathbb{1}\left[f(\boldsymbol{x}) \neq y\right],$$

by the *empirical risk*,

$$R^{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left[f(\boldsymbol{x}_i) \neq y_i\right],$$

and other quantities important to the learning process.

Theorem 2.2 (Generalization of Discrete Linear Classifiers)
Let $\mathcal{F}$ denote the set of linear classifiers with coefficients $\boldsymbol{\lambda} \in \mathcal{L}$:

$$\mathcal{F} = \left\{ f : \mathcal{X} \to \mathcal{Y} \mid f(\boldsymbol{x}) = \operatorname{sign}\left(\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle\right) \text{ and } \boldsymbol{\lambda} \in \mathcal{L} \right\}.$$

For every $\delta > 0$, with probability at least $1 - \delta$, every classifier $f \in \mathcal{F}$ obeys:

$$R^{\text{true}}(f) \leq R^{\text{emp}}(f) + \sqrt{\frac{\log(|\mathcal{L}|) - \log(\delta)}{2n}}.$$

A proof of Theorem 2.2 can be found in Section 3.4 of Bousquet et al. (2004). The result shows that more restrictive hypothesis spaces can lead to better generalization. The bound is vacuous for most real-world problems unless the sample size n is sufficiently large or the coefficient set $\mathcal{L}$ is sufficiently small. However, it provides some motivation for using sparse linear models with integer coefficients without necessarily expecting a loss in predictive accuracy.

We further improve the generalization bound in Theorem 2.2 for scoring systems in Chapter 3.5, and use them to design a subsampling procedure to reduce data-related computation in Chapter 6.3.3.

2.2 Mixed Integer Programming

We aim to recover certifiably optimal scoring systems using a *mixed integer programming* approach (see Wolsey, 1998, for an introduction). More precisely, given a discrete optimization problem with the form of (2.1), we first formulate the problems as an *integer program* (IP) or *mixed integer program* (MIP). We then solve these formulations with a MIP solver (see Table 2.4). In what follows, we briefly describe the branch and bound search process used by MIP solvers, then discuss how we aim to improve this process for our setting.

MIP Solver	Reference
CBC	Forrest and Ralphs (2017)
CPLEX	ILOG (2017)
SCIP	Gamrath et al. (2016)
Symphony	Ralphs et al. (2017)
Xpress	FICO (2017)

Table 2.4: A list of commercial and open-source MIP solvers that can solve the optimization problems in this dissertation. Annual performance benchmarks of several MIP solvers can be found in http://plato.asu.edu/ftp/milpc.html.

2.2.1 Branch and Bound

MIP solvers recover a certifiably optimal solution to discrete optimization problems through *branch and bound* (B&B) search (Land and Doig, 1960). In what follows, we describe the main components of B&B using Algorithm 1.

Algorithm 1 recovers a globally optimal solution to a discrete optimization problem through a smart exhaustive search process. The search process recursively splits the feasible region into disjoint partitions, and solves a linear programming (LP) relaxation of the discrete problem over each partition at each iteration. This process tracks the following quantities across iterations:

- $\boldsymbol{\lambda}^{\text{best}}$ – the best integer feasible solution found (i.e. the *incumbent solution*);
- $V^{\max}$ – an upper bound on the optimal value, set as the objective value of the incumbent solution;
- $V^{\min}$ – a lower bound on the optimal value, set as the smallest objective value of the LP relaxation over all unexplored partitions.
- $\mathcal{N} = (\mathcal{R}^t, v^t)_{t=1}^{|\mathcal{N}|}$ – a collection of nodes that represent unexplored areas of the feasible region. Each node is a convex partitions of the feasible region $\mathcal{R}^t$ and a lower bound for the LP relaxation over that region v^t.

Each B&B iteration aims to determine if a partition contains an integer feasible solution that has a better objective value than the incumbent solution. The LP relaxation can be used to assess this whenever:

Algorithm 1 Branch and Bound Search

Input

MIP	MIP formulation for optimization problem with the form of (2.1)
SelectNode	rule to choose a node from collection of nodes
SplitPartition	rule to split a partition into disjoint subsets
$\varepsilon^{\text{stop}} \in [0,1]$	optimality gap of acceptable solution

Initialize

$(V^{\min}, V^{\max}) \leftarrow (0, \infty)$	bounds on the optimal value
$\mathcal{R}^0 \leftarrow \operatorname{conv}(\mathcal{L})$	partition at root node
$v^0 \leftarrow V^{\min}$	lower bound at root node
$\mathcal{N} \leftarrow \{(\mathcal{R}^0, v^0)\}$	initial node set
$\varepsilon \leftarrow \infty$	initial optimality gap

```
 1: while $\varepsilon > \varepsilon^{\text{stop}}$ do
 2:     $(\mathcal{R}^t, v^t) \leftarrow$ SelectNode $(\mathcal{N})$          ▷$t$ is index of removed node
 3:     solve LP$(\mathcal{R}^t)$
 4:     $V^{\text{LP}} \leftarrow$ optimal value to LP$(\mathcal{R}^t)$
 5:     $\boldsymbol{\lambda}^{\text{LP}} \leftarrow$ optimal solution to LP$(\mathcal{R}^t)$
 6:     if $\boldsymbol{\lambda}^{\text{LP}}$ is integer feasible then
 7:         if $V^{\text{LP}} < V^{\max}$ then
 8:             $V^{\max} \leftarrow V^{\text{LP}}$          ▷update upper bound
 9:             $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^{\text{LP}}$          ▷update incumbent solution
10:         end if
11:     else if $\boldsymbol{\lambda}^{\text{LP}}$ is not integer feasible then
12:         $(\mathcal{R}^{t_1}, \mathcal{R}^{t_2}) \leftarrow$ SplitPartition$(\mathcal{R}^t, \boldsymbol{\lambda}^{\text{LP}})$          ▷$\mathcal{R}^{t_1}, \mathcal{R}^{t_2}$ are disjoint subsets of $\mathcal{R}^t$
13:         $(v^{t_1}, v^{t_2}) \leftarrow (V^{\text{LP}}, V^{\text{LP}})$          ▷$V^{LP}$ is lower bound for $\mathcal{R}^{t_1}, \mathcal{R}^{t_2}$
14:         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\mathcal{R}^{t_1}, v^{t_1}), (\mathcal{R}^{t_2}, v^{t_2})\}$          ▷add child nodes to $\mathcal{N}$
15:     end if
16:     $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(\mathcal{R}^s, v^s) \mid v^s \geq V^{\max}\}$          ▷prune suboptimal nodes
17:     $V^{\min} \leftarrow \min_{\mathcal{N}} v^s$          ▷global lower bound is smallest among nodes in $\mathcal{N}$
18:     $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$          ▷update optimality gap
19: end while
Output: $\boldsymbol{\lambda}^{\text{best}}$          ▷ $\varepsilon$-optimal solution to MIP
```

(i) The LP relaxation is feasible and has an integral solution. In this case, the search has found the best integer feasible solution for this partition.

(ii) The LP relaxation is feasible and has a continuous solution whose objective value exceeds the current upper bound. In this case, the search has shown that any integer feasible solution contained in this partition is provably suboptimal.

(iii) The LP relaxation is infeasible. In this case, the search has determined that there is no integer feasible solution over this partition.

If none of these conditions hold, then the LP relaxation cannot be used to determine if a partition contains a better integer feasible solution. In this case, the search splits the region into two disjoint partitions that will be solved at later iterations.

B&B terminates when: (i) it has found an integer feasible solution and shown that all other regions do contain a better solution; or (ii) the search has not found any integer feasible solution, and shown that all other regions do not contain one. In the first case, the incumbent solution corresponds to an optimal solution to the discrete optimization problem. Here, the proof of optimality follows from the fact that the search process has only discarded regions that did not contain an integer feasible solution with a better objective value than the incumbent.

B&B search can be visualized in terms of a search tree (i.e. the B&B tree). Here, each *branch* corresponds to disjoint region of the feasible region, and each *node* represents a partition of the feasible region for which we solve an LP relaxation. The total number of nodes in the tree reflects the total computation required to obtain a certifiably optimal solution. In our setting, the size of the B&B tree is limited by the number of distinct combinations of discrete variables in the MIP formulation of the optimization problem 2.1. Since the number of discrete variables in (2.1) is finite, the computation required to find an optimal solution is also finite. Formally, the worst-case running time of any B&B algorithm is $O(Tb^h)$ where T is the maximum time to process any given node, b is the maximum number of children generated at any node (i.e. $b = 2$ for most MIP solvers), and h is the maximum depth of the tree (i.e. $h \leq \min(n, |\mathcal{L}|)$ for the problems that we consider).

The practical running time of a B&B algorithm depends on several factors:

- *Node Processing Speed*: The overall running time of B&B search is directly dependent on the time to solve an LP at each node. This depends on the LP solver and the LP formulation.
- *Node Selection*: The node selection rule affects the order in which the nodes are processed (e.g. first-in-first-out, last-in-first-out, or node that defines the lower bound).
- *Branching Rule*: When the LP returns a continuous solution that is not provably suboptimal, the search process splits the partition into two subpartitions by means of a branching rule. This rule typically uses splits by choosing a continuous variable, and a direction to split (e.g. upward or downward).
- *Strength of LP Relaxation*: Stronger LP relaxations restrict the feasible region without discarding the optimal solution. The optimal value to these problems is

larger, which results in stronger lower bounds, and improves the ability of the search to discard provably suboptimal regions (e.g. in Step 16). Stronger LP relaxations may also improve the ability to find an integer feasible solution by eliminating non-integer solutions that would otherwise be solutions of the LP relaxation.

- *Symmetry*: If a problem has multiple optima, then the size of the B&B tree will be large because the algorithm will be unable to discard regions of the search space that contain optimal solutions.

2.2.2 Working Effectively with a MIP Solver

Modern MIP solvers are able to substantially improve the performance of B&B search through several techniques (see e.g. Achterberg, 2009, for a comprehensive overview). Some common techniques include:

- *Preprocessing.* These techniques are designed to eliminate redundant variables and constraints in the IP formulation. The goal is to reduce the size of the B&B tree and speed up LP solution times (Gamrath et al., 2015; Achterberg et al., 2016).
- *Variable Selection.* Choosing good variables to branch on often leads to a dramatic reduction in terms of the number of nodes needed to solve an instance Pëtursson.
- *Cuts.* These techniques produce constraints ("cuts") that can be included in the feasible region of the LP relaxation to eliminate non-integer solutions that would otherwise be solutions of the LP relaxation. Cuts reduce the size of the B&B tree as the additional constraints in the LP relaxation are either infeasible (which prevents branching), or with larger objective values (which improves the effectiveness of pruning).
- *Heuristics.* These techniques include: feasibility heuristics, to produce integer feasible solutions; and polishing heuristics, to improve integer feasible solutions found by the solver.

Although these techniques can dramatically improved the performance of B&B search in modern solvers (Achterberg and Wunderling, 2013), they are also designed to handle a broad class of discrete optimization problems. In practice, MIP solvers may not effectively identify the structure within a specific class of problems and may use techniques that lead to ineffective B&B search (see Klotz and Newman, 2013, for a discussion). On difficult problems, this can significantly increase the computation required to find good feasible solutions, and the time to recover a proof of optimality[1].

In order to address this, we develop several specialized techniques for our class of problems that can be used effectively with modern solvers to recover provably optimal solution. The key strategies that we use include:

[1]Klotz and Newman (2013), for example, mention: "[There] are many mathematically equivalent ways in which to express a model, and each optimizer has its own set of default algorithmic parameter settings. Choosing from these various model expressions and algorithmic settings can profoundly influence solution time. Although it is theoretically possible to try each combination of parameters settings, in practice, random experimentation would require vast amounts of time and would be unlikely to yield significant improvements."

- *Tighter Formulations*: We develop careful formulations that result in stronger LP relaxations and tighter lower bounds throughout B&B.
- *Symmetry Reduction*: We aim to reduce symmetry in our problem by enforcing preferences between solutions that would otherwise have the same objective value. In scoring system problem, for instance, we require that solutions have coprime coefficients, and use a small ℓ_1-penalty to discard coprime solutions via perturbation.
- *Reducing Data-Related Computation*: We present a data reduction technique that can be used to discard training redundant data (see Algorithm 3). We also present a cutting-plane algorithm that decouples data-related computation from the B&B search (see Algorithm 5).
- *Cuts, Bounds, Heuristics*: We design techniques to improve the lower bounds in B&B search (Algorithms 2, 8, which can be viewed as cuts). We also propose heuristics to polish solutions found by the MIP solver (Algorithm 6), and to produce integer feasible solutions via rounding (Algorithm 7). Since these techniques are used during B&B search, it is important for them to run quickly. Otherwise using the techniques will end up slowing progress due to overhead.

2.2.3 Computational Complexity

The optimization problems that we consider in Chapters 3 and 6 involve NP-hard decision problems, such as 0-1 loss minimization (Ben-David et al., 2003), ℓ_0 regularization (Amaldi and Kann, 1998), and minimization over integers (Gary and Johnson, 1979). NP-hardness is a property that describes the worst-case computational complexity for a general class of problems. Explicitly, these results state that there exists certain instances of the optimization problems for which the time to find a certifiably optimal solution using any algorithm has to scale exponentially with the size of the instance. Such results do not fully characterize the running time associated with solving these optimization problems. It may be that the datasets and constraints will result in instances that do not correspond to the worst-case instance. Moreover, even if running time scales exponentially, it may be manageable for instances that we need to solve.

2.2.4 Scalability

The performance benchmarks in Chapters 3 and 6 show that we can find provably optimal solutions to these problems for many real-world datasets. The running time and scalability of this approach depends on several factors, namely:

- *Constraints on Model Form.* The constraints on model form affect the number of feasible models and govern the size of the search space for the problem. For a fixed dataset, the time to recover a provably optimal solution will decrease as the constraints become more stringent. If the constraints are too stringent, however, then it may require more time to find a feasible solution that satisfies these constraints.

- *Convexity of the Loss Function.* If the problem uses a convex loss (e.g. the risk score problem), then running time scales linearly in the number of samples when the number of dimensions is fixed (see e.g. the cutting-plane algorithm in Chapter 6). In this case, the overall running time is characterized by feature selection, integrality restriction, and operational constraints.
- *Separability of the Classification Problem.* If the dataset contains more features than samples, the classification problem is likely to be linearly separable. In this setting, it becomes easy to find a solution that minimizes the 0-1 loss. Scalability then depends on the difficulty of feature selection, integer restriction, and operational constraints.
- *Desired Optimality Gap.* B&B may find the optimal solution early on in the search process, but require longer to pair it with a certificate of optimality. In practice, a good feasible solution without a certificate of optimality corresponds to a model that performs well on the training data. This model may still be deployed, as it satisfies obeys all constraints and its performance can be evaluated on a test set.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Optimized Scoring Systems

In this chapter, we consider the problem of learning scoring systems for decision-making.

Organization

This chapter is organized as follows. In Section 3.1, we define the scoring system problem and discuss its special properties. In Section 3.2, we present IP formulations for SLIM. In Section 3.3, we describe how SLIM can enforce operational constraints on accuracy and model form. In Section 3.5, we present theoretical bounds on the training and testing accuracy of scoring systems. In Section 3.4, we discuss techniques to reduce computation. In Section 3.6, we present experimental results to benchmark the accuracy, sparsity, and computation SLIM relative to other classification methods.

Notes

This chapter draws upon material in Ustun and Rudin (2016b), Zeng et al. (2016), Ustun et al. (2013). Software for SLIM is available at: http://github.com/ustunb/slim-python and http://github.com/ustunb/slim-matlab.

3.1 Problem Statement

We formalize the scoring system problem as follows. We start with a dataset of n i.i.d. training examples $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$ where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^{d+1}$ is a vector of features $[1, x_{i,1}, \ldots, x_{i,d}]^\top$ and $y_i \in \mathcal{Y} = \{-1, +1\}$ is a class label. We represent a scoring system as a linear classification model:

$$\hat{y} = \operatorname{sign}\left(\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle\right) = \begin{cases} +1 & \text{if } \langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle > 0, \\ -1 & \text{if } \langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle \leq 0. \end{cases}$$

Here, $\boldsymbol{\lambda} \subseteq \mathcal{L} \subseteq \mathbb{R}^{d+1}$ is a vector of coefficients $[\lambda_0, \lambda_1, \ldots, \lambda_d]^\top$ and λ_0 is an intercept term.

In this setup, the coefficient vector $\boldsymbol{\lambda}$ determines all parameters of a scoring system. In particular, the coefficient λ_j represents the *points* for feature j for $j = 1, \ldots, d$. Given an example with features $\boldsymbol{x}_i$, users first tally the points for all features such that $\lambda_j \neq 0$ to obtain a *score* $\sum_{j=1}^d \lambda_j x_{i,j}$, then obtain the prediction by comparing the score and the value of the intercept λ_0 as:

$$\hat{y}_i = \begin{cases} +1 & \text{if } \sum\limits_{j=1}^{d} \lambda_j x_{i,j} > \lambda_0, \\ -1 & \text{if } \sum\limits_{j=1}^{d} \lambda_j x_{i,j} \leq \lambda_0. \end{cases}$$

We learn the values of coefficients by solving a discrete optimization problem that we refer to as *scoring system problem* (SLIMIP), shown in Definition 3.1.

Definition 3.1 (Scoring System Problem, SLIMIP)
The scoring system problem is a discrete optimization problem with the form:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{L}, \\ & \gcd(\boldsymbol{\lambda}) = 1. \end{aligned} \tag{3.1}$$

where:

- $l_{01}(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{y}_i \neq y_i]$ *is the* $0--1$ *loss function;*
- $\|\boldsymbol{\lambda}\|_0 = \sum_{j=1}^{d} \mathbb{1}[\lambda_j \neq 0]$ *is the* ℓ_0*-seminorm;*
- $\mathcal{L} \subset \mathbb{Z}^{d+1}$ *is a finite set of feasible coefficient vectors (user-provided);*
- $C_0 > 0$ *is a trade-off parameter to balance accuracy and sparsity (user-provided);*
- $\gcd(\boldsymbol{\lambda}) = 1$ *is a symmetry-breaking constraint to ensure coefficients are coprime.*

Definition 3.2 (Supersparse Linear Integer Model, SLIM)
*A Supersparse Linear Integer Model (*SLIM*) is a scoring system built using the optimal solution to* SLIMIP.

SLIMIP is designed to learn scoring systems that are fully optimized for accuracy, sparsity, small integer coefficients, and operational constraints. Here, the objective minimizes the 0–1 loss function for accuracy, penalizes the ℓ_0-norm for sparsity. The feasible region restricts the coefficients to a small set of bounded integers such as $\mathcal{L} = \{-10, \ldots, 10\}^{d+1}$, and can be further customized to include additional operational constraints (see Section 3.3). Since the 0–1 loss and ℓ_0-norm are scale invariant, we include an additional constraint to restrict coefficients to coprime integers. As shown in Figure 3.4, this can substantially reduce the set of feasible coefficients, which can reduce computation within B&B by removing symmetry and improve generalization (see Section 3.5).

The scoring system problem is designed to fit models that attain a pareto-optimal trade-off between accuracy and sparsity: when we minimize 0–1 loss and the ℓ_0-penalty, we only sacrifice classification accuracy to attain higher sparsity, and vice versa. Minimizing the 0–1 loss produces scoring systems that are completely robust to outliers and attain the best learning-theoretic guarantee on predictive accuracy (see e.g. Brooks, 2011; Nguyen and Sanner, 2013). Similarly, controlling for sparsity via ℓ_0-norm prevents an additional loss in accuracy due to ℓ_1-regularization (see Lin et al., 2008, for a discussion).

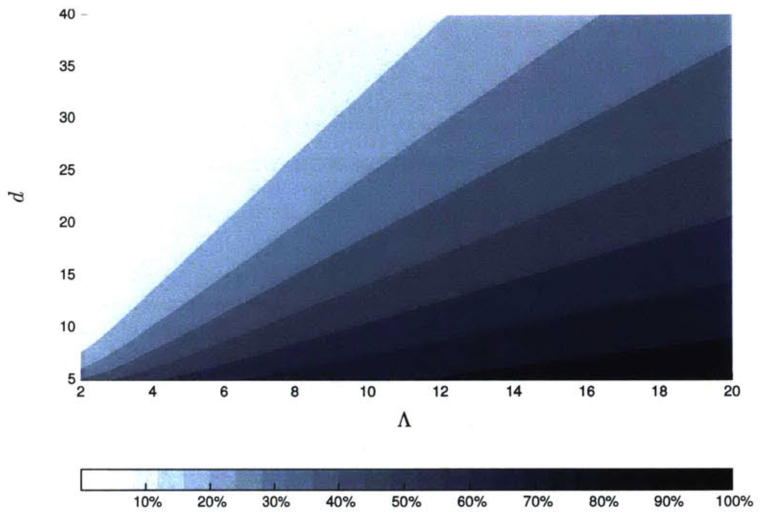


Figure 3.1: Proportion of d-dimensional integer vectors with $|\lambda_j| \leq \Lambda$ that are coprime. We plot the value of $|\mathcal{C}(\Lambda)|/|\{-\Lambda, \ldots, \Lambda\}^d|$ where $\mathcal{C}(\Lambda) = \{\boldsymbol{\lambda} \in \mathbb{Z}^d | \gcd(\boldsymbol{\lambda}) = 1 \text{ and } |\lambda_j| \leq \Lambda\}$.

3.1.1 Assumptions, Notation, and Terminology

We make the following assumptions regarding the coefficient set $\mathcal{L}$

(i) $\mathcal{L}$ contains the trivial set of coefficients $\mathcal{L}^{\text{trivial}} = \{\boldsymbol{\lambda}^+, \boldsymbol{\lambda}^-\}$.

(ii) the intercept λ_0 is not penalized in the objective.

(iii) the intercept λ_0 is not regularized by the constraints.

Assumption (i) ensures that the scoring system problem always has a feasible solution. The coefficient vectors $\boldsymbol{\lambda}^+ \in \mathcal{L}^{\text{trivial}}$ and $\boldsymbol{\lambda}^- \in \mathcal{L}^{\text{trivial}}$ can be set as the smallest coefficients required to predict the same class for all examples $\boldsymbol{x}_i \in \mathcal{X}$. The exact coefficients can be set as: $\lambda_0^+ = 1$ and $\lambda_j^+ = 0$ for $j = 1, \ldots, d$, $\lambda_0^- = -1$ and $\lambda_j^- = 0$ for $j = 1, \ldots, d$. If the dataset contains only binary features, then we can use $\lambda_0^- = 0$ and $\lambda_j^- = 0$ for $j = 1, \ldots, d$.

Assumption (ii) means that the more precise version of the ℓ_0-penalty is $C_0 \left\| \boldsymbol{\lambda}_{[1,d]} \right\|_0$ where $\boldsymbol{\lambda} = [\lambda_0, \boldsymbol{\lambda}_{[1,d]}]$. Assumption (iii) can be satisfied by setting the bounds of the intercept as described in Proposition 3.3.

Proposition 3.3 (Coefficient Set for Intercept Term)
Given a training dataset $\mathcal{D}_n$, denote a finite discrete set of coefficient vectors as

$$\mathcal{L} = \left\{ \boldsymbol{\lambda} \mid \lambda_0 \in \mathcal{L}_0,\ \boldsymbol{\lambda}_{[1:d]} \in \mathcal{L}_{[1:d]} \right\},$$

where:

- $\mathcal{L}_0 = \{\Lambda_0^{\min}, \ldots, \Lambda_0^{\max}\}$ *is a finite set of feasible values for the intercept;*
- $\mathcal{L}_{[1:d]} = \{(\lambda_1, \ldots, \lambda_d) \mid \lambda_j \in \mathcal{L}_j\}$ *is a finite set of feasible coefficient vectors.*

Denote a set of coefficient vectors where the intercept is not bounded as:

$$\mathcal{K} = \{\boldsymbol{\lambda} \mid \lambda_0 \in \mathbb{Z}, \boldsymbol{\lambda}_{[1:d]} \in \mathcal{L}_{[1:d]}\}.$$

If we choose,

$$\Lambda_0^{\max} \geq \max_{\boldsymbol{\lambda} \in \mathcal{L}_{[1:d]}} \sum_{j=1}^{d} \lambda_j x_{i,j}, \tag{3.2}$$

$$\Lambda_0^{\min} \leq \min_{\boldsymbol{\lambda} \in \mathcal{L}_{[1:d]}} \sum_{j=1}^{d} \lambda_j x_{i,j} \tag{3.3}$$

Then,

$$\underset{\boldsymbol{\lambda} \in \mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0 \in \underset{\boldsymbol{\lambda} \in \mathcal{K}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0$$

Thus, the bounds on the intercept will not affect the performance of the model.

3.1.2 Setting the Trade-off Parameter

One of the key benefits of using an exact formulation is that we can set the values of parameters such as C_0 purposefully (i.e. without the need for cross-validation). In the following remarks, we how the trade-off parameter C_0 can be set to achieve a desired trade-off between accuracy and sparsity, or to obtain a scoring system with the minimal (or maximal) level of sparsity.

Remark 3.4 (Meaning of the trade-off parameter)
*The trade-off parameter C_0 in the objective of the scoring system problem (*SLIMIP*) represents the maximum accuracy that will be sacrificed to remove a feature from an optimal model.*

In other words, given the non-zero coefficients of an optimal scoring system $\lambda_j^* \neq 0$ for $j = 1, \ldots, d$, setting $\lambda_j^* \leftarrow 0$ will reduce the training accuracy of the model by at least C_0.

Remark 3.5 (Bounds on the trade-off parameter)
Denote an optimal solution of the scoring system problem for a fixed trade-off parameter C_0 as:

$$\boldsymbol{\lambda}^*(C_0) \in \underset{\boldsymbol{\lambda} \in \mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0 .$$

The trade-off parameter C_0 can be bounded to values within the interval

$$[C_0^{\min}, C_0^{\max}] \subseteq \left(\frac{1}{nd}, \frac{\min(n^+, n^-)}{nd}\right).$$

Setting $C_0 \leq C_0^{\min}$ will fit a scoring system with maximal training accuracy:

$$\boldsymbol{\lambda}^*(C_0) \in \underset{\boldsymbol{\lambda} \in \mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}).$$

Setting $C_0 \geq C_0^{\max}$ will fit a scoring system with maximal sparsity:

$$\boldsymbol{\lambda}^*(C_0) \in \underset{\boldsymbol{\lambda} \in \mathcal{L}}{\operatorname{argmin}}\, \left\|\boldsymbol{\lambda}\right\|_0 .$$

3.1.3 Using the Optimality Gap

A separate benefit of working with an exact formulation is that the optimality gap is also meaningful.

Remark 3.6 (Rule of Thumb for the Optimality Gap)
Given a feasible solution to the scoring system problem $\boldsymbol{\lambda}^{\mathrm{f}}$ with a non-zero optimality gap $\varepsilon > 0$, the accuracy and sparsity of an optimal scoring system can be bounded using the following rules-of-thumb:

$$l(\boldsymbol{\lambda}^{\mathrm{f}}) - l(\boldsymbol{\lambda}^{*}) \leq \varepsilon l(\boldsymbol{\lambda}^{\mathrm{f}})$$
$$\left\|\boldsymbol{\lambda}^{\mathrm{f}}\right\|_0 - \|\boldsymbol{\lambda}^{*}\|_0 \leq \left\lfloor \varepsilon \left\|\boldsymbol{\lambda}^{\mathrm{f}}\right\|_0 \right\rfloor$$

In other words, if we had a solution $\boldsymbol{\lambda}^{\mathrm{f}}$ with an optimality gap of $\varepsilon = 10\%$. Then we would know that the optimal scoring system could have an error rate within 10% of the current model (i.e. error rate of and/or use 10% fewer features).

In practice, the bounds on the accuracy and sparsity of the best scoring system can be further improved as shown in Remark 3.7. Here, we make use of prior lower bounds on the error rate $L^{\min}$ and the number of non-zero coefficients $R^{\min}$ as well as the fact that accuracy and sparsity must be integer-valued.

Remark 3.7 (Bounds from the Optimality Gap)
Given a feasible solution $\boldsymbol{\lambda}^{\mathrm{f}}$ to the scoring system problem with a non-zero optimality gap $\varepsilon > 0$, the accuracy and sparsity of an optimal scoring system can be bounded as:

$$\frac{\left\lceil n[(1-\varepsilon)V(\boldsymbol{\lambda}^{\mathrm{f}}) - C_0 R^{\min}] \right\rceil}{n} \leq l(\boldsymbol{\lambda}^{*})$$
$$\left\lceil \frac{(1-\varepsilon)V(\boldsymbol{\lambda}^{\mathrm{f}}) - L^{\min}}{C_0} \right\rceil \leq \|\boldsymbol{\lambda}^{*}\|_0$$

where $0 \leq L^{\min} \leq l(\boldsymbol{\lambda}^{*})$ is a lower bound on the 0-1 loss function, and $0 \leq R^{\min}$ is a lower bound on the number of terms in the model.

3.2 Methodology

In the remainder of this section, we consider a minor extension of the scoring system problem for imbalanced classification problems. Here, we replace the 0–1 loss function in the objective with the weighted 0–1 loss. The weighed 0–1 loss incorporates unit misclassification costs for positive and negative examples, w^+ and w^-, which can be set to control the relative accuracy between positive and negative examples.

Definition 3.8 (Scoring System Problem for Imbalanced Data)
For a given training dataset $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$, trade-off parameter $C_0 > 0$, and coefficient set $\mathcal{L} \subset \mathbb{Z}^{d+1}$, $\text{SLIMIP}(\mathcal{D}, \mathcal{L}, C_0, w^+)$ is a discrete optimization problem with the form:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & \frac{w^+}{n} l_{01}^+(\boldsymbol{\lambda}) + \frac{w^-}{n} l_{01}^-(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{L} \\ & \gcd(\boldsymbol{\lambda}) = 1, \end{aligned} \tag{3.4}$$

where:

- $l_{01}^+(\cdot) = \sum_{i \in I^+} \mathbb{1}[\hat{y}_i \neq +1]$;
- $l_{01}^-(\cdot) = \sum_{i \in I^-} \mathbb{1}[\hat{y}_i \neq -1]$;
- $w^+ > 0$ *is the unit misclassification cost for positive examples;*
- $w^- > 0$ *is the unit misclassification cost for negative examples.*

We assume that $w^+ + w^- = 2$ so that the imbalanced formulation is equivalent to the standard scoring system problem in Definition 3.1 when $w^+ = 1$. This assumption also allows us to control the relative accuracy using only w^+ (i.e. given any value of w^+, we would set $w^- = 2 - w^+$). In addition, this assumption ensures that the trade-off C_0 parameter can be interpreted as the price between the normalized misclassification cost and sparsity for different values of w^+. The value of w^+ can also be bounded and set purposefully as shown in Remark 3.9.

Remark 3.9 (Bounds on Misclassification Costs Parameters)
Denote an optimal solution of the scoring system problem in (3.4) *for a fixed unit misclassification cost* w^+ *as and no* ℓ_0*-regularization as:*

$$\boldsymbol{\lambda}^*(w^+) \in \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} \frac{w^+}{n} l_{01}^+(\boldsymbol{\lambda}) + \frac{w^-}{n} l_{01}^-(\boldsymbol{\lambda}).$$

The value of w^+ *can be bounded to values in the interval:*

$$[w^{+,\min}, w^{+,\max}] \in \left(\frac{2}{1+n^+}, \frac{2n^-}{1+n^-}\right).$$

Setting $w^+ \geq w^{+,\max}$ *will fit a scoring system with maximal training accuracy on the positive examples:*

$$\boldsymbol{\lambda}^*(w^+) \in \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} l_{01}^+(\boldsymbol{\lambda}).$$

Setting $w^+ < w^{+,\min}$ *will fit a scoring system with maximal training accuracy on the negative examples:*

$$\boldsymbol{\lambda}^*(w^+) \in \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} l_{01}^-(\boldsymbol{\lambda}).$$

In what follows, we show two IP formulations to fit SLIM: the first formulation is for a general setting; the second is for a setting where the features are binary. The formulations have minor differences and could be combined into a single formulation. Here, we present the formulations separately, instead of combining them, because: (i) the formulations return slightly different scoring systems; (ii) the differences between formulations can significantly affect computation in B&B; (iii) we use both formulations in this dissertation SLIM (in particular, the formulation in (3.5) is used for the benchmarks in Sections 3.6 and the sleep apnea problem in Chapter 4 while the formulation in (3.7a) is used for the recidivism prediction application in Chapter 5.

3.2.1 IP Formulation

We fit a SLIM scoring system using the IP formulation in (3.5).

Definition 3.10 (SLIM IP Formulation)
The optimal solution to the scoring system problem $\text{SLIMIP}(\mathcal{D}, \mathcal{L}, C_0, w^+)$ *can be obtained by solving the following integer program:*

$$
\begin{array}{llrcllr}
\min & V & & & & & \text{(3.5a)} \\
\text{s.t.} & V & = & L + C_0 R + \epsilon \sum_{j=1}^{d} \beta_j & & \textit{obj. value} & \text{(3.5b)} \\
& L & = & \frac{w^+}{n} \sum_{i \in I^+} z_i + \frac{w^-}{n} \sum_{i \in I^-} z_i & & \textit{total loss} & \text{(3.5c)} \\
& R & = & \sum_{j=1}^{d} \alpha_j & & \textit{total size} & \text{(3.5d)} \\
& M_i z_i & \geq & \gamma - \sum_{j=0}^{d} y_i \lambda_j x_{i,j} & i = 1,\ldots,n & \textit{0–1 loss} & \text{(3.5e)} \\
& \Lambda_j^{\max} \alpha_j & \geq & \lambda_j & j = 1,\ldots,d & \ell_0\textit{-norm} & \text{(3.5f)} \\
& \Lambda_j^{\min} \alpha_j & \geq & -\lambda_j & j = 1,\ldots,d & \ell_0\textit{-norm} & \text{(3.5g)} \\
& \beta_j & \geq & \lambda_j & j = 1,\ldots,d & \ell_1\textit{-norm} & \text{(3.5h)} \\
& \beta_j & \geq & -\lambda_j & j = 1,\ldots,d & \ell_1\textit{-norm} & \text{(3.5i)} \\
& \lambda_j & \in & \mathcal{L}_j & j = 0,\ldots,d & \textit{coefficient set} & \\
& z_i & \in & \{0,1\} & i = 1,\ldots,n & \textit{error indicators} & \\
& \alpha_j & \in & \{0,1\} & j = 1,\ldots,d & \ell_0 \textit{ variables} & \\
& \beta_j & \in & \mathbb{R}_+ & j = 1,\ldots,d & \ell_1 \textit{ variables} &
\end{array}
$$

Here, the constraints in (3.5e) set the loss variables $z_i = \mathbb{1}\left[y_i \boldsymbol{\lambda}^\top \boldsymbol{x}_i \leq 0\right]$ to 1 if a linear classifier with coefficients $\boldsymbol{\lambda}$ misclassifies example i. This is a Big-M constraint for the 0–1 loss that depends on scalar parameters γ and M_i (see e.g., Rubin, 2009). The value of M_i represents the maximum score when example i is misclassified, and can be set as $M_i = \max_{\boldsymbol{\lambda} \in \mathcal{L}}(\gamma - y_i \boldsymbol{\lambda}^\top \boldsymbol{x}_i)$ which is easy to compute since $\mathcal{L}$ is finite. The value of γ represents the "margin" and should be set as a lower bound on $y_i \boldsymbol{\lambda}^\top \boldsymbol{x}_i$. When the features are binary, γ can be set to any value between 0 and 1. In other cases, the lower bound is difficult to calculate exactly so we set $\gamma = 0.1$, which makes an implicit assumption on the values of the features. The model size is set to R in constraint (3.5d) via the indicator variables $\alpha_j := \mathbb{1}\left[\lambda_j \neq 0\right]$. These variables are defined by Big-M constraints in (3.5f) – (3.5g), and $\beta_j := |\lambda_j|$ is defined by the constraints in (3.5h)–(3.5i).

Big-M Parameters

Restricting coefficients to a finite set $\boldsymbol{\lambda} \in \mathcal{L}$ has a major benefit for the SLIM IP formulation in comparison to other formulations that minimize the 0–1-loss and/or penalize the ℓ_0-norm. Many IP formulations compute the 0–1 loss and ℓ_0-norm by means of indicator constraints that require users to specify Big-M constraints (see Wolsey, 1998, for a description).

Restricting the coefficients to a finite set allows us to bound the Big-M constants in the SLIM IP formulation. The Big-M constant for computing the 0–1 loss in constraints (3.5e) can be bounded as $M_i \leq \max_{\boldsymbol{\lambda}\in\mathcal{L}}(\gamma - y_i\boldsymbol{\lambda}^\top\boldsymbol{x}_i)$, and the Big-M constant used to compute the ℓ_0-norm in constraints (3.5f) can be bounded as $\Lambda_j \leq \max_{\lambda_j\in\mathcal{L}_j}|\lambda_j|$ (c.f. Brooks, 2011; Guan et al., 2009, where these parameters have to be set using sufficiently large constants). Bounding these constants ensures that the IP formulation has a tighter LP relaxation, which improves the ability of a MIP solver to prune effectively and recover a certifiably optimal solution (see Camm et al., 1990; Belotti et al., 2016, for a discussion).

Perturbation Penalty

We address the coprimality requirement in the scoring system problem $\gcd(\boldsymbol{\lambda}) = 1$ using a *perturbation* technique (see e.g., Margot, 2010, for a discussion). Specifically, we include a small ℓ_1-penalty in the objective function so the optimal solution corresponds to the classifier with the smallest (i.e. coprime) coefficients, $\hat{y} = \text{sign}\,(x_1 + x_2)$. To illustrate the use of the ℓ_1 penalty, consider a classifier such as $\hat{y} = \text{sign}\,(x_1 + x_2)$. If the objective only minimized the 0–1 loss and an ℓ_0-penalty, then $\hat{y} = \text{sign}\,(2x_1 + 2x_2)$ would attain the same objective value as $\hat{y} = \text{sign}\,(x_1 + x_2)$ because it makes the same predictions and has the same number of non-zero coefficients. In Remark 3.11, we show how ϵ can be set to a value that is small enough to avoid any ℓ_1-regularization (i.e. to guarantee that SLIMIP never sacrifices accuracy or sparsity to attain a smaller ℓ_1-penalty).

Remark 3.11 (Setting the Perturbation Penalty to Prevent ℓ_1-Regularization) *Given an instance of the scoring system problem* $\textsc{SlimIP}(\mathcal{D}, C_0, \mathcal{L})$, *let*

$$\epsilon^{\max} = \frac{\min\left(\frac{1}{n}, C_0\right)}{\max_{\boldsymbol{\lambda}\in\mathcal{L}} \|\boldsymbol{\lambda}\|_1}.$$

If the perturbation parameter is set to a value such that $\epsilon \in (0, \epsilon^{\max})$ *then any optimal solution to the scoring system problem*

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda}\in\mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0,$$

satisfies

$$\boldsymbol{\lambda}^* \subseteq \underset{\boldsymbol{\lambda}\in\mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0 \text{ and } \gcd(\{\lambda_j^*\}_{j=0}^d) = 1. \tag{3.6}$$

Thus, the optimal solution is a scoring system with coprime coefficients that attains a Pareto-optimal trade-off between accuracy and sparsity.

Auxiliary Variables

The IP formulation includes auxiliary variables V, L, R to represent the values of the objective, the loss, and the model size respectively. These variables are redundant in the sense that they are fully determined by other constraints in the formulation and could be dropped without impacting the optimal solution (e.g. since $R = \sum_j \alpha_j$ we would substitute all values of R with $\sum_j \alpha_j$, which would reduce the size of the formulation by 1 variable and 1 constraint). In light of this, we include them so that we can update bounds on these quantities during the solution process when we use control callbacks. If we did not have these variables in the formulation, then it may still be able to do this by adding cuts. In comparison to this approach, however, auxiliary variables have the benefit in that they reduces overhead (e.g. it is much faster to update a bound on R, than to add a cut that restricts $\sum_j \alpha_j < R^{\max}$) and do not increase the size of the formulation. This approach also has some flexibility in that we can make use of a larger class of callback functions (e.g. most of the control callbacks in the CPLEX API let us update the bounds on a single variable, but do not necessarily let us add an additional constraint).

3.2.2 IP Formulation for Binary Data

In Definition 3.7a, we present an IP formulation for settings with binary features $\boldsymbol{x}_i \in \{0,1\}^{d+1}$. This formulation provides a tighter relaxation by exploiting the fact that we are likely to get repeated feature values among observations. In what follows, we assume that the training dataset is composed of unique examples $(\boldsymbol{x}_i, y_i) \neq (\boldsymbol{x}_{i'}, y_{i'})$ for all $i, i' \in I$. If the dataset does contain duplicate examples, then we count the number of duplicates for each example using the sample weights n_i^+ and n_i^-.

Definition 3.12 (SLIM IP Formulation for Binary Data)
Given a training dataset $\mathcal{D}$ with binary features $\boldsymbol{x}_i \in \{0,1\}^d$ for $i = 1, \ldots, n$, the optimal solution to the scoring system problem $\text{SLIMIP}(\mathcal{D}, \mathcal{L}, C_0, w^+)$ *can be obtained by solving the following integer program:*

$$
\begin{array}{lllr}
\min\limits_{\boldsymbol{\lambda}} & V & & \\
\text{s.t.} & V = \dfrac{w^+}{n} L^+ + \dfrac{w^-}{n} L^- + C_0 R + \epsilon \displaystyle\sum_{j=1}^{d} \beta_j & & \textit{obj. value}(3.7a) \\
& L^+ = \displaystyle\sum_{i \in I^+} n_i^+ z_i^+ & & \textit{total loss}(3.7b) \\
& L^- = \displaystyle\sum_{i \in I^-} n_i^- z_i^- & & \textit{total loss}(3.7c) \\
& R = \displaystyle\sum_{j=1}^{d} \alpha_j & & \textit{total size}(3.7d) \\
& M_i z_i^+ \geq 1 - \displaystyle\sum_{j=0}^{d} \lambda_j x_{i,j} & i \in I^+ & \textit{error on +}(3.7e) \\
& M_i z_i^- \geq \displaystyle\sum_{j=0}^{d} \lambda_j x_{i,j} & i \in I^- & \textit{error on -}(3.7f) \\
& 1 = z_i^+ + z_{i'}^- & i, i' \in I^{\text{conflict}} & \textit{conflict bound}(3.7g) \\
& \Lambda_j^{\max} \alpha_j \geq \lambda_j & j = 1,\ldots,d & \ell_0 \textit{ indicators}(3.7h) \\
& \Lambda_j^{\min} \alpha_j \geq -\lambda_j & j = 1,\ldots,d & \ell_0 \textit{ indicators}\ (3.7i) \\
& \beta_j \geq \lambda_j & j = 1,\ldots,d & \ell_1\textit{-norm}\ (3.7j) \\
& \beta_j \geq -\lambda_j & j = 1,\ldots,d & \ell_1\textit{-norm}(3.7k) \\
& & & \\
& V \in [V^{\min}, V^{\max}] & & \textit{objective} \\
& L \in [L^{\min}, L^{\max}] & & \textit{loss} \\
& R \in \{R^{\min},, \ldots, R^{\max}\} & & \ell_0 \textit{ norm} \\
& \lambda_j \in \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\} & j = 0,\ldots,d & \textit{coefficient values} \\
& z_i^+ \in \{0,1\} & i \in I^+ & \textit{error indicators} \\
& z_i^- \in \{0,1\} & i \in I^- & \textit{error indicators} \\
& \alpha_j \in \{0,1\} & j = 1,\ldots,d & \ell_0 \textit{ indicators} \\
& \beta_j \in \{0, \ldots, \max(\Lambda_j^{\min}, \Lambda_j^{\max})\} & j = 1,\ldots,d & \ell_1 \textit{ variables}
\end{array}
$$

There are three key differences between this IP formulation and the one in Definition 3.10.

1. The loss constraints are expressed in terms of the number of *distinct* points in the dataset. Here, the set I^+ represents the set of distinct points with positive labels, and the set I^- represents the set of distinct points with negative examples. Here, n_i^+ count the number of points in the original dataset with features $\boldsymbol{x}_i$ and $y_i = +1$. Similarly, n_i^- counts the number of points with features $\boldsymbol{x}_i$ and $y_i = -1$. Thus, $n^+ = \sum_{i \in I^+} n_i$ and $n^- = \sum_{i \in I^-} n_i$. This reduces the size of the problem that we pass to an MIP solver.

2. We can include a lower bound on the error rate by counting the number of points $i, i^{'}$ with identical features but opposite labels (i.e., $\boldsymbol{x}_i = \boldsymbol{x}_{i'}$ but $y_i = -y_i$). This strengthens the lower bound of the LP relaxation and speeds up the progress of B&B.

3. We set the margin for the negative examples is 0 while the margin for the positive examples is 1. This means that for positive examples, we have a correct prediction if and only if the score ≥ 1. For negative points, we have a correct prediction if and only if the score ≤ 0. This provides a slight computational advantage since the negative points do not need to have scores below -1 to be correctly classified, which reduces the size of the Big-M parameter and required coefficient set. For instance, say we would to produce a linear model that encode: "predict $\hat{y} = +1$ unless x_1 or x_2 are true." Using the IP formulation with the margin of $\gamma \in (0,1)$ on both positives and negatives, the optimal SLIM classifier would be: $\hat{y} = \text{sign}(1 - 2x_1 - 2x_2)$. In contrast, the margin of the current formulation is: $\hat{y} = \text{sign}(1 - x_1 - x_2)$, which uses smaller coefficients, and produces a slightly simpler model.

The first two changes could also be incorporated in (3.5), though they would not necessarily be effective in this setting because it is far less likely for a dataset to have duplicate rows using real-valued features.

3.3 Operational Constraints

One of the key benefits of SLIM is that it can address operational constraints related to the accuracy and sparsity of predictive models. The following techniques provide users with a practical approach to customize prediction models. They are made possible by the facts that: (i) the variables used to model the 0–1 loss z_i and ℓ_0-penalty α_j in the SLIM IP formulation can also be used to enforce additional constraints related to accuracy and sparsity; and (ii) the free parameters in the SLIM objective can be set without tuning.

3.3.1 Class-Based Accuracy

The majority of classification problems in the medical domain are imbalanced. Handling imbalanced data is incredibly difficult for most classification methods since maximizing classification accuracy often produces a trivial model (i.e., if the probability of heart attack is 1%, a model that never predicts a heart attack is still 99% accurate). SLIM has a unique benefit in these settings because it can fit a model at a specific point on the ROC curve without parameter tuning. Given hard constraints on sensitivity (or specificity), we can encode these as *loss constraints* into the IP formulation, then solve a single IP to obtain the least specific (or most sensitive) model.

Example 3.13 (Most Sensitive Model subject to an FPR Constraint)
To fit a scoring system that maximizes the true positive rate while maintaining a false positive rate of at most $\gamma \in [0, 1]$, we can solve an IP formulation with the constraint

$$\sum_{i \in I^-} z_i \leq \lfloor \gamma n^- \rfloor$$

where $w^+ \geq w^{+,\max}$.

Assuming that $w^+ + w^- = 2$, we set $w^+ \geq w^{+\max}$ so that the objective weighs a mistake on a single positive example as much as all of the negative examples. In a typical setting, this would return a scoring system that classifies all positive examples correctly at the expense of misclassify all of the negative examples in order to classify an additional positive example correctly. In this case, however, the loss constraint explicitly limits the error on negative examples to γ. Thus, the optimal solution corresponds to a scoring system that attains the highest sensitivity among models with a maximum error of γ on negative examples.

Similar versions can be used to fit models that satisfy constraints related to a range of accuracy-related measures, such as limits on sensitivity and specificity.

3.3.2 Fairness

The loss variables in the SLIM IP formulation can also be used to address constraints related to fairness. In comparison to existing approaches, this has the advantage that it can deal with a range of different measures. Consider, for example, the 80% rule (Feldman et al., 2015) that is commonly used to limit disparate impact among two groups of individuals A and B:

$$\frac{\Pr(\hat{y}=+1|i\in A)}{\Pr(\hat{y}=+1|i\in B)} \leq 80\% = \tau.$$

Example 3.14 (Disparate Impact)
To fit a scoring system that limits disparate impact to $\tau \in [0,1]$ on the training data, we can solve an IP formulation with the constraint:

$$\frac{1}{n_A^+}\sum_{i\in A^+} z_i + \frac{1}{n_A^-}\sum_{i\in A^-}(1-z_i) = \tau\left(\frac{1}{n_B^+}\sum_{i\in B^+} z_i + \frac{1}{n_B^-}\sum_{i\in B^-}(1-z_i)\right)$$

where,

- $A^+ = A\cap I^+$, $A^- = A\cap I^-$, $n_A^+ = |A^+|$, $n_A^- = |A^-|$
- $B^+ = B\cap I^+$, $B^- = B\cap I^-$, $n_B^+ = |B^+|$, $n_B^- = |B^-|$

3.3.3 Monotonicity and Side Information

The simplest way to incorporate side information is to enforce monotonic relationships between input variables and the predicted outcome (Feelders and Pardoel, 2003; Ben-David, 1995; Gupta et al., 2016). In Example 3.15, we show how this can be addressed by changing bounds on the coefficient variables in the IP formulations. Note that by adding these bounds, we can also drop the ℓ_1-norm variables β_j from the formulation.

Example 3.15 (Basic Monotonicity)
To fit a scoring system where λ_j has a positive relationship with the predicted outcome for $j \in \mathcal{J}^+$ and $\lambda_j \in \mathcal{J}^-$ has a negative relationship with the predicted outcome, set the variable bounds of SLIM IP formulation:

$$\lambda_j \geq 0 \text{ for } j \in \mathcal{J}^+,$$
$$\lambda_j \leq 0 \text{ for } j \in \mathcal{J}^-.$$

We can extend monotonicity requirements to several variables. In an extreme case, this can be used to enforce a constraint to fix the predicted output as shown in Example 3.16. Such constraints could be used to fit scoring systems with safety guarantees (Amodei et al., 2016). This may also be a useful tool when domains experts disagree with the predictions of a model. As a example, say domain experts expect a model to predict $\hat{y}_p = +1$ for a prototypical example with features $\boldsymbol{x}_p$, then we can enforce this condition by means of a prediction constraint. We can then compare the training accuracy of both models to report the loss in accuracy due to this constraint.

Example 3.16 (Prediction Constraints)
To fit a scoring system that guarantees that $\hat{y}_p = y_p$ for an example with features $\boldsymbol{x}_p$, we can solve an IP formulation with the constraint:

$$z_p = -y_p.$$

3.3.4 Feature-Based Constraints

SLIM provides fine-grained control over the composition of input variables in a scoring system by formulating feature-based constraints using the indicator variables for the ℓ_0-norm $\alpha_j := \mathbb{1}[\lambda_j \neq 0]$. In the following examples, we describe how the indicator variables α_j can be used to limit the model size, or to formulate logical constraints between features such as "either-or" conditions and "if-then" conditions. This presents a practical alternative to create classification models that obey structured sparsity constraints (Jenatton et al., 2011) or hierarchical constraints (Bien et al., 2013).

Example 3.17 (Limited Model Size)
To fit a scoring system with at most $R^{\max}$ input variables, we can solve an IP formulation with the constraint:

$$\sum_{j=1}^{d} \alpha_j \leq R^{\max}$$

Setting $C_0 = C_0^{\min}$ guarantees that the optimal solution to the scoring system problem with return the accurate scoring system with at most $R^{\max}$ non-zero coefficients.

Example 3.18 (Optimized Thresholding)
To fit a scoring system that chooses the best threshold for a real-valued feature, we can include a set of binary threshold variables to the training data

$$\boldsymbol{x}_{j,k} = \mathbb{1}[\boldsymbol{x}_j \geq t_k] \text{ for } k = 1, \ldots, K$$

Then add then constraint:

$$\sum_{k=1}^{K} \alpha_{j,k} \leq 1$$

where $\alpha_{j,k} = \mathbb{1}[\lambda_{j,k} \neq 0]$.

Example 3.19 (Hierarchical Constraints)
To fit a scoring system that guarantees that it will include Hypertension and HeartAttack if it also includes Stroke, we can solve an IP formulation with the constraint:

$$\alpha_{HeartAttack} + \alpha_{Hypertension} \leq 2\alpha_{Stroke}.$$

3.3.5 Feature-Based Preferences

Domain experts often have soft preferences between different input variables. SLIM allows practitioners to encode these preferences by specifying a distinct trade-off parameter for each coefficient $C_{0,j}$.

Example 3.20 (Variable Prices)
To fit a scoring system that will only use feature j if it provides an additional improvement in training accuracy of at least $\delta > 0$, set the trade-off parameter in the IP formulation to

$$C_{0,j} = C_0 + \delta.$$

This approach can also be used to handle problems with missing data.

Example 3.21 (Missing Data)
Consider training a model where feature j contains $m < n$ missing points. Instead of dropping these points, we can impute the values of the m missing examples, and adjust the trade-off parameter $C_{0,j}$ so that our model only uses feature j if it yields an additional gain in accuracy of more than m examples:

$$C_{0,j} = C_0 + \frac{m}{n}\min(n^+, n^-).$$

The adjustment factor in Example 3.21 is chosen so that: if $m = 0$ then $C_{0,j} = C_0$; and if $m = n$ then $C_{0,j} \geq \min(n^+, n^-)$ and the coefficient is dropped entirely. This ensures that features with lots of imputed values are more heavily penalized than features with fewer imputed values.

3.4 Algorithmic Improvements

3.4.1 Active Set Polishing

We now discuss a *polishing* procedure to optimize the values of coefficients over a fixed set of features. This procedure takes as input a feasible set of coefficients from the SLIM IP, $\boldsymbol{\lambda}^{\text{feasible}}$ and solves the polishing IP shown in (3.8) to obtain a set of polished coefficients $\boldsymbol{\lambda}^{\text{polished}}$. The polished coefficients $\boldsymbol{\lambda}^{\text{polished}}$ represent the optimal solution to the scoring system problem over the *active set* $A := \{j : \lambda_j^{\text{feasible}} \neq 0\}$.

$$
\begin{array}{llll}
\min & L & & (3.8\text{a}) \\
\text{s.t.} & L = \frac{w^+}{n} \sum_{i \in I^+} n_i z_i^+ + \frac{w^-}{n} \sum_{i \in I^-} n_i z_i^- & & \textit{total error}\,(3.8\text{b}) \\
& M_i z_i^+ \geq 1 - \sum_{j \in A} \lambda_j x_{i,j} & i \in I^+ & \textit{error for +ve}\,(3.8\text{c}) \\
& M_i z_i^- \geq \sum_{j \in A} \lambda_j x_{i,j} & i \in I^- & \textit{error for -ve}\,(3.8\text{d}) \\
& 1 = z_i^+ + z_{i'}^- & (i, i') \in I^{\text{conflict}} & \textit{conflicting labels}\,(3.8\text{e}) \\
& & & \\
& L \in [L^{\min}, L^{\max}] & & \textit{total loss} \\
& \lambda_j \in \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\} & j \in A & \textit{coefficient set} \\
& z_i^+, z_i^- \in \{0, 1\} & i \in I^+ & \textit{error indicators}
\end{array}
$$

The polishing IP can be solved to optimality very quickly because: (i) the IP contains far fewer variables since it does involve feature selection, and only optimizes coefficients in the active set A; (ii) the IP can be initialized with good upper and lower bounds on the loss using the objective value of $V(\boldsymbol{\lambda}^{\text{feasible}})$; (iii) for problems with binary input variables, the number of loss constraints can be significantly reduced by eliminating duplicates (e.g. if $|A| = 5$ then any dataset will contain at most $|\{-1, +1\}| \times |\{0, 1\}^5| = 64$ possible unique data points, and thus the same number of possible loss constraints).

The polishing procedure can be used as a post-processing technique if SLIM does not return a provably optimal solution. In this case, we would aim to keep a pool of solutions discovered by the MIP solver and polish all solutions within this pool. In Chapter 5, for example, we use the polishing procedure on all a subset of good feasible solutions when we are unable to recover a certifiably optimal solution. In all cases, we can solve the polishing IP to optimality in <1 second.

Alternatively, the polishing procedure can also be called dynamically via a control callback (i.e. to polish solutions that are found by a MIP solver). In Algorithm 2, we present a technique that we call *polish-and-cover* to illustrate this approach. This strategy works as follows: when the MIP solver discovers a new integer feasible solution, we call the polishing procedure to obtain a polished solution. We use the solution from the polished solution to update the upper bound in the IP. If the polishing procedure returns an optimal solution, we also include a cover cut to eliminate this particular solution from the search region. Thus, using the strategy improves both the upper and lower bound in B&B.

