Systems Pharmacology – Machine Learning Approaches in Profiling Oncology Drug Candidates

by

ML Ujwal

Ph.D. Biochemistry/Informatics

Indian Institute of Science (IISc), Bangalore/University of Texas, Austin

B.S. Math/Chemistry, New Science College, Hyderabad

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

May 2021

Signature of Author_____

System Design & Management
February 21, 2021

Certified by_____

Bryan R. Moser PhD
Academic Director and Senior Lecturer, System Design & Management
Thesis Supervisor

Certified by_____

Weber-Shaughness Professor of Mechanical Engineering, Prof. Warren Seering PhD
Thesis Supervisor

Accepted by_____

Joan Rubin
Executive Director, System Design & Management Program

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I wish to express my sincere gratitude to Dr. Bryan Moser and Prof. Warren Seering for being my thesis advisors. Besides, I extend my gratitude to Pat Hale and the rest of the core faculty, Prof. Oliver DeWeck, Dr. Bruce Cammeron, Dr. Bryan Moser, and late John Helferich. I am humbled by their inspiration, generosity, kindness, and consideration. Joan, I can never adequately thank you, your empathy and humanity sustained me through some rough patches pursuing this mission. I owe my debt to Dr. Kaushik Sinha for his participation in the preparation of this thesis. I enjoyed his intellectual camaraderie. I have great admiration for his extraordinary depth and humility.

I am grateful to Dr. Stan Finkelstein, Prof. Anthony Sinskey, Prof Charles Cooney at MIT for giving me an opportunity to help run their Principles and Practices in Drug Discovery course. I am equally grateful to Zen Chu and Dr. Maulik Mazmudar for the opportunity to organize and run the Healthcare Ventures course at MIT Sloan.

I have had enormous help and support from several people who diligently and painstakingly read my drafts. In particular, I mention Prof. Ken Marx and Prof. Valeri Barsegov, whose experience and expertise in computational chemistry and modeling enormously enriched my thesis. I recollect with warmth and admiration my colleagues in drug discovery in the industry, they deserve my gratitude and appreciation for their insights, suggestions, and some good and timely gotchas. Thanks indeed to Lalit, Sanjeeva, Elvis, and Oljora.

MIT had been an extraordinary experience in more ways than one. Here I met some extraordinarily bright people, exchanged ideas, and forged some everlasting friendships. I will always cherish friendships with Jae, Elvis, Hojin, Huiling, Dr. T, Nam, Subhanker, Burak, Erdem, Vikas, Ashley, and Jolly with warmth and respect.

I gratefully acknowledge the help and support from the following: Dr. Hongming Chen, AstraZeneca, Sweden for access to data; Prof. Todeschini's group at the University of Milano-Bicocca, Italy and Alberto Manganaro, Kode solutions; Dr. Steven Worthington, Research Computing Group, Institute of Quantitative Social Science (IQSS), Harvard University. I recollect with admiration and respect my long association with Dr. Patrick Hoffman, the data mining and modeling maestro, and Dr. John McCarthy of AnVil days.

I, Sriram, and Usha trace our friendship from good ole IISc days, thanks so much for everything over the years. My friendship with Sanjeeva and Anita is inextricably intertwined. Of course, none of my (mis) adventures would have been possible without the support of my family – my wife Sudha and our son Anuj. Sudha is the rock in the family. She stood unwaveringly through the thick and thin of it all. She did it with extraordinary dignity, poise, and grace. She does not suffer fools gladly and she is an engineer extraordinaire. Our son's sense of sarcasm is acute, and often, I am the hapless victim.

In closing, I affectionately remember my sister, parents – my first teachers, and all the great teachers I encountered – who inspired me, who encouraged me, and who thought me the joy of a lifetime of learning.

To
Anuj & Sudha

&

In Memory of
My sister – Anu and my parents

&

In Memory of
All those whose lives were cut short
for reasons
Natural or otherwise pursuing their dreams @ MIT

Systems Pharmacology – Machine Learning Approaches in Profiling Oncology Drug Candidates

by

ML Ujwal PhD

Submitted to the Engineering Management Program
on February 21, 2021, in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Engineering and Management

Abstract

While the thesis is framed from the *systems thinking* perspective, however, the main focus is on the drug discovery and application of machine learning approaches in profiling oncology drug candidates for a select subset of validated targets in the oncogenesis pathways. In this study, we built *in-silico* predictive models to predict prospective drug candidates from compound libraries. Robust predictive models help in saving enormous experimental, and resource overheads and compress product cycle times. We used several machine learning algorithms, in building models that include logistic regression (LR), support vector machines (SVMs), Naïve Bayes, Artificial neural nets (ANN), and Decision trees – classification and regression tree (CART) and multi-tree majority voting ensemble techniques i.e., random forest and XGBoost.The feature sets for building these models were extracted by computing chemical fingerprints and quantum chemical descriptors. We generated both sparse and dense matrices for modeling. We cross-validated, parameter hypertuned, and evaluated model performance on different statistical performance metrics, including Receiver-Operating Characteristic (ROC) curves.

We investigated the *full* and *reduced* model through feature engineering for model stability with LR models. We evaluated model regularization techniques, namely, LASSO, Ridge, Elastic Net, and Neural drop to prevent model *overfitting* both for LR and ANN models. We evaluated SVM kernels and showed non-linear radial basis function (RBF) performed better than others. We also showed that adding additional hidden layers, beyond three, to the ANN model with ADAM optimizer did not improve performance. Besides, multi-tree ensemble models were superior to single tree models (CART). Finally, we benchmarked the performance metrics of each of these machine learning algorithms in a side-by-side comparison and conclude that the ensemble random forest produced the lowest mean misclassification error.

Thesis Supervisor: Bryan R. Moser PhD
Title: Academic Director and Senior Lecturer, System Design & Management, MIT
Thesis Supervisor: Prof. Warren Seering PhD
Title: Weber-Shaughness Professor of Mechanical Engineering, Dept. of Mechanical Engineering, MIT

Prelude:

A general road map of the thesis. First, we define the problem statement, scope, solution, and value proposition.

I. **Problem**: Modern drug discovery is an enormously complex and resource-intensive undertaking involving several scientific and engineering disciplines, and multiple stakeholders. With the advent of high-throughput screening, robotic liquid handling, automation and compute infrastructure, there is an explosion of data. However, there is an unmet need to leverage the large volumes of data to extract 'patterns', 'associations', and 'insights' to drive innovation and accelerate drug development.

II. **Scope**: Although we frame the problem from the systems thinking perspective, invoking the interconnections among the entities - biological complexity, disease, and drug development. The scope of this thesis is restricted to the development of predictive models for profiling oncology drug candidates in the early phase of compound library screening in drug discovery.

III. **Solution**: We explored a wide range of machine learning classification algorithms to build predictive models by extracting feature sets by computing quantum chemical descriptors and chemical fingerprints. The bioassay read-outs aided in assigning the class labels to the samples.

IV. **Value Proposition**: Predictive models in drug development help in identifying potential drug candidates by reducing resource and investment overheads, reducing drug attrition, and compressing cycle times.

The following is an overview of the structure, organization, composition, and content of the thesis.

In **Chapter 1: Introduction** - Here we define, describe, and develop the background narrative on the topics of systems thinking, biological complexity, network biology, polypharmacology, machine learning models, cheminformatics, and chemical graph theory and its role in the development of chemical descriptors and fingerprints.

In **Chapter 2: Approaches** - Here we discuss data sources, computing feature sets, modeling work-flow, and evaluation metrics.

In **Chapter 3: Evaluation of Algorithms** - Here we give a detailed background to different machine learning algorithms and describe the implementation, experiments, and results.

In **Chapter 4: Discussion -** Here we offer the interpretations and implications of our results in the context of contributions from other researchers.

In **Chapter 5: Conclusions** – Here we summarize our findings, limitations, future direction, and conclude with a vision statement.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

"The task is not so much to see what other's have not yet seen; but to think what nobody has yet thought, about which everybody sees." - Erwin Schrodinger

1.1 Drug Discovery – A Systems Perspective

The fundamental scientific question we aim to address and tackle in this thesis is the role of the application of machine learning algorithms in accelerating drug discovery. More specifically, evaluation of predictive modeling in profiling oncology drug candidates. The pursuit of drug discovery is as ancient as mankind. Herbal concoctions from botanical sources formed the earliest ideas of drugs. Consumption of oral concoctions, applications of topical pastes, and poultice was routinely used to produced some relief, cure, and comfort from some simple ailments to some complicated human diseases. This was a common practice before the advent of modern pharmaceutical drug discovery. Systems thinking in understanding biological complexity is a more recent development. Deeply rooted in this biological complexity is the state of normal and perturbed physiological state. The perturbed physiological state is commonly referred to as disease. Modern drug development seeks to provide therapeutic interventions to restore a normal physiological state. In the present study, we frame our problem against the background of *systems thinking* and *biological complexity* to introduce some state-of-the-art *machine learning approaches*, as a mechanism to advance, innovate, and accelerate the drug discovery processes. More precisely, the focus of this thesis is on building, evaluating, and comparing predictive models to nominate potential drug candidates for a select subset of oncology drug targets.

1.1.1 Systems Thinking – A Definition

The earliest ideas of systems thinking was pioneered in the early 1950s by Jay Forrester at MIT. Since then many academics thinkers and industry researchers have contributed to refine the original idea. One of the ideas advanced by Arnold and Wade (Arnold and Wade, 2015) states, "Systems thinking is a set of synergistic analytic skills used to improve the capability of identifying and understanding systems, predicting their behaviors, and devising modifications to them in order to produce desired effects. These skills work together as a system."

1.1.2 Systems Thinking – Understanding Biological Complexity

Reductionism and systems thinking have been competing ideas that have dominated scientific discourse over the years. Here the reductionism and systems thinking is explicitly discussed in the narrow context of biological systems and drug discovery. Reductionists believe that all phenomena can be reconciled into fundamental physics and chemistry. Nobel prize-winning Francis Crick famously claimed in 1966 that the "The ultimate aim of the modern movement in biology is to explain all complex biological systems in terms of physics and chemistry." Reductionists analyze a larger system by breaking it down into pieces and determining the

connections between the parts. They assume that the isolated molecules and their structure have sufficient explanatory power to provide an understanding of the whole system (Van Regenmortel, 2004).

Biological systems are complex and have emergent properties. Often, understanding or predicting their behavior is a daunting task, that can not be accomplished by simply analyzing their constituents. The constituents of a complex system interact in a myriad ways, including positive feed-back, negative feed-back, feed-forward that leads to dynamic properties that cannot be predicted satisfactorily by linear mathematical models that disregard cooperativity and non-additive effects (Aderem and Smith, 2004). Another important property of complex systems is their robustness, and ability to adapt by compensatory mechanism and redundancy built into the system. Another characteristic of such systems is the modular design that ensures the partial failure modes are highly tolerated (Alm and Arkin, 2003). Complex biological systems are open, they exchange matter and energy with their environment, therefore they are not in thermodynamic equilibrium, leading to some radical ideas of systems approaches to biology and medicine.

Leroy Hood's group (Ideker et al., 2001), provided the first practical framework for the advancing 'systems thinking' to human diseases and drug discovery. Much of the field is shaped by numerous contributors who have advanced key concepts in life's complexity pyramid from individual molecules to interacting subsystems; dynamical diseases and importance of biological rhythms, and multiple parallel control systems (van der Greef and McBurney, 2005) as cited in (Glass and Mackey, 1988; Oltvai and Barabási, 2002). Biological interactions at many different levels of detail, from atomic interactions in a folded protein structure to relationships of organisms in a population or an ecosystem, can be modeled as networks.

1.1.3 Systems Biology – Perspectives on complex diseases

While the Mendelian genetics perspective of 'one gene - one disease' was foundational to our understanding of human diseases, we have also seen its limitation. It is being increasingly recognized that to gain a system-level understanding of disease at the molecular level, a systems view of the relevant physiological function is imperative. Every physiological function is an elaborate process involving an underlying signaling network of proteins, receptors, ligands, enzymes, metabolites, DNA/RNA modulators, transcription factors, genetic determinants, mRNA transcription, protein translation, hormones, ions, and electrical signals. Any number of these reactions have different reaction kinetics both at temporal and spatial levels. Therefore, each physiological function and the associated phenotype is a representation of a complicated network of biological signals. Network approaches – comprising of representational nodes and edges provide a broad framework of the interactome, in proving some fundamental understanding about the etiology of the diseases (Boran and Iyengar, 2010). Thus disease state may be defined as a perturbation of the network at a discrete node or many disparate nodes in the network.

1.1.4 Systems Pharmacology – At the intersection of Systems Biology, Disease and Drug Discovery

Systems pharmacology is the extension of the systems biology approach that seeks to understand the interaction of pathophysiology and drug action at the organ and organismal level. The application of experimental system biology and machine learning approaches including network analysis will enable investigation of the multiscale biological organization - molecular perturbations, therapeutic interventions, and adverse effects of drugs, if any. Recent advances in high-throughput methods and "omics" technologies (i.e., genomics, transcriptomics, metabolomics, and proteomics methods) have provided some glimpse of the overall structure of molecular interaction networks in biological systems. Instead of being viewed as a collection of the parts list, in complete isolation of connections, dependencies, chemical kinetics, and thermodynamics. Systems pharmacology seeks to combine "omics" data, global network analysis, pharmacokinetics, and pharmacodynamics models, and genome polymorphisms to develop polypharmacology of complex human diseases to predict therapeutic efficacy and adverse event risks by connecting biomolecular perturbations to the emergence of disease pathology phenotypes.

The concept of polypharmacology is an interesting attempt at treating polygenic complex diseases by targeting multiple disease targets in a regulatory network with one or more drugs to reset the interaction circuit. Such attempts at amelioration of the disease states also come with certain caveats of unintended consequences. Combination drug therapy is still in the early stages of experimentation requiring co-development of computational methodologies of calibrating dose-response curves of individual drugs within a regulatory network. Drug discovery can be viewed as a search for agents that significantly restore near cellular homeostasis to these pathological network perturbations (Alm and Arkin, 2003; Boran and Iyengar, 2010; Zhao and Iyengar, 2012). These drug agents by binding to their target nodes, exert their influence, either by attenuating aberrant signals or by modulation the signal. While target-driven drug discovery has been the mainstay of pharmaceutical drug development campaigns for over the last three decades and continues to be important. Network biology has added a rich repertoire to the molecular dynamics of the interactome that enriched and richly complemented the target-driven drug discovery. The adoption of new platforms, technologies and compute infrastructure in the pharmaceutical endeavor has led to the explosive growth of data across the entire value chain. The adoption of Machine learning (ML)/Artificial Intelligence (AI) opens up the limitless possibilities of delivering next-generation therapies for human diseases. Figure 1.1 gives an overview of the current pharmaceutical drug discovery paradigm that can be recast in the systems thinking perspective: inter-connected entities of functional sub-systems.

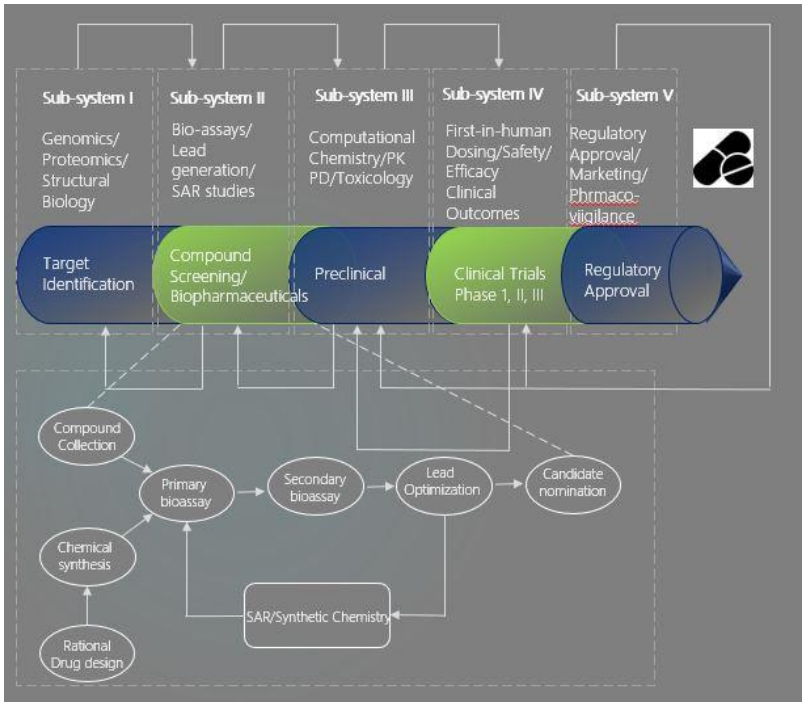


Figure 1.1 Systems Perspective of Pharmaceutical R&D
The functional areas in a typical pharmaceutical workflow are represented as sub-systems of a large complex system. The bulk of the present study is focused on the early discovery phase.

1.2 Machine Learning Models

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience (Mitchell, 1997). It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as training data to make predictions or decisions without being explicitly programmed to do so.

Mitchell provided a formal definition of the Machine Learning Model, that is simple, concise, and elegant (Mitchell, 1997)
A computer program is said to learn from experience $\boldsymbol{E}$ *with respect to some class of tasks* $\boldsymbol{T}$ *and performance measure* $\boldsymbol{P}$*, if its performance at tasks in* $\boldsymbol{T}$*, as measured by* $\boldsymbol{P}$*, improves with experience* $\boldsymbol{E}$*.*

Marshland (Marsland, 2014) expands upon Mitchell's definition
One of the most interesting features of machine learning is that it lies on the boundary of several different academic disciplines, principally computer science, statistics, mathematics, and engineering. Machine learning is usually studied as part of artificial intelligence, which puts it firmly into computer science, understanding why these algorithms work requires a certain amount of statistical and mathematical sophistication.

1.2.1 Supervised Learning

Supervised learning entails learning a mapping between a set of input variables X and an output variable Y and applying this mapping to predict the outputs for unseen data. In the supervised learning paradigm, the goal is to infer a function $f : \mathrm{X} \rightarrow \mathrm{Y}$, the classifier, from a sample data or training set $\mathrm{A_n}$ composed of pairs of (input, output) points, x_i belonging to some feature set X, and $y_i \in Y$:

$$A_n = (x_1 y_1), \ldots, (x_n y_n)) \in (X * Y)^n$$

Typically $\mathrm{X} \subset \mathbb{R}^d$ Rd, and $y_i \in \mathbb{R}$ for regression problems, and y_i is discrete for classification problems

In the statistical learning framework of supervised learning, the first fundamental hypothesis is that the training data is independently and identically generated from an unknown but fixed joint probability distribution function $P(x, y)$. The goal of the learning is to find a function $P(x, y)$. The goal of the learning is to find a function f attempting to model the dependency encoded in $P(x, y)$ between the input x and the output y. Let us denote H as a set of functions where the solution is sought: $f \in H$.

The second fundamental concept is the notion of loss to measure the agreement between the prediction $f(x)$ and the desired out y. An error (or cost) function $E : Y * Y \rightarrow \mathbb{R}^+$ is introduced to evaluate this loss. Error functions are classified according to their regularity or singularity properties and according to their ability to produce convex or non-convex criteria for optimization. A general formulation of E is given by:

$$E = \sum_{(x_n y_n) \in D} \mathcal{L}\,(y_n f(x_n))$$

With D being the dataset and $\mathcal{L}$ is some distance function. The learning process is terminated when E is sufficiently small or a failure criterion is met.

1.3 Motivation

The motivation for the present study: (1) To leverage the power of *in-silico* predictive modeling using machine learning algorithms (ML/AI) to implement robust and resilient models in the early screening phase of drug discovery. (2) To evaluate the performance and define the *fit-for-purpose* of these algorithms in drug discovery. (3) To reduce research overheads (4) To reduce drug attrition rates. (5) To compress cycle times and bolster productivity in drug discovery and development.

1.4 Building Predictive Models

First one builds prototype models for defined targets, evaluate model performance. Later, one scales the prototype models into production-ready models and deploys such models into the drug discovery workflow.

The compound collection from high-throughput bioassays provides the input for building the predictive models.

1.4.1 Mapping Compounds to Oncology Drug Targets

A large collection of compounds both from public and private sources for selected oncology drug targets derived by pooling multiple assays are assembled into discrete piles. Each pile is preprocessed for building models (see Figure 1.2).

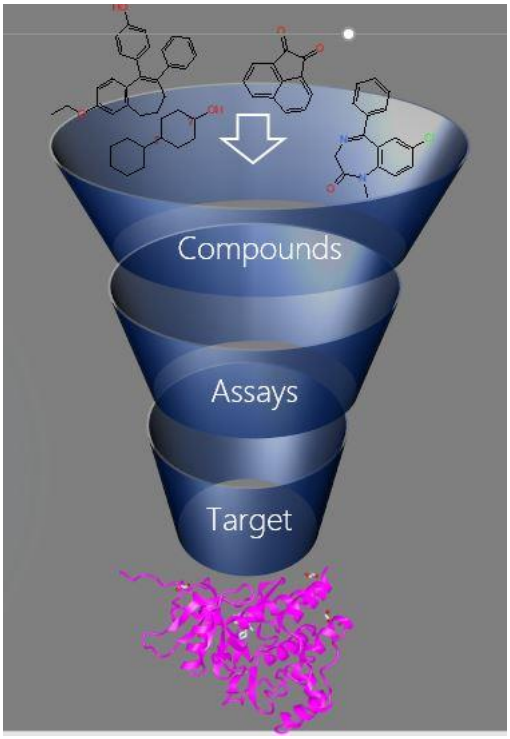


Figure 1.2 Mapping of Compounds to Oncology Drug Targets
For a given Oncology Drug Target, compounds from multiple sources and multiple bioassays are combined.

1.5 Cheminformatics – Convergence of Chemistry, Biology & Computations

Cheminformatics combines scientific disciplines like chemistry, biology, computer science, and information science to transform chemical and biological data into knowledge; knowledge into insights; insights into decisions in drug lead generation, lead optimization, and drug candidate nomination. This includes storage, indexing, and retrieval of structural information, datamining, predictive modeling, and model optimization (Chen et al., 2018). Cheminformatics is a core discipline on its own that grew in the late 90s with the emergence of technologies like combinatorial chemistry and high-throughput screening. One of the most significant early *concept* development was the representation of complex chemical structures in computationally-compatible 'string' structures – Simplified Molecular- Input Line Entry Systems (SMILES) (Weininger et al., 1989).

Classical molecular dynamics (MD) simulations approach to biological molecules is integral to cheminformatics, they play a crucial role in drug discovery. They help trace atomic motions that yield insights into molecular mechanisms that typically occur in the time scales of microseconds or milliseconds, for example, the "folding" of proteins into their native three-dimensional structure, the structural changes that underlie protein function and interaction between a protein and a candidate drug molecule. Anton architecture of massively-parallel, specialized supercomputers performs MD computations at astonishing time scales on the order of millisecond – about two orders of magnitude beyond the previous state of the art (Shaw et al., 2015).

1.5.1 Combinatorial Chemistry

Combinatorial chemistry involves the generation of a large array of structurally diverse compounds, called compound libraries, through systematic, repetitive, and covalent linkages of various chemical 'building blocks'. Once such compound collections are built, they can be screened for specific biological targets of interest by a specific bioassay. Active compounds can be identified, either directly (in position-addressable libraries) or via decoding (using genetic or chemical means) (Liu et al., 2017).

The concept of combinatorial chemistry was developed in the mid-1980s with Geysen's multi-pin technology (Geysen et al., 1984) and Houton's teabag technology to synthesize hundreds of thousands of peptides on a solid support in parallel (Houghten, 1985). One of the revolutionary developments came from the exploitation of biological agents for the production of combinatorial libraries. Smith described the phage display peptide library (Smith, 1985). Recent advances in DNA-encoded chemical libraries have allowed investigators to create and decode a huge diversity of small-molecule organic, peptide, or macrocyclic libraries.

1.5.2 Chemical Library Design

The term "library design" is usually taken to mean the design process, which occurs before the chemical synthesis, in which a set of structures are taken from a large chemical library. Usually, the chemical libraries are created to explore the structure-activity relations of the 'hit' series but also improve the physio-chemical properties – ADME (Absorption, Distribution, Metabolism, and Excretion). Diversity is an important factor in the design of libraries. Around a common chemical core or a scaffold, multiple functional groups are substituted. The compound structural diversity is measured, for example, by structural fingerprints or the Tanimoto index. While this design ensures a diversity of substituents in the peripheral part of the molecule while retaining the common core scaffold. Modern combinatorial libraries' strategies include the introduction of multiple scaffolds and multiple peripheral functionalizations (Klein and Lindell, 2013). Other approaches include multi-objective optimization methods capable of taking several chemical and biological criteria that have been used in the design of compound collections. Besides, other active learning strategies have been introduced, the core feature of these algorithms is their ability to adapt to the structure-activity landscapes through feed-back. Many of these computational methods have greater success in the design of focussed libraries of limited diversity(Liu et al., 2017).

1.5.3 Virtual Libraries and Chemical Space

By definition, virtual libraries are those virtual compounds that are generated by computational methods and the libraries thus generated are evaluated by virtual screening, which refers to a range of *in-silico* techniques used to search large compound databases to select a small fraction for biological testing. Notwithstanding chemical or virtual libraries, the challenge remains in screening for druggable compounds, how large is the chemical space? One of the earliest estimates of the size of chemical space came from Weininger (Weininger, 2002), who used a set of 150 substituents on hexane to estimate the size of the organic chemical space at 10^{29}. At the other extreme, Bohacek proposed 10^{63} molecules by a build-up of a linear chain consisting of C, N, O, or S atoms that consider the possibilities of 6 substitutions at every position on a 30 length chain to a total of 6^{29} (or ~2 x 10^{23}). The addition of ring closure led to an estimate of 10^{40} molecules, and if you extend to 4 rings and 10 branch points, the combined estimate is 10^{63} molecules (Bohacek et al., 1996). In more recent times, data-driven approaches have gained greater traction to traditional methods. Chemical space projects at the University of Berne have developed algorithms for exhaustively enumerating the molecular skeletons, on bond orders, atom types, and excluding those with rules of valency or contain potentially unstable bonds and built a database, namely, GDB (Generalized Database). GDB has been evaluated by numerous researchers, using the distribution of compounds generated in the database, which enumerates all possible atoms up to 17 atoms with the following equation. The most recent GDB-17 contains molecules up to 17 heavy atoms and has 166 billion molecules (Ruddigkeit et al., 2012).

$$\text{Log } M = 0.584N \log(N) + 0.356$$

where M is the number of molecules and N is the number of atoms in a molecule. By using the above equation to extrapolate to 36 atoms, the authors arrive at an estimate of 10^{33} drug-like compounds. In another interesting development, Chevillard and colleagues generated the reaction-oriented virtual libraries of 21 million products, by carrying out a virtual reaction on a pool of 8000 commercially available building blocks (Chevillard and Kolb, 2015). While the promise of virtual libraries is appealing, in practical terms it has its limitations because of the constraints imposed by the small number of validated synthetic chemical reactions available for chemical synthesis (Walters, 2019).

