

# ARDA: Automatic Relational Data Augmentation for Machine Learning

by

Nadiia Chepurko

Submitted to the Department of Electrical Engineering and Computer  
Science

in partial fulfillment of the requirements for the degree of  
Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

© Massachusetts Institute of Technology 2020. All rights reserved.

Signature redacted

Author .....

Department of Electrical Engineering and Computer Science  
January 30, 2020

Signature redacted

Certified by .....

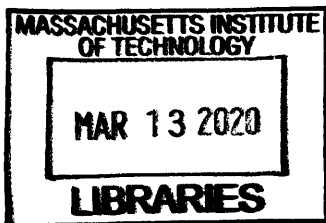
David R. Karger  
Professor of Electrical Engineering and Computer Science  
Thesis Supervisor

Signature redacted

Accepted by .....

/ Leslie A. Kolodziejski

Professor of Electrical Engineering and Computer Science  
Chair, Department Committee on Graduate Students



ARCHIVES



# ARDA: Automatic Relational Data Augmentation for Machine Learning

by

Nadiia Chepurko

Submitted to the Department of Electrical Engineering and Computer Science  
on January 30, 2020, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Computer Science and Engineering

## Abstract

This thesis is motivated by two major trends in data science: easy access to tremendous amounts of unstructured data and the effectiveness of Machine Learning (ML) in data driven applications. As a result, there is a growing need to integrate ML models and data curation into a homogeneous system such that the model informs the choice and extent of data curation. The bottleneck in designing such a system is to efficiently discern what additional information would result in improving the generalization of the ML models.

We design an end-to-end system that takes as input a data set, a ML model and access to unstructured data, and outputs an augmented data set such that training the model on this dataset results in better generalization error. Our system has two distinct components: 1) a framework to search and join unstructured data with the input data, based on various attributes of the input and 2) an efficient feature selection algorithm that prunes our noisy or irrelevant features from the resulting join. We perform an extensive empirical evaluation of system and benchmark our feature selection algorithm with existing state-of-the-art algorithms on numerous real-world datasets.

Thesis Supervisor: David R. Karger

Title: Professor of Electrical Engineering and Computer Science



## Acknowledgments

First and foremost, I would like to thank my advisor, David Karger, without whose unwavering support this thesis would not be possible. I would also like to thank Tim Kraska for outlining the goals of the project and providing valuable insights during crucial stages. In addition, I am grateful to my collaborators Raul Castro Fernandez, Ryan Marcus and Emmanuel Zraggen.



# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Introduction . . . . .	13
1.2	Related Work . . . . .	15
1.3	Problem Description . . . . .	18
<b>2</b>	<b>Data Augmentation</b>	<b>21</b>
2.1	Augmentation workflow . . . . .	21
2.1.1	Coreset Constructions . . . . .	24
2.2	Joins . . . . .	27
2.2.1	Join type . . . . .	27
2.2.2	Key Matches . . . . .	28
2.2.3	Join cardinality . . . . .	30
2.2.4	Imputation . . . . .	30
<b>3</b>	<b>Feature Selection</b>	<b>33</b>
3.1	Overview of Feature Selection . . . . .	33
3.2	Random Injection Based Feature Selection . . . . .	36
3.2.1	Random Feature Injection . . . . .	38
3.2.2	Ranking Ensembles . . . . .	39
3.2.3	Aggregate Ranking . . . . .	41
3.2.4	Putting it together. . . . .	41
3.3	$L_{2,1}$ -Norm Minimization and Sketching . . . . .	42

<b>4</b>	<b>Experiments</b>	<b>47</b>
4.1	Micro Benchmarks . . . . .	47
4.2	Coreset Comparison . . . . .	49
4.3	Experimental Evaluation of ARDA . . . . .	51

# List of Figures

1-1	Example Schema: Initially a user has a base table <code>TRAFFIC_ACCIDENT</code> and she finds a pool of joinable tables to see if some of them can help improve prediction error for severity of a traffic accident. . . . .	19
2-1	The ARDA . . . . .	22
4-1	Comparison of the running time for each feature selection algorithm with the improvement in prediction accuracy achieved over the base table (no augmentation). . . . .	49
4-2	Micro Benchmark for Feature Selection Algorithms on Digits, Kraken and Flights data sets. The original data sets are appended with 1000× artificial, noisy features (as described above). The $x$ -axis represents number of features, the $y$ -axis denotes the algorithm and the plot compares the true features extracted vs the artificial features extracted. .	50
4-3	Coreset comparison using various feature selection algorithm on School, Digits and Kraken data sets. The coresets are construction via Uniform or Stratified sampling 1000 training samples. The $x$ -axis denotes the feature selection algorithms and the $y$ -axis denotes the corresponding prediction accuracy. . . . .	53
4-4	Comparison of the running time for each feature selection algorithm with the improvement in prediction accuracy achieved over the base table (no augmentation). . . . .	54



# List of Tables

3.1	Comparison of popular subset selection algorithms, given a ranking of the features. We note that various learning models provide the ranking for these algorithms. $T(n)$ denotes the time evaluate the learning algorithm and $S(n)$ is the corresponding space used. . . . .	34
3.2	Comparison of popular learning models for feature selection. . . . .	35
4.1	We compare the performance of joining one table at a time vs joining at most $ S $ features at a time ( $ S $ is the size of a coreset). The first column denotes the multiplicative speedup factor and the second column denotes increase in accuracy w.r.t. full materialization. . . .	52
4.2	We compare the performance of joining all tables at once (full materialization) vs joining at most $ S $ features at a time ( $ S $ is the size of a coreset). The first column denotes the multiplicative speedup factor and the second column denotes increase in accuracy w.r.t. full materialization. . . . .	52



# Chapter 1

## Introduction

### 1.1 Introduction

Today we have the access to tremendous amounts of data on the web. Google Dataset Search provides a simple interface to mine data from various sources. Governments, academic institutions and companies are making their data sets publicly available. This, combined with the astonishingly increasing use of machine learning (ML) in data driven applications has led to a growing interest in integrating ML and data curation/processing. However, existing ML toolkits and frameworks assume that the data is stored in a single table on a single machine. On the contrary, most of the data is stored across the web, in an unstructured form across multiple tables connected by key-foreign key dependencies (KFKDs).

Since the success of modern ML techniques, such as Deep Learning, is contingent on larger training sets and more features, analysts spend a majority of their time finding relevant data to answer the questions at hand than trying various algorithmic solutions and analyzing the results. The need for data and feature discovery discovery has therefore opened the door for a new research area that tries to utilize this additional features for improving prediction tasks. Data Enrichment is one such area that only recently defined itself as a response to a need of improved ML predictions by agglomerating data from external knowledge sources. This paper focuses on the problem of Data Augmentation, a pre-processing step that supplements the base data

for a ML task by finding and adding relevant features from multiple heterogeneous sources.

For instance, consider the critical task of predicting the taxi demand for a given day. Accurate prediction is important for taxi companies in order to dispatch the right amount of cars for cost optimization. The company gathers the history of all the taxi trips over a period of time, as well as statistics of the trip like start time, end time, distance covered, route taken and so on. However, given information might not be enough for accurate predictions of the future demand, even for the "perfect" learning model. In this scenario the remaining solution is to search for external knowledge that we can augment in original dataset to improve the prediction. An example of such knowledge source is a weather dataset that is implicitly connected with taxi data through KFKD relationship on time series columns. Weather datasets are readily available online and can be successfully incorporated using the join procedure.

While in this example it wasn't hard for us to infer the strong correlation between taxi demand and weather conditions, automating such inference for different tasks over massive, unstructured data is extremely challenging. Knowledge Discovery and Data Mining study algorithms to extract a semantic understanding from large amounts of unstructured data. However, for a given predictive task, the space of possible augmentations grows exponentially in the size of the data and a semantic understanding does not provide new features to augment our initial dataset. Exploring each possibility quickly becomes computationally infeasible. Further, each augmentation adds a computational overhead in terms of data cleaning, pre-processing and performing a join.

Since most of the open source datasets have only meager unstructured description, or no description at all, we cannot rely on semantic relationships between datasets to infer their logical connections. It is also possible that semantic relationships do not imply feature-target correlation. For example, taxi and weather has small semantic correlation, but very strong predictive correlation for the aforementioned learning task. Therefore, apriori, we do not know what features are useful in the prediction task and what features are coming from spurious irrelevant datasets.

Further, agnostically adding all joinable features at once results in an extremely noisy training set. Existing ML models are brittle to noise and thus generalize poorly in such a setting. In particular, massive noise can wash out existing signal and prediction performance may drop below the baseline score altogether. Therefore, we require a system that automatically and efficiently discovers relevant features to extract from a large set of potentially joinable tables, such that the augmented data set improves the generalization error of the ML model. Our main contribution is to design an end-to-end system that incorporates data formatting, sampling, joining, aggregation, imputation and feature selection (or pruning noise) such that the ML model obtains better performance on prediction tasks.

## 1.2 Related Work

While there has been extensive prior work on Data Mining, Knowledge Discovery, Data Augmentation and Feature Selection, we are not aware of previous work on feature discovery via database joins to improve performance of predication tasks. We provide a brief overview of this literature and indicate how it differs from our setup.

**Data Mining and Knowledge Discovery.** Knowledge Discovery (KD) focuses on extracting useful information from large data sets and aims at a semantic understanding of this data. The algorithms developed extract new concepts or concept relationships hidden in volumes of raw data. See [ME08] for a survey on KD and [KM06] for overview in the context of databases. Data Mining (DM) is the application of specific algorithms for the extraction of nuggets of knowledge, i.e., pertinent patterns, pattern correlations, estimation or rules, from data. These algorithms include Clustering [Ber06], Regression and PCA [LL06]. In this vast literature, we focus on Data Discovery and Data Augmentation for finding related tables.

**Data Discovery.** Data Discovery systems deal with sharing datasets, searching new datasets, and discovering relationships withing heterogeneous data pool. There have been several works that extract tables from the Web [BDE<sup>+</sup>15, BBC<sup>+</sup>14, BCH<sup>+</sup>15, Hal13, GHJ<sup>+</sup>10]. For instance, [GB06] extract tables from arbitrary Web

pages relying on positional information of visualized DOM element nodes in a browser. Cafarella et al. [CHK09] developed the WebTables system for web-scale table extraction, implementing a mix of hand-written detectors and statistical classifiers, and Octopus [YGCC12] even try to identify additional information from online data sources. The Aurum system [FAK<sup>+</sup>18] helps find KFKDs between different sources of data. In addition to the above systems, there are many data portal platforms such as CKAN, Quandl, DataMarket, Kaggle etc. Work on data searching lakes includes but not limited to [TSRC15,HKN<sup>+</sup>16b,HKN<sup>+</sup>16a,TSRC15,CFDM<sup>+</sup>17]. Recently Google release Google Dataset Search engine that helps finding datasets across different web portals in no time.