Algorithm 2 Polish-and-Cover

```
Input
    $\textsc{SlimIP}(\mathcal{D}_n, \mathcal{L}, C_0, w^+)$        valid instance of scoring system problem
    $\textsc{PolishSolution}(\cdot)$        function to formulate and solve polishing IP
    $\varepsilon^{\text{stop}} \in [0, 1]$        optimality gap of acceptable solution
    $t^{\max}$        time limit to run polishing heuristic

1: repeat
2:     $\boldsymbol{\lambda}^{\text{feasible}} \leftarrow$ integer feasible solution to $\textsc{SlimIP}$
3:     $(\boldsymbol{\lambda}^{\text{polished}}, V^{\text{polished}}, \varepsilon^{\text{polished}}) \leftarrow \textsc{PolishSolution}(\boldsymbol{\lambda}^{\text{feasible}})$
4:     if $V^{\text{polished}} < V^{\max}$ then
5:         $V^{\max} \leftarrow V^{\text{polished}}$        ▷update upper bound
6:         $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^{\text{best}}$        ▷update best solution
7:     end if
8:     if $\varepsilon^{\text{polished}} = 0.0$ then
9:         $A \leftarrow \{j : \lambda_j^{\text{polished}} \neq 0 \text{ for } j = 0, \ldots, d\}$
10:        update feasible region of $\textsc{SlimIP}$ with constraint:
               $\sum_{j \in A} (1 - \alpha_j) + \sum_{j \notin A} \alpha_j \leq |A| - 1$
                                                        ▷done using callback
11:    end if
12: until $\varepsilon \leq \varepsilon^{\text{stop}}$
Output: $\boldsymbol{\lambda}^{\text{best}}$        $\varepsilon$-optimal solution to $\textsc{SlimIP}$
```

3.4.2 Data Reduction

Data reduction is a technique to reduce computation by discarding redundant training data. This technique can be used with any binary classification method where the model is fit by solving an optimization problem. However, it is best suited for methods where the algorithm used to solve the optimization problem scales poorly with the number of examples. In what follows, we present data reduction in general setting, then describe how it can be used to reduce computation in our setting.

Data Reduction for Binary Classification

Consider fitting a binary classification model $f : \mathcal{X} \to \mathcal{Y}$ by solving an optimization problem with the form:

$$\min_{f \in \mathcal{F}} V(f; \mathcal{D}_n) \tag{3.9}$$

Data reduction aims to reduce the computation in the training process by removing *redundant* examples from $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$ (i.e., examples that can be omitted without changing the optimal solution). The technique requires a *surrogate problem* that is cheaper to optimize:

$$\min_{f \in \tilde{\mathcal{F}}} \tilde{V}(f; \mathcal{D}_n) \tag{3.10}$$

Given training data $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$, data reduction solves $n+1$ *variants* of the surrogate to identify redundant examples. These examples are then removed to output a *reduced* dataset $\mathcal{D}_m \subseteq \mathcal{D}_n$ that is guaranteed to yield the same optimal classifier as $\mathcal{D}_n$. Thus, the computational gain from data reduction comes from training a model with $\mathcal{D}_m$ (i.e., solving an instance of the original problem with $n-m$ fewer examples).

We show the data reduction procedure in Algorithm 3. In what follows, we describe how this procedure works in greater detail. We use the following notation:

- *original objective function*: $V(\cdot) : \mathcal{F} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathbb{R}$
- *surrogate objective function*: $\tilde{V}(\cdot) : \tilde{\mathcal{F}} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathbb{R}$
- *original feasible classifiers*: $\mathcal{F}$
- *surrogate feasible classifiers*: $\tilde{\mathcal{F}}$
- *optimizers for the original problem*: $f^* \in \mathcal{F}^* = \text{argmin}_{f \in \mathcal{F}} V(f)$
- *optimizers for the surrogate problem*: $\tilde{f} \in \tilde{\mathcal{F}}^* = \text{argmin}_{f \in \tilde{\mathcal{F}}} \tilde{V}(f)$

Data reduction can be used with any surrogate so long as the ε-level set of the surrogate contains the optimizers of the original problem. That is, we can use any feasible set $\tilde{\mathcal{F}}$ and any objective function $\tilde{V}(\cdot)$ so long as we can specify a value of ε such that

$$\tilde{V}(f^*) \leq \tilde{V}(\tilde{f}) + \varepsilon \qquad \forall f^* \in \mathcal{F}^* \text{ and } \tilde{f} \in \tilde{\mathcal{F}}^*. \tag{3.11}$$

The width of the the surrogate level set ε is related to the amount of data that will be removed. If ε is too large, the method will not remove very many examples and will be less helpful for reducing computation (see Figure 3.3). In Theorem 3.23, we provide sufficient conditions for a surrogate loss function to satisfy the level set condition in (3.11).

In the first stage of data reduction, we solve the surrogate to: (i) compute the upper bound on the objective value of classifiers in the surrogate level set $\tilde{V}(\tilde{f}) + \varepsilon$; and (ii) to identify a baseline label $\tilde{y}_i := \text{sign}\left(\tilde{f}(\boldsymbol{x}_i)\right)$ for each example $i = 1, \ldots, n$. In the second stage of data reduction, we solve a variant of the surrogate problem for each example $i = 1, \ldots, n$. The i^{th} variant of the surrogate problem includes an additional constraint that forces example i to be classified as $-\tilde{y}_i$:

$$\min_{f} \; \tilde{V}(f; \mathcal{D}_n) \text{ s.t. } f \in \tilde{\mathcal{F}} \text{ and } \tilde{y}_i f(\boldsymbol{x}_i) < 0 \tag{3.12}$$

We denote an optimal classifier to the i^{th} variant as $\tilde{f}_{\text{-}i}$. If $\tilde{f}_{\text{-}i}$ lies outside of the surrogate level set (i.e., $\tilde{V}(\tilde{f}_{\text{-}i}) > \tilde{V}(\tilde{f}) + \varepsilon$) then no classifier in the surrogate level set can label point i as $-\tilde{y}_i$. In other words, all classifiers in the surrogate level set must label this point as $\tilde{y}_i$. Since the surrogate level set contains the optimal classifiers to the original problem, we can remove example i from the reduced dataset $\mathcal{D}_m$ because an optimal classifier to the original problem must label i as $\tilde{y}_i$. We illustrate this situation in Figure 3.2.

In Theorem 3.22, we prove that we obtain the same set of optimal classifiers if we train a model with the initial data $\mathcal{D}_n$ or the reduced data $\mathcal{D}_m$.

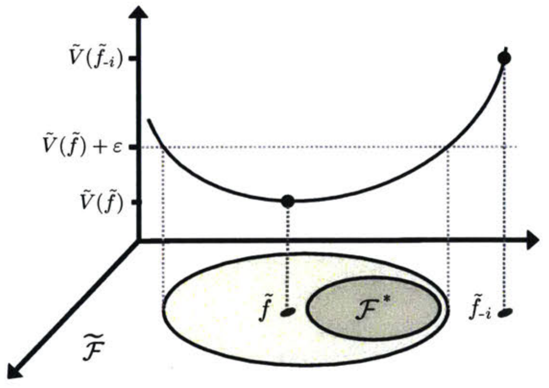


Figure 3.2: We initialize the data reduction procedure with a value of ε large enough so that $\tilde{V}(f^*) < \tilde{V}(\tilde{f}) + \varepsilon$ for all $f^* \in \mathcal{F}^*$ and all $\tilde{f} \in \tilde{\mathcal{F}}^*$. Here, $f^* \in \mathcal{F}^*$ is an optimal classifier to the original problem and $\tilde{f} \in \tilde{\mathcal{F}}^*$ is the optimal classifier to the surrogate problem. Data reduction fits a classifier $\tilde{f}_{\text{-}i}$ for each example in $\mathcal{D}_n$ by solving a variant of the surrogate with a constraint that forces $\tilde{f}_{\text{-}i}$ to classify i in a different way than $\tilde{f}$. If $\tilde{V}(\tilde{f}_{\text{-}i}) > \tilde{V}(\tilde{f}) + \varepsilon$, then we know the predicted class of example i under f^* and can remove it from $\mathcal{D}_m$.

Algorithm 3 Data Reduction from $\mathcal{D}_n$ to $\mathcal{D}_m$

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^n$                                   training data
    $\varepsilon > 0$                                   width of the surrogate level set
    $\min \tilde{V}(f; \mathcal{D}_n)$ s.t. $f \in \tilde{\mathcal{F}}$                      surrogate problem