1.5.4 Virtual Screening

Virtual screening can be classified into two broad categories, i.e., *structure-based methods* and *ligand-based methods*. Structure-based methods are applicable where 3-D structural geometry coordinates of the target of interest are available. Chemical libraries are searched for molecules that are compatible with a protein binding site. These methods require an exploration of the ligand conformation as well as binding site compatibility. Docking is the general term used for this process which calculates the score/energy associated with protein-ligand interaction. Docking programs can screen several million compounds in a short time. The time required to screen the compounds mainly depends on the size of the database, scoring function, and compute power. The choice of the scoring function is critical in docking experiments, one of the approaches called linear

interaction energy (LIE) approximation combines molecular mechanics (MM) calculations with experimentally available data for the model scoring function (Kumar et al., 2015; Singh et al., 2005). The Adaptive Poisson-Boltzmann solver calculates the electrostatic binding free energy, which accounts for the solvation energy of the protein-ligand interactions (Baker et al., 2001; Kumar et al., 2015). One of the issues, that is still being tackled is addressing the protein flexibility in the screening process. The soft docking methods allow a few steric clashes by lowering the steepness of the repulsion term in the Lennard-Jones potential function (Kumar et al., 2015). In ligand-based virtual screening, one starts with a known active molecule and searches a chemical library (either real or virtual) to identify similar molecules. Searches can be performed for molecules that are similar based on 2D topology, which is information on atom types and connectivity, or for molecules with similar 3D characteristics such as shape or arrangement of pharmacophoric features (Walters, 2019).

1.5.5 High-throughput Screening

A screening procedure, typically adopts automation, microfluidics, sensitive detectors, robotics, and data processing, at scale, for liquid handling and assay measurements. In a typical high-throughput screening (HTS), the measurements are carried in micro-titer plates with several thousand compounds and proteins/enzymes in micro incubator wells for specific assays. The automated systems shuffle plates from station to station on the assembly for sample and reagent addition, mixing and incubation, and assay readouts.

The single over-arching goal in drug discovery is to find a lead compound that can be optimized to produce a drug candidate. The acquisition, design, and screening of compound libraries are absolutely critical. The two major strategies, which is evolved over the years in the industry, that are currently being employed are diversity-based design and target-based design. Diversity-based design is a random collection of structurally diverse molecules, that are selected on the basis of the cell-based phenotypic assay. On the other hand, target-based libraries are specific to a particular therapeutic target or a family of related targets and are tailored to modulate their function. For example, kinases, G-protein-coupled receptors, voltage-gated ion-channels, or serine/cysteine proteases, to name a few. Fluorescence-based detection systems, which offered unrivaled sensitivity and adaptability are being transitioned to key breakthroughs in the label-free and the development of high-throughput mass spectrometry systems. High-throughput screening has been the mainstay of drug discovery over decades and with these new technologies it is poised to rein in.

To more efficiently search the vastness of the chemical space, it is imperative we take advantage of the advances in machine learning, which may help us navigate the chemical space. If we can develop a continuous representation of chemical space, we can potentially exploit the gradients in that space to identify an optimal region for exploration. Notwithstanding, building high-quality predictive models will dramatically improve productivity (Walters, 2019).

1.6 Chemical Graph Theory

Graph theory is a branch of discrete mathematics, related to combinatorics and topology. It deals with the way the objects are connected and with all the consequences of connectivity. The connectivity in a system is thus the fundamental quality of graph theory. *Chemical graph theory* is the adoption of the fundamental principles of graph theory to chemical structures to understand their atomic properties and their biological activities. A chemical graph, also known as 'molecular graph' or 'structural graph' is a mathematical construct comprising an ordered pair $G = (V, E)$, where V is a set of vertices (atoms) connected by a set of edges (bonds) E. A *graph invariant* is a number, sequence of numbers, or a matrix computed from the graph topology. There are multiple variations of representation of chemical structures as chemical graphs, depending on the diversity of atom types, bond types, ring structures, and atom grouping that form functional groups. Weighted chemical graph assign values to the edges to indicate the bond length and their atomic valency Pseudographs or reduced graphs use multiple edges and self-loops to capture detailed bond valency information (Lo et al., 2018). For any kind of a graph, we can define an *adjacency matrix* and a *distance matrix*.

The importance of graph theory for chemistry stems mainly from the phenomenon of isomerism, which is rationalized by chemical structure theory, which accounts for all constitutional and steric isomers by using purely graph-theoretical methods. Graph theory and theoretical chemistry are intimately related through the molecular topology. Huckel's MO theory and VB theory are topologically based, though steric considerations are relevant. It is revealing to know, that HMO eigenvalues (in β units) are identical with the eigenvalues of the adjacency matrix of the hydrogen-depleted graph (Balaban, 1985; Bonchev and Rouvray, 1991; Trinajstic, 1992). Comparing chemical graphs is a critical task in pattern recognition to map graph isomerism, subgraph isomerism, maximum common subgraphs. Indeed, extensive studies have been done to develop practical algorithms for this purpose. In computational complexity, polynomial-time algorithms are those, where the computational time is proportional n^d (i.e., computation time is $O(n^d)$ for some constant d, where n denotes the size of the input data. *NP-hard* problems are widely believed not to be solved in polynomial-time and are often regarded normally as intractable. However, the number of vertices in a chemical graph is somewhat limited, often below 100, they are close to *NP-hard* but remain reasonably tractable (Akutsu and Nagamochi, 2013).

Thus, computation of graph invariant topological indices or connectivity of molecules became important in *structure-activity* modeling in drug design and drug discovery, where we seek to correlate chemical structures to biological activity (Lo et al., 2018). There exist several types of such indices, especially those based on distances and independent sets of vertices and edges. The Wiener and Hosoya index is arguably the best-known global indices that describe the entire molecule. More recently, Ilic and Ilic (Ilic and Ilic, 2017) proposed improved algorithms for computing such indices. Particularly, the design of simpler recursive procedures for the Hosoya index of trees and acyclic graphs and better time complexity for computing other indices, for instance, Wiener, Merrifield-Simmons, and Balban (Ilic and Ilic, 2017; Mihalić and Trinajstić, 1992). Chemical graph theory is predicated on the premise, that chemical structures are fully specified by their graphs

representation, as such, they contain the information necessary to model and provide insight into a wide range of biological phenomena.

1.6.1 Associating Graphs with Matrices

A labeled (chemical) graph may be associated with several matrices, The two important graph-theoretical matrics are the vertex-adjacency matrix and distance matrix. The vertex-adjacency matrix, $A = A(G)$ of a labeled connected graph G with V vertices, is a square symmetric matrix of the order of N. It is commonly called adjacency matrix. It is defined below.

$$A_{ij} = \begin{cases} 1, & \text{if vertices } i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases}$$

The distance matrix, $D = D(G)$, of a labeled connected graph G with V vertices is a square matrix of order N, it is defined below.

$$D_{ij} = \begin{cases} l_{ij} & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

Where l_{ij} is the length of the shortest path (i.e., the distance) between the vertices i and j in G (Mihalić and Trinajstić, 1992).

1.6.2 Graph-Based Molecular Descriptors

A graph-based molecular descriptor, commonly know as the topological index (TI), is a graph-theoretic invariant characterizing numerically the topological structure of a molecule (Ernesto and Bonchev, 2014).

The *Wiener index* of a (molecular) graph is a TI is defined by

$$W = \sum_{i<=j}^{n} d_{ij}$$

Where d_{ij} is the *shortest-path distance* between vertices i and j

The *Hosoya index* of a (molecular) graph is a TI is defined by

$$H = \sum_{i=0}^{n/2} P(G, i)$$

Where $P(G, i)$ is the *number of selection of I mutually non-adjacent edges* in the graph, By definition

$P(G,0) = 1\ and\ P(G,1) = m$

The *Zagreb index* of a (molecular) graph is a TI is defined by

$$M_1 = \sum_{j=1}^{n} (\delta_j)^2$$

$$M_2 = \sum_{i,j \in E} \delta_i, \delta_j$$

The *Randić index* of a (molecular) graph is a TI is defined by

$$\mathrm{X} = \sum_{i,j \in E} (\delta_i \delta_j)^{-1/2}$$

The *Kier and Hall molecular connectivity index* of a (molecular) graph is a TI is defined as

$$k_x = \sum_{i,j,l,\ldots} (\delta_{i,} \delta_{j,} \delta_l, \ldots)^{-1/2}$$

Let $k = 0.1.2.3,\ldots$ is the number of adjacent vertices of degrees $\delta_{i,} \delta_{j,} \delta_l, \ldots$ in the graph G, where the summation is taken over all subgraphs of the size k, and the null term is the sum of all the vertex degree (the total adjacency of G).

The *Balaban index* of a (molecular) graph is a TI is defined as

$$\mathrm{J} = \frac{m}{C+1} \sum_{i,j \in E} (s_{i,} s_j)^{-1/2}$$

where $C = m - n + 1$ is the *cyclomatic number* of the graph

The *atom-bond connectivity index* of a (molecular) graph is a TI is defined as

$$\mathrm{ABC} = \sum_{i,j \in E} \sqrt{\frac{\delta_i + \delta_j - 2}{\delta_i \delta_j}}$$

1.6.3 Chemical Descriptors

Chemical descriptors are a discrete numerical representation of the molecular structure which provides feature vectors for predictive modeling. Besides, predictive modeling can be useful for molecular data mining and compound collection structural diversity analysis. Fundamentally it is a structural characterization that can range from simple graph invariants like atom counts, atom types, bonds types, number of bonds, molecular topology to quantum mechanical properties derived from molecular electron wave function (Todeschini and Consonni, 2008), for example, molecular surface charge, polarizability, dipole moment and other thermodynamic properties. Chemical descriptors can be broadly classified into 1-dimensional (1D), 2-dimensional (2D) or 3-dimensional (3D) descriptors. One-dimensional descriptors capture the bulk properties, i.e., molecular weight, molecular refractivity, logP (logarithm of the octanol/water partition coefficient); two-dimensional descriptors describe the molecular topology, in these approaches the molecule is often regarded as a graph annotated with complex properties, which allows several graph-based algorithms (Cook and Holder, 2007; Wilson, 1996) to compute graph invariant indices like the Wiener, Hosoya and Balaban and several other descriptors (Fechner et al., 2010); three-dimensional descriptors inform the spatial disposition of different biologically active and inactive conformations, that are not directly accessible by experimentation. At the core, is the pharmacophore mapping, an ensemble of steric and electronic features that are optimal for supramolecular interactions. Typically, hydrophobic regions, aromatic rings, lipophilic centers that participate in hydrogen bond donor-acceptor network, and ionic interactions between chemical ligand and the biomolecule. They have extracted chemical features from 3D coordinates representation, hence very sensitive to steric variation. Well-known 3D descriptors include autocorrelation descriptors, substituent constants, surface volume descriptors, and quantum-chemical descriptors (Guha and Willighagen, 2012; Lo et al., 2018; Oprea et al., 2007). Although 3D descriptors allow more detailed analysis, the key limitations are that it is computationally an expensive exercise for conformer generation and no absolute guarantee the predicted conformer is biologically active. Fourches and Ash have demonstrated the utility of yet another class of descriptors – the 4D descriptors, which is a formal extension of 3D descriptors. In a molecular dynamics (MD) simulations experiment, they computed 3D descriptors over a grid box based on the 20 ns trajectory and showed that 4D descriptors can effectively differentiate the most active ERK2 inhibitors from the inactive ones, with superior enrichment rates (Ash and Fourches, 2017). However, the conformation-dependent characteristics of flexible molecules and their dynamic interactions with the biological target(s) are not encoded by these descriptors, leading to limited prediction performances and reduced interpretibility. It must be clearly mentioned that here is renewed resurgence in this methodology, with the advent of GPU-driven high performance computing environments (Fourches and Ash, 2019). Notwithstanding, 2D descriptors are still the work-horse for drug discovery in the industry in the area of advanced predictive modeling for its computational speed, reliability, consistency and scaling metrics.

1.6.4 Chemical Fingerprints

Chemical fingerprints are a common method to combine the presence or the absence of different substructures in a molecule into one descriptor (Todeschini and Consonni, 2008). They were originally designed to assist in

chemical database substructure searching (Christie et al., 1993) but later used for analytical tasks, such as similarity searching (Schulz, 1992) and, clustering (McGregor and Pallai, 1997) and classification problems (Hastie et al., 2009), akin to what chemical descriptors are used for the latter two tasks (described in the earlier section). They are usually represented as a vector of bits with a fixed length that denotes the presence of a specific structural pattern. Many fingerprint implementations can be classified as non-hashed and hashed fingerprints. The non-hashed fingerprints are also known as *structural keys* are mainly based on a predefined dictionary of substructures, such that there is a unique mapping between a bit vector position and a specific hydrogen-stripped substructure. If the molecule has a predefined feature, the bit position corresponding to this feature is set to 1 (ON), otherwise, it is set to 0 (OFF). It is worth mentioning that structural keys cannot encode a structural feature that is not present in its pre-defined dictionary. The two commonly used *structural keys* are MACCS (Molecular ACCess System) and PubChem fingerprints. The MACCS was developed by MDL information systems (now Dassault Systemes/Biovia). There are two sets of MACC keys, one with 960 keys and the other containing a subset of 166 keys, the shorter set is available to the public (Durant et al., 2002; "MDL Keyset Technology," n.d.). These 166 public keys are implemented in several open-source software packages, including RDKit (Landrum, 2012), OpenBabel ("Open Babel,"), CDK (Steinbeck et al., 2006), etc. On the other hand, the PubChem fingerprint is a publicly-funded effort ("The PubChem Project," 2017), which is an 881-bit-long structural key. The keys can be accessed either interactively on the PubChem homepage or programmatically via PUG-REST web services.

Hashed Fingerprints are an alternative to the structural keys. Contrary to structural keys, hashed fingerprints do not need a predefined fragment dictionary. Instead, they are generated by enumerating all the possible subgraph fragments of topologically unique paths of different lengths of the molecule (composed of atoms and bond symbols). Each of these fragments is converted into canonical SMILES string representation and is passed to a "hash" function that generates unique fixed-length hash codes by standard hashing algorithms. Following which the random generator integer values are used to set the *position* in the bit string, which is set to 1 for a specific fragment pattern.

Hashed fingerprints may be further classified into topological or *path-based fingerprints* and *circular fingerprints*, depending on the way the sub-graph *patterns* are enumerated. As the name implies, path-based fingerprints count the topological structures of various unique bond lengths of atoms of unique composition. Although, this may appear to be uniform randomly distributed throughout the length of the fingerprint ("Daylight Theory: Fingerprints,"). It must be mentioned, that there are instances of 'collision' of assignment of pattern to the same position, which may not be a trivial task. The *circular fingerprints* are generated by considering the "circular" environment of each atom up to a given "radius". For example, the circular fingerprints are the *extended-connectivity fingerprints* (ECFPs) (Rogers and Hahn, 2010). ECFPs are recently developed fingerprint methodology explicitly designed to capture molecular features relevant to molecular activity. While not designed for substructure searching, they are well suited to the tasks related to predictive modeling and gaining insights into the drug activity. ECFPs are generated by using a variant of Morgan's algorithm (Morgan, 1965), which a method for solving the molecular graph isomorphism problem (i.e., in graph theory isomorphism is a bijection between the vertex sets of two graphs G and H

$f: V(G) = V(H)$

such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. This kind of bijection is commonly described as "edge-preserving bijection").

There have been tremendous efforts made, both in commercial and open-source communities to implement a variety of tool kits to encode the decomposition of chemical graphs into chemical fingerprints. Todeschini's group is one of the pioneers in the implementation of an exhaustive and comprehensive collection of both chemical descriptors and chemical fingerprints in their Dragon software package (Mauri et al., 2006), the commercial version of which, is referred to as Kode cheminformatics ("Kode - Chemoinformatics," n.d.). Other commercial packages include Daylight fingerprints ("Daylight Theory: Fingerprints," n.d.) and ChemAxon ("Chemical Fingerprints - ChemAxon Documentation," n.d.). JCompound mapper is an open-source Jave library and a command-line tool for a variety of fingerprints, which implemented popular fingerprinting algorithms such as depth-first search fingerprints, extended connectivity fingerprints, autocorrelation fingerprints (e.g., CATS2D), radial fingerprints (e.g., Molprint2D), geometric Molprint, atom pairs and pharmacophore finger prints (Hinselmann et al., 2011). The Cinfony platform is an effort to combine several open-source cheminformatics tool kits behind one common interface, while openBabel, CDK, and RDKit share the same core functionality, they support a different set of formats and force fields (O'Boyle and Hutchison, 2008). The main goal of the Chemfp project led by Dalke is to promote the FPS format, as a *de facto* file format for the text-based exchange format for dense binary cheminformatics fingerprints, which is very useful in Tanimoto sub-structure searching, which has demonstrated some impressive benchmarking metrics (Dalke, 2019). ECFPs circular fingerprints have a number of useful qualities: 1) they are very rapidly calculated; 2) they are not predefined by a dictionary of structural features, so essentially they generate an infinite number of different molecular features (including stereochemical information); 3) their features represent the presence of a particular substructure, allowing more straightforward interpretation of the analysis; and 4) the ECP algorithm can be tailored to generate different types of circular fingerprints, optimized for various uses. Only recently, it is beginning to be adopted and validated for more sophisticated machine learning approaches.

Chapter 2: Approach

"I doubt, therefore I think, therefore I am." – Rene Descartes

2.1 Data Requirements

By definition, data requirement is a prescriptive top-down approach that emphasizes, that solutions to a problem, begin with a problem statement, then the scope, and the approach. In this specific instance, the problem statement is how to accelerate drug development, which is a protracted and expensive undertaking by the adoption of machine learning and predictive modeling in the early stages of drug discovery. Predictive modeling is a *learning problem*, where a function learns the *pattern* in the data and maps the inputs to outputs. The first and the foremost requirement is data *per se*, for the explicitly stated *use case*; candidate data sources; data processing; data quality assessed using bonafide assessment procedures.

2.1.1 Data Sources

The primary source of the data used in the present study is ExCAPE-DB, which was built by Astra-Zeneca, Sweden (Sun et al., 2017) and made available for the work undertaken. ExCAPE-DB is an integrated large-scale dataset for facilitating big data analysis in chemogenomics for drug discovery. PubChem (Kim et al., 2019) and ChEMBL (Gaulton et al., 2017) are examples of large public domain repositories for storing small molecules and their biological activity data. Besides large pharmaceutical companies maintain their data collection originating from their in-house HTS screening campaigns and drug discovery projects.

2.1.2 Data Preprocessing

While PubChem and ChEMBL are invaluable resources in the public domain, leveraging the combined resources entailed an enormous amount of preprocessing of data and application of standardization routines. ExCAPE DB helped in the mitigation of some of the inherent shortcomings in these databases. Standardization of the PubChem and ChEMBL chemical structures was accomplished by using the AMBIT cheminformatics platform (Jeliazkova and Jeliazkov, 2011) and the Chemistry Development Toolkit Library (Steinbeck et al., 2006). It includes many structure preprocessing options like handling stereochemistry, implicit hydrogens, SMILES, InChI (International Chemical Identifier) generation, and several other chemical structural transformations. The bioactivity standardizations explicitly limited bioassays to a single target; The assay target was limited to human, rat, and mouse species; data points missing a compound identifier (CID) were removed; only active compounds whose dose-response value equal to or lower than 10 uM were retained; compounds that were labeled as inactive in PubChem screening assays; assays run with single concentration were also kept as inactive records; slightly relaxed Lipinski *rule-of-five* compounds were included; and finally the chemical structural identifiers (SMILES, InChI, and InChiKey) generated after standardization routines were joined for creating the curated dataset, which is representative of chemical space relevant for predictive modeling in drug discovery (Sun et al., 2017).

2.1.3 Processing Bioassay Data: Oncology Drug Targets

Since the emphasis is on profiling oncology drug candidates, the data was retrieved for each of the validated lists of oncology-specific drug targets: their compound collection and the associated bioassays (see Figure 2.1). These validated targets are implicated in human cancer cell-signaling, etiology, and pathology. To our knowledge, this is for the first time, results from primary screens from multiple bioassays platforms, from large-scale compound collections, being pooled for analysis. However, only a small subset of these drug targets, which met the criteria of completeness of data, were pursued for the downstream processing. Each entry has a unique compound ID, SMILES representation, original assay ID, Inhibitory concentration (pXC50), and activity flag (active vs. inactive). The bulk of the processing was done with *R scripts* written within the *RStudio Integrated Development Environment* (IDE). *R* is an open-source statistical programming language. Other software tools include JMP from SAS, ChemAxon, Data Warrior, and Dragon for computing molecular fingerprints and descriptors. Data cleansing pipeline scripts are a series of data manipulation steps that include – removal of highly correlated columns; duplicate records are removed; columns with zeros in all the rows are removed; rows with missing values are removed; columns with constant values are removed; columns with very low standard deviation (<0.001) are removed. This is is an overall framework for the present study utilizing - Oncology target drug candidate datasets, features, and modeling algorithms (see Figure 2.1).

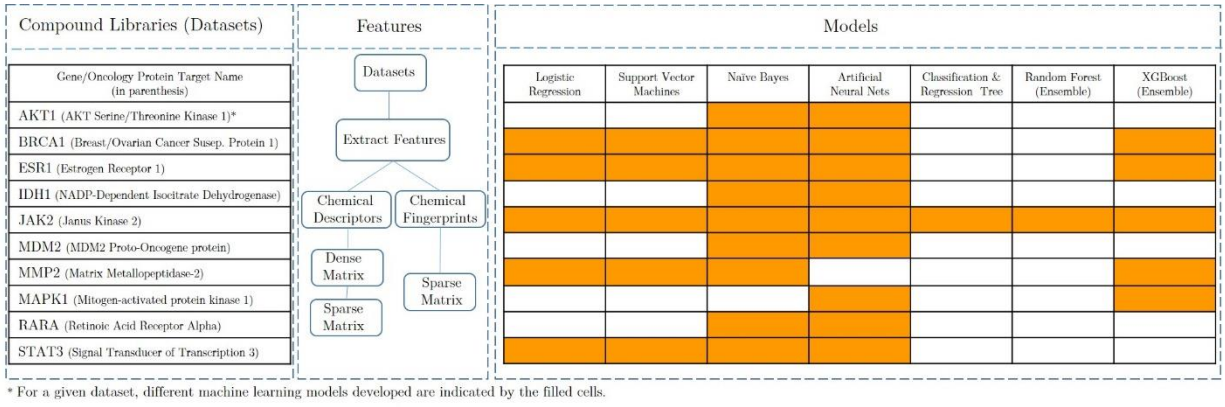


Figure 2.1 The overall framework for the present study
This shows at a glance all the salient aspects of the study - oncology target drug candidate datasets, feature extraction methods, matrix sparsity, and the algorithms for modeling.

2.2 Computing Features

The feature set for predictive modeling is built by computing chemical fingerprints and chemical descriptors for each compound entry in the dataset. Initialization scripts remove duplicate records, strip irrelevant columns, and retain only compound ID, SMILES, and Activity flag (class) columns, which is converted into a *.sd file (structure description) for ingestion by the Dragon engine to compute a feature set (Mauri et al., 2006). (see Figure 2.2)

2.2.1 Computing Chemical Fingerprints

In this study, we have calculated *hashed molecular fingerprints* and *extended connectivity fingerprints* (ECFP). Path-based fingerprints are generated identifying all the linear paths of the molecule while ECFP is generated exhaustively identifying circular fingerprints grown radially from each heavy atom of the molecule (Rogers and Hahn, 2010).

Both fingerprints types are calculated by defining a set of options like fingerprint size, minimum and maximum length of the identified fragment and an extensive list of atom properties to be considered to differentiate fragments (e.g., atom type, aromaticity, formal charge, connectivity, and bond order)

2.2.2 Computing Chemical Descriptors

In this study, we have computed about 1600 molecular descriptors that are divided into 20 logical blocks. The chemical behavior encoded within a symbolic representation of a molecule is extracted through computable data structures as the physicochemical properties of the molecule, for example, atom type, functional group, fragment count, hydrogen-bond acceptors and donors, number of rotatable bonds, charge, polarizability, aromaticity, and topological surface area (TPSA. 3D descriptors are not included in the analysis. To elaborate further, we can briefly describe a few of these blocks of *chemical descriptors* and what they encode (Dehmer et al., 2013).

1. *Constitutional descriptors*: This block reflects the composition of the molecule without any geometrical information, e.g., number of atoms, bond rings, specific atom types, rotatable bonds, etc.
2. *Topological indices*: These are structural graph measurements that take various structural features into account, e.g., distances and eigenvalues. The first topological index has been coined by Hosoya (Hosoya, 1988). The first and second Zagreb indices (Khalifeh et al., 2009).
3. *Connectivity indices*: These are calculated from the vertex-degree of a molecular graph, e.g., Randic index (Randić, 2001).
4. *Edge adjacency indices*: These indices are based on the edge adjacency matrix of a graph. The resulting descriptor-value is the sum of all edge entries of an adjacency matrix of a graph. Balban (Balaban, 1985) developed several indices by using graph-theoretical matrices.
5. *Walk path counts*: These indices are defined by counting paths or walks of a graph. The term walk refers to random walks that are based on probability measures described by Todeschini and Consonni (Todeschini and Consonni, 2008).
6. *Information indices*: These measures are based on Shannon's entropy. The assigned probability values to a graph are based on partition methods by using several graph invariants such as vertices, edges, vertex degree, and distances have been used. Bonchev-Trinajstic index (Ernesto and Bonchev, 2014) is one such information index. Dehmer developed partition-independent information-theoretic measures for graphs (Dehmer et al., 2013).

7. *2D Matrix-based indices*: These descriptors are calculated based on the elements of so-called graph-theoretic matrics (Mihalić and Trinajstić, 1992)using several algebraic operations. The Balaban-like indices inferred from the adjacency matrix are important examples of this category.