**Data Augmentation.** Data Augmentation has three main techniques: *embedding augmentation*, *entity augmentation*, and *join augmentation*. The goal of *embedding augmentation* is to learn document’s low dimensional representations and be able to compare these representations to decide which documents have higher chance to describe a similar data under some notion of similarity. Some popular text embeddings include Word2Vec [MSC<sup>+</sup>13], Doc2Vec [LM14], and Table2Vec [Den18] that let us convert an entire document into vector representation. *Entity augmentation* deals with finding and bringing additional data that describes the same entity. Octopus is a system that combines search, extraction, data cleaning and integration, and enables users to create new data sets from those found on the Web. InfoGather presents three core operations, namely entity augmentation by attribute name, entity augmentation by example and attribute discovery, that are useful for “information gathering” tasks. *Augmentation using Joins* is one of the most recent research areas within Data Augmentation. The motivating problem stems from the fact that ML systems let you to feed a single csv file with an assumption that this file contains all needed information for predictive tasks. Since most databases have normalized schemas and so facts are scattered across multiple tables one needs to perform joins before running ML models. In projects Hamlet/Hamlet++ and their corresponding papers [KNPZ16] and [SKZ17] authors study effects of KFKDs within normalized databases. They propose a set of decision rules to predict when it is safe to avoid

joins and reduce the processing time.

**Data Generation.** During join discovery, one almost certainly has to deal with missing values. This process is referred to as data imputation and is a challenging problem for mixed datasets. The two main types of data generation include crowd-sourcing and synthetic data generation that if further divided into statistics-based methods and machine learning based methods. We refer the reader to the following [MWYJ02,DVDHSM06,GBGB09,GLH15,PHW16,YWL<sup>+</sup>16,LG17,SPC<sup>+</sup>19] and references therein. While data generation can help to fill in missing entries, the techniques developed here do not lend themselves well to data augmentation.

**Robustness and Bias-Variance Tradeoff.** A recent line of work on adversarial learning also provides evidence to the challenging nature of selecting features in the presence of noise and unstructured data. A natural approach to feature augmentation would be to join all compatible tables and rely of the learning algorithm to ignore irrelevant features. However, this requires our learning models to be incredibly robust to noisy features.

Unfortunately, ML algorithms are highly sensitive to noisy, possibly adversarial features [MMS<sup>+</sup>17,WK17,SZS<sup>+</sup>13,CW17,AEIK17]. Therefore, the challenge of efficiently finding relevant features to join with our base table, while simultaneously pruning out noise remains.

Given that the feature set plays a crucial role the in the predictive power of ML models, removing redundant and irrelevant features is crucial and has been extensively studied. The conventional wisdom in designing feature selection algorithms is to reduce both bias and variance of the model. Recall, bias accumulates when the model tries to generalize using erroneous assumptions in the training data and noisy features contribute to the variance [Dom99].

It is well known that the more sensitive the model is to the training data, the lower the bias in exchange for a higher variance<sup>1</sup>. Therefore, classifiers with limited data use feature selection to find an optimum point where they can actually estimate

---

<sup>1</sup>See Wikipedia page on Bias-Variance Tradeoff : [https://en.wikipedia.org/wiki/Bias-variance\\_tradeoff](https://en.wikipedia.org/wiki/Bias-variance_tradeoff)

the statistical distribution of fewer features (variance reduction) versus less accurate estimation of more features (bias reduction) [Fri97, KJ97]. However, our approach instead relies on finding as many relevant features as possible. While such an approach works in practice, classical theory suggests it should lead to over-fitting.

### 1.3 Problem Description

In this section, we describe the precise setup and problem statement we consider. Our system is given as input a base table with labelled data and the goal is to perform inference on this dataset. We assume the learning model is also specified and typically is computationally expensive to train. The models we train include Random Forests, Logistic Regression, Lasso and SVMs. We also assume we have access to a hold-out set where we can test the performance of our algorithms on fresh data to inform hyper-parameter search. We train the learning model on our base table and test it on the holdout set to obtain a baseline score.

Our goal then is to augment features to our base table such that we can obtain a non-trivial improvement on the prediction task. To this end, we assume we know the key on which we should attempt to join new tables with the base table. Here, we allow for both hard and soft joins. For instance, if the base key specifies time-series data, we automatically perform soft join using nearest neighbour approach or linear interpolation over backward/forward join. We note that for soft joins over non-time data our system needs explicit indication of a soft join (specified as \* over the join key), otherwise standard exact-match join is performed.

In order to collect joinable datasets for our basetables described in Chapter 2 used the DataMart / Auctus<sup>2</sup> repository to search for datasets over the web. DataMart is a web crawler and search engine for datasets, specifically designed for join discovery. While DataMart provides multiple potential tables to join with, most of the augmented features are spurious since they are coming from datasets that have no useful information for the target learning task. The datasets are collected from numerous

---

<sup>2</sup><https://auctus.vida-nyu.org/>

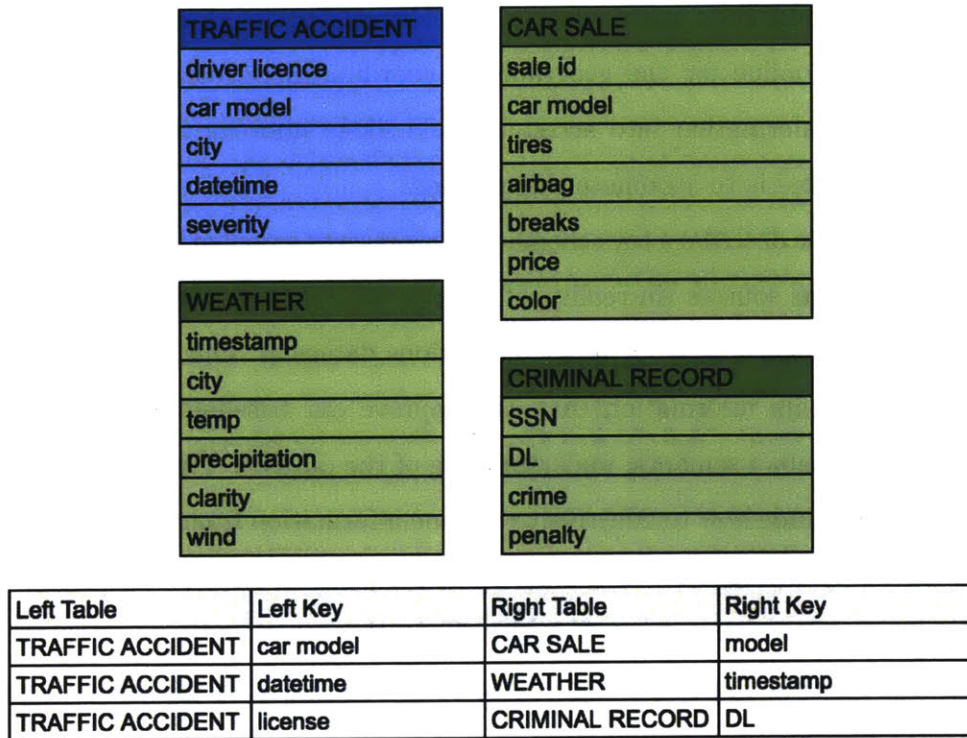


Figure 1-1: Example Schema: Initially a user has a base table `TRAFFIC_ACCIDENT` and she finds a pool of joinable tables to see if some of them can help improve prediction error for severity of a traffic accident.

sources, including Wiki Data <sup>3</sup>, Figshare <sup>4</sup>, OSF <sup>5</sup> etc.

**Example of an augmentation scenario.** For the purposes of this thesis, it is helpful to keep the following canonical example in mind: Given a dataset of car accidents as a relational database, consider a relation `TRAFFIC_ACCIDENT` which contains information about car accidents with a *severity* attribute that ranks the severity of an accident on a scale from 1 to 5. The learning task is to train a model that predicts the severity of an accident, given features such as time of day, geographical coordinates, vehicles involved and so on.

However, the prediction error on the base table is lower than desired. We then decide to try data discovery tools to find potentially relevant features that have direct

<sup>3</sup>[https://www.wikidata.org/wiki/Wikidata:Main\\_Page](https://www.wikidata.org/wiki/Wikidata:Main_Page)

<sup>4</sup><https://figshare.com/>

<sup>5</sup><https://osf.io/>

or transitive dependencies with relation `TRAFFIC_ACCIDENT`. Presumably, weather data is available online for the geographical coordinates we consider in our data. Taking weather information into account would likely improve the performance of our model. Our goal is to automate this process.

In Figure 1-1 we describe a toy schema that represents a pool of mined data coming from heterogeneous sources (in reality, of course, this schema would be much bigger since it would contain a large quantity of spurious datasets). Our task is to discover if any tables contain information that can improve our baseline accuracy, without attempting to obtain a semantic understanding of the dataset. The central questions that arise here include how to effectively combine information from multiple relations, how to efficiently determine which other relations provide valuable information and how to process a massive number of tables efficiently.

Once the relevant join is performed, we must contend with the massive number of irrelevant features interspersed with a small number of relevant features. Further, we empirically observed that the resulting dataset can often have worse prediction error than the baseline, given that modern machine learning models are sensitive to noise (as discussed above). Therefore, our implementation should efficiently select relevant features in the presence of a large amount of irrelevant or noisy features. The rest of this thesis is devoted to describing our system and how it addresses the aforementioned challenges.

# Chapter 2

## Data Augmentation

In this section, we dive into the challenges around data augmentation and describe the details of the **Automatic Relational Data Augmentation (ARDA)** system we implement.

### 2.1 Augmentation workflow

We begin with a high-level description of the workflow our system follows.

**Input to ARDA.** ARDA requires a reference to a data repository and a text file that describes on which keys the join should be performed on (KFKD file). We also account for an optional join ranking order to be specified in KFKD file. The ordering for joins is given by systems such as Aurum [FAK<sup>+</sup>18] or DataMart<sup>1</sup> where priority is based on the intersection size between two tables on a join key. For example, DataMart provides join meta information in json format where the ranking priority is specific by *score* attribute. While this ordering does not directly correspond to an importance of given a table to downstream learning tasks, we make use the size of intersection between two tables as a heuristic for implicit similarity between tables.

---

<sup>1</sup><https://auctus.vida-nyu.org/>

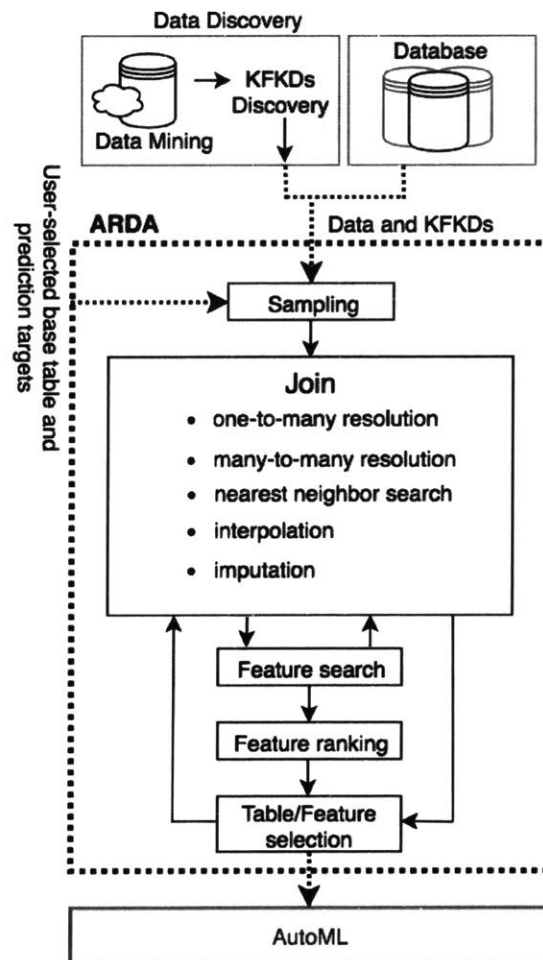


Figure 2-1: The ARDA

**Preprocessing and corest construction.** Having a well defined input, we proceed to preprocess the data and create a representative subset.