Initialize
    $\mathcal{D}_m \leftarrow \emptyset$                                          reduced training data
    $\tilde{f} \leftarrow \text{argmin}_f \tilde{V}(f; \mathcal{D}_n)$                             surrogate value
1: for $i = 1 \ldots n$ do
2:     $\tilde{y}_i \leftarrow \text{sign}(\tilde{f}(\boldsymbol{x}_i))$
3:     $\tilde{f}_{\text{-}i} \leftarrow \text{argmin}\, \tilde{V}(f; \mathcal{D}_n)$ s.t. $f \in \tilde{\mathcal{F}}$ and $\tilde{y}_i f(\boldsymbol{x}_i) < 0$
4:     if $\tilde{V}(\tilde{f}_{\text{-}i}; \mathcal{D}_n) \leq \tilde{V}(\tilde{f}; \mathcal{D}_n) + \varepsilon$ then
5:         $\mathcal{D}_m \leftarrow \mathcal{D}_m \cup (\boldsymbol{x}_i, y_i)$
6:     end if
7: end for
Output: $\mathcal{D}_m$                                          reduced training data
```

Theorem 3.22 (Equivalence of the Reduced Data)
Consider an optimization problem to fit a classifier $f \in \mathcal{F}$ with data $\mathcal{D}_n$,

$$\min_{f \in \mathcal{F}} \; V(f; \mathcal{D}_n),$$

as well as a surrogate problem to fit a classifier $f \in \tilde{\mathcal{F}}$ with data $\mathcal{D}_n$,

$$\min_{f \in \tilde{\mathcal{F}}} \; \tilde{V}(f; \mathcal{D}_n).$$

Denote the optimizers of these problems as:

$$f^* \in \mathcal{F}^* = \underset{f \in \mathcal{F}}{\text{argmin}}\, V(f; \mathcal{D}_n)$$
$$\tilde{f} \in \tilde{\mathcal{F}}^* = \underset{f \in \tilde{\mathcal{F}}}{\text{argmin}}\, \tilde{V}(f; \mathcal{D}_n)$$

If we choose a value of ε so that

$$\tilde{V}(f^*; \mathcal{D}_n) \leq \tilde{V}(\tilde{f}; \mathcal{D}_n) + \varepsilon \quad \forall f^* \in \mathcal{F}^* \text{ and } \tilde{f} \in \tilde{\mathcal{F}}^*, \tag{3.13}$$

then Algorithm 3 will output a reduced dataset $\mathcal{D}_m \subseteq \mathcal{D}_n$ such that

$$\underset{f \in \mathcal{F}}{\text{argmin}}\, V(f; \mathcal{D}_n) = \underset{f \in \mathcal{F}}{\text{argmin}}\, V(f; \mathcal{D}_m). \tag{3.14}$$

Theorem 3.23 (Sufficient Conditions to Satisfy the Level Set Condition)
Consider an optimization problem whose objective minimizes the 0–1 loss function $l_{01} : \mathbb{R}^d \to \mathbb{R}$ and a surrogate problem that minimizes a surrogate loss function $l_s : \mathbb{R}^d \to \mathbb{R}$. Denote the optimizers to these respective problems as:

$$\boldsymbol{\lambda}^*_{01} \in \operatorname*{argmin}_{\boldsymbol{\lambda} \in \mathbb{R}^d} l_{01}(\boldsymbol{\lambda}),$$
$$\boldsymbol{\lambda}^*_{s} \in \operatorname*{argmin}_{\boldsymbol{\lambda} \in \mathbb{R}^d} l_{s}(\boldsymbol{\lambda}).$$

If the surrogate loss l_s satisfies the following properties for all $\boldsymbol{\lambda}, \boldsymbol{\lambda}^*_{01}, \boldsymbol{\lambda}^*_s$:

I. *Upper bound on the 0–1 loss:* $l_{01}(\boldsymbol{\lambda}) \leq l_s(\boldsymbol{\lambda})$

II. *Lipschitz near* $\boldsymbol{\lambda}^*_{01}$: $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*_s\| < A \implies l_s(\boldsymbol{\lambda}) - l_s(\boldsymbol{\lambda}^*_s) < L\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*_s\|$

III. *Curvature near* $\boldsymbol{\lambda}^*_s$: $\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*_s\| > C_{\boldsymbol{\lambda}} \implies l_s(\boldsymbol{\lambda}) - l_s(\boldsymbol{\lambda}^*_s) > C_s$

IV. *Closeness of loss near* $\boldsymbol{\lambda}^*_{01}$: $|l_s(\boldsymbol{\lambda}^*_{01}) - l_{01}(\boldsymbol{\lambda}^*_{01})| < \varepsilon$

then it will also satisfy a level-set condition required for data reduction,

$$l_s(\boldsymbol{\lambda}^*_{01}) \leq l_s(\boldsymbol{\lambda}^*_s) + \varepsilon \qquad \forall \boldsymbol{\lambda}^*_{01} \text{ and } \boldsymbol{\lambda}^*_s,$$

whenever $\varepsilon = LC_{\boldsymbol{\lambda}}$ obeys $C_s > 2\varepsilon$.

Data Reduction for SLIM

Data reduction can applied to SLIM by using the LP relaxation of the SLIMIP as the surrogate problem. This can be used as a preprocessing technique to reduce the size of the IP formulation before training, or as an iterative procedure that is called by the solver during the training process as feasible solutions are found.

When we use the LP relaxation to SLIMIP as the surrogate problem, we can determine a suitable width for the surrogate level set ε using a feasible solution. To see this, let us denote the SLIMIP as $\min_f V(f)$ s.t. $f \in \mathcal{F}$, and denote its LP relaxation as $\min_f V(f)$ s.t. $f \in \tilde{\mathcal{F}}$. In addition, let us denote the optimal solution to SLIMIP as f^* and the optimal solution to the LP relaxation as $\tilde{f}$. Since $\mathcal{F} \subseteq \tilde{\mathcal{F}}$, we have that $V(\tilde{f}) \leq V(f^*)$. For any feasible solution to SLIMIP $\hat{f} \in \mathcal{F}$, we also have that $V(f^*) \leq V(\hat{f})$. Combining both inequalities, we see that,

$$V(\tilde{f}) \leq V(f^*) \leq V(\hat{f}).$$

Thus, use any feasible solution to SLIMIP $\hat{f} \in \mathcal{F}$ to set the width of the surrogate level set as

$$\varepsilon(\hat{f}) := V(\hat{f}) - V(\tilde{f}).$$

In Figure 3.3, we show much training data can be discarded when we train a SLIM scoring system on the `bankruptcy` dataset (see Table 3.1 for details). Here, we show the percentage of data removed for $\varepsilon \in [\varepsilon^{\min}, \varepsilon^{\max}]$ where $\varepsilon_{\min}$ and $\varepsilon_{\max}$ represent the smallest and largest widths of the surrogate level set that can be used in practice. In particular, $\varepsilon^{\min}$ is computed using the optimal solution to the IP as:

$$\varepsilon^{\min} := V(f^*) - V(\tilde{f}),$$

and $\varepsilon^{\max}$ is computed using a trivial solution $\boldsymbol{\lambda} = \mathbf{0}$):

$$\varepsilon^{\max} := V(\mathbf{0}) - V(\tilde{f}).$$

In this case, we can discard over 40% of the training data by using the trivial solution $\boldsymbol{\lambda} = \mathbf{0}$, and discard over 80% of the training data by using a higher quality feasible solution.

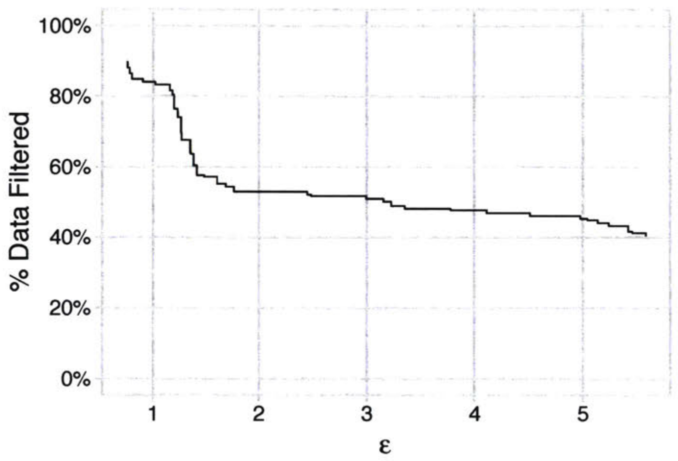


Figure 3.3: Effectiveness of data reduction on the `bankruptcy` dataset. We show the proportion of training data filtered across the full range of ε. Here, the original problem is an instance of SLIMIP with $C_0 = 0.01$ and $\mathcal{L} = \{-10, \ldots, 10\}^{d+1}$.

3.5 Bounds on the Accuracy of Scoring Systems

In this section, we present bounds on the training and testing accuracy of SLIM classifiers.

3.5.1 Generalization Bounds

In Theorem 3.24, we improve the generalization bound from Theorem 2.2 by excluding models that are provably suboptimal from the hypothesis space. Here, we exploit the fact that we can bound the number of non-zero coefficients based in terms of the trade-off parameter C_0.

Theorem 3.24 (Generalization of Sparse Discrete Linear Classifiers)
Let $\mathcal{F}$ denote the set of linear classifiers with coefficients $\boldsymbol{\lambda}$ from a finite set $\mathcal{L}$ such that:

$$\mathcal{F} = \left\{ f : \mathcal{X} \to \mathcal{Y} \mid f(\boldsymbol{x}) = \operatorname{sign}\left(\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle\right) \right\}$$
$$\boldsymbol{\lambda} \in \underset{\boldsymbol{\lambda} \in \mathcal{L}}{\operatorname{argmin}}\, l_{01}(\boldsymbol{\lambda}) + C_0 \left\| \boldsymbol{\lambda} \right\|_0$$

For every $\delta > 0$, with probability at least $1 - \delta$, every classifier $f \in \mathcal{F}$ obeys:

$$R^{\text{true}}(f) \leq R^{\text{emp}}(f) + \sqrt{\frac{\log(|\mathcal{H}_{d,C_0}|) - \log(\delta)}{2n}},$$

where:

$$\mathcal{H}_{d,C_0} = \left\{ \boldsymbol{\lambda} \in \mathcal{L} \,\middle|\, \left\| \boldsymbol{\lambda} \right\|_0 \leq \left\lfloor \frac{1}{C_0} \right\rfloor \right\}.$$

This theorem relates the trade-off parameter C_0 in the SLIM objective to the generalization of SLIM scoring systems. It indicates that increasing the value of the C_0 parameter will produce a model with better generalization properties.

In Theorem 3.25, we show an alternative generalization bound by exploiting the fact that SLIM scoring systems use coprime integer coefficients (see Remark 3.11). In particular, we express the generalization bound from Theorem 2.2 using the d-dimensional Farey points of level Λ (see Marklof, 2012, for a definition).

Theorem 3.25 (Generalization of Coprime Discrete Linear Classifiers)
Let $\mathcal{F}$ denote the set of linear classifiers with coprime integer coefficients bounded by Λ:

$$\begin{aligned}
\mathcal{F} &= \Big\{ f : \mathcal{X} \to \mathcal{Y} \;\Big|\; f(\boldsymbol{x}) = \operatorname{sign}\left(\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle\right) \text{ and } \boldsymbol{\lambda} \in \mathcal{L} \Big\}, \\
\mathcal{L} &= \Big\{ \boldsymbol{\lambda} \in \hat{\mathbb{Z}}^d \;\Big|\; |\lambda_j| \le \Lambda \text{ for } j = 1, \dots, d \Big\}, \\
\hat{\mathbb{Z}}^d &= \Big\{ \boldsymbol{z} \in \mathbb{Z}^d \;\Big|\; \gcd(\boldsymbol{z}) = 1 \Big\}.
\end{aligned}$$

For every $\delta > 0$, with probability at least $1 - \delta$, every classifier $f \in \mathcal{F}$ obeys:

$$R^{\text{true}}(f) \le R^{\text{emp}}(f) + \sqrt{\frac{\log(|\mathcal{C}_{d,\Lambda}|) - \log(\delta)}{2n}},$$

where $\mathcal{C}_{d,\Lambda}$ denotes the set of Farey points of level Λ:

$$\mathcal{C}_{d,\Lambda} = \left\{ \frac{\boldsymbol{\lambda}}{q} \in [0,1)^d : (\boldsymbol{\lambda}, q) \in \hat{\mathbb{Z}}^{d+1} \text{ and } 1 \le q \le \Lambda \right\}.$$

The proof involves a counting argument over coprime integer vectors, using the definition of Farey points from number theory. In Figure 3.4, we plot the generalization bound for different values of d and Λ for a setting where $\delta = 0.05$ and $n = 5000$. We see that the generalization bound can be significant, except in cases where the data are high dimensional and Λ is small.

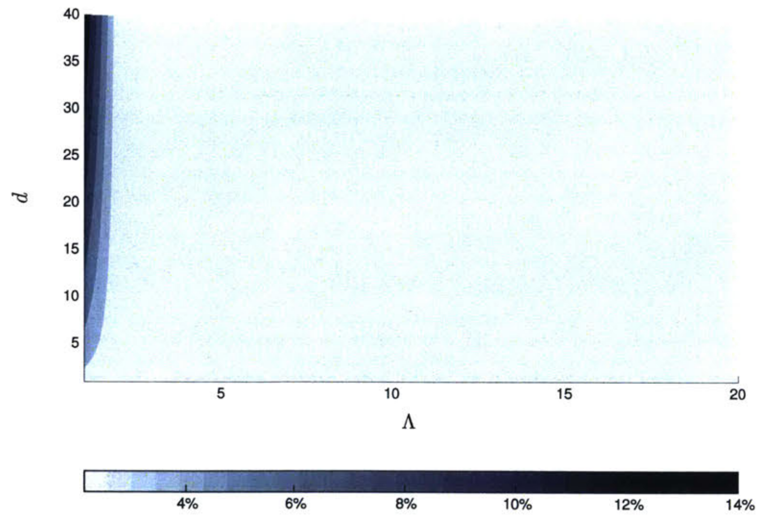


Figure 3.4: Bound on $R^{\text{true}}(f) - R^{\text{emp}}(f)$ from Theorem 3.25 for $\delta = 0.05$ and $n = 5000$.

3.5.2 Discretization Bounds for Binary Data

Theorem 3.26 (Coefficient Set for No Loss in Training Accuracy or Sparsity) *Given a dataset $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$ with binary features $\boldsymbol{x}_i \in \{0,1\}^{d+1}$, let*

$$\boldsymbol{\rho}^* \in \underset{\boldsymbol{\rho} \in \mathbb{R}^{d+1}}{\text{argmin}} \; l_{01}(\boldsymbol{\rho}) + C_0 \|\boldsymbol{\rho}\|_0,$$

denote the coefficients of the most accurate ℓ_0-regularized linear classifier for a trade-off parameter $C_0 > 0$.

If we solve an instance of the scoring system problem $\text{SLIMIP}(\mathcal{D}, C_0, \mathcal{L})$ *where:*

$$\mathcal{L} = \left\{ \boldsymbol{\lambda} \in \mathbb{Z}^{d+1} \;\middle|\; \begin{array}{l} |\lambda_j| \leq (R^{\max} + 2)^{\frac{d+1}{2}} \text{ for } j = 0, \ldots, d \\ R^{\max} \leq \left\lfloor \frac{\min(n^+, n^-)}{C_0} \right\rfloor \end{array} \right\} \tag{3.15}$$

then the optimal solution $\boldsymbol{\lambda}^*$ will satisfy:

$$l_{01}(\boldsymbol{\lambda}^*) = l_{01}(\boldsymbol{\rho}^*) \text{ and } \|\boldsymbol{\lambda}^*\|_0 = \|\boldsymbol{\rho}^*\|_0$$

Thus, the optimal scoring system is guaranteed to attain the same training accuracy and sparsity as the most accurate ℓ_0-regularized linear classifier.

Theorem 3.26 is an extension of Theorem 2.1 for the scoring system problem. The proof uses an upper bound on value of the largest coefficient for a linear classifier (Long and Servedio, 2014) for settings where the features are restricted to sparse Boolean vectors (i.e. $\max_i \|x_i\| \leq k$).

The coefficient set in Theorem 3.26 represents a coefficient set that is large enough to guarantee that the optimal scoring system will attain the same level of accuracy and sparsity as a linear classifier with real-valued coefficients. In other words, for any dataset with binary features, the coefficient set in (3.15) will guarantee that there will be no loss in training accuracy due to the integrality constraint.

3.5.3 Discretization Bounds for Real-Valued Data

Our first result shows that we can find a set of coefficients $\mathcal{L}$ that is large enough to ensure that the training accuracy of a linear classifier with discrete coefficients $\boldsymbol{\lambda} \in \mathcal{L}$ (e.g. SLIM) is no worse than the training accuracy of a baseline linear classifier with real-valued coefficients $\boldsymbol{\rho} \in \mathbb{R}^d$ (e.g. SVM).

Theorem 3.27 (Minimum Margin Resolution Bound)
Let $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_d]^\top \in \mathbb{R}^d$ denote the coefficients of a baseline linear classifier trained using data $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$. Let $X_{\max} = \max_i \|\boldsymbol{x}_i\|_2$ and $\gamma_{\min} = \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}$ denote the largest magnitude and minimum margin achieved by any training example, respectively.

Consider training a linear classifier with coefficients $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_d]^\top$ from the set $\mathcal{L} = \{-\Lambda, \ldots, \Lambda\}^d$. If we choose a resolution parameter Λ such that:

$$\Lambda > \frac{X_{\max}\sqrt{d}}{2\gamma_{\min}}, \tag{3.16}$$

then there exists $\boldsymbol{\lambda} \in \mathcal{L}$ such that:

$$l_{01}(\boldsymbol{\lambda}) \leq l_{01}(\boldsymbol{\rho}).$$

The proof of Theorem 3.27 uses a rounding procedure to choose a resolution parameter Λ so that the coefficient set $\mathcal{L}$ contains a classifier with discrete coefficients $\boldsymbol{\lambda}$ that has the same 0–1 loss as the baseline classifier with real-valued coefficients $\boldsymbol{\rho}$. If the baseline classifier $\boldsymbol{\rho}$ is obtained by minimizing a convex surrogate loss, then the optimal SLIM classifier fit using the coefficient set from Theorem 3.27 may attain a lower 0–1 loss than $\mathbb{1}\left[y_i \boldsymbol{\rho}^\top \boldsymbol{x}_i \leq 0\right]$ as SLIM directly minimizes the 0–1 loss.

The next corollary yields additional bounds on the training accuracy by considering progressively larger values of the margin. These bounds can be used to relate the resolution parameter Λ to a worst-case guarantee on training accuracy.

Corollary 3.28 (k^{th} Margin Resolution Bound)
Let $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_d]^\top \in \mathbb{R}^d$ denote the coefficients of a linear classifier trained with data $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^n$. Let $\gamma_{(k)}$ denote the value of the k^{th} smallest margin, $I_{(k)}$ denote the set of points with $\frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} \leq \gamma_{(k)}$, and $X_{(k)} = \max_{i \notin I_{(k)}} \|\boldsymbol{x}_i\|_2$ denote the largest magnitude of any example $\boldsymbol{x}_i \in \mathcal{D}$ for $i \notin I_{(k)}$.

Consider training a linear classifier with coefficients $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_d]^\top$ from the set $\mathcal{L} = \{-\Lambda, \ldots, \Lambda\}^d$. If we set the resolution parameter Λ such that:

$$\Lambda > \frac{X_{(k)}\sqrt{d}}{2\gamma_{(k)}},$$

then there exists $\boldsymbol{\lambda} \in \mathcal{L}$ such that the 0–1 loss of $\boldsymbol{\lambda}$ and the 0–1 loss of $\boldsymbol{\rho}$ differ by at most $k-1$:

$$l_{01}(\boldsymbol{\lambda}) - l_{01}(\boldsymbol{\rho}) \leq k - 1.$$

We have shown that good discretized solutions exist and can be constructed easily. This motivates that optimal discretized solutions, which by definition are better than rounded solutions, will also be good relative to the best non-discretized solution.

3.6 Performance Benchmarks

In this section, we benchmark the accuracy and sparsity of SLIM scoring systems against other popular classification models. Our goal is to illustrate the off-the-shelf performance of SLIM and show that it can fit accurate scoring systems for real-world datasets in minutes.

3.6.1 Datasets

We ran experiments on 8 datasets from the UCI Machine Learning Repository (Bache and Lichman, 2013) summarized in Table 3.1. We chose these datasets to explore the performance of each method as we varied the size and nature of the training data. We processed each dataset by binarizing all categorical features and some real-valued features.

Dataset	**Reference**	n	d	**Classification Task**
`adult`	Kohavi (1996)	32561	36	predict if a U.S. resident earns over $50K
`breastcancer`	Mangasarian et al. (1995)	683	9	detect breast cancer using a biopsy
`bankruptcy`	Kim and Han (2003)	250	6	predict if a firm will go bankrupt
`haberman`	Haberman (1976)	306	3	predict survival after cancer surgery
`heart`	Detrano et al. (1989)	303	32	identify patients a high risk of heart disease
`mammo`	Elter et al. (2007)	961	12	detect breast cancer using a mammogram
`mushroom`	Schlimmer (1987)	8124	113	predict if a mushroom is poisonous
`spambase`	Cranor and LaMacchia (1998)	4601	57	predict if an e-mail is spam

Table 3.1: Datasets used for performance benchmarks.

3.6.2 Methods

We summarize our training setup for all methods in Table 3.2. We fit SLIM scoring systems using CPLEX 12.6.0, and models with baseline methods using different packages in R 3.1.1 (R Core Team, 2014). For each method, each dataset, and each unique combination of free parameters, we trained 10 models using subsets of the data to estimate predictive accuracy via 10-fold cross-validation (10-CV), and 1 final model using all of the data to assess sparsity and interpretability. We ran all baseline methods without time constraints over a large grid of free parameters. We produced an ℓ_0-regularization path for SLIM by solving 6×11 IPs for each dataset (6 values of $C_0 \times 11$ training runs per C_0). We capped the runtime for each IP at 10 minutes, and solved 12 IPs in parallel on a 12-core 2.7 GHz CPU with 48 GB RAM. Thus, it took at most 1 hour to train SLIM scoring systems for each dataset. Since the `adult` and `haberman` datasets were imbalanced, we trained all methods on these datasets with a weighted loss function where we set $w^+ = n^-/n$ and $w^- = n^+/n$.

Method	Acronym	Software	Settings and Free Parameters
CART Decision Trees	CART	**rpart** Therneau et al. (2012)	default settings
C5.0 Decision Trees	C5.0T	**c50** Kuhn et al. (2012)	default settings
C5.0 Rule List	C5.0R	**c50** Kuhn et al. (2012)	default settings
ℓ_1-Penalized LR	Lasso	**glmnet** Friedman et al. (2010)	1000 values of λ chosen by **glmnet**
ℓ_2-Penalized LR	Ridge	**glmnet** Friedman et al. (2010)	1000 values of λ chosen by **glmnet**
$\ell_1 + \ell_2$-Penalized LR	Elastic Net	**glmnet** Friedman et al. (2010)	1000 values of λ chosen by **glmnet** $\times$ 19 values of $\alpha \in \{0.05, 0.10, \ldots, 0.95\}$
SVM Linear Kernel	SVM Linear	**e1071** Meyer et al. (2012)	25 values of $C \in \{10^{-3}, 10^{-2.75}, \ldots, 10^{3}\}$
SVM RBF Kernel	SVM RBF	**e1071** Meyer et al. (2012)	25 values of $C \in \{10^{-3}, 10^{-2.75}, \ldots, 10^{3}\}$
SLIM Scoring Systems	SLIM	**slim-matlab**	$C_0 \in \{0.01, 0.075, 0.05, 0.025, 0.001, 0.9/nd\}$ $\lambda_j \in \{-10, \ldots, 10\}$ $\lambda_0 \in \{-100, \ldots, 100\}$

Table 3.2: Classification methods used for performance benchmarks.

3.6.3 Results

We summarize the results of our experiments in Table 3.3 and Figures 3.5 to3.6. We benchmark the sparsity of models using the *model size*, which is the number of coefficients for linear models (Lasso, Ridge, Elastic Net, SLIM, SVM Linear), the number of leaves for decision tree models (C5.0T, CART), and the number of rules for rule-based models (C5.0R). For completeness, we set the model size for black-box models (SVM RBF) to the number of features in each dataset.

We show the accuracy and sparsity of all methods on all dataset in Figures 3.5 to 3.6. For each dataset, and each method, we plot a point at the 10-CV mean test error and final model size, and surround this point with an error bar whose height corresponds to the 10-CV standard deviation in test error. In addition, we include ℓ_0-regularization paths for SLIM and Lasso on the right side of Figures 3.5 to 3.6 to show how the test error varies at different levels of sparsity for sparse linear models.

We make the following observations regarding our results:

On Accuracy, Sparsity and Computation

Our results show that many methods are unable to produce models that attain the same levels of accuracy and sparsity as SLIM. As shown in Figures 3.5 to 3.6, SLIM always produces a model that is more accurate than Lasso at some level of sparsity, and sometimes more accurate at all levels of sparsity (e.g., `spambase`, `haberman`, `mushroom`, `breastcancer`). Although the optimization problems to train SLIM scoring systems were NP-hard, we did not find any evidence that computational issues hurt the performance of SLIM on any of the datasets. We obtained accurate and

Dataset	Details		Metric	SLIM	Lasso	Ridge	Elastic Net	C5.0R	C5.0T	CART	SVM Lin.	SVM RBF
`adult`	n	32561	**test error**	17.4 ± 1.4%	17.3 ± 0.9%	17.6 ± 0.9%	17.4 ± 0.9%	26.4 ± 1.8%	26.3 ± 1.4%	75.9 ± 0.0%	16.8 ± 0.8%	16.3 ± 0.5%
	d	37	**train error**	17.5 ± 1.2%	17.2 ± 0.1%	17.6 ± 0.1%	17.4 ± 0.1%	25.3 ± 0.4%	24.9 ± 0.4%	75.9 ± 0.0%	16.7 ± 0.1%	16.3 ± 0.1%
	$\Pr(y=+1)$	24%	**model size**	18	14	36	17	41	87	4	36	36
	$\Pr(y=-1)$	76%	**model range**	7 - 26	13 - 14	36 - 36	16 - 18	38 - 46	78 - 99	4 - 4	36 - 36	36 - 36
`breastcancer`	n	683	**test error**	3.4 ± 2.0%	3.4 ± 2.2%	3.4 ± 2.0%	3.1 ± 2.1%	4.3 ± 3.3%	5.3 ± 3.4%	5.6 ± 1.9%	3.1 ± 2.0%	3.5 ± 2.5%
	d	10	**train error**	3.2 ± 0.2%	2.9 ± 0.3%	3.0 ± 0.3%	2.8 ± 0.3%	2.1 ± 0.3%	1.6 ± 0.4%	3.6 ± 0.3%	2.7 ± 0.2%	0.3 ± 0.1%
	$\Pr(y=+1)$	35%	**model size**	2	9	9	9	8	13	7	9	9
	$\Pr(y=-1)$	65%	**model range**	2 - 2	8 - 9	9 - 9	9 - 9	6 - 9	7 - 16	3 - 7	9 - 9	9 - 9
`bankruptcy`	n	250	**test error**	0.8 ± 1.7%	0.0 ± 0.0%	0.4 ± 1.3%	0.0 ± 0.0%	0.8 ± 1.7%	0.8 ± 1.7%	1.6 ± 2.8%	0.4 ± 1.3%	0.4 ± 1.3%
	d	7	**train error**	0.0 ± 0.0%	0.0 ± 0.0%	0.4 ± 0.1%	0.4 ± 0.7%	0.4 ± 0.2%	0.4 ± 0.2%	1.6 ± 0.3%	0.4 ± 0.1%	0.4 ± 0.1%
	$\Pr(y=+1)$	57%	**model size**	3	3	6	3	4	4	2	6	6
	$\Pr(y=-1)$	43%	**model range**	2 - 3	3 - 3	6 - 6	3 - 3	4 - 4	4 - 4	2 - 2	6 - 6	6 - 6
`haberman`	n	306	**test error**	29.2 ± 14.0%	42.5 ± 11.3%	36.9 ± 15.0%	40.9 ± 14.0%	42.7 ± 9.4%	42.7 ± 9.4%	43.1 ± 8.0%	45.3 ± 14.7%	47.5 ± 6.2%
	d	4	**train error**	29.7 ± 1.5%	40.6 ± 1.9%	41.0 ± 9.7%	45.1 ± 12.0%	40.4 ± 8.5%	40.4 ± 8.5%	34.3 ± 2.8%	46.0 ± 3.6%	5.4 ± 1.5%
	$\Pr(y=+1)$	74%	**model size**	3	2	3	1	3	3	9	3	4
	$\Pr(y=-1)$	26%	**model range**	2 - 3	2 - 2	3 - 3	1 - 1	0 - 3	1 - 3	4 - 9	3 - 3	4 - 4
`mammo`	n	961	**test error**	19.5 ± 3.0%	19.0 ± 3.1%	19.2 ± 3.0%	19.0 ± 3.1%	20.5 ± 3.3%	20.3 ± 3.5%	20.7 ± 3.9%	20.3 ± 3.0%	19.1 ± 3.1%
	d	15	**train error**	18.3 ± 0.3%	19.3 ± 0.3%	19.2 ± 0.4%	19.2 ± 0.3%	19.8 ± 0.3%	19.9 ± 0.3%	20.0 ± 0.6%	20.3 ± 0.4%	18.2 ± 0.4%
	$\Pr(y=+1)$	46%	**model size**	9	13	14	14	5	5	5	14	14
	$\Pr(y=-1)$	54%	**model range**	9 - 11	12 - 13	14 - 14	13 - 14	3 - 5	4 - 6	3 - 5	14 - 14	14 - 14
`heart`	n	303	**test error**	16.5 ± 7.8%	15.2 ± 6.3%	14.9 ± 5.9%	14.5 ± 5.9%	21.2 ± 7.5%	23.2 ± 6.8%	19.8 ± 6.5%	15.5 ± 6.5%	15.2 ± 6.0%
	d	33	**train error**	14.9 ± 1.1%	14.0 ± 1.0%	13.1 ± 0.8%	13.2 ± 0.6%	10.0 ± 1.8%	8.5 ± 2.0%	14.3 ± 0.9%	13.6 ± 0.5%	10.4 ± 0.8%
	$\Pr(y=+1)$	46%	**model size**	3	12	32	24	7	16	6	31	32
	$\Pr(y=-1)$	54%	**model range**	3 - 3	10 - 13	30 - 32	22 - 27	9 - 17	12 - 27	6 - 8	28 - 32	32 - 32
`mushroom`	n	8124	**test error**	0.0 ± 0.0%	0.0 ± 0.0%	1.7 ± 0.3%	0.0 ± 0.0%	0.0 ± 0.0%	0.0 ± 0.0%	1.2 ± 0.6%	0.0 ± 0.0%	0.0 ± 0.0%
	d	114	**train error**	0.0 ± 0.0%	0.0 ± 0.0%	1.7 ± 0.0%	0.0 ± 0.0%	0.0 ± 0.0%	0.0 ± 0.0%	1.1 ± 0.3%	0.0 ± 0.0%	0.0 ± 0.0%
	$\Pr(y=+1)$	48%	**model size**	7	21	113	108	8	9	7	98	113
	$\Pr(y=-1)$	52%	**model range**	7 - 7	19 - 23	113 - 113	106 - 108	8 - 8	9 - 9	6 - 8	98 - 108	113 - 113
`spambase`	n	4601	**test error**	6.3 ± 1.2%	10.0 ± 1.7%	26.3 ± 1.7%	10.0 ± 1.7%	6.6 ± 1.3%	7.3 ± 1.0%	11.1 ± 1.4%	7.8 ± 1.5%	13.7 ± 1.4%
	d	58	**train error**	5.7 ± 0.3%	9.5 ± 0.3%	26.1 ± 0.2%	9.6 ± 0.2%	4.2 ± 0.3%	3.9 ± 0.3%	9.8 ± 0.3%	8.1 ± 0.8%	1.3 ± 0.1%
	$\Pr(y=+1)$	39%	**model size**	34	28	57	28	29	73	7	57	57
	$\Pr(y=-1)$	61%	**model range**	28 - 40	28 - 29	57 - 57	28 - 29	23 - 31	56 - 78	6 - 10	57 - 57	57 - 57

Table 3.3: Accuracy and sparsity of all methods on all datasets. Here: *test error* refers to the 10-CV mean test error ± the 10-CV standard deviation in test error; *train error* refers to the 10-CV mean training error ± the 10-CV standard deviation in training error; model size refers to the final model size; and model range refers to the 10-CV minimum and maximum model size. The results reflect the models produced by each method when free parameters are chosen to minimize the 10-CV mean test error. We report the 10-CV weighted test and training error for `adult` and `haberman`.

sparse models for all datasets in 10 minutes using CPLEX 12.6, and a proof of optimality (i.e., a MIPGAP of 0.0%) for all models we trained for `mammo`, `mushroom`, `bankruptcy`, `breastcancer`.

On the Loss in Training Accuracy due to Discrete Coefficients

We expect that methods that directly optimize accuracy and sparsity will achieve the best possible accuracy at every level of sparsity (i.e. the best possible trade-off between accuracy and sparsity). SLIM directly optimizes accuracy and sparsity. However, it may not necessarily achieve the best possible accuracy at each level of sparsity because it restricts coefficients to a finite discrete set $\mathcal{L}$.

By comparing SLIM to Lasso, we can identify a baseline loss in training accuracy due to this $\mathcal{L}$ set restriction. In particular, we know that when Lasso's performance dominates that of SLIM, and the SLIM scoring systems are certifiably optimal, then the loss in accuracy is due to the discrete or bounded nature of $\mathcal{L}$. Our results show that this tends to happen mainly at large model sizes (see e.g., the regularization path for `breastcancer`, `heart`, `mammo`). This suggests that the $\mathcal{L}$ set restriction has a more noticeable impact on accuracy at larger model sizes.

One interesting effect of the $\mathcal{L}$ set restriction is that the most accurate SLIM scoring system may not use all of the features in the dataset. In our experiments, we fit a SLIM scoring system for $C_0 = 0.9/nd$ to obtain the model with highest training accuracy among linear models with coefficients in $\boldsymbol{\lambda} \in \mathcal{L}$. On the `bankruptcy` dataset, for example, we find that the optimal scoring system only uses 3 out of 6 features. This is due to the $\mathcal{L}$ set restriction: if the $\lambda \in \mathcal{L}$ constraint were relaxed, then the method would use all features to improve its training accuracy (as is the case with Ridge or SVM Linear).

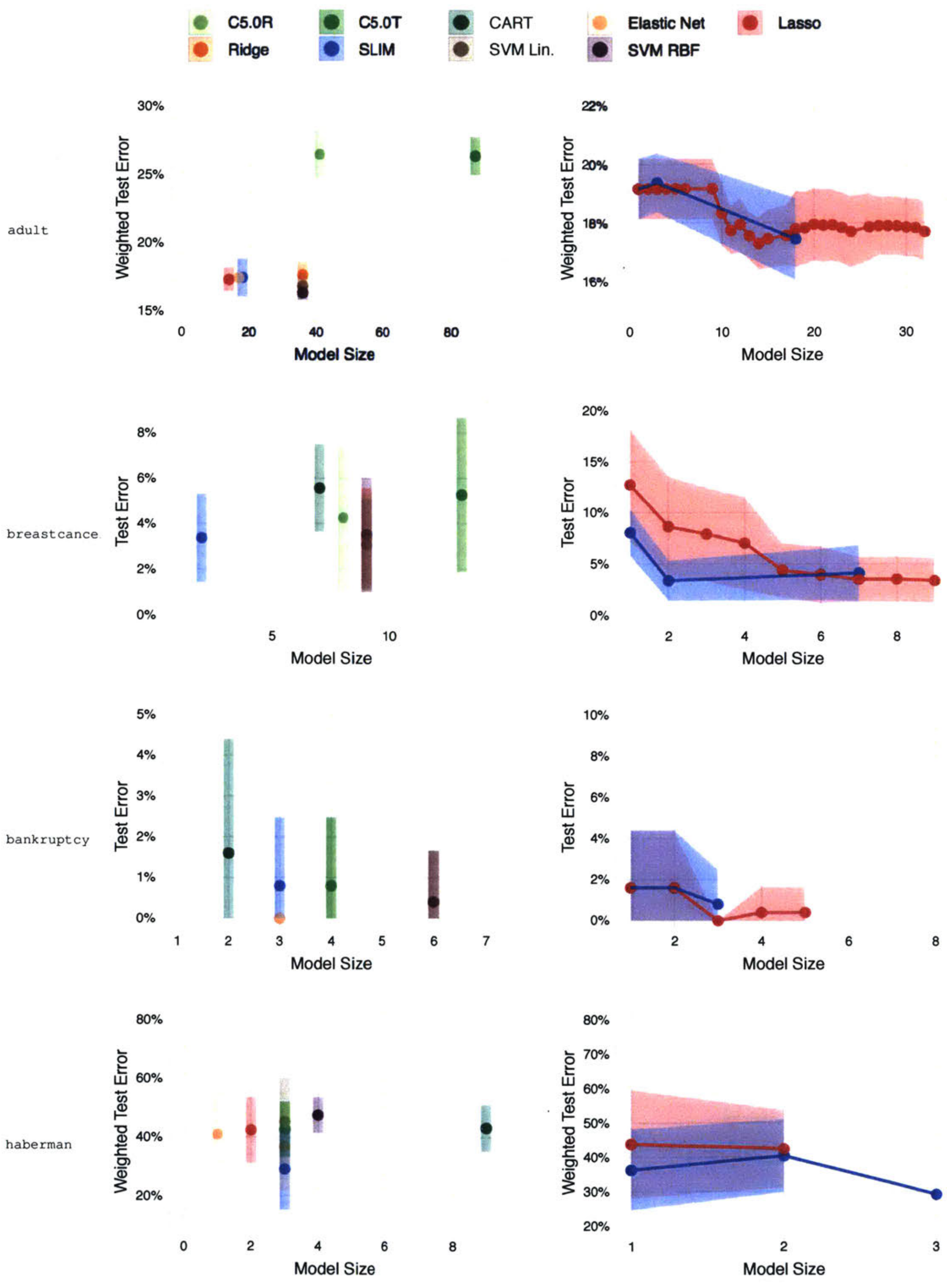


Figure 3.5: Accuracy and sparsity of all classification methods on all datasets. For each dataset, we plot the performance of models when free parameters are set to values that minimize the 10-CV mean test error (left), and plot the performance of SLIM and Lasso across the full ℓ_0-regularization path (right).

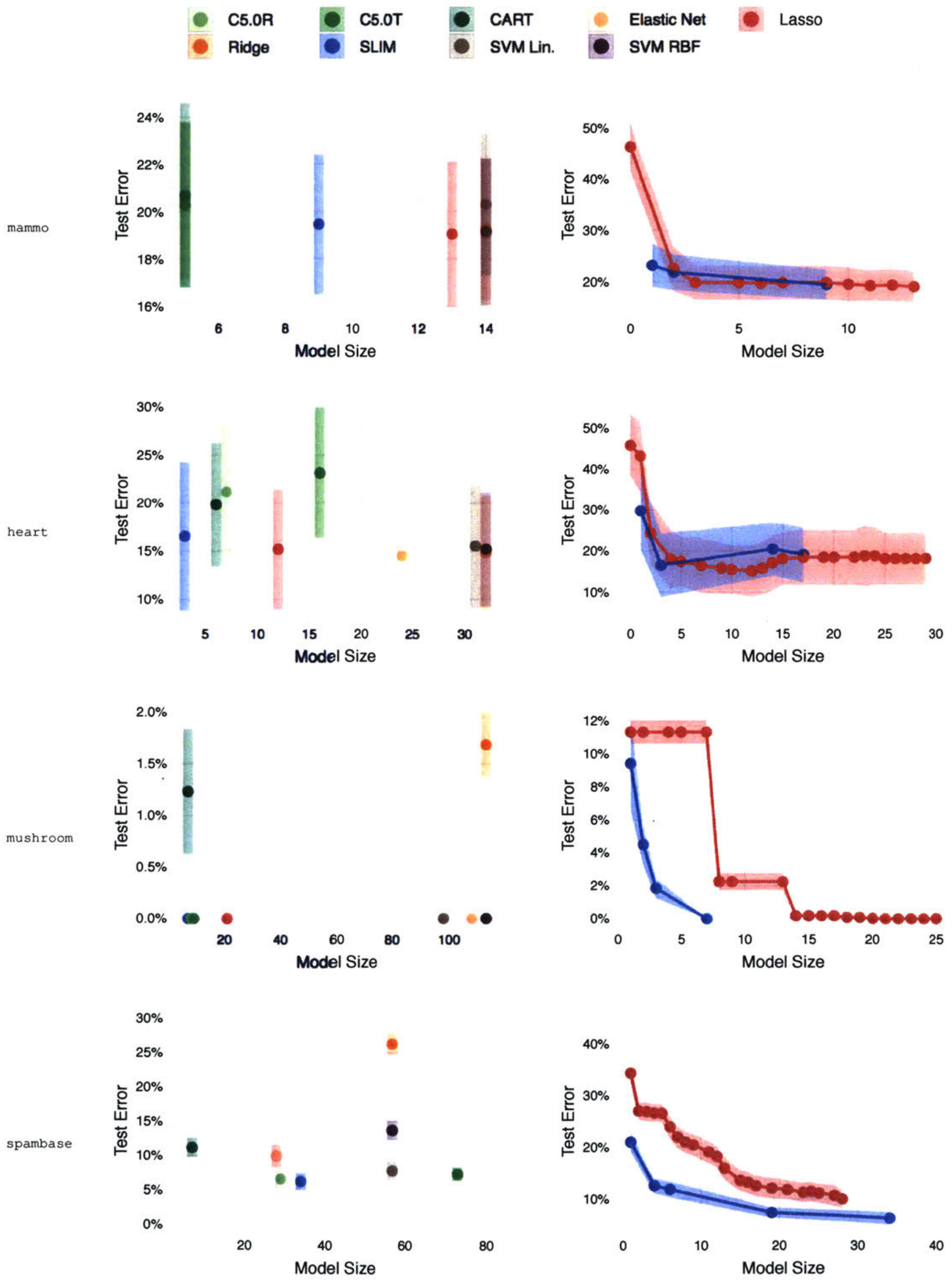


Figure 3.6: Accuracy and sparsity of all classification methods on all datasets. For each dataset, we plot the performance of models when free parameters are set to values that minimize the 10-CV mean test error (left)m and plot the performance of SLIM and Lasso across the full ℓ_0-regularization path (right).

On Interpretability

To discuss interpretability, we focus on the `mushroom` dataset, which provides a nice basis for comparison as multiple methods fit a model that attains perfect predictive accuracy. In Figures 3.7 to 3.10, we show the sparsest models that achieve perfect predictive accuracy. We omit models from some methods because they do not attain perfect accuracy (CART), or use far more features (Ridge, SVM Lin, SVM RBF).

Here, the SLIM scoring system uses 7 integer coefficients. However, it can be simplified into a 5 line scoring system since *odor=none*, *odor=almond*, and *odor=anise* are mutually exclusive variables with the same coefficient. The model lets users make predictions by hand, and uses a linear form that helps users gauge the influence of each input variable with respect to the others. Note that only some of these qualities are found in the other models. The Lasso model, for instance, has a linear form but uses far more features. In contrast, the C5.0 models let users to make predictions by hand, but have a hierarchical structure that makes it difficult to gauge the influence of each input variable with respect to the others.

PREDICT MUSHROOM IS POISONOUS IF SCORE > 3

1.	$spore\_print\_color = green$	4 points		$\cdots$
2.	$stalk\_surface\_above\_ring = grooves$	2 points	$+$	$\cdots$
3.	$population = clustered$	2 points	$+$	$\cdots$
4.	$gill\_size = broad$	-2 points	$+$	$\cdots$
5.	$odor \in \{none, almond, anise\}$	-4 points	$+$	$\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$	$\cdots$

Figure 3.7: SLIM scoring system for `mushroom` (10-CV mean test error of $0.0 \pm 0.0\%$).

10.86 *spore_print_color = green*	$+$	4.49 *gill_size= narrow*	$+$	4.29 *odor = foul*
$+$2.73 *stalk_surface_below_ring=scaly*	$+$	2.60 *stalk_surface_above_ring = grooves*	$+$	2.38 *population = clustered*
$+$0.85 *spore_print_color = white*	$+$	0.44 *stalk_root = bulbous*	$+$	0.43 *gill_spacing = close*
$+$0.38 *cap_color = white*	$+$	0.01 *stalk_color_below_ring = yellow*	$-$	8.61 *odor = anise*
$-$8.61 *odor = almond*	$-$	8.51 *odor = none*	$-$	0.53 *cap_surface = fibrous*
$-$0.25 *population = solitary*	$-$	0.21 *stalk_surface_below_ring = fibrous*	$-$	0.09 *spore_print_color = brown*
$-$0.00 *cap_shape = convex*	$-$	0.00 *gill_spacing = crowded*	$-$	0.00 *gill_size = broad*
$+$0.25				

Figure 3.8: Lasso score function for `mushroom` (10-CV mean test error of $0.0 \pm 0.0\%$).

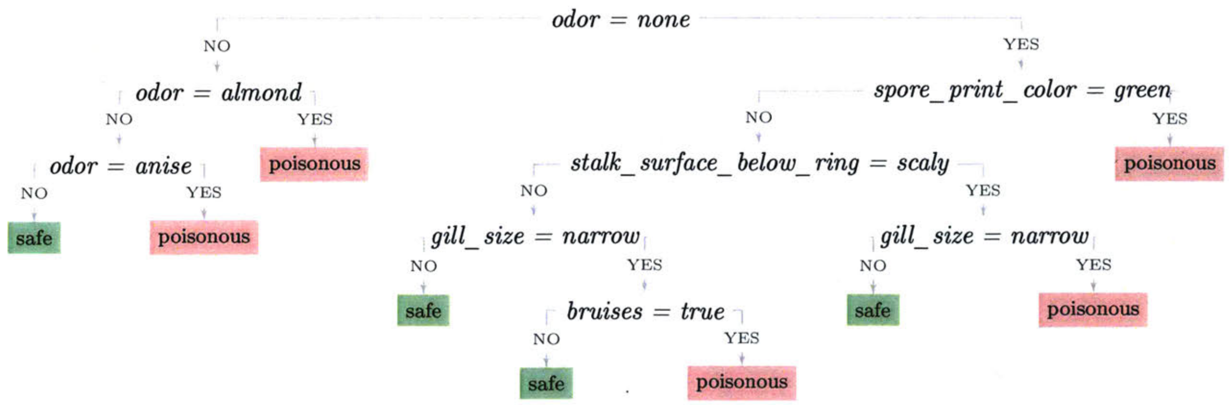


Figure 3.9: C5.0 decision tree for `mushroom` (10-CV mean test error of $0.0 \pm 0.0\%$).

Rule			**Confidence**	**Support**	**Lift**
$odor = none \wedge gill\_size \neq narrow \wedge spore\_print\_color \neq green$	$\Longrightarrow$	safe	1.00	3216	1.9
$bruises = false \wedge odor = none \wedge stalk\_surface\_below\_ring \neq scaly$	$\Longrightarrow$	safe	0.999	1440	1.9
$odor \in anise, almond$	$\Longrightarrow$	safe	0.998	400	1.9
$odor = anise$	$\Longrightarrow$	safe	0.998	400	1.9
$odor \neq almond \wedge odor \neq anise \wedge odor \neq none$	$\Longrightarrow$	poisonous	1.000	3796	2.1
$spore\_print\_color = green$	$\Longrightarrow$	poisonous	0.986	72	2.9
$gill\_size = narrow \wedge stalk\_surface\_below\_ring = scaly$	$\Longrightarrow$	poisonous	0.976	40	2.0

Figure 3.10: C5.0 rule list for `mushroom` (10-CV mean test error of $0.0 \pm 0.0\%$).

3.7 Discussion

In this chapter, we presented a machine learning approach to create scoring systems for decision-making, called a Supersparse Linear Integer Model (SLIM). Our approach is designed to build scoring systems that are fully optimized for accuracy, sparsity, small integer coefficients, and operational constraints. In contrast to traditional machine learning approaches, our approach requires the solution to an computationally challenging discrete optimization problem as it uses *exact* quantities on model fit and model form. We discussed how to solve this problem using mixed-integer programming (MIP), and presented techniques to recover certifiably optimal solutions that can be paired effectively with a MIP solver.

Our results in Section 3.6 show that we can build data-driven scoring systems for a large class of real-world datasets, and recover a certificate of optimality for small to mid-sized instances. These models perform well in comparison to state-of-the-art models, but are easier to use, understand and validate. In particular, SLIM scoring systems outperform popular sparse linear classification methods at small model sizes. This is due to the fact that we avoid the use of surrogate functions for accuracy and sparsity, can recover certifiably optimal solutions, and are may not incur a loss in training accuracy due to use of small integer coefficients (i.e. which is guaranteed for datasets with binary features as per Theorem 3.26). In Chapters 4 and 5, we show that the performance of these models is far superior in the presence of operational constraints. In addition, we will discuss other benefits that are difficult to benchmark, such as the ability to produce models that can be validated, that can be customized without parameter tuning, and that have a meaningful certificate of optimality.

The fact that we can recover certifiably optimal solutions for a large class of real-world problems is surprising as other MIP-based approaches typically produce models with large optimality gaps for this problem. Although one would have to carry out further experiments to identify why this is the case, some of it is likely due to the nature of the scoring system problem and the way in we have been able to exploit it. In contrast to other MIP-based approaches for 0-1 loss minimization (Brooks, 2011; Nguyen and Sanner, 2013), for example, the coefficients in our problem are restricted to a finite discrete set and we can use this to produce tight Big-M constants for the 0-1 loss (which lead to tighter relaxations, and more effective B&B). In addition, elements of the scoring system problem that may deteriorate the worst-case computational complexity can result in more effective improved B&B search. By using integer coefficients, for example, a MIP solver may search more effectively (e.g. it can generate cutting-planes to discard non-integer solutions, such as Gomory cuts; further it can discard parts of the feasible region by splitting nodes on the most fractional integer). These are meant to illustrate some potential differences between our approach and others in the literature which merit further study.

Potential Improvements

The performance of the scoring systems in our experiments primarily reflects the effect of using a tighter MIP formulation with the CPLEX MATLAB API, which does not support callback functions that allow us to intervene in the B&B search. In light of this, we note that we could further improve by using the techniques in this chapter and from Chapter 6 more effectively. In particular, a revised approach for SLIM would involve the following steps:

1. Find a good initial solution by rounding the solutions from a linear SVM (if the data is real), or via exhaustive search using low-dimensional threshold gates (if the data is binary).

2. Apply the data reduction procedure (Algorithm 3.4.2) to identify redundant examples in the training data.

3. Use a callback function to run the chained updates procedure (Chapter 6.3.2) to restrict the search region, and strengthen the LP relaxation.

4. Use a callback function to run the polish-and-cover procedure (Algorithm 2) to polish incumbent updates and add cover-cuts.

There are also other techniques in the literature that could be adapted to our setting. In particular, we could devise node selection rules in order to branch more effectively on the 0–1 loss indicators (by adapting the branching technique from Nguyen and Sanner, 2013). Similarly, one could also devise rules to branch more effectively on the ℓ_0-norm indicators (e.g., using the feature selection technique of Somol et al., 2004). Adapting these techniques would require callback functions that are only available in commercial solvers, but may further reduce computation if they can be implemented without introducing additional overhead.

Chapter 4

Sleep Apnea Screening

In this chapter, we present results a collaboration with the Massachusetts General Hospital Sleep Laboratory where we used SLIM to create a scoring system for sleep apnea screening. Our work illustrates how our approach can handle operational constraints on accuracy and model form without parameter tuning, and highlights the performance benefits of optimizing an exact objective under such constraints.

Notes

This chapter draws on material from Ustun and Rudin (2016b) and Ustun et al. (2016). An editorial regarding the clinical significance of the work from this chapter can be found in Combs et al. (2016).

4.1 Background

Obstructive sleep apnea (OSA) is a sleep disorder where a patient intermittently starts and stops breathing during sleep. The condition is a treatable contributor to morbidity and mortality, leading to decreased work performance and quality of life (AlGhanim et al., 2008). Recent studies estimate that OSA affects over 12 million people in the United States along (Kapur, 2010). In spite of this, a large number of patients with OSA remain undiagnosed (Collop et al., 2007).

The vast majority of OSA diagnoses originate from physician suspicion or patient-reported symptoms. Expert clinical impression has weak sensitivity and specificity (<70% each) for predicting OSA (Skomro and Kryger, 1999). This pathway of clinical suspicion may itself be a potential reason for under-recognition, as classic symptoms OSA are not strongly predictive of the presence of OSA. For example, the Epworth Sleepiness Scale (ESS) carries minimal predictive value for OSA (Gottlieb et al., 1999), or even objective measures of sleepiness (Chervin and Aldrich, 1999).

Although recent predictive models have shown improvements over the ESS (Abrishami et al., 2010), the best validated model – the STOP-BANG tool of Chung F (2008) – has important limitations when for screening applications. This is partially due to performance (i.e., STOP-BANG has a sensitivity of 83.6% and a specificity of 56.4%), but also due to its reliance on symptom-based features (e.g. snoring, nocturnal gasping, and sleepiness) which are hard to assess reliably in a clinical setting.

4.2 Problem Description

4.2.1 Data

The data for this study were collected from patients at the Massachusetts General Hospital Sleep Laboratory. The training dataset covers a cohort of $n = 1922$ patients and contains $d = 33$ features for each patient. Here, $y_i = +1$ if patient i has obstructive sleep apnea (OSA). There is significant class imbalance as $\Pr(y_i = +1) = 76.9\%$. The features include: (i) standard medical information on demographics and comorbidity (e.g. *Age*, *Male*, *Hypertension*, *Diabetes*); (ii) self-reported information related to sleep habits (e.g. *CaffeineConsumption*, *WakesUpAtNight*).

4.2.2 Model Requirements

Our collaborators specified three operational constraints to ensure that the scoring system we produced would be used and accepted by physicians:

1. *Limited FPR*: The model had to achieve the highest possible true positive rate (TPR) while maintaining a maximum false positive rate (FPR) of 20%. This would ensure that the model could diagnose as many cases of sleep apnea as possible but limit the number of faulty diagnoses.

2. *Limited Model Size*: The model had to be transparent and use at most 5 features. This would ensure that the model was could be explained and understood by other

physicians in a short period of time.

3. *Monotonicity*: The model had to obey established relationships between well-known risk factors and the incidence of sleep apnea (e.g. it could not suggest that a patient with hypertension had a lower risk of sleep apnea since hypertension is a positive risk factor for sleep apnea).

4.2.3 Training Setup

We trained a SLIM scoring system with integer coefficients $\lambda_j \in \{-10, \ldots, 10\}$. We addressed all operational constraints without parameter tuning, as follows:

- We added a loss constraint using the loss variables to limit the maximum FPR at 20%. We then set $w^+ = n^-/(1+n^-)$ to guarantee that the optimization would yield a classifier with the highest possible TPR with a maximum FPR less than 20% (see Section 3.3).
- We added a feature-based constraint using the loss variables to limit the maximum number of features to 5 (see Section 3.3.4). We then set the trade-off parameter to $C_0 = 0.9w^-/nd$ so that the optimization would yield a classifier that did not sacrifice accuracy for sparsity (see Remark 3.5).
- We added sign constraints to the coefficients to ensure that our model would not violate established relationships between features and the predicted outcome (i.e., we set $\lambda_j \geq 0$ if there had to be a positive relationship, and $\lambda_j \leq 0$ if there had to be a negative relationship).

With this setup, we trained 10 models with subsets of the data to assess predictive accuracy via 10-fold cross validation (10-CV), and 1 final model with all of data to hand over to our collaborators. We used the IP formulation in (3.5) and solved each IP for 1 hour, in parallel, on 12-core 2.7GHz machine with 48GB RAM using CPLEX 12.6.0. Thus, the entire training process for SLIM required 1 hour of computing time.

As a comparison, we also trained models with 8 baseline classification methods shown in Table 4.1. We dealt with the class imbalance by using a cost-sensitive approach, where we used a weighted loss function and varied its sensitivity parameter w^+ across a large range of possible values. When possible, we addressed the remaining operational constraints by searching over a fine grid of free parameters.

Model selection was difficult for baseline methods because they could not accomodate operational constraints in the same way as SLIM. For each baseline method, we chose the best model that satisfied all operational constraints by: (i) dropping any instance of the free parameters where operational constraints were violated; (ii) choosing the instance that maximized the 10-CV mean test TPR. We ruled that an instance of the free parameters violated an operational constraint if any of the following conditions were met:

- the 10-CV mean test FPR of the model produced with the instance was greater than the 10-CV mean test FPR of the SLIM model (20.9%)

- the model size[1] of the final model produced with the instance was greater than 5;
- the final model produced did not obey sign constraints. Note that this model selection procedure may have biased the results in favor of the baseline methods because we mixed testing and training data by looking at the final model to ensure that operational constraints were satisfied.

Method	**Controls**	**# Instances**	**Settings and Free Parameters**
CART	Max FPR Model Size	39	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$
C5.0T	Max FPR	39	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$
C5.0R	Max FPR Model Size	39	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$
Lasso	Max FPR Model Size Signs	39000	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$ $\times$ 1000 values of λ chosen by **glmnet**
Ridge	Max FPR Signs	39000	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$ $\times$ 1000 values of λ chosen by **glmnet**
Elastic Net	Max FPR Model Size Signs	975000	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$ $\times$ 1000 values of λ chosen by **glmnet** $\times$ 19 values of $\alpha \in \{0.05, 0.10, \ldots, 0.95\}$
SVM Lin.	Max FPR	975	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$ $\times$ 25 values of $C \in \{10^{-3}, 10^{-2.75}, \ldots, 10^3\}$
SVM RBF	Max FPR	975	39 values of $w^+ \in \{0.025, 0.05, \ldots, 0.975\}$ $\times$ 25 values of $C \in \{10^{-3}, 10^{-2.75}, \ldots, 10^3\}$
SLIM	Max FPR Model Size Signs	1	$w^+ = 2n^-/(1+n^-)$ $C_0 = 0.9w^-/dn$, $\lambda_0 \in \{-100, \ldots, 100\}$ $\lambda_j \in \{-10, \ldots, 10\}$

Table 4.1: Methods, software, and free parameters used to build screening models for obstructive sleep apnea. An instance is a unique combination of free parameters. Controls refer to operational constraints that we expect each method to handle.

[1]Model size represents the number of coefficients for linear models (Lasso, Ridge, Elastic Net, SLIM, SVM Linear), the number of leaves for decision trees (C5.0T, CART), and the number of rules for rule-based lists (C5.0R). For completeness, we set the model size for black-box models (SVM RBF) to the number of features d.

4.3 Results

We show the performance of the best model from each method in Table 4.2, and summarize the operational constraints they satisfied in Table 4.3. In what follows, we report our observations related to operational constraints, predictive performance and interpretability.

		OBJECTIVE	CONSTRAINTS		OTHER INFO.	
Method	**Constraints Satisfied**	**Test TPR**	**Test FPR**	**Final Model Size**	**Final Train TPR**	**Final Train FPR**
SLIM	All	61.4% 55.5 - 68.8%	20.9% 15.0 - 30.4%	5 -	62.0% -	19.6% -
Lasso	All	29.3% 19.2 - 60.0%	8.6% 0.0 - 33.3%	3 -	22.1% -	3.8% -
Elastic Net	All	44.2% 0.0 - 64.1%	18.8% 0.0 - 37.0%	3 -	54.3% -	20.7% -
Ridge	Max FPR	66.0% 60.5 - 68.5%	20.6% 8.6 - 32.6%	30 -	66.0% -	18.9% -
SVM RBF	Max FPR	64.3% 59.2 - 71.1%	20.8% 10.0 - 30.4%	33 -	67.8% -	12.4% -
SVM Linear	Max FPR	62.7% 57.9 - 69.0%	19.8% 7.5 - 28.6%	31 -	63.1% -	17.1% -
C5.0R	None	84.0% 78.9 - 87.7%	43.0% 32.6 - 54.2%	26 -	85.5% -	32.9% -
C5.0T	None	81.3% 77.4 - 84.8%	42.9% 29.6 - 62.5%	39 -	84.5% -	28.4% -
CART	None	93.0% 88.8 - 96.1%	70.4% 61.1 - 83.3%	8 -	95.9% -	73.9% -

Table 4.2: TPR, FPR and model size for all methods. We report the 10-CV mean TPR and FPR. The ranges in each cell represent the 10-CV minimum and maximum.

4.3.1 On the Difficulties of Handling Operational Constraints

Among the 9 classification methods that we used, only SLIM, Lasso and Elastic Net could produce a model that satisfied all of operational constraints given to us by physicians. Tree and rule-based methods such as CART, C5.0 Tree and C5.0 Rule were unable to produce a model with a maximum FPR of 20% (see Figure 4.1). Methods that used ℓ_2-regularization such as Ridge, SVM Lin. and SVM RBF were unable to produce a model with the required level of sparsity.

We did not expect all methods to satisfy all of the operational constraints. However, we included them to emphasize the following important points. Namely, popular methods for applied predictive modeling do not:

- Handle simple operational constraints that are crucial for models to be used and accepted. Implementations of popular classification methods do not have a mechanism to adjust important model qualities. That is, there is no mechanism to control sparsity in C5.0T (Kuhn et al. 2012) and no mechanism to incorporate sign constraints in SVM (Meyer et al. 2012). Finding a method with suitable controls is especially difficult when a model has to satisfy multiple operational constraints.
- Have controls that are easy-to-use and/or work correctly. When a method has suitable controls to handle operational constraints, producing a model often requires a tuning process over a high-dimensional free parameter grid. Even after extensive tuning, however, it is possible that we may never obtain a model that satisfies all operational constraints (e.g. CART, C5.0R, C5.0T for the Max FPR constraint as shown in Figure 4.1).
- Allow tuning to be portable when the training set changes. Consider a standard model selection procedure where we choose free parameters to maximize predictive accuracy. In this case, we would train models on several folds for each instance of the free parameters, choose an instance of the free parameters that maximized our estimate of predictive accuracy among the instances that met all operational constraints, and then train a final model using these values of the free parameters. Unfortunately, there is no guarantee that the final model will obey all operational constraints.

Method	% of Instances that Satisfied		
	Max FPR	Max FPR Model Size	Max FPR Model Size Signs
SLIM	100.0%	100.0%	100.0%
Lasso	19.6%	4.8%	4.8%
Elastic Net	18.3%	1.0%	1.0%
Ridge	20.9%	0.0%	0.0%
SVM Linear	18.7%	0.0%	0.0%
SVM RBF	15.8%	0.0%	0.0%
C5.0R	0.0%	0.0%	0.0%
C5.0T	0.0%	0.0%	0.0%
CART	0.0%	0.0%	0.0%

Table 4.3: Percentage of instances that satisfied various combinations of operational constraints. Each instance is a unique combination of free parameters for a given method.

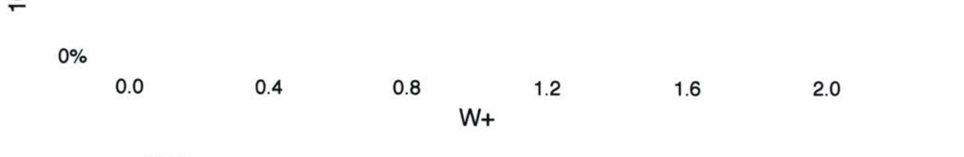


Figure 4.1: 10-CV mean test FPR for CART, C5.0R and C5.0T models across the full range of the misclassification cost for positive examples w^+. These methods cannot produce a model that satisfies the max FPR $\leq 20\%$ constraint.

4.3.2 On Performance Trade-offs under Operational Constraints

Among the three methods that produced acceptable models, the scoring system produced by SLIM had significantly higher sensitivity than the models produced by Lasso and Elastic Net – a result that we expected given that SLIM minimizes the 0–1 loss and an ℓ_0-penalty while Lasso and Elastic Net minimize convex surrogates of these quantities.

This result held true even when we relaxed various operational constraints. In Figure 4.2, for instance, we plot the sensitivity and sparsity of models that satisfied the max FPR and sign constraints. Here, we see that Lasso and Elastic Net need at least 8 coefficients to produce a model with the same degree of sensitivity as SLIM. In Figure 4.3, we plot the TPR and FPR of models that satisfied the sign and model size constraints. As shown, SLIM scoring systems dominate Lasso and Elastic Net models across the entire ROC curve.

These sensitivity advantages are also evident in Table 4.2. Here, SLIM yields a model with a similar level of sensitivity and specificity as Ridge and SVM Linear even as it is fitting models from a far smaller hypothesis space.

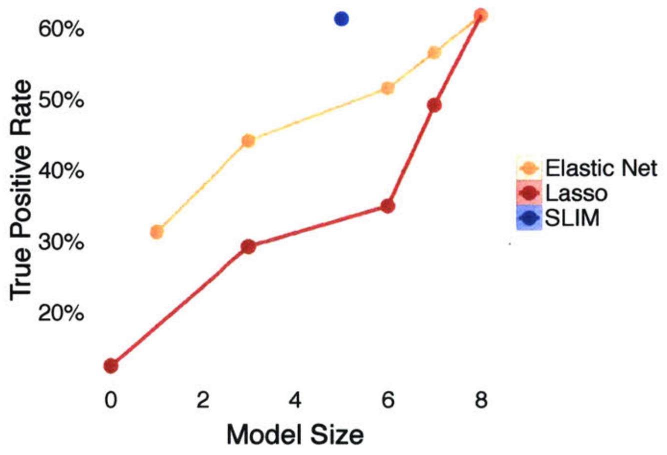


Figure 4.2: Sensitivity and model size of Lasso and Elastic Net models that satisfy the sign and FPR constraints. For each method, we plot the instance that attains the highest 10-CV mean test TPR at model sizes between 0 and 8. Lasso and Elastic Net need at least 8 variables to attain the same sensitivity as SLIM.

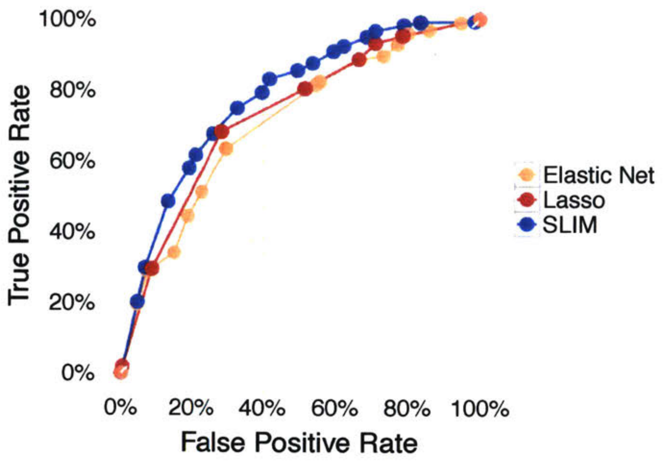


Figure 4.3: ROC curve for SLIM, Lasso and Elastic Net models that satisfy the sign and model size constraints. For each method, we plot the instance that attains the highest 10-CV mean test TPR for 10-CV mean FPR values of $5\%, 10\%, \ldots, 95\%$. Note that we had to train 19 additional instances of SLIM to create this plot.

4.3.3 On the Usability and Interpretability of Feasible Models

We compare the best models from the baseline methods that satisfied all operational constraints in Figure 4.4, and present the SLIM scoring system in Figure 4.5.

Lasso	$Score = 0.13\ Snoring$	$+\ 0.12\ Hypertension$	$-\ 0.26\ Female$	$-\ 0.17$
Elastic Net	$Score = 0.03\ Snoring$	$+\ 0.02\ Hypertension$	$-\ 0.09\ Female$	$-\ 0.02$

Figure 4.4: Score functions of the most sensitive models that satisfy all operational constraints. The baseline models have very poor sensitivity as shown in Table 4.2.

In this case, our collaborators found that all three models were aligned with domain knowledge as they obeyed sign constraints and had large coefficients for well-known risk factors such as *BMI*, *Female*, *Age*, *Snoring* and/or *Hypertension*. Unfortunately, the Lasso and Elastic Net models could not be deployed as screening tools due to their poor sensitivity (29.3% for Lasso and 44.2% for Elastic Net). This was not the case for the SLIM model, which had much higher sensitivity (i.e. 61.4%).

Our results highlighted some of benefits of sparse linear models with small integer coefficients. Specifically, our collaborators were able to understand how the model worked by making quick predictions on prototypical examples. Using this process, they were list the exact conditions when the score exceeds the threshold and obtain a simple rule-based explanation for when the model predicted that a patient has OSA:

If *Male*, predict OSA if at least 1 of $\{BMI \geq 30,\ Age \geq 60,\ Hypertension\}$
If *Female*, predict OSA if at least 2 of $\{BMI \geq 40,\ Age \geq 60,\ Hypertension\}$

PREDICT PATIENT HAS OBSTRUCTIVE SLEEP APNEA IF SCORE > 1

1.	*Age* $\geq$ *60*	4 points		$\cdots$
2.	*Hypertension*	4 points	$+$	$\cdots$
3.	*BMI* $\geq$ *30*	2 points	$+$	$\cdots$
4.	*BMI* $\geq$ *40*	2 points	$+$	$\cdots$
5.	*Female*	-6 points	$+$	$\cdots$
	ADD POINTS FROM ROWS 1 – 5	**SCORE**	$=$	$\cdots$

Figure 4.5: SLIM scoring system for sleep apnea screening. This model achieves a 10-CV mean test TPR/FPR of 61.4/20.9%, obeys all operational constraints, is certifiably optimal, and was trained without parameter tuning. It also generalizes well due to the simplicity of the hypothesis space (training TPR/FPR of the final model is 62.0/19.6%).

4.4 Discussion

The scoring system in Figure 4.5 is noteworthy in that it performs well, is certifiably optimal, was built without parameter tuning, and can be validated by inspection.

The results from this chapter may explain why so many medical scoring systems are not built in a way that is entirely data-driven. Software tools for modern machine learning methods are developed under the assumption that practitioners can address operational constraints by incorporating auxiliary techniques into their training pipeline. In turn, an approach that is entirely data-driven requires a training pipeline that combines suitable methods, tunes the necessary free parameters, and performs nested cross-validation. As our results show, there is no guarantee that such a pipeline will produce an acceptable model. In such cases, it may be easier to address certain model requirements using expert judgement, or even have a panel of experts craft the entire model by hand.

In this case, the operational constraints of our collaborators were relatively mild compared to the constraints of other medical scoring systems (see e.g. Section ??). However, we were unable to address these requirements using modern machine learning methods. In most cases, the software for these methods did not contain built-in controls to directly address these requirements. When such controls did exist, they introduced free parameters, which could to be tuned. Ultimately, only two methods were able to handle these constraints (e.g. Lasso and Elastic Net), and produced models that performed poorly. Further, the resulting trade-offs then make it appear as if it were impossible to learn a screening tool that performs well while satisfying the constraints required for deployment.

Clinical Significance

One of the key insights from the scoring system in Figure 4.5 was it did not use any of the symptom-based features in the dataset. This result was important because it meant that our model could screen effectively using features that can be derived through electronic health records (i.e. without the need for information related to sleep habits or symptoms). It made sense to our collaborators, as symptoms are self-reported, noisy, and less valuable for screening. However, it was unexpected as even sparse Lasso and Elastic Net models used symptom-based features (see Figure 4.4). Since that our model was optimized for exact feature selection, however, this suggested that we could drop self-reported symptoms entirely.

We investigated the value of self-reported symptoms in predicting obstructive sleep apnea in Ustun et al. (2016). In Figure 4.6, we illustrate the value of this information by showing the ROC curves for SLIM scoring systems built using three different sets of features: all features; only features for patient-reported symptoms; and features that could be extracted from an electronic health record. In this case, the finding held across every machine learning method that we used: models that used only patient-reported symptoms performed poorly, whereas models that used only features from electronic health records performed almost as well (often as well) as models that used both sets of information.

The clinical significance of this finding is further discussed in the editorial of Combs et al. (2016). We note that it not have been possible had we not built a model that our collaborators could understand, and that had selected an optimal subset of features for screening.

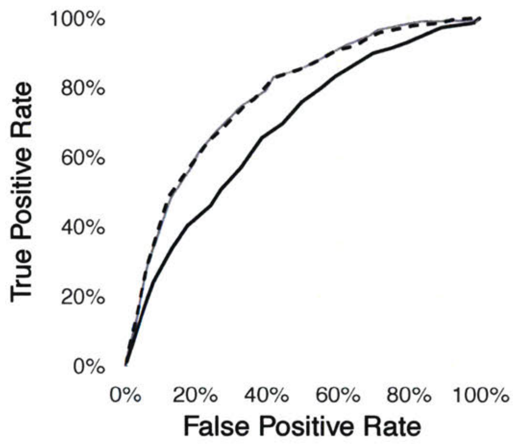


Figure 4.6: Decision points of SLIM models over the full ROC curve for: (i) all features (gray); (ii) features that can be extracted from an electronic health record (dashed); (iii) features related to patient-reported symptoms (black).

Chapter 5

Recidivism Prediction

In this chapter, we use SLIM to build scoring systems for a collection of recidivism prediction problems. Our results highlight the performance of the IP formulation in and polishing techniques on a large dataset. We use these models to address two key questions in the criminal justice community, such as whether there is a trade-off between accuracy and simplicity, and whether we should be using predictive models for recidivism prediction.

Notes

The material from this chapter is drawn from Zeng et al. (2016). Code to process the ICPSR data used in this chapter can be found at https://github.com/ustunb/recidivism-prediction. Code for the experimental comparisons can be found at http://github.com/ustunb/classification-pipeline.

5.1 Background

Forecasting has been used for criminal justice applications since the 1920s (Borden, 1928; Burgess, 1928) when various factors derived from age, race, prior offense history, employment, grades, and neighborhood background were used to predict parole violation.

Since then the use of *recidivism prediction instruments* has grown substantially, and several U.S. states now mandated the use of predictive models for sentencing decisions (Wroblewski, 2014). Outside of the U.S., other countries that currently use recidivism prediction instruments include Canada (Hanson and Thornton, 2003), the Netherlands (Tollenaar and van der Heijden, 2013), and the U.K. (Howard et al., 2009).

Modern applications of recidivism prediction models include evidence-based sentencing (Hoffman, 1994), corrections and prison administration (Belfrage et al., 2000), informing release on parole (Pew Center of the States, Public Safety Performance Project Washington, 2011), determining the level of supervision during parole (Barnes and Hyatt, 2012; Ritter, 2013), determining appropriate sanctions for parole violations (Turner et al., 2009), and targeted policy interventions (Lowenkamp and Latessa, 2004).

5.2 Problem Description

We consider a total of 6 recidivism prediction problems. Each problem is a binary classification problem with $n = 33,796$ prisoners and $d = 48$ input variables. The goal is to predict whether a prisoner will be arrested for a certain type of crime within 3 years of being released from prison.

5.2.1 Data and Sample Description

We defined recidivism problems using raw data from the study "Recidivism of Prisoners Released in 1994" (U.S. Department of Justice, Bureau of Justice Statistics, 2014). The data from this study is currently the largest publicly available database on recidivism in the United States. It tracks a sample of 38,624 prisoners for 3 years following their release from prison in 1994. These prisoners are representative of the population of all prisoners released from 15 states, and account for over 65% of all prisoners released from prison in 1994.

The raw data contains 6,427 features for each of the 38,624 prisoners. The information was sourced from record-of-arrest-and-prosecution sheets maintained by state law enforcement agencies and the FBI. The 6,427 features consist of 91 fields recorded before or during release from prison in 1994 (e.g., date of birth, effective sentence length), and 64 fields that were recorded for up to 99 different arrests in the 3 year follow-up period.

In order to exclude extraordinary or unrepresentative release cases, we restricted our analysis to a subsample of 33,796 prisoners as defined by the U.S. Bureau of

Justice Statistics (Langan and Levin, 2002). Explicitly, our analysis only includes prisoners who were alive over the entire 3 year follow-up period, and were released from prison in 1994 after serving a sentence of at least 1 year. Prisoners with certain release types – release to custody/detainer/warrant, absent without leave, escape, transfer, administrative release, and release on appeal – were excluded.

5.2.2 Recidivism Prediction Problems

We created 6 recidivism prediction problems as follows. We defined binary outcome variables $y_i \in \{-1, +1\}$ where $y_i = +1$ if a prisoner is arrested for a particular type of crime within 3 years after being released from prison (see Table 5.2). We considered the following types of crime: an arrest for any crime (`arrest`); an arrest for a drug-related offense (`drug`); or an arrest for a certain type of violent offense (`general_violence`, `domestic_violence`, `sexual_violence`, `fatal_violence`).

We paired each outcome with the same set of $d = 48$ input variables shown in Table 5.1. We selected variables that: (i) could be easily obtained by law enforcement officials; (ii) were not directly related to socioeconomic factors (e.g. race), as this would rule out the potential to use these tools in applications such as sentencing. The final set of variables represent well-known predictors of recidivism (Bushway and Piehl, 2007; Crow, 2008) that have been used in recidivism prediction tools 1928 (see e.g., Borden, 1928; Maxfield et al., 2005; Berk et al., 2006; Baradaran, 2013).

The variables in our problems were all binary variables. There were two reasons for this. Binary variables produced scoring systems that were easier to use and understand as the scores could be computed without multiplication (this is the standard for many tools e.g., Pennsylvania Bulletin, 2017). Binarizing the input variables was useful for SLIM as it let us use the tighter IP formulation in (3.7a). Since the effect of binarization on the performance of other methods was not clear, we also looked at whether other methods would perform better if they used continuous variables or both continuous and binary variables (see Appendix E of Zeng et al., 2016). Specifically, we examined the change in predictive accuracy if continuous variables are used and showed that the changes in performance are minor for most methods (with the following exceptions: CART and C5.0T experienced an improvement of 4.6% for `drug` and SVM RBF experienced a 7.7% improvement for `fatal_violence`).

Input Variable	$\Pr(x_{ij}=1)$	Conditions required for $x_{ij}=1$
female	0.06	prisoner i is female
prior_alcohol_abuse	0.20	prisoner i has history of alcohol abuse
prior_drug_abuse	0.16	prisoner i has history of drug abuse
age_at_release≤17	0.00	prisoner i was ≤17 years old at release in 1994
age_at_release_18_to_24	0.19	prisoner i was 18-24 years old at release in 1994
age_at_release_25_to_29	0.21	prisoner i was 25-29 years old at release in 1994
age_at_release_30_to_39	0.38	prisoner i was 30-39 years old at release in 1994
age_at_release≥40	0.21	prisoner i was ≥40 years old at release in 1994
released_unconditional	0.11	prisoner i released at expiration of sentence
released_conditional	0.87	prisoner i released by parole or probation
time_served≤6mo	0.23	prisoner i served ≤6 months
time_served_7_to_12mo	0.20	prisoner i served 7–12 months
time_served_13_to_24mo	0.23	prisoner i served 13–24 months
time_served_25_to_60mo	0.25	prisoner i served 25–60 months
time_served≥61mo	0.10	prisoner i served ≥61 months
infraction_in_prison	0.24	prisoner i has a record of misconduct in prison
age_1st_arrest≤17	0.14	prisoner i was ≤17 years old at 1st arrest
age_1st_arrest_18_to_24	0.61	prisoner i was 18-24 years old at 1st arrest
age_1st_arrest_25_to_29	0.10	prisoner i was 25-29 years old at 1st arrest
age_1st_arrest_30_to_39	0.09	prisoner i was 30-39 years old at 1st arrest
age_1st_arrest≥40	0.04	prisoner i was ≥40 years at 1st arrest
age_1st_confinement≤17	0.03	prisoner i was ≤17 years old at 1st confinement
age_1st_confinement_18_to_24	0.46	prisoner i was 18-24 years old at 1st confinement
age_1st_confinement_25_to_29	0.18	prisoner i was 25-29 years old at 1st confinement
age_1st_confinement_30_to_39	0.21	prisoner i was 30-39 years old at 1st confinement
age_1st_confinement≥40	0.12	prisoner i was ≥40 years at 1st confinement
prior_arrest_for_drug	0.47	prisoner i was once arrested for drug offense
prior_arrest_for_property	0.67	prisoner i was once arrested for property offense
prior_arrest_for_public_order	0.62	prisoner i was once arrested for public order offense
prior_arrest_for_general_violence	0.52	prisoner i was once arrested for general violence
prior_arrest_for_domestic_violence	0.04	prisoner i was once arrested for domestic violence
prior_arrest_for_sexual_violence	0.03	prisoner i was once arrested for sexual violence
prior_arrest_for_fatal_violence	0.01	prisoner i was once arrested for fatal violence
prior_arrest_for_multiple_types	0.77	prisoner i was once arrested for multiple types of crime
prior_arrest_for_felony	0.84	prisoner i was once arrested for a felony
prior_arrest_for_misdemeanor	0.49	prisoner i was once arrested for a misdemeanor
prior_arrest_for_local_ordinance	0.01	prisoner i was once arrested for local ordinance
prior_arrest_with_firearms_involved	0.09	prisoner i was once arrested or an incident involving firearms
prior_arrest_with_child_involved	0.17	prisoner i was once arrested for an incident involving children
no_prior_arrests	0.12	prisoner i has no prior arrests
prior_arrests≥1	0.88	prisoner i has at least 1 prior arrest
prior_arrests≥2	0.78	prisoner i has at least 2 prior arrests
prior_arrests≥5	0.60	prisoner i has at least 5 prior arrests
multiple_prior_prison_time	0.43	prisoner i has been to prison multiple times
any_prior_jail_time	0.47	prisoner i has been to jail at least once
multiple_prior_jail_time	0.29	prisoner i has been to prison multiple times
any_prior_probation_or_fine	0.42	prisoner i has been on probation or paid a fine at least once
multiple_prior_probation_or_fine	0.22	prisoner i has been on probation or paid a fine multiple times

Table 5.1: Input variables for all recidivism prediction problems. Each variable is a binary variable $x_{ij} \in \{0, 1\}$.

Prediction Problem	$\Pr(y_i = +1)$	**Conditions Required for** $y_i = +1$
`arrest`	59.0%	prisoner i is arrested for any offense within 3 years of release from prison
`drug`	20.0%	prisoner i is arrested for drug-related offense (e.g., possession, trafficking) within 3 years of release from prison
`general_violence`	19.1%	prisoner i is arrested for a violent offense (e.g., robbery, aggravated assault) within 3 years of release from prison
`domestic_violence`	3.5%	prisoner i is arrested for domestic violence within 3 years of release from prison
`sexual_violence`	3.0%	prisoner i is arrested for sexual violence within 3 years of release from prison
`fatal_violence`	0.7%	prisoner i is arrested for murder or manslaughter within 3 years of release from prison

Table 5.2: Outcomes for recidivism prediction problems. The values of $\Pr(y_i = +1)$ do not add up to 100% because each arrest can be associated with more than one type of crime, and a prisoner may be arrested multiple times over the 3 year follow-up period.

5.3 Methodology

We compared SLIM scoring systems to models produced by 8 baseline classification methods shown in Table 5.3.

Setup

For each recidivism prediction problem and each method, we fit 19 decision-making models using a standard cost-sensitive approach. We chose 19 values of the misclassification cost parameter w^+ to fit models across the full ROC curve. Thus, our comparison involved required fitting a total of 1,026 decision-making models (i.e. 6 recidivism problems $\times$ 9 methods $\times$ 19 values of w^+).

By default, we used values of $w^+ \in \{0.1, 0.2, \ldots, 1.9\}$ and set $w^- = 2 - w^+$. These values of w^+ were inappropriate for problems with significant class imbalance because many methods would produce trivial models for most values of w^+. Thus, for significantly imbalanced problems, such as `domestic_violence` and `sexual_violence`, we used values of $w^+ \in \{1.815, 1.820, \ldots, 1.995\}$. For `fatal_violence`, which was extremely imbalanced, we used $w^+ \in \{1.975, 1.976, \ldots, 1.995\}$.

We fit the "best" model for a given value of w^+ as follows. We used 2/3 of the data as the *training set* and 1/3 of the data as an out-of-sample *test set*. We used a standard 5-fold cross-validation (5-CV) setup for parameter tuning, and chose an instance of the parameters that minimized the mean weighted 5-CV validation error. Using these free parameter values, we then fit *final model* using the entire training set, and reported its performance on the test set (1/3). We used the same test set, training set, and validation folds in order to allow for comparisons across methods and prediction problems.

Computation

We fit models for baseline methods using publicly available packages in R 3.2.2 without imposing time constraints. We fit SLIM scoring systems by solving the IP formulation in (3.7a) using the CPLEX 12.6 API in MATLAB 2013a. We solved each IP for 4 hours on a computing cluster with 2.7GHz CPUs. Each time we solved a IP we kept 500 feasible solutions, and polished them using the polishing IP in Section 3.4.1. We then used the same CV setup as the other methods to tune the number of terms in the final model. Polishing all 500 solutions took less than 1 minute. Thus, the total number of optimization problems we solved were 500 polishing IPs $\times$ (5 folds + 1 final model) $\times$ 6 problems $\times$ 19 values of w^+ = 342,000 IPs.

Performance Metrics

We summarize the overall classification performance for each method over the ROC curve using a metric that we refer to as *mAUC* (i.e. method AUC). Here, we use mAUC to clearly distinguish this metric from AUC, which we will use as a performance metric for ranking. The mAUC of a given method reflects the area under the ROC curve built using all models from that method (i.e. the best models obtained using the 19 values of w^+). A method that with mAUC = 1 always produces models that are more accurate than a method with mAUC = 0. Other than this basic case, however, it is not possible to state that a method with high mAUC always produces models that are more accurate than a method with low mAUC.

Limitations

Our setup in this chapter differs from the experimental setup in Chapter 3.6.1 as we sought to abide by guidelines for methodological benchmarks in the criminology literature (see Berk and Bleich, 2013). In particular, we had to ensure that all methods used the same cost weighting scheme to deal with imbalanced problems. Certain classification methods may fare better on imbalanced problemsif we dealt with class imbalance using an alternative approach (e.g. by using sampling instead of weighing). This also applies to SLIM, which would have have performed better at several decision points if we used FPR constraints (as these improve the effectiveness of B&B by strengthening the formulation and allowing the solver to prune more effectively as in Chapter 4).

Method	Acronym	Software	Free Parameters and Settings
CART Decision Trees	CART	**rpart** (Therneau et al., 2012)	minSplit $\in (3, 5, 10, 15, 20) \times$ CP $\in (0.0001, 0.001, 0.01)$
C5.0 Decision Trees	C5.0T	**c50** Kuhn et al. (2012)	default settings
C5.0 Decision Rules	C5.0R	**c50** Kuhn et al. (2012)	default settings
Logistic Regression $+\ell_1$-Penalty	Lasso	**glmnet** Friedman et al. (2010)	100 values of ℓ_1-penalty chosen by **glmnet**
Logistic Regression $+\ell_2$-Penalty	Ridge	**glmnet** Friedman et al. (2010)	100 values of ℓ_2-penalty chosen by **glmnet**
Random Forests	RF	**randomForest** Liaw and Wiener (2002)	sampsize $\in (0.632n, 0.4n, 0.2n) \times$ nodesize $\in (1, 5, 10, 20)$ with unbounded tree depth
SVM with RBF Kernel	SVM RBF	**e1071** Meyer et al. (2012)	$C \in (0.01, 0.1, 1, 10) \times$ $\gamma \in (\frac{1}{10d}, \frac{1}{5d}, \frac{1}{2d}, \frac{1}{d}, \frac{2}{d}, \frac{5}{d}, \frac{10}{d})$
Stochastic Gradient Boosting	SGB	**gbm** Ridgeway (2006)	shrinkage $\in (0.001, 0.01, 0.1) \times$ interaction.depth $\in (1, 2, 3, 4) \times$ ntrees $\in (100, 500, 1500, 3000)$
SLIM Scoring Systems	SLIM	**CPLEX 12.6**	C_0 and ϵ set to find most accurate model with ≤ 8 coefficients where $\lambda_0 \in \{-100, \ldots, 100\}$ $\lambda_j \in \{-10, \ldots, 10\}$

Table 5.3: Classification methods used to fit models for all recidivism prediction problems. We ran each method for 19 values of w^+ and all combinations of free parameters shown in the table. For each value of w^+, we selected the model that minimized the mean weighted 5-CV validation error.

5.4 Results

We show ROC curves for all methods and prediction problems in Figure 5.1 and summarize the test mAUC of each method in Table 5.4. We make the following observations, which we believe carry over to a large class of problems beyond recidivism prediction:

- All methods did well on the general recidivism prediction problem, `arrest`. In this case, we observe only small differences in predictive accuracy of different methods. All methods other than CART attain a test mAUC above 0.72; the highest test mAUC of 0.73 was achieved by SGB, Ridge, and RF. This multiplicity of good models reflects the *Rashomon effect* of Breiman (2001b).

- Major differences between methods appeared in their performance on imbalanced prediction problems. We expected different methods to respond differently to changes in the misclassification costs, and therefore trained each method over a large range of misclassification costs. Even so, it was difficult to tune certain methods to produce models at certain points of the ROC curve (see e.g., problems with significant imbalance, such as `fatal_violence`).

- SVM RBF, SGB, Lasso and Ridge were able to produce accurate models at different points on the ROC curve for most problems. SGB usually achieved the highest mAUC on most problems (e.g., `arrest`, `drug`, `general_violence`). Lasso, Ridge, and SVM RBF often produce comparable mAUCs. We find that these methods respond well to cost-sensitive tuning, but it is difficult to determine suitable misclassification costs on highly imbalanced problems (e.g. `fatal_violence`) to fit models at specific points on the ROC curve.

- C5.0T, C5.0R and CART were unable to produce accurate models across the full ROC curve on imbalanced prediction problems. We found that these methods do not respond well to cost-sensitive tuning. For `drug` and `general_violence`, for instance, these methods could not produce models with high TPR. The issue becomes more severe as problems become more imbalanced. For `fatal_violence`, `sexual_violence`, and `domestic_violence`, these methods typically produced trivial models that predict $\hat{y}_i = +1$ or $\hat{y}i = -1$ (resulting in mAUCs of 0.5). This result may be attributed to the greedy nature of the algorithms used to fit the trees, as opposed to the use of tree models in general. The issue is unlikely to be software-related as it affects both C5.0 and CART, and has been observed by others (see e.g., Goh and Rudin, 2014).

- SLIM performs well despite being restricted to a small class of simple linear models (e.g., models with at most 8 non-zero integer coefficients from -10 to 10). Even on highly imbalanced problems such as `domestic_violence` and `sexual_violence`, it responds well to changes in misclassification costs. This is expected by nature of its exact formulation.

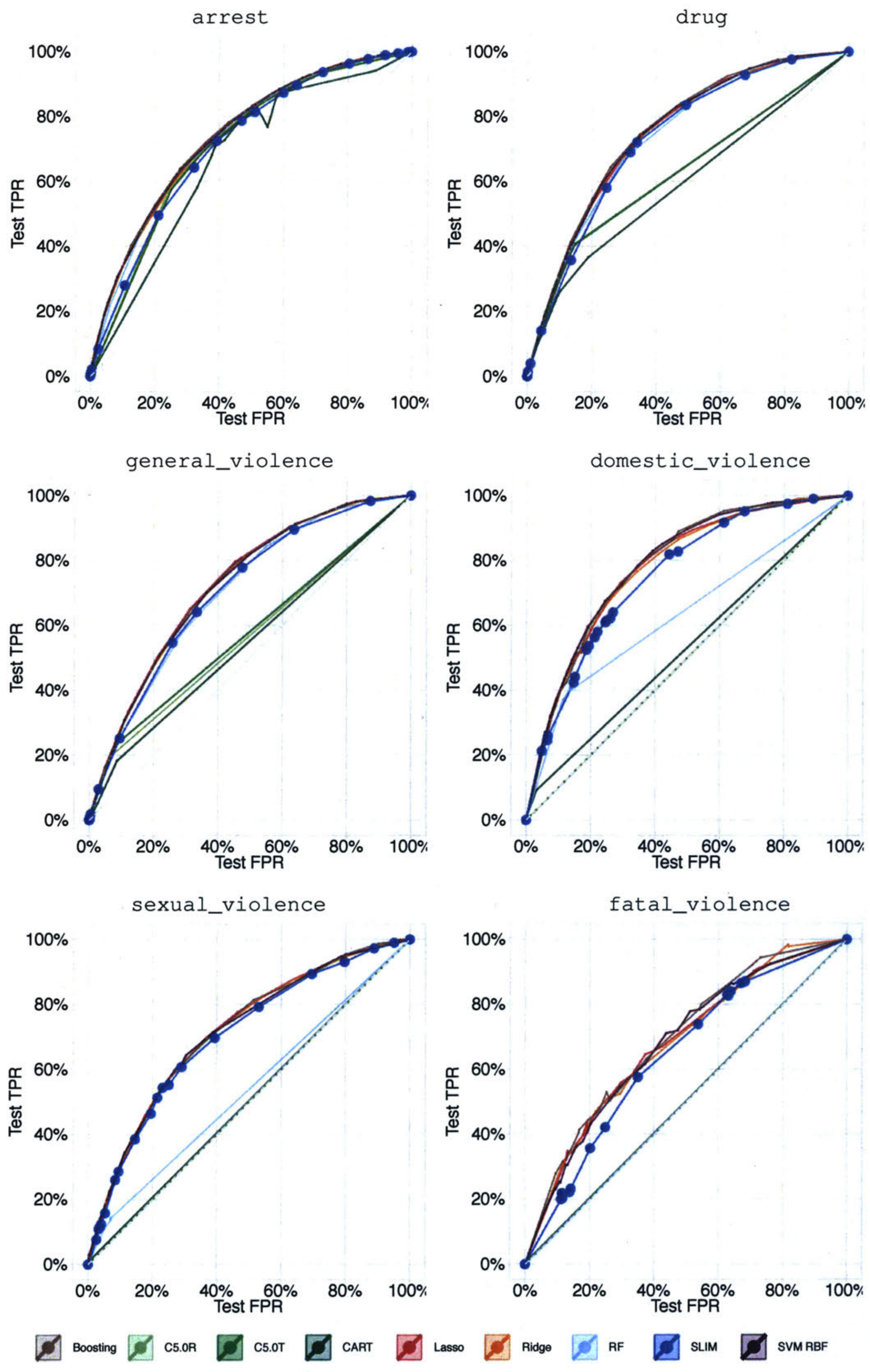


Figure 5.1: ROC curves for recidivism prediction problems. We plot SLIM models using large blue dots. All models perform similarly except for C5.0R, C5.0T, and CART.

Problem	Lasso	Ridge	C5.0R	C5.0T	CART	RF	SVM RBF	SGB	SLIM
arrest	0.72	0.73	0.72	0.72	0.68	0.73	0.72	0.73	0.72
drug	0.74	0.74	0.63	0.63	0.59	0.75	0.73	0.75	0.74
general_violence	0.72	0.72	0.56	0.57	0.56	0.71	0.70	0.72	0.71
domestic_violence	0.77	0.77	0.50	0.50	0.53	0.64	0.77	0.78	0.76
sexual_violence	0.72	0.72	0.50	0.50	0.51	0.54	0.69	0.70	0.70
fatal_violence	0.67	0.68	0.50	0.50	0.50	0.50	0.69	0.70	0.62

Table 5.4: Test mAUC for all methods on all prediction problems.

5.4.1 On Performance under a Model Size Constraint

Baseline methods were unable to maintain the same level of accuracy under a simple model size constraint. In this setting, the only methods that can consistently produce accurate models along the full ROC curve and also have the potential for interpretability are SLIM and Lasso. Tree and rule-based methods such as CART, C5.0T and C5.0R were generally unable to produce models that attain high degrees of accuracy. Worse, even for balanced problems such as arrest, where these methods did produce accurate models, the models are complicated and use a very large number of rules or leaves (similar behavior for C5.0T/C5.0R is also observed by, for instance, Lim et al., 2000).

In Table 5.5, we show the percentage change in test mAUC for the methods due to the model size restriction. As shown, the overall accuracy of the models produced by each method are significantly affected by the model size constraint. Note that C5.0R and C5.0T are unable to produce a suitably sparse model for some of the problems as their implementation does not provide control over model sparsity.

Problem	Lasso	C5.0R	C5.0T	CART	SLIM
arrest	-3.8%	-	-	-2.8%	0.0%
drug	-4.0%	-	-	-15.7%	0.0%
general_violence	-2.2%	-11.0%	-12.7%	-10.3%	0.0%
domestic_violence	-4.1%	-	-	-5.4%	0.0%
sexual_violence	-2.2%	-	-	-1.8%	0.0%
fatal_violence	-11.2%	-	-	0.0%	0.0%

Table 5.5: Percentage change in Test mAUC when transparent methods are restricted to models with at most 8 coefficients, 8 leaves or 8 rules.

5.4.2 On the Interpretability of Equally Accurate Models

To discuss interpretability, we show the models from SLIM, Lasso and CART for the `arrest` problem in Figures 5.5 – 5.4. This setup provides a nice basis for comparison as all methods fit transparent models at roughly the same degree of accuracy and sparsity. For this comparison, we considered any transparent model with at most 8 coefficients (Lasso), 8 rules (C5.0R) or 8 leaves (C5.0T, CART) and had a test FPR of below 50%. In this case, neither C5.0R nor C5.0T could produce an acceptable model with at most 8 rules or 8 leaves. We make the following observations:

- All three models attain similar levels of predictive accuracy. Test TPR values ranged between 70-79% and test FPR values ranged between 43-48%. There may not exist a classification model that can attain substantially higher accuracy. The highest test TPR attained by models with test FPR $\leq$ 50% was produced by the SVM RBF model which had a TPR of 80%.
- The SLIM model uses 5 input variables and small integer coefficients. Here, the model can be further simplified by combining *age_ at_ release* into single variable (see e.g., Figure 5.5). There is a natural rule-based interpretation. In this case, the model implies that if the prisoner is young (*age_ at_ release_ of_ 18_ to_ 24*) or has a history of arrests (*prior_ arrests$\geq$5*), we should predict that they will be rearrested. On the other hand, if the prisoner is older (*age_ at_ release$\geq$40*) or has no history of arrests (*no_ prior_ arrests*), we should not.
- The Lasso model allows users to gauge the importance of each feature by comparing the size of different coefficients, as the size of each coefficient reflects an odds ratio. The composition of variables in this model is similar to that of the SLIM model – as both *prior_ arrest* and *age_ at_ release$\geq$40* are important factors. In comparison to the SLIM model, the Lasso model is slightly more difficult to comprehend as it uses 7 input variables (instead of 4) with *real* coefficients that range between (0.61 to 0.005). The Lasso model does not benefit from the ability to collapse mutually exclusive features, make hands-on predictions, or yield the same kind of rule-based insights as the SLIM model.
- The CART model also allows users to make predictions without a calculator. In comparison to the SLIM model, however, the hierarchical structure of the CART model makes it difficult to gauge the relationship of each input variable on the predicted outcome. Consider, for instance, the relationship between age at release and the outcome. In this case, users are immediately aware that there is an effect, as the model branches on the variables *age_ at_ release$\geq$40* and *age_ at_ release_ 18_ to_ 24*. However, the effect is difficult to comprehend since it depends on prior arrests for misdemeanor: if *prior_ arrests$\geq$5* $= 1$ and *age_ at_ release_ 18_ to_ 24* $= 1$ then the model predicts $\hat{y} = +1$; if *prior_ arrests$\geq$5* $= 0$ and *age_ at_ release$\geq$40* $= 0$ then $\hat{y} = +1$; however, if *prior_ arrests$\geq$5* $= 0$ and *age_ at_ release$\geq$40* $= 1$ then $\hat{y} = +1$ only if *prior_ arrest_for_ misdemeanor* $= 1$. Such issues do not affect linear models such as SLIM and Lasso, where users can immediately gauge the direction and strength of the relationship between a input variable and the predicted outcome by the size and sign of a coefficient.

PREDICT ARREST FOR ANY OFFENSE IF SCORE > 1

1.	*age_at_release_18_to_24*	2 points		$\cdots$
2.	*prior_arrests≥5*	2 points	$+$	$\cdots$
3.	*prior_arrest_for_misdemeanor*	1 point	$+$	$\cdots$
4.	*no_prior_arrests*	-1 point	$+$	$\cdots$
5.	*age_at_release≥40*	-1 point	$+$	$\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$	$\cdots$

Figure 5.2: SLIM scoring system for `arrest`. This model has a test TPR/FPR of 76.6%/44.5%, and a mean 5-CV validation TPR/FPR of 78.3%/46.5%.

PREDICT ARREST FOR ANY OFFENSE IF SCORE > 31

1.	*prior_arrests≥5*	63 points		$\cdots$
2.	*age_1st_confinement_18_to_24*	15 points	$+$	$\cdots$
3.	*prior_arrest_for_property*	9 points	$+$	$\cdots$
4.	*prior_arrest_for_misdemeanor*	5 points	$+$	$\cdots$
5.	*age_at_release≥40*	-20 points	$+$	$\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$	$\cdots$

Figure 5.3: Lasso model for `arrest` with coefficients rounded to two significant digits and scaled by 100. This model has a test TPR/FPR of 70.9%/43.8%, and a mean 5-CV validation TPR/FPR of 72.2%/44.0%.

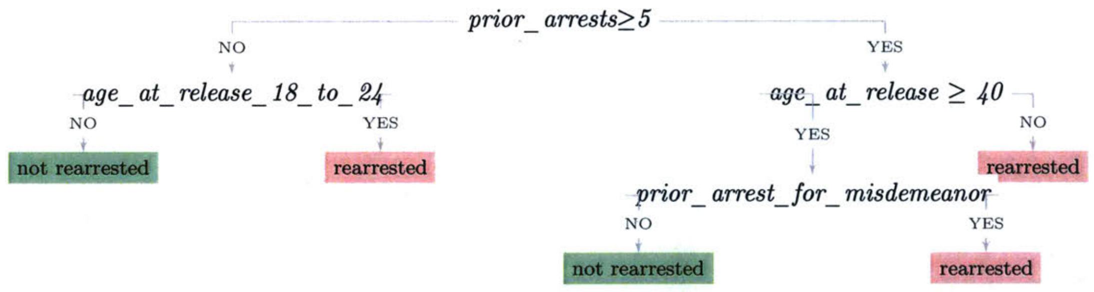


Figure 5.4: CART decision tree for `arrest`. This model has a test TPR/FPR of 79.1%/47.9%, and a mean 5-CV validation TPR/FPR of 79.9%/48.5%.

Scoring Systems for Recidivism Prediction

We present SLIM scoring systems for other problems in Figures 5.5 to 5.8. The models are chosen for the decision-point 5-CV FPR$\leq$ 50%, and would be suitable for screening applications. To produce a model for sentencing, a point on the ROC curve with a much higher TPR would be needed. Once again, we observe that these models generalize well, as evidenced by the close match between test TPR/FPR and training TPR/FPR.

PREDICT ARREST FOR DRUG OFFENSE IF SCORE > 7

1.	*prior_arrest_for_drugs*	9 points	···
2.	*age_at_release_18_to_24*	5 points	+ ···
3.	*age_at_release_25_to_29*	3 points	+ ···
4.	*prior_arrest_for_multiple_types_of_crime*	2 points	+ ···
5.	*prior_arrest_for_property*	1 points	+ ···
6.	*age_at_release_30_to_39*	-1 point	+ ···
7.	*no_prior_arrests*	-6 points	+ ···
	ADD POINTS FROM ROWS 1-7	**SCORE**	= ···

Figure 5.5: SLIM scoring system for drug. This model has a test TPR/FPR of 85.7%/51.1%, and a mean 5-CV validation TPR/FPR of 82.3%/49.7%.

PREDICT ARREST FOR GENERAL VIOLENCE IF SCORE > 7

1.	*prior_arrest_for_general_violence*	8 points	···
2.	*prior_arrest_for_misdemeanor*	5 points	+ ···
3.	*infraction_in_prison*	3 points	+ ···
4.	*prior_arrest_for_local_ord*	3 points	+ ···
5.	*prior_arrest_for_property*	2 points	+ ···
6.	*prior_arrest_for_fatal_violence*	2 points	+ ···
7.	*prior_arrest_with_firearms_involved*	1 point	+ ···
8.	*age_at_release*$\geq$*40*	-7 points	+ ···
	ADD POINTS FROM ROWS 1-8	**SCORE**	= ···

Figure 5.6: SLIM scoring system for general_violence. This model has a test TPR/FPR of 76.7%/45.4%, and a mean 5-CV validation TPR/FPR of 76.8%/47.6%.

PREDICT ARREST FOR DOMESTIC VIOLENCE IF SCORE > 3

1.	*prior_arrest_for_misdemeanor*	4 points	···
2.	*prior_arrest_for_felony*	3 points	+ ···
3.	*prior_arrest_for_domestic_violence*	2 points	+ ···
4.	*age_1st_confinement_18_to_24*	1 point	+ ···
5.	*infraction_in_prison*	-5 points	+ ···
	ADD POINTS FROM ROWS 1-5	**SCORE**	= ···

Figure 5.7: SLIM scoring system for domestic_violence. This model has a test TPR/FPR of 85.5%/46.0%, and a mean 5-CV validation TPR/FPR of 81.4%/48.0%.

PREDICT ARREST FOR FATAL VIOLENCE OFFENSE IF SCORE > 4

1.	*age_1st_confinement*$\leq$*17*	5 points	···
2.	*prior_arrest_with_firearms_involved*	3 points	+ ···
3.	*age_1st_confinement_18_to_24*	2 points	+ ···
4.	*prior_arrest_for_felony*	2 points	+ ···
5.	*age_at_release_18_to_24*	1 point	+ ···
6.	*prior_arrest_for_drugs*	1 point	+ ···
	ADD POINTS FROM ROWS 1-6	**SCORE**	= ···

Figure 5.8: SLIM scoring system for fatal_violence. This model has a test TPR/FPR of 55.4%/35.5%, and a mean 5-CV validation TPR/FPR of 64.2%/42.4%

.

5.5 Discussion

Trade-offs Between Simple and Complex Predictive Models

There has been some controversy in the criminology community as to whether modern machine learning methods such as random forests (see e.g., Breiman, 2001b; Berk et al., 2009; Ritter, 2013) are necessary to produce accurate predictive models of recidivism, or if "traditional" approaches such as logistic regression and linear discriminant analysis would suffice (see e.g., Tollenaar and van der Heijden, 2013; Berk and Bleich, 2013; Bushway, 2013).

Our results in this chapter show that the answer can be far subtle than a simple yes or no. In particular, the answer depends on how the models will be used for decision-making. For each use case (e.g., sentencing, parole decisions, policy interventions), one might need a different true positive rate (TPR) and/or false positive rate (FPR) (see also Ritter, 2013). In order to determine if one method is better than another, one must consider the appropriate point along the ROC curve for decision-making.

As we show, for a wide range of recidivism prediction problems, many machine learning methods (support vector machines, random forests) produce equally accurate predictive models along the ROC curve. However, there are trade-offs between accuracy, transparency, and interpretability: methods that are designed to yield transparent models (CART, C5.0) cannot be tuned to produce as accurate models along the ROC curve, and do not consistently output models that are easy to use, understand, or validate.

This is not to say that such models for not exist. The fact that many methods produce models with similar levels of predictive accuracy indicates that there is a large class of approximately-equally-accurate predictive models (called the "Rashomon" effect by Breiman 2001a). In this case, there may exist scoring systems that also attain the same level of accuracy as other predictive models. Finding such models, however, may be computationally challenging.

On the Importance of Validation

Historically, the use of recidivism prediction models was fueled by work on *clinical versus actuarial judgment*, which showed that humans – on their own – are not as good at decision-making assessment as statistical models (Dawes et al., 1989; Grove and Meehl, 1996). Recently, however, the use and development of models in the modern criminal justice system has been a controversial topic among policy-makers and legal scholars (see e.g. the critiques of Harcourt, 2008; Hannah-Moffat, 2013).

Several articles (e.g. Barry-Jester et al., 2015; Angwin et al., 2016) have led to broader public awareness that predictive models in the criminal justice system may lead to unintended consequences, such as discrimination. In light of these articles, the central question within the legal community is not whether we can build predictive models that do not discriminate among individuals, but whether it is ethical to use a predictive model to make decisions that affect humans in the first place. Starr (2014), for instance, calls the use of these models "unconstitutional", and states that the "I doubt many policy makers would publicly defend the claim that people should be imprisoned longer because they are poor, for instance."

Modern predictive modeling, in general, does not provide a good approach to address this concern. In particular, one approach to show that income is not an important predictor (e.g. by pairing a predictive model with a variable importance plot). This is misleading as the model can still discriminate based on income through correlations with proxy variables. A different approach would be to provide some evidence that the predictive model does not discriminate on the basis of income on the training data (by analysis, or through a fairness constraint). This approach can more effectively address the potential to discriminate through proxy variables. However it does not guarantee that the model will not discriminate when it is deployed on a new population of individuals (Barry-Jester et al., 2015).

Such issues are inherently difficult to address from a methodological standpoint. In fact, there may be no practical approach to address them other than through careful validation before and *during* deployment. Validation does guarantee that the model will not discriminate based on income, but it can reduce the potential for the discrimination to go unchecked. In light of this, a simple scoring system such as the one in Figure is valuable as users can extract a boolean function that represents the exact decision-rule. This allows users to fully understand the interactions between the variables, and validate predictions each time they are used. This degree of validation differs from other techniques in that it does not require access to the data, in that it can be done without training, and in that it provides an exact representation of how the model operates.

5.5.1 On the Importance of Certifiable Optimality

The results from this chapter highlight the value of a certificate of optimality for predictive models in criminal justice.

Model development in criminal justice not only involves significant methodological work (to address operational constraints), but also requires extensive validation (to prove to policy-makers that the models attain a reasonable performance trade-off). As a example, note that the scoring system developed by the Pennsylvania Sentencing Commission in Figure 1.1 was developed over 7 years (Pennsylvania Bulletin, 2017).

SLIM scoring systems can address this need for validation through meaningful performance guarantees. In particular, a certifiably optimal scoring system attains the best training accuracy among all scoring systems. In addition, since the models can be expected to generalize, the training accuracy can be used as a proxy for their testing accuracy. These guarantees provide an effective way to address real-world applications that involve performance guarantees and constraints.

Say, for example, that policy makers ask a practitioner to create a scoring system for parole violation. We note:

- If practitioner is unable to recover a certifiably model, a small optimality gap may still be useful because the objective function involves meaningful quantities (e.g. if the gap is 5%, then a certifiable optimal model will be no more than 5% more accurate).

- If a certifiably optimal scoring system performs poorly at a certain point on the ROC curve (e.g. the model has low TPR at the maximum FPR that policy makers are willing to tolerate), then a practitioner can tell policy-makers that they will need to tolerate a larger FPR.
- If a certifiably optimal scoring system performs poorly at a certain point on the ROC curve, then no scoring system for risk assessment will contain a decision-point that does better (this is useful given that many tools are developed for risk assessment).
- If the policy-makers imposed additional constraints on the scoring system (e.g on fairness), then the practitioner can evaluate the impact of these constraints by comparing the performance of a certifiably optimal scoring system with/without these constraints.
- If a certifiably scoring system performs worse than a human-decision maker given their constraints, then policy makers can make an informed decision on whether or not it should be deployed.

Such guarantees do not apply to many predictive models, as methods often do not produce a model that attains the best performance in the model class: the model may perform well in terms of the performance metric that policy makers care about, such as accuracy or risk calibration, but there may exist a model that performs better. In some cases, the models may not generalize reliably (which introduces additional trade-offs that are difficult to work with). Lastly, the performance of the model may suffer due to disproportionality under certain constraints, which makes it difficult to evaluate the impact of their constraints (i.e. if the performance of the model suffers, it is difficult to tell if this is because of the constraint, or because the method cannot handle it).

In theory, scoring systems can achieve the best possible training accuracy within the class of linear models (see Theorem (3.26)) In practice, this requires a dataset with binary features, and may be difficult to recover for models that use a large number of variables. In domains such as criminal justice, however, this guarantee may be relevant as: (i) The vast majority of tools that have been deployed have been sparse linear models with binary features. Domain experts may only be interested in such models as they are more usable. (ii) The applications involve small datasets (e.g. there is only a limited number prisoners within the criminal justice system at any time. Even if the performance of the tool would increase by using a larger dataset, it may not necessarily be ethical to do so as this would use prisoners from a different population, or a different time period.)

Chapter 6

Optimized Risk Scores

In this chapter, we consider the problem of learning sparse linear classification models with small integer coefficients for risk assessment (*risk scores*).

Organization

This chapter is organized as follows. In Section 6.1, we formally define the risk score problem and discuss its special properties. In Section 6.2, we briefly review cutting plane algorithms, and present a new cutting plane algorithm for the risk score problem, LCPA. In Sections 6.3 and 6.3.3, we present specialized techniques to improve the performance of LCPA for the risk score problem. In Section 6.4, we present experimental results to benchmark risk scores built using RISKSLIM and other advanced heuristics.

Notes

This chapter primarily contains material from Ustun and Rudin (2016a, 2017), which builds upon our earlier work in (Ustun and Rudin, 2014), where we proposed solving problems such as the risk score problem in (6.1) using the CPA algorithm in Algorithm 4. Software for RISKSLIM is available online at http://github.com/ustunb/risk-slim.

6.1 Problem Statement

Our goal is to build scoring systems for risk assessment (i.e., *risk scores*) such as the one in Figure 6.1.

1.	*Congestive Heart Failure*	1 point	$\cdots$
2.	*Hypertension*	1 point	$+ \cdots$
3.	*Age $\geq$ 75*	1 point	$+ \cdots$
4.	*Diabetes Mellitus*	1 point	$+ \cdots$
5.	*Prior Stroke or Transient Ischemic Attack*	2 points	$+ \cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$= \cdots$

SCORE	0	1	2	3	4	5	6
RISK	1.9%	2.8%	4.0%	5.9%	8.5%	12.5%	18.2

Figure 6.1: CHADS$_2$ risk score of Gage et al. (2001) for assessing stroke risk.

We formalize this problem as follows. We start with a dataset of n i.i.d. training examples $(\boldsymbol{x}_i, y_i)_{i=1}^n$ where $\boldsymbol{x}_i \subseteq \mathbb{R}^{d+1}$ denotes a vector of features $[1, x_{i,1}, \ldots, x_{i,d}]^T$ and $y_i \in \{-1, +1\}$ denotes a class label. We consider a linear score function $\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle$ where $\boldsymbol{\lambda} \subseteq \mathbb{R}^{d+1}$ is a vector of coefficients $[\lambda_0, \lambda_1, \ldots, \lambda_d]^\top$, and λ_0 is an intercept term. We estimate the *predicted risk* that example i belongs to the positive class using the logistic link function as

$$\Pr(y_i = +1 \mid \boldsymbol{x}_i) = \frac{1}{1 + \exp(-\langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle)}.$$

In this setup, each coefficient λ_j represents the number of points for a given feature. Given an example with features $\boldsymbol{x}_i$, users tally the points to obtain a total score $s_i := \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle$, and use the total score s_i to obtain an estimate of predicted risk. Alternatively, users can obtain a predicted label $\hat{y}_i \in \{\pm 1\}$ by comparing the predicted risk to a threshold risk (e.g., predict $\hat{y}_i = +1$ if and only if $\Pr(y_i = +1) > 50\%$).

In practice, the desirable performance characteristics of a risk score include:

- *Rank Accuracy*: A rank-accurate model is a model with high AUC. Such a model outputs scores that can be used to order examples in terms of their true risk.
- *Risk Calibration*: A risk-calibrated model yields risk predictions that match observed risk. A risk-calibrated model has high AUC, but the converse is not necessarily true.

Risk Score Problem

We learn the values of the coefficients from data by solving the following mixed integer nonlinear program (MINLP), which we refer to as the *risk score problem* or RISKSLIMMINLP. We provide a formal description of this problem in Definition 6.1.

Definition 6.1 (Risk Score Problem, RISKSLIMMINLP)
The risk score problem is a discrete optimization problem with the form:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{6.1}$$

where:

- $l(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-\langle \boldsymbol{\lambda}, y_i \boldsymbol{x}_i \rangle))$ *is the logistic loss function;*
- $\|\boldsymbol{\lambda}\|_0 = \sum_{j=1}^{d} \mathbb{1}[\lambda_j \neq 0]$ *is the ℓ_0-seminorm;*
- $\mathcal{L} \subset \mathbb{Z}^{d+1}$ *is a set of feasible coefficient vectors (user-provided);*
- $C_0 > 0$ *is a trade-off parameter to balance fit and sparsity (user-provided);*

Definition 6.2 (Risk-calibrated Supersparse Linear Integer Model, RISKSLIM)
*A Risk-calibrated Supersparse Linear Integer Model (*RISKSLIM*) is a scoring system built using the optimal solution to* RISKSLIMMINLP.

Here, the objective minimizes the *logistic loss* for AUC and risk calibration, and penalizes the ℓ_0-norm for sparsity. The trade-off parameter C_0 controls the balance between these competing objectives, and represents the maximum log-likelihood that is sacrificed to remove a feature from the optimal model. The feasible region restricts coefficients to a small set of bounded integers such as $\mathcal{L} = \{-5, \ldots, 5\}^{d+1}$, and may be further customized to include operational constraints, such as those in Table 6.1.

Constraint Type	**Example**
Feature Selection	Choose up to 5 features
Group Sparsity	Include either $Male$ or $Female$ in the model but not both
Optimal Thresholds	Use ≤ 3 thresholds for Age: $\sum_{k=1}^{100} \mathbb{1}[Age \leq k] \leq 3$
Logical Structure	If $Male$ is in model, then include $Hypertension$ or $BMI \geq 30$ as a control
Probability	Predict $\Pr(y = 1 \mid \boldsymbol{x}) \geq 0.90$ if $Male = \text{TRUE}$ and $Hypertension = \text{TRUE}$

Table 6.1: Operational constraints that can be enforced on risk scores by including additional constraints in the feasible region of RISKSLIMMINLP.

RiskSlimMINLP aims to capture the *exact* objectives and constraints of risk scores, so that its optimizer attains the minimum logistic loss among feasible models on the training data. In particular, an optimal solution to RiskSlimMINLP attains the lowest value of the logistic loss among feasible models on the training data – provided that C_0 is small enough (see Appendix B for a proof). In Section 6.4, we show that models that minimize the logistic loss achieve high AUC and risk calibration on training data, and that this generalizes to test data due to the simplicity of our hypothesis space.

There are some theoretical results to motivate why minimizers of the logistic loss have good risk calibration and AUC. In particular, the logistic loss is a strictly proper loss (Reid and Williamson, 2010) which yields calibrated estimates of predicted risk under the parametric assumption that the true risk can be modeled using a logistic link function (see Menon et al., 2012). In addition, the work of Kotlowski et al. (2011) shows that a "balanced" version of the logistic loss forms a lower bound on 1−AUC, which means that minimizing the logistic loss indirectly maximizes a surrogate of AUC (i.e., a lower bound on a "scaled" AUC).

Trade-off Parameter

Using an exact formulation provides an alternative way to set the trade-off parameter C_0:

- If we are given a limit on the model size (e.g. $\|\lambda\|_0 \leq k$), we can add this as a constraint in the formulation and set C_0 to a small value (e.g. $C_0 = 10^{-8}$). In this case, the optimal solution corresponds to the best model that obeys the model size constraint, provided C_0 is small enough (see Appendix B).
- Alternatively, we can choose the model size based on cross-validated (CV) performance. In this case, we would repeat the previous process for $\|\boldsymbol{\lambda}\|_0 \leq k$ for $k = 1 \ldots d$. This lets us fit the full range of risk scores (i.e. the full ℓ_0-regularization path) by solving at most d instances of RiskSlimMINLP. In comparison, a standard CV-based approach (i.e. where we treat C_0 as the hyperparameter) is likely to require solving more than d instances as one cannot determine d values of C_0 to return the full range of risk scores *a priori*.

Computational Complexity

Optimizing RiskSlimMINLP is a difficult computational task given that ℓ_0 -regularization, minimizing over integers, and MINLP problems are all NP-hard (Bonami et al., 2012). These worst-case complexity results mean that finding an optimal solution to RiskSlimMINLP may be intractable for high dimensional datasets. As we show, however, RiskSlimMINLP can be solved to optimality for many real-world datasets in minutes, and in a way that scales linearly in n.

Assumptions, Notation and Terminology

We denote the objective function of RiskSlimMINLP as

$$V(\boldsymbol{\lambda}) = l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0, \tag{6.2}$$

and denote an optimal solution as $\boldsymbol{\lambda}^* \in \operatorname{argmin}_{\boldsymbol{\lambda} \in \mathcal{L}} V(\boldsymbol{\lambda})$. We bound the values of objective value function and its components at the optimal solution $\boldsymbol{\lambda}^*$ as follows: $V(\boldsymbol{\lambda}^*) \in [V^{\min}, V^{\max}]$, $l(\boldsymbol{\lambda}^*) \in [L^{\min}, L^{\max}]$, $\|\boldsymbol{\lambda}^*\|_0 \in [R^{\min}, R^{\max}]$. In addition, we define the set of feasible values for each coefficient as $\lambda_j \in \mathcal{L}_j$ for $j = 0, \ldots, d$, and denote bounds as $\Lambda_j^{\min} = \min_{\lambda_j \in \mathcal{L}_j} \lambda_j$ and $\Lambda_j^{\max} = \max_{\lambda_j \in \mathcal{L}_j} \lambda_j$.

We make two following assumptions for clarity of exposition: (i) $\mathbf{0} \in \mathcal{L}$, which ensures that RiskSlimMINLP is always feasible; (ii) the intercept is not regularized, which means the precise version of the RiskSlimMINLP objective is $V = l(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}_{[1,d]}\right\|_0$ where $\boldsymbol{\lambda} = [\lambda_0, \boldsymbol{\lambda}_{[1,d]}]$.

6.2 Methodology

In this section, we introduce the cutting plane algorithm that we use to solve the risk score problem, RiskSlimMINLP. In Section 6.2.1, we discuss a simple cutting plane algorithm to explain the benefits of using cutting plane algorithms to solve RiskSlimMINLP and explain why such algorithms stall in non-convex settings. In Section 6.2.2, we present a new cutting plane algorithm that does not stall in non-convex settings. In Section 6.2.3, we compare the performance of both algorithms to a commercial MINLP solvers on difficult instances of RiskSlimMINLP.

6.2.1 Cutting Plane Algorithms

In Algorithm 4, we present a simple cutting plane algorithm to solve RiskSlim-MINLP that we refer to as CPA. In what follows, we use CPA to briefly introduce cutting plane algorithms, explain why they are well-suited to solve RiskSlimMINLP, and why they stall on risk minimization problems with non-convex constraints and regularizers.

CPA recovers the optimal solution to RiskSlimMINLP by repeatedly solving a mixed-integer programming (MIP) *surrogate problem.* We refer to this surrogate problem as RiskSlimMIP, and provide a MIP formulation in Definition 6.3. RiskSlim-MIP replaces the original loss function $l(\boldsymbol{\lambda})$ with a linear approximation composed of cutting planes. A *cutting plane* or *cut* is a supporting hyperplane to the loss function at a point $\boldsymbol{\lambda}^t \in \mathcal{L}$ with the form

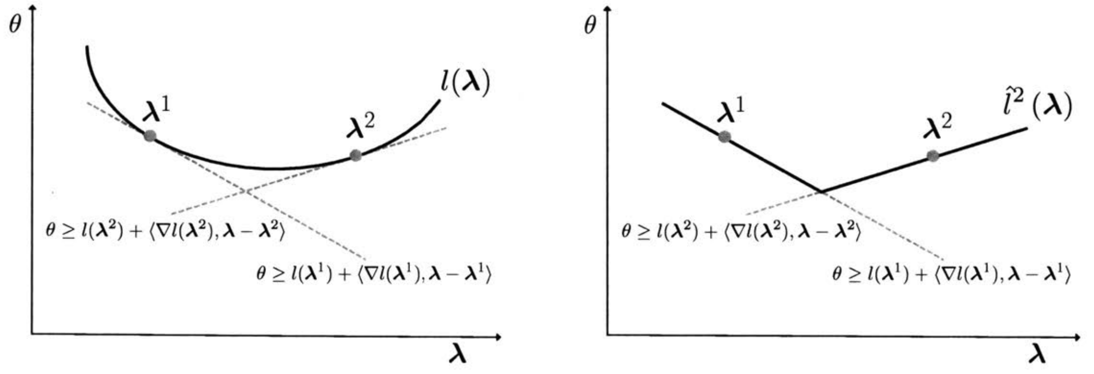
$$l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle$$

where $l(\boldsymbol{\lambda}^t) \in \mathbb{R}_+$ and $\nabla l(\boldsymbol{\lambda}^t) \in \mathbb{R}^d$ are *cut parameters* that represent the value and gradient of the loss function at $\boldsymbol{\lambda}^t$

$$\begin{aligned} l(\boldsymbol{\lambda}^t) &= \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-\langle \boldsymbol{\lambda}^t, y_i\boldsymbol{x}_i\rangle)) \\ \nabla l(\boldsymbol{\lambda}^t) &= \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i\boldsymbol{x}_i}{1+\exp(-\langle \boldsymbol{\lambda}^t, y_i\boldsymbol{x}_i\rangle)}. \end{aligned} \tag{6.3}$$

Multiple cuts can be combined to produce a piecewise linear approximation of the loss function as shown in Figure 6.2. We denote the *cutting plane approximation* of the loss function built using k cuts at the points $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^k$ as

$$\hat{l}^k(\boldsymbol{\lambda}) = \max_{t=1\ldots k} l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t\rangle.$$



θ
$l(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}^1$
$\boldsymbol{\lambda}^2$
$\theta \geq l(\boldsymbol{\lambda}^2) + \langle \nabla l(\boldsymbol{\lambda}^2), \boldsymbol{\lambda} - \boldsymbol{\lambda}^2\rangle$
$\theta \geq l(\boldsymbol{\lambda}^1) + \langle \nabla l(\boldsymbol{\lambda}^1), \boldsymbol{\lambda} - \boldsymbol{\lambda}^1\rangle$
$\boldsymbol{\lambda}$
θ
$\hat{l}^2(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}^1$
$\boldsymbol{\lambda}^2$
$\theta \geq l(\boldsymbol{\lambda}^2) + \langle \nabla l(\boldsymbol{\lambda}^2), \boldsymbol{\lambda} - \boldsymbol{\lambda}^2\rangle$
$\theta \geq l(\boldsymbol{\lambda}^1) + \langle \nabla l(\boldsymbol{\lambda}^1), \boldsymbol{\lambda} - \boldsymbol{\lambda}^1\rangle$
$\boldsymbol{\lambda}$

Figure 6.2: Cutting plane algorithms such as CPA build a piecewise linear approximation of the loss function using cutting planes (i.e., cuts). The plot on the left shows the loss function $l(\boldsymbol{\lambda})$ with cuts at the points $\boldsymbol{\lambda}^1$ and $\boldsymbol{\lambda}^2$. The plot on the right shows the approximate loss function $\hat{l}^2(\boldsymbol{\lambda}) = \max_{t=1,2} l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t\rangle$.

On iteration k, CPA solves the surrogate RiskSlimMIP($\hat{l}^k(\boldsymbol{\lambda})$) whose objective contains the approximate loss $\hat{l}^k(\boldsymbol{\lambda})$. CPA uses the optimizer of the surrogate $(\theta^k, \boldsymbol{\lambda}^k)$ to: (i) improve $\hat{l}^k(\boldsymbol{\lambda})$ with a new cut at $\boldsymbol{\lambda}^k$; (ii) compute bounds on the optimal value of RiskSlimMINLP to check convergence. The upper bound is set as the objective value of the best solution from all iterations $V^{\max} = \min_{t=1\ldots k} l(\boldsymbol{\lambda}^t) + C_0\|\boldsymbol{\lambda}^t\|_0$. The lower bound is set as the optimal value of the surrogate at the last iteration $V^{\min} = \hat{l}^k(\boldsymbol{\lambda}^k) + C_0\|\boldsymbol{\lambda}^k\|_0$.

CPA converges to an ε-optimal solution of RiskSlimMINLP in a finite number of iterations (see Kelley, 1960, for a proof). The cutting plane approximation of a convex loss function improves with each cut:

$$\hat{l}^k(\boldsymbol{\lambda}) \leq \hat{l}^{k+m}(\boldsymbol{\lambda}) \leq l(\boldsymbol{\lambda}) \text{ for all } \boldsymbol{\lambda} \in \mathcal{L} \text{ and } k, m \in \mathbb{N}.$$

Since the cuts at each iteration are not redundant, the lower bound improves monotonically as CPA progresses. Once the optimality gap ε is less than a stopping threshold

$\varepsilon^{\text{stop}}$, CPA terminates and returns an ε-optimal solution $\boldsymbol{\lambda}^{\text{best}}$ to RiskSlimMINLP.

Algorithm 4 Cutting Plane Algorithm (CPA)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^n$                      training data
    $\mathcal{L}$                                  constraint set for RiskSlimMINLP
    $C_0$                                 $\ell_0$ penalty parameter for RiskSlimMINLP
    $\varepsilon^{\text{stop}} \in [0,1]$                  maximum optimality gap of acceptable solution

Initialize
    $k \leftarrow 0$                                iteration counter
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$                      initial approximation of loss function
    $(V^{\min}, V^{\max}) \leftarrow (0, \infty)$          bounds on the optimal value
    $\varepsilon \leftarrow \infty$                               optimality gap
 1: while $\varepsilon > \varepsilon^{\text{stop}}$ do
 2:     $(\theta^k, \boldsymbol{\lambda}^k) \leftarrow$ provably optimal solution to RiskSlimMIP($\hat{l}^k(\cdot)$)
 3:     compute cut parameters $l(\boldsymbol{\lambda}^k)$ and $\nabla l(\boldsymbol{\lambda}^k)$
 4:     $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^k) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^k \rangle\}$     ▷update approximation $\forall \boldsymbol{\lambda}$
 5:     $V^{\min} \leftarrow \theta^k + C_0 \|\boldsymbol{\lambda}^k\|_0$          ▷optimal value of RiskSlimMIP is lower bound
 6:     if $V(\boldsymbol{\lambda}^k) < V^{\max}$ then
 7:         $V^{\max} \leftarrow V(\boldsymbol{\lambda}^k)$                        ▷update upper bound
 8:         $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^k$                           ▷update best solution
 9:     end if
10:     $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$
11:     $k \leftarrow k + 1$
12: end while
Output: $\boldsymbol{\lambda}^{\text{best}}$                   $\varepsilon$-optimal solution to RiskSlimMINLP
```