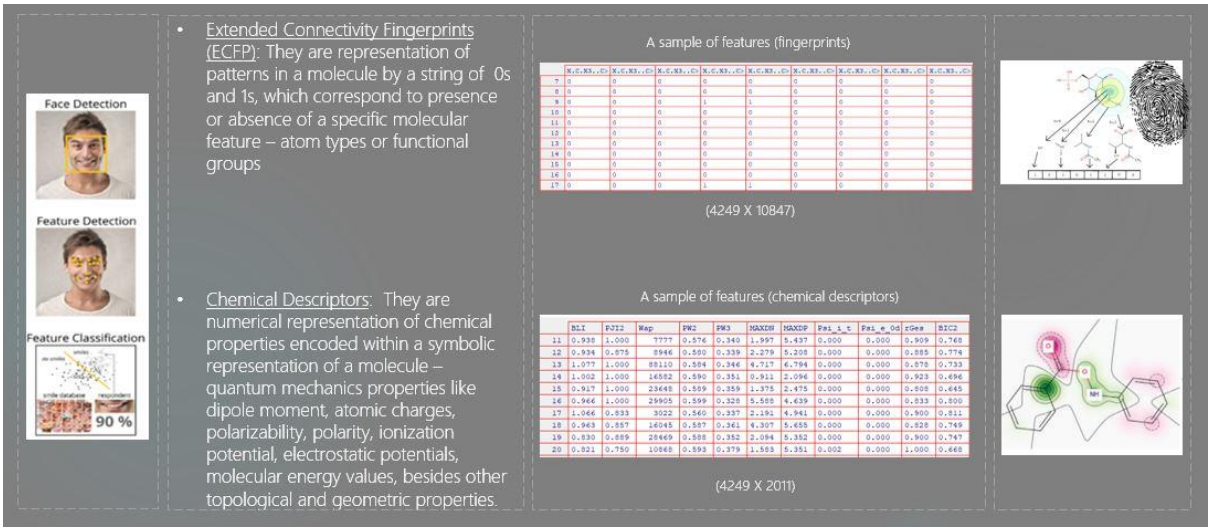


Figure 2.2 Computing Features|
The figure describes the two approaches in computing features of chemical compounds: 1. Extended Connectivity Fingerprints (top) 2. Chemical Descriptors (bottom). On the left is an intuitive visual of how facial features are used in facial recognition, in the middle is a sample of data with rows representing records and columns representing features and on the right is the depiction of a binary representation of fingerprints of presence or absence of a chemical group and numerical representation of theoretical quantum mechanical description of chemical properties of a compound.

2.2.3 Feature Engineering – Full Model vs. Reduced Models

Features are the numeric or categorical representation of the raw data. Feature engineering is the process of formulating the most appropriate features given the data, the task, and the model to make predictions. Feature engineering alludes to the task of computing features as in graph-theoretic invariants representing chemical structures and also extracting features from raw pixels, as in image processing. There are several aspects of feature engineering that are relevant to modeling tasks. Understanding the feature distribution of the individual features is critical in the detection of information redundancy, when features are highly correlated and erroneous data such as outliers. Models with smooth function are sensitive to the scale of feature inputs, for example, k-means clustering, nearest neighbor methods, radial basis function (RBF) kernels (Zheng and Casari, 2018). The distribution of the input features matters in some models more than others. For instance, the training process of a linear regression model assumes the prediction errors are distributed like Gaussian. Feature transformation, which is also referred to as normalization is another aspect of feature engineering. For example, log transforms is an example of a family of transformations know as power transform used for variance stabilizing for correcting heavy-tailed distribution that places probability mass in the tail range compared to Gaussian distribution. Standardizing features so that they are centered around 0 and a standard deviation of 1 is important in many machine learning algorithms. Intuitively, one can make a connection

between feature set to *gradient descent* (first-order optimization algorithm often used in logistic regression, Support Vector Machines (SVM) and Neural nets) on weight updates, which would suffer latency, must the values of the features x_j be on different scales (Raschka, 2014a).

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i (t^i - o^{(i)}) x_j^{(i)}$$

Such that $w_j := w_j + \Delta w_j$, where η is the learning rate, t is the target class, and o the actual output. In fact, the tree-based classifiers are the only ones that are not sensitive to feature scaling.

2.3 Modeling Work Flow

The overall modeling workflow begins with and involves the following steps: retrieval of data from the databases, pre-processing, processing, exploratory data mining, data normalization, data partition for training and testing, creating sparse matrices, building the model using different algorithms (Logistic Regression, Support Vector Machine (SVMs), Naïve Bayes, Decision Trees – CART, Random Forest, XGBoost), cross-validation, testing model performance, Optimize model hyperparameter tuning, and evaluating model performance on test data with various metrics. In a typical large-scale deployment, the prototype model is scaled. (see Figure 2.3 for details)

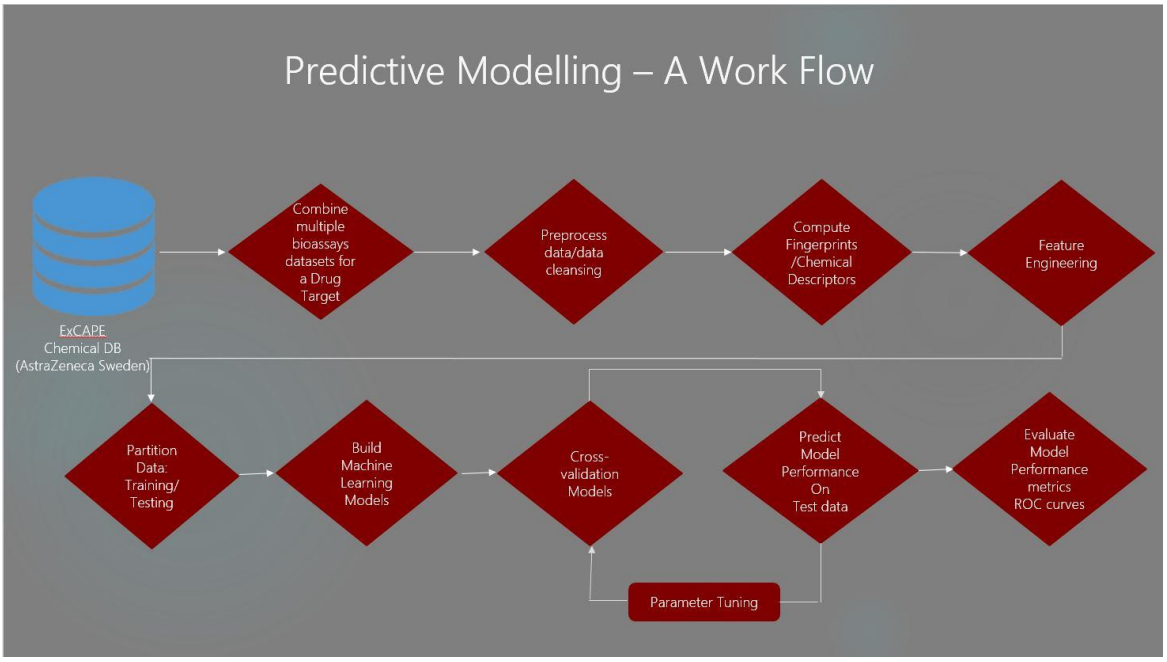


Figure 2.3 The Complete Modeling Workflow

This is a schematic of the complete modeling workflow from preprocessing, through modeling and evaluation of model performance.

2.4 Evaluation Methods

The first requirement for comparing performance between learning models is to ascertain methods or criteria for evaluation or benchmarking. This section will present a number of common statistics for measuring the performance of machine learners. The statistics will range from simple model accuracy, misclassification error, to measures of specific characteristics of the model like cross-validation, and Receiver Operator Curves (ROC).

2.4.1 Confusion Matrix

In a typical binary classification problem, a *confusion matrix* is a 2×2 frequency table used to categorize class predictions according to how they match with class assignments in the reference data. By convention, the columns are the reference, and rows are the predictions. There are instances, where columns are rows are flipped. The two classes that occupy the cells on the diagonal are labeled *True positive* and *True Negative*. The off-diagonal cells are empty. Upon prediction, the numbers shuffle and they now occupy the off-diagonal cells and they are labeled *False Negative* and *False Positive*. If the prediction was 100% accurate then the off-diagonal cells will be empty, which rarely happens (Bishop, 2006; Hastie et al., 2009; Kuhn and Johnson, 2013).

True-positive (TP): The original class is *positive* and the prediction is *positive*.

$$TP = n_{11} = |\{x_i \,|\widehat{y_i} = y_i = c_1\}|$$

True-negative (TN): The original class is *negative* and the prediction is *negative*.

$$TN = n_{22} = |\{x_i \,|\widehat{y_i} = y_i = c_2\}|$$

False-positive (FP): The original class is *negative* and the prediction is *positive*.

$$TP = n_{12} = |\{x_i \,|\widehat{y_i} = c_1 \; and \; y_i = c_2\}|$$

False-negative (FN): The original class is *positive* and the prediction is negative.

$$TN = n_{21} = |\{x_i \,|\widehat{y_i} = c_2 \; and \; y_i = c_1\}|$$

2.4.2 Accuracy vs. Error (Misclassification)

There are many measures of performance in machine learning that have been developed for specific purposes. Accuracy is perhaps the simplest one, describing the success rate of a prediction. It is defined as the proportion of correct classifications in relation to all classifications made

Accuracy $= (TP + TN)/(TP + TN + FP + FN)$
Error or Misclassification $= (FN + FP)/(TP + TN + FP + FN)$

The terms $TP, TN, FP,$ and FN refer to the number of times the predictions fell into each of these categories.

2.4.3 Sensitivity, Specificity, False Negative Rate, False Positive Rate, Precision

Sensitivity: True Positive Rate

The true positive rate, also called *sensitivity* or *recall*, is the fraction of correct predictions with respect to all points in the positive class:

$$TPR = recall_P = \frac{TP}{TP + FN} = \frac{TP}{n_1}$$

Where n_1 is the size of the *positive* class

Specificity: True Negative Rate

The true negative rate, also called *specificity*, is the fraction of correct predictions with respect to all points in the negative class:

$$TNR = recall_N = \frac{TN}{TN + FP} = \frac{TN}{n_2}$$

Where n_2 is the size of the *negative* class

False Negative Rate

The false-negative rate, is the fraction of correct predictions with respect to all points in the positive class:

$$FNR = \frac{FN}{FN + TP} = \frac{FN}{n_1} = 1 - sensitivity$$

False Positive Rate

The false-positive rate, is the fraction of correct predictions with respect to all points in the negative class:

$$FPR = \frac{FP}{FP + TN} = \frac{FP}{n_2} = 1 - specificity$$

Precision

The precision is the fraction of correct predictions with respect to all points in the positive class:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{FP}{n_1}$$

2.4.4 Cross-Validation (CV)

It is a standard technique in model validation in machine learning. This is accomplished by partitioning the dataset $\boldsymbol{D}$ into K chunks called folds, the procedure is referred to as K-fold cross-validation. The training set R, is formed by $K - 1$ chunks, and the last chunk serves as the validation set V. In cross-validation, we iterate through all the combinations of assignment to R and the procedure is repeated for all the K partitions for the validation set, and the performance of the model is averaged for K runs.

The performance of the predictor f after training is assessed on the validation set V. We could elaborate this as follows, for each partition k the training data $R^{(k)}$ produces a predictor $f^{(k)}$, which is then applied to the validation set $V^{(k)}$ to compute the empirical misclassification risk $R(f^{(k)}, V^{(k)})$. After cycling through all the partitions, we compute the average generalization error of the predictor. Cross-validation approximates the expected generalization error (Deisenroth et al., 2020).

$$\mathrm{E}_v[R(f, V)] \approx \frac{1}{K}\sum_{k=1}^{K} R\left(f^{(k)}, V^{(k)}\right)$$

Cross-validation can be computationally expensive, especially with very large datasets, however, given the present compute scalability this may not be an issue.

2.4.5 ROC Curve – Vizualization of Model Performance

The Receiver Operator Characteristic (ROC) curve is by far, the most critical measure for assessing the performance of classifiers when there are two classes. It is a graphical plot that illustrates the diagnostic ability of a binary classifier, as its discrimination threshold is varied. Visualizations are often helpful for human comprehension.

The ROC curves are created by plotting the *false-positive rate* (1-specificity) x-axis against the *true positive rate* (sensitivity) y-axis. The fundamental basis for the ROC curve is the probability distribution of *true positives* vs. *true negatives* in the sample at different thresholds i.e, cut-off values that separate samples into two discrete classes. Under ideal conditions, when the classes are completely separated, the distributions do not overlap, otherwise, they begin to overlap. These probability distributions change depending on the elected thresholds that impose binning boundaries that result in spillage of *true-positives* into *the negatives* bin and therefore acquire the label of a *false-negative*. Likewise, *true-negatives* spill into the *positives* bin and get the label *false-positive*. It can also be thought of as a plot of the power as a function of the *Type 1 Error* of the decision rule (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities). ROC curves can be generated by plotting the cumulative distribution function (area under the probability distribution from $-\infty$ to the discrimination threshold) of the detection probability on the y-axis versus the cumulative distribution function false positive probability on the x-axis. A ROC space is defined by FPR and TPR as x and y axes. Since TPR is equivalent to sensitivity and FPR is equal to 1-specificity, each prediction is an instance of a confusion matrix and represents one point in the ROC space.

The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100 sensitivity (no false negatives) and 100% specificity (no false positives). The point(0,1) is also called a perfect classification. A random guess would give a point along the diagonal line, the so-called *line of no discrimination*. An important feature of ROC curves is that they can naturally be reduced to a single quantitative, index measure for assessing the performance of the model is the Area Under the Curve (AUC) (Junge and Dettori, 2018).

2.5 Model Complexity and Generalization

Model complexity in machine learning models refers to the number of *degrees of freedom* in a learned model, often measured as the number of *adjustable weights* or *parameters* in the architecture of doing the learning. A very large number of trainable parameters with limited training samples leads to the 'memorization' of patterns in the data in such exhaustive detail (including aberrant noise), specific to examples in the training

set but not broadly representative of the larger population, from where samples are drawn. Consequently, this leads to building complex models that perform well on the training set but do very poorly on out-of-sample data. Thus, they suffer from not being readily adapted for generalization (Sejdinovic, 2015).

2.5.1 Bias Variance Tradeoff

In general, machine learning models display *bias-variance tradeoff*, a property where predictive models with lower bias in parameter estimation have a higher variance of the parameters across the sample, and vice-versa. The *bias-variance* problem is the tension of trying to simultaneously minimize these two sources of error that often, interfere with the ability of the supervised learning algorithms from generalizations beyond the training set (Bishop, 2006; Bruce and Bruce, 2017; Kohavi and Wolpert, 1996).

Intuitively, the *bias* of a classifier refers to the systematic deviation of its predicted decision boundary from the true decision boundary. In other words, it measures how good is the predicted value to the actual value. On the other hand, the *variance* measures how different are the predictions of each of the bootstrapped training sets from the actual values, which refers to the deviation among the learned decision boundaries over different training sets (Bruce and Bruce, 2017).

Formally, given a training set, $\boldsymbol{D} = \{x_i, y_i\}_{i=1}^{n}$, comprising of n points $x_i \in \mathbb{R}_{i=1}^{d}$ with the corresponding class y_i, a trained model M predicts the class for a given test point x. A commonly used *loss function* for classification, which assigns a value of 0 or 1.

$$\boldsymbol{L}\big(y, M(x)\big) = \begin{cases} 0 \; if \; M(x) = y \\ 1 \; if \; M(x) \neq y \end{cases}$$

If the prediction is correct, the value is zero and one otherwise. Another commonly used loss function is the *squared loss*, defined as

$$\boldsymbol{L}\big(y, M(x)\big) = (\mathrm{y} - M(x))^2$$

Minimizing the *cost function* is the goal of any optimum classifier. Formally, $\boldsymbol{M}$ depends on the training set, given the test point x, we denote the predicted value as $\boldsymbol{M(x, D)}$, $\boldsymbol{E_y}$ is the expectation loss of class $\boldsymbol{y}$ and given as:

$$E_y[\boldsymbol{L}\big(y, M(x)\big)|x, D] = E_y[(\mathrm{y} - M(x, D))^2|\mathrm{x}, \mathrm{D}]$$

To encapsulate the idea, an ideal classifier is one that balances the *bias-variance* trade off and can be graphically represented as one with low *bias* and low *variance*.

2.5.2 Regularization

Briefly, regularization is a technique for reducing the model complexity and expanding its ability for generalizability. It does this by restricting the *degrees of freedom* in the model. To set the context, the original goal in an unregularized model is to minimize the *cost function* i.e, we want to find feature weights that correspond to the global minima of the first-order optimization of a differentiable function (using gradient descent). In this context, we add a *penalty* term to the cost function (J). Intuitively, we can think of regularization as a penalty against model complexity. Increasing the regularization strength penalizes *large* weight coefficients. The main goal is to attenuate the *noise* while modeling the *pattern* in the data, which translates to preventing *overfitting* the model. Overfitting leads to erosion of the generalization of the model. We can also think of regularization, from *bias-variance* tradeoff, as adding (or increasing the) *bias,* if the model suffers from high *variance*. On the other hand, high bias leads to model *underfitting* (Alex Smola and S.V.N. Vishwanathan, 2008; Deisenroth et al., 2020; Kuhn and Johnson, 2013; Sejdinovic, 2015). A generic *cost function* may be written as:

$$\text{Cost}_{w,b} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

Regularization is a *loss function* with an additional penalty term, it turns out this is the L2 norm.

$$\text{Cost}_{w,b} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) + \frac{\lambda}{n}\ w_j^2$$

The goal is the minimization of the loss function and the penalty term, by minimizing the combined expression we are seeking solutions where w is closer to 0. The hyperparameter λ, adjusts the tradeoff between low training loss and having low weights. Regularization constrains the gradient descent by *weight shrinking* or decreasing the value of the feature weights (Hastie et al., 2009; Raschka, 2019).

2.5.3 Ridge

This is by far the most common type of regularization, where the regularizer is a squared L2 norm $||w||^2$, this is called L2 or Ridge regularization (Hoerl and Kennard, 1970). The imposition of this constrain results in decreasing the value of the coefficients and tend toward zero but does not become absolute zero, as the value of lambda becomes larger. Shrinking the coefficients leads to a decrease in the variance. The net effect is decreasing the model complexity. It is most commonly used in regression implementations like linear regression and logistic regression. Logistic regression use L2 regression by default. It is also widely adopted to other algorithms like the perceptron. The combined function is still a convex function.

$$\text{Cost}_{w,b} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) + \frac{\lambda}{n} \sum_{j} w_j^2$$

2.5.4 LASSO

Unlike ridge regularization, LASSO, which stands for Least Absolute Shrinkage and Selection Operator is a L1 norm regularizer (Tibshirani, 1996). Thus, this penalty term to the *loss function* is simply a summation of the absolute weights of the features. Interestingly, this penalty term shrinks the value of some of the feature coefficients to absolute zero. Consequently, this leads to the elimination of a few features. Thus LASSO is useful for dimensional reduction. Enforcing sparsity constraints on weights can lead to simpler and more interpretable models. LASSO which may be expressed as:

$$\text{Cost}_{w,b} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) + \frac{\lambda}{n} \sum_{j} |w_j|$$

2.5.5 Elastic Net

Elastic Net emerged as a regularizer that combines the attributes, both from ridge and LASSO regularization. This was also an attempt to address some shortcomings of LASSO, whose feature selection was dependent on the data and is often unstable (Zou and Hastie, 2005). Elastic Net outperforms the LASSO and has a similar representation of sparsity. The other interesting feature in Elastic Net is that its highly correlated predictors tend to be in or out of the model altogether. The Elastic Net may be written as:

$$\text{Cost}_{w,b} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) + \frac{\lambda}{n} \sum_{j} w_j^2 + \frac{\lambda}{n} \sum_{j} |w_j|$$

2.5.6 Random Neuron Drop

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on the training data. Neural networks and deep neural nets with a large number of parameters are very powerful machine learning systems (Hinton et al., 2012; Srivastava et al., 2014). However, these implementations are highly susceptible to overfitting issues because of the access to a large pool of parameters. In this technique, a single model can be used to simulate a large number of different network architectures by randomly dropping out nodes during training. Dropout is a case of simulation a sparse activation from a given layer, which forces the network to learn from sparse representation, which in effect, emulates model activity regularization, conceptually akin to LASSO. The dropout method has been successfully used in multiple classification tasks ranging from computer vision, speech recognition, and text classification (Srivastava et al., 2014). In this thesis, we have applied this regularization to chemical representation data.

Chapter 3: Evaluation of Algorithms

"All models are wrong, but some are useful." – George E.P. Box

The *core objective* of this thesis is to evaluate some of the machine learning algorithms that can be applied in drug discovery. The choice of algorithms must be based upon the problem at hand and algorithms that fulfill the criterion of *fitness-for-purpose*. Many practical problems require an exhaustive search through the solution space, which are represented as combinatorial structures such as permutations, combinations, set partitions, integer partitions, discrete classes, and trees. The combinatorial algorithms and graph-theoretic models have applications in chemistry, in the enumeration of *isomorphic* or *equivalent objects* in the computation of molecular descriptors and fingerprints in predicting biological activity. The ability to predict a molecule's biological activity by computational means is central to drug discovery. Some of the more sophisticated machine learning algorithms include Support Vector Machine (SVM), Naïve Bayes, Artificial Neural Nets (ANN), and Ensemble Classifiers which are leveraged in addressing problems of set partitions or classification. A case can be made, that theory and application are mutually dependent. The algorithmic community cannot be detached from the reality of the applications, in by making theoretical assumptions, suitable for proving theorems and designing new algorithms. As much, practitioners of 'application' must be sufficiently sophisticated in *math* under the hood for critical interpretations. The bulk of the emphasis in the present study, will be on the application of these algorithms in solving practical problems in cheminformatics of drug discovery and much less on the algorithmic theory (Nayak and Stojmenovic, 2008).

3.1 Exploratory Visualizations

Exploratory visualizations are visual interrogations of initial data before embarking on the task of model development, with the explicit mandate to uncover patterns, pick anomalies, check missing data, probability distributions, collinearity diagnostics, and outliers among other things through aggregated summary statistics and graphical representations.

3.1.1 Distributions and data normalization

Sample distributions give a quantitative characteristic of the variables in the data, such as sample mean, sample variance, sample standard deviation, sample skewness, sample kurtosis, and sample outliers. Outliers are sample data values that are outside the areas of the distribution and correspond to areas of probability density function with low density. Box plots are useful in locating outliers in the data. Sample feature distributions of representative sample features and the box plots are shown (Figure 3.1). In many datasets in the study, the features are in the high dimensional space (ranging from 200 – 1600). Another common aspect of the data is normalization, whose goal is to transform features to be on the same scale. Normalization helps in improving the model performance and training stability of the model. Many machine learning algorithms like logistic regression, support vector machine (SVM), and neural nets use first-order optimization techniques

like *gradient descent* and *stochastic gradient descent*. Normalization of features in the data, usually help in faster convergence of the optimization algorithms to the local minima. From the plots, we see that the data more or less behaves like a typical normal distribution. We also explored different normalization techniques, for example, range transform, Z-transform, Box-Cox transform (Box and Cox, 1964), and Yoe-Johnson transform (Yeo and Johnson, 2000), and concluded that data normalized by range transform produced better performing models. Thus, range transform was uniformly applied to all the datasets with continuous variables in the present study.

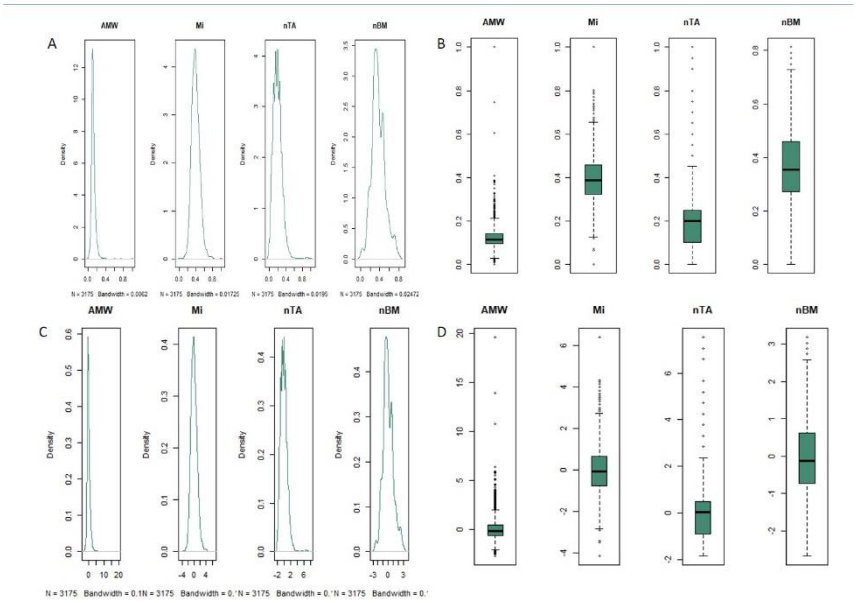


Figure 3.1 Feature Distributions and Box-plots
(A) Normal distributions of data normalized by range transform; (B) Box-plots of range transform; (C) Normal distribution of data normalized by Z-transform; (D) Box-plots of Z-transform.

3.1.2 Collinearity Diagnostics

Multicollinearity is a statistical phenomenon in which the feature variables in a machine learning model are highly correlated. The existence of high collinearity among the feature variables inflates the variances of the parameter estimates of the model and consequently results in incorrect inferences about the relationship between the explanatory variables and the response variable. The correlation matrix provides direct visualization of the collinearity in the data. As a standard preprocessing practice, we remove highly correlated columns in the data (drop one of them). To demonstrate, we selected a random sample of 100 features in a sample dataset free of collinearity ($r >= 0.67$) as seen in the correlation matrix plot (see Figure 3.2 with reduced intensity except along the diagonal which are correlations of a feature with itself).

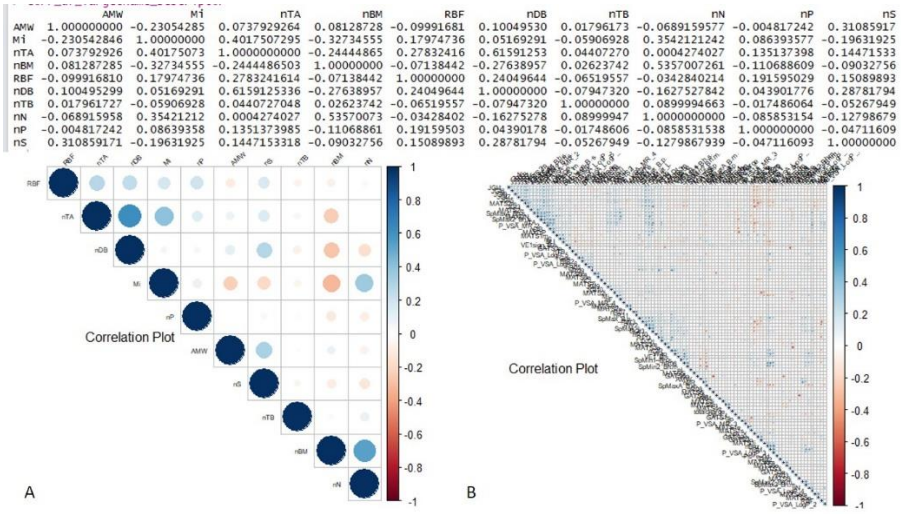


Figure 3.2 Feature Correlation Plot

Highly correlated features correlated (r >= 0.67) were removed. The correlations plots of (A) a sample of 10 random features and (B) a sample of 100 random features are shown. The positively correlated features are color-coded blue and the negatively correlated features are red. From the intensity of the dots, it is clear the highly correlated columns are removed. The diagonal represents the correlations of a feature with itself, as expected, which are highly correlated. The plot is symmetric along the diagonal, only the upper half is shown.