1. *Sampling for coresets construction.* ARDA allows several types of corest construction described in Subsection 2.1.1. If sample size is specified, ARDA samples rows according to a customized procedure, the default being uniform sampling.
2. *Column-type resolution.* Careful resolution of data types is important since we want to identify categorical and numerical columns for applying the correct data imputation technique and formatting the features. ARDA uses statistics about column distribution and data format to infer the type.
3. *Time series format resolution.* If the input is time series data, ARDA converts every datetime column to the same format and granularity.

**Joining.** There are 3 types of a join that ARDA supports:

1. *Table-join:* One table at a time in the priority order specified by input meta file. Based on our experiments it is the least desirable type of join since it adds significant time overhead and does not consider co-predictor variables that might be contained in other tables.
2. *Budget-join:* As many tables at a time as we can fit within a predefined *budget*. The budget is a user defined parameter, set to be the number of rows by default. Size of *budget* trades off the number of co-predictor variables we might consider versus amount of noise the model can tolerate to distinguish between good and bad features.
3. *Full materialization join:* All the tables prior to performing feature selection.

We compare performance of *table-join* and *full materialization join* methods in respect to *budget-join* method in Table 4.2.

**Hyper-parameter optimization.** Before feature selection begins ARDA performs light hyper-parameter optimization for selected join algorithms using random search for large search spaces or grid search for small search spaces. Improving hyper-parameter optimization techniques is left for the future work.

**Feature Selection.** ARDA considers various types of feature selection algorithms that can be run simultaneously, which we discuss in subsequent chapters. These methods include convex and non-linear models, such as Sparse Regression and Random Forests. Unless explicitly specified ARDA uses a new feature selection algorithm method that we introduce in Chapter 3. This algorithm is based on injecting random noise into the dataset and abbreviated as RIFS. Since it consistently out-performed the state-of-the-art feature selection algorithms, it is a clear choice to default to. Finally, we compare the running time and accuracy for different methods Chapter 4.

**Final estimate.** After ARDA processes all tables and saved intermediate features it tests the prediction performance using auto-optimized learning models to report change in prediction accuracy. In the case of successful data augmentation selected features are saved with corresponding table references.

### 2.1.1 Coreset Constructions

In this subsection, we discuss various approaches used by ARDA to sample rows of our input to reduce the time we spend on joining, feature selection, model training, and score estimation. While we discuss generic approaches to sampling rows, often the input data is structured and the downstream tasks are known to the users. In such a setting, the user can use specialized coreset constructions to sample rows. We refer the reader to an overview of coreset constructions in [Phi16] and the references therein.

Coreset construction can be viewed as a technique to replace large data set with a smaller number of well-representative points. This corresponds to reducing the number of rows (training points) of our input data. We consider two main techniques

for coreset construction: sampling and sketching. Sampling, as the name suggests, selects a subset of the rows and re-weights them to obtain a coreset. On the other hand, Sketching relies on taking sparse linear combination of rows, which inherently results in modified row values. This limits our use of sketching before we join tables since the sketched data may result in joins that are vastly inconsistent with the original data.

**Uniform Sampling.** The simplest strategy to construct a coreset is uniformly sampling the rows of our input tables. This process is extremely efficient since it does not require reading more of the input.

However, uniform sampling does not have any provable guarantees and is not sensitive to the data. It is also agnostic to outliers, labels and anomalies in the input. For instance, if our input tables are labelled data for classification tasks, and one label appears way more often than others, a uniform sample might miss out of sampling rows corresponding to some labels. In other words, the sample we obtain may not be diverse or well-balanced.

**Stratified Sampling.** To address the shortcomings of uniform sampling, we consider stratified sampling. Stratification is the process of dividing the input into homogeneous subgroups before sampling, such that the subgroups form a partition of the input. Then simple random sampling or systematic sampling can be applied within each stratum.

The objective is to improve the precision of the sample by reducing sampling error. It can produce a weighted mean that has less variability than the arithmetic mean of a simple random sample of the population. For classification tasks, if we stratify based on labels and use uniform sampling within each stratum, we obtain a diverse, well-balanced sub-sample, and no label is overlooked.

**Matrix Sketching.** Finally, consider sketching algorithms to sub-sample the rows of our input tables. Sketching has become a useful algorithmic primitive in many big-data tasks and we refer the reader to a recent survey [Woo14] . We note that under

the right conditions, sketching the rows of the input data approximately preserves the subspace spanned by the columns.

An important primitive in the sketch-and-solve paradigm is a subspace embedding [CW13, NN13], where the goal is to construct a concise describe of data matrix that preserves the norms of vectors restricted to a small subspace. Constructing subspace embeddings has the useful consequence that accurate solutions to the sketched problem are approximate accurately solutions to the original problem.

**Definition 1.** (*Oblivious subsapce embedding.*) Given  $\epsilon, \delta > 0$  and a matrix  $A$ , a distribution  $\mathcal{D}(\epsilon, \delta)$  over  $\ell \times n$  matrices  $\Pi$  is an oblivious subspace embedding for the column space of  $A$  if with probability at least  $1 - \delta$ , for all  $x \in \mathbb{R}^d$ ,

$$(1 - \epsilon)\|Ax\|_2 \leq \|\Pi Ax\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

We use OSNAP matrix for  $\Pi$  from [NN13], where each column has only one non-zero entry. In this case,  $\Pi A$  can be computed in  $\text{nnz}(A) \log(n)$  time, where  $\text{nnz}$  denotes the sparsity of  $A$ .

**Definition 2.** (*OSNAP Matrix.*) Let  $\Pi \in \mathbb{R}^{\ell \times n}$  be a sparse matrix such that for all  $i \in [n]$ , we pick  $j \in [\ell]$  uniformly at random, such that  $\Pi_{i,j} = \pm 1$  uniformly at random and repeat  $\log(n)$  times.

For obtaining a subspace embedding for  $A$ ,  $\Pi$  needs to have  $\ell = d \log(n)/\epsilon^2$  rows. We note that for tables where the number of samples(rows) is much larger than the number of features(columns) the above algorithm can be used to get an accurate representation of the feature space in nearly linear time. However, here we note that  $\Pi$  takes linear combinations of rows of  $A$  and thus does not preserve numeric values. If we then try to join sketched tables, we would obtained augmentation that is inconsistent with the input data. Further, for clustering tasks, taking linear combinations of labels results in a loss of semantic interpretation and it unclear what labels should be assigned to a linear combination of vectors from distinct classes.

An alternative approach here is to sample rows of  $A$  proportional to leverage scores, which requires rescaling the rows to obtain the subspace embedding guarantee.

The leverage score of a given row measures the importance of this row in composing the rowspan. Leverage scores have found numerous applications in regression, preconditioning, linear programming and graph sparsification [Sar06, SS11, LS15, CLM<sup>+</sup>15]. In the special case of Graphs, they are often referred to as *effective resistances*.

**Definition 3.** (*Leverage Scores.*) Given a matrix  $A \in \mathbb{R}^{n \times d}$ , let  $\mathbf{a}_i = A_{i,*}$  be the  $i$ -th row of  $A$ . Then, for all  $i \in [n]$  the  $i$ -th row leverage score of  $A$  is given by

$$\tau_i(A) = \mathbf{a}_i(A^T A)^\dagger \mathbf{a}_i^T$$

Leverage scores can be computed approximately in  $nnz(A)$  time. Sampling  $d \log(d)/\epsilon^2$  rows proportional results in a subspace embedding. However, in our setting the number of features is much larger than the number of samples and naive leverage score computation is intractable. Another drawback of this approach in our setting is we may make some joins invalid as different rows can be scaled by different values. Therefore, an interesting direction here is to efficiently compute data dependent row sampling probabilities that result in a diverse sample.

## 2.2 Joins

In this section we dive into details of table join specifics for ML augmentation task. The main requirement from our join procedure is to preserve all base table rows and hence preserving the original distribution that we try to learn. The objective of the join is to incorporating as much additional information from a foreign table as possible that would improve accuracy on the prediction task. To understand how different approaches to joining tables affect the final result we consider *join types*, *join cardinalities*, *data imputation*, *format resolution*, *Key-match types*.

### 2.2.1 Join type

There four common join types between two tables `INNER JOIN`, `FULL JOIN`, `RIGHT JOIN`, and `LEFT JOIN`. However, most of these types are not suitable for augmentation

task since they either result in introducing irrelevant rows or deleting existing base table rows and hence changing the original learning task. The augmentation workflow requires preserving the base table rows, and thus the only valid operation our join is allowed to perform is the addition of new columns.

1. **INNER JOIN** selects only records that have matching values in both tables. This type of join does not work for data augmentation since it changes the distribution of the base table. For example, assume relation **ACCIDENT** from Figure 1-1 performed an inner join with relation **CAR** on `car_model` attribute such that the relation **CAR** one has one non-zero in each row. In this case, we end up losing all information for the records that have other car models in **ACCIDENT**.
2. **FULL JOIN** selects all records when there is a match in left (base) or right (foreign) table records. This join cannot be used in data augmentation as it introduces irrelevant examples that do not match base table records and do not have target labels.
3. **RIGHT JOIN** selects all records from the right table (foreign table), and matched records from the left table (base table). The result is **NULL** from the left side, when there is no match. This join cannot be used for the same reason as **FULL JOIN**.
4. **LEFT JOIN** selects all records from the left table (base table), and the matched records from the right table (foreign table). The result is **NULL** from the right side, if there is no match. This is the only join type that works for augmentation task since it both preserves every record of the base table and brings only records from the foreign table that match join key values in base table.

### 2.2.2 Key Matches

ARDA handles any type and number of join keys. Join on multiple keys can be specified as a compound key or as a separate join options for the same table. The

latter case implies different options of joining the foreign table and ARDA chooses a join with larger intersection.

In ARDA, we handle joining on **hard keys**, **soft keys**, and a custom combinations of them. Joining on a **hard key** implies joining rows of two tables where the values in column-keys are an exact match. When we join on a **soft key** we do not require an exact match between keys. Instead, we join rows with the column corresponding to the closest value. Further, ARDA allows the user to specify the tolerance in the soft join. We note that in the special case of ARDA receiving keys from the time series data it automatically performs soft join. ARDA has two settings for soft join:

1. *Nearest Neighbour Join*. This type of a join matches base table row with nearest value row in foreign table. If tolerance threshold is specified and nearest neighbour does not satisfy the threshold then **null** values are filled instead.
2. *Two-way nearest neighbour Join*. This join matches one row in base table with two rows from foreign table, such that one row is matched on a key that is less or equal to base table key and another is matched on a key that is larger or equal to base table keys. Then the two rows from the foreign tables are combined into one using linear interpolation on numeric values. This is used to account for how far backward and forwards the keys are from the base table key value. If the values are categorical, they are selected uniformly at random.

Consider a situation when base table has time series data specified in **month/day/year** format, while foreign table has format **month/day/year hr:min:sec**. In order to perform nearest neighbour join one option would be to resolve base table format to midnight timestamp **month/day/year 00:00:00**. However, this would result in joining with only one row from a foreign table that has closest time to a midnight for all the rows in base table for the same day. ARDA identifies differences in time formats and aggregates data over the span of time specified by less precise format. In our scenario all rows that correspond to the same day would be pre-aggregated in foreign table before the join takes place.