RiskSlimMIP($\hat{l}(\cdot)$) is a MIP surrogate of RiskSlimMINLP where the loss function $l(\cdot)$ is replaced by the cutting plane approximation $\hat{l}(\cdot)$:

$$\begin{aligned} \min_{\theta, \boldsymbol{\lambda}} \quad & \theta + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\ & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{6.4}$$

We provide a MIP formulation for RiskSlimMIP in Definition 6.3.

Definition 6.3 (MIP Formulation for RiskSlimMIP)
The optimal solution to RiskSlimMIP($\hat{l}^k(\cdot)$) *can be obtained by solving the following MIP formulation:*

$$
\begin{array}{lrcllr}
\min\limits_{\theta,\boldsymbol{\lambda},\alpha} & V & & & & \\
\text{s.t.} & V & = & \theta + C_0 R & \textit{objective value} & \text{(6.5a)} \\
& R & = & \sum_{j=1}^{d} \alpha_j & \ell_0 \textit{ norm} & \text{(6.5b)} \\
& \theta & \geq & l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle \quad t = 1,\dots,k & \textit{loss cuts} & \\
& \lambda_j & \leq & \Lambda_j^{\max} \alpha_j \quad j = 1,\dots,d & \textit{set } \ell_0 \textit{ indicators} & \\
& \lambda_j & \geq & -\Lambda_j^{\min} \alpha_j \quad j = 1,\dots,d & \textit{set } \ell_0 \textit{ indicators} & \\
& & & & & \text{(6.5c)} \\
& V & \in & [V^{\min}, V^{\max}] & \textit{objective bounds} & \text{(6.5d)} \\
& \theta & \in & [L^{\min}, L^{\max}] & \textit{loss bounds} & \text{(6.5e)} \\
& R & \in & \{R^{\min}, \dots, R^{\max}\} & \ell_0 \textit{ bounds} & \text{(6.5f)} \\
& \lambda_j & \in & \{\Lambda_j^{\min}, \dots, \Lambda_j^{\max}\} \quad j = 1,\dots,d & \textit{coefficient bounds} & \\
& \alpha_j & \in & \{0, 1\} \quad j = 1,\dots,d & \ell_0\textit{-indicators} &
\end{array}
$$

The MIP formulation in (6.5) uses $2d + 3$ variables and $k + 2d + 2$. Here, the constraints in (6.5g) bound λ_j to finite integer values. The approximate loss function is captured through cuts constraints in (6.5c). Here, $\theta \in \mathbb{R}_+$ is an auxiliary variable that represents the value of the approximate loss function. The ℓ_0-norm is computed through binary indicator variables $\alpha_j = \mathbb{1}[\lambda_j \neq 0]$ in constraints (6.5c) and (6.5c). The formulation includes two additional variables that will be useful for specialized techniques presented in Section 6.3: V which captures the objective value as per (6.5a); and R, which captures the ℓ_0-norm as per (6.5b). Including these variables will be useful for setting bounds during B&B without introducing additional constraints (i.e. via a *HeuristicCallback* in CPLEX 12.6).

Benefits of Solving RiskSlimMINLP with a Cutting Plane Algorithm

CPA highlights two major benefits of cutting plane algorithms for empirical risk minimization: (i) scalability in the sample size; (ii) control over data-related computation. Since cutting plane algorithms only use the training data to compute cut parameters, which can be achieved using elementary matrix-vector operations in O(Nd) time at each iteration, running time scales linearly in n for fixed d (see Figure 6.3). Since cut parameters are computed in an isolated step (e.g. Step 3 in Algorithm 4), users can further reduce data-related computation by easily customizing their implementation to compute cut parameters efficiently (e.g. via parallelization, or techniques that exploit structural properties of their model class such as those in Section 6.3.3).

CPA also highlights a unique benefit of cutting plane algorithms in our setting. Specifically, it recovers the optimal solution to the non-linear problem RiskSlim-MINLP by iteratively solving a linearized surrogate RiskSlimMIP. In practice, this allows us to fit risk scores with a MIP solver instead of a MINLP solver. As shown in

Figure 6.6, this can substantially improve our ability to solve RiskSlimMINLP since MIP solvers typically exhibit better off-the-shelf performance than MINLP solvers (as MIP solvers have better implementations of branch-and-bound, and MINLP solvers are designed to handle a far more diverse set of optimization problems).

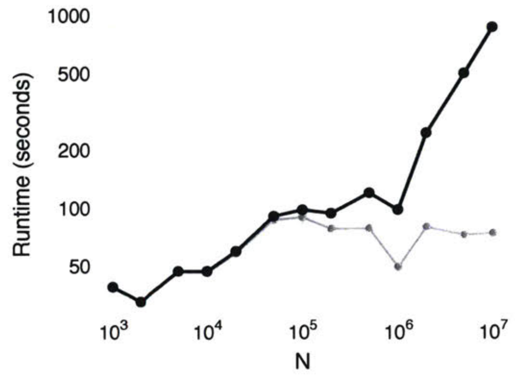


Figure 6.3: CPA runtime (log-scale) on RiskSlimMINLP instances for simulated datasets with $d = 10$ and $n \in [10^3, 10^8]$ (see Appendix B.3 for details). As n increases, total running time (black) scales at $O(n)$, which reflects the time to compute cut parameters. Solver time (grey) remains roughly constant.

Cutting Plane Algorithms Stall in Non-Convex Settings

Cutting plane algorithms for empirical risk minimization (Joachims et al., 2009; Franc and Sonnenburg, 2009; Teo et al., 2009) are similar to CPA in that they solve a surrogate problem at each iteration (i.e., on Step 5 of Algorithm 4). When these algorithms are used to solve convex problems, the surrogate problem is convex and therefore tractable. When these algorithms are applied to problems with non-convex regularizers or constraints, the surrogate problems are non-convex and may require an unreasonable amount of time to solve to optimality (especially on on higher-dimensional problems). In practice, this prevents the algorithm from improving the cutting plane approximation and computing a valid lower bound. We refer to this behavior as *stalling.*

There is no easy fix to prevent cutting plane algorithms such as CPA from stalling in non-convex settings. This is because they need a *provably* optimal solution at each iteration to compute a valid lower bound (i.e., a solution with an optimality gap of 0.0%). If, for example, CPA only solved RiskSlimMIP until it found a feasible solution with a non-zero optimality gap, the resulting lower bound could exceed the true optimal value, leading the algorithm to terminate early and return a suboptimal solution. Seeing how stalling is related to the mechanism to check convergence, a tempting (but flawed) solution is to use a cutting plane algorithm that adds cuts at central points of RiskSlimMIP (e.g., the center of gravity as in Levin 1965, or the analytic center as in Atkinson and Vaidya 1995), as these algorithms are guaranteed to converge in a fixed number of iterations and do not require computing a lower bound. In this case, however, stalling would still occur as we would have to solve a non-convex optimization problem at each iteration to compute central points.

In Figure 6.4, we provide insight into the stalling behavior of CPA. Here, the first few iterations are quick as the surrogate problem is easy to solve to optimality when it contains a trivial approximation of the loss function. However, the surrogate becomes increasingly difficult to optimize with each iteration. On the $d = 10$ instance, CPA does not stall as the MIP solver is powerful enough to solve the surrogate problem RiskSlimMIP for all iterations. On the $d = 20$ instance, however, the time to solve RiskSlimMIP increases exponentially with each iteration and CPA stalls on iteration $k = 87$ as it attempts to optimize the surrogate MIP. In this case, the best feasible solution that we recover after 6 hours has a large optimality gap as well as a highly suboptimal loss (which makes sense as the solution optimizes a cutting plane approximation that uses at most 86 cuts). Given that the value of the loss is closely related to the performance of the model, this means that the risk score we obtain after 6 hours performs poorly.

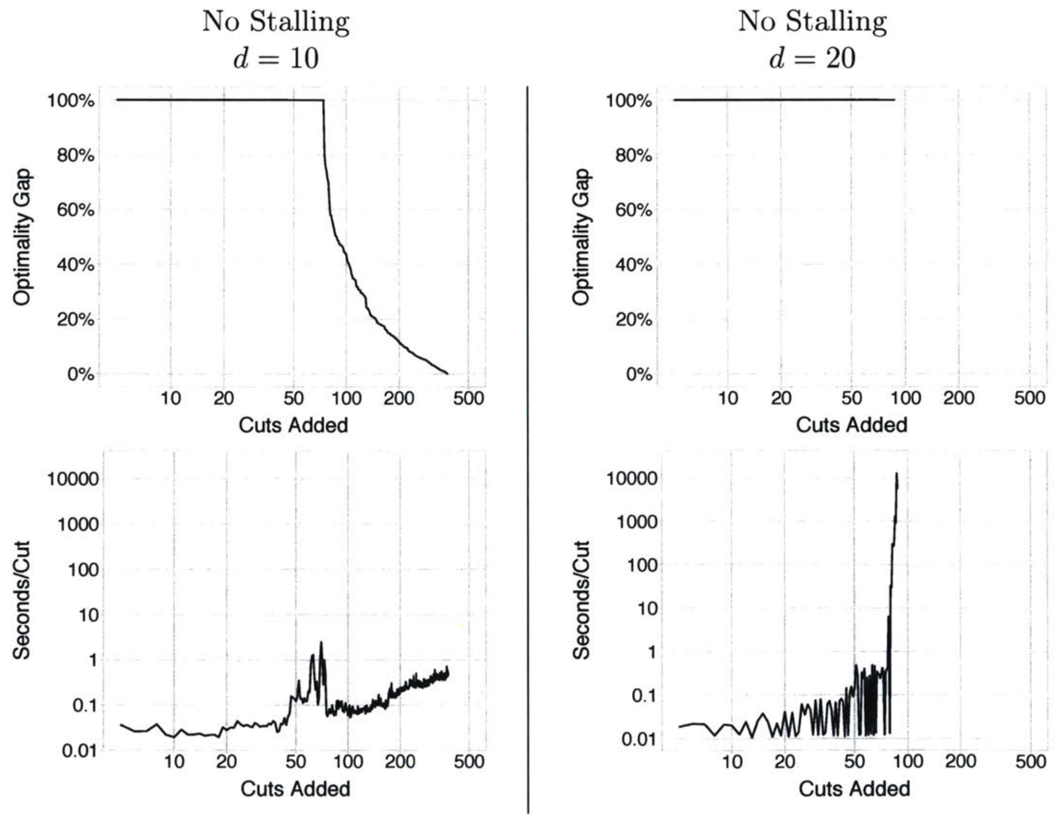


Figure 6.4: Progress of CPA on RiskSlimMINLP for simulated datasets with $d = 10$ (left) and $d = 20$ (right) and $n = 50{,}000$ (see Appendix B.3 for details). We show the optimality gap (top) and time per iteration (bottom, in log-scale) for each iteration over 6 hours. CPA solves the $d = 10$ instance, but stalls on the $d = 20$ instance as the time to solve RiskSlimMIP to optimality increases exponentially starting on iteration 86, and the best solution obtained after 6 hours corresponds to a risk score with poor performance.

6.2.2 Lattice Cutting Plane Algorithm

In order to avoid the stalling behavior of existing cutting plane algorithms in non-convex settings, we solve the risk score problem using the *lattice cutting plane algorithm* (LCPA; Algorithm 5).

LCPA is a cutting plane algorithm that recovers the optimal solution to RISKSLIMMINLP via *branch-and-bound* (B&B) search. The search recursively splits the feasible region of RISKSLIMMINLP into disjoint partitions, discarding partitions that are infeasible or provably suboptimal. LCPA solves a *surrogate linear program* (LP) over each partition. In this approach, the cutting plane approximation is updated whenever the surrogate LP yields an integer feasible solution. The lower bound is set as the smallest possible value of the surrogate LP over the remaining search region.

As shown in Figure 6.5, LCPA (in red) does not stall. This is because, unlike CPA, LCPA does not need to optimize a non-convex surrogate to add cuts and compute a valid lower bound. Even so, LCPA retains the key benefits of CPA such as: scalability in the sample size, control over data-related computation, and the ability to use a MIP solver.

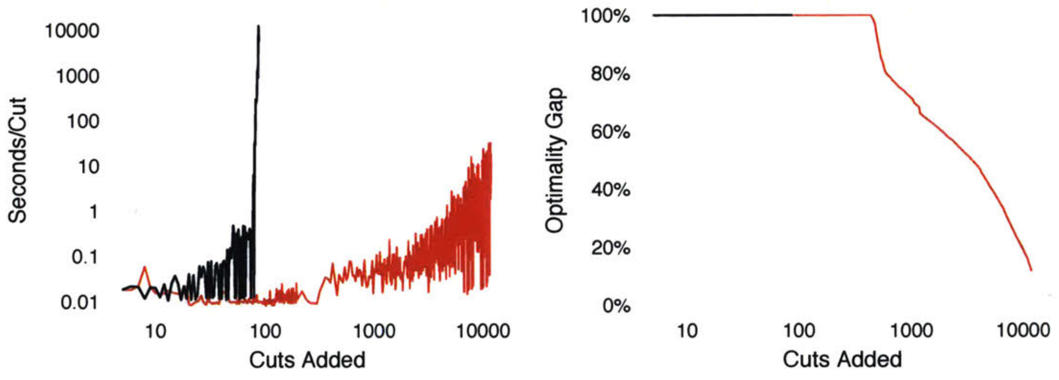


Figure 6.5: Progress of LCPA (red) and CPA (black) on the RISKSLIMMINLP instance with $d = 20$ from Figure 6.4. Unlike CPA, LCPA does not stall. The algorithm finds a solution corresponding to a high-quality risk score in 9 minutes after adding 4655 cuts, and the optimal solution in 234 minutes after adding 11,665 cuts. The remaining time is used to reduce the optimality gap.

In what follows, we describe the main elements of LCPA in greater detail.

Branch and Bound Search

In Algorithm 5, we represent the state of the B&B search using a B&B tree. This tree is composed of nodes (i.e. leaves) in the *node set* $\mathcal{N}$. Each *node* $(\mathcal{R}^t, v^t) \in \mathcal{N}$ consists of a *partition* of the convex hull of the coefficient set $\mathcal{R}^t \subseteq \operatorname{conv}(\mathcal{L})$, and a lower bound for the optimal value of the surrogate over this partition, v^t.

Each iteration of LCPA starts by removing a node $(\mathcal{R}^t, v^t)$ from the node set $\mathcal{N}$ and solving the surrogate over $\mathcal{R}^t$. The next steps depend on the feasibility of RISKSLIMLP$(\hat{l}^k(\cdot), \mathcal{R}^t)$:

- If $\textsc{RiskSlimLP}(\hat{l}^k(\cdot), \mathcal{R}^t)$ is infeasible, the node is discarded.
- If $\textsc{RiskSlimLP}(\hat{l}^k(\cdot), \mathcal{R}^t)$ yields an integer solution $\boldsymbol{\lambda}^{\text{LP}} \in \mathcal{L}$, LCPA updates the cutting plane approximation $\hat{l}^k(\cdot)$ with a cut at $\boldsymbol{\lambda}^{\text{LP}}$ in Step 8.
- If $\textsc{RiskSlimLP}(\hat{l}^k(\cdot), \mathcal{R}^t)$ yields a continuous solution $\boldsymbol{\lambda}^{\text{LP}} \notin \mathcal{L}$, then LCPA splits the partition $\mathcal{R}^t$ into disjoint subsets $\mathcal{R}'$ and $\mathcal{R}''$. Each subset is paired with the optimal value of the surrogate LP to yield the child nodes $(\mathcal{R}', v^{\text{LP}})$ and $(\mathcal{R}'', v^{\text{LP}})$. The child nodes are added back into $\mathcal{N}$ in Step 18.

The search process uses two rules that are typically provided by a MIP solver:

- SelectNode, which takes as input the node set $\mathcal{N}$ and outputs a node $(\mathcal{R}^t, v^t)$ (e.g., the node with the smallest v^t).
- SplitPartition, which takes as input a partition $\mathcal{R}^t$ and the current solution $\boldsymbol{\lambda}^{\text{LP}}$ and outputs disjoint partitions that do not cover $\mathcal{R}^t$ (e.g. split on a fractional component of the solution λ_j^{LP}, which returns $\mathcal{R}' = \{\boldsymbol{\lambda} \in \mathcal{R}^{\text{LP}} \,|\, \lambda_j^{\text{LP}} \geq \lceil \lambda_j^{\text{LP}} \rceil\}$ and $\mathcal{R}'' = \{\boldsymbol{\lambda} \in \mathcal{R}^{\text{LP}} \,|\, \lambda_j^{\text{LP}} \leq \lfloor \lambda_j^{\text{LP}} \rfloor\}$). The output conditions ensure that: (i) the partitions of all nodes in the node set remain disjoint; (ii) the search region shrinks even if the solution to the surrogate is not integer feasible; (iii) the number of nodes is finite.

Algorithm 5 Lattice Cutting Plane Algorithm (LCPA)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                                  training data
    $\mathcal{L}$                                                      coefficient set for RiskSlimMINLP
    $C_0$                                                              $\ell_0$ penalty parameter
    $\varepsilon^{\text{stop}} \in [0,1]$                             optimality gap of acceptable solution
    SelectNode                  rule to pick a node from a node set (provided by MIP solver)
    SplitPartition      rule to split a partition into disjoint subsets (provided by MIP solver)

Initialize
    $k \leftarrow 0$                                                   number of cuts
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$                 cutting plane approximation of loss function
    $(V^{\min}, V^{\max}) \leftarrow (0, \infty)$                      bounds on the optimal value
    $\varepsilon \leftarrow \infty$                                    optimality gap
    $\mathcal{R}^0 \leftarrow \text{conv}(\mathcal{L})$                partition for initial node
    $v^0 \leftarrow V^{\min}$                                          lower bound for initial node
    $\mathcal{N} \leftarrow \{(\mathcal{R}^0, v^0)\}$                  initial node set

1: while $\varepsilon > \varepsilon^{\text{stop}}$ do
2:     $(\mathcal{R}^t, v^t) \leftarrow$ SelectNode$(\mathcal{N})$             ▷$t$ is index of removed node
3:     solve RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{R}^t)$
4:     $\boldsymbol{\lambda}^{\text{LP}} \leftarrow$ coefficients from optimal solution to RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{R}^t)$
5:     $v^{\text{LP}} \leftarrow$ optimal value of RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{R}^t)$
6:     if optimal solution is integer feasible then
7:         compute cut parameters $l(\boldsymbol{\lambda}^{\text{LP}})$ and $\nabla l(\boldsymbol{\lambda}^{\text{LP}})$
8:         $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^{\text{LP}}) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^{\text{LP}} \rangle\}$      ▷update approximation $\forall \boldsymbol{\lambda}$
9:         if $v^{\text{LP}} < V^{\max}$ then
10:            $V^{\max} \leftarrow v^{\text{LP}}$                          ▷update lower bound
11:            $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^{\text{LP}}$        ▷update best solution
12:            $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(\mathcal{R}^s, v^s) \mid v^s \geq V^{\max}\}$     ▷prune suboptimal nodes
13:        end if
14:        $k \leftarrow k + 1$
15:    else if optimal solution is not integer feasible then
16:        $(\mathcal{R}', \mathcal{R}'') \leftarrow$ SplitPartition$(\mathcal{R}^t, \boldsymbol{\lambda}^{\text{LP}})$     ▷$\mathcal{R}'$, $\mathcal{R}''$ are disjoint subsets of $\mathcal{R}^t$
17:        $(v', v'') \leftarrow (v^{\text{LP}}, v^{\text{LP}})$                ▷$v^{LP}$ is lower bound for $\mathcal{R}', \mathcal{R}''$
18:        $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\mathcal{R}', v'), (\mathcal{R}'', v'')\}$        ▷add child nodes to $\mathcal{N}$
19:    end if
20:    $V^{\min} \leftarrow \min_{\mathcal{N}} v^s$        ▷lower bound is smallest lower bound among nodes in $\mathcal{N}$
21:    $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$                           ▷update optimality gap
22: end while
Output: $\boldsymbol{\lambda}^{\text{best}}$                          $\varepsilon$-optimal solution to RiskSlimMINLP
```

RiskSlimLP$(\hat{l}(\boldsymbol{\lambda}), \mathcal{R})$ is the LP relaxation of RiskSlimMIP$(\hat{l}(\boldsymbol{\lambda}))$ over the partition $\mathcal{R} \subseteq \text{conv}(\mathcal{L})$:

$$
\begin{aligned}
\min_{\theta, \boldsymbol{\lambda}, \boldsymbol{\alpha}} \quad & \theta + C_0 \sum_{j=1}^{d} \alpha_j \\
\text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{R} \\
& \theta \geq \hat{l}(\boldsymbol{\lambda}) \\
& \alpha_j = \max(\lambda_j, 0)/\Lambda_j^{\max} + \min(\lambda_j, 0)/\Lambda_j^{\min} \text{ for } j = 1 \ldots d.
\end{aligned}
\tag{6.6}
$$

Definition 6.4 (Formulation for RiskSlimLP)
The optimal solution to RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{R})$ *can be obtained by solving the following linear programming formulation:*

$$
\begin{array}{rrcllr}
\min\limits_{\theta,\boldsymbol{\lambda},\alpha} & V & & & & \\
\text{s.t.} & V & = & \theta + C_0 R & & \textit{objective value} \quad (6.7a) \\
& R & = & \displaystyle\sum_{j=1}^{d} \alpha_j & & \ell_0 \textit{ norm} \quad (6.7b) \\
& \theta & \geq & l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle & t = 1,\ldots,k & \textit{loss cuts} \\
& \lambda_j & \leq & \Lambda_j^{\max} \alpha_j & j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicators} \\
& \lambda_j & \geq & -\Lambda_j^{\min} \alpha_j & j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicators} \\
& & & & & (6.7c) \\
& \boldsymbol{\lambda} & \in & \mathcal{R} & & \textit{B\&B partition} \quad (6.7d) \\
& V & \in & [V^{\min}, V^{\max}] & & \textit{objective bounds} \quad (6.7e) \\
& \theta & \in & [L^{\min}, L^{\max}] & & \textit{loss bounds} \quad (6.7f) \\
& R & \in & [R^{\min}, R^{\max}] & & \ell_0 \textit{ bounds} \quad (6.7g) \\
& \lambda_j & \in & [\Lambda_j^{\min}, \Lambda_j^{\max}] & j = 1,\ldots,d & \textit{coefficient bounds} \\
& \alpha_j & \in & [0, 1] & j = 1,\ldots,d & \ell_0 \textit{ indicators}
\end{array}
$$

Convergence

LCPA checks convergence using bounds on the optimal value of RiskSlimMINLP. The upper bound $V^{\max}$ is set as the objective value of the best integer feasible solution in Step 11. The lower bound $V^{\min}$ is set as the smallest lower bound among all nodes in Step 20. This quantity is a lower bound on the optimal value of the surrogate over the *remaining search region* $\bigcup_t \mathcal{R}^t$; that is, the optimal value of RiskSlimLP$(\hat{l}^k(\cdot), \bigcup_t \mathcal{R}^t)$. Thus, $V^{\min}$ improves when we add cuts or reduce the remaining search region.[1]

Each iteration of LCPA reduces the remaining search region as it either finds an integer feasible solution, identifies an infeasible partition, or splits a partition into disjoint subsets. Thus, $V^{\min}$ increases monotonically as the search region becomes smaller, and cuts are added at integer feasible solutions. Likewise, $V^{\max}$ decreases monotonically as the search is guaranteed to find the optimal solution. Since there are a finite number of nodes in the worst-case, LCPA terminates after a finite number of iterations and returns an ε-optimal solution to the risk score problem (see 6.6).

[1]Observe that the LCPA lower bound is typically weaker than the CPA lower bound. Specifically, the CPA lower bound is built by solving a surrogate problem that requires an integer feasible solution while the LCPA lower bound is built by solving a surrogate problem that does not require an integer feasible solution. As such, on instances where it does not stall, CPA may converge faster than the LCPA since it produces a stronger lower bound at each iteration.

Remark 6.5 (Worst-Case Data-Related Computation for LCPA)
Given any training dataset $(\boldsymbol{x}_i, y_i)_{i=1}^n$, any trade-off parameter $C_0 > 0$, and any finite coefficient set $\mathcal{L} \subset \mathbb{Z}^{d+1}$, Algorithm 5 will return an optimal solution to the risk score problem after computing at most $|\mathcal{L}|$ cutting planes.

Remark 6.6 (Worst-Case Termination for LCPA)
Given any training dataset $(\boldsymbol{x}_i, y_i)_{i=1}^n$, any trade-off parameter $C_0 > 0$, and any finite coefficient set $\mathcal{L} \subset \mathbb{Z}^{d+1}$, Algorithm 5 will return an optimal solution to the risk score problem after processing at most $2^{D(\mathcal{L})} - 1$ nodes, where

$$D(\mathcal{L}) = \prod_{j=0}^{d} (\Lambda_j^{\max} - \Lambda_j^{\min} + 1).$$

Implementation and Optional Improvements

We implement LCPA using a MIP solver that provides *control callbacks*, such as CPLEX. The solver handles all B&B related steps in Algorithm 5 and control callbacks let update the cutting plane approximation by intervening in the search. In a basic implementation, we use a control callback to intervene when Algorirthm 5 reaches Step 6. Our code retrieves the integer feasible solution, computes the cut parameters, adds a cut, and returns control back to solver by Step 9.

LCPA can be substantially improved using techniques that we present in Section 6.3. In what follows, we describe these techniques at a high level.

- *Polishing Heuristic.* We polish all integer feasible solutions that are found by the MIP solver in Step 6 using a technique that we call discrete coordinate descent (Algorithm 6; Section 6.3.1). Polished solutions may update the best solution found by LCPA, which results in stronger upper bounds over the course of LCPA, and reduces the time for LCPA to return a high-quality solution.
- *Rounding Heuristic.* We produce new integer feasible solutions using a new rounding technique that we call sequential rounding (Algorithm 7; Section 6.3.1). Specifically, we round the continuous solution to the surrogate LP in Step 15, and polish the resulting integer feasible solution using discrete coordinate descent. Rounded solutions may improve the best solution found by LCPA, which produces stronger upper bounds over the course of LCPA, and reduces the time to find a high-quality solution.
- *Bounds on Objective Terms.* We design a procedure to strengthen bounds on the optimal values of the objective function, loss function, and number of non-zero coefficients (Algorithm 8; Section 6.3.2). We call this procedure whenever the solver updates the upper bound in Step 11 or the lower bound in Step 20. Using this procedure improves the lower bound and the optimality gap over the course of LCPA.
- *Initialization Procedure*: Since solution quality is affected by the fidelity of the approximate loss function, and LCPA only adds cuts at integer feasible solutions,

early solutions from LCPA may correspond to low-quality models. In Section 6.3.2, we present an initialization procedure to mitigate this issue by quickly generating a set of cutting planes to warm-start LCPA. This reduces the time required for LCPA to return a high-quality solution, and improves both the upper and the lower bound over the course of LCPA.

6.2.3 Experimental Comparison with an MINLP Solver

To illustrate the computational properties of CPA and LCPA, we used each algorithm to solve difficult instances of RiskSlimMINLP on simulated datasets with varying dimensions d and sample sizes n (see Appendix B.3 for details). As a baseline, we also solved these using 3 MINLP algorithms as implemented in a commercial MINLP solver[2] [3].

In Figure 6.6, we compare all methods in terms of the following metrics:

- *time to find a near-optimal solution*: that is, a solution whose loss is within 10% of the optimal loss. This measures the time needed to fit a risk score with good AUC and risk calibration, but without a proof of optimality.
- *optimality gap of best solution at termination*, which is 0.0% if and only if the method finds the optimal solution *and* provides the proof of optimality within a 6-hour time limit.
- *% of time spent on data-related computation*, meaning the time spent evaluating the value/gradient/Hessian of the loss function.

As shown, a basic implementation of LCPA finds the optimal solution to RiskSlim-MINLP for almost all instances, and pairs it with a small optimality gap. This performance reflects a basic implementation of LCPA on difficult instances of RiskSlim-MINLP. We improve the performance of LCPA with regards to these metrics using specialized techniques in Section 6.3, and use LCPA to recover the optimal solution to larger real-world problems in Section 6.4 and Chapters 7 and 8.

CPA performs similarly to LCPA on low-dimensional instances. On instances with $d \geq 15$, however, CPA stalls after a few iterations and ultimately returns a highly suboptimal solution that corresponds to a risk score with poor performance.

In comparison to the cutting plane algorithms, the MINLP solver could only handle instances of RiskSlimMINLP for datasets with limited sample sizes and/or dimensions – regardless of the algorithm that we used to solve the problem. On large instances, the solver spends the majority of its time dealing with operations that involve data-related computation, fails to converge in the 6-hour time limit, and fails to yield high quality feasible solution. Seeing how a MINLP solver is designed to solve a diverse set of optimization problems, it is unlikely that it can identify and exploit the structure of the risk score problem in the same way as the cutting plane algorithms.

[2]Since all 3 MINLP algorithms behave similarly, we only show the best performing algorithm in Figure 6.6 (i.e., ActiveSetMINLP), and include results for other MINLP algorithms in Appendix B.3.

[3]There exist several off-the-shelf MINLP solvers (see Bussieck and Vigerske, 2010, for a list). We used Artelsys Knitro 9.0 (i.e., an updated version of the MINLP solver from Byrd et al., 2006) because it let us: (i) monitor and minimize data-related computation, by letting us write our own functions to evaluate the objective, its gradient and Hessian; and (ii) solve LP subproblems using CPLEX, which is the same solver used in CPA and LCPA.

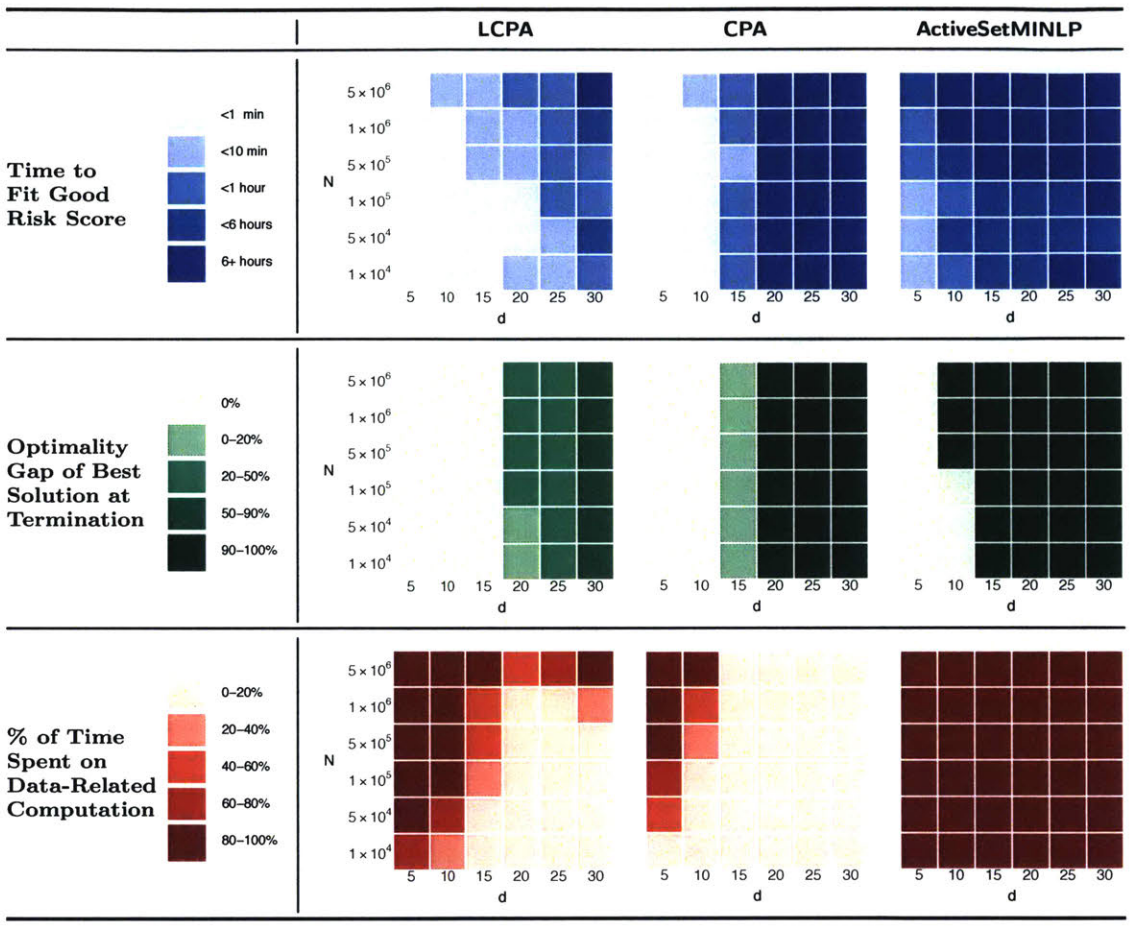


Figure 6.6: Performance of algorithms on hard instances of RiskSlimMINLP for simulated datasets with varying dimensions d and sample sizes n (see Appendix B.3 for details). ActiveSetMINLP fails on instances with large d or n as it struggles with data-related computation. In comparison, CPA and LCPA scale linearly in n – that is, if they can solve an instance for fixed d, then they can solve instances for larger n in $O(n)$ time. CPA stalls on all instances when $d \geq 15$ and returns highly suboptimal solutions when $d \geq 20$. In contrast, LCPA does not stall on any instances, and recovers a near optimal solution in all cases, pairing them with optimality gaps between 0.0 - 62.2% depending on d. Results for LCPA reflect the performance for basic implementation without the improvements in Sections 6.3 and 6.3.3. We include results for additional MINLP algorithms in Appendix B.3 as they perform similarly to ActiveSetMINLP.

6.3 Algorithmic Improvements

In this section, we describe specialized techniques to improve the performance of the lattice cutting plane algorithm (LCPA) on the risk score problem (RISKSLIMMINLP).

6.3.1 Generating Feasible Solutions

In what follows, we present two techniques to generate and improve integer feasible solutions for RISKSLIMMINLP. We pair these techniques with LCPA to produce new integer solutions by rounding continuous solutions and to polish integer feasible solutions obtained by rounding, or found by the MIP solver. In Section 6.4, we make further use of these techniques to design new heuristics for risk scores.

Discrete Coordinate Descent

Discrete coordinate descent (DCD) is a technique for polishing integer feasible solutions (Algorithm 6). It takes as input an integer feasible solution $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_d) \in \mathcal{L}$ and iteratively moves along a single dimension j to attain an integer feasible solution with a lower objective value. The descent direction at each iteration is chosen greedily as the dimension that minimizes the objective value $j \in \operatorname{argmin} V(\boldsymbol{\lambda} + \delta_j e_j)$.

DCD terminates once it can no longer strictly improve the objective value along any dimension. This eliminates the possibility of cycling, and ensures that it terminates after a finite number of iterations. The polished solution satisfies a type of local optimality guarantee in the discrete setting: formally, the solution is *1-opt* with respect to the objective, meaning that the objective cannot improve in any single dimension (see e.g., Park and Boyd, 2015, for a technique to find a 1-opt point for a different optimization problem).

In practice, the most expensive part of DCD involves determining a step-size $\delta_j \in \Delta_j$ that minimizes the objective in dimension j (Step 4 of Algorithm 6). The computation for this step can be significantly reduced by using a bisection search algorithm that exploits the convexity of the loss function, to reduce the number of loss function evaluations. This approach requires a total of $\log_2(|\mathcal{L}_j|)Nd$ flops per iteration, which is an improvement over the $(|\mathcal{L}_j| - 1)Nd$ flops per iteration required from the naïve exhaustive search strategy (i.e., where we evaluate the loss for all $|\mathcal{L}_j| - 1$ feasible values of λ_j other than current value).

In Figure 6.7, we show how DCD can improve the performance of LCPA when we use it to polish feasible solutions found by the MIP solver (Step 6 of Algorithm 5).

Algorithm 6 Discrete Coordinate Descent (DCD)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^n$                training data
    $\mathcal{L}$                                   constraint set for RiskSlimMINLP
    $C_0$                                           $\ell_0$ penalty parameter for RiskSlimMINLP
    $\boldsymbol{\lambda} \in \mathcal{L}$          integer feasible solution to RiskSlimMINLP

Initialize
    $V \leftarrow V(\boldsymbol{\lambda})$          objective value at current solution
    $\mathcal{J} \leftarrow \{0, \ldots, d\}$       valid search dimensions
 1: repeat
 2:     for $j \in \mathcal{J}$ do
 3:         $\Delta_j \leftarrow \{\delta \in \mathbb{Z} \mid \boldsymbol{\lambda} + \delta e_j \in \mathcal{L}\}$     ▷list feasible moves along dim j
 4:         $\delta_j \leftarrow \text{argmin}_{\delta \in \Delta_j} V(\boldsymbol{\lambda} + \delta)$     ▷find best move along dim j
 5:         $v_j \leftarrow V(\boldsymbol{\lambda} + \delta_j e_j)$     ▷store objective value for best move along dim j
 6:     end for
 7:     $m \leftarrow \text{argmin}_{j \in \mathcal{J}} v_j$     ▷descend along dim that minimizes objective
 8:     if $v_m < V$ then
 9:         $V \leftarrow v_m$
10:         $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \delta_m e_m$
11:         $\mathcal{J} \leftarrow \{0, \ldots, d\} \setminus \{m\}$     ▷ignore dim m on next iteration
12:     end if
13: until $v_m \geq V$
Output: $\boldsymbol{\lambda}$, solution that is 1-opt with respect to the objective of RiskSlimMINLP
```

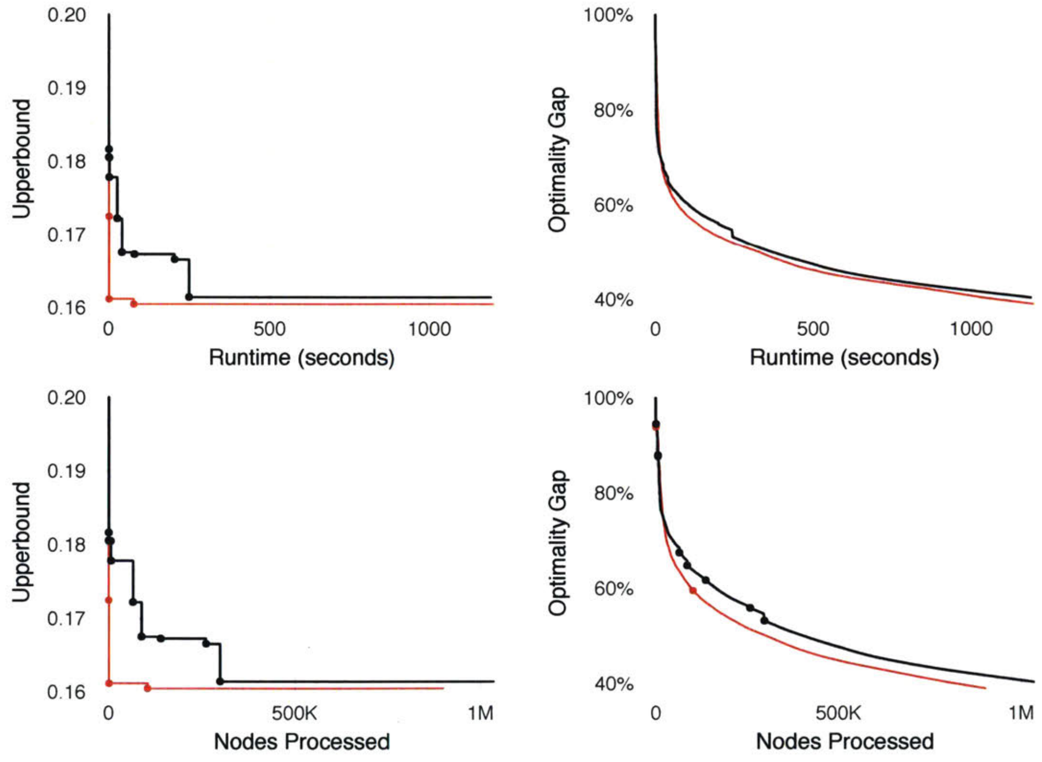


Figure 6.7: Performance profile of LCPA in a basic implementation (black) and with DCD (red). We use DCD to polish every integer feasible solution found by the MIP solver whose objective value is within 10% of the current upper bound. We plot large points to show when LCPA updates the incumbent solution. Results reflect performance on RISKSLIMMINLP for a simulated dataset with $d = 30$ and $n = 50{,}000$ (see Appendix B.3 for details).