3.1.3 Missing Data and Statistical Imputations

Missing data (NA) is often a vexing problem in building machine learning models, real-world data often suffers from these limitations. The inability to spot these anomalies in the early stage of exploratory data analysis can lead to building unstable models. As a preprocessing routine, we have carefully removed these inconsistencies in the datasets. There are many statistical imputations techniques (Little and Rubin, 2020) and libraries available in R to remedy the situation, however, we did not have such issues with our datasets that called for such remediation. Besides, we ensured that *classes* are approximately balanced in our datasets before we built any classifiers.

3.1.4 Sparse Matrix speeds up Computations

The use of large sparse matrices is common in machine learning. Matrices that mostly contain zero elements are called sparse matrices as opposed to dense matrices where most of the elements are non-zero. A scheme, wherein zero elements are assumed to be unspecified, can perform faster operations and use less memory than the corresponding dense matrix. Since we used both sparse and dense matrices in building models, we made some performance measurements using these two data formats (see Figure 3.3). Sparse matrices speed up computations.

Figure 3.3 Performance Metrics Dense vs. Sparse Matrix
(A) A sample dense matrix; (B) A sample sparse matrix; (C) A comparative metrics for a computational task for the same data in two different data formats - dense vs. sparse matrix.

3.2 Logistic Regression - Background

Logistic Regression is a classical probabilistic linear method for binary classification problems. This class of classifiers belongs to the larger family of Generalized Linear Models (GLM) (McCullagh and Nelder, 1983). Logistic Regression has a lot in common with linear regression, which is used for the tasks of predicting continuous numerical values. While Logistic regression is reserved for tackling classification problems. Both techniques model the target variable with a line (or hyperplane), depending on the number of dimensions of the input. Importantly, linear regression fits a line to the data, which we can use to directly predict the target numerical value, while logistic regression fits a line that best separates two classes (Cramer, 2002).

For a typical machine learning model, for instance, a linear regression model, the input data is denoted by $\boldsymbol{X}$ with n samples and the output is denoted by $\mathbf{y}$ with one output for each input. The prediction of the model is denoted as $\mathbf{y}$.

$$\hat{y} = \text{model } (\boldsymbol{X})$$

The model is defined in term of parameters called coefficients (β_i), one coefficient per input and an additional coefficient (β_0) for the intercept or bias term.

$$\hat{y} = \beta_0 + \beta_i x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{3.2.1}$$

Logistic Regression describes how a class labels of a binary (“0” or “1”) response (dependent)variable is associated with a set of input explanatory (independent) variables (categorical or continuous). At the core of

logistic regression is the adoption of a *sigmoidal function*. It is an *S*-shaped curve that takes real numbers and maps them into values between 0 and 1. This function has some attractive features

It bounds the values to the limits on the lower end to 0 and the upper end to 1. Therefore it does not violate the principle of probability, this acts as a 'link' function. The derivative of a sigmoid curve is mathematically convenient to determine the slope at any point on the curve.

$$S(x) = \frac{1}{1+ e^{-x}} \tag{3.2.2}$$

In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model. Mathematically, a binary logistic model has dependent variable/s with two possible outcomes, for instance, pass/fail, success/failure, active/inactive, etc., and these two values are labeled "0" or "1".

The logistic transformation can be explained as follows: *p* as the proportion of observations with an outcome of 1, then *1-p* is the probability of an outcome of 0. The ratio of *p/(1-p)* is called *odds* and *logit* is the logarithm of the odds or just *log odds*. We can use the *logit* as a 'link' function and thus the logistic regression model can be written as.

$$logit(p) = \ln(\frac{p}{1-p}) = \beta_0 \ + \beta_{\mathrm{i}} x_1 + \beta_2 x_2 + \ldots + \beta_{\mathrm{n}} x_n \tag{3.2.3}$$

In the logistic model, the *log odds* (the logarithm of the odds) for the labeled class "1" is a linear combination of one or more independent variables (predictors). The logistic function converts log-odds to probabilities. The goal of predictive models is to infer a rule to predict the response. The outcome $y = \{0,1\}$ with the given data X, for example, the possibility that X belongs to a particular class: $\Pr(y = 1|x)$. Logistic Regression model $\Pr(y = 1|x)$ using the logistic function.

$$\Pr(y = 0|x;\ \theta) = \frac{\exp(\beta_0 \ + \beta_{\mathrm{i}} x_1 \ + \beta_2 x_2 \ + \ \ldots \ + \beta_{\mathrm{n}} x_n)}{1 + \exp(\beta_0 \ + \beta_{\mathrm{i}} x_1 \ + \beta_2 x_2 \ + \ \ldots \ + \beta_{\mathrm{n}} x_n)} \tag{3.2.4}$$

$$\Pr(y = 1|x;\ \theta) = 1 - \left[\frac{\exp(\beta_0 \ + \beta_{\mathrm{i}} x_1 \ + \beta_2 x_2 \ + \ \ldots \ + \beta_{\mathrm{n}} x_n)}{1 + \exp(\beta_0 \ + \beta_{\mathrm{i}} x_1 \ + \beta_2 x_2 \ + \ \ldots \ + \beta_{\mathrm{n}} x_n)}\right] \tag{3.2.5}$$

Where θ represent model parameters: β_0 *and* β_i are two unknown coefficients of the regression model which can be estimated by maximizing the likelihood function:

$$L(\beta_0, \beta_{\mathrm{i}}|y) = \ p_i^{y_i} * 1 - p_i^{((1-y_i)} \tag{3.2.6}$$

We can get the joint probability by simply multiplying the individual distribution. Thus, the joint probability of the data (the likelihood) is given by

$$L(\beta_0, \beta_{i|}|y) = \prod_{i=1}^{n} p_i^{y_i} * 1 - p_i^{((1-y_i)} \tag{3.2.7}$$

Because the logarithm is a monotonically increasing function of its argument, the maximization of the log of a function is equivalent to the maximization of the function itself. Taking the log not only simplifies the subsequent mathematical analysis, but it also helps numerically because the product of a large number of small probabilities can easily underflow the numerical precision of the computer, and this is resolved by computing the log-likelihood.

$$l = \log(L) = \sum_{i=1}^{n} [y_i * \log(p_i)p_i + (1 - y_i) * log(1 - p_i)] \tag{3.2.8}$$

Maximization of log-likelihood is often implemented using efficient derivative-based numerical optimization algorithms techniques like gradient descent, Broyden-Fletcher-Goldfarb-Shano (BFGS) algorithm, conjugate gradient method, and Newton-Raphson method (Nesterov, 2018).

3.3 Logistic Regression - Implementation & Experiments

Logistic Regression was implemented in the statistical programming language R. The generalized linear models (GLMs) are a broad class of models that include linear regression, ANOVA, log-linear models, Logistic Regression, etc. The generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for a response variable that has error distribution models other than the normal distribution. The GLM generalizes linear regression by allowing the linear model to be related to the response variable via a *link function*. Several R libraries were used in the implementation that includes *glmnet* for binomial models (Friedman et al., 2010), *SparseM* (Koenker and Ng, 2003) for sparsity representation, *ggplot2* (Wickham, 2009) for plotting besides using native R plotting utilities.

3.3.1 Logistic Regression (LR) Model Development and initial Validation

After the initial processing of the following datasets: JAK2, MMP2, STAT3, ESR1, and BRCA1 we have developed several logistic regression models. However, for brevity, we will highlight the results of the JAK2 analysis. One of the key concepts in generalized linear models is *deviance* (McCullagh and Nelder, 1983; Portugués, 2020). Intuitively, it measures the deviance of the fitted model w.r.t. a perfect model, which is also called a *saturated model* that perfectly fits the data. To expand, the fitted response $(\hat{Y}_i)$ are the same as observed responses (Y_i). In case of logistic regression the models can be described as:

$$\hat{Y}[c = 1|X = x_{1,}x_{2,}\dots, x_n] = Y_i, \qquad \text{where } i = 1,2,\dots n$$

Formally, one could define deviance as the difference of the log-likelihoods between the fitted model $l(\hat{\beta})$, and the saturated model l_s. By invoking the canonical *link* function is used, this corresponds to setting $\theta_i = g(Y_i)$. Thus deviance is then defined as:

$$D \coloneqq -2[l(\hat{\beta}) - l_s]\,\phi.$$

While we try to maximize the log-likelihood $l(\hat{\beta})$, yet it is always smaller than l_s. As a consequence, the deviance is always larger or equal to zero. Being zero, if and only the fit of the model is perfect, which is highly unlikely.

The fraction deviance is a measure of how much unexplained variation there is in the Logistic model – the higher the fraction explained, the more accurate the model (See Figure 3.4). From these results, it is clear the model has captured a greater than 90% proportion of variability in the response variable. In other words, the model predictors have accounted for a large proportion of the explained variance. The plots are also illustrative in informing that a large proportion of the explained variance is accomplished by a smaller subset of features in the model, leading to the possibility that a reduced model (fewer features) might fit the data as well as the full model (all the features in the high dimensional space included). We also examined the effects of the regularization penalty on the model coefficient. It is clear from these plots that as the model is penalized that there is a proportional shrinkage in the model coefficients (see figure 3.4). In one of the later sections, we will describe in some detail the development of recursive logistic regression, which describes the full model and a family of reduced models.

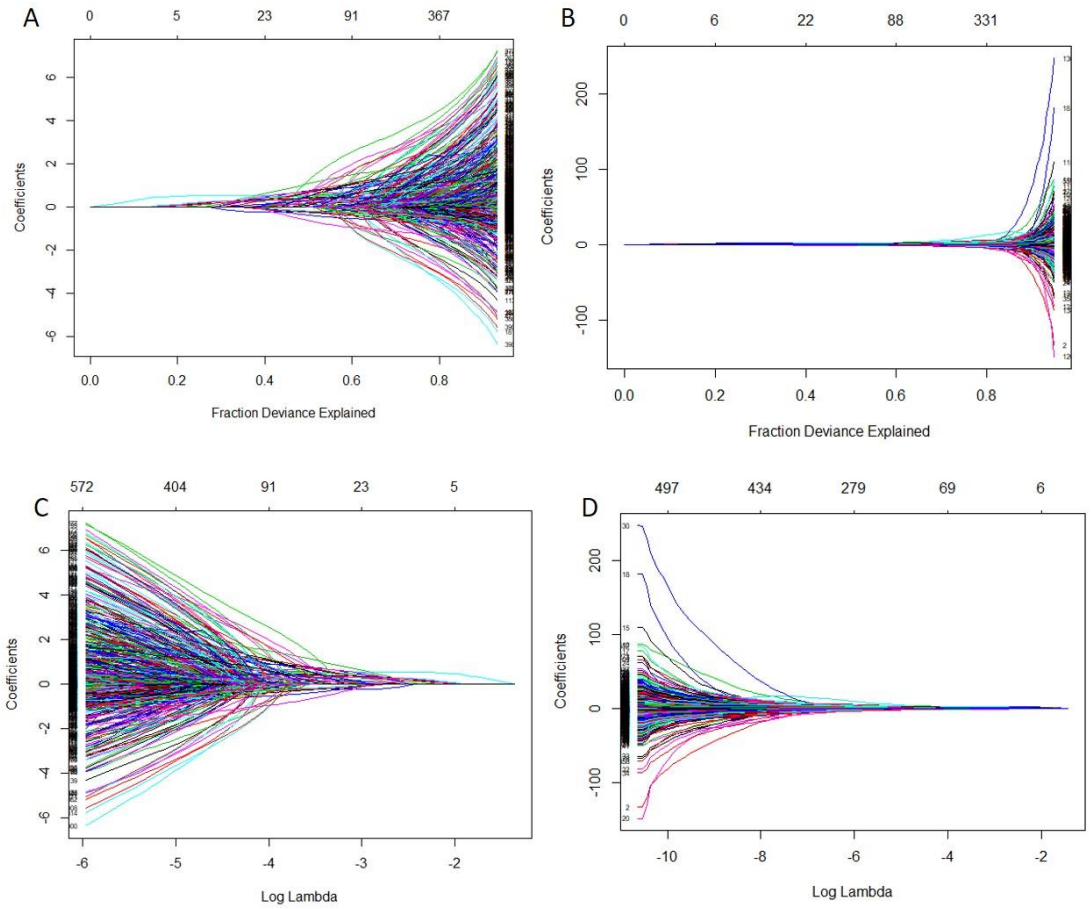


Figure 3.4 LR Model Fraction Deviance Measurement and log lambda
The top panel shows the percent of the variance in the response variable is plotted on the *x-axis* and the coefficient values are plotted on the *y-axis*. Each line represents a feature in the model and the number of non-zero features are listed above. (A) Fraction deviance with Fingerprint data (B) Fraction deviance with descriptor data. The bottom panel shows the effect of model regularization penalty lambda on the coefficient values (C) Effects with fingerprint data (D) Effects with descriptor data. Each line represents a feature in the model and the number of non-zero features are listed above.

3.3.2 Logistic Regression (LR) Model Cross-validation

We evaluated the LR models using external 10-fold cross-validation (Deisenroth et al., 2020). Each of the training set records was randomly partitioned into 10 equal parts, with each record in a partition has the class label (active vs. inactive). The model was built with 9 parts and tested on the "left-out" fraction. Likewise, the model-building exercise is sequentially iterated ten times with different 9 parts and tested on the different "left-out fractions", for cross-validation schema (see Figure 3.5 inset). The plots (see Figure 3.5) show the results of such cross-validation both with the fingerprint data and descriptor data. While performing the cross-validation, we also invoke the model regularization penalty to inform the effect of the regularization on the misclassification error. Here were are seeking the value of lambda that produces the most optimal model among several models that were generated. Each data point is the representation of an average 10 fold cross-

validation, and several such cross-validations are performed for a range of lambda values for tuning the model. The plot also shows the number of non-zero features at a given lambda value.

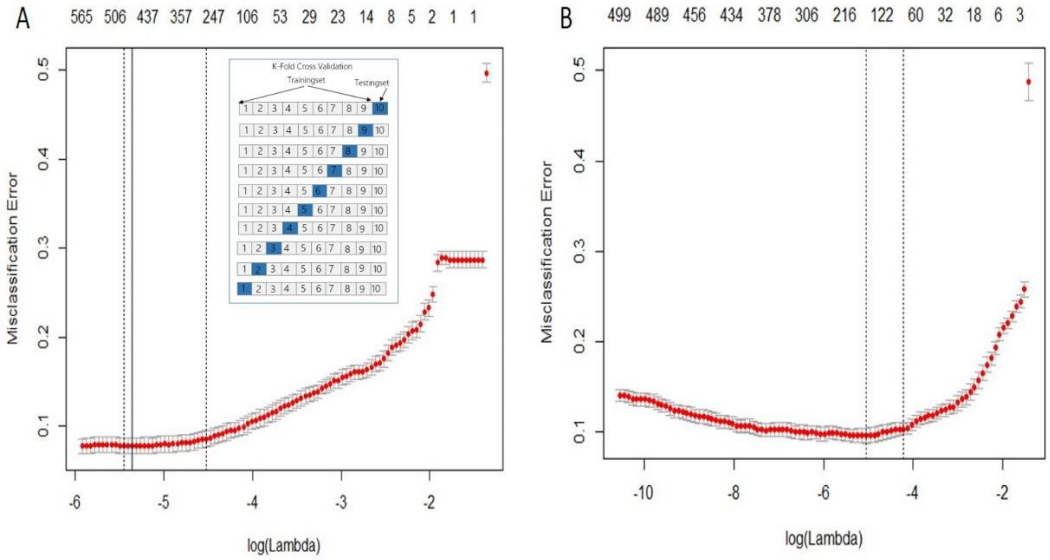


Figure 3.5 LR Misclassification error in a 10-fold cross-validation
Each point on the graph represents a mean classification error from 10 models iterated over ten-folds (see inset). The plot also shows the effect of the penalty factor on the misclassification error. The numbers on the top show the number of features in each model at various lambda values. (A) Fingerprint data (B) Descriptor data.

3.3.3 LR Model Performance Evaluation - Confusion Matrix

The confusion matrix is a standard table of n x n matrix used to describe the performance of a classification model on a set of test data for which the ground truth values are known. It allows for in performance evaluation of a typical classifier. It allows the visualization of the performance of the classifier. The table in Figure 3.6 gives the performance metric of logistic regression using fingerprint and descriptor data. The performance of the model was evaluated both on the training data and testing data. LR models performed exceedingly well with accuracy (>90%), the LR model fitted with fingerprint data was slightly better on the accuracy metric compared with LR models fitted with descriptor data on the testing data. As expected, accuracy with testing data will be less than with training data, on which the models are built. However, when looking only at the testing data, the results are somewhat mixed, where sensitivity scores are better with descriptor data, while specificity score is superior with fingerprint data. In this study, model sensitivity is of interest, because we have a vested interest in lowering false negatives while allowing a slight erosion in model specificity. Ideally, we also want to control the increase in false positives. We invoke model regularization techniques to prevent model overfitting i.e., models that perform exceeding well on the training data but fare very poorly on the testing data.

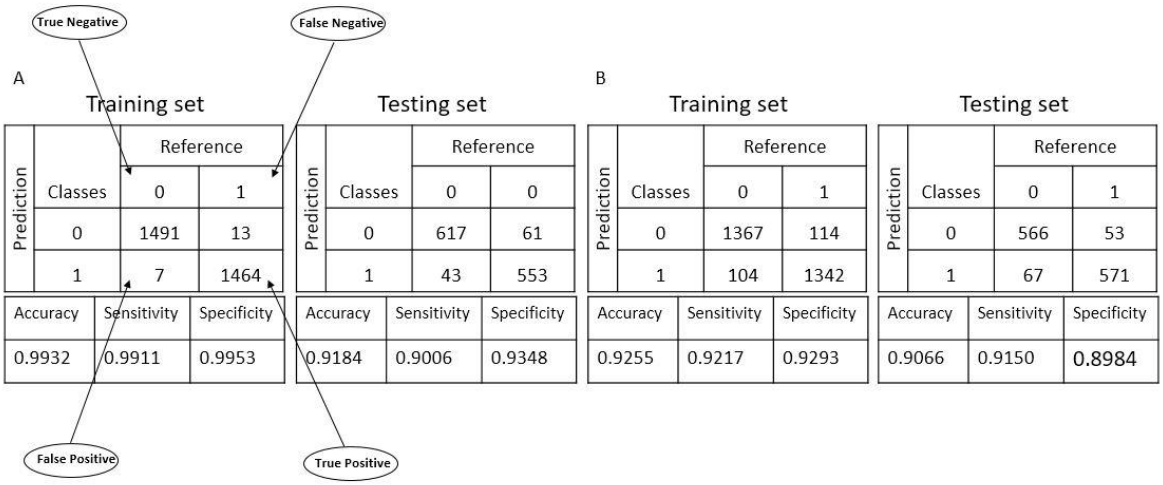


Figure 3.6 LR Confusion Matrix
A typical confusion matrix shows the model performance – ground truth (reference) vs. prediction. The cells along the diagonal in a binary classifier represent the true class (true positive and true negative) and the cells in the off-diagonal represent the misclassification error (false negative and false positive). From the confusion matrix, one can extract other model performance metrics, for instance, Accuracy, Sensitivity, Specificity, etc. (A) Represents the training and testing set for models built with fingerprint data (B) Represent the training and testing set for models built with descriptor data.

3.3.4 LR Model Performance Evaluation – ROC curves

The Receiver Operator Characteristic (ROC) curve is by far, the most critical measure for assessing the performance of classifiers when there are two classes, as we discussed earlier. It is a graphical plot that illustrates the diagnostic ability of a binary classifier, to discriminate *true positives* from *false-positive* as we vary the threshold i.e., cut-off values to make the call if it belongs to the *true positive* class or the *false positive* call. The first application of ROC curves in machine learning was introduced by Spackman who demonstrated the value of ROC curves in comparing and evaluating different classification algorithms (Spackman, 1989). Here we plot the ROC curves for the LR model with fingerprint data and descriptor data (see Figure 3.7). These results demonstrate that LR models have fairly robust predictive power, as measured by AUC. In general, ROC curves are generated by plotting the cumulative distribution function (area under the probability distribution from $-\infty$ to the discriminating threshold).

The class prediction, for each instance, is often based on the continuous random variable x, which in the case of logistic regression is the estimated probability. Given a threshold parameter T, the instance is classified as "positive", if $x > T$ and "negative" otherwise, x follows a probability density $f_{1(x)}$ if the instance belongs to class "positive" and $f_{0(x)}$ if otherwise. Therefore, the *true positive rate* is given by $TPR(T) = \int_T^{\infty} f_1(x)dx$ and the *false positive rate* is given by $TPR(T) = \int_T^{\infty} f_0(x)dx$. ROC curve plots parametrically TPR versus FPR with T as the varying parameter. The AUC for fingerprint and descriptor models on training and testing data

are reported in Figure 3.7. this is in fair concordance with the model accuracy described earlier (see Figure 3.6).

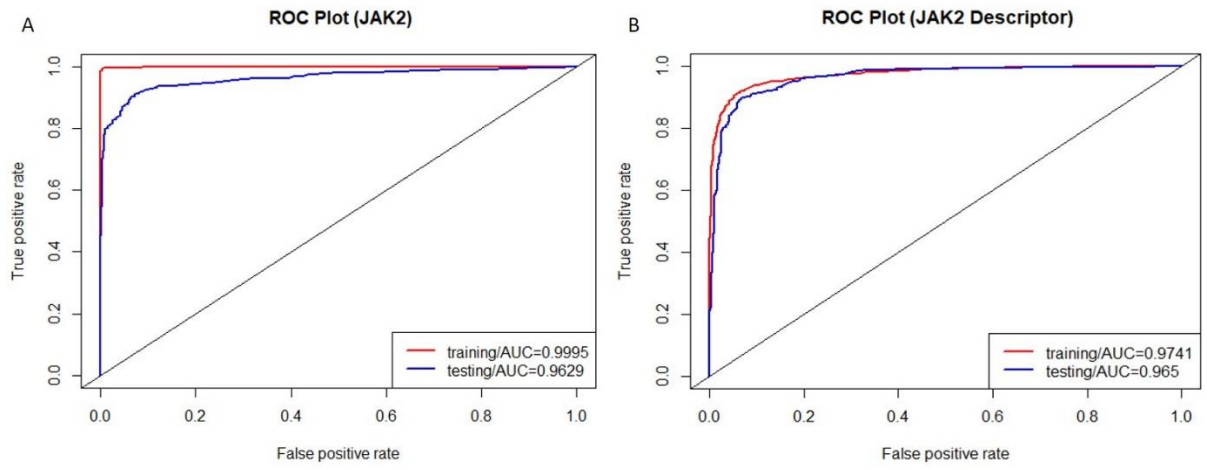


Figure 3.7 LR Model ROC Curves
The plot shows the probability distribution of true positives (x-axis) to false positives (y-axis) at different thresholds i.e., cut-off values that separate samples into discrete classes. ROC curves close to the diagonal indicate that the classifier is a poor performer. The area under the curve (AUC) is shown in the inset. The colored lines indicate the model performance with training and testing data respectively. (A) Model built with fingerprint data (B) Model built with descriptor data.

3.3.5 Full vs. Reduced LR Models

Given the high-dimensional space of the feature set with fingerprint **data (**>10,000 features), we explored the possibility of feature engineering through *dimensional reduction* building a family of reduced models, and evaluate model predictive power. We present the results of the experiment of implementing *recursive logistic regression* in Figure 3.8. Staring with the dataset with all the features (>10,000), we developed a full model, after cross-validation and determining the regularization penalty factor (λ) that produced the highest performing model with minimal misclassification errors. Following this, we iteratively built a family of reduced models, each with fewer features than the earlier model, and evaluated its performance metrics. Each model went through an elaborate series of cross-validations, as described earlier. From each of the highest performing models, we extracted the most important features, ranked these features, and retained only these features, to build the next model. In the end, we developed a full model and a small set of reduced models. From the model evaluation based on the metrics of accuracy, sensitivity, and specificity, we can draw some broad conclusions: the proportion of variability in the model can be explained by a few predictors. In other words, only a small subset of features makes a large contribution to the model *weight* coefficients. Reduced models perform as well as the full model, until a certain critical threshold of important features, beyond which the performance degrades. Reduced models also have the advantage of reducing model complexity, preventing *overfitting*. Besides, it also has the advantage of reducing computational costs in building the models, especially with very large datasets.

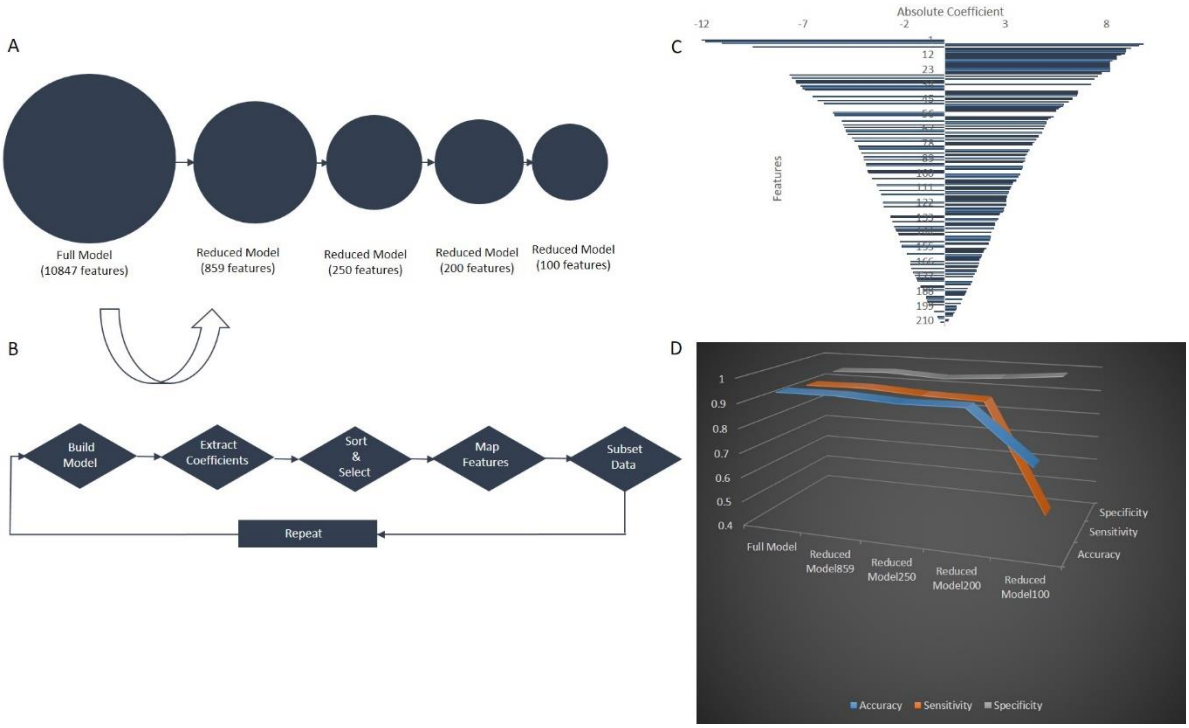


Figure 3.8 Recursive Logistic Regression
(A) Shows the graphical representation of a full model and a series of reduced models with a successive reduction in the number of features (B) Show the iterative workflow in building reduced models (C) The absolute values of the coefficients extracted from the penultimate reduced model (D) Shows the graph of the full and the reduced models and its effect on model metrics.