### 2.2.3 Join cardinality

There are four types of join cardinality: *one-to-one*, *one-to-many*, *many-to-one*, and *many-to-many*. Both *one-to-one* and *many-to-one* preserve the initial distribution of the base table (training examples) by avoiding changes in number of rows. Recall, a *one-to-one* join matches exactly one key in base table to exactly one key in a foreign table. Similarly, a *many-to-one* join matches many keys in our base table to one row from foreign table, which again suffices.

However, for *one-to-many* and *many-to-many* joins, we would need to match at least one record from base table to multiple records from foreign table. This would require repeating the base table records to accommodate all the matches from a foreign table and introduce redundancy. Since we cannot change the base table distribution, such a join is infeasible.

To illustrate such a situation, assume that the `TRAFFIC ACCIDENT` table from Figure 1-1 has several records for the same type of car. Further, table `CAR SALES` has multiple sales records of the same car model. If we join every matching entry in base table `TRAFFIC ACCIDENT` with every entry in `CAR SALES` the distribution of car models changes in the resulting training matrix and would result in poor generalization. To address the issue with *one-to-many* and *many-to-many* joins we pre-aggregate foreign tables on join keys, thereby effectively reducing to the *one-to-one* and *many-to-one* cases.

### 2.2.4 Imputation

Missing data is a major challenge for modern Big-Data systems. This data can be real, Boolean, or ordinal. Canonical examples of missing data appears in Netflix recommendation system data, health surveys, census questionnaires etc. Common imputation techniques include hot deck, cold deck and mean substitution. The deck methods are heuristics, where missing data is replaced by a randomly selected similar data point. More principled methods assume the data is generated from a simple parametric statistical model. The goal is to then learn the the best fit parameters for

this model.

One canonical model considered for data imputation tasks is the Gaussian copula model<sup>2</sup>. [H<sup>+</sup>07] shows how to model data using Gaussian copula and develop a Bayesian (MCMC) framework to fit the model with incomplete data. Subsequent work explored several other parametric methods for mixed data imputation methods. The down side to such methods is the requirement to make strong distributional assumptions that may not hold in practice. While parametric approaches are widely used in [FLNZ17], non-parametric methods such as [FN19], and iterative imputation methods such as those based on random forests may perform better in practice.

Finally, the low-rank matrix completion literature, solves data imputation by modelling the input as a low-rank matrix. Since the rank objective is highly non-convex, [CR09, CRT06] minimize convex envelope for the rank function. They characterize conditions under which this relaxation works well. However, for handling imputation for mixed data using a low-rank model we run into the challenge of choosing an appropriate loss function, to ensure data of each type is treated correctly.

In ARDA, we implemented a simple imputation technique: we use the median value for numeric data and uniform random sampling is used for categorical values. Since we do not assume anything about the input data, we work with simple approaches that reduce the total running time of the system.

---

<sup>2</sup>[https://en.wikipedia.org/wiki/Copula\\_\(probability\\_theory\)](https://en.wikipedia.org/wiki/Copula_(probability_theory))



# Chapter 3

## Feature Selection

### 3.1 Overview of Feature Selection

Feature selection algorithms can be broadly categorized into filter models, wrapper models and embedded models. The filter model separates feature selection from classifier learning so that the bias of a learning algorithm does not interact with the bias of a feature selection algorithm. Typically, these algorithms rely on general characteristics of the training data such as distance, consistency, dependency, information, and correlation. Examples of such methods include Pearson Correlation Coefficient <sup>1</sup>, Chi-Squared test <sup>2</sup>, Mutual Information <sup>3</sup> and numerous other statistical significance tests. We note that filter methods only look at the input features and do not use any information about the labels, and are thus sensitive to noise and corruption in the features.

The wrapper model uses the predictive accuracy of the learning algorithm to determine the quality of selected features. Wrapper-type feature selection methods are tightly coupled with a specific classifier, such as correlation-based feature selection (CFS) [GWBV02], support vector machine recursive feature elimination (SVM-RFE) [Hal99]. The trained classifier is then used select a subset of features. Popular approaches to this include *forward selection*, *backward elimination* and *recursive fea-*

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Pearson\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Pearson_correlation_coefficient)

<sup>2</sup>[https://en.wikipedia.org/wiki/Chi-squared\\_test](https://en.wikipedia.org/wiki/Chi-squared_test)

<sup>3</sup>[https://en.wikipedia.org/wiki/Mutual\\_information](https://en.wikipedia.org/wiki/Mutual_information)

*ture elimination*, which iteratively add or remove features and compute performance of these subsets on learning tasks. Such methods are likely to get stuck in local minimax [JKP94, YH98, Hal99]. Further, forward selection may ignore weakly correlated features and backward elimination may erroneously remove relevant features due to noise. They often have good performance, but their computational cost is very expensive for large data and training massive non-linear models [LMSZ10, ETPZ09].

Algorithm	Type	Time	Space
Picking Top- $k$	Wrapper	$O(k)$	$O(n + d)$
Repeated Doubling	Wrapper	$O(\log(d)) \times T(n)$	$O(n + d)$ $S(n)$
Forward Selection	Wrapper	$O(n + d) + d \times T(n)$	$O(n + d) + S(n)$
Backward Elimination	Wrapper	$O(n + d) + d \times T(n)$	$O(n + d) + S(n)$
Recursive Feature Elimination	Wrapper	depth $\times$ $d \times T(n)$	$O(n + d) + S(n)$
CFS [GWBV02]	Filter & Wrapper	Large polynomial	Large polynomial
Mutual Info [Eva08]	Filter	$O(nd \log(n))$	$O(n + d)$

Table 3.1: Comparison of popular subset selection algorithms, given a ranking of the features. We note that various learning models provide the ranking for these algorithms.  $T(n)$  denotes the time evaluate the learning algorithm and  $S(n)$  is the corresponding space used.

Given the shortcomings in the two models above, we focus on the embedded model, which includes information about labels, and incorporates the performance of the model on holdout data. Typically, the first step is to obtain a ranking of the features by optimizing a convex loss function [CTG07, LY05]. Popular objective functions include quadratic loss, hinge loss and logistic loss, combined with various regularizers, such as  $\ell_1$  and elastic net. The resulting solution is used to select a subset of features and evaluate their performance. This information is then used to pick a better subset of features.

One popular embedded feature selection algorithm is Relief, which is considered to

Algorithm	Time	Space	Guarantee
Lasso [EHJ <sup>+</sup> 04]	$O(nd^2 + d^3)$	$O(n + d)$	provable
Logistic Regression	$O(ndc)$ per iter $c : \# \text{ classes}$	$O(n + d)$	Heuristic
Linear SVM [Joa06]	$O(n + d)$	$O(n + d)$	provable
Random Forests <sup>4</sup>	$O(n^2 dt)$ $t : \text{no. of trees}$	$O(n^2 dt)$	Heuristic
Relief [KR92]	$O(n+d)$ per iteration	$O(n+d)$ per iteration	Heuristic
I-Relief [Sun07]	Unbounded EM Algo	poly(n)	provable
NDFS [LYL <sup>+</sup> 12]	$O(d^3 + n^2)$ Spectral	$O(n^2 + d^2)$	provable
RUFS [QZ13]	$O(nd + n^2)$ l-BFGS	$O(n + d)$	provable

Table 3.2: Comparison of popular learning models for feature selection.

be effective and efficient in practice [KR92, RŠK03, Sun07]. We describe the algorithm when the labels are binary. Given our data matrix  $X$ , Relief picks a row uniformly at random. Let this row be  $X_{i,*}$ . Let nearHit be the data point from the same label as  $X_{i,*}$  that is closest to  $X_{i,*}$  in Euclidean distance. Let nearMiss belong to the opposite label and defined similarly. The Relief algorithm initializes a weight vector  $W_0$  to be all 0s. It iteratively updates this vector as

$$W_t = W_{t-1} - (X_{i,*} - \text{nearHit})^2 + (X_{i,*} - \text{nearMiss})^2$$

Relief repeats the update  $m$  times and uses the vector  $\frac{1}{m}W_t$  as the ranking for the features. Each iterative step of the algorithm can be implemented in  $O(n + d)$  time. There are no convergence guarantees but it works well in practice.

However, a crucial drawback of Relief is that in the presence of noise, performance degrades severely [Sun07]. Since Relief relies on finding nearest-neighbors in the original feature space, having noisy features can change the objective function and converge to a solution arbitrarily far from the optimal.

While [Sun07] offers an iterative algorithm to fix Relief, this requires running Expectation-Maximization (EM) at each step, which has no convergence guarantees. Further, each step of EM takes time quadratic in the number of data points. Unfortunately, this algorithm quickly becomes computationally infeasible on real-world data sets.

Given that all existing feature selection algorithms that can tolerate noise are either computationally expensive, use prohibitively large space or both, it's not obvious if such methods could be effective in data augmentation scenario when we deal with massive number of spurious features.

### 3.2 Random Injection Based Feature Selection

#### Algorithm 1 : Feature Selection via Random Injection

**Input:** An  $n \times d$  data matrix  $A$ , a threshold  $\tau$  and fraction of random features to inject  $\eta$ , the number of random experiments performed  $k$ .

1. Create  $t = \eta d$  random feature vectors  $n_1, n_2, \dots, n_t \in \mathbb{R}^n$  using Algorithm
2. Append the random features to the input and let the resulting matrix be  $A' = [A \mid N]$
2. Run a feature selection algorithm (see discussion below) on  $A'$  to obtain a ranking vector  $r \in [0, 1]^{d+t}$ .
3. Repeat Steps 1-2  $k$  times.
4. Aggregate the number of times each feature appears in front of all the random features  $n_1, \dots, n_t$  according to the ranking  $r$ . Normalize each value by  $k$ . Let  $r^* \in [0, 1]^d$  be the resulting vector.

**Output:** A subset  $\mathcal{S} \subseteq [d]$  of features such that the corresponding value in  $r^*$  is at least  $\tau$ .

In this section, we describe our random injection based feature selection algorithm,

including the key algorithmic ideas we introduce. Recall, we are given a dataset, where the number of features are significantly larger than the number of samples and most of the features are spurious. Our main task is to find a subset of features that contain signal relevant to our downstream learning task and prune out irrelevant features.

We typically think of the learning task to require training a large, complex model, e.x. Resnet on Imagenet data. We do not assume any prior knowledge or distribution over our data and thus require our algorithm to handle most real world scenarios.

This is a challenging task since bereft of assumptions on the input, any *filter based* feature selection algorithm would not work since it does not take prediction error of a subset of features into account. For instance, consider a set of spurious input features that have high Pearson Correlation Coefficient or Chi-Squared test value. Selecting these features would lead to poor generalization error in our learning task but filter methods do not take this information into account.

Consider instead, the ideal feature selection algorithm : for each subset of the features, train the complex model and select the subset that minimizes the prediction error on the holdout set.

Since the number of subsets grows exponentially in  $d$  and each iteration of training the model is expensive, the ideal feature selector is intractable. We can thus restate our goal as designing an efficient feature selector that takes both input labels and prediction error into account to generate subsets of features such that it does not require re-training the learning model too many times.