Sequential Rounding

Sequential rounding (Algorithm 7) is a technique to round continuous solutions in a way that accounts for objective of RiskSlimMINLP. Given a continuous solution $\boldsymbol{\lambda}^{\text{cts}} \in \operatorname{conv}(\mathcal{L})$, SequentialRounding rounds one component at a time, either up or down, in a way that minimizes the objective value of RiskSlimMINLP. In comparison to naïve rounding, which returns the closest rounding from a set of 2^{d+1} possible roundings, SequentialRounding returns a rounding that optimizes the value of the objective function.

On Step k, the technique has already rounded k components, and needs to round one of the remaining $d-k+1$ components to either $\lceil \lambda_j^{\text{cts}} \rceil$ or $\lfloor \lambda_j^{\text{cts}} \rfloor$. To this end, it computes the objective value of each feasible component-direction pair, and chooses the best one. The minimization on Step k requires $\sum_{i=1}^{i=d-k+1} 2i = (d-k+1)(d-k+2)$ evaluations of the loss function. Thus, given that there are $d+1$ steps, the technique terminates after $\frac{1}{3}d(d^2+3d+2)$ evaluations of the loss function.

In Figure 6.8, we show the impact of using SequentialRounding in LCPA to round the continuous solution to RiskSlimLP when the lower bound changes (i.e., Step 3 of Algorithm 5). We then polish the rounded solution via DCD to increase the likelihood that it will update the incumbent solution. As shown, this reduces the time needed for LCPA to produce a higher quality risk score, and attain a lower optimality gap.

Algorithm 7 SequentialRounding

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^n$                                training data
    $\mathcal{L}$                                    constraint set for RiskSlimMINLP
    $C_0$                              $\ell_0$ penalty parameter for RiskSlimMINLP
    $\boldsymbol{\lambda} \in \operatorname{conv}(\mathcal{L})$                        feasible solution to RiskSlimLP

Initialize
    $\mathcal{J}^{\text{cts}} \leftarrow \{0, \ldots, d\}$           index set of features that need to be rounded
 1: repeat
 2:     $\boldsymbol{\lambda}^{\text{floor}(j)} \leftarrow (\lambda_1, \ldots, \lfloor \lambda_j \rfloor, \ldots, \lambda_d)$ for all $j \in \mathcal{J}^{\text{cts}}$
 3:     $\boldsymbol{\lambda}^{\text{ceil}(j)} \leftarrow (\lambda_1, \ldots, \lceil \lambda_j \rceil, \ldots, \lambda_d)$ for all $j \in \mathcal{J}^{\text{cts}}$
 4:     $v^{\text{floor}} \leftarrow \min_{j \in \mathcal{J}^{\text{cts}}} V(\boldsymbol{\lambda}^{\text{floor}(j)})$
 5:     $v^{\text{ceil}} \leftarrow \min_{j \in \mathcal{J}^{\text{cts}}} V(\boldsymbol{\lambda}^{\text{ceil}(j)})$
 6:     if $v^{\text{floor}} \leq v^{\text{ceil}}$ then
 7:         $k \leftarrow \operatorname{argmin}_{j \in \mathcal{J}} V(\boldsymbol{\lambda}^{\text{floor}(j)})$
 8:         $\lambda_k \leftarrow \lfloor \lambda_k \rfloor$
 9:     else
10:         $k \leftarrow \operatorname{argmin}_{j \in \mathcal{J}} V(\boldsymbol{\lambda}^{\text{ceil}(j)})$
11:         $\lambda_k \leftarrow \lceil \lambda_k \rceil$
12:     end if
13:     $\mathcal{J}^{\text{cts}} \leftarrow \mathcal{J}^{\text{cts}} \setminus \{k\}$
14: until $\mathcal{J}^{\text{cts}} = \varnothing$
Output: $\boldsymbol{\lambda} \in \mathcal{L}$, integer feasible solution
```

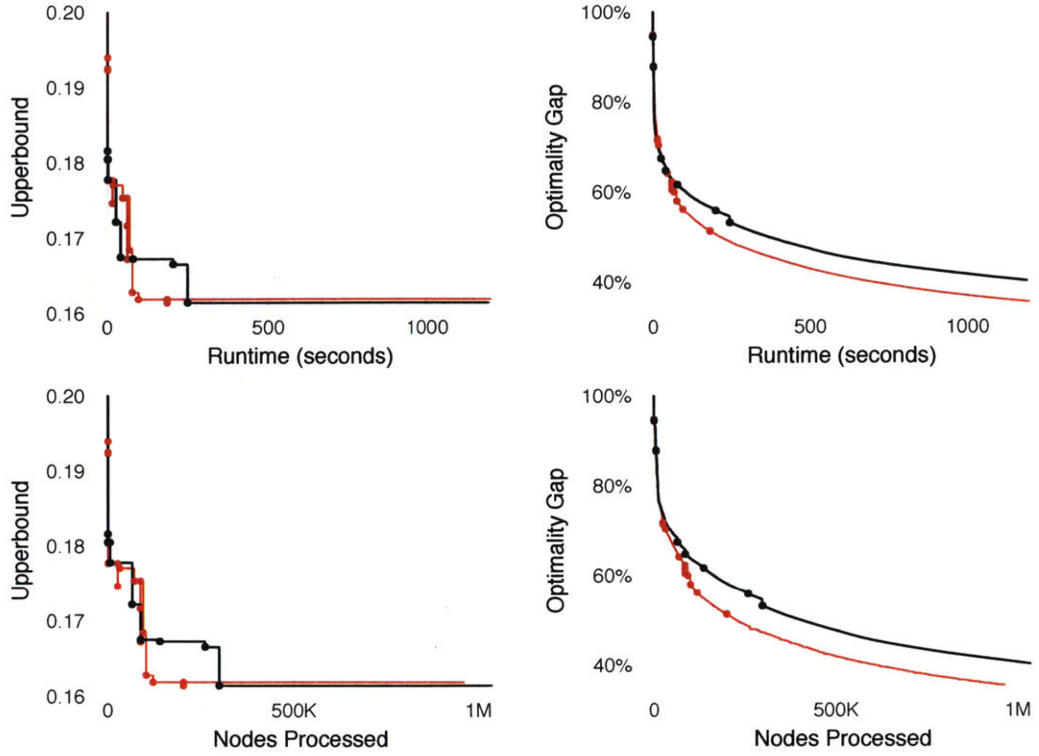


Figure 6.8: Performance of LCPA in a basic implementation (black) and with SequentialRounding and DCD. We call SequentialRounding in Step 15 for continuous solutions to RISKSLIMLP, and the integer feasible solution using DCD. We plot large points to show when LCPA updates the incumbent solution. Results reflect performance on RISKSLIM-MINLP for a simulated dataset with $d = 30$ and $n = 50{,}000$ (see Appendix B.3 for details).

6.3.2 Reducing the Optimality Gap

We will present two techniques to reduce the optimality gap produced by LCPA. These techniques require that we run LCPA with a formulation of RiskSlimLP that includes *auxiliary variables* to bound the values of the objective function, loss function, and ℓ_0-penalty at the optimal solution $\boldsymbol{\lambda}^*$. Our proposed techniques are designed to strengthen the bounds on these quantities over the course of a B&B search. In doing so, they effectively restrict the search region without discarding the optimal solution, thereby improving the lower bound and reducing the optimality gap.

Chained Updates

Chained updates (Algorithm 8) is a simple procedure to strengthen the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$ over the course of LCPA. It requires no assumptions, is easy to implement, and involves minimal computation.

Initial Bounds on the Objective, Loss and Model Size

To initialize the procedure, we need bounds that can be computed using only the training data $(\boldsymbol{x}_i, y_i)_{i=1}^n$ and the coefficient set $\mathcal{L}$. We start with Proposition 6.7, which bounds on the logistic loss by exploiting the fact that $\mathcal{L}$ is bounded.

Proposition 6.7 (Bounds on Logistic Loss over a Bounded Coefficient Set)
Let $(\boldsymbol{x}_i, y_i)_{i=1}^n$ denote a dataset where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1,+1\}$ for $i = 1, \ldots, n$. Consider the value of the normalized logistic loss for a linear classifier with coefficients $\boldsymbol{\lambda} \in \mathcal{L} \subset \mathbb{R}^d$

$$l(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \log(1 + \exp(-\langle \boldsymbol{\lambda}, y_i\boldsymbol{x}_i\rangle)).$$

If the coefficient set $\mathcal{L}$ is bounded, then $l(\boldsymbol{\lambda}) \in [L^{\min}, L^{\max}]$ for all $\boldsymbol{\lambda} \in \mathcal{L}$ where

$$\begin{aligned}
L^{\min} &= \frac{1}{n}\sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\max})\right) + \frac{1}{n}\sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\min})\right),\\
L^{\max} &= \frac{1}{n}\sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\min})\right) + \frac{1}{n}\sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\max})\right),\\
s_i^{\min} &= \min_{\boldsymbol{\lambda}\in\mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle \text{ for } i = 1, \ldots, n,\\
s_i^{\max} &= \max_{\boldsymbol{\lambda}\in\mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle \text{ for } i = 1, \ldots, n.
\end{aligned}$$

The value of $L^{\min}$ in Proposition 6.7 represents the best-case loss in a perfectly separable setting when we assign each positive example the largest possible score $s_i^{\max}$, and each negative example the smallest possible score $s_i^{\min}$. Conversely, $L^{\max}$ represents the worst-case loss when we assign each positive example the smallest score $s_i^{\min}$, and each negative example the largest score $s_i^{\max}$. Both $L^{\min}$ and $L^{\max}$

can be computed in $O(n)$ flops using only the training data and the coefficient set by evaluating $s_i^{\min}$ and $s_i^{\max}$ as follows:

$$s_i^{\min} = \min_{\boldsymbol{\lambda}\in\mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle = \sum_{j=0}^{d} \mathbb{1}\left[x_{ij} > 0\right] x_{ij}\Lambda_j^{\min} + \mathbb{1}\left[x_{ij} < 0\right] x_{ij}\Lambda_j^{\max}, \tag{6.8}$$

$$s_i^{\max} = \max_{\boldsymbol{\lambda}\in\mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle = \sum_{j=0}^{d} \mathbb{1}\left[x_{ij} > 0\right] x_{ij}\Lambda_j^{\max} + \mathbb{1}\left[x_{ij} < 0\right] x_{ij}\Lambda_j^{\min}. \tag{6.9}$$

The values of $L^{\min}$ and $L^{\max}$ may be strengthened when we have a non-trivial limit on the number of features (i.e., $R^{\max} < d$).

We set the initial bounds for the number of non-zero coefficients R to $[0, d]$, trivially. In some cases, however, these bounds are stronger since users limit the number of non-zero coefficients (e.g., if we wish to fit models with at most 5 features, then $R \in [0, 5]$). In this case, the values of $L^{\min}$ and $L^{\max}$ are further restricted since $s_i^{\min}$ and $s_i^{\max}$ are also bounded by the number of non-zero coefficients. Here, $s_i^{\min}$ or $s_i^{\max}$ can still be computed efficiently in $O(n)$ flops by choosing the $R^{\max}$ smallest or largest terms in the left-hand side of equation (6.8) or (6.9). Having initialized $L^{\min}$, $L^{\max}$, $R^{\min}$ and $R^{\max}$, we can set the bounds on the optimal objective value as $V^{\min} = L^{\min} + C_0 R^{\min}$ and $V^{\max} = L^{\max} + C_0 R^{\max}$, respectively.

Dynamic Bounds on the Objective, Loss and Model Size

In Propositions 6.8–6.10, we provide additional bounds that can strengthen the initial values of $L^{\min}$, $L^{\max}$, $R^{\max}$, $V^{\min}$ and $V^{\max}$ using information provided by the MIP solver in LCPA.

Proposition 6.8 (Upper Bound on Optimal Number of Non-Zero Coefficients)
Given an upper bound on the optimal objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the optimal loss $L^{\min} \leq l(\boldsymbol{\lambda}^*)$, we can derive an upper bound on the optimal number of non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$ as

$$R^{\max} = \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor.$$

Proposition 6.9 (Upper Bound on Optimal Loss)
Given an upper bound on the optimal objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the optimal number of non-zero coefficients $R^{\min} \leq \|\boldsymbol{\lambda}^*\|_0$, we can derive an upper bound on the optimal loss $L^{\max} \geq l(\boldsymbol{\lambda}^*)$ as

$$L^{\max} = V^{\max} - C_0 R^{\min}.$$

Proposition 6.10 (Lower Bound on Optimal Loss)
Given a lower bound on the optimal objective value $V^{\min} \leq V(\boldsymbol{\lambda}^*)$, and an upper bound on the optimal number of non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$, we can derive a lower bound on value of the loss function $L^{\min} \leq l(\boldsymbol{\lambda}^*)$ as

$$L^{\min} = V^{\min} - C_0 R^{\max}.$$

Chained Updates Procedure

In Algorithm 8, we present a simple procedure that uses Propositions 6.8–6.10 to strengthen the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$ in RiskSlimLP.

Algorithm 8 ChainedUpdates

```
Input
    $C_0$                                          $\ell_0$ penalty parameter for RiskSlimMINLP
    $V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}$      initial bounds on $V(\boldsymbol{\lambda}^*)$, $l(\boldsymbol{\lambda}^*)$ and $\|\boldsymbol{\lambda}^*\|_0$

1: repeat
2:     $V^{\min} \leftarrow \max\left(V^{\min},\ L^{\min} + C_0 R^{\min}\right)$      ▷update lower bound on $V(\boldsymbol{\lambda}^*)$
3:     $V^{\max} \leftarrow \min\left(V^{\max},\ L^{\max} + C_0 R^{\max}\right)$      ▷update upper bound on $V(\boldsymbol{\lambda}^*)$
4:     $L^{\min} \leftarrow \max\left(L^{\min},\ V^{\min} - C_0 R^{\max}\right)$      ▷update lower bound on $l(\boldsymbol{\lambda}^*)$
5:     $L^{\max} \leftarrow \min\left(L^{\max},\ V^{\max} - C_0 R^{\min}\right)$      ▷update upper bound on $l(\boldsymbol{\lambda}^*)$
6:     $R^{\max} \leftarrow \min\left(R^{\max},\ \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor\right)$      ▷update upper bound on $\|\boldsymbol{\lambda}^*\|_0$
7: until there are no more bound updates due to Steps 2 to 6.
Output: $V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}$
```

Propositions 6.8–6.10 impose dependencies between the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, $R^{\min}$ and $R^{\max}$ that may lead to a complex "chain" of updates. As shown in Figure 6.9, it may be possible to update more than one value, and update some values multiple times. Consider a case where we call the procedure once our MIP solver improves the lower bound on the objective value $V^{\min}$. If it updates $L^{\min}$ on Step 4, but does not update $R^{\max}$ in Step 6, then it will not update $V^{\max}$, $L^{\min}$, $L^{\max}$, $V^{\min}$ at the next iteration. However, if $R^{\max}$ is updated after rounding, then $V^{\max}$, $L^{\min}$, $L^{\max}$, $V^{\min}$ will be updated.

In light of these dependencies, Algorithm 8 cycles through Propositions 6.8–6.10 until it cannot update any of the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$. This ensures that Algorithm 8 returns the strongest possible bounds, regardless of the term that was first updated. In addition, it allows us to call Algorithm 8 in other settings, such as the initialization procedure in Section 6.3.2.

In Figure 6.10, we show how the chained updates procedure improves the lower bound and optimality gap produced over the course of LCPA. Here, we call ChainedUpdates whenever LCPA updates $V^{\max}$ in Step 11, or $V^{\min}$ in Step 20. If the bounds

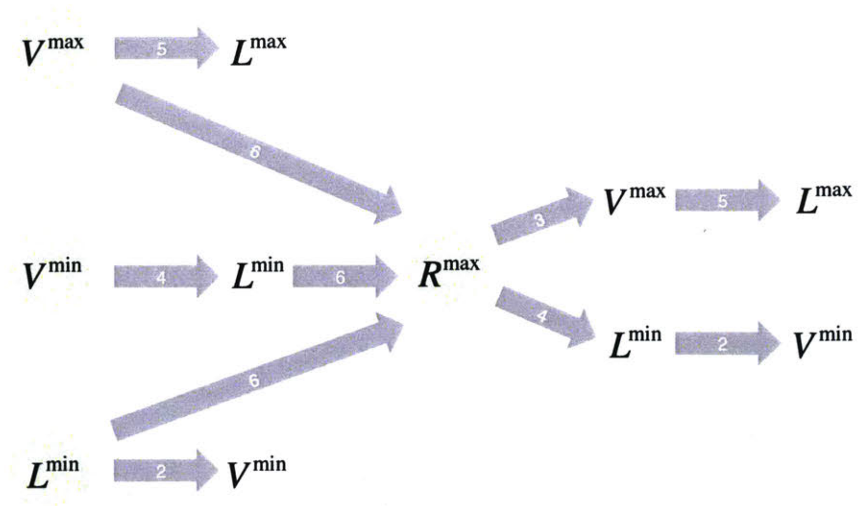


Figure 6.9: All possible update chains in Algorithm 8. We circle terms that can be updated externally by the MIP solver in LCPA. When any of these terms is updated, we run Algorithm 8 and potentially strengthen all terms in the outward path marked by the arrows. The numbers inside each arrow refer to the relevant step in Algorithm 8.

improve as a result of this call, we pass this information to the MIP solver by updating the bounds in the LP formulation in Definition 6.4.

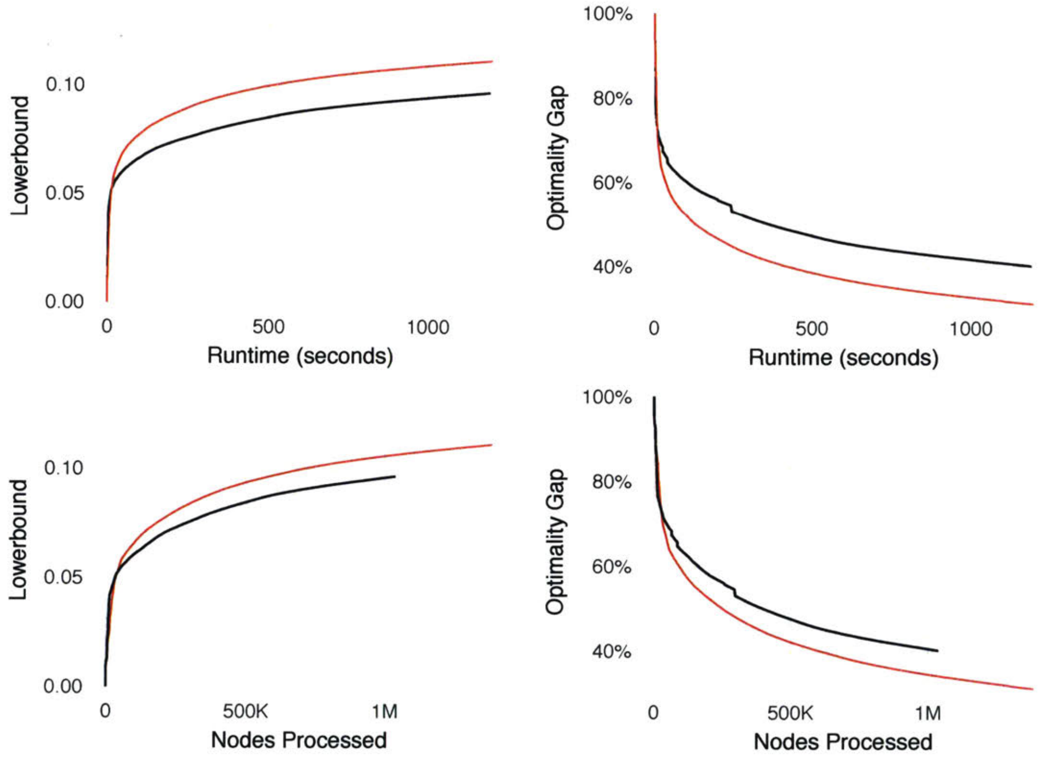


Figure 6.10: Performance of LCPA in a basic implementation (black) and with ChainedUpdates (red). Results reflect performance on an RiskSlimMINLP instance for a simulated dataset with $d = 30$ and $n = 50{,}000$ (see Appendix B.3 for details).

Initialization Procedure

In Algorithm 9, we present an initialization procedure to kick-start LCPA with a high quality feasible solution, a set of initial cutting planes, and strong bounds on the values of the objective, loss, and number of non-zero coefficients. The procedure uses all techniques presented so far, as follows:

1. *Run CPA on* RISKSLIMLP: We apply CPA to solve the surrogate LP, RISKSLIMLP until a time limit is reached (or any other user-specified stopping condition). We store the cuts from RISKSLIMLP to initialize LCPA, and record the lower bound from CPA on the objective value of RISKSLIMLP as it is also a lower bound on the optimal value of RISKSLIMMINLP.

2. *Sequential Rounding and Polishing*: We collect the solutions produced at each iteration of CPA. For each solution, we run SequentialRounding (Algorithm 7) to obtain an integer feasible solution for RISKSLIMMINLP. We then polish this solution using DCD (Algorithm 6). We use the best solution to update the upper bound on the optimal value to RISKSLIMMINLP.

3. *Chained Updates*: Having obtained strong bounds on $V^{\min}$ and $V^{\max}$, we update all bounds using ChainedUpdates (Algorithm 8).

In Figure 6.11, we show how the initialization procedure in Algorithm 9 improves the lower bound and the optimality gap over the course of LCPA.

Algorithm 9 Initialization Procedure for LCPA

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^n$                                   training data
    $\mathcal{L}$                                                  constraint set for RiskSlimMINLP
    $C_0$                                                  $\ell_0$ penalty parameter for RiskSlimMINLP
    $V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}$     initial bounds on $V(\boldsymbol{\lambda}^*)$, $l(\boldsymbol{\lambda}^*)$ and $\|\boldsymbol{\lambda}^*\|_0$
    $T^{\max}$                                                 time limit for CPA on RiskSlimLP

Initialize
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$                     initial approximation of loss function

Step I: Solve RiskSlimLP with CPA
 1: Solve RiskSlimLP$(\hat{l}^0(\boldsymbol{\lambda}), \mathrm{conv}(\mathcal{L}))$ using CPA        ▷Algorithm 4
 2: $k \leftarrow$ number of CPA iterations completed in $T^{\max}$
 3: $\hat{l}^{\text{initial}}(\boldsymbol{\lambda}) \leftarrow \hat{l}^k(\boldsymbol{\lambda})$            ▷store cuts
 4: $\mathcal{Q}^{\text{cts}} \leftarrow \{\boldsymbol{\lambda}^t\}_{t=1}^k$                  ▷store solutions
 5: $V^{\min} \leftarrow$ lower bound from CPA        ▷CPA LB for RiskSlimLP is LB for $V(\boldsymbol{\lambda}^*)$

Step II: Round and Polish Continuous Solutions from CPA
 6: for each $\boldsymbol{\lambda}^{\text{cts}} \in \mathcal{Q}^{\text{cts}}$ do
 7:     $\boldsymbol{\lambda}^{\text{sr}} \leftarrow \mathsf{SequentialRounding}(\boldsymbol{\lambda}^{\text{cts}}, \mathcal{L}, C_0)$
 8:     $\boldsymbol{\lambda}^{\text{dcd}} \leftarrow \mathsf{DCD}(\boldsymbol{\lambda}^{\text{sr}}, \mathcal{L}, C_0)$
 9:     $\mathcal{Q}^{\text{int}} \leftarrow \mathcal{Q}^{\text{int}} \cup \{\boldsymbol{\lambda}^{\text{dcd}}\}$     ▷store polished integer feasible solutions
10: end for
11: $\boldsymbol{\lambda}^{\text{best}} \leftarrow \mathrm{argmin}_{\boldsymbol{\lambda} \in \mathcal{Q}^{\text{int}}} V(\boldsymbol{\lambda})$
12: $V^{\max} \leftarrow V(\boldsymbol{\lambda}^{\text{best}})$      ▷best integer feasible solution is upper bound for $V(\boldsymbol{\lambda}^*)$

Step III: Update Bounds on Objective Terms
13: $(V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}) \leftarrow \mathsf{ChainedUpdates}(V^{\min}, \ldots, R^{\max}, C_0)$

Output: $\boldsymbol{\lambda}^{\text{best}}$, $\hat{l}^{\text{initial}}(\boldsymbol{\lambda})$, $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, $R^{\min}$, $R^{\max}$
```

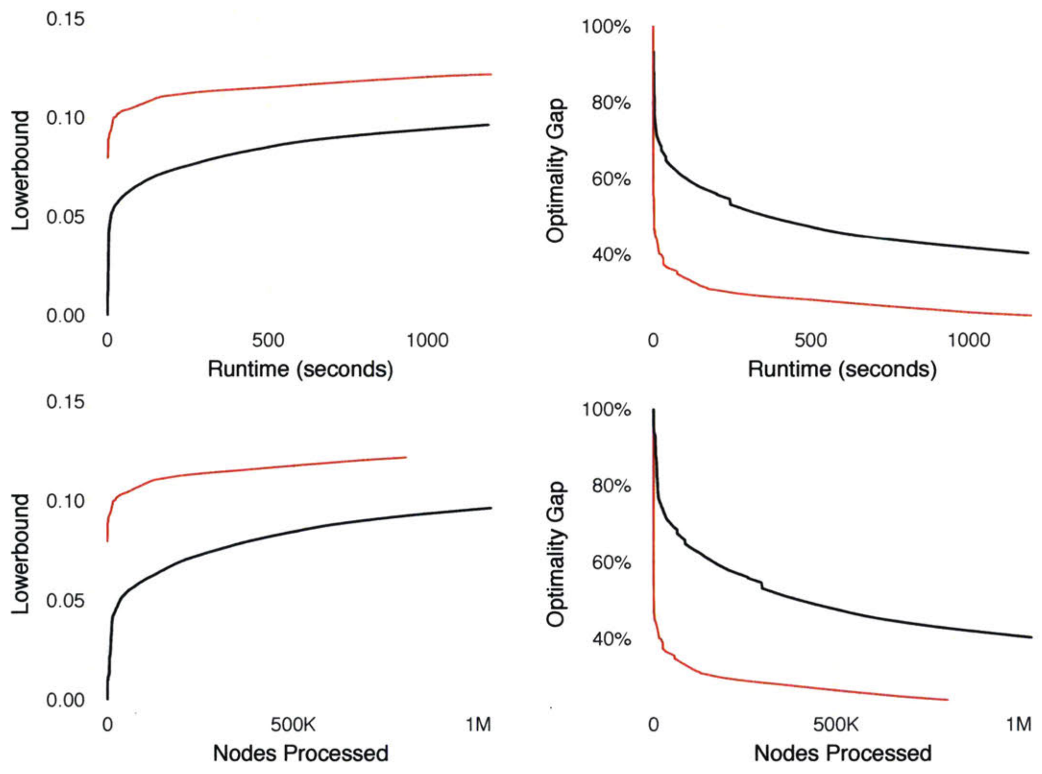


Figure 6.11: Performance profile of LCPA in a basic implementation (black) and with the initialization procedure in Algorithm 9 (red). Results reflect performance on an RISKSLIM-MINLP instance for a simulated dataset with $d = 30$ and $n = 50{,}000$ (see Appendix B.3 for details).

6.3.3 Reducing Data-Related Computation

In this section, we present techniques to reduce data-related computation for the risk score problem. The reduction is achieved by exploiting the fact that coefficients belong to a bounded discrete set $\mathcal{L}$.

Fast Loss Evaluation via a Lookup Table

The first technique aims to reduce computation when evaluating the loss function and its gradient, which affects runtime when we compute cut parameters (6.3) and run the rounding and polishing procedures in Section 6.3.1. The technique requires that the features $\boldsymbol{x}_i$ and coefficients $\boldsymbol{\lambda}$ belong to sets that are bounded, discrete, and regularly spaced, such as $\boldsymbol{x}_i \in \mathcal{X} \subseteq \{0,1\}^d$ and $\boldsymbol{\lambda} \in \mathcal{L} \subseteq \{-10,\dots,10\}^{d+1}$.

Evaluating the logistic loss, $\log(1+\exp(-\langle\boldsymbol{\lambda}, y_i\boldsymbol{x}_i\rangle))$, is a relatively expensive computation because it involves exponentiation and must be carried out in multiple steps to avoid numerical overflow/underflow when the *scores* $s_i = \langle\boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle$ are too small or large[4]. When the training data and coefficients belong to discrete bounded sets, the scores $s_i = \langle\boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle$ belong to a discrete and bounded set

$$\mathcal{S} = \left\{\langle\boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \;\middle|\; i = 1,\dots,n \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\right\}.$$

If the elements of the feature set $\mathcal{X}$ and the coefficient set $\mathcal{L}$ are regularly spaced, then the scores belong to the set of integers $\mathcal{S} \subseteq \mathbb{Z} \cap [s^{\min}, s^{\max}]$ where:

$$\begin{aligned} s^{\min} &= \min_{i,\boldsymbol{\lambda}} \{\langle\boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \text{ for all } (\boldsymbol{x}_i, y_i) \in \mathcal{D} \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\}, \\ s^{\max} &= \max_{i,\boldsymbol{\lambda}} \{\langle\boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \text{ for all } (\boldsymbol{x}_i, y_i) \in \mathcal{D} \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\}. \end{aligned}$$

Thus, we can precompute and store all possible values of the loss function in a lookup table with $s^{\max} - s^{\min} + 1$ rows, where row m contains the value of $[\log(1+\exp(-(m+s^{\min}-1)))]$.

This strategy can reduce the time to evaluate the loss as we replace a computationally expensive operation with a simple lookup operation. In practice, the lookup table is usually small enough to be cached in memory, which yields a substantial runtime speedup. Further, since the values of $s^{\min}$ and $s^{\max}$ can be computed exactly in $O(n)$ time, the lookup table can be narrowed down as $R^{\max}$ is updated over the course of LCPA.

As shown in Figure 6.12, using a lookup table reduces the total amount of data-related computation compared to a standard high performance numerical computation library. The reduction in data-related computation may translate into a significant difference in the ability of the algorithm to return a high-quality risk score, with a stronger proof of optimality under a time constraint.

[4]The value of $\exp(s)$ can be computed reliably using IEEE 754 double precision floating point numbers for $s \in [-700, 700]$. The term will overflow to ∞ when $s < -700$, and underflow to 0 when when $s > 700$.

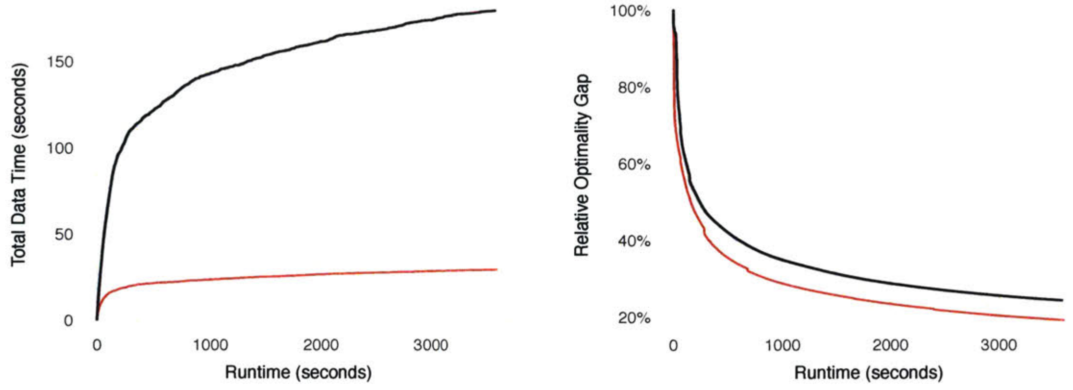


Figure 6.12: Time spent on data-related computation and optimality gap for LCPA when we evaluate the loss function using a standard high performance numerical computation library (black) and a lookup table (red). Results reflect performance on a simulated dataset with $d = 30$ and $n = 10^6$ (see Appendix B.3 for details).

Faster Heuristics via Subsampling

The next technique aims to reduce data-related computation for heuristics such as SequentialRounding by using a subsample of the full training dataset.

In theory, we wish to run heuristic procedures frequently because they can yield feasible solutions to RISKSLIMMINLP that may update the incumbent solution. In practice, however, each procedure requires multiple evaluations of the loss function, which means that runs that fail to update the incumbent solution effectively slow down the progress of LCPA. If, for example, we ran SequentialRounding each time we found a new set of continuous coefficients in LCPA (i.e, in Step 15 in Algorithm 5), then we would spend too much time rounding, without necessarily finding a better solution.

Our proposed technique works as follows. Before running LCPA, we set aside a *heuristics dataset* $\mathcal{D}_m$ by sampling m points without replacement from the full training dataset $\mathcal{D}_n$. Once we run LCPA, we then reduce data-related computation by running heuristics with $\mathcal{D}_m$. To clarify when the loss and objective are computed using $\mathcal{D}_m$ or $\mathcal{D}_n$, we let $l_i(\boldsymbol{\lambda}) = \log(1 + \exp(\langle \boldsymbol{\lambda}, y_i\boldsymbol{x}_i, \rangle))$ and define:

$$l_m(\boldsymbol{\lambda}) = \frac{1}{m}\sum_{i=1}^{m} l_i(\boldsymbol{\lambda}), \qquad V_m(\boldsymbol{\lambda}) = l_m(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0,$$

$$l_n(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} l_i(\boldsymbol{\lambda}), \qquad V_n(\boldsymbol{\lambda}) = l_n(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0.$$

Consider a case where a heuristic procedure returns a promising solution $\boldsymbol{\lambda}^{\text{hr}}$ such that:

$$V_m(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}. \tag{6.10}$$

In this case, we compute the objective value on the full training dataset $\mathcal{D}_n$ by evaluating the loss at each of the $n - m$ points that were not included in $\mathcal{D}_m$. As usual, we then update the incumbent solution if $\boldsymbol{\lambda}^{\text{hr}}$ attains an objective value that is less than the current upper bound on RiskSlimMINLP:

$$V_n(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}. \tag{6.11}$$

Note that, although we need to evaluate the loss for the full training dataset $\mathcal{D}_n$ to confirm an incumbent update, this strategy still reduces data-related computation because heuristic procedures require multiple evaluations of the loss functions (e.g., sequential rounding requires $\frac{1}{3}d(d^2 - 1)$ evaluations of the loss). In the interest of reducing computation, this technique also ignores "false negative" solutions $\boldsymbol{\lambda}^{\text{hr}}$ that do poorly on the heuristics dataset $V_m(\boldsymbol{\lambda}^{\text{hr}}) \geq V^{\max}$ but would update the incumbent on the true dataset $V_n(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}$.

The main draw of using the subsampling technique is the following generalization bound that guarantees that any solution that updates the incumbent when the objective is evaluated with $\mathcal{D}_m$ will also update incumbent when the objective is evaluated with $\mathcal{D}_n$ (i.e., that any solution that satisfies (6.10) will also satisfy (6.11)).

Theorem 6.11 (Generalization of Sampled Loss on Finite Coefficient Set) *Let $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$ denote a training dataset with $n > 1$ points, and let $\mathcal{D}_m = (\boldsymbol{x}_i, y_i)_{i=1}^m$ denote a sample of m points drawn without replacement from $\mathcal{D}_n$. Let $\boldsymbol{\lambda}$ denote the coefficients of a linear classifier from a finite set $\mathcal{L}$. For all $\varepsilon > 0$, it holds that*

$$\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda})\right) \geq \varepsilon\right) \leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta^{\max}(\mathcal{L},\mathcal{D}_n)^2}\right),$$

where:

$$\Delta^{\max}(\mathcal{L},\mathcal{D}_n) = \max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(\max_{i=1,\ldots,n} l_i(\boldsymbol{\lambda}) - \min_{i=1,\ldots,n} l_i(\boldsymbol{\lambda})\right).$$

Theorem 6.11 is a generalization bound that is derived from a concentration inequality for problems where we are sampling without replacement, known as the Hoeffding-Serfling inequality (see Bardenet et al., 2015). The Hoeffding-Serfling inequality can be significantly tighter than the classical Hoeffding inequality as it ensures that $\Pr(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda}) \geq \epsilon) \to 0$ as $m \to n$ for all $\epsilon > 0$. Here, $\Delta^{\max}(\mathcal{L},\mathcal{D}_n)$ is a normalization term that represents the maximum range of loss values on the full training dataset $\mathcal{D}_n$ for the coefficient set $\mathcal{L}$. This term can be computed cheaply using the smallest and largest values of the coefficients and features as shown in Proposition 6.7 from Section 6.3.2.

Most machine learning settings are unlike the one we consider here. In such settings, the $|\mathcal{L}|$ term in Theorem 6.11 produces a bound that is infinite, and thus vacuous. In this case, however, rounding ensures that the $|\mathcal{L}|$ term has at most 2^d elements, which effectively constrains the difference between $l_n(\boldsymbol{\lambda})$ and $l_m(\boldsymbol{\lambda})$. Thus, Theorem 6.11 can be used to assess the probability that a proposed incumbent update will lead to an actual incumbent update, as shown in Corollary 6.12. Alternatively, it can be used to choose the size of the subsampled dataset m so that an incumbent update on $\mathcal{D}_m$ is yield an incumbent update on $\mathcal{D}_n$. In either case, the bound can be strengthened by recomputing the normalization term $\Delta^{\max}(\mathcal{L}(\boldsymbol{\rho}), \mathcal{D}_n)$ separately for each continuous solution ρ, or periodically over the course of LCPA (as the MIP solver reduces the set of feasible coefficients via B&B).

Corollary 6.12 (Update Probabilities of Rounding Heuristics on Sampled Data) *Consider a rounding heuristic that takes as input a vector of continuous coefficients* $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d) \in \operatorname{conv}(\mathcal{L})$ *and produces as output a vector of integer coefficients* $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$ *where*

$$\mathcal{L}(\boldsymbol{\rho}) = \left(\boldsymbol{\lambda} \in \mathcal{L} \mid \lambda_j \in \{\lceil \rho_j \rceil, \lfloor \rho_j \rfloor\} \text{ for } j = 1, \ldots, d\right).$$

Consider evaluating the rounding heuristic using a sample of m points $\mathcal{D}_m = (\boldsymbol{x}_i, y_i)_{i=1}^m$ drawn without replacement from the full training dataset $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$. Pick a tolerance $\delta > 0$. Given rounded coefficients $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$, compute $V_m(\boldsymbol{\lambda})$. If

$$V_m(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta,$$

then w.p. at least $1 - \delta$, we have

$$V_n(\boldsymbol{\lambda}) \leq V^{\max},$$

where

$$\varepsilon_\delta = \Delta^{\max}(\mathcal{L}(\boldsymbol{\rho}), \mathcal{D}_n)\sqrt{\frac{\log(1/\delta) + d\log(2)}{2}\left(1 - \frac{m}{n}\right)\left(1 + \frac{m}{n}\right)}.$$

6.4 Performance Benchmarks

In this section, we benchmark several methods to learn risk scores with few terms and small integer coefficients. In addition to RiskSLIM, we consider advanced heuristics that process the coefficients of penalized logistic regression using the rounding and polishing techniques from Section 6.3.1.

6.4.1 Setup

Datasets

We ran experiments on 6 publicly available datasets shown in Table 6.2. We picked these datasets to explore the performance of each method as we varied the size and nature of the training data. Other than `arrest`, all datasets can be found at the UCI ML repository (Lichman, 2013). The `arrest` dataset must be requested from ICPSR as described in Zeng et al. (2016).

Dataset	**Reference**	n	d	**Classification Task**
`adult`	Kohavi (1996)	32561	36	predict if a U.S. resident earns over $50K
`arrest`	Zeng et al. (2016)	22530	48	predict if a prisoner is arrested after release
`bank`	Moro et al. (2014)	41188	57	predict if a firm will go bankrupt
`mammo`	Elter et al. (2007)	961	14	detect breast cancer using a mammogram
`mushroom`	Schlimmer (1987)	8124	113	predict if a mushroom is poisonous
`spambase`	Cranor and LaMacchia (1998)	4601	57	predict if an e-mail is spam

Table 6.2: Datasets used for benchmarking RiskSLIM.

Methods

For each dataset, we fit a risk score with small integer coefficients $\lambda_j \in \{-5, 5\}$ and limited model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ to match models used in practice (e.g., Gage et al., 2001). We used a total of 8 methods, described below.

- RiskSLIM (Optimized Risk Score): We formulate an instance of RiskSlimMINLP with the following constraints: $\lambda_0 \in \{-100, \ldots, 100\}$, $\lambda_j \in \{-5, \ldots, 5\}$, and $\|\boldsymbol{\lambda}\|_0 \leq 5$. We set C_0 to a small value (10^{-8}) to recover the best model under these constraints (see Appendix B.1). We solve each instance using LCPA along with the improvements in Sections 6.3 and 6.3.3. We cap runtime to 20 minutes, and use the CPLEX 12.6.3 Python API on a 3.33GHz CPU with 16GB RAM.
- PLR (Penalized Logistic Regression): We use the glmnet package (Friedman et al., 2010) to fit logistic regression models with a combined $\ell_1 + \ell_2$ penalty. We add constraints to bound $\lambda_j \in [-5, 5]$, and consider models for 1,100 distinct combinations of free parameters: 11 values of the mixing parameter $\{0.0, 0.1, \ldots, 1.0\} \times 100$ values of the regularization penalty (chosen by glmnet). These free parameters mean

that PLR also covers the following variants of logistic regression as special cases: standard logistic regression (no penalty); Lasso (pure ℓ_1-penalty); and Ridge (pure ℓ_2-penalty).

- RD (PLR + Naïve Rounding): We fit a pool of models with PLR. For each model in the pool, we round each coefficient to the nearest integer in $\{-5, \ldots, 5\}$ by setting $\lambda_j \leftarrow \lceil \min(\max(\lambda_j, -5), 5) \rfloor$. We round the intercept to the nearest integer by setting $\lambda_0 \leftarrow \lceil \lambda_0 \rfloor$.
- RSRD (PLR + Rescaled Rounding): We fit a pool of models with PLR. For each model in the pool, we rescale coefficients so that the largest coefficient is ± 5, then round to the nearest integer (i.e. $\lambda_j \rightarrow \lceil \gamma \lambda_j \rfloor$ where $\gamma = 5/\max_j |\lambda_j|$). Rescaling aims to prevent rounding coefficients to zero when $|\lambda_j| < 0.5$ for many j.
- SEQRD (PLR + Sequential Rounding): We fit a pool of models using PLR. For each model in the pool, we round the coefficients using SequentialRounding (Algorithm 7).
- RD*/RSRD*/SEQRD* (Polished Versions of RD/RSRD/SEQRD): We fit a pool of models using RD/RSRD/SEQRD and polish the coefficients using DCD (Algorithm 6). To ensure that the number of non-zero coefficients does not increase (which would violate the model size constraint), we only run DCD on the set of non-zero coefficients $\{j \mid \lambda_j \neq 0\}$.

Performance Metrics

We evaluate all models in terms of *risk-calibration* (measured by CAL) and *rank accuracy* (measured by AUC). We use *reliability diagrams* to show how the predicted risk (x-axis) matches the observed risk (y-axis) for each distinct score (see DeGroot and Fienberg, 1983). We estimate the observed risk at each score s as

$$\bar{p}_s = \frac{1}{|\{i : s_i = s\}|} \sum_{i:s_i=s} \mathbb{1}\left[y_i = +1\right].$$

We summarize calibration over the full reliability diagram using the *calibration error* (see Caruana and Niculescu-Mizil, 2004, 2006).

$$\text{CAL} = \frac{1}{n}\sqrt{\sum_{s}\sum_{i:s_i=s}(p_i - \bar{p}_s)^2}.$$

A model with perfect calibration yields predicted risk estimates that are perfectly aligned with observed values, meaning that the points on the reliability diagram should fall on the $x = y$ line, and CAL should equal 0.0%.

Model Selection

We used a standard nested 5-fold cross-validation (5-CV) to select a final model and assess its predictive accuracy. We fit a final model using all of training data for the instance of the free parameters that: (i) satisfied the model size constraint; and (ii) maximized the 5-CV mean test AUC. Since 5-CV statistics are used to choose free parameters for the final model, they provide an optimistic measure of expected performance for this model. To avoid this bias, we used a nested CV setup where we ran the previous model selection procedure for each fold. Explicitly, for each of the 5 "outer" folds, we ran an "inner" 5-CV and fit a final model using all the data for that fold. We picked a final model for each outer fold for the free parameter instance that: (i) satisfied the model size constraint; and (ii) maximized the inner 5-CV mean test AUC. We do not tune parameters for RISKSLIM since we include the model size constraint in the coefficient set.

6.4.2 Results

In Table 6.3, we summarize the performance of risk scores from all methods on all datasets, and provide reliability diagrams to show calibration performance of these models in greater detail in Figure 6.13. In Figures 6.14–6.15, we plot ℓ_0-regularization paths to show how the CAL and AUC of risk scores change with the model size constraint. In Figures 6.16–6.18, we show RISKSLIM models for `arrest`, `adult` and `bank`. In what follows, we discuss these results.

On Performance

As shown in Table 6.2, RISKSLIM models have superior risk calibration and rank accuracy compared to models built with other methods. Specifically, RISKSLIM models have the best 5-CV mean test CAL (i.e.,*test CAL*) on 6/6 datasets, and the best 5-CV mean test AUC (i.e., *test AUC*) on 5/6 datasets. In comparison, models built using baseline methods (i.e., all methods other than PLR) perform slightly worse in terms of test AUC and significantly worse in terms of test CAL. As shown in Figures 6.14 and 6.15, the relative performance advantages of RISKSLIM models are more notable for smaller model sizes – which is beneficial since risk scores typically need to have few terms.

Most of these results can be explained by noting that: (i) models that attain low values of the logistic loss have good risk calibration (as shown in Table 6.3, and observed by Caruana and Niculescu-Mizil, 2004); and, (ii) since we are fitting from a simple class of models, almost all risk scores in Table 6.2 generalize well (i.e., the test CAL/AUC is very close to their training CAL/AUC). RISKSLIM models attain the smallest value of the logistic loss as they minimize the loss over exact constraints on model form. As such, they perform well in terms of training CAL as per (i), and subsequently in terms of test CAL as per (ii). Likewise, baseline methods that use loss minimizing heuristics such as DCD and SequentialRounding (i.e., RD*, RSRD* and SEQRD*) produce models with lower loss relative to baseline methods that do not use such techniques (i.e., RD, RSRD, and SEQRD). As a result, models built using RD*, RSRD* and SEQRD* have improved risk calibration.

The results in Table 6.2 suggest a similar relationship between the logistic loss and rank accuracy, but this has some caveats. In particular, a method such as RSRD, can provably improve AUC by rescaling coefficients before rounding. However, given that the logistic loss is not scale invariant, the rescaled models have high loss and poor risk calibration (see e.g., the reliability diagrams for RSRD in Figure 6.13). In addition, it is possible for a model that minimizes the logistic loss to have the best training AUC (see e.g., `mammo` where the RISKSLIM model has an optimality gap of 0.0%).

On Computation

RISKSLIM is the only method to pair models with a measure of optimality. Although the risk score problem is NP-hard, we fit models with small optimality gaps in ≤ 20 minutes by pairing LCPA with the techniques in Sections 6.3 and 6.3.3.

There are also some practical benefits that are difficult to measure. In particular, RISKSLIM can build and assess the predictive risk scores without the need for parameter tuning and nested CV, meaning that we had to train only 6 models. In comparison, baseline methods do require parameter tuning and nested CV, meaning that we had to run rounding and polishing techniques and compute AUC for over 33,000 models. Thus, even though the baseline methods are sometimes much faster to run for a single choice of parameters, when we consider the computation time needed for parameter tuning, the entire training process could take far longer than the time used to fit RISKSLIM, especially on large datasets.

On Pitfalls and Best-Practices for Heuristic Methods

Our results show that the performance of risk scores built with heuristics methods depends on a range of factors, including: the rounding technique; the constraints on model form; and the range of feature values. In some cases, risk scores built by simply rounding coefficients to the nearest integer perform well (see e.g., RD on `bank`). In others, however, performance can falter (see e.g., RD for `spambase`).

In practice, performance issues are often overlooked as common heuristics result in good AUC but poor CAL (e.g. the rescaling in RSRD, used by U.S. Department of Justice, 2005; Pennsylvania Commission on Sentencing, 2012, and many others). In addition, summary statistics such as AUC and CAL may conceal performance issues over the full reliability diagram and ROC curve. The fact that that risk scores built with heuristic methods may have inconsistent performance highlights the importance of an optimality gap, which can help determine if performance issues are due to overly restrictive constraints on the model class. The optimality gap is especially valuable in this setting because PLR models, which would provide a natural performance baseline for risk scores as they do not obey the integrality constraints, may perform poorly due to suboptimal feature selection and overfitting (see e.g., `adult`, `mushroom` and `spambase` in Table 6.3).

To mitigate these issues, we recommend the following practices when using or designing heuristics:

1. *Select models after rounding.* If we selected a final model from the pool of PLR models before rounding the coefficients, we could greatly alter the loss and thus reduce performance. To mitigate this risk, we first round the coefficients of all models in the pool, and then select among the rounded models.
2. *Avoid scaling.* Rescaling coefficients may improve AUC but drastically reduces CAL. This is because the logistic loss is not scale invariant (see e.g. the reliability diagram for RSRD in Figure 6.13). The decrease in CAL due to scaling is reflected by the much higher values of the loss for RSRD in Table 6.3.
3. *Select models that optimize K-CV AUC instead of K-CV CAL.* We compared both procedures. Choosing a model that optimizes the K-CV CAL leads to models with slightly better CAL (though not as good as RISKSLIM) but far worse AUC. This is because trivial and near-trivial models have low CAL on problems with class imbalance.
4. *Binarize real-valued features.* When datasets contain real-valued features (e.g. `spambase`), PLR may assign small coefficients to features with large values. In such cases, rounding can greatly impact performance by removing features such that $|\lambda_j| < 0.5$. This issue is best addressed by binarizing: rescaling coefficients before rounding affects calibration; normalizing reduces usability as it requires users to also normalize when using the model.