Table 3.1 Metrics for Full vs. Reduced Logistic Regression Models

Model	Accuracy	Sensitivity	Selectivity
Full Model (10847)	0.9399	0.9253	0.9453
Reduced Model (859)	0.9417	0.9287	0.9549
Reduced Model (250)	0.9279	0.9138	0.9418
Reduced Model (200)	0.9325	0.9077	0.9583
Reduced Model (100)	0.7316	0.4685	0.9802

3.3.6 Model Regularization – LASSO, Ridge and Elastic Net

Model regularization is motivated by the desire to reduce model complexity and expanding its ability for generalizability. It does this by restricting the *degrees of freedom* in the model (Raschka, 2019). Empirical learning of classifiers from a finite dataset is always an undetermined problem because it attempts to infer a function of any x given only examples $x_1, x_2, \ldots, x_n$. A regularization term (or regulaizer) $R(f)$ is added to the loss/cost function

$$\min_{f} \sum_{i=1}^{n} V\left(f(x_i), y_i\right) + \lambda R(f))$$

where V is the underlying loss/cost function that describes the cost of $f(x)$ when the label is y and λ is the parameter that controls the importance of the penalty term. $R(f)$ is typically chosen to impose a penalty on the complexity of *f*. Concrete notions of complexity include bounds on the vector space norm (Bishop, 2006). Early stopping can be viewed as regularization in time, intuitively, training procedures like *gradient descent* tend to learn more and more complex functions as the number of iterations increases. Likewise, enforcing sparsity constraints can lead to simpler and more interpretable models. Several approaches towards model regularization have been described in some detail (see chapter 2: Approaches).

Both L1 and L2 penalized estimation methods shrink the estimates of the regression coefficients towards zero relative to maximum likelihood estimates (Ng, 2004). The purpose of this shrinkage is to prevent *overfitting* arising due to either collinearity of the covariates or high-dimensionality. Perhaps it will be instructive to see Figure 3.4, here we clearly show in the plot, the path of shrinkage of the model coefficients as a function of the penalty controlling factor λ. Here we have implemented the L1 absolute penalty – LASSO (Tibshirani, 1996), an L2 quadratic penalty – Ridge (Hoerl and Kennard, 1970), and a combination of L1 and L2 penalties – Elastic Net (Zou and Hastie, 2005) on the JAK2 dataset (see Figure 3.9). Although both LI and L2 are shrinkage methods, each of them, do it a bit differently. LI penalty tends to shrink a subset of coefficients exactly to zero while modestly shrinking others. On the other hand, the L2 penalty shrinks all coefficients but tends to remain non-zero values. The results demonstrate that regularizations led to LG model generalization, i.e., they are free from the handicaps of *overfitting*. We developed LG models independently implementing all the three regularization techniques, namely, LASSO, Ridge and Elastic Net. We followed it up with an external 10-fold cross-validation (Figure 3.9 top row), extracted the parameter λ, which returned minimum misclassification error. We hypertuned the models and established coefficient shrinkage at varying values of the parameter λ (see Figure 3.9 middle row). Finally hypertuned models were used for predictions on the *unseen* testing data and ROC curves plotted (see Figure 3.9 bottom row). In each case, the regularized models met the critical benchmarks on the testing data i.e., free from *overfitting* and with remarkable performance

gains, as measured by AUC. The Elastic Net is slightly superior to other regularizers in reducing the model complexity on this dataset.

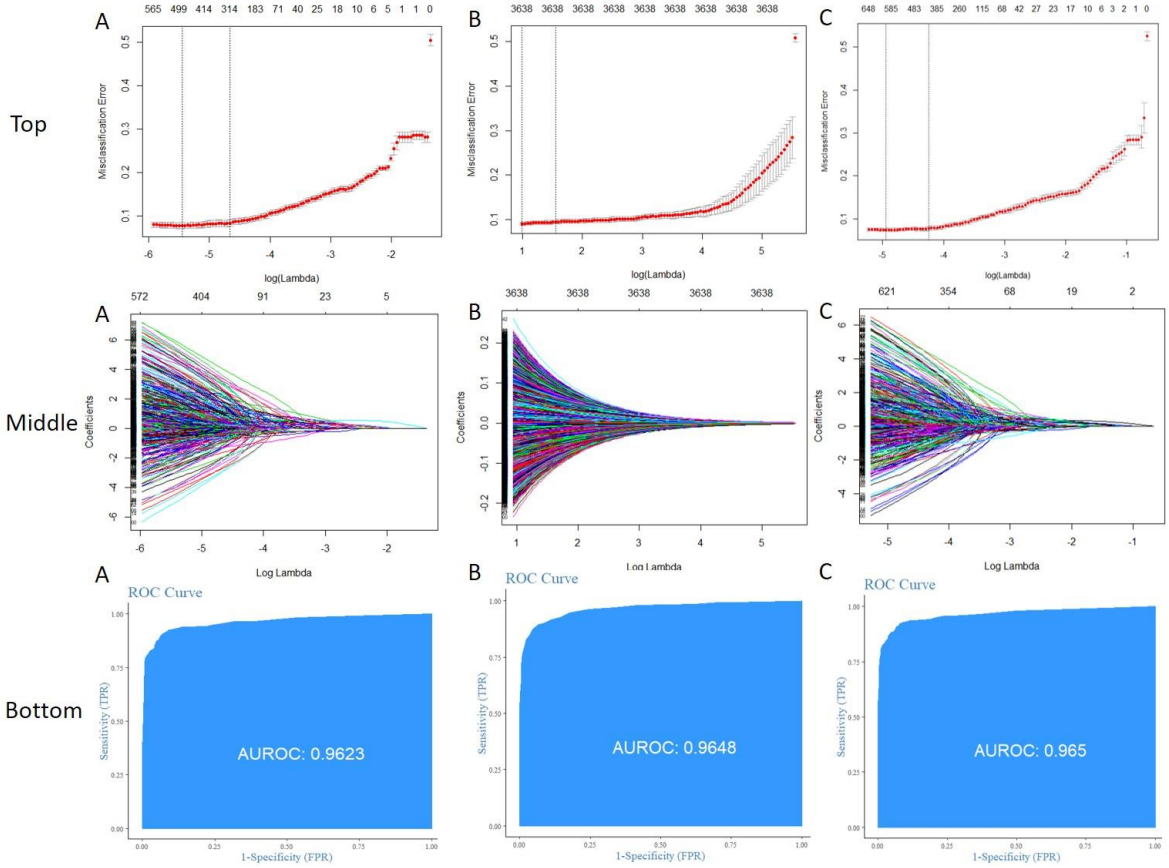


Figure 3.9 Logistic Regression Model Regularization

Cross-validation with different regularizers: top row (A) LASSO (B)Ridge (C) Elastic Net. Each point is the average of 10 fold validation and it tested over a range of λ for misclassification error; Coefficient shrinkage paths with different regularizers, each line represent a continuous trace of the coefficient values at different values of the penalty parameter - λ : middle row (A) LASSO (B) Ridge (C) Elastic Net; ROC curves on testing data with different regularizers: bottom row (A) LASSO (B) Ridge (C) Elastic Net.

3.4 Support Vector Machines - Background

Support Vector Machine (SVMs) is a classification algorithm based on the maximum margin linear discriminants. SVM was first described by Vapnik in the early 70s (Vapnik, 1979). Some excellent books followed Vapnik's publication (Vapnik, 1998, 1995). Other excellent sources include tutorials and books on pattern recognition by SVMs (Bishop, 2006; Burges, 1998; Kowalczyk, 2017; Theodoridis and Koutroumbas, 2009). The goal is to find the optimal hyperplane that maximized the gap or margin between the classes. The kernel is a method to find the optimal nonlinear decision boundary between classes, which correspond to a hyperplane in some high-dimensional "non-linear" space.

Zaki and Meira provided some conceptual framework to develop the background to Support Vector Machine (SVM) (Zaki and Meira, Jr, 2014). For instance, for a dataset $\mathbf{D}$ that is being used to build an SVM classification model with n points (records) in a d-dimensional space (feature variables), could be written as $\mathbf{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$ and such a dataset, which explicitly carry two class labels, namely positive and negative class, which could be represented i.e., $y_i \in \{+1, -1\}$. A set of all points $\boldsymbol{x} \in \mathbb{R}^d$, that satisfy the equation $h(\mathbf{x}) = 0$ is formally referred to as hyperplane $\mathcal{H}$, can be written as:

$$h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b \tag{3.4.1}$$

$$= w_1x_1 + w_2x_2 + \cdots + w_{n1}x_n + b$$

Where $\mathbf{w}$ is the d dimensional *weight vector* and b is a scalar, often called the *bias* term. Expanding upon which, we can simply write for all the points on the hyperplane as follows:

$$h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = 0 \tag{3.4.2}$$

One could rearrange as $\boldsymbol{w}^T\boldsymbol{x} = -b$ which is equivalent to defining all the points are on the hyperplane. It turns out that b determines the offset and gives the distance from the origin.

$$w_1x_1 = -b \text{ or } x_1 = \frac{-b}{w_1}$$

3.4.1 Separating Hyperplane

A canonical hyperplane by definition splits the original d-dimensional space into two half-spaces. We call a dataset *linearly separable* if each half-space has points only from one exclusive class, where we can find the separating hyperplane $\boldsymbol{h(x) = 0}$ such that for all points labeled $\boldsymbol{y_i = +1}$, we have $\boldsymbol{h(x_i) > 0}$.

A dataset is set to be *linearly separable* if each half-space has points only from a single class. If the input dataset is linearly separable, then we can find a separating hyperplane $\boldsymbol{h(x) = 0}$ such that for all points labeled $y_i = +1$, we have $\boldsymbol{h(x_i) > 0}$. The hyperplane function $\boldsymbol{h(x)}$ serves as a linear classifier or a linear discriminant, which predicts the class $\boldsymbol{y}$ for any given point $\boldsymbol{x}$, thus the decision rule may be written in the following form.

$$h(\boldsymbol{x_i}) = \begin{cases} +1 \text{ if } w.\boldsymbol{x_i} + b > 0 \\ -1 \text{ if } w.\boldsymbol{x_i} + b < 0 \end{cases} \tag{3.4.3}$$

3.4.2 The distance of a Point to the Hyperplane

Let's consider some point $\boldsymbol{x}$, that is not present on the hyperplane and $\boldsymbol{d}$ a vector of minimum length originating from the hyperplane $\mathcal{H}$. Let $\boldsymbol{x_p}$ be the orthogonal projection of $\boldsymbol{x}$ on the $\mathcal{H}$. One could write

$$\boldsymbol{x_p} = \boldsymbol{x} - \boldsymbol{d} \text{ and } \boldsymbol{d} \text{ is parallel to the normal vector } \boldsymbol{w} \tag{3.4.4}$$

The implication of this expression is $\boldsymbol{x_p} \in \mathcal{H}$ is $\boldsymbol{w^T x_p} + b = 0$

Therefore, the length of $\boldsymbol{d}$ is written as follows:

$$||\boldsymbol{d}|| = \frac{|\boldsymbol{w^T x}|}{||\boldsymbol{w}||} \tag{3.4.5}$$

3.4.3 Maximum Margin Classifier

One can recast the entire formulation of the maxim margin separating hyperplane as a constrained optimization problem, where we choose values of the weight vector $\boldsymbol{w}$ and the bias b, that yields the maximum margin among all possible separating hyperplanes. One can represent margin for hyperplane as $\boldsymbol{m}$ for the hyperplane h(x) = 0, such that,

$$\text{m} = \arg\max_{w,b}\{\frac{1}{||w||}\}$$

Recollect our goal is to maximize the margin $\frac{1}{||\boldsymbol{w}||}$, subject to the n constraints given $\boldsymbol{w^T x_i} + b \geq 1, \text{for all points } \mathbf{x_i} \in \boldsymbol{D}$. Equivalently, we could minimize $||w||$ and accomplish the same goal.
In which case, the new minimization formulation can is rewritten as:

$\min_{w,b}\{\frac{||w||^2}{2}\}$ with the incorporated linear constraints i.e., $y_i\ (\boldsymbol{w^T x_i} + b > 1, \forall \mathbf{x_i} \in \boldsymbol{D}$

At this stage one can invoke any standard optimization algorithms, to solve such a class of convex minimization problems that are bound by linear constraints. The *Lagrange multipliers* come in handy in providing the solution, this is accomplished by introducing a Lagrange multiplier $\boldsymbol{\alpha_1}$ for each constrain, which satisfies the Karush-Kuhn-Tucker (KKT) conditions at the optimal solution.

$\boldsymbol{\alpha_i}(y_i(\boldsymbol{w^T x_i} + b)$ -1) = 0 and $\boldsymbol{\alpha_i} \geq 0$

This can be reduced to this expression by incorporating all the n constraints. Essentially, the objective function becomes:

$$\min L = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} (\boldsymbol{\alpha_i}(\boldsymbol{w^T x_i} + b) - 1) \tag{3.4.6}$$

Finally, one could summarize it, in this formulation that seeks to minimize L w.r.t to $\boldsymbol{w}$ and b, and it should be maximized with respect to $\boldsymbol{\alpha_i}$. This obtained by taking the derivatives of L w.r.t to $\mathbf{w}$ and b, and setting them to zero yields the solution to the optimal weight vector $\boldsymbol{w}$.

3.4.4 Kernel SVM: Non-linear case

The linear SVM approach can be used for datasets with a nonlinear decision boundary kernel. Conceptually, the idea is to map the original d-dimensional points $\mathbf{x_i}$ in the input space to points $\boldsymbol{\emptyset(\mathbf{x_i})}$ in a high dimensional feature space via some nonlinear transformation $\emptyset$. Given, the extra flexibility, it is more likely the points $\boldsymbol{\emptyset(\mathbf{x_i})}$ might be linearly separable in the feature space. However, a linear decision surface in feature space corresponds to a nonlinear decision surface in the input space. Further, the kernel trick allows us to carry out all operations via kernel function computed in input space, rather than having to map the points into feature space

3.5 Support Vector Machines - Implementation & Experiments

Support Vector Machines (SVM) was implemented in the statistical programming language R. The principal objective in SVM is to find the optimal hyperplane that maximized the gap or margin between the classes. The kernel is a method to find the optimal nonlinear decision boundary between classes, which correspond to a hyperplane in some high-dimensional "non-linear" space. SVM was implemented using the R library LIBSVM (Chang and Lin, 2011). The library e1071 (Meyer, 2017; Meyer et al., 2019) provides an excellent interface for file manipulations (csv to libsvm format). The other libraries include *SparseM* (Koenker and Ng, 2003) for sparsity representation, *ggplot2* (Wickham, 2009) for plotting besides using native R plotting utilities.

3.5.1 Matrix Sparsity

The sparse matrix is especially useful in the evaluation of SVM kernels, The inner dot product of sparse vectors on model training times is significantly lowered by several-fold compared to the dense matrix. As part of exploratory data mining, the sparse and dense matrices were evaluated for performance metrics (see Figure 3.3). We developed both linear and non-linear SVM models with fingerprint data. Initially, we built the entire end-to-end model building routine with one dataset namely MMP2, and subsequently extended it to other datasets, namely, STAT3, ESR1, and JAK2.

3.5.2 Grid-search For SVM Model Hypertuning

First, we developed SVM models (untuned) with a smaller dataset MMP2 to evaluate model performance and accomplish the exhaustive search task in minimal computational times. From the results, it was clear that tuned models deliver higher performance compared to untuned models. The AUC for the tuned models is 5.85% larger than the untuned model, in the context of the absolute number of active compounds, the percentage is significant (see Figure 3.10). Grid-search is a technique for model hypertuning for optimal parameters to extract better model performance. We tested a range of parameters for C and gamma (γ) by grid-search. Intuitively, the C parameter controls the proportion of misclassification. It is a trade-off between simple versus complex models. Larger values of C lead to higher model variance and lower bias. In other words, increases model complexity. On the other hand, a smaller value of C leads to a simpler model with low variance and high bias, which makes the cost of misclassification low (soft margin). The kernel parameter gamma (γ) also influences the model, a small gamma (γ) implies the class of this support vector has a wider influence in deciding the class of a sample training vector (hard margin). Technically, large gamma leads to higher bias and low variance models, and vice versa. For each model, we performed a grid-search by choosing a range of values of C and gamma (γ) simultaneously, to find the optimal hyperparameters. The optimal parameters of this dataset are C = 3 and gamma (γ) = 0.0066 (see Figure 3.10) Similar optimization of SVM kernel function were applied in QSAR studies in C-Aryls glucoside inhibitors in SGLT2 inhibitors (Prasoona et al., 2013) and cholesterol ester transfer protein inhibitors (Riahi et al., 2009).

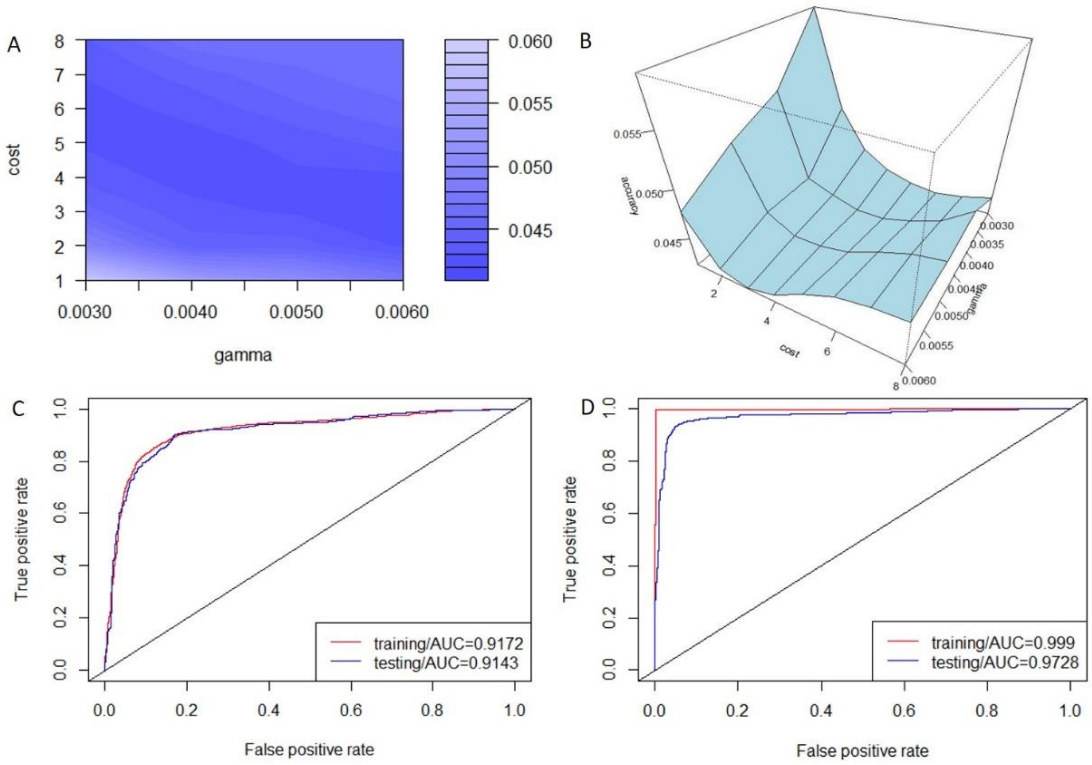


Figure 3.10 Grid Search for SVM Model Hypertuning and Evaluation Metrics
(A) A range of values of the parameters gamma and C are tested, dark blue regions denote optimal values for MMP2 dataset (B) A 3D plot of the grid search (C) ROC curve of the SVM model (untuned) (D) ROC curve of the SVM model (tuned).

3.5.3 SVM Linear vs. Non-linear Kernels – Validation and Evaluation

We systematically implemented the kernel functions on a larger dataset JAK2 with fingerprint features, which include linear kernel, polynomial kernel, Gaussian kernel (RBF), and Sigmoidal kernel. As a routine exercise, we conducted an exhaustive grid search to find optimal parameters to test these kernel functions on each of these datasets (See Figure 3.11 and Table 3.2).

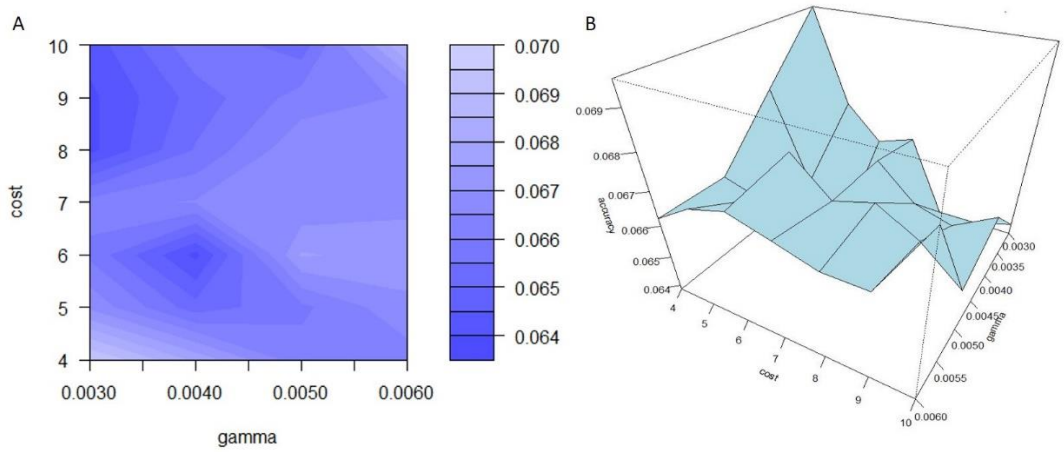


Figure 3.11 Grid Search for SVM Model Hypertuning
(A) A range of values of the parameters gamma and C are tested, dark blue regions denote optimal values for JAK2 dataset (B) A 3D plot of the grid search.

Conceptually, the kernel turns a non-linear classification problem in the input space into an equivalent linear classification problem. This is accomplished by projecting the input feature space into a linearly separable *higher dimensional space* within the Lagrange multiplier dual constrain formulation while allowing the benefits of maximum-margin. There is an implied suggestion that there is a functional transformation of the original space into the higher dimensional space. In fact, the *kernel trick* provides a solution through a pairwise similarity comparison between the original training data instances, instead of applying the transformation and representing the data in the projected space.

$\kappa(x, y)$ replaces $\emptyset(x)^T(y)$

$$w = \sum_{i=n}^{n} \alpha_i y_i \emptyset(x_i)$$

However, w can be infinite dimensional too. Hence, the predictions are performed through the *kernel trick*, as

$$f(x) = w^T \emptyset(x_i) + b$$

$$= \sum_{i=n}^{n} \alpha_i y_i \emptyset(x_i)^T \emptyset(x) + b$$

$$= \sum_{i=n}^{n} \alpha_i y_i \, \kappa(x_i, x) + b$$

Assuming the complexity of $\kappa(x, y)$ is $O(n)$, prediction of one example may take $O(n^d)$ steps, where n is the number of training examples and d is the number of dimensions in the projected space. However, in reality, the intrinsic mathematical structure because of the *kernel trick* lowers the computational burden, where the actual prediction cost is much lower than $O(n^d)$. If the training example x_i is not a support vector, then its Lagrange multiplier α_i is 0. Thus only *support vectors* are stored in the SVM model (Wu, 2020).

The linear kernel is the simplest kernel function, therefore we first evaluated it and followed it up with other kernels. It is given by the inner product (x, y) and an optional constant C. It may be simply described as $\kappa(x, y) = x^T y$. Training a linear classifier is more efficient because we are not conducting any kernel operations. We followed up with the Radial Basis Function (RBF): $\kappa(x, y) = \exp(-\gamma||x - y||^2)$. RDF kernel gives access to all the analytical functions since it is infinitely differentiable because of the exponential function. The tunable parameters are C and gamma (γ). Polynomial kernel: $\kappa(x, y) = \exp(\gamma x^T y + c)^D)$ of the order d will give access to the functions whose $(d + 1)$ the order derivatives are constant and whose derivatives higher than that are zero, can be tuned with polynomial d in addition to C and gamma (γ). Sigmoidal kernel: $\kappa(x, y) = \tanh(\gamma x^T y + c)$, this similar to the sigmoidal function in logistic regression.

3.5.4 SVM Model Cross-validation and ROC

Different linear and non-linear SVM kernels and the appropriately chosen kernel parameters for each of these kernels were cross-validated (see Figure 3.12). The JAK2 fingerprint training data was partitioned into 10 equal folds, SVM models were developed with pooled 9 folds and tested on the 10th fold, and subsequently iterated over ten times, and the models evaluated for their performance. The results show that the linear kernel returned a mean accuracy of 91.49%, while the implementation of a non-linear RBF kernel boosted the mean accuracy to 93.47%. The non-linear polynomial kernel performed slightly better than the linear kernel at 92.47% mean accuracy. However, the non-linear sigmoidal kernel fared poorly by comparison at 66.11% mean accuracy. Overall, RBF delivered the best performance in cross-validation. It is unclear, what led to the numerical instability of the sigmoidal kernel. Further, each of these kernels was evaluated for their accuracy and other evaluation metrics both on the training and testing datasets (see Table 3.2). By far, Receiver Operator Characteristic ROC curves serve as the most credible measure of machine learning model performance. We tested these models with different kernels both on the training and testing datasets (see Figure 3.13). The pattern of model performance is somewhat similar to cross-validation results, which is expected. It turns out, that the RBF kernel is superior in model performance compared to other kernel

implementations. The sigmoidal kernel displayed failure mode with this dataset. It would be interesting to test the behavior of the sigmoidal kernel on other datasets to gain some insight into the numerical instability.