To this end, we design a comparison-based feature selection algorithm that circumvents the requirement of testing each subset of features. We do this by injecting carefully constructed random features into our dataset. We use the random features as a baseline to compare the input features with.

We train an ensemble of random forests and linear predictors to compute a joint ranking over the input and injected features. Finally, we use a wrapper method on the resulting ranking to determine a subset of input features contain signal. However, our wrapper method only requires training the complex learning model a constant number of times. We discuss each part in detail below:

### 3.2.1 Random Feature Injection

#### Algorithm 2 : Random Feature Injection Subroutine

**Input:** An  $n \times d$  data matrix  $A$ .

1. Compute the empirical mean of the feature vectors by averaging the column vectors of  $A$ , i.e. let  $\mu = \frac{1}{d} \sum_{i \in [d]} A_{*,i}$ .
2. Compute the empirical covariance  $\Sigma = \frac{1}{d} \sum_{i \in [d]} (A_{*,i} - \mu)(A_{*,i} - \mu)^T$ .
3. We model the distribution of features as  $\mathcal{N}(\mu, \Sigma)$  and generate  $\eta d$  i.i.d. samples from this distribution.

**Output:** A set of  $\eta d$  random vectors that match the empirical mean and covariance of the input dataset.

Given that we make no assumptions on the input data, our implementation should capture the following extreme cases: when most of the input features are relevant for the learning task, we should not prune out too many features. On the other hand, when the input features are mostly uncorrelated with the labels and do not contain any signal, we should aggressively prune out these features. We thus describe a random feature injection strategy that interpolates smoothly between these two corner cases.

We show that in the setting where a majority of the features are good, injecting random features sampled from the standard Normal, Bernoulli, Uniform or Poisson distributions suffices, since our ensemble ranking can easily distinguish between random noise and true features. The precise choice of distribution depends on the input data. The challenging setting is where the features constituting the signal is a small fraction of the input. Here, we use a more aggressive strategy with the goal of generating random features that look a lot like our input. To this end, we fit a statistical feature model that matches the empirical moments of the input data. Statistical modelling has been extremely successful in the context of modern machine learning and moment matching is an efficient technique to do so. We refer the reader to the surveys and references therein [McC02, Dav03, KKN14, SB19].

In particular, we compute the empirical mean  $\mu = \frac{1}{d} \sum_{i \in [d]} A_{*,i}$  and the empirical covariance  $\Sigma = \frac{1}{d} \sum_{i \in [d]} (A_{*,i} - \mu)(A_{*,i} - \mu)^T$  of the input features. Intuitively, we assume that the input data was generated by some complicated probabilistic process that cannot be succinctly describes and match the empirical moments of this probabilistic process. An alternative way to think about our model is to consider  $\mu$  to be a typical feature vector with  $\Sigma$  capturing correlations between the coordinates. We then fit a simple statistical model to our observations, namely  $\mathcal{N}(\mu, \Sigma)$ . Finally, we inject features drawn i.i.d. from  $\mathcal{N}(\mu, \Sigma)$ .

### 3.2.2 Ranking Ensembles

We use a combination of two ranking models, Random Forests and regularized Sparse Regression. Random Forests are models that have large capacity and can capture non-linear relationships in the input. They also typically work well on real world data and our experiments indicate that the inherent randomness helps identifying signal on average. However, since Random Forests have large capacity, as we increase the depth of the trees, the model may suffer from over-fitting. Additionally, Random Forests do not have provable guarantees on running time and accuracy. We use an off-the-shelf implementation of Random Forests and tune hyper-parameters appropriately.

We use  $\ell_{2,1}$ -norm minimization as a convex relation of sparsity. The  $\ell_{2,1}$ -norm of a matrix sums the absolute values of the  $\ell_2$ -norms of its rows. Intuitively, summing the absolute values can be thought of as a convex relaxation of minimizing sparsity. Such a relaxation appears in Sparse Recovery, Compressed Sensing and Matrix Completion problems [CRT06, CR09, CP10, MMN<sup>+</sup>18] and references therein. The  $\ell_{2,1}$ -norm minimization objective has been considered before in the context of feature selection [Sto10, XWH12, QZ13, MLM<sup>+</sup>16, LGMX18]. Formally, let  $X$  be our input data matrix and  $Y$  is our label matrix. We consider the following regularized loss function :

$$\mathcal{L}(W) = \min_{W \in \mathbb{R}^{c \times d}} \|WX - Y\|_{2,1} + \gamma \|W^T\|_{2,1} \quad (3.1)$$

While this loss function is convex (since it is a linear combination of norms), we note

that both terms in the loss function are not smooth functions since  $\ell_1$  norms are not differentiable around 0. Apriori it is unclear how to minimize this objective without using Ellipsoid methods, which are computationally expensive (large polynomial running time). A long line of work studies efficient algorithms to optimize the above objective and use the state-of-the-art efficient gradient based solver from [QZ13] to optimize the loss function in Equation 3.1. However, this objective does not capture non-linear relationships among the feature vectors. We show that a combination of the two approaches works well on real world datasets.

Note, in the case of Regression, the aforementioned loss is exactly the LASSO objective. Additionally, for classification tasks we consider setting where the labels of our dataset may also be corrupted. Here, we use a modified objective from [QZ13], where the labels are included as variables in the loss function. The modified loss function fits a consistent labelling that minimizes the  $\ell_{2,1}$ -norm from Equation 3.1. We observe that on certain datasets, the modified loss function discovers better features.

**Algorithm 3 : Repeated Doubling and Binary Search**

**Input:** An  $n \times d$  data matrix  $A$ , a ranking of features  $r \in [0, 1]^d$ .

1. For each  $i \in [\log(d)]$ :
  - (a) Let  $\mathcal{T}_i \subseteq [d]$  be the set of top  $2^i$  features according to the ranking  $r$ .  
Let  $A_{\mathcal{T}_i}$  be the dataset restricted to the set of features indexed by  $\mathcal{T}_i$ .
  - (b) Train the learning model on  $A_{\mathcal{T}_i}$ , i.e. the subset of features indexed by  $\mathcal{T}_i$  and test accuracy on the holdout set.
  - (c) If the accuracy decreases, let  $r = 2^i$  and  $l = 2^{i-1}$ .
2. Binary search over test accuracy in the range  $l$  to  $r$ . Let  $\mathcal{S}$  be the resulting subset with highest accuracy.

**Output:** A subset of features indexed by set  $\mathcal{S}$ .

### 3.2.3 Aggregate Ranking

We use a straightforward re-weighting strategy to combine the rankings obtained from Random Forests (RF) and Sparse Regression (SR). Given the aforementioned rankings, we compute an aggregate ranking parameterized by  $\nu \in [0, 1]$  such that we scale the RF rankings by  $\nu$  and the SR ranking by  $(1 - \nu)$ . Given an aggregate ranking, one natural way to do feature selection is repeated doubling (Algorithm 3). Here, in the  $i$ -th iteration, we select the top  $2^i$  features, train the model and compute the prediction error on the holdout set. We experimentally observe that even the aggregate rankings are not monotone in prediction error. If instead, we perform a linear search over the ranking, we end up training our model  $n$  times, which may be prohibitively expensive (this strategy is also known as forward selection).

We therefore inject random features into our dataset and compute an aggregate ranking on the joint input. We repeat this experiment  $t$  times, using fresh random features in each iteration. We then compute how often each input feature appears in front of all the injected random features in the ordering obtained by the aggregate ranking. Intuitively, we expect that features that contain signal are ranked ahead of the random features on average. Further, input features that are consistently ranked worse than the injected random features are either very weakly correlated or systematic noise.

### 3.2.4 Putting it together.

Our experimental setup is amenable to user input and a user can customize the range of parameters we search over. For our experiments, we inject 20% random features drawn i.i.d from standard Normal, Bernoulli, Poisson, Uniform or Algorithm 2. We repeat this process  $t$  times (in our experiments  $t = 10$ ) and in each iteration we train a Random Forest model as well as a Sparse Regression model to obtain two distinct rankings. We then aggregate these rankings by taking a  $\nu = 0.5$  combination. For each feature, we compute the fraction of times it appears in front of all random features. We then discard all features less than  $\tau$ , for  $\tau \in \mathbb{T}$ . We only increase

the threshold as long as the performance of the resulting features on the holdout set increases monotonically.

**Algorithm 4 : Wrapper Algorithm**

**Input:** An  $n \times d$  data matrix  $A$ , a set of thresholds  $T$

1. For each  $\tau \in T$  in increasing order:
  - (a) For each group run Algorithm 1 with  $A$  and  $\tau$  as input.
  - (b) Let  $\mathcal{S} \subseteq [d]$  denote the indices of the features selected by Algorithm 1.
  - (c) Train the learning model on  $A_{\mathcal{S}}$ , i.e. the subset of features indexed by  $\mathcal{S}$  and test accuracy on the holdout set.
  - (d) If the accuracy is monotone, proceed to the new threshold. Else, output the previous subset.

**Output:** A subset of features indexed by set  $\mathcal{S}$ .

### 3.3 $L_{2,1}$ -Norm Minimization and Sketching

In this section, we discuss the state of the art algorithms for  $\ell_{2,1}$ -norm minimization and show that we can use techniques from matrix sketching to obtain a faster algorithm. Two popular algorithms for  $\ell_{2,1}$ -norm minimization are Unsupervised Discriminative Feature Selection (UDFS) [YSM<sup>+</sup>11] and Nonnegative Discriminative Feature Selection (NDFS) [LYL<sup>+</sup>12]. UDFS aims to select the most discriminative features for data representation, accounting for manifold structure. NDFS performs non-negative spectral analysis. Both these algorithms are not designed to be robust to feature or label noise.

Further, the computational complexity of both these algorithms is  $\Omega(d^3 + n^2)$  and the space complexity is  $\Omega(n^2 + d^2)$ . The technical details for why they do not perform well under noise can be found in [QZ13]. However, [QZ13] propose a robust

unsupervised feature selection algorithm (RUFS) that performs robust clustering and robust selection simultaneously. The loss function for RUFS also minimizes  $\ell_{2,1}$  norm, however [QZ13] present an iterative algorithm that can be solved using projected BFGS methods [LN89]. The computational complexity of RUFS is  $O(nd+n^2)$  and the space complexity is  $O(n+d)$ . Therefore, it performs better than previous algorithms when  $d > n$  and uses much less space.

We use sketching techniques to improve this running time to be proportional to the number of non-zeros in our input. For sparse datasets, this would be significantly faster, while obtaining a provably close solution. We note that while this technique is of independent interest in feature selection, it does not apply to our setting, since we assume the number of features  $d$  to be much larger than the number of data points  $n$ .

**Algorithm 5 : Iterative Algorithm for  $\ell_{2,1}$ -Norm Minimization**

**Input:** An  $n \times d$  data matrix  $A = [X^T \ \gamma I]$ , variable matrix  $U = [W \ E]^T$ , Lagrangian Variables  $\Lambda$ , number of iteration  $T$ .