These recommendations are for heuristics only. RISKSLIM does not need them.

Dataset	Metric	PLR	Rd	RsRd	SeqRd	Rd*	RsRd*	SeqRd*	RiskSLIM
	test cal	5.5%	4.3%	9.1%	4.3%	4.2%	6.3%	4.2%	2.6%
adult	**test auc**	0.817	0.830	0.830	0.830	0.832	0.832	0.832	0.854
$n = 32561$	**model size**	4	4	4	4	4	4	4	5
$d = 36$	**loss value**	0.451	0.417	0.484	0.417	0.417	0.458	0.417	0.385
	optimality gap	-	-	-	-	-	-	-	9.7%
	test cal	7.5%	5.7%	20.8%	5.7%	3.8%	15.6%	3.8%	1.7%
arrest	**test auc**	0.700	0.691	0.691	0.691	0.677	0.690	0.677	0.697
$n = 22530$	**model size**	5	5	5	5	4	5	4	5
$d = 48$	**loss value**	0.638	0.626	1.282	0.626	0.624	0.895	0.624	0.609
	optimality gap	-	-	-	-	-	-	-	4.0%
	test cal	2.2%	1.4%	9.5%	1.4%	1.3%	7.2%	1.3%	1.3%
bank	**test auc**	0.725	0.759	0.759	0.759	0.760	0.749	0.760	0.760
$n = 41188$	**model size**	2	5	5	5	5	2	5	5
$d = 57$	**loss value**	0.339	0.289	0.953	0.289	0.289	0.333	0.289	0.289
	optimality gap	-	-	-	-	-	-	-	3.5%
	test cal	7.3%	8.1%	15.3%	8.1%	7.4%	7.2%	7.4%	5.0%
mammo	**test auc**	0.845	0.845	0.845	0.845	0.845	0.836	0.845	0.843
$n = 961$	**model size**	3	3	3	3	3	3	3	5
$d = 14$	**loss value**	0.482	0.480	0.624	0.480	0.480	0.496	0.480	0.469
	optimality gap	-	-	-	-	-	-	-	0.0%
	test cal	20.9%	12.3%	6.5%	12.3%	5.4%	3.1%	5.4%	1.8%
mushroom	**test auc**	0.976	0.973	0.977	0.973	0.978	0.980	0.978	0.989
$n = 8124$	**model size**	5	5	5	5	5	5	5	5
$d = 113$	**loss value**	0.362	0.200	0.162	0.200	0.144	0.139	0.144	0.069
	optimality gap	-	-	-	-	-	-	-	0.0%
	test cal	10.5%	24.2%	23.6%	24.2%	17.9%	10.3%	17.9%	11.7%
spambase	**test auc**	0.823	0.908	0.862	0.908	0.908	0.913	0.908	0.928
$n = 4601$	**model size**	4	5	5	5	5	5	5	5
$d = 57$	**loss value**	0.553	0.472	5.670	0.472	0.402	0.381	0.402	0.349
	optimality gap	-	-	-	-	-	-	-	27.8%

Table 6.3: Performance of risk scores with model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ and integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$. Note PLR models have real-valued coefficients $\lambda_j \in [-5, 5]$. Here: *test cal* is the 5-CV mean test CAL; *test auc* is the 5-CV mean test AUC; *model size*, *loss value* and *optimality gap* pertain to a final model fit using the entire dataset.

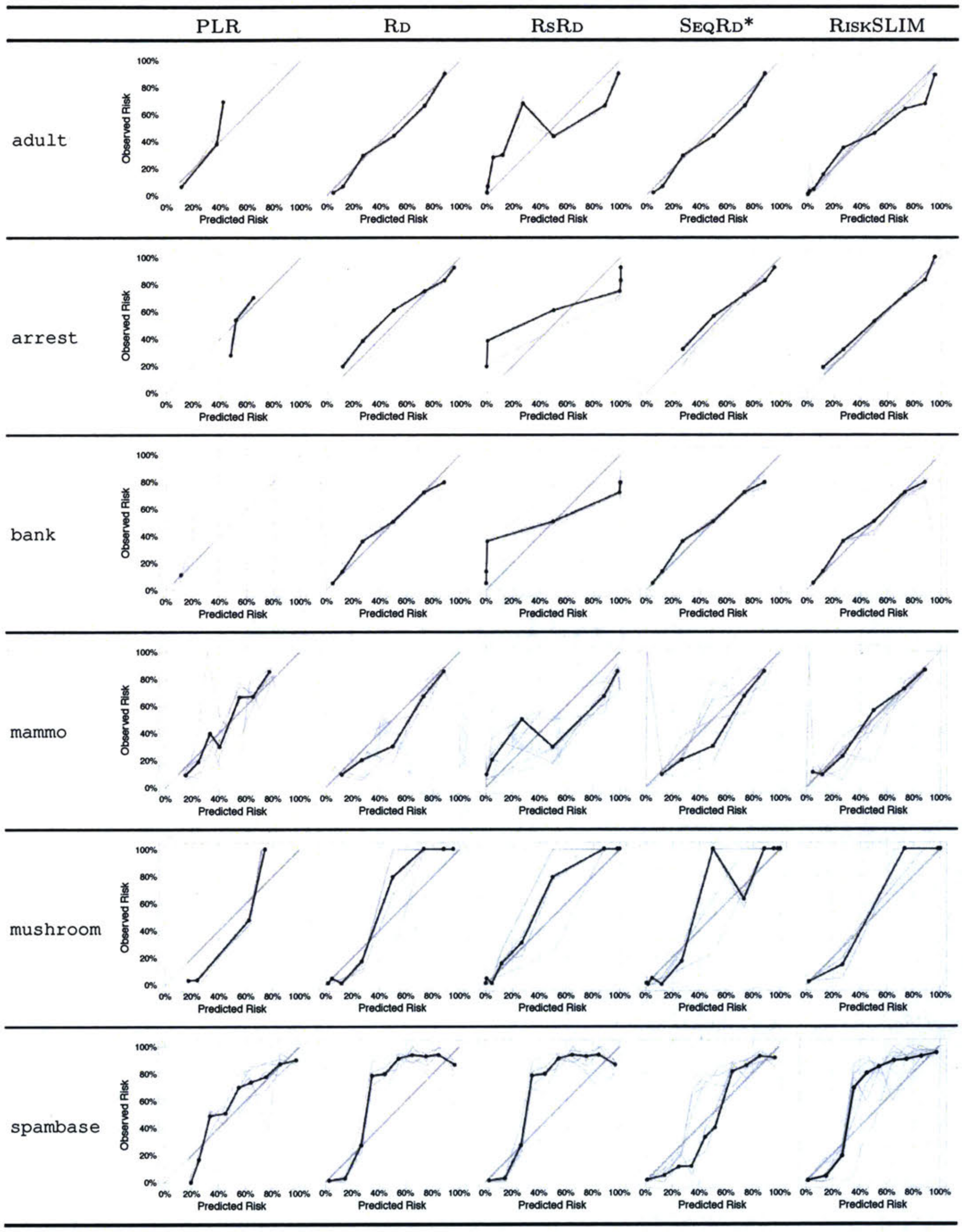


Figure 6.13: Reliability diagrams for risk scores with model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ and integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$. Note PLR models have real-valued coefficients $\lambda_j \in [-5, 5]$. We plot results for models from each fold on the test data in grey, and for the final model on training data in black.

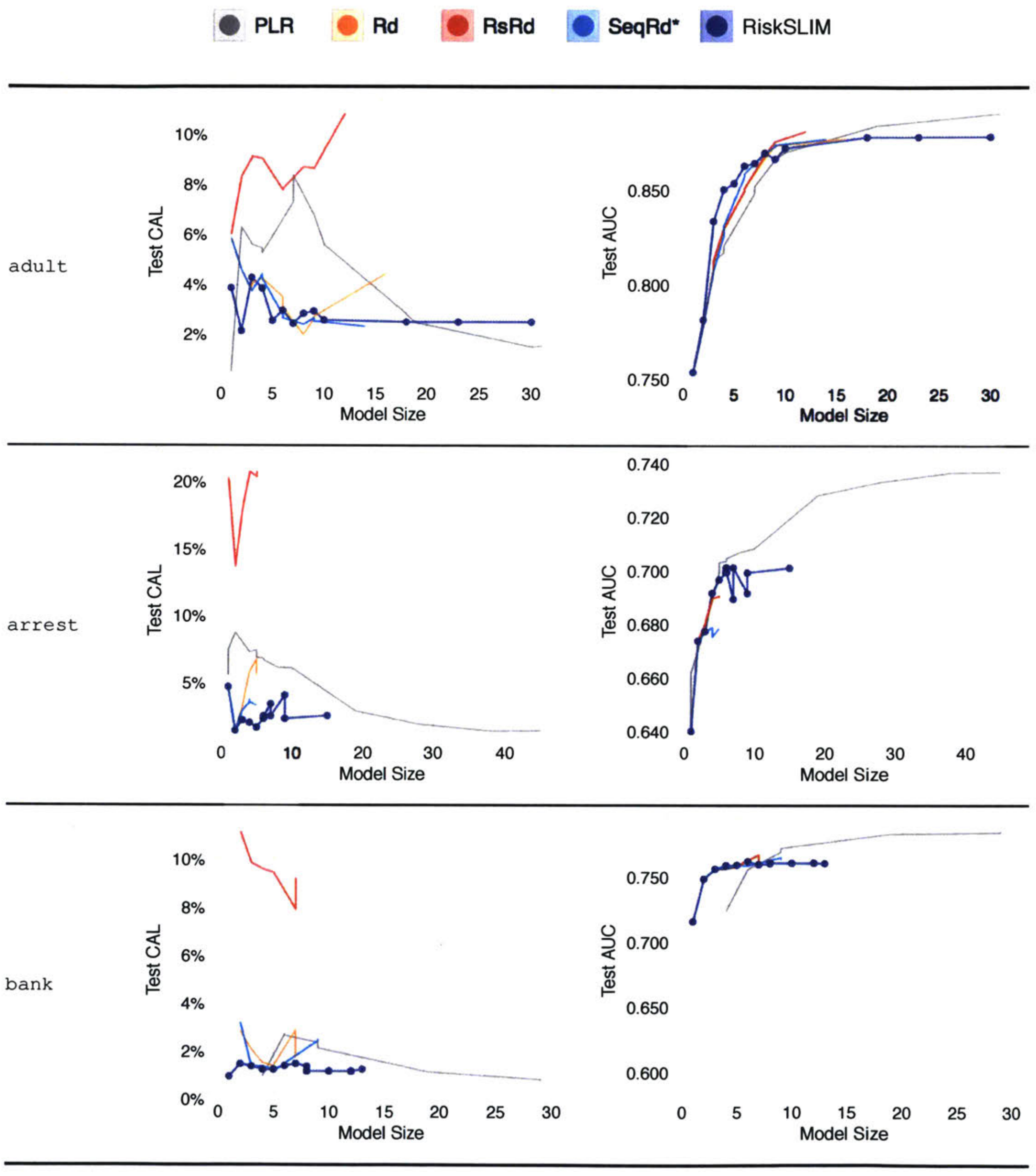


Figure 6.14: Test CAL (left) and test AUC (right) of risk scores with integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$ and model sizes $\|\boldsymbol{\lambda}\|_0 \leq R^{\max}$ for $R^{\max} \in \{1, 2, \ldots, 10, 20, 30, 40, 50, \infty\}$. Note that PLR risk scores have real-valued coefficients $\lambda_j \in [-5, 5]$.

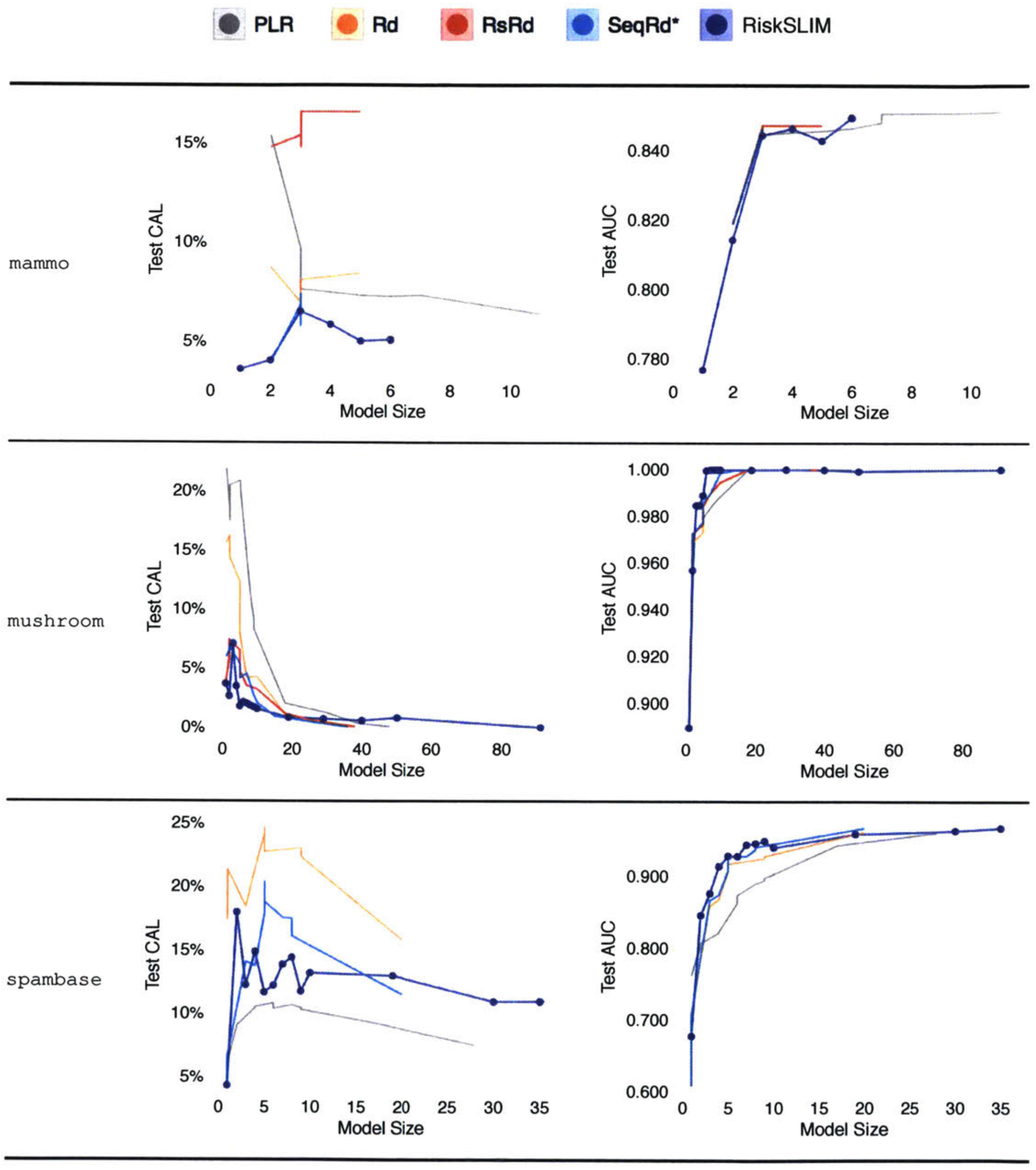


Figure 6.15: Test CAL (left) and test AUC (right) of risk scores with integer coefficients $\lambda_j \in \{-5, \dots, 5\}$ and model sizes $\|\boldsymbol{\lambda}\|_0 \leq R^{\max}$ for $R^{\max} \in \{1, 2, \dots, 10, 20, 30, 40, 50, \infty\}$. Note that PLR risk scores have real-valued coefficients $\lambda_j \in [-5, 5]$.

1.	*Prior Arrests $\geq$ 2*	1 point	$\cdots$
2.	*Prior Arrests $\geq$ 5*	1 point	$+$ $\cdots$
3.	*Prior Arrests for Local Ordinance*	1 point	$+$ $\cdots$
4.	*Age at Release between 18 to 24*	1 point	$+$ $\cdots$
5.	*Age at Release $\geq$ 40*	-1 points	$+$ $\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots$

SCORE	-1	0	1	2	3	4
RISK	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

Figure 6.16: RISKSLIM model for the arrest dataset. RISK represents the predicted probability that a prisoner is arrested for any offense within 3 years of release from prison. This model has a 5-CV mean test CAL/AUC of 1.7%/0.697 and training CAL/AUC of 2.6%/0.701.

1.	*Married*	3 points	$\cdots$
2.	*Reported Capital Gains*	2 points	$+$ $\cdots$
3.	*Age between 22 to 29*	-1 point	$+$ $\cdots$
4.	*Highest Level of Education is High School Diploma*	-2 points	$+$ $\cdots$
5.	*No High School Diploma*	-3 points	$+$ $\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots$

SCORE	≤ -1	0	1	2	3	4	5
RISK	5.0%	11.9	26.9%	50.0%	73.1%	88.1%	95.3

Figure 6.17: RISKSLIM model for the adult dataset. RISK represents the predicted probability that a US resident earns over $50 000. This model has a 5-CV mean test CAL/AUC of 2.4%/0.854 and training CAL/AUC of 4.1%/0.860.

1.	*Call between January and March*	1 point	$\cdots$
2.	*Called Previously*	1 point	$+$ $\cdots$
3.	*Previous Call was Successful*	1 point	$+$ $\cdots$
4.	*Employment Indicator < 5100*	1 point	$+$ $\cdots$
5.	*3 Month Euribor Rate $\geq$ 100*	-1 point	$+$ $\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots$

SCORE	-1	0	1	2	3	4
RISK	4.7%	11.9	26.9%	50.0%	73.1%	88.1%

Figure 6.18: RISKSLIM model for the bank dataset. RISK represents the predicted probability that a client opens a new bank account after a marketing call. This model has a 5-CV mean test CAL/AUC of 1.3%/0.760 and a training CAL/AUC of 1.1%/0.760.

6.5 Discussion

Our goal in this chapter was to develop a machine learning method to build data-driven risk scores. As in Chapter 3, we approached this problem by formulating a hard optimization problem to produce risk scores that were optimized for feature selection, small integer coefficients, and operational constraints. We then solved the risk score problem using a specialized cutting plane method that did not stall on non-convex problems, and that we paired with specialized techniques to generate feasible solutions, narrow the optimality gap, and reduce data-related computation.

An important characteristic of these is that they are more powerful when used together. For instance, using SequentialRounding and DCD to generate a feasible solution can lead to a tighter bound on model size via ChainedUpdates, which can subsequently reduce computation by reducing the size of the lookup table. As we discuss below, these techniques were also important in improving the performance of a cutting plane method within a B&B framework.

One of the key benefits of our approach is that it can learn risk scores using a MIP solver. Using a MIP solver was important in that practitioners who purchase a commercial solver (e.g. CPLEX and Gurobi) could use the same solver for SLIM and RiskSLIM. However, it was also important as MINLP solvers were not able to handle our problem off-the-shelf, and may not have performed well through a specialized approach given that their B&B implementations are typically slower and offer fewer callbacks to intervene in the search process.

In the scoring system problem in Chapter 3, we could define an exact performance measure (e.g. accuracy) and directly include it in the optimization problem (via the 0-1 loss). In this setting, however, the relevant performance measure of a model is risk-calibration, and the performance guarantee of our models depends on the assumption that a logistic risk model with small integer coefficients is a good approximation for the true risk model for a classification dataset. Our empirical results in Section 6.4 suggest that this is a reasonable assumption, as models with lower loss typically attain better risk calibration, and models with optimal loss typically attain the best possible calibration.

In Chapter 7, we see that the performance improvements shown in Section 6.4 are more significant in the presence of constraints, where even advanced heuristics that avoid pitfalls can perform poorly. In addition, we discuss several of the practical benefits of our approach, namely that: (i) it can fit models that obey complex operational constraints without parameter tuning, which greatly reduces the number of models that are required to build a risk score and evaluate its performance; and (ii) it can assess the optimality of each model, which can help users tell if poor performance is due to the model class or the fitting process.

Technical Discussion

The performance of RiskSLIM models hinges on the ability to fit models that attain optimal or near-optimal values of the logistic loss. In this chapter, we found these models using a cutting plane algorithm (LCPA) that decoupled the loss from the optimization problem, and provided scalability and control over data-related computation. LCPA can be used to learn models for other problems with non-convex regularizers and or non-convex feasible regions. In practice, however, its effectiveness depends on how cutting planes are generated and incorporated in a B&B search process. In our implementation, we were able to overcome several pitfalls of using a cutting plane algorithm within a B&B framework using the following strategies:

- *Initialization*: Since LCPA only adds cuts when it discovers an integer feasible solution, B&B search may be slow at first, as it would have to discover feasible solutions before it can construct a useful approximation of the loss function. To address this, we start B&B with an initial set of cuts (generated through traditional CPA on the convex relaxation).
- *Limited # of Cuts*: LCPA has a limited number of cuts it only adds cuts at integer feasible solutions. In theory, this could be further reduced as LCPA would converge if cuts are added at solutions that update the incumbent.
- *Lazy Cut Evaluation*: If cuts are incorporated in the LP formulation at each node, then adding a large number of cuts increases the time to process each node, and slows down the B&B search. In our implementation, which uses CPLEX (ILOG, 2017), we address this by adding cuts as *lazy constraints*. This means the cuts are not included in the LP formulation at each node, but stored in memory until the B&B finds an integer feasible solution (at which point they are checked). Lazy evaluation reduces the time to solve the LP relaxation at each node, improving the effectiveness of B&B search. Since we only evaluate the cuts when we find an integer feasible solution, this strategy also reduces the marginal computational cost for each cut. This mitigates the risk slowing down progress by adding redundant cuts.
- *Loss Bounds in the LP Relaxation*: The downside of lazy cut evaluation is that information related to the loss function is not included in individual nodes, which leads to weaker relaxations. In this case, we addressed this by including an auxiliary variable for the loss in our formulation, and by updating the bounds on this variable through the Initialization Procedure and ChainedUpdates. The lower bound on the loss provides a stronger LP relaxation, which helps the B&B process prune nodes. The effectiveness of ChainedUpdates in this setting suggests that incorporating information from the loss function in the LP relaxation can improve the performance of LCPA so long as it does not greatly increase node processing time. This could be done by generating a small set of judiciously chosen cuts.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

ICU Seizure Prediction

In this chapter, we discuss a collaboration with the Massachusetts General Hospital where we used RiskSLIM to build a customized risk score to predict seizures in intensive care units (ICUs).

Notes

This chapter primarily draws upon material in Ustun and Rudin (2017) and Struck et al. (2017).

7.1 Background

Seizure prediction in the ICU is a difficult problem. Current clinical practice is based on *continuous electroencephalography* (cEEG), which is a technique to monitor electrical activity in the brain by means of electrodes placed on the scalp (see Figure 7.1).

Neurologists are trained to recognize a large set of patterns in cEEG output, some of which may be predictive of seizures (see e.g., Hirsch et al., 2013). The presence and characteristics of cEEG patterns are then used to assess seizure risk, and to decide if patients require a medical intervention, which may be dangerous, or further monitoring, which is expensive.

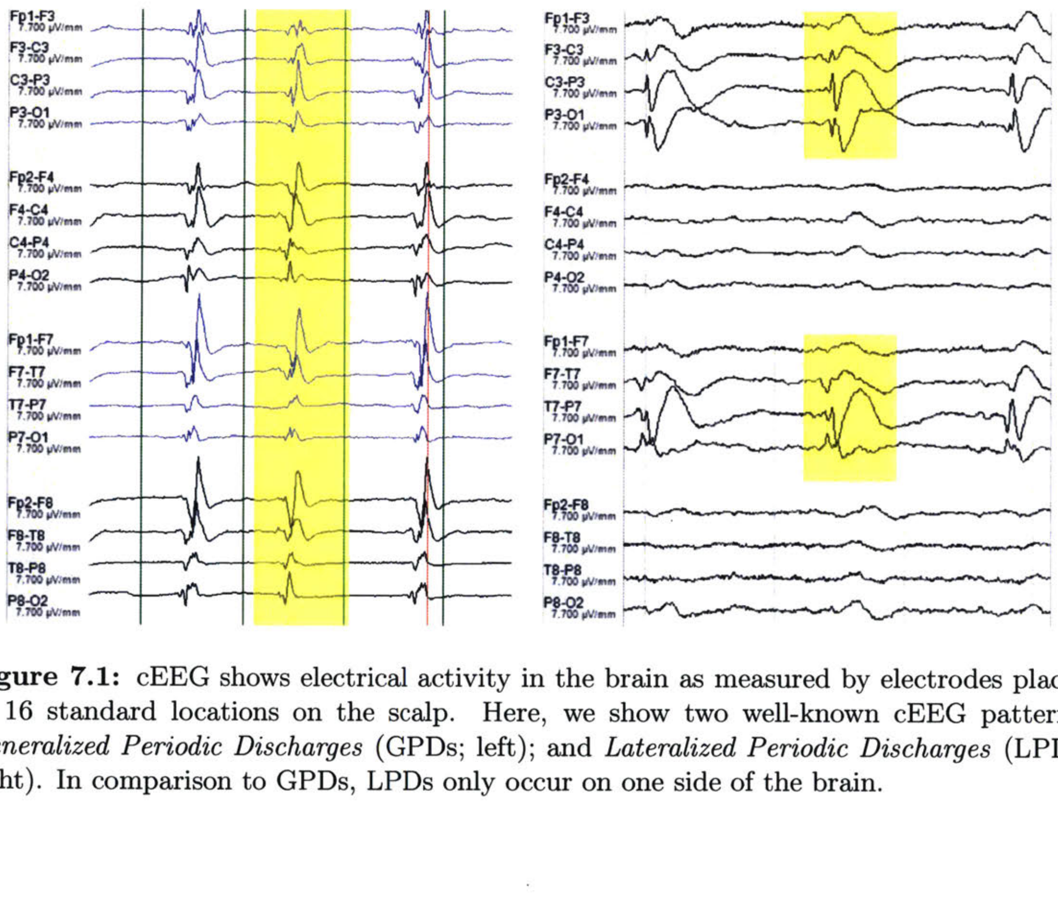


Figure 7.1: cEEG shows electrical activity in the brain as measured by electrodes placed at 16 standard locations on the scalp. Here, we show two well-known cEEG patterns: *Generalized Periodic Discharges* (GPDs; left); and *Lateralized Periodic Discharges* (LPDs; right). In comparison to GPDs, LPDs only occur on one side of the brain.

7.2 Problem Description

Our goal was to build a risk score for seizure prediction that would be easy to use in an intensive care unit and aligned with the domain expertise of the medical community. This required a model that was risk-calibrated, sparse, aligned with domain knowledge, and that let clinicians make predictions without checking too many cEEG patterns.

7.2.1 Data

The training dataset for this problem was built from an extensive set of cEEG recordings of patients at 41 hospitals, curated by the Critical Care EEG Monitoring Research Consortium. It contains a total of $n = 5,427$ recordings and $d = 87$ input variables for each recording. Here, the outcome is $y_i = +1$ if a patient who has been in the ICU for 24 hours will have seizure over the next 24 hours. The classes are highly imbalanced, with $\Pr(y_i = +1) = 12.5\%$ patients having a seizure. The input variables for each recording include information on patient medical history, secondary neurological symptoms, and the presence/characteristics/frequency of 5 well-known cEEG patterns: *Lateralized Periodic Discharges* (LPD); *Lateralized Rhythmic Delta* (LRDA); *Generalized Periodic Discharges* (GPD); *Generalized Rhythmic Delta* (GRDA); *Bilateral Periodic Discharges* (BiPD).

7.2.2 Model Requirements

Our collaborators were interested in customizing the risk assessment tool so that it would prevent doctors from having to check an extensive set of cEEG patterns to make a prediction, and would choose between a large set of potentially redundant cEEG patterns. To this end, we worked with them to identify a large set of operational constraints related to model size, monotonicity, thresholding, and feature composition, namely:

- *Limited Model Size*: In order to be easy to remember, validate, and use by hand in an ICU, our collaborators wanted the model to use at most 4 input variables.
- *Sign Constraints for Established Risk Factors*: To be aligned with domain knowledge, the model had to obey established relationships between well-known risk factors for seizures (e.g., it could not suggest that prior seizures lower seizure risk).
- *No Redundancy between Categorical Variables*: In order to be easy to understand, the model could not use sets of variables that were linearly dependent (e.g., it could include *Male* or *Female* but not both).
- *Specific cEEG Patterns or Any cEEG Pattern*: The dataset included variables for the presence and characteristics of 5 specific cEEG patterns (e.g., *MaxFrequencyLPD*) as well as variables for the presence and characteristics of any pattern (e.g., *MaxFrequencyAnyPattern*[1]). To reduce redundancy, our collaborators wanted the

[1]Here, *MaxFrequencyAnyPattern* := max (*MaxFrequencyLPD*, . . . , *MaxFrequencyBiPD*)

model to use variables for specific patterns or any pattern, but not both.

- *Frequency in Continuous Encoding or Thresholded Encoding*: The dataset included two kinds of variables to represent frequency of each cEEG pattern: (i) a real-valued variable (e.g., $\textit{MaxFrequencyLPD} \in [0, 3.0]$); and (ii) 7 binary threshold variables (e.g., $\textit{MaxFrequencyLPD} \leq \textit{0.5 Hz}, \ldots, \textit{MaxFrequencyLPD} \geq \textit{3.0 Hz}$). Here, our collaborators preferred the binary threshold variables as were easier to check while scanning cEEG output. Since it was not clear if the real-valued variable would result in a better-performing model, however, we included a constraint to ensure that models either used the real-valued variable or the thresholded variable, but not both.
- *Limited # of Thresholds for Thresholded Encoding*: The model could include at most 2 binary thresholded variables related to the frequency of any cEEG pattern. This was done to prevent clinician from having to check multiple thresholds.

We provide a full list of input variables and constraints in Appendix C.

7.2.3 Training Setup

We used the same methods and metrics as in Section 6.4, adapting them to handle the operational constraints as follows. We fit an optimized risk score by solving RiskSlimMINLP with the operational constraints (RiskSLIM). The resulting MINLP had 20 additional constraints, 2 additional variables, and could be solved to optimality in < 20 minutes.

We compared the RiskSLIM risk score to baseline models that were built using advanced heuristics. These methods were designed to avoid the pitfalls of traditional approaches described in Section 6.4. Each method processed the coefficients from $\ell_1 + \ell_2$ penalized logistic regression (PLR) with different kinds of rounding and polishing techniques. These include: naïve rounding (Rd); rescaling then rounding (RsRd); sequential rounding (SeqRd); and versions of these methods where we polished and rounded the coefficients with DCD (i.e., Rd*, RsRd*, and SeqRd* respectively).

All baseline methods had built-in mechanisms to handle sign constraints and model size constraints, but could only handle the other operational constraints by tuning. We used a nested 5-fold cross-validation setup (5-CV) and selected a final model that: (i) satisfied all operational constraints and (ii) maximized the mean 5-CV test AUC.

7.3 Results

We show the performance of the best risk score from all methods in Table 7.3.1. We show the best RiskSLIM, Rd, and RsRd models along with reliability diagrams and ROC curves in Figures 7.2 to 7.4. We include results for other baselines methods in Appendix C, as they perform similarly to Rd model for this problem. Here, Rd, SeqRd, SeqRd*, Rd* produce the same final model. However, Rd*, SeqRd* produce slightly different models on the training folds, which have slightly better performance in terms of test calibration. We show the Rd and RsRd models since they were built using the simplest post-processing techniques.

	OBJECTIVE		CONSTRAINTS			OTHER INFORMATION			
Method	**Test CAL**	**Test AUC**	**Model Size**	**Instances Trained**	**% Feasible Instances**	**Loss Value**	**Opt. Gap**	**Train CAL**	**Train AUC**
RiskSLIM	2.5% 1.9 - 3.4%	0.801 0.758 - 0.841	4	1	100%	0.293	0.0%	2.0%	0.806
PLR	4.5% 3.4 - 6.6%	0.731 0.712 - 0.772	2	1100	0%	0.326	-	3.9%	0.731
Rd	3.7% 2.9 - 5.0%	0.738 0.712 - 0.805	3	1100	12%	0.313	-	1.9%	0.767
RsRd	11.5% 10.7 - 12.7%	0.738 0.712 - 0.805	3	1100	11%	1.003	-	10.3%	0.767
SeqRd	3.7% 2.9 - 5.0%	0.738 0.712 - 0.805	3	1100	12%	0.313	-	1.9%	0.767
Rd*	3.3% 2.8 - 3.7%	0.738 0.712 - 0.805	3	1100	12%	0.313	-	1.9%	0.767
RsRd*	8.2% 6.6 - 10.0%	0.738 0.712 - 0.804	3	1100	12%	0.553	-	9.7%	0.766
SeqRd*	3.3% 2.8 - 3.7%	0.738 0.712 - 0.805	3	1100	12%	0.313	-	1.9%	0.767

Table 7.1: Performance of risk scores for seizure prediction built using RiskSLIM, PLR, and advanced heuristics (see 6.4.1 for details). We report the mean 5-CV mean test CAL and 5-CV mean test AUC. The ranges in each cell represent the 5-CV minimum and maximum. An instance is a unique combination of free parameters for a given method.

1.	*AnyBriefRhythmicDischarge*	2 points		$\cdots$
2.	*PatternsInclude LPD*	2 points	+	$\cdots$
3.	*AnyPriorSeizure*	1 point	+	$\cdots$
4.	*Epiletiform or Discharge*	1 point	+	$\cdots$
	ADD POINTS FROM ROWS 1–4	**SCORE**	=	$\cdots$

SCORE	0	1	2	3	4	5	6
RISK	4.7%	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

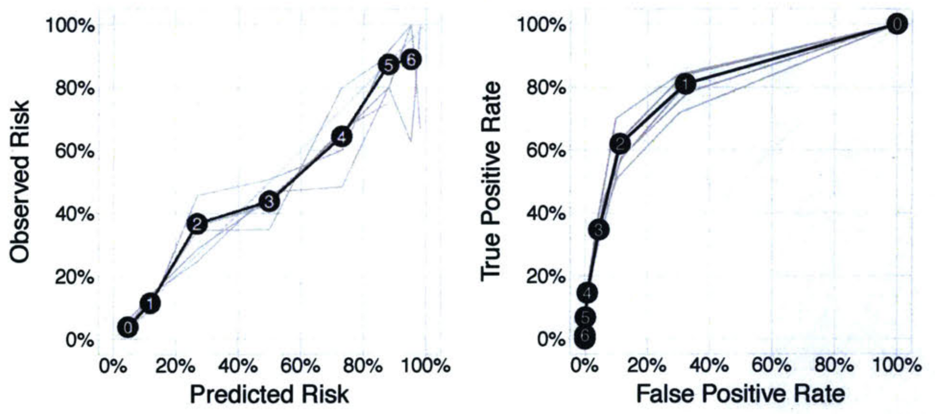


Figure 7.2: RiskSLIM risk score (top), reliability diagram (bottom left), and ROC curve (bottom right) for the `seizure` dataset. We plot results for the final model on training data in black, and results for the fold models on test data in grey. This model has a 5-CV mean test CAL/AUC of 2.5%/0.801.

1.	*AnyPriorSeizure*	1 point	···
2.	*PatternsInclude_ BiPD_ or_ LRDA_ or_ LPD*	1 point	+ ···
3.	*MaxFrequencyLPD*	× 1 point per Hz	+ ···
	ADD POINTS FROM ROWS 1–3	**SCORE**	= ···

SCORE	0.0	1.0	2.0	2.5	3.0	3.5	4.0	4.5	5.0
RISK	4.7%	11.9%	26.9%	37.8%	50.0%	62.2%	73.1%	81.8%	88.1%

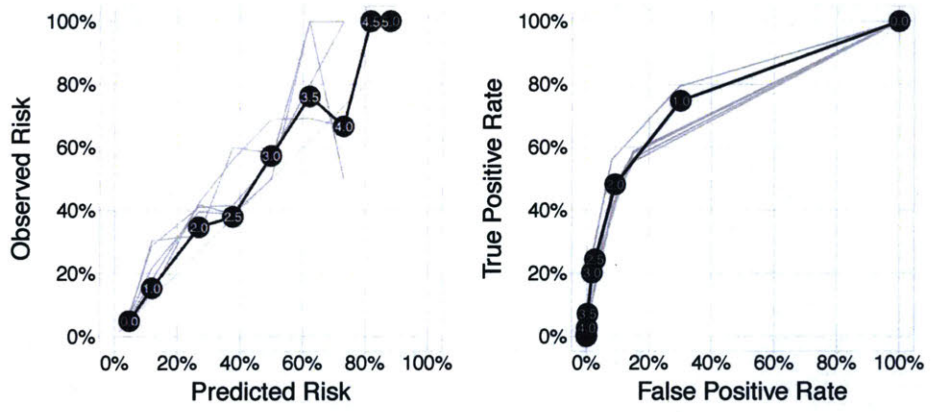


Figure 7.3: RD risk score (top), reliability diagram (bottom left), and ROC curve (bottom right) for the `seizure` dataset. We plot results for the final model on training data in black, and results for the fold models on test data in grey. This model has a 5-CV mean test CAL/AUC of 0.738/3.7%.

1.	*AnyPriorSeizure*	5 points		···
2.	*PatternsInclude_BiPD_or_LRDA_or_LPD*	1 point	+	···
3.	*MaxFrequencyLPD*	5 points per Hz	+	···
	ADD POINTS FROM ROWS 1–3	**SCORE**	=	···

SCORE	0 to 10	12.5	15.0	20.0	20 to 25
RISK	$< 5.0\%$	7.6%	50.0%	92.4%	$> 95.0\%$

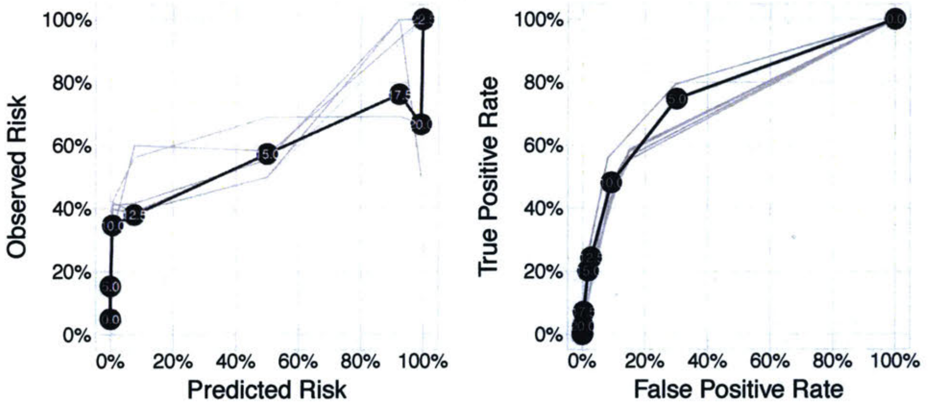


Figure 7.4: RsRd model (top), reliability diagram (bottom left), and ROC curve (bottom right) for the `seizure` dataset. We plot results for fold models on test data in grey, and for the final model on training data in black. This model has a 5-CV mean test CAL/AUC of 11.5%/0.738.

7.3.1 On Performance

Table illustrates the performance benefits of an optimization based approach in a constrained setting: the RiskSLIM model has a 5-CV mean test CAL/AUC of 2.5%/0.801 while the best model from the baseline methods has a 5-CV mean test CAL/AUC of 3.7%/0.738.

As shown in Figures 7.2 to 7.4, models may have important differences in risk calibration over the full reliability diagram despite small differences in CAL. Here: RiskSLIM risk estimates are roughly monotonic and stable; Rd risk estimates are unstable and non-monotonic; and RsRd are skewed towards extreme values due to scaling. As noted by our collaborators, the non-monotonicity of Rd and RsRd risk estimates is problematic as it suggests patients with a score of 3.5 may have more seizures compared to patients with a score of 4.0.

Although the baseline methods did not handle some operational constraints well, these methods still do not achieve the same level of performance as RiskSLIM even when we relax these constraints. If we only consider simple constraints on model size and integer coefficients, for instance, the best model that we can produce using the baseline methods is a SeqRd* model that has a 5-CV mean test CAL/AUC of 2.6%/0.754. To ensure that the operational constraints were not overly restrictive, we also compared the RiskSLIM model to an unconstrained PLR model. In this case, the unconstrained PLR model attains a 5-CV mean test CAL/AUC of 2.7%/0.845. However, our collaborators emphasized that the improved AUC of the PLR model did not outweigh the usability benefits as the PLR model uses 27 terms, does not allow for quick predictions or validation, and violates almost all operational constraints.

7.3.2 On Usability

The RiskSLIM and Rd models have some subtle differences in terms of usability. Here, the RiskSLIM model includes three cEEG patterns (i.e., *PatternsIncludeLPD*, *BriefRhythmicDischarges*, and *EpiletiformDischarge*) while the Rd model includes two (i.e., *PatternsInclude_BiPD_or_LRDA_or_LPD* and *MaxFrequencyLPD*). Thus, the RiskSLIM model requires clinicians to scan cEEG to see if LPD, BriefRhythmicDischarges, and EpiletiformDischarge are present. In comparison, the Rd model requires clinicians to check if BiPD, LRDA, LPD are present, and to also record the frequency of LPD - which requires slightly more time and training.

Figures 7.2 to 7.4 also highlight some of the usability benefits of linear models with small integer coefficients. When input variables belong to a small discrete set, scores also belong to a small discrete set. This reduces the number of operating points on the ROC curve and reliability diagram and makes it easy to pick an operating point. When input variables are binary, risk scores have yet another benefit in that the decision rule at each operating point is a Boolean function. For the RiskSLIM model, for example, the decision rule

$$\text{predict } \hat{y}_i = +1 \text{ if score} \geq 2$$

is equivalent to the Boolean function:

$$\begin{aligned}\text{predict } \textit{Seizure} \text{ if } & \textit{AnyBriefRhythmicDischarge}\\ & \vee \textit{PatternsIncludeLPD}\\ & \vee (\textit{AnyPriorSeizure} \wedge \textit{EpiletiformDischarge}).\end{aligned}$$

Small integer coefficients make it easy to extract such rules by listing the conditions when the score exceeds the threshold. To illustrate this, we show the score function of the PLR model in Table 7.3.1 below.

$$\begin{aligned}\text{score} = & -2.35\\ & + 0.91\ \textit{PatternsIncludeBiPD or LRDA or LPD}\\ & + 0.03\ \textit{AnyPriorSeizure.}\\ & + 0.61 \times \textit{MaxFrequencyLPD}\end{aligned}$$

In this case, it is much harder for users to extract a Boolean function as the score function uses real-valued coefficients, and computing the score requires multiplication due to a real-valued feature (*MaxFrequencyLPD*).

7.3.3 Benefits of Direct Customization

It was difficult to coerce baseline methods to fit models that satisfied the operational constraints as software implementations of penalized logistic regression cannot handle non-trivial operational constraints (e.g., on feature composition).

In practice, when risk scores do not obey constraints, practitioners tweak their model to satisfy the constraints (see e.g., tweaks for EDACS Than et al., 2014, discussed in Section 1.1) . Potential approaches include: manually changing the model to obey constraints (e.g., if the model uses 3 binary threshold variables, remove one); (ii) training models with a dataset that satisfies the constraints a priori (i.e., to satisfy the "frequency in continuous form or thresholded form" constraint, drop the thresholded variables from the dataset), which can impact accuracy and/or usability; or (iii) some kind of exhaustive search over datasets, which is intractable.

In this case, we could find suitable risk scores with the baseline methods by first producing a "pool" of models for over 1,100 free parameter instances, and discarding models that violated any of the operational constraints. As shown in Table 7.3.1, at most 12% of the instances satisfied these constraints for any baseline – which was fortunate given that this approach had no guarantee of producing models that satisfied the operational constraints. Since our approach involved tuning free parameters, we also had to use nested CV to obtain unbiased estimates of performance for each instance, which required fitting a total of 33,000 models for each baseline method (our nested CV setup had 5 outer folds and 5 inner folds on 1,100 free parameter instances, which requires fitting $1,100 \times 5 \times (5 + 1) = 33,000$ models.).

In comparison, RiskSLIM does provide such a guarantee because it explicitly encodes constraints into the risk score problem. Since this approach does not involve parameter tuning, we only need to fit 6 models: 1 model with all the data for deployment, and 5 models with subsets of the data to assess performance using 5-fold CV – far fewer than the 33,000 required with the baseline methods.

7.4 Discussion

Our results in this chapter highlight the benefits of an exact approach in terms of flexibility and performance when learning with operational constraints. In this case, we can address all operational constraints without the need for parameter tuning, nested CV, or post-processing. More importantly, we can recover a certifiably optimal risk score, which attains far superior calibration and AUC. In comparison to the sleep apnea application in Chapter 4, the performance of the model in this chapter is noteworthy because the problem involves a larger number of operational constraints, and because risk calibration is a more difficult performance objective (i.e. we are assuming that the true risk can be modeled as a logit function).

A key question is whether the model in Figure 7.5 that we developed would outperform a model developed using existing methods? Answering this question is difficult given that the model is likely to be developed through an ad hoc training pipeline and make use of domain expertise. We can, however, state the following. A risk score built from using an ad hoc training pipeline would attain lower logistic loss as decisions regarding feature selection and rounding would be made sequentially instead of simultaneously. Unless practitioners could specify a better approximation for the true risk model than the logit model (which would be difficult as the logit model appears to fit the data well), then the ad hoc model would have worse risk calibration. In addition, the ad hoc model could also perform worse than the baseline in this Chapter (which are representative of advanced heuristics that use modern penalized logistic regression, have been extensively tuned, and are designed to avoid several pitfalls listed in Section 6.4).

Clinical Significance

The risk score from this chapter is valuable as the literature in this area has mostly focused on identifying individual cEEG patterns associated with seizures, without controlling for other patterns (see e.g., Shafi et al., 2012). In addition to this model, we show an earlier risk score that we developed for this problem in Figure 7.5 (see Struck et al., 2017, for details). This model was developed using RiskSLIM, outperforms our baselines and those of our collaborators, but does not adhere to the operational constraints of the model presented in this chapter.

1.	*Any cEEG Pattern with Frequency* > ***2 Hz***	1 point		$\cdots$
2.	***E****pileptiform Discharges*	1 point	+	$\cdots$
3.	*Patterns include* ***L****PD or LRDA or BIPD*	1 point	+	$\cdots$
4.	***P****atterns Superimposed with Fast, or Sharp Activity*	1 point	+	$\cdots$
5.	*Prior History of* ***S****eizures*	1 point	+	$\cdots$
6.	***B****rief Rhythmic Discharges*	2 points	+	$\cdots$
	ADD POINTS FROM ROWS 1–6	**SCORE**	=	$\cdots$

SCORE	0	1	2	3	4	5	6+
RISK	<5%	12%	27%	50%	73%	88 %	>95%

Figure 7.5: 2HELPS2B risk score built by RiskSLIM (see Struck et al., 2017, for details). This model has a 5-CV mean test CAL/AUC of 2.7%/0.819.

Automated Seizure Prediction with Humans in the Loop

Our interesting area for future work is to design a hybrid system for automated seizure prediction where we build a risk score such as the one in Figure 7.2 to predict seizures, and use black-box models (e.g. recurrent convolutional neural nets) to detect individual cEEG patterns.

In comparison to a fully automated model for seizure prediction (see e.g., Thodoroff et al., 2016), where, for example, we train a single neural network to directly predict seizures from cEEG input, a hybrid system would be more difficult to develop. It would require a dataset where individual cEEG patterns are labeled by humans and would require training several neural networks. In light of these difficulties, a hybrid model would provide physicians with the ability to validate predictions, which may lead to improved adoption and performance during deployment.

A hybrid model would not outperform the risk score (unless the cEEG patterns in the dataset were mislabeled). However, it may outperform a fully automated model because it may be easier to detect individual cEEG patterns rather than the final output, and because we can train the risk score using far more data (e.g., the model in Thodoroff et al., 2016, was built using $n = 23$ patients). In this approach, operational constraints could also be used to reduce the number of neural networks that need to be trained. For instance, we could use hierarchical feature selection constraint to enforce priorities between increasingly difficult prediction problems (e.g. predict specific cEEG pattern $\prec$ predict any cEEG pattern).

Chapter 8

Adult ADHD Diagnosis

In this chapter, we discuss a collaboration where we used RISKSLIM to develop a risk score for adult ADHD diagnosis (Ustun et al., 2017). Our results provide insight into the different aspects of real-world model development, and illustrates the flexibility and performance benefits our approach in this setting.

Notes

This chapter primarily contains material from Ustun et al. (2017). An editorial regarding the clinical significance of this work can be found in Shaw et al. (2017).

8.1 Background

Attention-Deficit/Hyperactivity Disorder (ADHD) is a common childhood-onset disorder often persisting into adulthood (Faraone et al., 2006). Adult ADHD is associated with work impairment (Kessler et al., 2005a), accidents (Küpper et al., 2012), and early mortality (London and Landes, 2016). Nevertheless, the condition remains undetected in the general population(Fayyad et al., 2017), and untreated despite evidence that treatment may be effective (Surman et al., 2013). As a result, there has been substantial interest in effective screening tools for adult ADHD in primary care (Minkoff, 2009) and workplace settings (Kessler et al., 2009). The vast majority of screening tools for adult ADHD calibrated to older diagnostic criteria (Taylor et al., 2011), and built through ad hoc training pipelines that combine statistical techniques with expert judgment.

Previous ASRS Model

The RISKSLIM risk score in this chapter replaces a widely used scoring system, known as the *Adult ADHD Self-Report Scale* (ASRS v1.1 by Kessler et al., 2005b), shown in Figure 8.1. In comparison to its predecessor, which was designed as a decision-making model, our model outputs risk estimates that provide several decision-points. The risk estimates are calibrated for the general population of the United States, and that reflect the most recent diagnostic criteria for adult ADHD.

As many other scoring systems and screening tools, ASRS v1.1 was also built using an ad hoc training pipeline (see Kessler et al., 2005b, for details). In this case, the approach used statistical significance tests to transform the responses to each question into thresholded binary variables. The authors then used stepwise feature selection determine a subset of thresholded responses. The coefficients in the final model were rounded so that the score could be computed by hand. This procedure was repeated model times to find a model that performed well for different subpopulations at a clinically relevant decision point (e.g. for males and females).

	Never	Rarely	Sometimes	Often	Very Often
How often do you have trouble wrapping up the final details of a project, once the challenging parts have been done?	0	0	1	1	1
How often do you have difficulty getting things in order when you have to do a task that requires organization?	0	0	1	1	1
How often do you have problems remembering appointments or obligation?	0	0	1	1	1
When you have a task that requires a lot of thought, how often do you avoid or delay getting started?	0	0	1	1	1
How often do you fidget or squirm with your hands or feet when you have to sit down for a long time?	0	0	0	1	1
How often do you feel overly active and compelled to do things, like you were driven by a motor?	0	0	0	1	1

SCREEN PATIENT for ADULT ADHD IF SCORE > 3

Figure 8.1: Previous version of the Adult ADHD Self-Report Scale (ASRS v1.1) developed by Kessler et al. (2005b). This model was to diagnose DSM-4 adult ADHD. It was built using an ad hoc training pipeline that was designed over the course of several months in order to effectively balance predictive accuracy with other constraints on model form. The model is designed to be used as a decision-making tool, which reduces its applicability in other settings.

8.2 Problem Description

We built our model using a *national sample* and validated it on a *clinical sample.* In what follows, we describe the outcome variable, input variables, and the various samples associated with the problem. Additional details related to data collection and study design can be found in (Ustun et al., 2017).

Outcome Variable

The outcome variable is a clinical diagnosis for adult ADHD (i.e. $\hat{y}_i = +1$ if patient i has received a DSM-5 diagnosis for adult ADHD). The clinical diagnosis is based on a well-established semi-structured research diagnostic interview, known as the *Adult Clinician ADHD Diagnostic Scale* (ACDS). The ACDS provides specific prompts to assess the presence and severity of ADHD symptoms during childhood and adulthood. The clinical diagnosis is determined as per DSM-5 criteria for adult ADHD, which uses presence and severity of these symptoms. Explicitly, the DSM-5 criteria for adult ADHD require:

(A) At least 5-of-9 symptoms of attention (6-of-9 for patients under 17) OR 5-of-9 symptoms of hyperactivity (6-of-9 for patients under 17).

(B) Several symptoms were present prior to age 12.

(C) Evidence that several symptoms occur in 2 or more settings (e.g. at work, school, or home) over the past 6 months.

(D) Evidence that several symptoms result in clinically significant impairment over the same time period over the past 6 months.

(E) Evidence that symptoms do not occur exclusively during the course of a pervasive developmental disorder or psychotic disorder, and are not better accounted for by another mental disorder.

Our study not did consider the DSM-5 requirement of impairment before age 12 (Criterion B), or the possibility that ADHD may have been better explained by a different mental illness (Criterion E). If these requirements would have resulted in an ADHD diagnosis, then they were not diagnosed.

Input Variables

The input variables were derived from a self-administered questionnaire about adult ADHD used for the ASRS v1.1 Kessler et al. (2005b). We show the full set of 27 questions in Table 8.1. The questions were developed by two board-certified psychiatrists and a World Health Organization advisory group in order to effectively determine the presence and severity of a particular symptom of adult ADHD (see Kessler et al., 2010, for details). The responses to each question recorded on a 5-level Likert scale as *Never*, *Rarely*, *Sometimes*, *Often*, or *Very Often.*

ASRS Question
How often do you have trouble wrapping up a project, once the challenging parts are done?†
How often do you have difficulty getting things in order when you have to do a task that requires organization?
How often do you have problems remembering appointments or obligations?
When you have a task that requires a lot of thought, how often do you avoid or delay getting started?
How often do you fidget or squirm with your hands or feet when you have to sit down for a long time?
How often do you feel overly active and compelled to do things, like you were driven by a motor?
How often do you make careless mistakes when you have to work on a boring or difficult project?
How often do you have difficulty keeping your attention when you are doing boring or repetitive work?
How often do you have difficulty concentrating on what people say to you, even if they speak to you directly?†
How often do you misplace or have difficulty finding things at home or at work?
How often are you distracted by activity or noise around you?
How often do you leave your seat in meetings or other situations in which you are expected to remain seated?
How often do you feel restless or fidgety?
How often do you have difficulty unwinding and relaxing when you have time to yourself?
How often do you find yourself talking too much when you are in social situations?
How often do you find yourself finishing the sentences of the people you are talking to?
How often do you have difficulty waiting your turn in situations when turn-taking is required?
How often do you interrupt others when they are busy?
How often do you waste or mismanage time?
How often do you have trouble making a plan and sticking to it in situations where planful behavior is needed?†
How often do you have difficulty prioritizing work when you are in a situation where setting priorities is needed?
How often do you depend on others to keep your life in order and attend to details?
How often do you put things off until the last minute?
How often is it hard for you to complete tasks in the allotted time?
How often do you have trouble remembering the main idea in things that you have read?
How often do you find that your mood is easily changeable?
How often do you feel more easily hassled or overwhelmed than other people in your situation?
How often do you have a hard time controlling your temper?
How often are your feelings easily hurt when you are criticized?

Table 8.1: Questions in the Adult ADHD Self-Report Scale (Kessler et al., 2005b). Questions marked with † have been abridged in order to fit in the table (see Kessler et al., 2010, for a full list).

National Sample

The data for the national sample was collected through two general population surveys: (i) a national household survey that covered a cohort of 119 patients; (ii) a survey from a large managed healthcare plan that covered a cohort of 218 patients.

Both studies first administered the full ASRS questionnaire to each patient, then followed up with a semi-structured diagnostic interview to pair these responses with a DSM-5 adult ADHD diagnosis. To ensure the final model could output risk estimates that were calibrated to the general population, each patient in the national sample was paired with a sampling weight w_i to adjust for differential sampling across strata.

Clinical Sample

The clinical sample covered a cohort of 300 patients who were obtaining a free evaluation for adult ADHD at the NYU Langone Medical Center. As with the national sample, all patients were administered the full ASRS questionnaire, and then given a DSM-5 adult ADHD diagnosis through a blinded semi-structured interview. In contrast to the national sample, the overall prevalence of adult ADHD is much higher in the clinical sample (8.2% in the national sample vs. 57.7% in the clinical sample).

8.3 Methodology

We aimed to create a DSM-5 ASRS screening scale with the same form as its predecessor: a simple risk score that used a few questions and small integer coefficients.