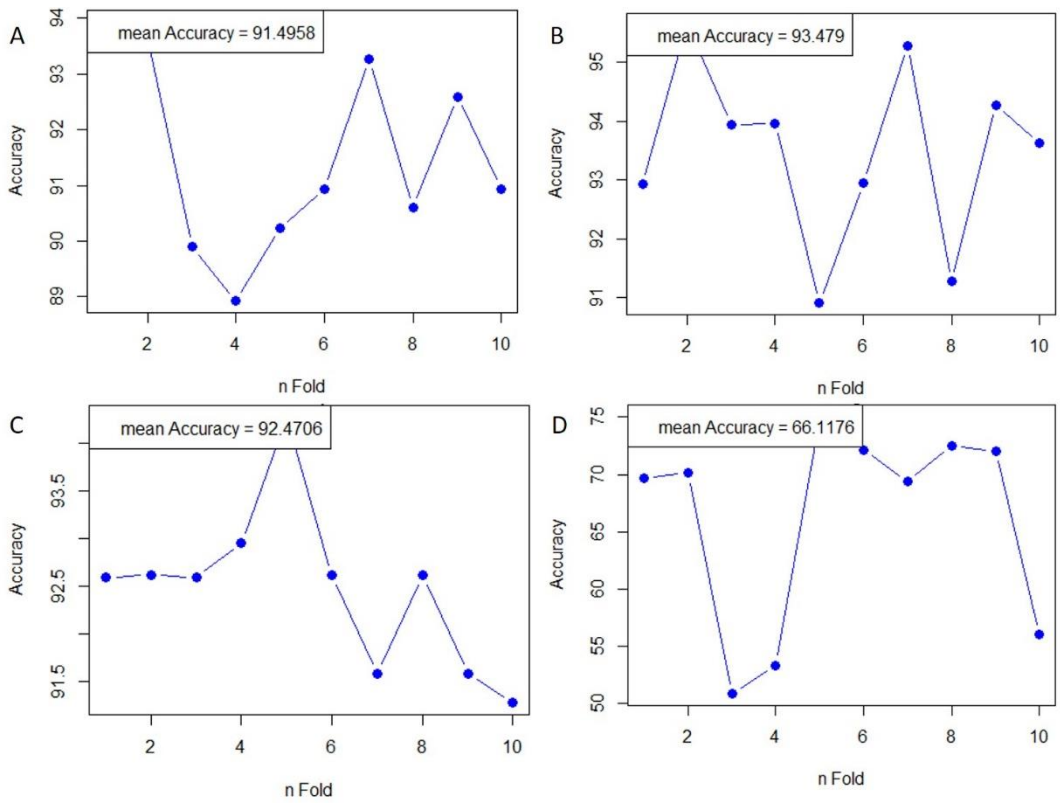


Figure 3.12 SVM Kernels Cross-validation
Different SVM kernels were evaluated for accuracy with the JAK2 dataset. (A) Linear kernel (B) Radial Basis Function (RBF) kernel (C) Polynomial kernel (D) Sigmoidal kernel.

Table 3.2 SVM Kernel Performance Metrics JAK2 Dataset

	JAK2: Linear		JAK2: Radial		JAK2: Polynomial		JAK2: Sigmoid	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.9997	0.9316	0.9871	0.9315	0.9822	0.9275	0.5405	0.5411
Sensitivity	1.0000	0.9239	0.9808	0.9097	0.9658	0.9017	0.4856	0.4834
Specificity	0.9993	0.9393	0.9934	0.9533	0.9987	0.9533	0.5954	0.5988
Pos Pred Value	0.9993	0.9373	0.9931	0.9503	0.9986	0.9499	0.5363	0.5417
Neg Pred Value	1.0000	0.9264	0.9817	0.9149	0.9680	0.9081	0.5457	0.5415
Precision	0.9993	0.9373	0.9931	0.9503	0.9986	0.9499	0.5363	0.5417
Recall	1.0000	0.9239	0.9808	0.9097	0.9658	0.9017	0.4856	0.4834
F1	0.9997	0.9306	0.9869	0.9296	0.9819	0.9252	0.5097	0.5109
Prevalence	0.4908	0.4953	0.4908	0.4953	0.4908	0.4953	0.4908	0.4953
Detection Rate	0.4908	0.4576	0.4813	0.4505	0.4739	0.4466	0.2383	0.2394
Detection Prevalence	0.4911	0.4882	0.4847	0.4741	0.4746	0.4702	0.4444	0.4419

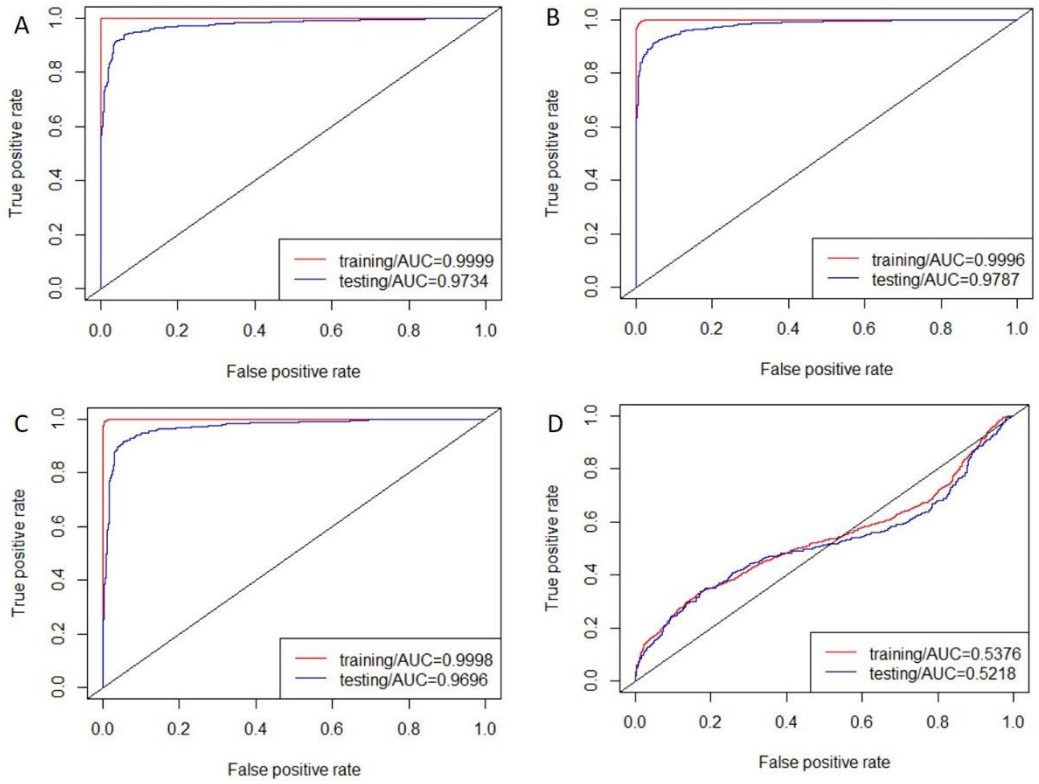


Figure 3.13 Performance Metrics ROC curves

ROC curved for different SVM kernels are plotted (A) Linear kernel (B) Radial Basis Function (RBF) kernel (C) Polynomial kernel (D) Sigmoidal kernel.

3.5.5 Sigmoidal Kernel Failure Mode

The JAK2 dataset demonstrated failure mode with the sigmoidal kernel implementation. We presumed that this was a unique situation with this particular dataset. So we investigated the kernel behavior with other datasets, namely, MMP2, STAT3, ESR1, and BRCA1. In every single case, the sigmoidal kernel failed (see Table 3.3). It is unclear what is the cause of this numerical instability. In general, stability is associated with how the problem is defined, some problems may be inherently unstable, and how an algorithm is used to solve the problem. Error propogation may be due to the unstable nature of the algorithm. In stable algorithms, the error remains bounded and does not amplify. In some instances, the structural representation of data and mathematical operations, which act on such structure can also be the source of numerical instability. In SVM, the Lagrangian dual problem formulation the matrix is symmetric and positive semi-definite (PSD) with kernel application. Vapnik pointed out that the sigmoidal kernel may not be positive semi-definite (PSD) for certain values of the parameters a and r. When K is not PSD the expression of projection in the higher dimensional space is not satisfied and the primal-dual relationship does not exist (Vapnik, 1995). Perhaps this may be a valid rationale for the reconciliation of the sigmoidal kernel failure mode observed.

Table 3.3 SVM Kernel Performance Metrics on other Oncology Target Datasets

	MMP2: Linear		MMP2: Radial		MMP2: Polynomial		MMP2: Sigmoid	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.9960	0.9372	0.9763	0.9472	0.9916	0.9480	0.6799	0.6817
Sensitivity	0.9969	0.9532	0.9754	0.9532	0.9880	0.9448	0.8355	0.8391
Specificity	0.9952	0.9212	0.9772	0.9413	0.9952	0.9513	0.5243	0.5244
	STAT3: Linear		STAT3: Radial		STAT3: Polynomial		STAT3: Sigmoid	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.9797	0.6916	0.8728	0.7588	0.9952	0.7047	0.5000	0.5000
Sensitivity	0.9743	0.6480	0.8199	0.6720	0.9963	0.6080	0.0000	0.0000
Specificity	0.9852	0.7353	0.9258	0.8456	0.9941	0.8015	1.0000	1.0000
	ESR1: Linear		ESR1: Radial		ESR1: Polynomial		ESR1: Sigmoid	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.9879	0.7876	0.9136	0.8572	0.9282	0.8312	0.4999	0.5152
Sensitivity	0.9860	0.7804	0.8674	0.7982	0.8773	0.7656	0.6016	0.6098
Specificity	0.9898	0.7948	0.9599	0.9161	0.9790	0.8968	0.3982	0.4206
	BRCA1: Linear		BRCA1: Radial		BRCA1: Polynomial		BRCA1: Sigmoid	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.9050	0.5565	0.7730	0.6104	0.7544	0.5973	0.5132	0.5412
Sensitivity	0.9641	0.7288	0.8084	0.6656	0.8565	0.7265	0.8306	0.8549
Specificity	0.8459	0.3841	0.7376	0.5552	0.6523	0.4682	0.1957	0.2274

It was proposed by Lin (Lin and Lin, 2003) that the sigmoid kernel matrix is conditionally positive definite (CPD) and thus are valid kernels and further suggest that the training of non-PSD kernels by SMO-type methods. We also show that Radial Basis Function (RBF) slightly outperforms sigmoidal kernels in all the datasets tested.

3.6 Naïve Bayes - Background

Naïve Bayes classifiers are a family of probabilistic machine learning algorithms that are based on Baye's probability theorem. The *naïve* assumption that is made in model building is, that the features that go in the model are independent of each other. In practice, the independence assumption is rarely upheld. In other words, this assumption is often violated but the naïve Bayes classifier seems to perform extremely well in most cases (Bruce and Bruce, 2017; Hastie et al., 2009; Raschka, 2014b).

Zaki and Meira's provided some conceptual framework to develop the background to Support Vector Machine (Naïve Bayes) (Zaki and Meira, Jr, 2014). At the core of the Naïve Bayes classifier, is the estimation of the posterior probability. The simplest interpretation one can offer in terms of a classification problem is as follow: Given a certain feature set for an object, you ask to which class does one such object belongs. More formally, given a dataset $\mathbf{D}$ consisting of n points x_i is a d-dimensional space, where the $y_i \in \{c_1, c_2, \ldots, c_k\}$. The task of predicting the membership of a new instance x is simply estimating the posterior probability P(c_i|x) based on the Bayes theorem, which can be written as follows:

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i) \,.\, P(c_i)}{P(\mathbf{x})} \tag{3.6.1}$$

The *likelihood* term in the numerator is expressed as $P(\mathbf{x}|c_i)$, which tells that the probability of observing $\mathbf{x}$ i.e. $x_1, x_2, x_3, \ldots, x_n$ under the assumption that it truly belongs to the class c_i. The other term $P(c_i)$ in the numerator is the *prior probability* or simply *priors* which tells the proportion instances in that class. Finally, the denominator $P(\mathbf{x})$ is the probability $\mathbf{x}$ for all the K classes of c_i in dataset $\mathbf{D}$.

Now the denominator term remains constant for a given input data, we can conveniently eliminate it. This is now expressed as a summation of $x_1, x_2, x_3, \ldots, x_n$ times the *prior probability*.

$$P(c_i|\mathbf{x}) = \sum_{j=1}^{n} P(x|c_i) \,.\, P(c_i)$$

The assumption of independence of attributes in naïve Bayes permits us to recast the likelihood as a product of dimension-wise probabilities.

$$P(\mathbf{x}|c_{i,}) = \mathrm{P}(x_1, x_2, \ldots, x_n|c_i) = \prod_{j=1}^{d} P(x_j|c_i) \tag{3.6.2}$$

$$\hat{y} = argmax \prod_{j=1}^{d} P(x_j|c_i).\ \mathrm{P}(c_{i,}) \tag{3.6.3}$$

In the case of categorical attributes, the class prediction essentially depends on the *likelihood* and its *prior probability*

In the case of continuous attributes, the *likelihood term* is expressed as follows for a single dimension like any standard Gaussian distribution.

$$P(x_j|c_i) \propto f(x_j\,|\mu_{ij}, \sigma_{ij}^2) = \frac{1}{\sqrt{2\pi\sigma_{ij}}} \exp\{-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\}$$

For a multi-dimensional feature set, it becomes a joint probability.

$$(x_j|c_i) = \prod_{j=1}^{d}(\frac{1}{\sqrt{2\pi\sigma_{ij}}} \exp\{-\frac{(x_j - \mu_{ij})^2}{2\sigma_{ij}^2}\})\hat{y} \tag{3.6.4}$$

3.7 Naïve Bayes – Implementation & Experiments

Naïve Bayes was implemented in the statistical programming language R. It belongs to a family of simple probabilistic classifiers, which makes the naïve assumption that variables are independent in predicting the outcome. The bulk of the implementation is based on the library *naive Bayes* (Majka, 2020) other libraries include *caret* (Kuhn et al., 2016), *amelia* (Honaker et al., 2011), *ggplot2* (Wickham, 2009) for plotting besides using native R plotting utilities.

3.7.1 Range Normalized Naïve Bayes Models – Validation and ROC

The initial Naïve Bayes models development and evaluation were done with AKT1 chemical descriptor dataset. The data set after passing through the routine data preprocessing pipeline yielded a high dimensional data matrix of the size (4535 X 531). Here we tested the suitability of different normalization techniques that included range transform, Z-transform, Box-Cox transform (Box and Cox, 1964), and Yeo-Johnson transform (Yeo and Johnson, 2000). From the results it clear, the model performance metric (see Table 3.4) was comparable for all transforms, except for Z-transform, which was slightly eroded on accuracy. Range transform performed better on the *class* metrics, namely, sensitivity (active compounds) and selectivity (inactive compounds), which is of interest in the present binary classification problem. The models with each of the normalizations were tested both on the independently partitioned training data and test data.

Table 3.4 Effect of Data Normalization on Naïve Bayes Model Metrics

	Normalizations Tuned Models							
	Range Transform		Z-Transform		Box-Cox Transform		Yei-Johnson Transform	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Accuracy	0.8863	0.8750	0.8044	0.7640	0.8743	0.8706	0.8797	0.8743
Sensitivity	0.8105	0.8024	0.9794	0.9572	0.7803	0.7847	0.7983	0.8009
Selectivity	0.9592	0.9472	0.6360	0.5718	0.9648	0.9560	0.9580	0.9472
Pos Pred Value	0.9503	0.9379	0.7214	0.6897	0.9552	0.9466	0.9481	0.9378
Neg Pred Value	0.8403	0.8282	0.9698	0.9308	0.8203	0.8170	0.8315	0.8271

We further validated the models through *k-fold* cross-validation and bootstrap resampling techniques (Kohavi, 1995) (see Figure 3.14). We have recorded the overall model mean accuracy both for *k-fold* cross-validation (85.35%) and bootstrap resampling (82.04%). Besides, we also calculated the mean Cohen kappa metrics for *k-fold* cross-validation (0.7058) and bootstrap resampling (0.6395). Accuracy as a measure of model assessment can lead to erroneous results, especially in unbalanced datasets. Therefore, In general, Cohen kappa

measurement is often suggested to be a desirable metric, because it directly measures probabilities of the *observed* and the *predicted* class with no consideration to an arbitrary threshold for class assignments.

Boostrap resampling though computationally slightly expensive provides an approach for better quantification of uncertainty around a *point estimator* and model assessment. In this case, sampling with replacements provided an additional 100 surrogate models, while not affecting the probability of the samples being drawn. Though, the bootstrap samples may contain duplicates and may well omit a few samples in the original dataset during training. Nonetheless, it offers an opportunity to directly examine the Naïve Bayes model's susceptibility to *overfitting*, as well. The variation in accuracy in different folds and bootstrap resamples suggest that the risk of *overfitting* in this particular case is low. Therefore, Naïve Bayes models are easily generalizable to unseen data for forward validation.

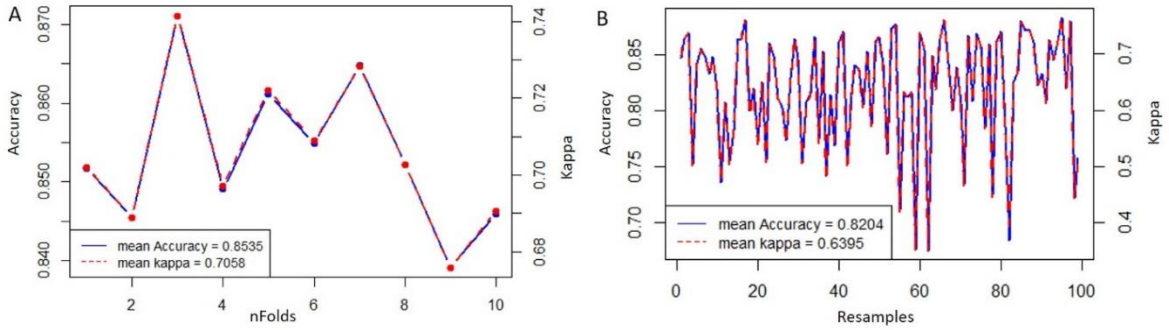


Figure 3.14 Naïve Bayes Models k-Fold Cross-validation
AKT1 dataset is cross-validated the training set to find the optimal hypertuning parameters
(A) 10-fold cross-validation (B) Bootstrap resampling.

We used kernel density estimation to get a more realistic estimate of the class conditional densities of the two classes, since the distribution of the class conditional densities is far from normal. Here we show class conditional densities of four randomly chosen features, among several other features (530 variables) (see Figure 3.15). Although the class priors determine the decision region, there is somewhat significant overlap of the gaussian-like distributions between the two classes. We applied Laplace smoothing to the Naïve Bayes models to avoid the pitfalls of *zero probabilities*, where the maximum likelihood solution could find values of zero or one. Such values can lead to numerical instability, wherein the posterior probabilities of the class membership can be either one or zero. This situation arises when some features are rare or very common across the data (Lawerence, 2015).

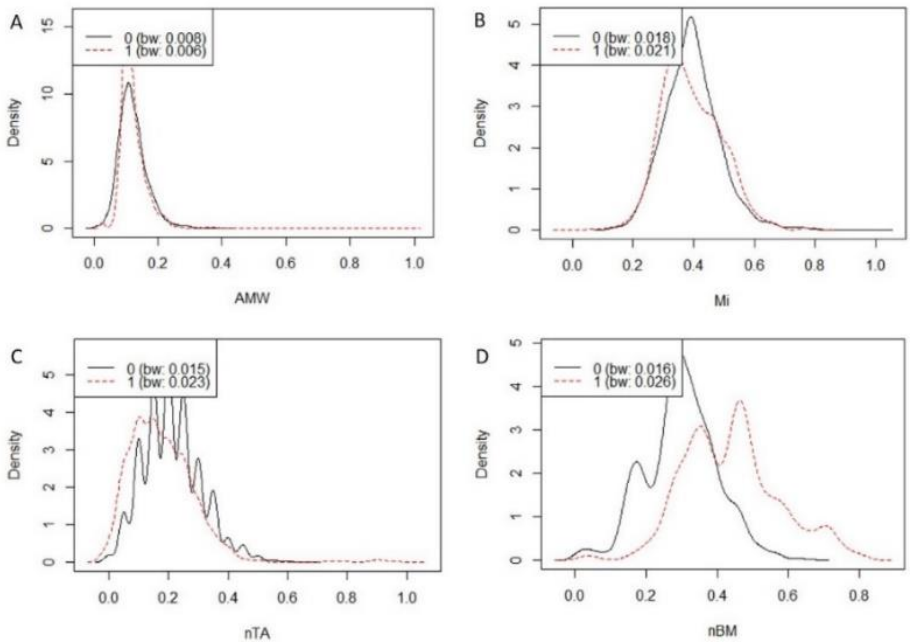


Figure 3.15 Class Conditional Probabilities
The figure shows the probability density distribution from two different classes (red dotted and black dotted lines). A, B, C, and D represent four random features. The decision rule is the product of the individual class probabilities and priors.

Finally, we validated the Naïve Bayes model performance with ROC curves. Plots show the performance with different normalization techniques. As seen earlier, in model accuracy measurements, there is no significant difference in the effects of different normalization techniques on the model AUC measurements, excepts with the Z-transform, which is slightly less (see Figure 3.16).

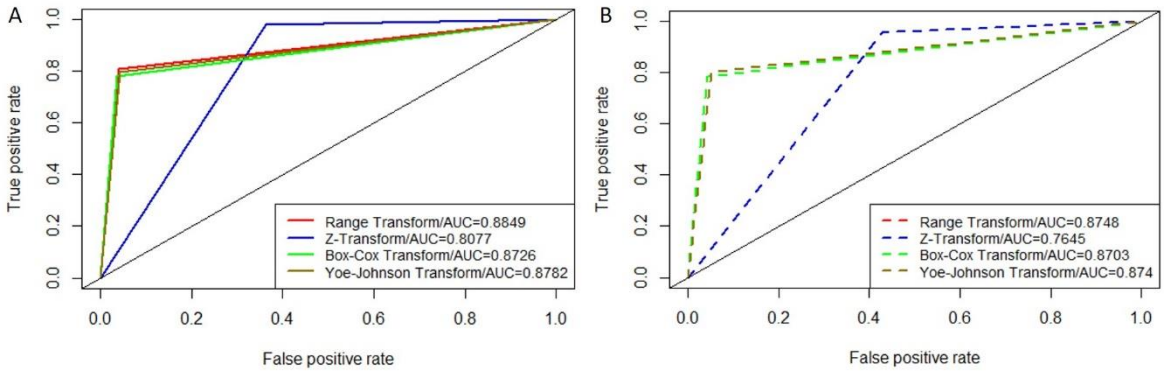


Figure 3.16 Naïve Bayes Models Performance Metrics ROC curves
ROC curves are plotted for different normalization techniques that were adopted for data preprocessing (A) AUC with a training dataset (B) AUC with a testing dataset.

3.7.2 Naïve Bayes Decision Boundary and RadViz Visualization

Visualization is the process of representing data, information, and knowledge in a visual form to support the tasks of exploration, confirmation, presentation, and understanding. Grinstein and his colleagues did much of the pioneering work in the area of interactive data visualization (Ward et al., 2010). Statistics is the study of patterns using mathematics and mathematics is the foundation of algorithms. Machine learning is inherently an iterative process of computations, model building, and model tuning with many parts of the process opaque to visual interrogation. However, there are serious efforts now to incorporate visualizations into modeling work-flow, to turn black-box data analysis into glass-box data analysis. The representation of high-dimensional data features and drawing classifier decision boundaries is one such challenge, where you are limited, in most situations, in representing high-dimensional feature space features on a 2D plane (Hahsler, 2017). Here we show the naïve Bayes decision boundary of two classes plotted using two features with AKT1 data to *visually* access model misclassification errors (see Figure 3.17).

RadViz, short for *Radial Coordinate Visualization* was originally conceived by Hoffman and colleagues, as a computational tool for visualization of high-dimensional data on a plane in 2D (Hoffman et al., 1997). Conceptually, imagine a circle and a collection of points inside the circle. Also imagine springs, one end of which is connected to the point inside the circle, and the other end is anchored on to the circumference of the circle. The points inside the circle represent *records* $\boldsymbol{x}$, the anchors on the circumference represent *features* $\boldsymbol{m}$, and the springs represent the *data vectors*. Finally, assume the stiffness constant (in term of Hooke's law) of the jth string is x_{ij} for one of the data points i. When the point i is released and allowed to reach an equilibrium

position. The coordinates of this position $(u_i, v_j)^T$ are the projection in two dimensional space of point $(x_{i1}, x_{i2}, \ldots, x_{im})^T$ in m-dimensional space. RadViz thus combines the class discrimination algorithm with high-dimensional visualization (Hoffman et al., 1999). RadViz has been extensively applied in the area of biological data mining in building classifiers in identifying quinone subtypes and scaffold effective against melanoma and leukemia cell lines, and in profiling ovarian cancers using chemical descriptor, gene expression, and mass spectral data (Marx et al., 2003; McCarthy et al., 2004; Ujwal et al., 2007). Here we demonstrate the successful application of RadViz with the AKT1 dataset. The two classes are segregated (blue and yellow dots) with some spillage of members of one class into another. The important features are anchored on the circumference of the circle. One can also build *full* and *reduced models* and show the entire range of misclassification errors (see Figure 3.17).