1. Let  $D_0 = I$ . For  $t \in [T]$ ,
  - (a)  $U_{t+1} = D_t^{-1}(AD_t^{-1}A^T)^{-1}Y$
  - (b)  $D_{t+1} = \text{diag}\left(\frac{1}{2\|(U_{t+1})_i\|_2}\right)$

**Output:** Resulting matrix  $U_T$ .

We begin by describing the details of the loss functions considered in [NHCD10, QZ13], the regularizers used and the iterative algorithms presented to solve these objectives. Refer to the papers for further details. Recall,  $X$  is our data matrix and  $Y$  is our label matrix. The loss function considered here is Equation 3.1. Let  $E = WX - Y$ . Via standard reformulation techniques, Equation 3.1 can be rewritten

as

$$\min_U ||U||_{2,1} - \text{Tr}(\Lambda(AU - Y)) \quad (3.2)$$

where  $U = [W \ E]^T$ ,  $A = [X^T \ \gamma I]$  and  $\Lambda$  are Lagrangian variables. To solve Equation 2, [NHCD10] present the following iterative algorithm:

The above algorithm is guaranteed to converge to the global optimum for Equation 3.2. The unsupervised robust feature selection algorithm builds on the aforementioned algorithm and is quite involved. Therefore, we refer the reader to [QZ13].

**Robust Sketching.** The quadratic dependence on the number of samples for the running time of the algorithms described above is the main bottleneck to make robust feature selection algorithms scale. Ideally, we would like to obtain algorithms with run time independent of the number of training examples ( $n$ ), such that cost of the objective is approximately preserved. Here, we use the general framework of sketching, [CW13, NN13, CW15, Woo14] etc., to reduce the number of training examples we work with. Observe, this is not a dimension reduction technique since the matrix we optimize over is  $c \times d$ , and we have pay  $\Omega(d)$  time to simply write the output.

Intuitively, one might hope for subspace embeddings (Definition 1) for  $\ell_{p,q}$  norms, which would reduce the size of our optimization problem in Equation 3.1 and preserve all vectors in the relevant subspace. However, such a guarantee is not known. Instead, our main tool is a result of Clarkson and Woodruff [CW15], for robust subspace approximation. This obtains a weaker guarantee than a subspace embedding, but suffices for our setting. Our key observation is that the central result of Clarkson and Woodruff though stated in full generality for M-estimators applies  $\ell_{p,q}$  norms as well.

**Theorem 1.** (*Lopsided Sketches [CW15].*) *Let  $S \in \mathbb{R}^{n \times \ell}$  be a sparse embedding matrix (as defined above) such that  $\ell = O(d^2/\epsilon^2)$ . Let  $X \in \mathbb{R}^{d \times n}$  be the data matrix and  $Y \in \mathbb{R}^{c \times n}$  be the label matrix. Then, with probability at least 99/100,*

1. *For all  $W$ ,*

$$\|WXS - YS\|_{p,q} \geq (1 - \epsilon)\|WX - Y\|_{p,q}$$

2. Let  $W^* = \arg \min_W \|WX - Y\|_{p,q}$ . Then,

$$\|W^*XS - YS\|_{p,q} \leq (1 + \epsilon)\|W^*X - Y\|_{p,q}$$

We note that we get the property that the sketch is non-contracting for all vectors in subspace, but is non-expanding for only the minimizer. An immediate corollary of the above theorem is that we can minimize the sketched problem such that the cost is approximately preserved (exactly as we saw for regression). We precompute  $XS$  and  $YS$ . We then solve the following optimization problem:

$$\mathcal{L}'(W) = \min_{W \in \mathbb{R}^{c \times d}} \|WXS - YS\|_{2,1} + \gamma \|W^T\|_{2,1} \quad (3.3)$$

We use the resulting matrix  $W^*$  as our candidate solution. From Theorem 1, it follows that  $W^*$  has cost at most  $(1 + \epsilon)\text{OPT}$ , where  $\text{OPT}$  is the minimum objective value for Equation 3.1. We observe that the overall running time is  $O\left(\text{nnz}(X) + \frac{d^2}{\epsilon}\right)$ . A compelling open question here is to obtain linear or near-linear dependence on  $d$ .



# Chapter 4

## Experiments

In this chapter, we describe our extensive experimental evaluation. We begin with micro benchmarks to show the effectiveness of our feature selection algorithm on standard ML datasets. Next, we evaluate each component of our system, including various coreset constructions, joining algorithms, hyper-parameter search for learning models, feature selection algorithms and quality of data augmentation on real-world datasets.

### 4.1 Micro Benchmarks

In this section, we construct synthetic data to test the performance of feature selection algorithms. We do this by starting with standard ML data sets (such as Kraken, Digits and Flights) and appending artificial features. The artificial features we append are random, uncorrelated noise vectors sampled from standard distributions such as uniform, Gaussian, and Bernoulli with randomly initialized parameters for these distributions.

We consider a setting where the number of noisy, uncorrelated features are overwhelming large. In particular, we the number of noise features we append is  $1000\times$  more than the number of true features. We compute the number of true and noisy features recovered by each feature selection algorithm. Recall, our feature selection algorithm consists of a learning model used to obtain a ranking and a subset selection

such as forward or backward selection. In our plots, we use the following feature selection methods: *Forward Selection*, *Backward Selection*, *Recursive Feature Elimination*, *Random Forests*, *Sparse Regression*, *Mutual Information*, *Logistic Regression*, *Lasso*, *Relief*, *Linear SVM*, *RIFS*.

Methods such as Random Forest, Sparse Regression, Mutual Information, Logistic Regression, Lasso, Relief, and Linear SVM return ranking that we use to select features using repetitive doubling and binary search algorithm. Forward Selection, Backward Selection, and Recursive Feature elimination (RFE) use Random Forest ranker. RIFS method is describe in Chapter 3.

Our choice is based on Random Forests performing consistent performing well for various subset selection heuristics. For comparing ranking algorithms consistently, we use repeated doubling and binary search, as this accurately captures the monotonicity properties of the ranking and performed the best on our data sets. Recall, we discussed these algorithms in more detail in Chapter 3. We plot the resulting data in Figure 4-2.

**Description of Datasets.** Our Micro benchmarks use Kraken, Digits and Flights datasets as base tables :

1. *Kraken* : This is a classification dataset consisting of anonymized sensors and usage statistics from the Kraken supercomputer at the University of Tennessee, which we received via personal communication. The target represents machine failure within a 24-hour window. The labels are binary, with 568 samples being 0 and 432 being 1.
2. *Digits* : This is a standard multi-label classification data set with roughly 180 samples for each digit between 0 and 9. It ships with Sklearn<sup>1</sup>.
3. *Flights*: This is a regression dataset consisting of weather data and flight departure/arrival data spanning from 2005-2010. It appears in various Kaggle sources, and the target is to predict how long a flight is delayed.

---

<sup>1</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\\_digits.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html)

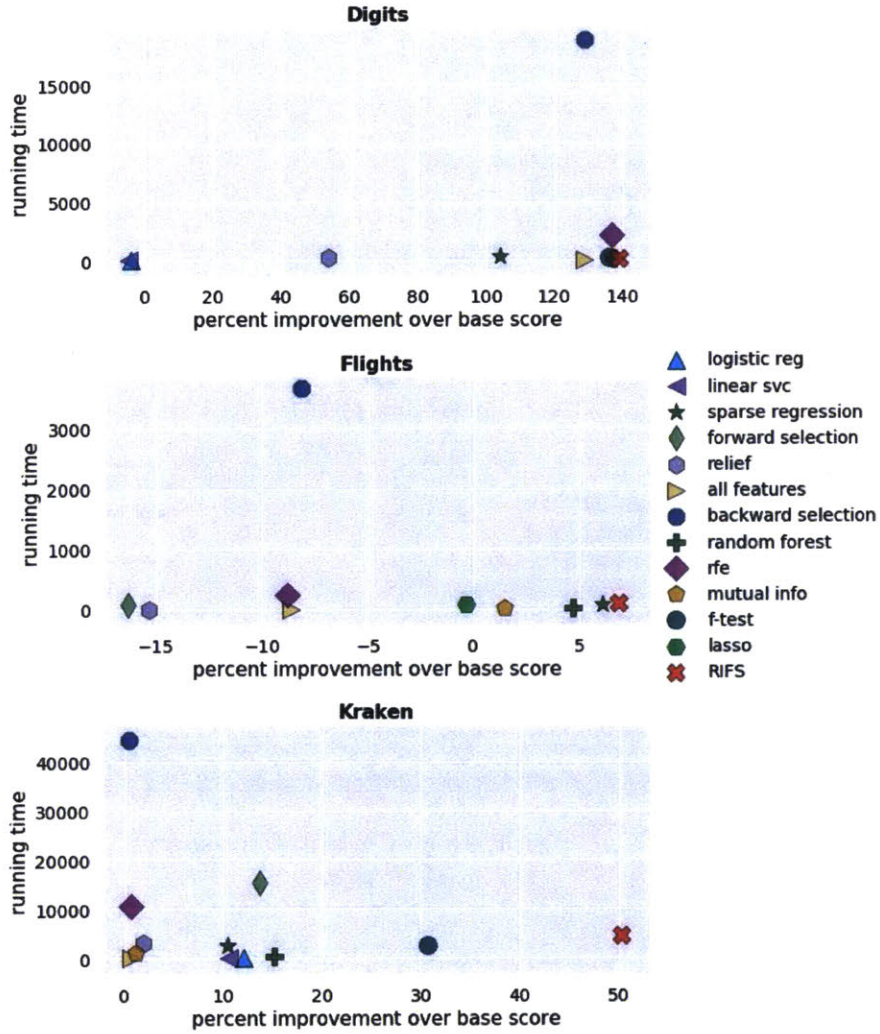


Figure 4-1: Comparison of the running time for each feature selection algorithm with the improvement in prediction accuracy achieved over the base table (no augmentation).

## 4.2 Coreset Comparison

In this section, we compare the performance of constructing coresets via Uniform sampling and Stratified sampling on the Digits, School and Flights data sets. We use benchmark numerous feature selection algorithms and compare performance with our Random Injection based Feature Selection (RIFS) algorithm. For most feature selection algorithms, Uniform sampling performs better than Stratified sampling on the School data set, whereas the opposite holds for the Digits data set. Therefore,