Model Requirements

Our collaborators specified the following requirements for the model:

- *Limited Number of Questions*: Like its predecessor, the model has to output a prediction using the answers to at most 6 questions.
- Additive Model: The model had to be highly usable. In particular, we were required to use only addition, minimize the use of subtraction, and avoid multiplication.
- *Monotonicity in the Presence of Symptoms*: Since the response to each was associated with a well-known symptom for ADHD, responses such as *Rarely*, *Sometimes*, *Often* and *Very Often* had to increase predicted risk (i.e. the points for these responses had to be ≥ 0). In the same way, a response such as *Never* had to decrease the predicted risk (i.e. the points for these responses had to be ≤ 0).
- *Monotonicity in the Severity of Symptoms*: Since the response to each question captured the severity of each symptom, the predicted risk of the model had to increase when the response to each question went from less frequent to more frequent. (i.e. the points for *Rarely* could not exceed the points for *Sometimes*).

Data Processing

We processed the dataset before training as follows.

We defined 4 nested threshold variables using the responses to each question. Specifically, given the response of patient i to question k,

$$r_{i,k} \in \{\textit{Never}, \textit{Rarely}, \textit{Sometimes}, \textit{Often}, \textit{Very Often}\},$$

we defined:

$$\begin{aligned}
x_{i,k_1} &= \mathbb{1}\left[r_k \in \{\textit{Rarely}, \textit{Sometimes}, \textit{Often}, \textit{Very Often}\}\right], \\
x_{i,k_2} &= \mathbb{1}\left[r_k \in \{\textit{Sometimes}, \textit{Often}, \textit{Very Often}\}\right], \\
x_{i,k_3} &= \mathbb{1}\left[r_k \in \{\textit{Often}, \textit{Very Often}\}\right], \\
x_{i,k_4} &= \mathbb{1}\left[r_k \in \{\textit{Very Often}\}\right].
\end{aligned}$$

The nested threshold variables let us address monotonicity requirements in a way that reduced the size of the problem. Specifically, with the nested threshold variables, each coefficient represents the number of additional points received for each response level. Thus, we can ensure that the number of points for each response level increases by constraining the sign of each coefficient to be positive. This also allowed to capture information for the *Never* option via the intercept as

$$\mathbb{1}\left[r_{i,k} \in \{\textit{Never}\}\right] = 1 - \mathbb{1}\left[r_k \in \{\textit{Rarely}, \textit{Sometimes}, \textit{Often}, \textit{Very Often}\}\right].$$

In other words, the points would be "hidden" in the intercept term (assuming that the coefficient were negative, and the intercept is large enough).

In addition, we removed the sampling weights as follows. We divided the original weights in each sample by the smallest weight, rounded the scaled weight to the nearest whole number, and duplicated records for each respondent a number of times equal to the rounded weight.

As a result of these changes, the final training dataset had $n = 33,672$ points and $d = 116$ variables. Since the clinical sample did not contain sampling weights, the validation dataset had $n = 300$ points and $d = 116$ variables.

MINLP Formulation

We trained our model by solving a customized version of the risk score problem. We present the formulation in Definition (8.1) as it can be used to reproduce a popular format for scoring systems used in the field of mental health (see e.g. the models of Altman et al., 1997; Weathers et al., 2013, which also use responses on a Likert scale).

Definition 8.1 (RISKSLIMMINLP for Risk Scores with a Likert Questionnaire) *Given a dataset where the features are responses to questions on a 5-level Likert scale, the coefficients of a risk score that uses at most $Q^{\max}$ questions can be determined by solving the following MINLP:*

$$
\begin{array}{llll}
\min & V & & (8.1a)\\
\text{s.t.} & V = L + C_0 Q & & \textit{objective}\ (8.1b)\\
& L = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-\langle \boldsymbol{\lambda}, y_i \boldsymbol{x}_i\rangle)) & & \textit{total loss}\ (8.1c)\\
& Q = \sum_{k=1}^{K} \mu_k & & \textit{total questions}\ (8.1d)\\
& R = \sum_{j=1}^{d} \alpha_j & & \textit{total coefs}\ (8.1e)\\
& \mu_k \geq \sum_{j\in J_k} \alpha_j & k=1,\ldots,21 & \textit{indicator for q k}\ (8.1f)\\
& \Lambda^{\max}\alpha_j \geq \lambda_j & j=1,\ldots,d & \textit{indicator for coef j}\ (8.1g)\\
& \lambda_j \geq \alpha_j & j=1,\ldots,d & \textit{indicator for coef j}\ (8.1h)\\
& 4R \leq Q & & \textit{max 4 coefs/q}\ (8.1i)\\
& R \geq Q & & \textit{min 1 coef/q}\ (8.1j)\\
& \lambda_0 \in \{-4\Lambda^{\max}Q^{\max},\ldots,0\} & & \textit{intercept}\\
& \lambda_j \in \{0,\ldots,\Lambda^{\max}\} & j=1,\ldots,d & \textit{coefficient values}\\
& \alpha_j \in \{0,1\} & j=1,\ldots,d & \textit{coefficient indicators}\\
& \mu_k \in \{0,1\} & j=1,\ldots,k & \textit{question indicators}\\
& V \in [V^{\min}, V^{\max}] & & \textit{objective value}\\
& L \in [L^{\min}, L^{\max}] & & \textit{loss value}\\
& Q \in \{0,\ldots,Q^{\max}\} & & \textit{total questions}\\
& R \in \{R^{\min},\ldots,R^{\max}\} & & \textit{total coefs}
\end{array}
$$

The MINLP formulation includes several changes to address operational constraints and reduce computation, namely:

- We define an integer variable Q to represent the total number of questions in the model. We set the value of Q in Constraint (8.1d) as the sum of binary indicator variables μ_k for $k = 1,\ldots,21$. Here, we force $\mu_k = 1$ when the model uses a non-zero coefficient from question k in Constraint (8.1f).

- We penalize the total number of questions Q instead of the number of coefficients R in (8.1b). We restrict the total number of questions to 6 through variable bounds on Q. We then set $C_0 = 10^{-8}$ to recover a solution from the C_0-level set of minimizers to the logistic loss.
- We strengthen the LP relaxation used in LCPA using additional constraints. Specifically, we include variable bounds on V and L, which we set using values from the initialization procedure in Chapter 9). Since there can be at most 4 non-zero coefficients for each question, we limit the maximum number of non-zero coefficients using Constraint (8.1i). Since there must be at least 1 non-zero coefficient whenever a question is included, we force this condition in Constraint (8.1i). This also requires an additional constraint (8.1h) so that $\alpha_j = 1$ if $\lambda_j > 0$.

Computation

We trained our model using a standard 10-CV setup on the national sample (i.e. we fit 10 instances on subsets of the training dataset, and 1 instance using the full training dataset). For each instance, we first ran the initialization procedure (Algorithm 9) to produce an initial collection of cutting planes. We paired LCPA with the lookup technique from Section 6.3.3 to reduce data-related computation. We solved each instance using LCPA using the CPLEX 12.6.3 Python API on a 3.33GHz CPU with 16GB RAM, setting a time limit of 6 hours.

8.4 Results

We show the RISKSLIM model that we built in Figure 8.2, and summarize its performance via a reliability diagram and ROC curve in Figure 8.3. As shown, the model is highly usable and obeys all constraints specified by our collaborators. Here, the model attains near optimal loss (optimality gap of 6.1%), has excellent CAL/AUC, and generalizes (training CAL/AUC of 1.5%/0.973; the 10-CV CAL ranges 1.4 to 2.1% and the 10-CV AUC ranges between 0.965 to 0.980).

	Never	Rarely	Sometimes	Often	Very Often
How often do you have trouble concentrating on what people say to you even when they speak to you directly?	0	4	4	5	5
How often do you leave your seat in meetings or other situations in which you are expected to remain seated?	0	0	1	1	5
How often do you have difficulty unwinding and relaxing when you have time to yourself?	0	4	4	6	6
How often do you find yourself finishing the sentences of the people you talk to before they can finish them themselves?	0	0	2	2	2
How often do you put things off until the last minute?	0	2	2	4	4
How often do you depend on others to keep your life in order and attend to details?	0	2	3	3	3

SCORE	0 to 13	14	15	16	17	18	19 to 26
RISK	<5.0%	11.9	26.9%	50.0%	73.1%	88.1%	>95.0

Figure 8.2: RISKSLIM risk score for DSM-5 Adult ADHD.

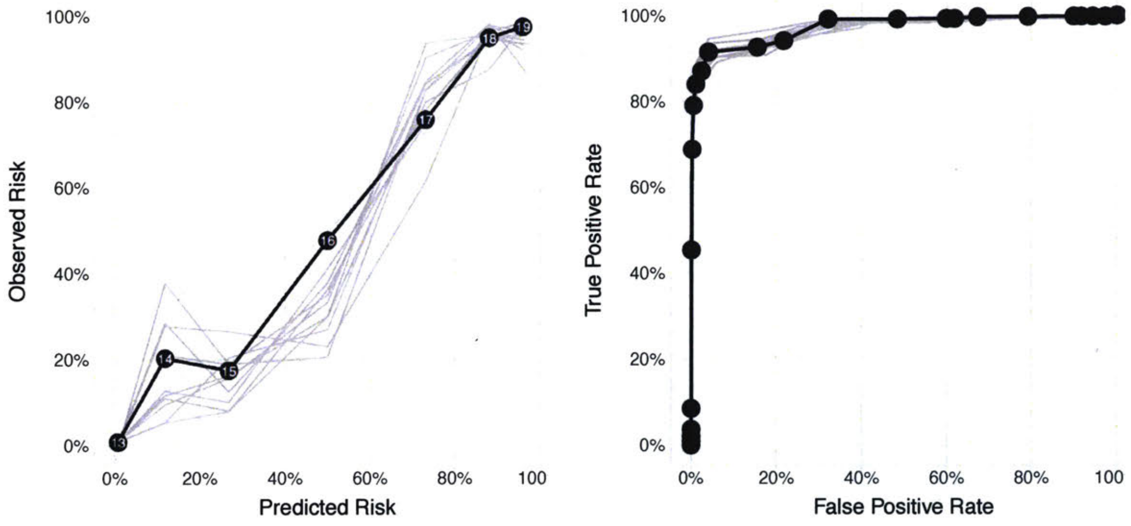


Figure 8.3: Performance of RiskSLIM risk score for DSM-5 adult ADHD on the national sample.

One of the advantages in building a risk assessment tool for this problem was that it provided a range of decision rules that would be useful in different settings (e.g. for screening on a new clinical population, or for a non-clinical application such as an epidemiological study). In Table 8.2, we present the performance of decision rules from the RiskSLIM model at various threshold scores on the national sample. Using this table, our collaborators determined several thresholds for different use cases. They decided, for instance, that the suitable threshold for screening was 14. At this threshold score, the decision rule achieves a sensitivity of 91.4%, specificity of 96.0% and PPV of 67.3% (i.e. we are expected to screen around 11.2% of respondents, capturing 91.4% of cases, and expect 67.3% of screened positives to be diagnosed with ADHD). For an epidemiological study, they recommended that a suitable threshold was 16, as this would result in an estimated prevalence that matched the estimated prevalence of adult ADHD in the literature.

Threshold Score	Predicted Risk	% Cases Screened	Sensitivity	Specificity	PPV
0	0.00%	100.00%	100.00%	0.00%	8.21%
1	0.00%	97.46%	99.75%	2.74%	8.40%
2	0.00%	94.83%	99.75%	5.61%	8.63%
3	0.00%	94.79%	99.75%	5.65%	8.64%
4	0.00%	92.24%	99.75%	8.43%	8.88%
5	0.00%	90.71%	99.75%	10.10%	9.03%
6	0.00%	80.87%	99.75%	20.81%	10.12%
7	0.01%	69.91%	99.75%	32.76%	11.71%
8	0.03%	64.86%	99.39%	38.23%	12.58%
9	0.09%	63.03%	99.39%	40.22%	12.94%
10	0.25%	52.44%	99.24%	51.74%	15.53%
11	0.67%	37.48%	99.24%	68.05%	21.74%
12	1.80%	27.52%	94.00%	78.43%	28.04%
13	4.74%	21.79%	92.59%	84.54%	34.88%
14	11.92%	11.16%	91.43%	96.02%	67.26%
15	26.89%	9.33%	86.98%	97.61%	76.53%
16	50.00%	7.82%	84.06%	99.00%	88.24%
17	73.11%	6.97%	79.21%	99.49%	93.23%
18	88.08%	5.88%	69.27%	99.78%	96.62%
19	95.26%	3.80%	45.08%	99.89%	97.35%
20	98.20%	0.58%	6.51%	99.95%	91.37%
21	99.33%	0.19%	2.13%	99.98%	90.77%
22	99.75%	0.12%	1.45%	100.00%	100.00%
23	99.91%	0.12%	1.45%	100.00%	100.00%
24	99.97%	0.05%	0.61%	100.00%	100.00%
25	99.99%	0.04%	0.43%	100.00%	100.00%

Table 8.2: Performance of decision rules from the RiskSLIM model on the national sample. Each row shows accuracy metrics for the decision rule "predict $\hat{y}_i = +1$ if score $\geq s$" for scores between 0 to 25.

8.4.1 Impact of Heuristic Feature Selection and Rounding

Our collaborators were interested to see how the performance of the RiskSLIM model in Figure 8.2 fared against two other baseline models:

- ASRS v1.1 (Figure 8.1), to see how this model would perform on the national sample;
- A RiskSLIM model that used the same questions as ASRS v1.1 (Figure 8.4), which would be useful for researchers in the field who had already collected data using ASRS v1.1.

In Table 8.3, we compare the RiskSLIM and these baselines on the national sample. As shown, ASRS v1.1 fares poorly in terms of risk calibration and AUC, which is expected given that the model was trained using a different dataset, and designed to optimize performance at a particular decision point. In comparison, the RiskSLIM model built using the same questions attains far better CAL and AUC as it can search effectively over a far larger set of acceptable models. However, it still underperforms compared to a model that optimizes the choice of questions. These results highlight the performance benefits from an optimization-based approach, as well as the need for a principled method to build such models. Without such a method in place, the ASRS v1.1 would still be used in practice and lead to poor decisions.

Model	**Heuristic Components**	**CAL**	**AUC**	**Loss**
RiskSLIM	-	1.5% 1.4 - 2.1	0.973 0.965 - 0.980	0.082 0.073 - 0.098
RiskSLIM with ASRS v1.1 Questions	Feature Selection	2.8% 1.6 - 3.4	0.928 0.914 - 0.937	0.156 0.144 - 0.174
ASRS v1.1	Feature Selection Coefficient Values	89.6% 89.0 - 90.1	0.822 0.803 - 0.835	3.978 3.937 - 4.027

Table 8.3: Performance of the RiskSLIM model (Figure 8.2), ASRS v1.1 (Figure 8.1), and a baseline model built using the same questions as ASRS v1.1 (Figure 8.4). We show the CAL, AUC, and logistic loss of the model trained on the full national sample (top) and the 10-CV min-max (bottom).

	Never	Rarely	Sometimes	Often	Very Often
How often do you have trouble wrapping up the final details of a project, once the challenging parts have been done?	0	3	3	4	6
How often do you have difficulty getting things in order when you have to do a task that requires organization?	0	0	1	1	2
How often do you have problems remembering appointments or obligation?	0	0	0	0	2
When you have a task that requires a lot of thought, how often do you avoid or delay getting started?	0	1	1	1	1
How often do you fidget or squirm with your hands or feet when you have to sit down for a long time?	0	1	1	3	3
How often do you feel overly active and compelled to do things, like you were driven by a motor?	0	3	3	3	6

SCORE	0 to 9	10	11	12	13	14	15 to 20
RISK	<5.0%	11.9	26.9%	50.0%	73.1%	88.1%	>95.0

Figure 8.4: RiskSLIM model fit using the same 6 questions as ASRS v1.1.

8.4.2 Clinical Validation

In Figure 8.5, we show the performance of the RiskSLIM model from Figure 8.2 on the clinical sample. As shown, the performance of the model decreases in terms of risk calibration (CAL of 10.0% vs 2.0% on the national sample) and rank accuracy (AUC of 0.921 vs. 0.978 on the national sample). However, the performance is still remarkable given the significant differences in the demographic composition of the national and clinical samples (e.g. the prevalence of adult ADHD is 8.2% in the national sample, but 57.7% in the clinical sample).

As shown in Table 8.4, the RiskSLIM model maintains its sensitivity across several decision-points and the previous screening rule (i.e. predict ADHD if the total score ≥ 14) captures 91.9% of patients with adult ADHD. In light of the increased prevalence of ADHD in the clinical population, the false positive rate (26.0%) is considerably higher than in the national sample, and PPV quite high (82.8%). As in the national sample, however, the screening ability decreases markedly if we change the threshold score to 13. In this case, our collaborators determined that the marginal increase in the false positive rate (37.0%) did not justify the marginal increase in sensitivity (93.1%). Thus, they determined that a threshold of 14 was still appropriate for a clinical setting.

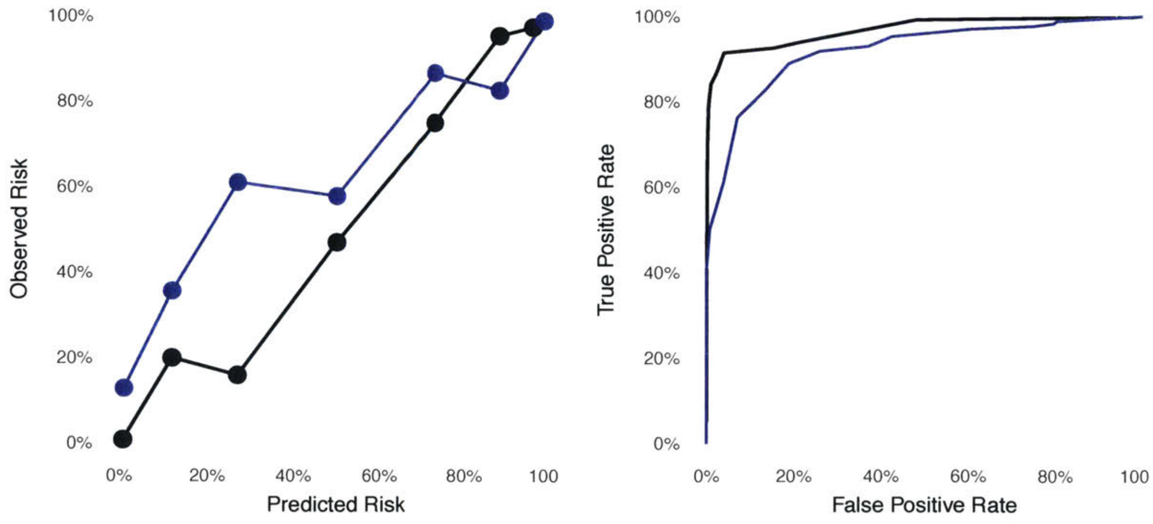


Figure 8.5: Reliability diagram (left) and ROC curve (right) for the RiskSLIM model on national sample (black), and the clinical sample (blue)

Threshold Score	**Predicted Risk**	**% Cases Screened**	**Sensitivity**	**Specificity**	**PPV**
0	0.00%	100.00%	100.00%	0.00%	57.67%
1	0.00%	99.00%	100.00%	2.36%	58.25%
2	0.00%	97.67%	100.00%	5.51%	59.04%
3	0.00%	97.67%	100.00%	5.51%	59.04%
4	0.00%	97.33%	100.00%	6.30%	59.25%
5	0.00%	95.67%	100.00%	10.24%	60.28%
6	0.00%	91.00%	98.84%	19.69%	62.64%
7	0.01%	90.33%	98.27%	20.47%	62.73%
8	0.03%	88.00%	97.69%	25.20%	64.02%
9	0.09%	86.67%	97.69%	28.35%	65.00%
10	0.25%	81.67%	97.11%	39.37%	68.57%
11	0.67%	80.00%	97.11%	43.31%	70.00%
12	1.80%	73.00%	95.38%	57.48%	75.34%
13	4.74%	69.33%	93.06%	62.99%	77.40%
14	11.92%	64.00%	91.91%	74.02%	82.81%
15	26.89%	59.33%	89.02%	81.10%	86.52%
16	50.00%	53.33%	82.66%	86.61%	89.38%
17	73.11%	47.00%	76.30%	92.91%	93.62%
18	88.08%	37.00%	61.27%	96.06%	95.50%
19	95.26%	29.33%	50.29%	99.21%	98.86%
20	98.20%	23.33%	40.46%	100.00%	100.00%
21	99.33%	16.33%	28.32%	100.00%	100.00%
22	99.75%	6.33%	10.98%	100.00%	100.00%
23	99.91%	5.67%	9.83%	100.00%	100.00%
24	99.97%	4.00%	6.94%	100.00%	100.00%
25	99.99%	3.67%	6.36%	100.00%	100.00%

Table 8.4: Performance of decision rules from the RISKSLIM model on the clinical sample. Each row shows accuracy metrics for the decision rule "predict $\hat{y}_i = +1$ if score $\geq t$" for thresholds $t = 0, \ldots, 25$.

8.5 Discussion

Our work in this chapter illustrates how our approach can build a risk assessment tool that performs well and adheres to a strict set of requirements on model form. In this case, our model can reproduce DSM-5 ADHD diagnosis in a way that is effective but also provides users with the ability to make quick predictions and to fully validate the model (see Ustun et al. 2017 and the editorial of Shaw et al. 2017).

The MINLP formulation in this chapter captures the exact requirements of many scoring systems developed for mental health applications (see e.g. the models of Altman et al., 1997; Weathers et al., 2013, which also use questions on a Likert scale). In comparison to current models, the optimization-based approach used in this chapter avoids the need for an ad hoc training pipeline that can take several months to design. As shown by the results in Section 8.4.1, the resulting models are likely to be attain better risk-calibration and AUC. They may also be easier to use in other aspects of model development as they generalize, have a clearly defined performance metric, and a certificate of optimality.

One of the key lessons from this chapter was that, although our collaborators wanted a model that could output calibrated risk estimates on the general population, the broader acceptance of our model was implicitly dependent on other performance metrics (e.g. a decision point with good specificity/sensitivity, performance on a clinical population). Here, we were fortunate as our model that performed well on the national sample as well as the clinical sample, and produced a suitable decision-point for both settings. Nevertheless, our experience revealed a need for further methodological work in this area. Specifically:

(i) Methods that can enforce constraints to ensure calibration across subpopulations.

(ii) Methods that can produce risk assessment models that are guaranteed to produce at least one good decision rule.

It may be possible to address both issues using the methods presented in this dissertation. In particular, the first issue could be addressed by including fairness constraints on major demographic groups in the RiskSlimMINLP formulation in Definition 8.1 (e.g. to limit the disparity in FPR across males, female, and major age groups). The second issue could be addressed by exploiting the fact that the decision rules of an optimized risk score cannot outperform the decision rule of an optimized scoring system (assuming that both models have to obey the same constraints). As a result, the latter issue could be addressed by first training a SLIM scoring system that optimizes TPR subject to an FPR constraint as in the sleep apnea problem in Chapter 4, and then using the training TPR of the SLIM to formulate appropriate TPR and FPR constraints for the RiskSlimMINLP formulation in Definition 8.1. In this case, suitable constraints would force RiskSlimMINLP to fit a risk score such that $\text{TPR}_{\textit{RiskSLIM}} \leq TPR_{SLIM} - \varepsilon_{\text{TPR}}$ and $\text{FPR}_{\textit{RiskSLIM}} \leq \text{FPR}_{SLIM} - \varepsilon_{\text{FPR}}$ for at least one threshold λ_0, where $\varepsilon_{TPR} \geq 0$ and $\varepsilon_{\text{FPR}} \geq 0$ are acceptable deviations in the TPR and FPR.

Why Discrete Linear Models Perform Well in Mental Health Applications

The performance of the RiskSLIM model in Figure 8.2 is noteworthy. It has a 10-CV test AUC ranging between 0.965-0.980 on the national sample, and a test AUC of 0.921 on the clinical sample. From a machine learning perspective, one possible explanation for this result is that there exists a well-defined "true model" for this prediction problem, and that we can learn a very good approximation of this model from a real-world dataset.

In this case, the true model is a DSM-5 diagnostic rule for adult ADHD. In our study, this is based on the following explicitly-defined criteria:

1. Patient has at least 6-of-9 symptoms associated with hyperactivity OR at least 6-of-9 symptoms associated with inattention;
2. At least 1 symptom appears in 2 or more settings (e.g. work, home);
3. At least 1 symptom interferes with functioning.

In other words, the true model is a Boolean function of the form

$$f^{\text{DSM}} : (\text{symptom}, \text{setting}, \text{interference}) \rightarrow \{-1, +1\}.$$

Here, we have sought to fit approximate this model using models of the form

$$f^{\text{ASRS}} : (\text{question}, \text{response}) \rightarrow \{-1, +1\}.$$

Our result suggests that we can approximate f^{DSM} using f^{ASRS}, where f^{ASRS} uses small integer coefficients, only considers a small subset of all symptoms, and produces calibrated risk estimates. There are several reasons to explain why this is the case:

- The questions from the ASRS are designed to not only identify symptoms, but also assess severity, and capture to key interactions with the diagnosis (e.g. setting, interference).
- We can learn the model from data because the outcome variable is obtained through a careful application of the DSM-5 guidelines (i.e., the labels represent the true output of f^{DSM} and are recorded without noise).
- The true model f^{DSM} is very similar to a linear threshold rule, which can be represented by a linear classifier with finite integer coefficients as per Theorem 2.1 (or approximated, as suggested by the results in Long and Servedio, 2014). In particular, the diagnostic rule primarily depends on two 5-of-9 rules, and the additional conditions may be captured through the ASRS questionnaire or may be unimportant.

Given that a wide range of mental health conditions have similar rule-based DSM diagnoses, it should be possible to build scoring systems that perform similarly for these conditions. These models would not only help in screening, but may help inform future DSM guidelines (which are typically decided by a committee of experts). Say, for example, that we are able to construct a predictive model to replicate the DSM diagnoses on a broad population. If this model can consistently replicate a DSM diagnosis using a subset of symptoms, different questions, or alternative criteria, then it may be possible use this result to reduce the number of symptoms in future DSM iterations.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

A. Abrishami, A. Khajehdehi, and F. Chung. A systematic review of screening questionnaires for obstructive sleep apnea. *Canadian Journal of Anesthesia*, 57(5): 423–438, 2010.

T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technischen Universitat Berlin, 2009.

T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization*, pages 449–481. Springer, 2013.

T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. Technical report, Technical Report 16-44, ZIB, Takustr. 7, 14195 Berlin, 2016.

N. AlGhanim, V. R. Comondore, J. Fleetham, C. A. Marra, and N. T. Ayas. The economic impact of obstructive sleep apnea. *Lung*, 186(1):7–12, 2008.

H. Allahyari and N. Lavesson. User-oriented Assessment of Classification Model Understandability. In *SCAI*, pages 11–19, 2011.

E. G. Altman, D. Hedeker, J. L. Peterson, and J. M. Davis. The Altman self-rating mania scale. *Biological psychiatry*, 42(10):948–955, 1997.

E. I. Altman et al. Predicting financial distress of companies: revisiting the Z-score and ZETA models. *Stern School of Business, New York University*, pages 9–12, 2000.

E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1): 237–260, 1998.

D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine Bias. *ProPublica*, 2016. URL `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

E. M. Antman, M. Cohen, P. J. Bernink, C. H. McCabe, T. Horacek, G. Papuchis, B. Mautner, R. Corbalan, D. Radley, and E. Braunwald. The TIMI risk score for unstable angina/non–ST elevation MI. *The Journal of the American Medical Association*, 284(7):835–842, 2000.

O. K. Asparouhov and P. A. Rubin. Oscillation Heuristics for the Two-group Classification Problem. *Journal of Classification*, 21:255–277, 2004.

O. K. Asparoukhov and A. Stam. Mathematical programming formulations for two-group classification with binary variables. *Annals of Operations Research*, 74:89–112, 1997.

D. S. Atkinson and P. M. Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

J. Austin, R. Ocker, and A. Bhati. Kentucky pretrial risk assessment instrument validation. *Bureau of Justice Statistics. Grant*, (2009-DB), 2010.

T. F. Babor, J. C. Higgins-Biddle, J. B. Saunders, M. G. Monteiro, W. H. Organization, et al. AUDIT: The alcohol use disorders identification test: Guidelines for use in primary health care. 2001.

K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

S. Baradaran. Race, Prediction, and Discretion. *Geo. Wash. L. Rev.*, 81:157–222, 2013.

R. Bardenet, O.-A. Maillard, et al. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

G. C. Barnes and J. M. Hyatt. Classifying Adult Probationers by Forecasting Future Offending. Technical report, National Institute of Justice, U.S. Department of Justice, 2012.

A. M. Barry-Jester, B. Casselman, and D. Goldstein. The New Science of Sentencing. *The Marshall Project, August*, 8, 2015.

H. Belfrage, R. Fransson, and S. Strand. Prediction of violence using the HCR-20: A prospective study in two maximum-security correctional institutions. *The Journal of Forensic Psychiatry*, 11(1):167–175, 2000.

P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, and D. Salvagnin. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications*, 65(3):545–566, 2016.

A. Ben-David. Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms. *Machine Learning*, 19(1):29–43, 1995.

S. Ben-David, N. Eiron, and P. M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, May 2003.

M. D. Beneish, C. M. Lee, and D. C. Nichols. Earnings manipulation and expected returns. *Financial Analysts Journal*, 2013.

R. Berk, L. Sherman, G. Barnes, E. Kurtz, and L. Ahlman. Forecasting murder within a population of probationers and parolees: a high stakes application of statistical learning. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 172(1):191–211, 2009.

R. A. Berk and J. Bleich. Statistical Procedures for Forecasting Criminal Behavior. *Criminology & Public Policy*, 12(3):513–544, 2013.

R. A. Berk, B. Kriegler, and J.-H. Baek. Forecasting dangerous inmate misconduct: An application of ensemble statistical procedures. *Journal of Quantitative Criminology*, 22(2):131–145, 2006.

D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *arXiv preprint arXiv:1507.03133*, 2015.

J. Bien and R. Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

J. Bien, J. Taylor, R. Tibshirani, et al. A lasso for hierarchical interactions. *The Annals of Statistics*, 41(3):1111–1141, 2013.

R. Bixby and E. Rothberg. Progress in computational mixed integer programming: a look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, 2007.

R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mixed integer programming: a progress report. *The Sharpest Cut*, pages 309–326, 2004.

P. Bobko, P. L. Roth, and M. A. Buster. The usefulness of unit weights in creating composite scores. A literature review, application to content validity, and meta-analysis. *Organizational Research Methods*, 10(4):689–709, 2007.

P. Bonami, M. Kilinç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*, pages 1–39. Springer, 2012.

R. Bone, R. Balk, F. Cerra, R. Dellinger, A. Fein, W. Knaus, R. Schein, W. Sibbald, J. Abrams, G. Bernard, et al. American College of Chest Physicians/Society of Critical Care Medicine Consensus Conference: Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis. *Critical Care Medicine*, 20(6):864–874, 1992.

H. G. Borden. Factors for predicting parole success. *Journal of the American Institute of Criminal Law and Criminology*, pages 328–336, 1928.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Advanced Lectures on Machine Learning*, pages 169–207. Springer, 2004.

L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001a.

L. Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001b. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`.

L. A. Breslow and D. W. Aha. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12(01):1–40, 1997.

J. P. Brooks. Support vector machines with the ramp loss and the hard margin loss. *Operations Research*, 59(2):467–479, 2011.

E. W. Burgess. Factors determining success or failure on parole. *The workings of the indeterminate sentence law and the parole system in Illinois*, pages 221–234, 1928.

S. D. Bushway. Is There Any Logic to Using Logit. *Criminology & Public Policy*, 12 (3):563–567, 2013.

S. D. Bushway and A. M. Piehl. The inextricable link between age and criminal history in sentencing. *Crime & Delinquency*, 53(1):156–183, 2007.

M. R. Bussieck and S. Vigerske. MINLP solver software. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

J. D. Camm, A. S. Raturi, and S. Tsubakitani. Cutting big M down to size. *Interfaces*, 20(5):61–66, 1990.

E. Carrizosa, A. Nogales-Gómez, and D. R. Morales. Strongly agree or strongly disagree?: Rating features in Support Vector Machines. *Information Sciences*, 329: 256–273, 2016.

R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM, 2004.

R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

R. D. Chervin and M. S. Aldrich. The Epworth Sleepiness Scale may not reflect objective measures of sleepiness or sleep apnea. *Neurology*, 52(1):125–125, 1999.

Y. Chevaleyre, F. Koriche, and J.-D. Zucker. Rounding methods for discrete linear classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 651–659, 2013.

L. P. e. a. Chung F, Yegneswaran B. STOP Questionnaire: a tool to screen patients for obstructive sleep apnea. *Anesthesiology*, 108:812–821, 2008.

N. A. Collop, W. M. Anderson, B. Boehlecke, D. Claman, R. Goldberg, D. J. Gottlieb, D. Hudgel, M. Sateia, and R. Schwab. Clinical guidelines for the use of unattended portable monitors in the diagnosis of obstructive sleep apnea in adult patients. Portable Monitoring Task Force of the American Academy of Sleep Medicine. *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine*, 3(7):737–747, 2007.

D. Combs, S. Shetty, and S. Parthasarathy. Big-Data or Slim-Data: Predictive Analytics Will Rule with World, 2016.

N. Cowan. The magical mystery four how is working memory capacity limited, and why? *Current directions in psychological science*, 19(1):51–57, 2010.

L. F. Cranor and B. A. LaMacchia. Spam! *Communications of the ACM*, 41(8): 74–83, 1998.

M. S. Crow. The complexities of prior record, race, ethnicity, and policy: Interactive effects in sentencing. *Criminal Justice Review*, 33(4):502–523, 2008.

R. M. Dawes. The robust beauty of improper linear models in decision making. *American psychologist*, 34(7):571–582, 1979.

R. M. Dawes, D. Faust, and P. E. Meehl. Clinical versus actuarial judgment. *Science*, 243(4899):1668–1674, 1989.

M. H. DeGroot and S. E. Fienberg. The comparison and evaluation of forecasters. *The statistician*, pages 12–22, 1983.

R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J.-J. Schmid, S. Sandhu, K. H. Guppy, S. Lee, and V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. 2017.

G. Duwe and K. Kim. Sacrificing Accuracy for Transparency in Recidivism Risk Assessment: The Impact of Classification Method on Predictive Performance. *Corrections*, pages 1–22, 2016.

C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM, 2012.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

H. J. Einhorn and R. M. Hogarth. Unit weighting schemes for decision making. *Organizational Behavior and Human Performance*, 13(2):171–192, 1975.

M. Elter, R. Schulz-Wendtland, and T. Wittenberg. The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. *Medical Physics*, 34:4164, 2007.

S. S. Erenguc and G. J. Koehler. Survey of mathematical programming models and experimental results for linear discriminant analysis. *Managerial and Decision Economics*, 11(4):215–225, 1990.

Ş. Ertekin and C. Rudin. A Bayesian Approach to Learning Scoring Systems. *Big Data*, 3(4):267–276, 2015.

Fair Credit Reporting Act. USC §1681.

Y.-J. Fan and W. A. Chaovalitwongse. Deterministic and probabilistic optimization models for data classification. In *Encyclopedia of Optimization*, pages 694–702. Springer, 2009.

S. V. Faraone, J. Biederman, and E. Mick. The age-dependent decline of attention deficit hyperactivity disorder: a meta-analysis of follow-up studies. *Psychological medicine*, 36(2):159–165, 2006.

J. Fayyad, N. A. Sampson, I. Hwang, T. Adamowski, S. Aguilar-Gaxiola, A. Al-Hamzawi, L. H. Andrade, G. Borges, G. de Girolamo, S. Florescu, et al. The descriptive epidemiology of DSM-IV Adult ADHD in the World Health Organization World Mental Health Surveys. *ADHD Attention Deficit and Hyperactivity Disorders*, 9(1):47–65, 2017.

A. Feelders and M. Pardoel. Pruning for monotone classification trees. In *Advances in intelligent data analysis V*, pages 1–12. Springer, 2003.

M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM, 2015.

A. Fiat and D. Pechyony. Decision trees: More theoretical justification for practical algorithms. *Lecture notes in computer science*, pages 156–170, 2004.

FICO. Xpress Optimization Suite 8.1. http://www.fico.com/en/products/fico-xpress-optimization-suite, 2017.

S. Finlay. *Credit scoring, response modeling, and insurance rating: a practical guide to forecasting consumer behavior.* Palgrave Macmillan, 2012.

J. F. Forrest and T. Ralphs. COIN Branch and Cut. https://projects.coin-or.org/Cbc, 2017.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10:2157–2192, 2009.

A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, Mar. 2014.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 32–40. ACM, 2005.

B. F. Gage, A. D. Waterman, W. Shannon, M. Boechler, M. W. Rich, and M. J. Radford. Validation of clinical classification schemes for predicting stroke. *The Journal of the American Medical Association*, 285(22):2864–2870, 2001.

G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. Progress in presolving for mixed integer programming. *Mathematical Programming Computation*, 7(4):367–398, 2015.

G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, et al. The SCIP Optimization Suite 3.2. *ZIB Report*, pages 15–60, 2016.

M. R. Gary and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness, 1979.

J. Glen. Integer programming methods for normalisation and variable selection in mathematical programming discriminant analysis models. *Journal of the Operational Research Society*, pages 1043–1053, 1999.

S. Goel, J. M. Rao, and R. Shroff. Precinct or Prejudice? Understanding Racial Disparities in New York City's Stop-and-Frisk Policy. *Annals of Applied Statistics*, 10(1):365–394, 2016.

S. T. Goh and C. Rudin. Box drawings for learning with imbalanced data. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 333–342. ACM, 2014.

N. Goldberg and J. Eckstein. Sparse weighted voting classifier selection and its linear programming relaxations. *Information Processing Letters*, 112:481–486, 2012.

B. Goodman and S. Flaxman. EU regulations on algorithmic decision-making and a " right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

D. M. Gottfredson and H. N. Snyder. The Mathematics of Risk Classification: Changing Data into Valid Instruments for Juvenile Courts. NCJ 209158. *Office of Juvenile Justice and Delinquency Prevention Washington, D.C.*, 2005.

D. J. Gottlieb, C. W. Whitney, W. H. Bonekat, C. Iber, G. D. James, M. Lebowitz, F. J. Nieto, and C. E. Rosenberg. Relation of sleepiness to respiratory disturbance index: the Sleep Heart Health Study. *American journal of respiratory and critical care medicine*, 159(2):502–507, 1999.

W. M. Grove and P. E. Meehl. Comparative efficiency of informal (subjective, impressionistic) and formal (mechanical, algorithmic) prediction procedures: The clinical–statistical controversy. *Psychology, Public Policy, and Law*, 2(2):293–323, 1996.

W. Guan, A. Gray, and S. Leyffer. Mixed-Integer Support Vector Machine. In *NIPS Workshop on Optimization for Machine Learning*, 2009.

M. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. Van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47, 2016.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

S. J. Haberman. Generalized residuals for log-linear models. In *Proceedings of the 9th international biometrics conference, Boston*, pages 104–122, 1976.

K. Hannah-Moffat. Actuarial sentencing: An "unsettled" proposition. *Justice Quarterly*, 30(2):270–296, 2013.

R. Hanson and D. Thornton. Notes on the development of Static-2002. *Ottawa, Ontario: Department of the Solicitor General of Canada*, 2003.

B. E. Harcourt. *Against prediction: Profiling, policing, and punishing in an actuarial age.* University of Chicago Press, 2008.

J. Håstad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7(3):484–492, 1994.

T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley. Least angle and LÂ§1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

L. Hirsch, S. LaRoche, N. Gaspard, E. Gerard, A. Svoronos, S. Herman, R. Mani, H. Arif, N. Jette, Y. Minazad, et al. American Clinical Neurophysiology Society's standardized critical care EEG terminology: 2012 version. *Journal of Clinical Neurophysiology*, 30(1):1–27, 2013.

P. B. Hoffman. Twenty years of operational use of a risk prediction instrument: The United States Parole Commission's Salient Factor Score. *Journal of Criminal Justice*, 22(6):477–494, 1994.

R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993.

R. C. Holte. Elaboration on Two Points Raised in "Classifier Technology and the Illusion of Progress". *Statistical Science*, 21(1):24–26, Feb. 2006.

P. Howard, B. Francis, K. Soothill, and L. Humphreys. OGRS 3: The revised offender group reconviction scale. Technical report, Ministry of Justice London, UK, 2009.

J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

I. ILOG. CPLEX Optimizer 12.6. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/, 2017.

R. Jenatton, J.-Y. Audibert, and F. Bach. Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

D. Jennings, T. M. Amabile, and L. Ross. Informal covariation assessment: Data-based vs. theory-based judgments. In D. Kahneman, P. Slovic, and A. Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

E. A. Joachimsthaler and A. Stam. Mathematical programming approaches for the classification problem in two-group discriminant analysis. *Multivariate Behavioral Research*, 25(4):427–454, 1990.

J. Jung, C. Concannon, R. Shroff, S. Goel, and D. G. Goldstein. Simple rules for complex decisions. 2017.

T. Kamishima, S. Akaho, and J. Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650. IEEE, 2011.

V. K. Kapur. Obstructive sleep apnea: diagnosis, epidemiology, and economics. *Respiratory care*, 55(9):1155–1167, 2010.

J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

R. Kessler, M. Lane, P. Stang, and D. Van Brunt. The prevalence and workplace costs of adult attention deficit hyperactivity disorder in a large manufacturing firm. *Psychological medicine*, 39(1):137–147, 2009.

R. C. Kessler, L. Adler, M. Ames, R. A. Barkley, H. Birnbaum, P. Greenberg, J. A. Johnston, T. Spencer, and T. B. Üstün. The prevalence and effects of adult attention deficit/hyperactivity disorder on work performance in a nationally representative sample of workers. *Journal of Occupational and Environmental Medicine*, 47 (6):565–572, 2005a.

R. C. Kessler, L. Adler, M. Ames, O. Demler, S. Faraone, E. Hiripi, M. J. Howes, R. Jin, K. Secnik, T. Spencer, et al. The World Health Organization Adult ADHD Self-Report Scale (ASRS): a short screening scale for use in the general population. *Psychological medicine*, 35(02):245–256, 2005b.

R. C. Kessler, J. G. Green, L. A. Adler, R. A. Barkley, S. Chatterji, S. V. Faraone, M. Finkelman, L. L. Greenhill, M. J. Gruber, M. Jewell, et al. Structure and diagnosis of adult attention-deficit/hyperactivity disorder: analysis of expanded symptom criteria from the Adult ADHD Clinical Diagnostic Scale. *Archives of general psychiatry*, 67(11):1168–1178, 2010.

B. Kim, J. A. Shah, and F. Doshi-Velez. Mind the gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*, pages 2260–2268, 2015.

M.-J. Kim and I. Han. The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms. *Expert Systems with Applications*, 25 (4):637–646, 2003.

E. Klotz and A. M. Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1): 18–32, 2013.

W. A. Knaus, J. E. Zimmerman, D. P. Wagner, E. A. Draper, and D. E. Lawrence. APACHE-acute physiology and chronic health evaluation: a physiologically based classification system. *Critical Care Medicine*, 9(8):591–597, 1981.

W. A. Knaus, E. A. Draper, D. P. Wagner, and J. E. Zimmerman. APACHE II: a severity of disease classification system. *Critical Care Medicine*, 13(10):818–829, 1985.

W. A. Knaus, D. Wagner, E. Draper, J. Zimmerman, M. Bergner, P. Bastos, C. Sirio, D. Murphy, T. Lotring, and A. Damiano. The APACHE III prognostic system. Risk prediction of hospital mortality for critically ill hospitalized adults. *Chest Journal*, 100(6):1619–1636, 1991.

Y. Kodratoff. The comprehensibility manifesto. *KDD Nugget Newsletter*, 94(9), 1994.

R. Kohavi. The power of decision tables. In *Machine Learning: ECML-95*, pages 174–189. Springer, 1995.

R. Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *KDD*, pages 202–207, 1996.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

R. Kohavi and D. Sommerfield. Targeting Business Users with Decision Table Classifiers. In *KDD*, pages 249–253, 1998.

W. Kotlowski, K. J. Dembczynski, and E. Huellermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1113–1120, 2011.

M. Kuhn, S. Weston, and N. C. C. code for C5.0 by R. Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*, 2012. R package version 0.1.0-013.

T. Küpper, J. Haavik, H. Drexler, J. A. Ramos-Quiroga, D. Wermelskirchen, C. Prutz, and B. Schauble. The negative impact of attention-deficit/hyperactivity disorder on occupational health in adults and adolescents. *International archives of occupational and environmental health*, 85(8):837–847, 2012.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1675–1684. ACM, 2016.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

P. A. Langan and D. J. Levin. Recidivism of prisoners released in 1994. *Federal Sentencing Reporter*, 15(1):58–65, 2002.

E. Latessa, P. Smith, R. Lemke, M. Makarios, and C. Lowenkamp. Creation and validation of the Ohio risk assessment system: Final report. *Center for Criminal Justice Research, School of Criminal Justice, University of Cincinnati, Cincinnati, OH. Retrieved from http://www. ocjs. ohio. gov/ORAS_FinalReport. pd f*, 2009.

J.-R. Le Gall, S. Lemeshow, and F. Saulnier. A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study. *The Journal of the American Medical Association*, 270(24):2957–2963, 1993.

E. K. Lee and T.-L. Wu. Classification and disease prediction via mathematical programming. In *Handbook of Optimization in Medicine*, pages 1–50. Springer, 2009.

B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. An Interpretable Stroke Prediction Model using Rules and Bayesian Analysis. In *Proceedings of AAAI Late Breaking Track*, 2013.

B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, 2015.

A. Y. Levin. On an algorithm for the minimization of convex functions. In *Soviet Mathematics Doklady*, volume 160, pages 1244–1247, 1965.

A. Liaw and M. Wiener. Classification and Regression by randomForest. *R News*, 2 (3):18–22, 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

M. Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3):203–228, 2000.

W. Lim, M. Van der Eerden, R. Laing, W. Boersma, N. Karalus, G. Town, S. Lewis, and J. Macfarlane. Defining community acquired pneumonia severity on presentation to hospital: an international derivation and validation study. *Thorax*, 58(5): 377–382, 2003.

D. Lin, E. Pitler, D. P. Foster, and L. H. Ungar. In defense of l0. In *Workshop on Feature Selection,(ICML 2008)*, 2008.

Y. Liu and Y. Wu. Variable selection via a combination of the L0 and L1 penalties. *Journal of Computational and Graphical Statistics*, 16(4), 2007.

A. S. London and S. D. Landes. Attention deficit hyperactivity disorder and adult mortality. *Preventive medicine*, 90:8–10, 2016.

P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. *Machine learning*, 78(3):287–304, 2010.

P. M. Long and R. A. Servedio. On the Weight of Halfspaces over Hamming Balls. *SIAM Journal on Discrete Mathematics*, 28(3):1035–1061, 2014.

C. T. Lowenkamp and E. J. Latessa. Understanding the risk principle: How and why correctional interventions can harm low-risk offenders. *Topics in community corrections*, 2004:3–8, 2004.

D. Malioutov and K. Varshney. Exact rule learning via boolean compressed sensing. In *Proceedings of The 30th International Conference on Machine Learning*, pages 765–773, 2013.

O. L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5(4):309–323, 1994.

O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer, 2010.

J. Marklof. Fine-scale statistics for the multidimensional Farey sequence. *ArXiv e-prints*, July 2012.

D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European journal of operational research*, 183(3):1466–1476, 2007.

D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011.

L. Maxfield, M. Harer, T. Drisko, C. Kitchens, S. Meacham, and M. Iaconetti. A Comparison of the Federal Sentencing Guidelines Criminal History Category and the US Parole Commission Salient Factor Score. *Research Series on the Recidivism of Federal Guideline Offenders. Washington, DC: United States Sentencing Commission*, 2005.

A. McGinley and R. M. Pearse. A national early warning score for acutely ill patients, 2012.

N. Meinshausen et al. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010.

A. K. Menon, X. J. Jiang, S. Vembu, C. Elkan, and L. Ohno-Machado. Predicting accurate probabilities with a ranking loss. In *Machine learning: proceedings of the International Conference. International Conference on Machine Learning*, volume 2012, page 703. NIH Public Access, 2012.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. R package version 1.6-1.

N. Minkoff. ADHD in managed care: an assessment of the burden of illness and proposed initiatives to improve outcomes. *The American journal of managed care*, 15(5 Suppl):S151–9, 2009.

R. P. Moreno, P. G. Metnitz, E. Almeida, B. Jordan, P. Bauer, R. A. Campos, G. Iapichino, D. Edbrooke, M. Capuzzo, and J.-R. Le Gall. SAPS 3 - From evaluation of the patient to evaluation of the intensive care unit. Part 2: Development of a prognostic model for hospital mortality at ICU admission. *Intensive Care Medicine*, 31(10):1345–1355, 2005.

S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

S. Mukherjee. A.I. vs M.D. What happens when diagnosis is automated?, April 2017. URL `http://www.newyorker.com/magazine/2017/04/03/ai-versus-md`.

S. Muroga. Threshold logic and its applications. 1971.

J. Neumann, C. Schnörr, and G. Steidl. Combined SVM-based feature selection and classification. *Machine learning*, 61(1):129–150, 2005.

H. T. Nguyen and K. Franke. A general lp-norm support vector machine via mixed 0-1 programming. In *Machine Learning and Data Mining in Pattern Recognition*, pages 40–49. Springer, 2012.

T. Nguyen and S. Sanner. Algorithms for Direct 0–1 Loss Optimization in Binary Classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1085–1093, 2013.

J. Park and S. Boyd. A Semidefinite Programming Method for Integer Convex Quadratic Minimization. *arXiv preprint arXiv:1504.07672*, 2015.

J.-F. Payen, O. Bru, J.-L. Bosson, A. Lagrasta, E. Novel, I. Deschaux, P. Lavagne, and C. Jacquot. Assessing pain in critically ill sedated patients by using a behavioral pain scale. *Critical care medicine*, 29(12):2258–2263, 2001.

M. Pazzani, S. Mani, and W. Shankle. Acceptance of rules generated by machine learning among medical experts. *Methods of information in medicine*, 40(5):380–385, 2001.

M. J. Pazzani. Knowledge discovery from data? *Intelligent systems and their applications, IEEE*, 15(2):10–12, 2000.

Pennsylvania Bulletin. Sentence Risk Assessment Instrument, April 2017.

Pennsylvania Code. Adoption of Risk Assessment Instrument, 2010.

Pennsylvania Commission on Sentencing. Interim Report 4: Development of Risk Assessment Scale. Technical report, June 2012.

K. B. Pétursson. Discovering Branching Rules for Mixed Integer Programming by Computational Intelligence.

D. Pew Center of the States, Public Safety Performance Project Washington. Risk/needs assessment 101: science reveals new tools to manage offenders. The Pew Center of the States, 2011.

J. D. Piotroski. Value investing: The use of historical financial statement information to separate winners from losers. *Journal of Accounting Research*, pages 1–41, 2000.

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. R. Quinlan. Simplifying decision trees. *International Journal of Human-Computer Studies*, 51(2):497–510, 1999.

J. R. Quinlan. *C4.5: programs for machine learning.* Elsevier, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2014.

T. Ralphs, M. Güzelsoy, and A. Mahajan. SYMPHONY 5.6. https://projects.coin-or.org/SYMPHONY, 2017.

M. D. Reid and R. C. Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 11:2387–2422, 2010.

B. M. Reilly and A. T. Evans. Translating Clinical Research into Clinical Practice: Impact of Using Prediction Rules To Make Decisions. *Annals of internal medicine*, 144(3):201–209, 2006.

V. F. Reyna and C. J. Brainerd. The importance of mathematics in health and human judgment: Numeracy, risk communication, and medical decision making. *Learning and Individual Differences*, 17(2):147–159, 2007.

G. Ridgeway. *gbm: Generalized boosted regression models*, 2006. URL `https://cran.r-project.org/web/packages/gbm/gbm.pdf`.

N. Ritter. Predicting recidivism risk: New tool in Philadelphia shows great promise. *NIJ Journal*, 271:4–13, 2013.

R. L. Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

P. A. Rubin. Heuristic Solution Procedures for a Mixed-Integer Programming Discriminant Model. *Managerial and Decision Economics*, 11:255–266, 1990.

P. A. Rubin. Solving Mixed Integer Classification Problems by Decomposition. *Annals of Operations Research*, 74:51–64, 1997.

P. A. Rubin. Mixed integer classification problems. In *Encyclopedia of Optimization*, pages 2210–2214. Springer, 2009.

S. Rüping. *Learning interpretable models.* PhD thesis, Universität Dortmund, 2006.

T. Sato, Y. Takano, and R. Miyashiro. Piecewise-Linear Approximation for Feature Subset Selection in a Sequential Logit Model. *arXiv preprint arXiv:1510.05417*, 2015.

T. Sato, Y. Takano, R. Miyashiro, and A. Yoshise. Feature subset selection for logistic regression via mixed integer optimization. *Computational Optimization and Applications*, pages 1–16, 2016.

J. C. Schlimmer. Concept acquisition through representational adjustment. 1987.

M. Schmitt. A slightly improved upper bound on the size of weights sufficient to represent any linearly separable Boolean function. 2012.

R. A. Servedio. Every linear threshold function has a low-weight approximator. *computational complexity*, 16(2):180–209, 2007.

M. M. Shafi, M. B. Westover, A. J. Cole, R. D. Kilbride, D. B. Hoch, and S. S. Cash. Absence of early epileptiform abnormalities predicts lack of seizures on continuous EEG. *Neurology*, 79(17):1796–1801, 2012.

P. Shaw, K. Ahn, and J. L. Rapoport. Good News for Screening for Adult Attention-Deficit/Hyperactivity Disorder. *Jama psychiatry*, 74(5):527–527, 2017.

A. Six, B. Backus, and J. Kelder. Chest pain in the emergency room: value of the HEART score. *Netherlands Heart Journal*, 16(6):191–196, 2008.

R. P. Skomro and M. H. Kryger. Clinical presentations of obstructive sleep apnea syndrome. *Progress in cardiovascular diseases*, 41(5):331–340, 1999.

R. C. Soltysik and P. R. Yarnold. The Warmack-Gonzalez algorithm for linear two-category multivariable optimal discriminant analysis. *Computers & operations research*, 21(7):735–745, 1994.

E. Sommer. *Theory Restructuring: A Perspective on Design and Maintenance of Knowledge Based Systems.* PhD thesis, Universität Dortmund, 1996.

P. Somol, P. Pudil, and J. Kittler. Fast branch &amp; bound algorithms for optimal feature selection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(7):900–912, 2004.

S. B. Starr. Sentencing, By the Numbers. *New York Times*, 2014.

A. F. Struck, B. Ustun, A. Rodriguez Ruiz, J. W. Lee, S. LaRoche, L. J. Hirsch, E. J. Gilmore, C. Rudin, and B. M. Westover. A Practical Risk Score for EEG Seizures in Hospitalized Patients. *Forthcoming in JAMA Neurology*, 2017.

G. H. Subramanian, J. Nosek, S. P. Raghunathan, and S. S. Kanitkar. A comparison of the decision table and tree. *Communications of the ACM*, 35(1):89–94, 1992.

C. B. Surman, P. G. Hammerness, K. Pion, and S. V. Faraone. Do stimulants improve functioning in adults with ADHD?: A review of the literature. *European Neuropsychopharmacology*, 23(6):528–533, 2013.

J. Tashea. Courts are using AI to sentence criminals. That must stop now. *Wired*, 2017. URL `https://www.wired.com/2017/04/courts-using-ai-sentence-criminals-must-stop-now`.

A. Taylor, S. Deb, and G. Unwin. Scales for the identification of adults with attention deficit hyperactivity disorder (ADHD): a systematic review. *Research in Developmental Disabilities*, 32(3):924–938, 2011.

C. H. Teo, S. Vishwanathan, A. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 1:55, 2009.

M. Than, D. Flaws, S. Sanders, J. Doust, P. Glasziou, J. Kline, S. Aldous, R. Troughton, C. Reid, W. A. Parsonage, et al. Development and validation of the Emergency Department Assessment of Chest pain Score and 2 h accelerated diagnostic protocol. *Emergency Medicine Australasia*, 26(1):34–44, 2014.

T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning*, 2012. URL `http://CRAN.R-project.org/package=rpart`. R package version 4.1-0.

P. Thodoroff, J. Pineau, and A. Lim. Learning Robust Features using Deep Learning for Automatic Seizure Detection. *arXiv preprint arXiv:1608.00220*, 2016.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

N. Tollenaar and P. van der Heijden. Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 176(2): 565–584, 2013.

S. Turner, J. Hess, and J. Jannetta. Development of the California Static Risk Assessment Instrument (CSRA). University of California, Irvine, Center for Evidence-Based Corrections, 2009.

U.S. Department of Justice. The Mathematics of Risk Classification: Changing Data into Valid Instruments for Juvenile Courts. July 2005.

U.S. Department of Justice, Bureau of Justice Statistics. Recidivism of Prisoners Released in 1994. http://doi.org/10.3886/ICPSR03355.v8, 2014.

U.S. Sentencing Commission. 2012 GUIDELINES MANUAL: CHAPTER FOUR - CRIMINAL HISTORY AND CRIMINAL LIVELIHOOD, November 1987. URL `http://www.ussc.gov/guidelines-manual/2012/2012-4a11`.