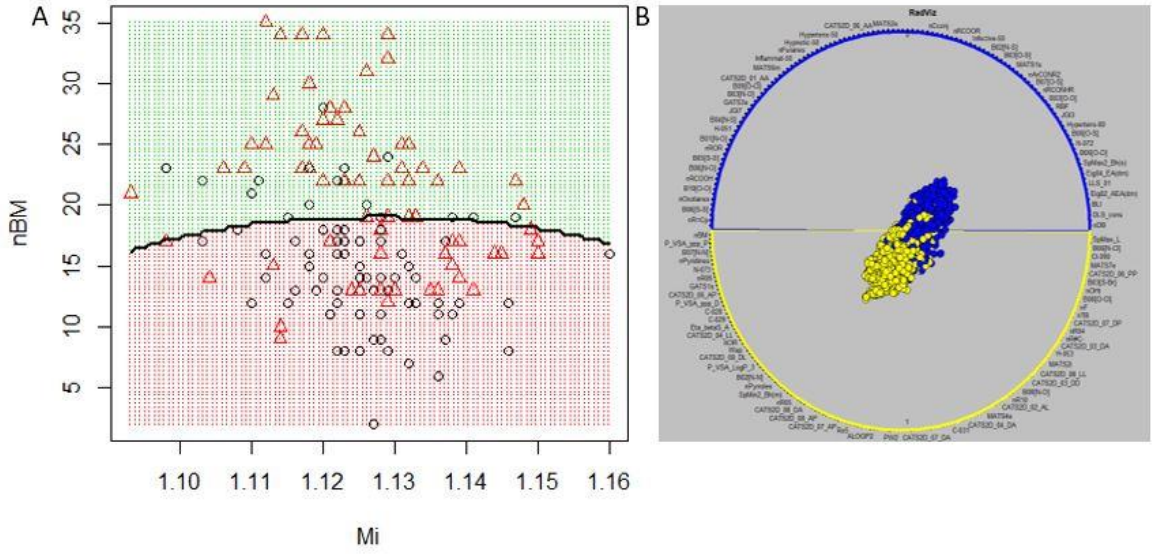


Figure 3.17 2D Visualization of Class Separation
(A) Naïve Bayes decision boundary of instances. The open red triangles and black circles represent two classes (B) RadViz plot with records and features. The solid blue and yellow circles represent two classes and dimensions/features are marked on the circumference of the circle.

3.7.3 Multiple Models Validation

The model development and evaluation were extended to other chemical descriptor datasets: BRCA1, IDH1, MDM2, RARA, ESR1, STAT3, MMP2, and JAK2. Here we summarize these results on different performance metrics (see Table 3.5).

Table 3.5 Naïve Bayes Model Metrics on other Oncology Target Datasets

	Accuracy		Sensitivity		Specificity		Pos. Pred. Value		Neg. Pred. Value	
Targets	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
AKT1	0.8849	0.8748	0.8105	0.8024	0.9592	0.9472	0.9503	0.9379	0.8403	0.8282
BRCA1	0.7051	0.7122	0.7145	0.7234	0.6957	0.7010	0.7013	0.7075	0.7090	0.7171
IDH1	0.7091	0.6802	0.6824	0.6579	0.7359	0.7025	0.7210	0.6886	0.6985	0.6725
MDM2	0.8668	0.8293	0.7337	0.6585	1.0000	1.0000	1.0000	1.0000	0.7897	0.7455
RARA	0.7583	0.7868	0.5199	0.6047	0.9967	0.9690	0.9937	0.9512	0.6749	0.7102
ESR1	0.5900	0.5813	0.9938	0.9782	0.1861	0.1843	0.5536	0.5630	0.9673	0.8874
STAT3	0.6632	0.6885	0.3885	0.4215	0.9379	0.9556	0.8438	0.8947	0.6398	0.6482
MMP2	0.6432	0.6178	0.9973	0.9863	0.2890	0.2493	0.6208	0.6057	0.9893	0.9396
JAK2	0.8130	0.8175	0.6381	0.6430	0.9878	0.9920	0.9815	0.9881	0.7294	0.7286

3.8 Artificial Neural Net (ANN) - Background

The Artificial Neural Net (ANN) is yet another class of learning methods. The central idea in ANN is to extract linear combinations of inputs as derived features and model the target as a nonlinear function of these features. The biological metaphor of neural network as a mathematical representation of information processing may provide some intuition in creating a computational model for neural networks based on neural plasticity in Hebbian learning, by adjustment of weights to input signals to the 'neuron'. The demonstration by Rosenblatt on such a simple network of a model neuron was called 'perceptron' (Rosenblatt, 1958). Simple 'perceptron', while effective in solving linearly separable problems had its limitations, which led to the proposal of a multilayer perceptron (MLP) (Hastie et al., 2009). The error back-propogation method was the next major development to solve, not linearly separable problems (Rumelhart, et al., 1986). The core ideas central to neural networks are the Neural Net Architecture; Forward Propogation and Backpropogation. Several excellent resources include textbooks by leading researchers in the area of neural networks (Bishop, 2006; Hastie et al., 2009; Mitchell, 1997; Zhang et al., 2020).

3.8.1 Neural Net Architecture

A typical neural net architecture consists of a set of interconnected nodes also referred to as neurons beginning with an input layer, followed by one or more hidden layers and an output layer (see Figure 3.18). Neural networks perform affine transformations to concatenate the input features with some random *weights* and converge at a specific node in the network. The concatenated input is evaluated by the activation function and output obtained. The prediction is evaluated against the observed class label and the model error is calculated. Subsequently, the prediction is improved iteratively by updating the network parameters (*weights* and *biases*) by the technique of back propogation.

3.8.2 Forward Propogation

Forward propagation refers to the sequential steps involved in moving the inputs through the hidden layer/s to the output in a neural network architecture. For example, we can represent this for illustrative purposes for a single hidden layer as $\in \mathbb{R}^d$ as described by Zhang and others (Zhang et al., 2020).

$$z^{(2)} = x\, W^{(1)} \tag{3.8.1}$$

where $z^{(2)}$ is the intermediate variable that is produced by the concatenation of the x is the data matrix of dimension $n\, x\, d$ with the initial random weight parameter of the hidden layer, such that, $W^{(1)} \in \mathbb{R}^{nxd}$. The standard notation for weights representation in a neural network: $W_{i,j}^{l}$, where the superscript l is the lth hidden layer, subscript i is the ith weight and subscript j is the jth neuron.

Now that we have the activities of the second layer, we apply the activation function $\emptyset\left(z^{(2)}\right)$ to each entry of the matrix $z^{(2)}$ which yields a matrix exactly as the same dimensions as $z^{(2)}$, and this is referred to as $a^{(2)}$

$$a^{(2)} = f(z^{(2)}) \tag{3.8.2}$$

In the subsequent steps the output $a^{(2)}$ after passage through the activation function is concatenated with the weight vector $W^{(2)}$, one each for each neuron in the previously hidden layer and apply one more time the activation function, This yields a vector, $z^{(3)}$ whose length is the same as the number of samples in the original data x.

$$z^{(3)} = a^{(2)}\, W^{(2)} \tag{3.8.3}$$

Finally, we apply the activation function again to estimate the prediction $\hat{y}$, which is written as follows:

$$\hat{y} = f(z^{(3)}) \tag{3.8.4}$$

We can calculate the loss term for a single data example, as follows making the assumption the *loss function* is l and the class label is y.

$$L = l(y, \hat{y}) \tag{3.8.5}$$

Since we have described already regularization in the earlier section, we can simply write the final *objective function* J as follows, where L is the loss term, and s is the regularization term.

$$J = L + s \tag{3.8.6}$$

3.8.3 Backpropogation

The backpropogation is a procedure that computes the *gradient* of the *loss function* with respect to the weights, in fitting a neural network. It uses gradient methods for training a single layer or a multilayer network, updating weights to minimize the loss function using *gradient descent* or variants of it like *stochastic gradient descent* (Goodfellow et al., 2016). The term backpropogation and its general use in neural networks was a seminal contribution in the field, and it was proposed by Rumelhart and collaborators (Rumelhart, et al., 1986). Somewhat similar ideas, while not as compelling existed dating back to the early 1960s.

Zhang and others (Zhang et al., 2020) described that in the case of a simple network one hidden layer with the parameters, namely, $W^{(1)}$ and $W^{(2)}$, The first step is to calculate the gradients of the *objective function* J, with respect to the loss term L and regularization s which we described in the forward propogation (incidentally the last step in the sequence). Here the order of operations is exactly in the reverse, as opposed to the forward propogation.

$$\frac{\partial J}{\partial L} = 1 \text{ and } \frac{\partial J}{\partial s} = 1 \tag{3.8.6}$$

Next, steps include calculation of a series of gradients (first-order differentiation) w.r.t. the weight parameters $W^{(2)}$, the activation layer $a^{(2)}$, and finally, the weight parameters $W^{(1)}$.

$$\frac{\partial J}{\partial W^{(2)}}, \quad \frac{\partial J}{\partial a^{(2)}}, \quad \frac{\partial J}{\partial W^{(1)}} \tag{3.8.7}$$

In summary, the first pass of the model training involves one sweep in the forward direction and a reverse sweep in the backward direction along the compute graph. The subsequent steps in the model training involve

several iterations, yielding the appropriate parameter values with the ultimate goal of lowering the *cost function*.

3.9 Artificial Neural Net (ANN) – Implementation & Experiments

Artificial Neural Nets (ANN) was implemented in the statistical programming language R. The bulk of the implementation is through *keras* (Falbel et al., 2020), an R interface to *Keras* (Chollet, 2015), a python deep learning package, it is a high-level neural networks API, which is capable of running on top of *TensorFlow*^{TM}, which is Google's open-source software library for numerical computations using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors). The flexible architecture of which facilitates, both running the same code on both CPU and GPU high-performance clusters. The other libraries include *ggplot2* (Wickham, 2009) for plotting besides using native R plotting utilities.

3.9.1 Effect of Number of Hidden Layers on ANN Model Performance

Here we have evaluated the effect of the number of the hidden layer on the model accuracy and loss function. The schematic of the neural net architecture is shown, ranging in two to five layers, with a specific number of neurons in each layer (see Figure 3.18).

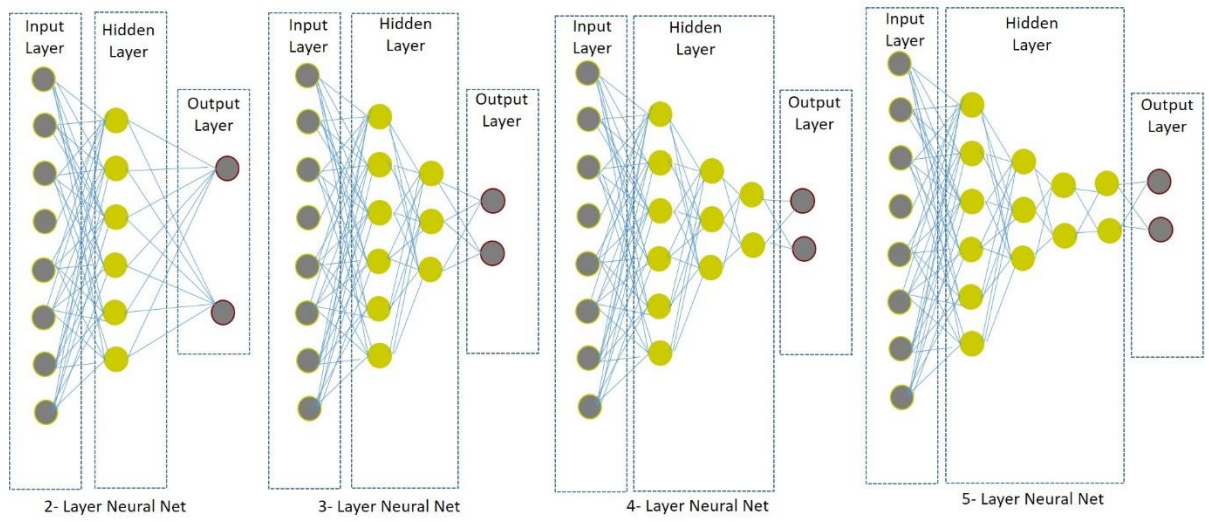


Figure 3.18 Artificial Neural Net (ANN)
A schematic of ANN architecture of the number of hidden layers in neural nets.

We set up sequential ANN models with the AKT1 dataset which has 530 dimensions and iteratively evaluated the different number of layers. Formally, an n layers model represents *n-1* hidden layers. At each epoch, we recorded the model accuracy and loss function, where each epoch is one complete sweep of the data through the hidden layers (forward propogation/backpropogation). The results show that the addition of new layers slightly improves the model accuracy both in the training data and validation data but the addition of layers beyond a certain number of layers degrades accuracy (see Figure 3.19). However, there is a marked oscillation in accuracy measurements in all the models with the varying number of hidden layers in early epochs (between 1 and 20) that begin to stabilize later (after 20 epochs). Permutation or shuffling of the validation data or varying the batch size did not smoothen the curves. Likewise, the loss function also behaves similarly, with additional layers minimizing the loss function in the training data, while in the validation data, it minimizes very early in the epochs and begins to spike, a little later suggesting *overfitting*. A single hidden layer in neural networks, with a finite number of neurons, is capable of universal approximation, implying that neural nets can approximate any continuous function when given appropriate weights (Csáji, 2001). One of the first versions for the sigmoidal function was proposed by (Cybenko, 1989).

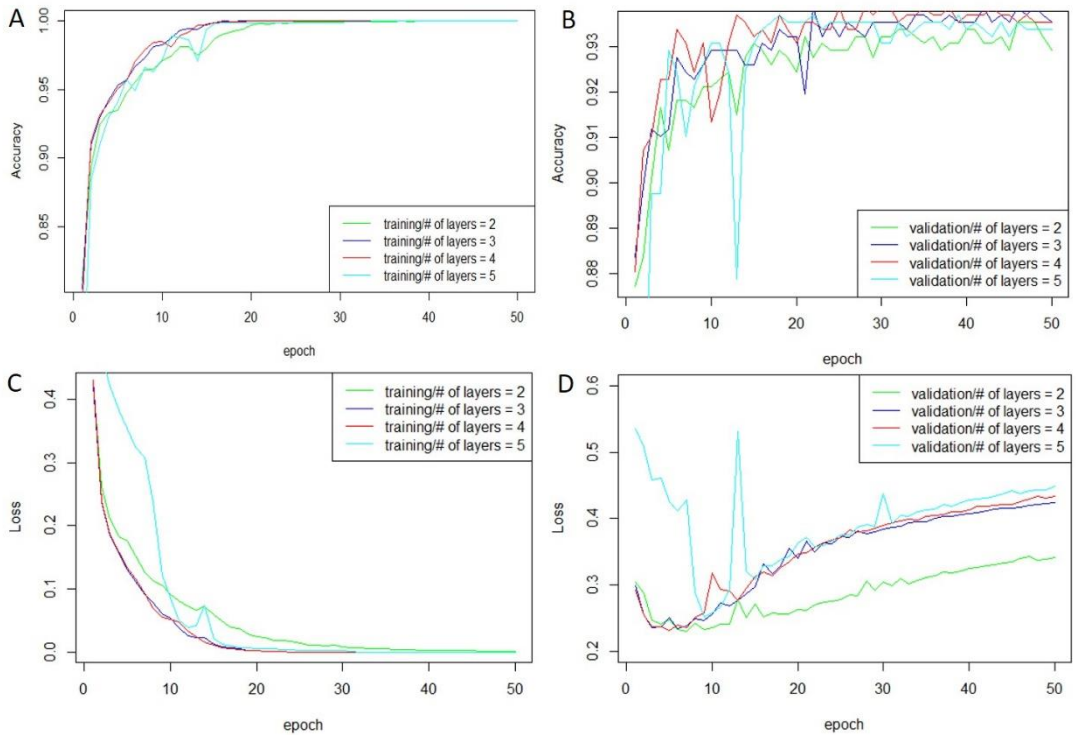


Figure 3.19 Effect of Number of Hidden Layers on ANN Models
Different numbers of layers were evaluated for model performance with the AKT1 dataset. (A) Accuracy on the training data (B) Accuracy on the validation data (C) Loss on the training data (D) Loss on the validation data.

Hornick (Hornik, 1991) expanded upon the idea of universal approximation, not limited to any specific activation function but rather to the multi-layer perceptron architecture, which is foundational to deep neural networks. Hinton's group provided some ground-breaking advances in deep learning for models to learn complex representation through multiple hidden layers (Hinton et al., 2006).

3.9.2 Effect of Different Optimizers on ANN Model Performance

Here we have evaluated the effect of different optimizers on the model accuracy and loss function. Selecting the right optimizer is critical in contemporary deep learning model development. Optimization algorithms are typically defined by their update rule, which is controlled by hyperparameters that determine its behavior (e.g., $\propto$ the learning rate). Consider a differentiable loss function $\ell: \mathbb{R}^d \rightarrow \mathbb{R}$ whose vector of the first partial derivatives is given by $\nabla\ell(\theta)$. In this context, ℓ represents the loss function computed over the entire dataset by a neural net and $\theta \in \mathbb{R}^d$ represent a vector of model parameters. The optimization problem is to find a

point that (at least locally) minimizes ℓ. First-order iterative methods for this problem construct a sequence θ_t of iterates converging to a local minimum θ_* using the queries to ℓ and $\nabla\ell$. The sequence θ_t is constructed by an update rule $\mathcal{M}$, which determines the next iterate θ_{t+1} from the history $\hbar_t$. Thus, given an initial parameter value $\theta_0 \in \mathbb{R}^d$, the sequence of points visited by an optimizer with update rule $\mathcal{M}$ is given by, $\theta_{t+1} = \mathcal{M}(\hbar_t, \theta_t)$ (Choi et al., 2020; Nesterov, 2018). From the results (see Figure 3.20), the ANN model compiled with ADAM optimizer (Kingma and Ba, 2017) consistently outperformed when compared with other optimizers, both on accuracy and loss function metrics with training and validation data (see Figure 3.20). By contrast, the Stochastic Gradient Descent (SGD) optimization (Bottou, 2012), was ranked the lowest in the same experiments, which is rather surprising, considering its wide adoption. While there is no consensus on the choice of an optimizer, empirical comparisons of various optimizers have been cited in the literature using different types of data (Wilson et al., 2017). Therefore, ADAM seems to be *fit-for-purpose* for the present experiments.

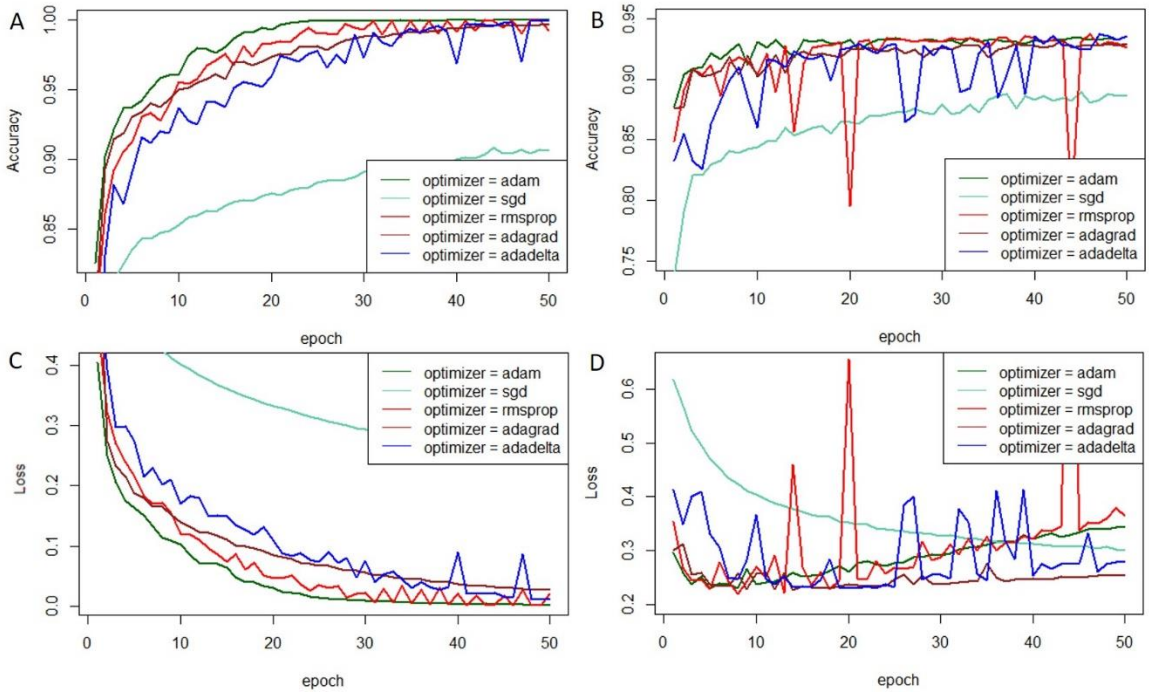


Figure 3.20 Effect of Different Optimizers on ANN Models
Different optimizers were evaluated for model performance with the AKT1 dataset. (A) Accuracy on the training data (B) Accuracy on the validation data (C) Loss on the training data (D) Loss on the validation data.

3.9.3 Effect of Different Learning Rates on ANN Model Performance

Learning rate $(\propto)$ is one of the key hyperparameters that controls the *weights* in training neural nets with respect to the *gradient descent*. It scales the magnitude of the weights update in order to minimize the neural networks *loss function*. For instance, if the learning rate is set to smaller values, it makes very small updates to the weights, though this ensures that we are not missing any *local minima* but can run the risk of inadvertently getting struck in one such situation. And this may take a longer time for convergence. On the other, hand, larger values can risk the possibility of divergent behavior i.e., overshoot the *minima* and fail to converge. Often, the best learning rates are associated with the steepest drop in the differentiable *loss function*. In simple terms, we are seeking the values $\propto$, that help in converging to a local minimum θ_*

$$\theta_{t+1} := \theta_i - \propto \frac{\partial}{\partial \theta_i} \ell(\theta_i).$$

In the following experiments, we tested a range of values for $\propto$ and evaluated its effect on the model performance both on the training and testing data (see Figure 3.21). The choice of smaller values of $(\propto = 0.001, 0.005)$ compared to larger values $(\propto = 0.015, 0.0172)$ seems to show better performance, when scored on the metrics of model accuracy and model loss. As expected, the training accuracy is always higher than the testing data, and training loss is always lower than the testing data. Besides, ANN models with the smallest $(\propto)$ values tested have shown the lowest loss and the loss curve remains reasonably stable across all the epochs. Models with slightly larger learning rates begin to diverge around ten epochs. Other systematic approaches to finding the optimal learning rates include techniques like cyclical learning rates which are bound between two values and exhibit multiple modes of decay from fixed to exponential (Smith, 2017). Stochastic Gradient Descent with warm Restarts (SGDR) is similar to the cyclical approach, where an aggressive annealing schedule is combined with periodic restarts with the original learning rates (Loshchilov and Hutter, 2017).

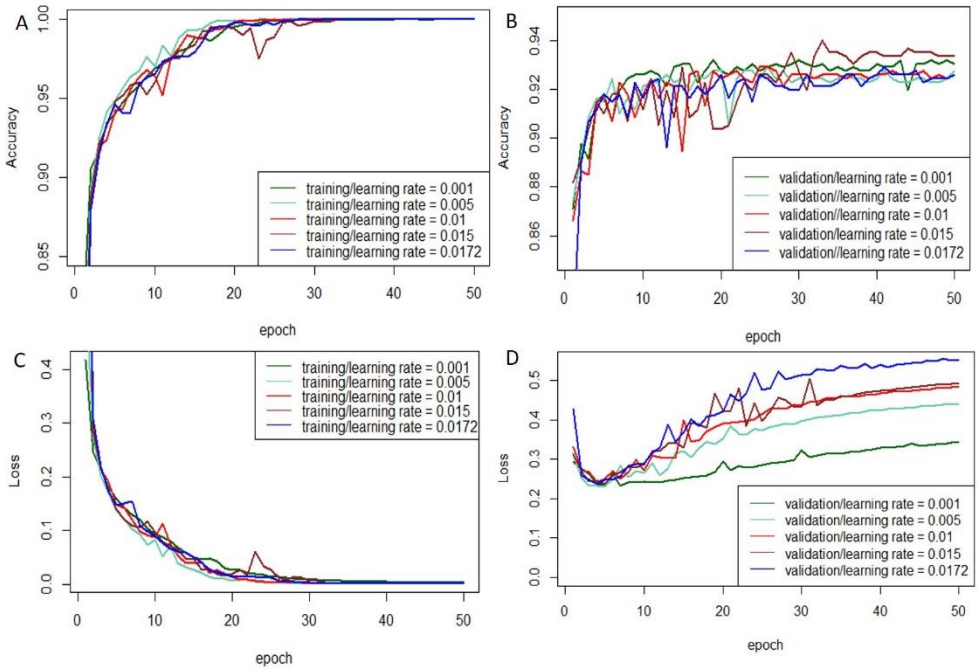


Figure 3.21 Effect of Different Learning Rates on ANN Models
Different learning rates were evaluated for model performance with the AKT1 dataset. (A) Accuracy on the training data (B) Accuracy on the validation data (C) Loss on the training data (D) Loss on the validation data.

3.9.4 Effect of Regularization on ANN Model Performance

Just to recollect, model regularization is motivated by the desire to reduce model complexity and expanding its ability for generalizability. It does this by restricting the *degrees of freedom* in the model (Raschka, 2019). In the earlier sections, we have written in some detail explaining the concept and different techniques for model regularization (see sections Approach: 2.5.2 –2.5.5, Evaluation of Algorithms: 3.2.7). Here we just report the findings of applying some of these techniques on ANN models (see Figure 3.22) – Random Neuron Drop regularization (Srivastava et al., 2014), L1 (Tibshirani, 1996), and L2 regularization (Hoerl and Kennard, 1970). First, we applied Random Neuron Drop regularization. We observe that the model accuracy drops in the training data as a greater percentage of neurons are randomly dropped from the hidden layers. Interestingly, accuracy on the validation data improves slightly, under identical regularization regimens, albeit with fluctuations. Evaluating the loss function under the same conditions, in training data the loss increases, while in the validation data the loss decreases (epochs > 30). In effect, neuron drop regularization leads to model performance improvement in the validation data, while resisting *overfitting* in the training data, which

is the intended outcome. This augurs well for the model generalizability of unseen test data in forward validation.

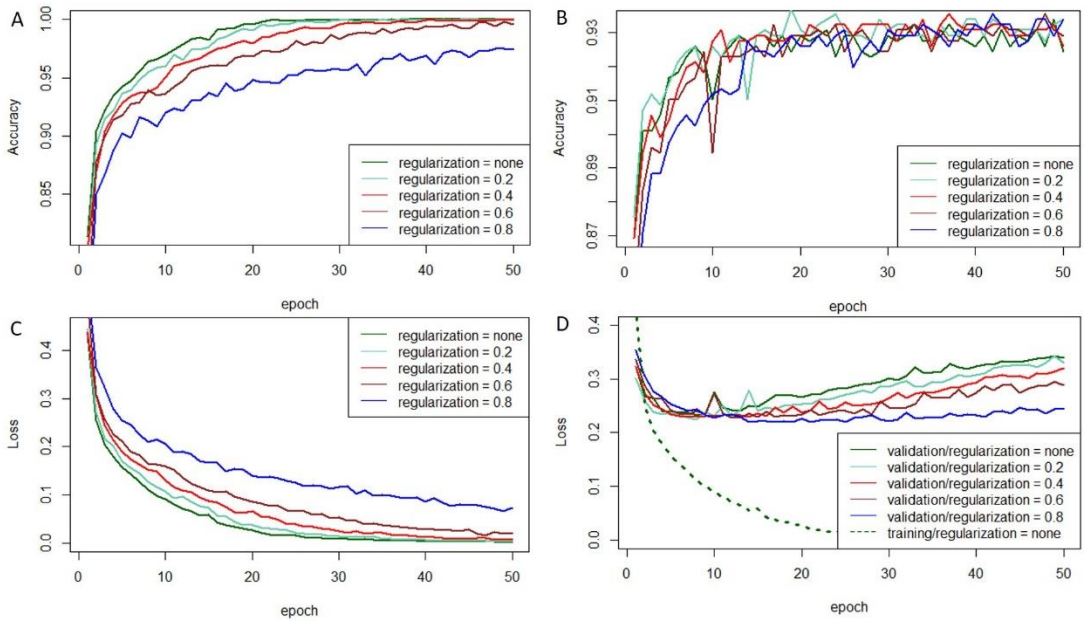


Figure 3.22 ANN Model Random Drop Regularization
A range of regularization values was evaluated for model performance with the AKT1 dataset. (A) Accuracy on the training data (B) Accuracy on the validation data (C) Loss on the training data (D) Loss on the validation data.