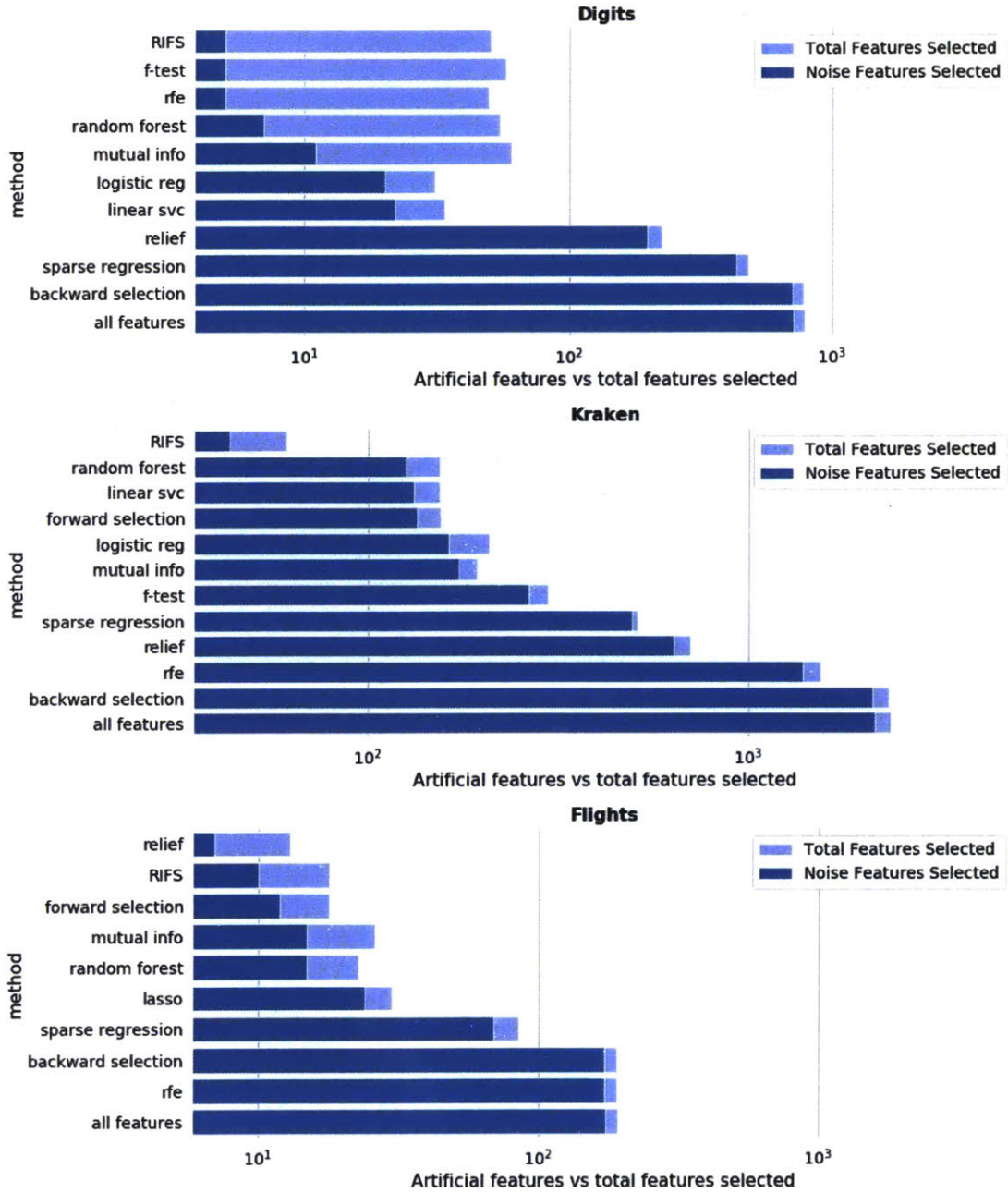


Figure 4-2: Micro Benchmark for Feature Selection Algorithms on Digits, Kraken and Flights data sets. The original data sets are appended with  $1000\times$  artificial, noisy features (as described above). The  $x$ -axis represents number of features, the  $y$ -axis denotes the algorithm and the plot compares the true features extracted vs the artificial features extracted.

we observe there is no one approach that always performs better, and the coreset performance is data-dependent.

The School data set consists of 33 features, and we append and 649 samples in the training set. The data set has numerical values and binary labels with 731 being true and 1043 being false. We create a coreset of size 300 rows via Uniform or Stratified sampling and plot the results in Figure 4-3. We do the same for the Figures data set and sample 1000 rows for the Kraken dataset.

## 4.3 Experimental Evaluation of ARDA

**Real World Datasets.** We pick our base tables to be data sets provided by the DARPA D3M<sup>2</sup> competition. We use the DataMart/Auctus framework to search over new tables to augment. We briefly describe the data sets we use for our base tables:

1. *Taxi and Vehicle Collisions*: This regression dataset contains information about vehicle collision in NYC. The data is available through NYC Open Data<sup>3</sup>, which can be retrieved using the Socrata API<sup>4</sup>. It has total of 30 joined tables.
2. *Pickup*: This is a regression taxi demand dataset that contains hourly passenger pickup numbers from LGA airport by Yellow cabs between Jan 2018 to June 2018. The target prediction is the number of passenger pickups for a given hour. It has total of 23 joined tables.
3. *Poverty*: This regression dataset consists of socio-economic indicators like poverty rates, population change, unemployment rates, and education levels vary geographically across U.S. States and counties. It has total of 40 joined tables.
4. *School-s*: This is a classification dataset from the DataMart API using exact match as the join operation. The target prediction is the performance of each school on a standardized test based on student attributes. It has total of 12 joined tables.

---

<sup>2</sup><https://www.darpa.mil/program/data-driven-discovery-of-models>

<sup>3</sup><https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95>

<sup>4</sup><https://dev.socrata.com/>

	$\times$ Times Faster	Score Improvement
Sparse Regression	3.1	5.65%
RIFS	12.0	1.90 %
Relief	2.5	5.50 %
Forward Selection	1.3	4.08 %
Backward Selection	1.2	0.44 %
RFE	3.8	3.63 %

Table 4.1: We compare the performance of joining one table at a time vs joining at most  $|S|$  features at a time ( $|S|$  is the size of a coreset). The first column denotes the multiplicative speedup factor and the second column denotes increase in accuracy w.r.t. full materialization.

Method	$\times$ Running Time	Score Improvement
Sparse Regression	0.18	5.57%
RIFS	13.92	3.88%
Relief	0.24	4.55%
Forward Selection	2.70	-3.85%
Backward Selection	9.95	-1.31%

Table 4.2: We compare the performance of joining all tables at once (full materialization) vs joining at most  $|S|$  features at a time ( $|S|$  is the size of a coreset). The first column denotes the multiplicative speedup factor and the second column denotes increase in accuracy w.r.t. full materialization.

5. *School-l*: This is a classification dataset which is created from the same base table as school-s, except with the join operation being any possible match on any column. It has total of 352 joined tables.

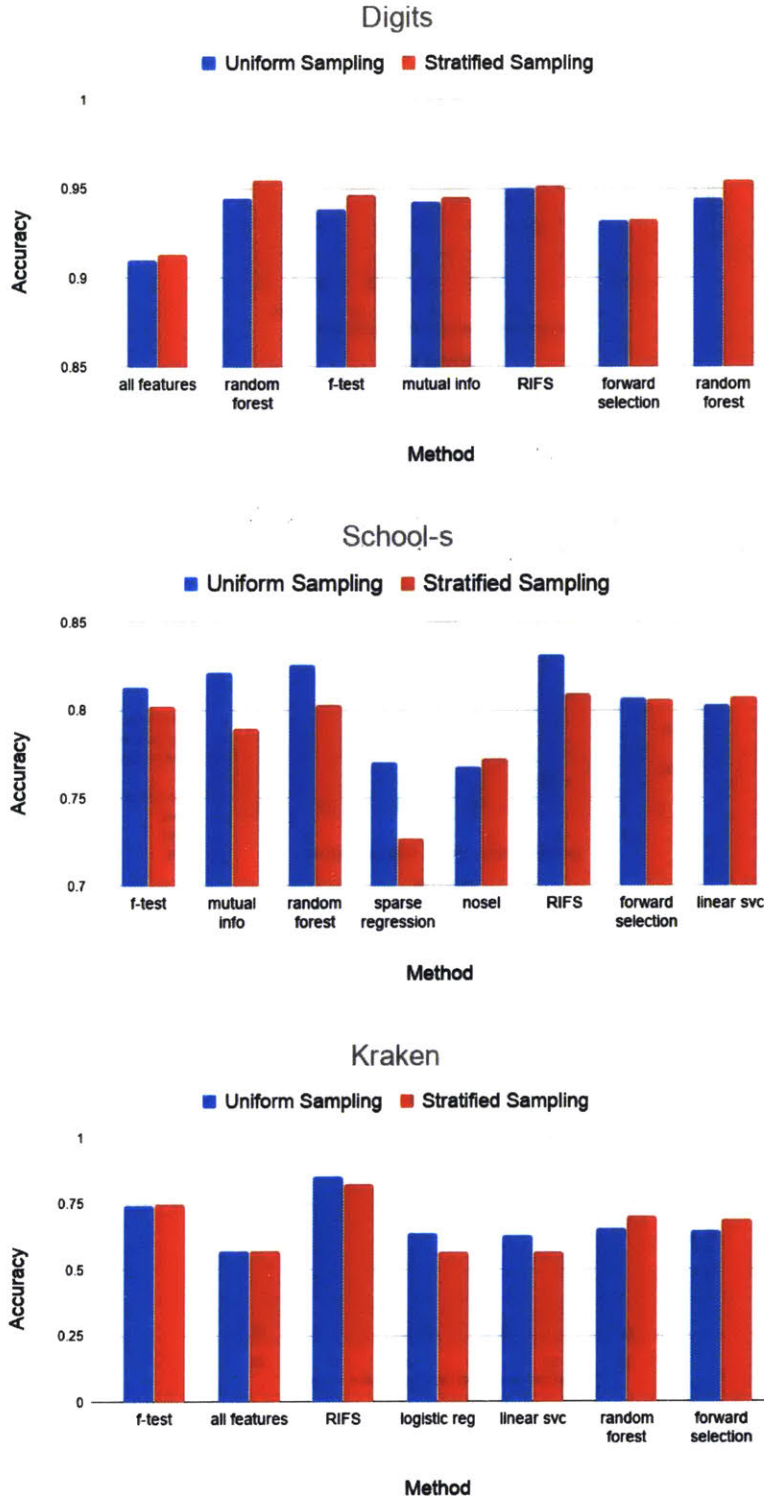


Figure 4-3: Coreset comparison using various feature selection algorithm on School, Digits and Kraken data sets. The coresets are construction via Uniform or Stratified sampling 1000 training samples. The  $x$ -axis denotes the feature selection algorithms and the  $y$ -axis denotes the corresponding prediction accuracy.

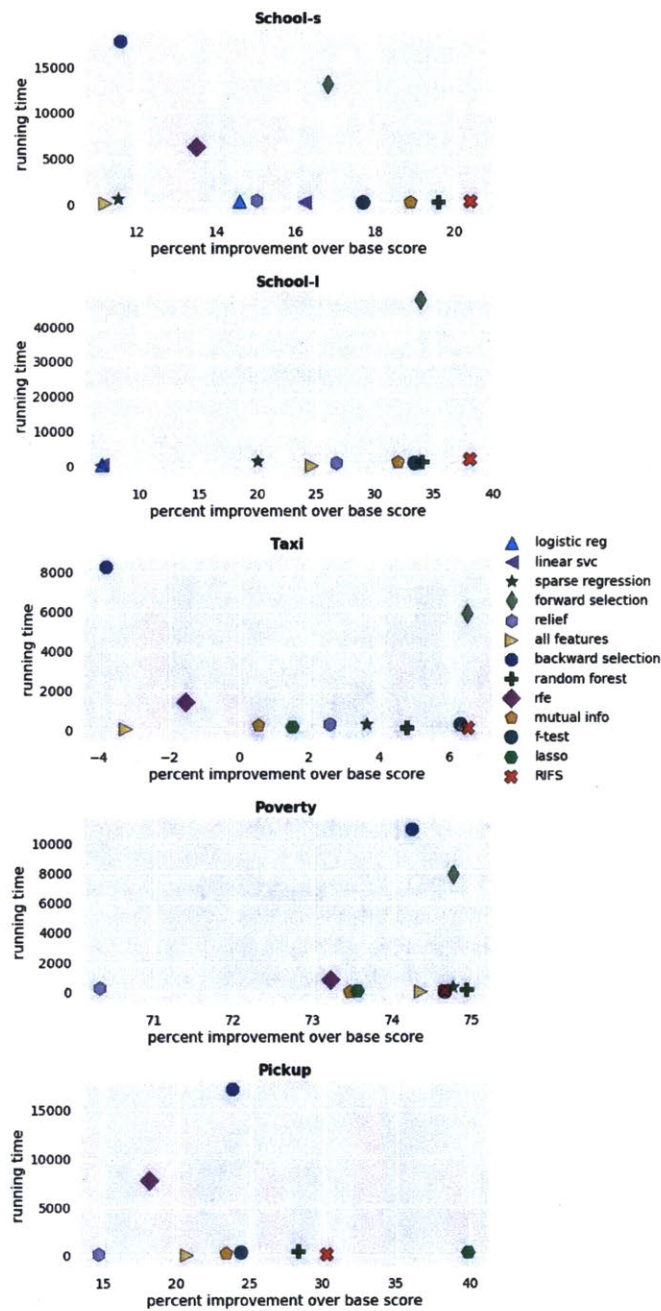


Figure 4-4: Comparison of the running time for each feature selection algorithm with the improvement in prediction accuracy achieved over the base table (no augmentation).