B. Ustun and C. Rudin. Methods and Models for Interpretable Linear Classification. *Technical Report*, 2014.

B. Ustun and C. Rudin. Learning Optimized Risk Scores for Large-Scale Datasets. *arXiv:1610.00168*, 2016a.

B. Ustun and C. Rudin. Supersparse Linear Integer Models for Optimized Medical Scoring Systems. *Machine Learning*, 102(3):349–391, 2016b.

B. Ustun and C. Rudin. Optimized Risk Scores. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.

B. Ustun, S. Traca, and C. Rudin. Supersparse Linear Integer Models for Predictive Scoring Systems. In *AAAI Late-Breaking Developments*, 2013.

B. Ustun, M. Westover, C. Rudin, and M. T. Bianchi. Clinical Prediction Models for Sleep Apnea: The Importance of Medical History over Symptoms. *Journal of Clinical Sleep Medicine*, 12(2):161–168, 2016.

B. Ustun, L. A. Adler, C. Rudin, S. V. Faraone, T. J. Spencer, P. Berglund, M. J. Gruber, and R. C. Kessler. The World Health Organization Adult Attention-Deficit / Hyperactivity Disorder Self-Report Screening Scale for DSM-5. *JAMA Psychiatry*, 74(5):520–526, 2017.

P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2):161–186, 1989.

A. Van Assche and H. Blockeel. Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In *Machine Learning: ECML 2007*, pages 418–429. Springer, 2007.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

W. Verbeke, D. Martens, C. Mues, and B. Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354–2364, 2011.

H. Wainer. Estimating coefficients in linear models: It don't make no nevermind. *Psychological Bulletin*, 83(2):213, 1976.

T. Wang, C. Rudin, F. Doshi, Y. Liu, E. Klampfl, and P. MacNeille. Bayesian Or's of And's for interpretable classification with application to context aware recommender systems, 2015.

F. W. Weathers, B. T. Litz, T. M. Keane, P. A. Palmieri, B. P. Marx, and P. P. Schnurr. The PTSD Checklist for DSM-5 (PCL-5). *Scale available from the National Center for PTSD at www.ptsd.va.gov.*, 2013.

L. A. Wolsey. *Integer Programming*, volume 42. Wiley New York, 1998.

J. J. Wroblewski. Annual Letter, U.S. Department of Justice: Criminal Division, July 2014.

Y. Wu and Y. Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.

N. Yanev and S. Balev. A combinatorial approach to the classification problem. *European Journal of Operational Research*, 115(2):339–350, 1999.

M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: A mechanism for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

J. Zeng, B. Ustun, and C. Rudin. Interpretable Classification Models for Recidivism Prediction. *Journal of the Royal Statistical Society: Series A*, 2016.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Supporting Material for Chapter 3

A.1 Omitted Proofs

Proof (Theorem 3.24). *Let* $V(\boldsymbol{\lambda}) = l_{01}(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$. *Note that* $\boldsymbol{\lambda} = \mathbf{0}$ *is a feasible solution since we assume that* $\mathbf{0} \in \mathcal{L}$. *Since* $\boldsymbol{\lambda} = 0$ *achieves an objective value of* $V(\mathbf{0}) \leq 1$, *any optimal solution,* $\boldsymbol{\lambda} \in \operatorname{argmin}_{\boldsymbol{\lambda} \in \mathcal{L}} V(\boldsymbol{\lambda})$, *must attain an objective value* $V(\boldsymbol{\lambda}) \leq 1$. *This implies*

$$\begin{aligned} V(\boldsymbol{\lambda}) &\leq 1 \\ \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i \neq \boldsymbol{\lambda}^\top \boldsymbol{x}_i \leq 0\right] + C_0 \|\boldsymbol{\lambda}\|_0 &\leq 1, \\ C_0 \|\boldsymbol{\lambda}\|_0 &\leq 1, \\ \|\boldsymbol{\lambda}\|_0 &\leq \left\lfloor \frac{1}{C_0} \right\rfloor \end{aligned}$$

The last line uses that $\|\boldsymbol{\lambda}\|_0$ *is an integer.* ■

Proof (Theorem 3.22). *Let us denote the set of classifiers whose objective value is less or equal to* $\tilde{V}(\tilde{f}; \mathcal{D}_n)$ *as*

$$\tilde{\mathcal{F}}^{\varepsilon} = \left\{ f \in \tilde{\mathcal{F}} \;\middle|\; \tilde{V}(f; \mathcal{D}_n) \leq \tilde{V}(\tilde{f}; \mathcal{D}_n) + \varepsilon \right\},$$

and the set of examples that have been removed by the data reduction algorithm

$$\mathcal{S} = \mathcal{D}_n \setminus \mathcal{D}_m.$$

By definition, data reduction only removes an example if its sign is fixed. This means that $\operatorname{sign}(f(\boldsymbol{x}_i)) = \operatorname{sign}\left(\tilde{f}(\boldsymbol{x}_i)\right)$ *for all* $i \in \mathcal{S}$ *and* $f \in \tilde{\mathcal{F}}^{\varepsilon}$. *Thus, we can see that for all classifiers* $f \in \tilde{\mathcal{F}}^{\varepsilon}$,

$$\begin{aligned} V(f; \mathcal{D}_n) &= V(f; \mathcal{D}_m) + \sum_{i \in \mathcal{S}} \mathbb{1}\left[y_i f(\boldsymbol{x}_i) \leq 0\right] && \text{(A.1)} \\ &= V(f; \mathcal{D}_m) + \sum_{i \in \mathcal{S}} \mathbb{1}\left[y_i \tilde{f}(\boldsymbol{x}_i) \leq 0\right] && \text{(A.2)} \\ &= V(f; \mathcal{D}_m) + \textit{constant}. && \text{(A.3)} \end{aligned}$$

We now proceed to prove the statement in (3.14). *For the case when* $\mathcal{S} = \emptyset$, *then* $\mathcal{D}_n = \mathcal{D}_m$, *and* (3.14) *follows trivially. For the case when* $\mathcal{S} \neq \emptyset$, *we note that*

$$\begin{aligned} \mathcal{F}^* = \operatorname*{argmin}_{f \in \mathcal{F}} V(f; \mathcal{D}_n) &= \operatorname*{argmin}_{f \in \mathcal{F}} V(f; \mathcal{D}_m \cup \mathcal{S}), \\ &= \operatorname*{argmin}_{f \in \mathcal{F}} V(f; \mathcal{D}_m) + V(f; \mathcal{S}), \\ &= \operatorname*{argmin}_{f \in \mathcal{F}} V(f; \mathcal{D}_m) + \text{constant}, && \text{(A.4)} \\ &= \operatorname*{argmin}_{f \in \mathcal{F}} V(f; \mathcal{D}_m). \end{aligned}$$

Here, the statement in (A.4) *follows directly from* (A.1). ■

Proof (Theorem 3.23). *We assume that we have found a surrogate function,* s, *that satisfies conditions I–IV and choose* $C_\mathrm{s} > 2\varepsilon$.

Our proof uses the following result: if $\|\boldsymbol{\lambda}_{01}^* - \boldsymbol{\lambda}_\mathrm{s}^*\| > C_{\boldsymbol{\lambda}}$ *then* $\boldsymbol{\lambda}_{01}^*$ *cannot be a minimizer of* $l_{01}(\boldsymbol{\lambda})$. *To see this, consider condition III for the case where* $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{01}^*$. *In this case, condition III states:*

$$\|\boldsymbol{\lambda}_{01}^* - \boldsymbol{\lambda}_\mathrm{s}^*\| > C_{\boldsymbol{\lambda}} \implies l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) - l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) > C_\mathrm{s}.$$

Thus,

$$\begin{aligned}
l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + C_\mathrm{s} &< l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) \\
l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + C_\mathrm{s} &< l_{01}(\boldsymbol{\lambda}_{01}^*) + \varepsilon && \text{(A.5)} \\
l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + C_\mathrm{s} - \varepsilon &< l_{01}(\boldsymbol{\lambda}_{01}^*) \\
l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + C_\mathrm{s} - \varepsilon &< l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) && \text{(A.6)} \\
l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + \varepsilon &< l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*). && \text{(A.7)}
\end{aligned}$$

Here the inequality in (A.5) *follows from condition IV, the inequality in* (A.6) *follows from condition I, and the inequality in* (A.7) *follows from our choice that* $C_\mathrm{s} > 2\varepsilon$.

We proceed by looking at the LHS and RHS of (A.7) *separately. Using condition I on the LHS of* (A.7) *we get that:*

$$l_{01}(\boldsymbol{\lambda}_\mathrm{s}^*) + \varepsilon \le l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) + \varepsilon. \tag{A.8}$$

Using condition IV on the RHS of (A.7) *we get that:*

$$l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) \le l_{01}(\boldsymbol{\lambda}_{01}^*) + \varepsilon. \tag{A.9}$$

Combining the inequalities in (A.7), (A.8) *and* (A.9), *we get that:*

$$l_{01}(\boldsymbol{\lambda}_\mathrm{s}^*) < l_{01}(\boldsymbol{\lambda}_{01}^*). \tag{A.10}$$

The statement in (A.10) *is a contradiction of the definition of* $\boldsymbol{\lambda}_{01}^*$. *Thus, we know that our assumption was incorrect and thus* $\|\boldsymbol{\lambda}_{01}^* - \boldsymbol{\lambda}_\mathrm{s}^*\| \le C_{\boldsymbol{\lambda}}$. *We plug this into the Lipschitz condition II as follows:*

$$\begin{aligned}
l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) - l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*) \le L\|\boldsymbol{\lambda}_{01}^* - \boldsymbol{\lambda}_\mathrm{s}^*\| &< LC_{\boldsymbol{\lambda}}, \\
l_\mathrm{s}(\boldsymbol{\lambda}_{01}^*) &< LC_{\boldsymbol{\lambda}} + l_\mathrm{s}(\boldsymbol{\lambda}_\mathrm{s}^*).
\end{aligned}$$

Thus, we have satisfied the level set condition with $\varepsilon = LC_{\boldsymbol{\lambda}}$. ■

Proof (Theorem 3.27). *We will prove the statement of the theorem for normalized versions of the vectors, $\boldsymbol{\rho}/\|\boldsymbol{\rho}\|_2$ and $\boldsymbol{\lambda}/\Lambda$. This is without loss of generality because the 0–1 loss is scale invariant:*

$$l_{01}(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i \boldsymbol{\lambda}^\top \boldsymbol{x}_i \le 0\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i \frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda} \le 0\right],$$
$$l_{01}(\boldsymbol{\rho}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i \boldsymbol{\rho}^\top \boldsymbol{x}_i \le 0\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} \le 0\right].$$

We set $\Lambda > \frac{X_{\max}\sqrt{d}}{2\gamma_{\min}}$ as in (3.16). *Given Λ, we then set λ_j/Λ as $\rho_j/\|\boldsymbol{\rho}\|_2$ rounded to the nearest $1/\Lambda$ for $j = 1, \ldots, d$. In order to prove the statement of the theorem, we need to show that these choices of Λ and $\boldsymbol{\lambda}$ ensure that $\boldsymbol{\rho}/\|\boldsymbol{\rho}\|_2$ and $\boldsymbol{\lambda}/\Lambda$ classify each point in the same way.*

We will first show that Λ and $\boldsymbol{\lambda}$ guarantees that the difference between the margin of $\boldsymbol{\rho}/\|\boldsymbol{\rho}\|_2$ and the margin of $\boldsymbol{\lambda}/\Lambda$ on all training examples is always less than the minimum margin of $\boldsymbol{\rho}/\|\boldsymbol{\rho}\|_2$, defined as $\gamma_{\min} = \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}$. To see this, note that for all $i \in I$:

$$\begin{aligned}
\left|\frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda} - \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2}\right| &\le \left\|\frac{\boldsymbol{\lambda}}{\Lambda} - \frac{\boldsymbol{\rho}}{\|\boldsymbol{\rho}\|_2}\right\|_2 \|\boldsymbol{x}_i\|_2 && \text{(A.11)}\\
&= \left(\sum_{j=1}^{d}\left|\frac{\lambda_j}{\Lambda} - \frac{\rho_j}{\|\boldsymbol{\rho}\|_2}\right|^2\right)^{1/2} \|\boldsymbol{x}_i\|_2 \\
&\le \left(\sum_{j=1}^{d}\frac{1}{(2\Lambda)^2}\right)^{1/2} \|\boldsymbol{x}_i\|_2 && \text{(A.12)}\\
&= \frac{\sqrt{d}}{2\Lambda} X_{\max} \\
&< \frac{\sqrt{d}X_{\max}}{2\left(\frac{X_{\max}\sqrt{d}}{2\min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}}\right)} && \text{(A.13)}\\
&= \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}. && \text{(A.14)}
\end{aligned}$$

Here: the inequality in (A.11) *uses the Cauchy-Schwarz inequality; the inequality in* (A.12) *is due to the fact that the distance between $\rho_j/\|\boldsymbol{\rho}\|_2$ and λ_j/Λ is at most $1/2\Lambda$; and the inequality in* (A.13) *is due to our choice of Λ.*

We can now show that $\boldsymbol{\rho}/\|\boldsymbol{\rho}\|_2$ and $\boldsymbol{\lambda}/\Lambda$ classify each point in the same way. We consider three cases: first, the case where $\boldsymbol{x}_i$ lies on the margin; second, the case where $\boldsymbol{\rho}$ has a positive margin on $\boldsymbol{x}_i$; and third, the case where $\boldsymbol{\rho}$ has a negative *margin on* $\boldsymbol{x}_i$.

For the case when $\boldsymbol{x}_i$ *lies on the margin,* $\min_i |\boldsymbol{\rho}^\top \boldsymbol{x}_i| = 0$ *and the theorem holds trivially.*

For the case where $\boldsymbol{\rho}$ *has positive margin,* $\boldsymbol{\rho}^\top \boldsymbol{x}_i > 0$*, the following calculation using* (A.14) *is relevant:*

$$\frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} - \frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda} \le \left| \frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda} - \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} \right| < \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}.$$

We will use the fact that for any $i' \in I$*, by definition of the minimum:*

$$0 \le \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_{i'}|}{\|\boldsymbol{\rho}\|_2} - \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2},$$

and combine this with a rearrangement of the previous expression to obtain:

$$0 \le \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} - \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} = \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} - \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} < \frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda}.$$

Thus, we have shown that $\boldsymbol{\lambda}^\top \boldsymbol{x}_i > 0$ *whenever* $\boldsymbol{\rho}^\top \boldsymbol{x}_i > 0$.

For the case where $\boldsymbol{\rho}$ *has a negative margin on* $\boldsymbol{x}_i$, $\boldsymbol{\rho}^\top \boldsymbol{x}_i < 0$*, we perform an analogous calculation:*

$$\frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\|\boldsymbol{\lambda}\|_2} - \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} \le \left| \frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda} - \frac{\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} \right| < \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2}.$$

and then using that $\boldsymbol{\rho}^\top \boldsymbol{x}_i < 0$,

$$0 \le \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} - \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} = \frac{-\boldsymbol{\rho}^\top \boldsymbol{x}_i}{\|\boldsymbol{\rho}\|_2} - \min_i \frac{|\boldsymbol{\rho}^\top \boldsymbol{x}_i|}{\|\boldsymbol{\rho}\|_2} < -\frac{\boldsymbol{\lambda}^\top \boldsymbol{x}_i}{\Lambda}.$$

Thus, we have shown $\boldsymbol{\lambda}^\top \boldsymbol{x}_i < 0$ *whenever* $\boldsymbol{\rho}^\top \boldsymbol{x}_i < 0$. *Combining the results from the cases where the margin is positive and the margin is negative, we get*

$$\mathbb{1}\left[y_i \boldsymbol{\rho}^\top \boldsymbol{x}_i \le 0\right] = \mathbb{1}\left[y_i \boldsymbol{\lambda}^\top \boldsymbol{x}_i \le 0\right] \text{ for all } i \in I.$$

Summing over i *and dividing both sides by* n *yields the statement of the theorem.* ■

Proof (Corollary 3.28). *The proof follows by applying Theorem 3.27 to the reduced dataset* $\mathcal{D}_n \backslash I_{(k)}$. ■

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Supporting Material for Chapter 6

B.1 Small Trade-off Parameters do not Influence Accuracy

In Section 6.1, we state that if the trade-off parameter C_0 in the objective of the risk score problem is sufficiently small, then its optimal solution will attain the best possible trade-off between logistic loss and sparsity. Here, we formalize this statement. In what follows, we will omit the intercept term for clarity and explicitly show the regularization parameter C_0 in the RISKSLIM objective so that $V(\boldsymbol{\lambda}; C_0) := l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$. We also make use of some new notation shown in Table B.1.

Notation	Description
$\mathcal{M} = \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda})$	minimizers of the logistic loss
$\mathcal{L}(k) = \{\boldsymbol{\lambda} \in \mathcal{L} \mid \|\boldsymbol{\lambda}\|_0 \leq k\}$	feasible coefficients of models with model size $\leq k$
$\mathcal{M}(k) = \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}(k)} l(\boldsymbol{\lambda})$	minimizers of the logistic loss with model size $\leq k$
$L(k) = \min_{\boldsymbol{\lambda}\in\mathcal{L}(k)} l(\boldsymbol{\lambda})$	logistic loss of minimizers with model size $\leq k$
$\boldsymbol{\lambda}^{\text{opt}} \in \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{M}} \|\boldsymbol{\lambda}\|_0$	sparsest minimizer among all minimizers of the logistic loss
$k^{\text{opt}} = \|\boldsymbol{\lambda}^{\text{opt}}\|_0$	model size of sparsest minimizer

Table B.1: Notation used in Remarks B.1 and B.2

Remark B.1 (Minimizers of the Risk Score Problem)
Any optimal solution to the risk score problem will achieve a logistic loss of $L(k)$ for some $k \geq 0$:

$$\min_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0) = \min_{k\in\{0,1,\ldots,k^{\text{opt}}\}} L(k) + C_0 k.$$

Proof (Remark B.1). *Since $L(k)$ is the minimal value of the logistic loss for all models with at most k non-zero coefficients, we have:*

$$L(k) + C_0 k \leq l(\boldsymbol{\lambda}) + C_0 k \text{ for any } \boldsymbol{\lambda} \in \mathcal{L}(k) \tag{B.1}$$

Denote a feasible minimizer of $V(\boldsymbol{\lambda}; C_0)$ as $\boldsymbol{\lambda}' \in \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0)$, and let $k' = \|\boldsymbol{\lambda}'\|_0$. Since $\boldsymbol{\lambda}' \in \mathcal{L}(k')$, we have that:

$$L(k') + C_0 k' \leq V(\boldsymbol{\lambda}'; C_0) \tag{B.2}$$

Taking the minimum of the left hand side of (B.2)*:*

$$\min_{k\in\{0,1,2,\ldots,k^{\text{opt}}\}} L(k) + C_0 k \leq L(k') + C_0 k' \tag{B.3}$$

Combining (B.2) *and* (B.3)*, we get:*

$$\min_{k\in\{0,1,2,\ldots,k^{\text{opt}}\}} L(k) + C_0 k \leq L(k') + C_0 k' \leq \min_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0)$$

If these inequalities are not all equalities, we have a contradiction with the definition of $\boldsymbol{\lambda}'$ and k' as the minimizer of $V(\boldsymbol{\lambda}; C_0)$ and its size. So all must be equality. This proves the statement. ■

Remark B.2 (Small Trade-Off Parameters Do Not Influence Accuracy)
There exists an integer $z \geq 1$ such that if

$$C_0 \leq \frac{1}{z}\left[L(k^{\text{opt}} - z) - L(k^{\text{opt}})\right],$$

then

$$\boldsymbol{\lambda}^* \in \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0).$$

For this z, the right hand side $\frac{1}{z}\left[L(k^{\text{opt}} - z) - L(k^{\text{opt}})\right]$ is strictly greater than 0. Thus, as long as C_0 is sufficiently small, the regularized objective is minimized by any model $\boldsymbol{\lambda}^*$ which has the smallest size among the most accurate feasible models.

Proof (Remark B.2). *Note that if $C_0 = 0$, then any minimizer of $V(\boldsymbol{\lambda}; C_0)$ has k^{opt} non-zero coefficients. Say we increase the value of C_0 in the objective starting from zero until we attain a threshold value ${C^{\min}}_0$ such that the optimal solution will sacrifice some loss to remove at least one non-zero coefficient. Let $z \geq 1$ be the number of non-zero coefficients removed to obtain the smaller model. At ${C^{\min}}_0$ where we choose the smaller model rather than the one with size k^{opt}, we have*

$$L(k^{\text{opt}}) + {C^{\min}}_0 k^{\text{opt}} > L(k^{\text{opt}} - z) + C_0(k^{\text{opt}} - z).$$

Simplifying, we obtain:

$$\frac{1}{z}\left[L(k^{\text{opt}} - z) - L(k^{\text{opt}})\right] < C_0.$$

Thus RISKSLIM *does not sacrifice sparsity for logistic loss when*

$$\frac{1}{z}\left[L(k^{\text{opt}} - z) - L(k^{\text{opt}})\right] \geq C_0.$$

Here, we know that the value on the left side is strictly greater than 0 because otherwise it would contradict the definition of k^* as the smallest size at which the optimal value of the loss would be achieved. ■

Remark B.2 states that there exists a small enough value of C_0 to guarantee that we will obtain the best possible solution. Since we do not know z in advance and since it is just as difficult to compute L as it is to solve the risk score problem, we will not know in advance how small C_0 must be to avoid sacrificing sparsity. However, it does not matter which value of C_0 we choose as long as it is sufficiently small. In practice, we set $C_0 = 10^{-8}$, which is the smallest value that we can use without running into numerical issues with the MIP solver.

B.2 Omitted Proofs

Proof (Remarks 6.5 and 6.6). *We first explain why LCPA attains the optimal objective value, and then justify the upper bounds on the number of cuts and number of nodes.*

Observe that LCPA finds the optimal solution to RiskSlimMINLP *through an exhaustive search over the feasible region $\mathcal{L}$. Thus, LCPA is bound to encounter the optimal solution, since it only discards a node $(v^t, \mathcal{R}^t)$ if: (i) the surrogate problem is infeasible over $\mathcal{R}^t$ (in which case* RiskSlimMINLP *is also infeasible over $\mathcal{R}^t$); or (ii) the surrogate problem has an objective value that exceeds than $V^{\max}$ (in which case, there any integer feasible solution $\mathcal{R}^t$ is also suboptimal).*

The bound on the number of cuts follows from the fact that Algorithm **??** *only adds cuts at integer feasible solutions, of which there are at most $|\mathcal{L}|$. The bound on the number of processed nodes represents a worst-case limit produced by bounding the depth of the branch-and-bound tree. To do this, we exploit the fact that the splitting rule SplitPartition splits a partition into two mutually exclusive subsets by adding integer-valued bounds such $\lambda_j \geq \lceil \lambda_j \rceil$ and $\lambda_j \leq \lfloor \lambda_j \rfloor - 1$ on a single coefficient to the feasible solution. Consider applying SplitPartition a total of $\Lambda_j^{\max} - \Lambda_j^{\min} + 1$ times in succession on a fixed dimension j. This results in a total of $\Lambda_j^{\max} - \Lambda_j^{\min} + 1$ nodes where each node fixes the coefficient in dimension j to an integer value $\lambda_j \in \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\}$. Pick any node and repeat this process for coefficients in the remaining dimensions. The resulting B&B tree will have at most $D(\mathcal{L})$ leaf nodes where $\boldsymbol{\lambda}$ is restricted to integer feasible solutions.* ■

Proof (Proposition 6.7). *Since the coefficient set $\mathcal{L}$ is bounded, the data $(\boldsymbol{x}_i, y_i)_{i=1}^n$ are bounded, and the normalized logistic loss $l(\boldsymbol{\lambda})$ is continuous, it follows that the value of $l(\boldsymbol{\lambda})$ is also bounded:*

$$l(\boldsymbol{\lambda}) \in [\min_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda}), \max_{\boldsymbol{\lambda}\in\mathcal{L}}, l(\boldsymbol{\lambda})] \text{ for all } \boldsymbol{\lambda} \in \mathcal{L}.$$

Thus we only need to show that $L^{\min} \leq \min_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda})$, and $L^{\max} \geq \max_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda})$. For the lower bound, we observe that:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda}) &= \min_{\boldsymbol{\lambda}\in\mathcal{L}} \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-\langle\boldsymbol{\lambda}, y_i\boldsymbol{x}_i\rangle)) \\
&= \min_{\boldsymbol{\lambda}\in\mathcal{L}} \frac{1}{n}\sum_{i:y_i=+1} \log(1+\exp(-\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{n}\sum_{i:y_i=-1} \log(1+\exp(\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&\geq \frac{1}{n}\sum_{i:y_i=+1} \min_{\boldsymbol{\lambda}\in\mathcal{L}} \log(1+\exp(-\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{n}\sum_{i:y_i=-1} \min_{\boldsymbol{\lambda}\in\mathcal{L}} \log(1+\exp(\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&= \frac{1}{n}\sum_{i:y_i=+1} \log(1+\exp(-\max_{\boldsymbol{\lambda}\in\mathcal{L}}\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{n}\sum_{i:y_i=-1} \log(1+\exp(\min_{\boldsymbol{\lambda}\in\mathcal{L}}\langle\boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&= \frac{1}{n}\sum_{i:y_i=+1} \log(1+\exp(-s_i^{\max}) + \frac{1}{n}\sum_{i:y_i=-1} \log(1+\exp(s_i^{\min})) \\
&= L^{\min}.
\end{aligned}$$

The upper bound can be derived in a similar manner. ■

Proof (Proposition 6.8). *We are given that $V^{\max} \geq V$ where $V := l(\boldsymbol{\lambda}^*)+C_0\left\|\boldsymbol{\lambda}^*\right\|_0$ by definition. Thus, we can recover the upper bound from Proposition 6.8 as follows:*

$$\begin{aligned}
l(\boldsymbol{\lambda}^*) + C_0\left\|\boldsymbol{\lambda}^*\right\|_0 &\leq V^{\max}, \\
\left\|\boldsymbol{\lambda}^*\right\|_0 &\leq \frac{V^{\max} - l(\boldsymbol{\lambda}^*)}{C_0}, \\
\left\|\boldsymbol{\lambda}^*\right\|_0 &\leq \frac{V^{\max} - L^{\min}}{C_0}, \qquad \text{(B.4)} \\
\left\|\boldsymbol{\lambda}^*\right\|_0 &\leq \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor. \qquad \text{(B.5)}
\end{aligned}$$

Here, (B.4) *follows from the fact that $L^{\min} \leq l(\boldsymbol{\lambda}^*)$ by definition, and* (B.5) *follows from the fact that the number of non-zero coefficients is a natural number.* ■

Proof (Proposition 6.9). *We are given that* $V^{\max} \geq V$ *where* $V := l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0$ *by definition. Thus, we can recover the upper bound from Proposition 6.9 as follows:*

$$\begin{aligned} l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0 &\leq V^{\max}, \\ l(\boldsymbol{\lambda}^*) &\leq V^{\max} - C_0 \|\boldsymbol{\lambda}^*\|_0, \\ l(\boldsymbol{\lambda}^*) &\leq V^{\max} - C_0 R^{\min}. \end{aligned}$$

Here, the last line follows from the fact that $R^{\min} \leq \|\boldsymbol{\lambda}^*\|_0$ *by definition.* ■

Proof (Proposition 6.10). *We are given that* $V^{\min} \leq V$ *where* $V := l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0$ *by definition. Thus, we can recover the lower bound from Proposition 6.10 as follows:*

$$\begin{aligned} l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0 &\geq V^{\min}, \\ l(\boldsymbol{\lambda}^*) &\geq V^{\min} - C_0 \|\boldsymbol{\lambda}^*\|_0, \\ l(\boldsymbol{\lambda}^*) &\geq V^{\min} - C_0 R^{\max}. \end{aligned}$$

Here, the last line follows from the fact that $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$ *by definition.* ■

Proof (Theorem 6.11). *For a fixed set coefficients $\boldsymbol{\lambda} \in \mathcal{L}$, consider a finite sample of n points composed of the values for the loss function $l_i(\boldsymbol{\lambda})$ for each example in the full training dataset $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1}^n$. Let $l_n(\boldsymbol{\lambda}) = \frac{1}{n}\sum_{i=1}^n l_i(\boldsymbol{\lambda})$ and $l_m(\boldsymbol{\lambda}) = \frac{1}{m}\sum_{i=1}^m l_i(\boldsymbol{\lambda})$. Then, the Hoeffding-Serfling inequality (see e.g., Theorem 2.4 in Bardenet et al., 2015) guarantees the following for all $\varepsilon > 0$:*

$$\Pr\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda}) \geq \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta(\boldsymbol{\lambda}, \mathcal{D}_n)^2}\right),$$

where

$$\Delta(\boldsymbol{\lambda}, \mathcal{D}_n) = \max_{i=1,\ldots,n} l_i(\boldsymbol{\lambda}) - \min_{i=1,\ldots,n} l_i(\boldsymbol{\lambda}).$$

We recover the desired inequality by generalizing this bound to hold for all $\boldsymbol{\lambda} \in \mathcal{L}$ as follows.

$$\begin{aligned}
\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda})\right) \geq \varepsilon\right) &= \Pr\left(\bigcup_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda}) \geq \varepsilon\right)\right), \\
&\leq \sum_{\boldsymbol{\lambda}\in\mathcal{L}} \Pr\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda}) \geq \varepsilon\right), \qquad \text{(B.6)} \\
&\leq \sum_{\boldsymbol{\lambda}\in\mathcal{L}} \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta(\boldsymbol{\lambda}, \mathcal{D}_n)^2}\right), \qquad \text{(B.7)} \\
&\leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta^{\max}(\mathcal{L}, \mathcal{D}_n)^2}\right). \qquad \text{(B.8)}
\end{aligned}$$

Here, (B.6) *follows from the union bound,* (B.7) *follows from the Hoeffding Serling inequality,* (B.8) *follows from the fact that $\Delta(\boldsymbol{\lambda}, \mathcal{D}_n) \leq \Delta^{\max}(\mathcal{L}, \mathcal{D}_n)$ given that $\boldsymbol{\lambda} \in \mathcal{L}$.* ■

Proof (Proof of Corollary 6.12). *We will first show that for any tolerance that we pick $\delta > 0$, the prescribed choice of ε_δ will ensure that $V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1 - \delta$. Restating the result of Theorem 6.11, we have that for any $\varepsilon > 0$:*

$$\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda})\right) \geq \varepsilon\right) \leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta^{\max}(\mathcal{L},\mathcal{D}_n)^2}\right). \tag{B.9}$$

Note that $l_n(\boldsymbol{\lambda}) - l_m(\boldsymbol{\lambda}) = V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda})$ for any fixed $\boldsymbol{\lambda}$. In addition, note that the set of rounded coefficients $\mathcal{L}(\boldsymbol{\rho})$ contains at most $|\mathcal{L}(\boldsymbol{\rho})| \leq 2^d$ coefficients vectors. Therefore, in this setting, (B.9) *implies that for any $\varepsilon > 0$,*

$$\Pr\left(V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) \geq \varepsilon\right) \leq 2^d \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{m})(1-\frac{m}{n})(1+\frac{m}{n})\Delta(\mathcal{L}(\boldsymbol{\rho}),\mathcal{D}_n)^2}\right). \tag{B.10}$$

By setting $\varepsilon = \varepsilon_\delta$ and simplifying the terms on the right hand side in (B.10), *we can see that*

$$\Pr\left(V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) \geq \varepsilon_\delta\right) \leq \delta.$$

Thus, the prescribed value of ε_δ ensures that $V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1 - \delta$.

Since we have set ε_δ so that $V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1-\delta$, we now only need to show any $\boldsymbol{\lambda}$ that satisfies $V_m(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta$ will also satisfy $V_n(\boldsymbol{\lambda}) \leq V^{\max}$ to complete the proof. To see this, observe that:

$$\begin{aligned} V_n(\boldsymbol{\lambda}) - V_m(\boldsymbol{\lambda}) &\leq \varepsilon_\delta, \\ V_n(\boldsymbol{\lambda}) &\leq V_m(\boldsymbol{\lambda}) + \varepsilon_\delta, \\ V_n(\boldsymbol{\lambda}) &< V^{\max}. \end{aligned} \tag{B.11}$$

Here, (B.11) *follows from the fact that $V_m(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta \implies V_m(\boldsymbol{\lambda}) + \varepsilon_\delta < V^{\max}$*
■

B.3 Simulation Procedure

We simulated data from the `breastcancer` dataset (Mangasarian et al., 1995). The original dataset can be obtained from the UCI ML repository (Lichman, 2013), and has a total of $n = 683$ samples and $d = 9$ features $x_{ij} \in \{0, \ldots, 10\}$. Using the original dataset, we generated a collection of simulated datasets using the procedure in Algorithm B.1. Specifically, we first simulated the largest dataset we needed (with $n^{\max} = 5 \times 10^6$ samples and $d^{\max} = 30$ features) by replicating features and samples from the original dataset and adding a small amount of normally distributed noise. Next, we created smaller datasets by taking *nested* subsets of the samples and the features. This ensured that any simulated dataset with d features and n samples contains all of the features and examples for a simulated dataset with $d' < d$ features and $n' < n$ samples. We designed this procedure to have two useful properties:

- It would produce difficult instances of the risk score problem. Here, RiskSlim-MINLP instances for simulated datasets with $d > 9$ are challenging in terms of feature selection because they contain replicates of the original 9 features, which are strongly correlated with each another. Feature selection becomes exponentially harder when collections of highly correlated features are used, since this means that there are an exponentially larger number of slightly suboptimal solutions.
- We could make inferences about the optimal objective value of RiskSlimMINLP instances we may not have been able to solve. Say, for example, that we could not solve an instance of the risk score problem for the simulated dataset with $(d, n) = (20, 10^6)$, but could solve an instance for the simulated dataset with $(d, n) = (10, 10^6)$. In this case, we knew that the optimal objective value of the $(d, n) = (20, 10^6)$ instance had to be less than or equal to the optimal objective value of the $(d, n) = (10, 10^6)$ instance because the $(d, n) = (20, 10^6)$ dataset contained all of the features as the $(d, n) = (10, 10^6)$ dataset.

B.3.1 Implementation Details

We fit risk scores for all datasets by formulating an RiskSlimMINLP instance where $C_0 = 10^{-8}$, $\lambda_0 \in \{-100, 100\}$, and $\lambda_j = \{-10, \ldots, 10\}$ for $j = 1, \ldots, d$. We solved this instance using: (i) CPA (Algorithm 4); (ii) LCPA (Algorithm 5); (iii) an active set MINLP algorithm (ActiveSetMINLP); (iv) an interior point MINLP algorithm (InteriorMINLP); (v) an interior CG MINLP algorithm (InteriorCGMINLP).

We solved all instances on a 3.33 GHz Intel Xeon CPU with 16GB of RAM for up to 6 hours. If an algorithm could not solve the problem to optimality before the 6 hour time limit, we reported results for the best feasible solution it had found. We implemented both CPA and LCPA using the CPLEX 12.6.3 Python API. We solved instances using MINLP algorithms using the Artelsys Knitro 9.0 MINLP solver (Byrd et al., 2006), which we accessed using MATLAB 2015b.

Algorithm B.1 Simulation Procedure

Input

```
    $X^{\text{original}} \leftarrow [x_{ij}]_{i=1\ldots n^{\text{original}}, j=1\ldots d^{\text{original}}}$        feature matrix of original dataset
    $Y^{\text{original}} \leftarrow [y_i]_{i=1\ldots n^{\text{original}}}$        labels of original dataset
    $d^1 \ldots d^{\max}$ s.t. $0 < d^1 \ldots d^{\max}$        desired dimensions for simulated datasets
    $n^1 \ldots n^{\max}$ s.t. $0 < n^1 \ldots n^{\max}$        desired sample sizes for simulated datasets
```

Initialize

```
    $\mathcal{J}^{\text{original}} \leftarrow [1, \ldots, d^{\text{original}}]$        index array for original features
    $\mathcal{J}^{\max} \leftarrow []$        index array of features for largest simulated dataset
    $m^{\text{full}} \leftarrow \lfloor d^{\max} / d^{\text{original}} \rfloor$
    $m^{\text{remainder}} \leftarrow d^{\max} - d^{\text{original}}$
```

Step I: Generate Largest Dataset

```
1: for $m = 1, \ldots, m^{\text{full}}$ do
2:     $\mathcal{J}^{\max} \leftarrow [\mathcal{J}^{max}, \text{RandomPermute}(\mathcal{J}^{\text{original}})]$
3: end for
       $\mathcal{J}^{\max} \leftarrow [\mathcal{J}^{max}, \text{RandomSampleWithoutReplacement}(\mathcal{J}^{\text{original}}, m^{\text{remainder}})]$

4: for $i = 1, \ldots, n^{\max}$ do
5:     sample $l$ with replacement from $1, \ldots n^{\text{original}}$
6:     $y_i^{\max} \leftarrow y_i$
7:     for $j = 1, \ldots, d^{\max}$ do
8:         $k \leftarrow \mathcal{J}^{\max}[j]$
9:         sample $\varepsilon$ from Normal(0, 0.5)
10:        $x_{ij}^{\max} \leftarrow \lceil x_{l,k} + \varepsilon \rfloor$        ▷*new features are noisy versions of original features*
11:        $x_{ij}^{\max} \leftarrow \min(10, \max(0, x_{ij}^{\max}))$        ▷*new features have same bounds as old features*
12:    end for
13: end for
14: $X^{\max} \leftarrow [x_{ij}]_{i=1\ldots n^{\max}, j=1\ldots d^{\max}}$
15: $Y^{\max} \leftarrow [y_i^{max}]_{i=1\ldots n^{\max}}$
```

Step II: Generate Smaller Datasets

```
16: for $d = [d^1, \ldots, d^{\max}]$ do
17:     for $n = [n^1, \ldots, n^{\max}]$ do
18:         $X^{(n,d)} = X^{\max}[1:n, 1:d]$
19:         $Y^n = Y^{\max}[1:n]$
20:     end for
21: end for
```

Output: simulated datasets $(X^{(n,d)}, Y^n)$ for all $n^1 \ldots n^{\max}$ and $d^1 \ldots d^{\max}$.

B.3.2 Results for All MINLP Algorithms

We examined the performance of 3 MINLP algorithms in Artelsys Knitro 9.0:

- ActiveSetMINLP: an active set algorithm.
- InteriorMINLP: an interior-point algorithm.
- InteriorCGMINLP: an interior-point algorithm where the primal-dual KKT system is solved with a conjugate gradient method.

We only show results from ActiveSetMINLP in Chapter 6 as all three algorithms behave similarly. For completeness, we show the results for the omitted MINLP algorithms in Figure B.1.

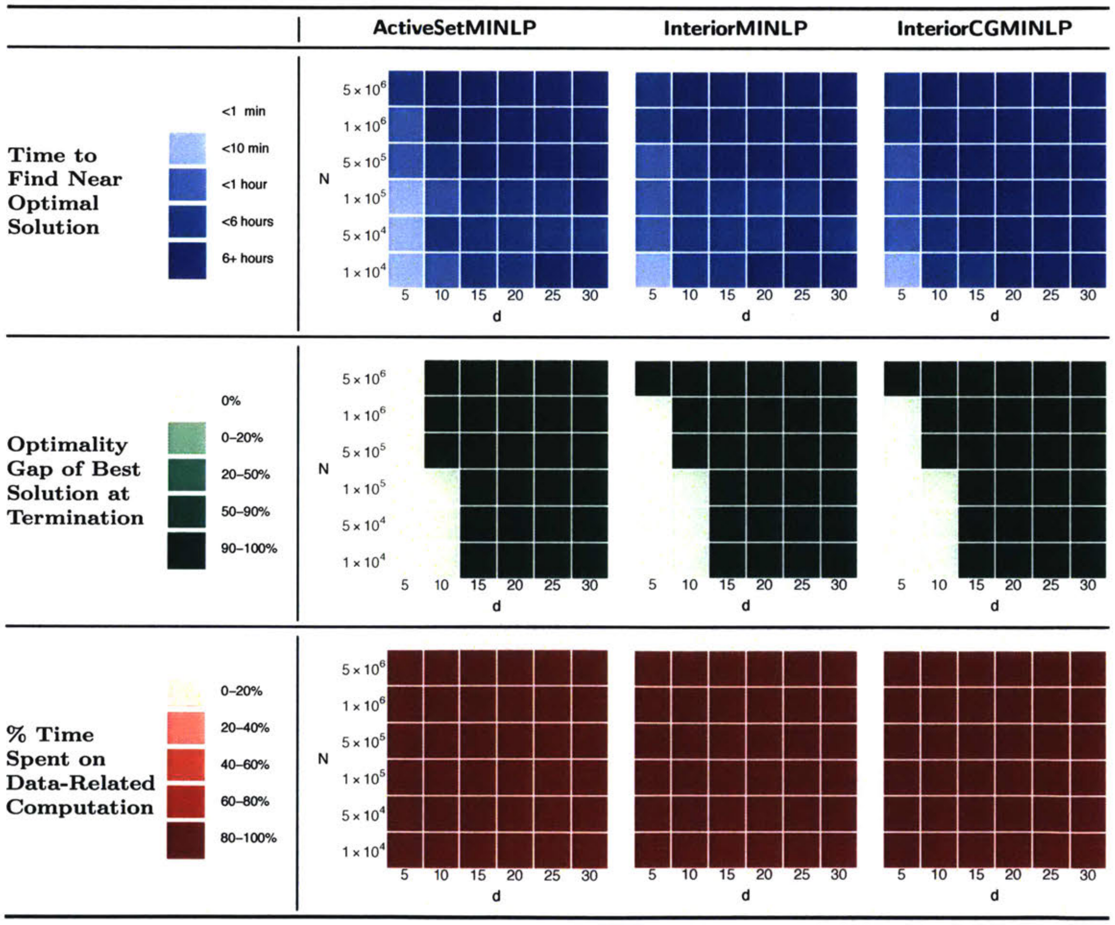


Figure B.1: Performance of MINLP algorithms on hard instances of RISKSLIMMINLP for simulated datasets with varying dimensions d and sample sizes n. All algorithms perform similarly. We report results for ActiveSetMINLP in Chapter 6 because it solves the most instances to optimality.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Supporting Material for Chapter 7

In this appendix, we provide additional results pertaining to the seizure prediction problem in Chapter 7. In Tables C.1 to C.2, we provide a list of all input variables in the training dataset. In Section C.1, we explicitly list all operational constraints for the model. In Section C.2, we show risk scores that were built using the baseline methods that were omitted from the main text.

Input Variable	Range	Sign
Male	$\{0, 1\}$	
Female	$\{0, 1\}$	
AnyPriorSeizure	$\{0, 1\}$	+
PosteriorDominantRhythmPresent	$\{0, 1\}$	−
AnyBriefRhythmicDischarge	$\{0, 1\}$	+
NoReactivityToStimulation	$\{0, 1\}$	
EpileptiformDischarges	$\{0, 1\}$	+
SecondaryDXIncludesMentalStatusFirst	$\{0, 1\}$	
SecondaryDXIncludesCNSInfection	$\{0, 1\}$	
SecondaryDXIncludesCNSInflammatoryDisease	$\{0, 1\}$	
SecondaryDXIncludesCNSNeoplasm	$\{0, 1\}$	
SecondaryDXIncludesHypoxisIschemicEncephalopathy	$\{0, 1\}$	
SecondaryDXIncludesIntracerebralHemorrhage	$\{0, 1\}$	
SecondaryDXIncludesIntraventricularHemorrhage	$\{0, 1\}$	
SecondaryDXIncludesMetabolicEncephalopathy	$\{0, 1\}$	
SecondaryDXIncludesIschemicStroke	$\{0, 1\}$	
SecondaryDXIncludesSubarachnoidHemmorage	$\{0, 1\}$	
SecondaryDXIncludesSubduralHematoma	$\{0, 1\}$	
SecondaryDXIncludesTraumaticBrainInjury	$\{0, 1\}$	
SecondaryDXIncludesHydrocephalus	$\{0, 1\}$	
PatternIsStimulusInducedAny	$\{0, 1\}$	
PatternIsStimulusInducedBiPD	$\{0, 1\}$	
PatternIsStimulusInducedGPD	$\{0, 1\}$	
PatternIsStimulusInducedGRDA	$\{0, 1\}$	
PatternIsStimulusInducedLPD	$\{0, 1\}$	
PatternIsStimulusInducedLRDA	$\{0, 1\}$	
PatternIsSuperImposedAny	$\{0, 1\}$	+
PatternIsSuperImposedBiPD	$\{0, 1\}$	+
PatternIsSuperImposedGPD	$\{0, 1\}$	+
PatternIsSuperImposedGRDA	$\{0, 1\}$	+
PatternIsSuperImposedLPD	$\{0, 1\}$	+
PatternIsSuperImposedLRDA	$\{0, 1\}$	+
PatternsInclude_BiPD	$\{0, 1\}$	+
PatternsInclude_GPD	$\{0, 1\}$	+
PatternsInclude_GRDA	$\{0, 1\}$	+
PatternsInclude_LPD	$\{0, 1\}$	+
PatternsInclude_LRDA	$\{0, 1\}$	+
PatternsInclude_GRDA_or_GPD	$\{0, 1\}$	+
PatternsInclude_BiPD_or_LRDA_or_LPD	$\{0, 1\}$	+

Table C.1: Names, ranges, and sign constraints for input variables in the `seizure` dataset.

Input Variable	**Range**	**Sign**
$MaxFrequencyAnyPattern$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyAnyPattern = 0.0Hz$	$\{0, 1\}$	
$MaxFrequencyAnyPattern \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyAnyPattern \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyAnyPattern \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyAnyPattern \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyAnyPattern \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyAnyPattern \geq 3.0Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyBiPD = 0.0$	$\{0, 1\}$	
$MaxFrequencyBiPD \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyBiPD \geq 3.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyGPD = 0.0$	$\{0, 1\}$	
$MaxFrequencyGPD \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGPD \geq 3.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyGRDA = 0.0$	$\{0, 1\}$	
$MaxFrequencyGRDA \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyGRDA \geq 3.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyLPD = 0.0$	$\{0, 1\}$	
$MaxFrequencyLPD \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLPD \geq 3.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA$	$\{0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$	+
$MaxFrequencyLRDA = 0.0$	$\{0, 1\}$	
$MaxFrequencyLRDA \geq 0.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA \geq 1.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA \geq 1.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA \geq 2.0Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA \geq 2.5Hz$	$\{0, 1\}$	+
$MaxFrequencyLRDA \geq 3.0Hz$	$\{0, 1\}$	+

Table C.2: Names, ranges, and sign constraints for input variables in the `seizure` dataset.

C.1 Operational Constraints

No Redundancy between Categorical Variables

1. Use either *Male* or *Female*.
2. Use either *PatternsInclude_GRDA_or_GPD* or any one of (*PatternsInclude_GRDA*, *PatternsInclude_GPD*).
3. Use either *PatternsInclude_BiPD_or_LRDA_or_LPD* or any one of (*PatternsInclude_BiPD*, *PatternsInclude_LRDA*, *PatternsInclude_LPD*).
4. Use either *MaxFrequencyAnyPattern*$=$ 0.0 or *MaxFrequencyAnyPattern*$\geq$ 0.5.
5. Use either *MaxFrequencyLPD*$=$ 0.0 or *MaxFrequencyLPD*$\geq$ 0.5.
6. Use either *MaxFrequencyGPD*$=$ 0.0 or *MaxFrequencyGPD*$\geq$ 0.5.
7. Use either *MaxFrequencyGRDA*$=$ 0.0 or *MaxFrequencyGRDA*$\geq$ 0.5
8. Use either *MaxFrequencyBiPD*$=$ 0.0 or *MaxFrequencyBiPD*$\geq$ 0.5.
9. Use either *MaxFrequencyLRDA*$=$ 0.0 or *MaxFrequencyLRDA*$\geq$ 0.5.

Frequency in Continuous Encoding or Thresholded Encoding

10. Choose between *MaxFrequencyAnyPattern* or (*MaxFrequencyAnyPattern*$=$ 0.0, ..., *MaxFrequencyAnyPattern*$\geq$ 3.0).
11. Choose between *MaxFrequencyGPD* or (*MaxFrequencyGPD*$=$ 0.0, ..., *MaxFrequencyGPD*$\geq$ 3.0).
12. Choose between *MaxFrequencyLPD* or (*MaxFrequencyLPD*$=$ 0.0, ..., *MaxFrequencyLPD*$\geq$ 3.0).
13. Choose between *MaxFrequencyGRDA* or (*MaxFrequencyGRDA*$=$ 0.0, ..., *MaxFrequencyGRDA*$\geq$ 3.0).
14. Choose between *MaxFrequencyBiPD* or (*MaxFrequencyBiPD*$=$ 0.0, ..., *MaxFrequencyBiPD*$\geq$ 3.0).
15. Choose between *MaxFrequencyLRDA* or (*MaxFrequencyLRDA*$=$ 0.0, ..., *MaxFrequencyLRDA*$\geq$ 3.0).

Limited # of Thresholds for Thresholded Variables

16. Use at most 2 of the following: *MaxFrequencyAnyPattern*$=$ 0.0, *MaxFrequencyAnyPattern*$\geq$ 0.5, ..., *MaxFrequencyAnyPattern*$\geq$ 3.0.
17. Use at most 2 of the following: *MaxFrequencyLPD*$=$ 0.0, *MaxFrequencyLPD*$\geq$ 0.5, ..., *MaxFrequencyLPD*$\geq$ 3.0.

18. Use at most 2 of the following: *MaxFrequencyGPD*$=$ 0.0, *MaxFrequencyGPD*$\geq$ 0.5, ..., *MaxFrequencyGPD*$\geq$ 3.0.

19. Use at most 2 of the following: *MaxFrequencyGRDA*$=$ 0.0, *MaxFrequencyGRDA*$\geq$ 0.5, ..., *MaxFrequencyGRDA*$\geq$ 3.0.

20. Use at most 2 of the following: *MaxFrequencyBiPD*$=$ 0.0, *MaxFrequencyBiPD*$\geq$ 0.5, ..., *MaxFrequencyBiPD*$\geq$ 3.0.

21. Use at most 2 of the following: *MaxFrequencyLRDA*$=$ 0.0, *MaxFrequencyLRDA*$\geq$ 0.5, ...,*MaxFrequencyLRDA*$\geq$ 3.0.

Specific cEEG Patterns or Any cEEG Pattern

22. Use either *PatternIsStimulusInducedAny* or any one of (*PatternIsStimulusInducedBiPD*, *PatternIsStimulusInducedGRDA*, *PatternIsStimulusInducedGPD*, *PatternIsStimulusInducedLPD*, *PatternIsStimulusInducedLRDA*).

23. Use either *PatternIsSuperImposed* or any one of (*PatternIsSuperImposedBiPD*, *PatternIsSuperImposedGPD*, *PatternIsSuperImposedGRDA*, *PatternIsSuperImposedLPD*, *PatternIsSuperImposedLRDA*).

24. Use either *MaxFrequencyAnyPattern* (or its thresholded versions) or any one of *MaxFrequencyBiPD*, *MaxFrequencyGRDA*, *MaxFrequencyGPD*, *MaxFrequencyLPD*, *MaxFrequencyLRDA*, (or their thresholded versions).

C.2 Risk Scores for Omitted Baseline Methods

1.	*AnyPriorSeizure*	1 point		···
2.	*PatternsInclude_ BiPD_ or_ LRDA_ or_ LPD*	1 point	+	···
3.	*MaxFrequencyLPD*	× 1 point per Hz	+	···
	ADD POINTS FROM ROWS 1–3	**SCORE**	=	···

SCORE	0.0	1.0	2.0	2.5	3.0	3.5	4.0	4.5	5.0
RISK	4.7%	11.9%	26.9%	37.8%	50.0%	62.2%	73.1%	81.8%	88.1%

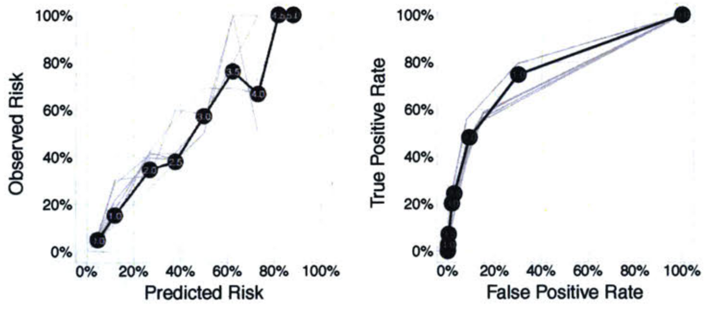


Figure C.1: SEQRD model (top), reliability diagram (bottom left), and ROC curve (bottom right). We plot results for fold models on test data in grey, and for the final model on training data in black. This model has a 5-CV mean test CAL/AUC of 3.7%/0.738.

1.	*AnyPriorSeizure*	1 point	···
2.	*PatternsInclude_BiPD_or_LRDA_or_LPD*	1 point	+ ···
3.	*MaxFrequencyLPD*	× 1 point per Hz	+ ···
	ADD POINTS FROM ROWS 1–3	**SCORE**	= ···

SCORE	0.0	1.0	2.0	2.5	3.0	3.5	4.0	4.5	5.0
RISK	4.7%	11.9%	26.9%	37.8%	50.0%	62.2%	73.1%	81.8%	88.1%

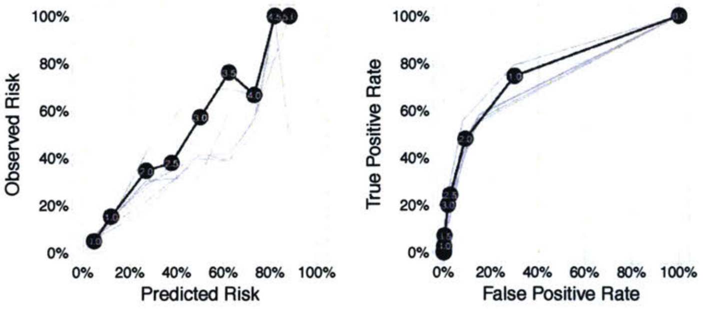


Figure C.2: RD* model (top), reliability diagram (bottom left), and ROC curve (bottom right). We plot results for fold models on test data in grey, and for the final model on training data in black. This model has a 5-CV mean test CAL/AUC of 3.3%/0.738.

1.	*AnyPriorSeizure*	5 points	···
2.	*PatternsIncludeBiPD_or_LRDA_or_LPD*	5 points	+ ···
3.	*MaxFrequencyLPD*	× 2 points per Hz	+ ···
	ADD POINTS FROM ROWS 1–3	**SCORE**	= ···

SCORE	0 to 5	7	8	9	10	11	12 to 16
RISK	<5.0%	11.9%	26.9%	50.0%	73.1%	81.8%	>95.0%

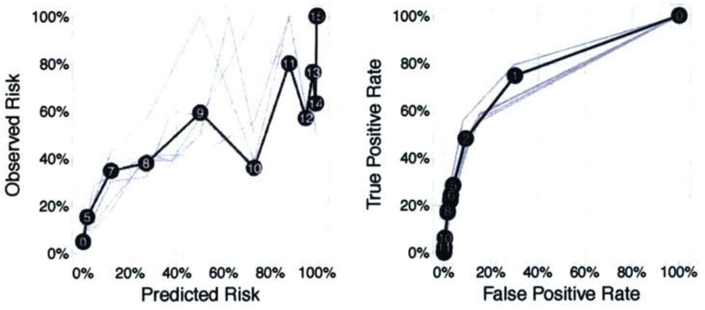


Figure C.3: RSRD* model (top), reliability diagram (bottom left), and ROC curve (bottom right). We plot results for fold models on test data in grey, and for the final model on training data in black. This model has a 5-CV mean test CAL/AUC of 8.2%/0.738.

1.	*AnyPriorSeizure*	1 point	···
2.	*PatternsIncludeBiPD_or_LRDA_or_LPD*	1 point	+ ···
3.	*MaxFrequencyLPD*	× 1 point per Hz	+ ···
	ADD POINTS FROM ROWS 1–3	**SCORE**	= ···

SCORE	0.0	1.0	2.0	2.5	3.0	3.5	4.0	4.5	5.0
RISK	4.7%	11.9%	26.9%	37.8%	50.0%	62.2%	73.1%	81.8%	88.1%

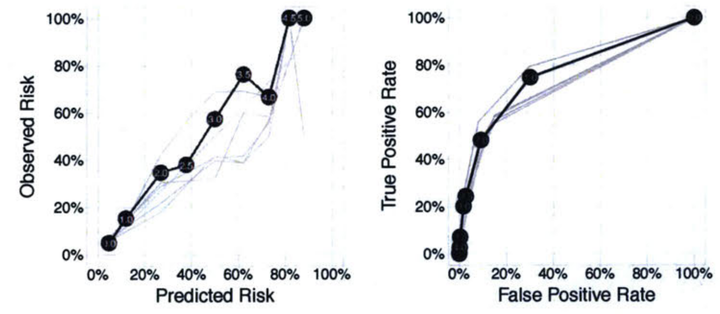


Figure C.4: SEQRD* model (top), reliability diagram (bottom left), and ROC curve (bottom right). We plot results for fold models on test data in grey, and for the final model on training data in black. This model has a 5-CV mean test CAL/AUC of 3.3%/0.738.