# Bibliography

- [AEIK17] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.
- [BBC<sup>+</sup>14] Anant Bhardwaj, Souvik Bhattacharjee, Amit Chavan, Amol Deshpande, Aaron J Elmore, Samuel Madden, and Aditya G Parameswaran. Datahub: Collaborative data science & dataset version management at scale. *arXiv preprint arXiv:1409.0798*, 2014.
- [BCH<sup>+</sup>15] Souvik Bhattacharjee, Amit Chavan, Silu Huang, Amol Deshpande, and Aditya Parameswaran. Principles of dataset versioning: Exploring the recreation/storage tradeoff. *Proceedings of the VLDB Endowment*, 8(12):1346–1357, 2015.
- [BDE<sup>+</sup>15] Anant Bhardwaj, Amol Deshpande, Aaron J Elmore, David Karger, Sam Madden, Aditya Parameswaran, Harihar Subramanyam, Eugene Wu, and Rebecca Zhang. Collaborative data analytics with datahub. *Proceedings of the VLDB Endowment*, 8(12):1916–1919, 2015.
- [Ber06] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.
- [CFDM<sup>+</sup>17] Raul Castro Fernandez, Dong Deng, Essam Mansour, Abdulhakim A Qahtan, Wenbo Tao, Ziawasch Abedjan, Ahmed Elmagarmid, Ihab F Ilyas, Samuel Madden, Mourad Ouzzani, et al. A demo of the data civilizer system. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1639–1642. ACM, 2017.
- [CHK09] Michael J Cafarella, Alon Halevy, and Nodira Khoussainova. Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101, 2009.
- [CLM<sup>+</sup>15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.
- [CP10] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

- [CR09] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.
- [CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.
- [CTG07] Gavin C Cawley, Nicola L Talbot, and Mark Girolami. Sparse multinomial logistic regression via bayesian l1 regularisation. In *Advances in neural information processing systems*, pages 209–216, 2007.
- [CW13] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.
- [CW15] Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329. IEEE, 2015.
- [CW17] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.
- [Dav03] Anthony Christopher Davison. *Statistical models*, volume 11. Cambridge University Press, 2003.
- [Den18] Li Deng. Table2vec: Neural word and entity embeddings for table population and retrieval. Master’s thesis, University of Stavanger, Norway, 2018.
- [Dom99] Pedro Domingos. The role of occam’s razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999.
- [DVDHSM06] A Rogier T Donders, Geert JMG Van Der Heijden, Theo Stijnen, and Karel GM Moons. A gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–1091, 2006.
- [EHJ<sup>+</sup>04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.
- [ETPZ09] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, 2009.

- [Eva08] Dafydd Evans. A computationally efficient estimator for mutual information. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 464(2093):1203–1215, 2008.
- [FAK<sup>+</sup>18] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE, 2018.
- [FLNZ17] Jianqing Fan, Han Liu, Yang Ning, and Hui Zou. High dimensional semiparametric latent graphical model for mixed data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(2):405–421, 2017.
- [FN19] Huijie Feng and Yang Ning. High-dimensional mixed graphical model with ordinal data: Parameter estimation and statistical inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 654–663, 2019.
- [Fri97] Jerome H Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.
- [GB06] Wolfgang Gatterbauer and Paul Bohunsky. Table extraction using spatial reasoning on the css2 visual box model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1313. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
- [GBGB09] Jerzy W Grzymala-Busse and Witold J Grzymala-Busse. Handling missing attribute values. In *Data mining and knowledge discovery handbook*, pages 33–51. Springer, 2009.
- [GHJ<sup>+</sup>10] Hector Gonzalez, Alon Halevy, Christian S Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, and Warren Shen. Google fusion tables: data management, integration and collaboration in the cloud. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 175–180. ACM, 2010.
- [GLH15] Salvador García, Julián Luengo, and Francisco Herrera. Dealing with missing values. In *Data preprocessing in data mining*, pages 59–105. Springer, 2015.
- [GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

- [H<sup>+</sup>07] Peter D Hoff et al. Extending the rank likelihood for semiparametric copula estimation. *The Annals of Applied Statistics*, 1(1):265–283, 2007.
- [Hal99] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.
- [Hal13] Alon Y Halevy. Data publishing and sharing using fusion tables. In *CIDR*, 2013.
- [HKN<sup>+</sup>16a] Alon Halevy, Flip Korn, Natalya F Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. Goods: Organizing google’s datasets. In *Proceedings of the 2016 International Conference on Management of Data*, pages 795–806. ACM, 2016.
- [HKN<sup>+</sup>16b] Alon Y Halevy, Flip Korn, Natalya Fridman Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. Managing google’s data lake: an overview of the goods system. *IEEE Data Eng. Bull.*, 39(3):5–14, 2016.
- [JKP94] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994.
- [Joa06] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.
- [KJ97] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.
- [KKN14] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*, pages 372–378. IEEE, 2014.
- [KM06] Lukasz A Kurgan and Petr Musilek. A survey of knowledge discovery and data mining process models. *The Knowledge Engineering Review*, 21(1):1–24, 2006.
- [KNPZ16] Arun Kumar, Jeffrey Naughton, Jignesh M Patel, and Xiaojin Zhu. To join or not to join?: Thinking twice about joins before feature selection. In *Proceedings of the 2016 International Conference on Management of Data*, pages 19–34. ACM, 2016.
- [KR92] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.

- [LG17] Yuzhe Liu and Vanathi Gopalakrishnan. An overview and evaluation of recent machine learning imputation methods using cardiac imaging data. *Data*, 2(1):8, 2017.
- [LGMX18] Beiyi Liu, Guan Gui, Shinya Matsushita, and Li Xu. Dimension-reduced direction-of-arrival estimation based on l2/l1-norm penalty. *IEEE Access*, 6:44433–44444, 2018.
- [LL06] Daniel T Larose and Daniel T Larose. *Data mining methods and models*, volume 12. Wiley Online Library, 2006.
- [LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.
- [LMSZ10] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In *Feature Selection in Data Mining*, pages 4–13, 2010.
- [LN89] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.
- [LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 230–249. IEEE, 2015.
- [LY05] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge & Data Engineering*, (4):491–502, 2005.
- [LYL<sup>+</sup>12] Zechao Li, Yi Yang, Jing Liu, Xiaofang Zhou, and Hanqing Lu. Un-supervised feature selection using nonnegative spectral analysis. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [McC02] Peter McCullagh. What is a statistical model? *Annals of statistics*, pages 1225–1267, 2002.
- [ME08] Faouzi Mhamdi and Mourad Elloumi. A new survey on knowledge discovery and data mining. In *2008 Second International Conference on Research Challenges in Information Science*, pages 427–432. IEEE, 2008.
- [MLM<sup>+</sup>16] Yong Ma, Chang Li, Xiaoguang Mei, Chengyin Liu, and Jiayi Ma. Robust sparse hyperspectral unmixing with l2/l1 norm. *IEEE Transactions on Geoscience and Remote Sensing*, 55(3):1227–1239, 2016.

- [MMN<sup>+</sup>18] Elaine Crespo Marques, Nilson Maciel, Lirida Naviner, Hao Cai, and Jun Yang. A review of sparse recovery algorithms. *IEEE Access*, 7:1300–1322, 2018.
- [MMS<sup>+</sup>17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [MSC<sup>+</sup>13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [MWYJ02] Carol M Musil, Camille B Warner, Piyanee Klainin Yobas, and Susan L Jones. A comparison of imputation techniques for handling missing data. *Western Journal of Nursing Research*, 24(7):815–829, 2002.
- [NHCD10] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010.
- [NN13] Jelani Nelson and Huy L Nguyễn. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2013.
- [Phi16] Jeff M Phillips. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.
- [PHW16] Maria Pampaka, Graeme Hutcheson, and Julian Williams. Handling missing data: analysis of a challenging data set using multiple imputation. *International Journal of Research & Method in Education*, 39(1):19–37, 2016.
- [QZ13] Mingjie Qian and Chengxiang Zhai. Robust unsupervised feature selection. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.
- [RŠK03] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.
- [Sar06] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.
- [SB19] Xudong Sun and Bernd Bischl. Tutorial and survey on probabilistic graphical model and variational inference in deep reinforcement learning. *arXiv preprint arXiv:1908.09381*, 2019.

- [SKZ17] Vraj Shah, Arun Kumar, and Xiaojin Zhu. Are key-foreign key joins safe to avoid when learning high-capacity classifiers? *Proceedings of the VLDB Endowment*, 11(3):366–379, 2017.
- [SPC<sup>+</sup>19] Miriam Seoane Santos, Ricardo Cardoso Pereira, Adriana Fonseca Costa, Jastin Pompeu Soares, João Santos, and Pedro Henriques Abreu. Generating synthetic missing data: A review by missing mechanism. *IEEE Access*, 7:11651–11667, 2019.
- [SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.
- [Sto10] Mihailo Stojnic. L2/l1-optimization in block-sparse compressed sensing and its strong thresholds. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):350–357, 2010.
- [Sun07] Yijun Sun. Iterative relief for feature weighting: algorithms, theories, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1035–1051, 2007.
- [SZS<sup>+</sup>13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [TSRC15] Ignacio G Terrizzano, Peter M Schwarz, Mary Roth, and John E Colino. Data wrangling: The challenging journey from the wild to the lake. In *CIDR*, 2015.
- [WK17] Eric Wong and J Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.
- [Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.
- [XWH12] Yunhai Xiao, Soon-Yi Wu, and Bing-Sheng He. A proximal alternating direction method for l2/l1 norm least squares problem in multi-task feature learning. *Journal of Industrial and Management Optimization*, 8(4):1057, 2012.
- [YGCC12] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 97–108. ACM, 2012.

- [YH98] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.
- [YSM<sup>+</sup>11] Yi Yang, Heng Tao Shen, Zhigang Ma, Zi Huang, and Xiaofang Zhou. L2, 1-norm regularized discriminative feature selection for unsupervised. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [YWL<sup>+</sup>16] Chen Ye, Hongzhi Wang, Jianzhong Li, Hong Gao, and Siyao Cheng. Crowdsourcing-enhanced missing values imputation based on bayesian network. In *International Conference on Database Systems for Advanced Applications*, pages 67–81. Springer, 2016.