Next, we tested L1 and L2 regularization (see Figure 3.23), here we demonstrate some interesting effects of LI and L2 regularizations. Unlike neural drop regularization, we did not see any model performance improvements but a case can be made that L1 and L2 regularizations help resist model *overfitting*. The effects of L1 are more drastic compared to L2 on the accuracy measurement both for the training and validation data (see Figure 3.23 (A) and (C) with double arrow bars). Each accuracy measurement in the training and testing set has corresponding controls (no regularization). An almost similar pattern is observed in the loss measurements, L1 shows greater resistance to *overfitting* compared to L2 both in training and testing data. This also carries appropriate controls.

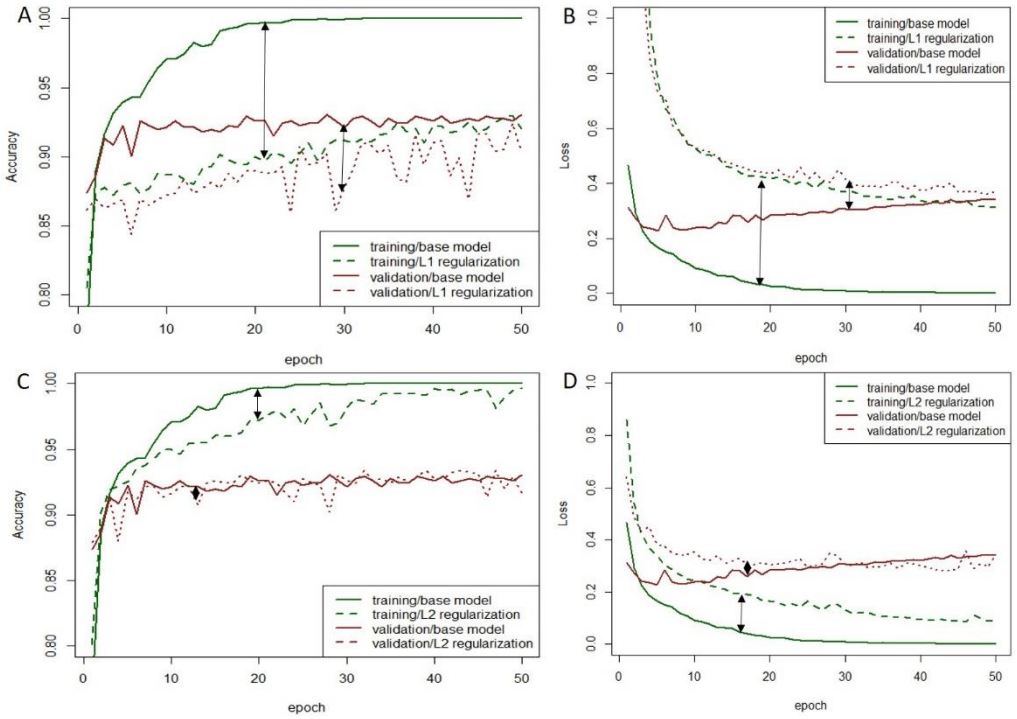


Figure 3.23 ANN Model L1 and L2 Regularization
A range of L1 and L2 regularization values were evaluated for model performance with the AKT1 dataset. (A) Accuracy on the training data ((B) Loss on the training data (C) Accuracy on the validation data (D) Loss on the validation data.

3.9.5 ANN Model Hypertuning

Here we show ANN model performance after hypertuning the model parameters. The AUC curves

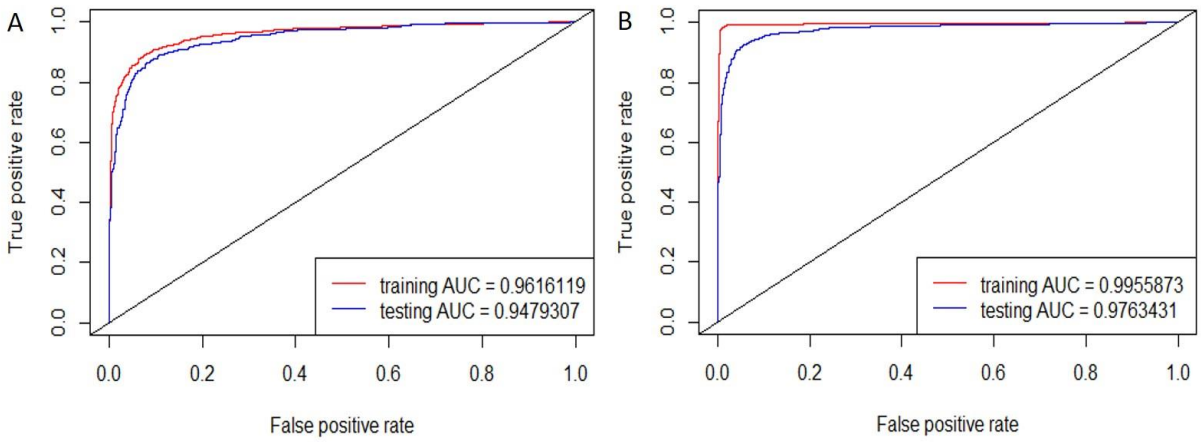


Figure 3.24 ANN Models Performance Metrics ROC curves
ROC curves are plotted for AKT1 dataset (A) AUC with training and testing data (before model hypertuning) (B) AUC with training and testing data (after model hypertuning).

were recorded both for the training and testing datasets before and after hypertuning (see Figure 3.24). The results clearly demonstrate the performance gains. The ANN modeling was further expanded to other datasets and the summary of the hypertuned performance metrics is recorded (see Table 3.6).

Table 3.6 ANN Hypertuned Model Metrics on other Oncology Target Datasets

Targets	Accuracy		Sensitivity		Specificity		Pos Pred Value		Neg Pred Value	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
AKT1	0.9852	0.9302	0.9846	0.9422	0.9858	0.9182	0.9852	0.9191	0.9851	0.9416
BRCA1	0.9532	0.7946	0.9538	0.7961	0.9526	0.7930	0.9527	0.7937	0.9537	0.7955
IDH1	0.9333	0.7205	0.9382	0.7499	0.9283	0.6912	0.9290	0.7083	0.9376	0.7343
MAPK1	0.8866	0.6121	0.8259	0.4248	0.9472	0.7994	0.8799	0.4979	0.9208	0.7480
MDM2	0.9739	0.8811	0.9791	0.8537	0.9687	0.9085	0.9690	0.9032	0.9789	0.8613
RARA	0.9536	0.8023	0.9536	0.7984	0.9536	0.8062	0.9536	0.8047	0.9536	0.8000
ESR1	0.9090	0.7806	0.9933	0.8701	0.8248	0.6911	0.8536	0.7461	0.9917	0.8361
STAT3	0.8670	0.6703	0.9856	0.7851	0.7484	0.5556	0.7718	0.6129	0.9837	0.7426

3.10 Decision Trees - Background

Decision trees build classification models in the form of the canonical tree structure. The main goal of the construction of the decision tree is finding the attributes that return the highest information gain (i.e., the most homogeneous branches). The entropy of each branch is calculated. The result is information gain or a decrease in entropy. Most algorithms that have been developed for learning decision trees are variations of the core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the ID3 algorithm (Quinlan, 1986) and its successor C4.5 (Quinlan, 1993). The later iteration of Quinlan C5.0 is covered by a patent and not available in the public domain. CART or Classification and Regression Trees is often used as a generic acronym for the term Decision Tree, though it has a more specific meaning. In sum, CART implementation is very similar to C4.5 (Breiman et al., 1993). The one notable difference is that CART constructs the tree based on the numerical splitting criterion recursively applied to the data, whereas the C4.5 includes the intermediate step of constructing *rule sets*.

Zaki and Meira (Zaki and Meira, Jr, 2014) elegantly described the conceptual framework, which forms the basis of the background to decision trees. Briefly, given a training dataset $\boldsymbol{D}$. Typically, $\mathcal{R}$ denote the data space of n samples and d-dimensional feature space either as numeric or categorical attributes with a class assignment where $y_i \in \{c_1, c_1, \ldots, c_k\}$. In other words, y_i is one of the several classes. A decision tree recursively partitions the tree model $\boldsymbol{D} = \{x_i, y_i\}_{i=1}^{n}$del that predicts the class $\hat{y}_i$ for each point x_i, via an axis-parallel hyperplane, such that the partition so obtained is relatively homogenous of a given class c_k. The decision tree model is essentially a structural representation of the hierarchical split decisions. The class assignment of a new sample occurs by traversal along the split branches to the terminal leaf node of the decision tree.

3.10.1 Entropy

Entropy is a concept in physics that refers to a measure of uncertainty or disorder in a system. In information theory, the concept communicates the 'informational value' of an event, also called Shannon entropy (Shannon, 1948). By extension, in classification, a relatively homogeneous partition has lower entropy compared to a non-homogeneous population suggesting that most of the points in such a partition carry the same class label. By contrast, a population of mixed labels has the higher entropy. Formally, one can define the entropy of a set of labeled points D as following: (Zaki and Meira, Jr, 2014)

$$\boldsymbol{H(D)} = \sum_{i=1}^{k} p(c_i|\,\boldsymbol{D})\ log_2\,|\,p(c_i|\boldsymbol{D})$$

where $p(c_i|\boldsymbol{D})$ is the probability of class c_i in $\boldsymbol{D}$, and k is the number of classes. If $p(c_i|\boldsymbol{D}) = 0.5$, which means that there are an equal number of points belonging to both the classes, and hence the entropy is the highest value $\boldsymbol{H(D) = log_2 k}$. On the other hand, if all the points are pure i.e., belonging to the same class, then the entropy is zero. Intuitively, one can define the highest information gain as the largest difference in the entropy of the class and the entropy of the weighted attribute.

When $\boldsymbol{n = |D|}$ is the number of points in D, and $\boldsymbol{n_x = |D_x|}$ and $\boldsymbol{n_y = |D_y|}$ are the number of points in $\boldsymbol{D_x}$ and $\boldsymbol{D_y}$, which are attribute values extracted from a contingency table. Therefore, Information Gain can be compactly expressed as:

$$\text{Gain}(\boldsymbol{D, D_{x,y,}}) = \boldsymbol{H(D) - H(D_x, D_y)}.$$

3.10.2 Gini Index

Gini index is another measure to gauge the purity of a split point, which is defined as follows:

$$G(D) = 1 - \sum_{i=1}^{k} p(c_i|\mathbf{D})^2$$

Where $p(c_i|\boldsymbol{D})$ is the probability of class c_i in $\boldsymbol{D}$, and k is the number of classes For example, if the probability of the majority class is 1, it suggests that the partition is homogeneous. In other words, there is less disorder, implying that all members in the partition have the same class label. Lower values of the Gini index indicate order, while higher values indicate a disorder.

3.10.3 Ensemble Methods: Bagging and Boosting

Ensemble learning is a framework for combining multiple *weak* learners' algorithms to produce a *strong* learning algorithm. Ensemble methods have become a major learning paradigm since the 1990s, with promotion by two pieces of pioneering work. One is empirical, in which it was found that predictions made by combinations of a set of classifiers are often more accurate than the prediction made by a best single classifier. Ensemble methods create *combined classifiers* using the output of multiple *base classifiers*, which are trained on different data subsets (Hansen and Salamon, 1990). The other is theoretical, in which it was proved that weak learners can be boosted to strong learners (Schapire et al., 1998). Overall, ensemble classifiers

help reduce variance and bias, leading to better overall performance. Zhou has written an excellent textbook bringing together multiple topics in ensemble learning (Zhou, 2012).

3.10.4 Bagging

The concept of bagging was proposed by Breiman, who made some significant contributions in the area of ensemble learners (Breiman et al., 1993; Breiman, 1996a). Bagging, is short for *Boostrap Aggregation*, it is an ensemble classification method that builds models by a sampling method called bootstrapping (sampling with replacement), where random subsets of data $\boldsymbol{D_i}$, $i = 1, 2, \ldots, k$. are used as input for training. Several base classifiers $\boldsymbol{M_i}$ are built while being trained on $\boldsymbol{D_i}$. The assignment of the new test sample x, to one of the class $\boldsymbol{c_i}$ using each of the K base classifiers, $\boldsymbol{M_i}$. This is accomplished by a system of majority voting. Bagging is believed to help in reducing variance, especially if the base classifiers are unstable, due to the averaging effect of majority voting, while its effect on the bias is almost minimal (Breiman, 1996b).

3.10.5 Random Forests

The Random forest algorithm is a special case of the bagging algorithm, which draws random bootstrap samples from the training dataset. However, in addition to the bootstrap samples, it draws random subsets of features for training the individual trees, then it builds a large collection of de-correlated treas and averages over this collection of trees (Breiman, 2001, 1996a). It is suggested that its better predictive performance comes because of its better bias-variance tradeoffs. The essential idea in bagging is to average many noisy approximately unbiased models and hence reduce variance. Not all estimators can be improved by shaking up the data as we see in random forest but highly nonlinear estimators, such as trees benefit the most When used for classification, random forest obtains a class vote for each tree, and then classifies using the majority vote. When used for regression, the predictions from each tree at a target point x are simply averaged. An important feature of random forest is its use of out-of-bag (OOB) sample error estimate which is identical to that obtained by cross-validation (Hastie et al., 2009).

3.10.6 Boosting

Boosting is yet another example of a multi-tree ensemble technique that trains the *base classifiers* on different samples in a *sequential* order as opposed to the *parallel* building of decision trees, as in bagging. The core idea in the boosting technique is to carefully select samples to boost the performance on hard-to-classify instances, which were mislabeled. The technique works by selectively assigning higher weights to escape the trap of being misclassified. Schapire proposed the original theoretical framework for a set of classification problems that

turns *weak* learners into *strong* learners (Schapire et al., 1998). Freund and Schapire's ARCing (Adaptive Resampling and Combining) paradigm is the next significant development, as a general technique, which is more or less synonymous with boosting (Breiman et al., 1993; Freund and Schapire, 1997).

Briefly, the process begins with a base classifier $\boldsymbol{M_1}$, which is built on the training sample $\boldsymbol{D_1}$ and the misclassification error is evaluated, internally the misclassified instances $\boldsymbol{D_2}$ are selected at a higher probability for the next round of model training $\boldsymbol{M_2}$ and once again the misclassification error is evaluated. Likewise, in the subsequent round $\boldsymbol{D_3}$ the hard to classify instances by $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ are selected, again with a higher probability, and these iterations proceed through $\boldsymbol{K}$ times, until such time when the misclassification fails to be improved. Boosting is suggested to better at reducing *bias*, while bagging works better at reducing *variance*. Thus, unlike bagging that uses independent random samples for the input dataset, boosting employs weighted or biased samples to construct the different training sets, with the current sample depending on the previous error evaluations. Finally, the combined classifier is obtained via weighted voting over the output of $\boldsymbol{K}$ *base classifiers* $\boldsymbol{M_1}$, $\boldsymbol{M_2}$,..., $\boldsymbol{M_k}$.

3.10.7 XGBoost

XGBoost is a special case of a multi-tree-based ensemble algorithm that uses the gradient boosting framework. Chen and Guestrin pioneered the development, implementation, and deployment of the framework (Chen and Guestrin, 2016). XGBoost and Gradient Boosting Machines (GBM) both operate on the principle of boosting weak learners. However, XGBoost improves upon the base GBM framework through system optimizations and algorithmic enhancements (Chen and Guestrin, 2016). XGBoost approaches the process of sequential tree building using parallelized implementation. It approaches tree pruning differently than the standard stopping criterion as in a typical greedy search algorithm. The 'depth-first' approach significantly enhances computational performance. The algorithm has been designed to efficiently leverage hardware resources by the allocation of internal buffers in each tread to store gradient descent statistics, cache access patterns, utilize data compression, and sharding to build a scalable boosting system. Algorithmic enhancement includes regularization to prevent overfitting, sparsity awareness, and, weighted quantile sketch for optimal split points among weighted data points (Morde, 2019).

3.11 Decision Trees – Implementation & Experiments

Decision trees were implemented in the statistical programming language R. Bulk of the implementation was accomplished using the libraries *rPART* (Therneau et al., 2019), *randomForest* (Liaw and Wiener, 2018) based on the Fortran original Classification and Regression Trees (CART) (Breiman and Cutler, 2001), *randomForestExplainer* (Paluszynska et al., 2020), *XGBoost* (Chen and Guestrin, 2016), and *mlr* frameworks (Bischl et al., 2016). The other libraries include *ggplot2* (Wickham, 2009) for plotting besides using native R plotting utilities.

3.11.1 CART Models – Tree Structure, Validation, and Evaluation

Classification and Regression Trees (CART) is the decision tree learning models, the R implementation of the CART algorithm is called rPART (Recursive Partitioning and Regression Trees). Here we implemented rPART for JAK2 data with chemical descriptors feature set. It is a binary recursive partitioning that uses one of the several measures of impurity – information index or Ginni index that returns the maximum information gain or maximal impurity reduction. In other words, we are seeking to maximize the homogeneity of a given class by lowering the entropy and every subsequent split uses similar criteria until it reaches the terminal nodes for class prediction. First, we built a *base model* and evaluated its performance in k-fold cross-validation, rART is a single tree unlike ensembles, and more easily interpretable (see Figure 3.25). At the root node (C.027) we get an error value of 0.497, which is interpreted as having roughly 50% purity for the two classes and this relative error progressively decreases with successive splits until limited by a certain parameter. Next, we traced the cross-validation error by following the complexity parameter (cp) over an arbitrary range of values. The complexity parameter (cp) controls the size of the decision tree and selects the optimal tree size. If the cost of adding another variable is above the value of *cp*, the model tree building growth ceases. We see a slightly high error rate (40%) at reasonably small values of *cp*. In this case, the tree structure is rather sparse suggesting premature growth termination leading to poor performance. In general, tree pruning is adopted for the reduction of variance. In our experience, pruning did not alter the tree structure (data not shown).

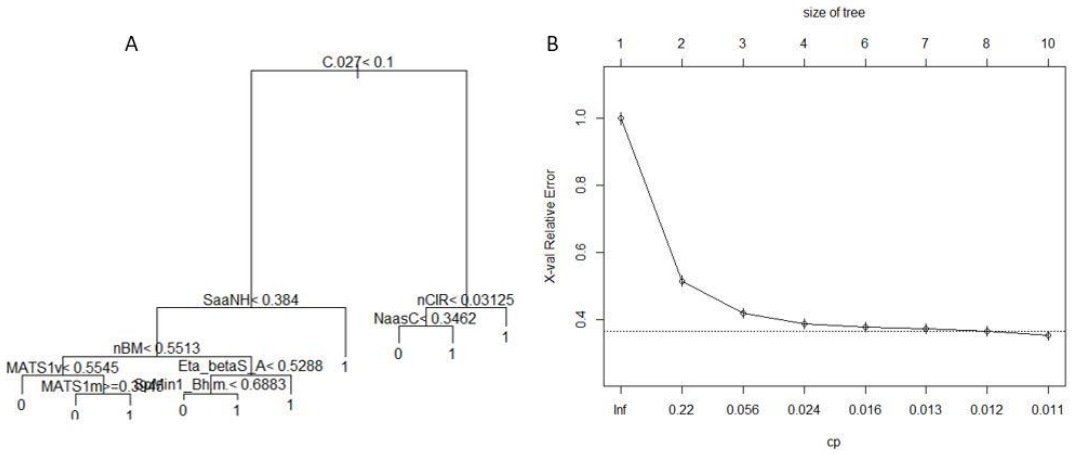
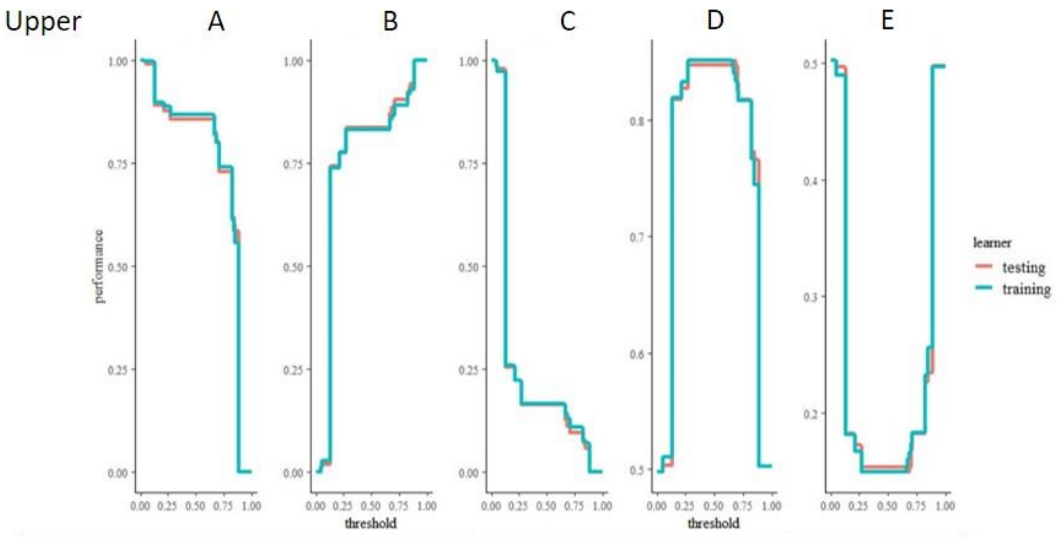


Figure 3.25 CART Tree Structure and k-fold Cross-validation
(A) JAK2 dataset tree structure (B) Model cross-validation error measurement.

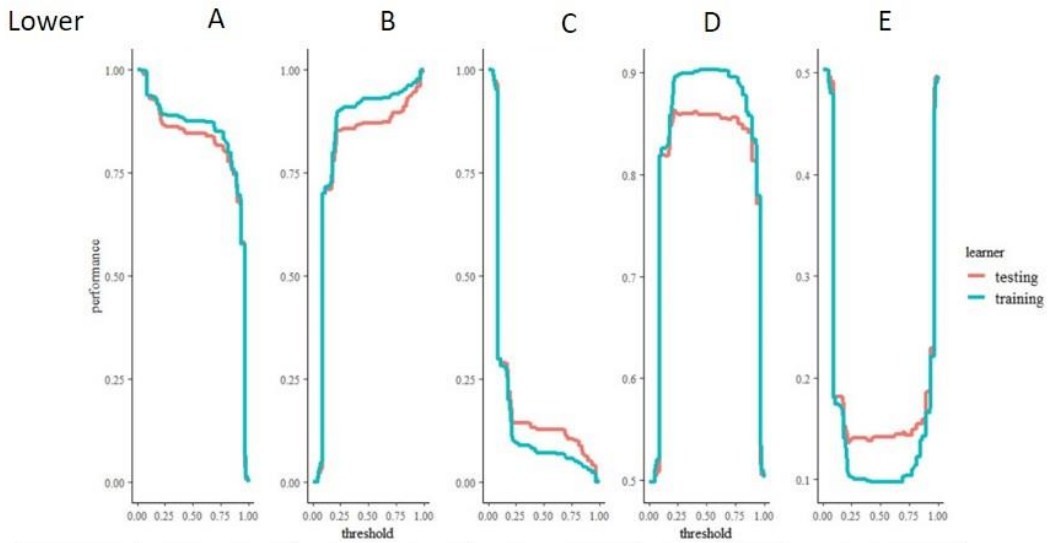
3.11.2 CART Models - Hyppertuning

The results suggest that we can leverage the tunable parameters tree depth (max depth) and *cp* to extract better performance from the model. Therefore, we set up a parameter tuning regimen for a computational grid search to obtain optimal values of max depth in the range of (1 - 20 in the increments of 1) and cp in the range of (0.001 – 10 in the increments of 10%). The search space is a cartesian joint of two tunable parameters. Each tree model uses two-thirds of the records for the training set and one-third as a test set and the model is cross-validated five-folds and the average error is computed. Here we report some gains in the model performance metrics, both before and after hypertuning (see Figure 3.26). The models are tested both on the training data and testing data, as expected the performance is slightly eroded in the testing data. Each of these model metrics is scored at different probability thresholds, which often is useful in choosing the appropriate threshold to remedy for class imbalance in data (Kuhn and Johnson, 2013). To concretely prove the point, we carried out simulation experiments with synthetic data with different ratios of class imbalance (i.e., ~1:1, ~1.5:1, and ~31:1 respectively). Next, we built single tree predictive models and showed how True Positive Rate (Sensitivity) and True Negative Rate (Selectivity) are affected at different probability thresholds. In balanced classes, sensitivity and specificity are approximately at the same level. In unbalanced class sensitivity (increases) and specificity (decreases) depending on the class ratio (see Figure 3.27). We further evaluated the model performance with ROC curves (see Figure 3.28). There is clear evidence for performance gains as we integrate the area under the curve, both before and after hypertuning confirming the improved performance both on the training data and the unseen test data. We also performed bootstrap

resampling and cross-validation to provide multi-point measurement as opposed to a single-point AUC measurement, and the confidence interval band was included in the measurement for robust validation.



	True positive rate	True negative rate	False positive rate	Accuracy	Misclassification error
Training	0.8360	0.8579	0.1421	0.8469	0.1531
Testing	0.8178	0.8571	0.1429	0.8373	0.1627



	True positive rate	True negative rate	False positive rate	Accuracy	Misclassification error
Train	0.9113	0.9109	0.0891	0.9111	0.0889
Test	0.8660	0.8571	0.1429	0.8616	0.1384

Figure 3.26 CART Model Hypertuning Performance Metrics

The upper panel shows base model metrics and the lower panel shows hypertuned model metrics at different thresholds. (A) True positive rate (B) True negative rate (C) False positive rate (D) Accuracy (E) Misclassification error.

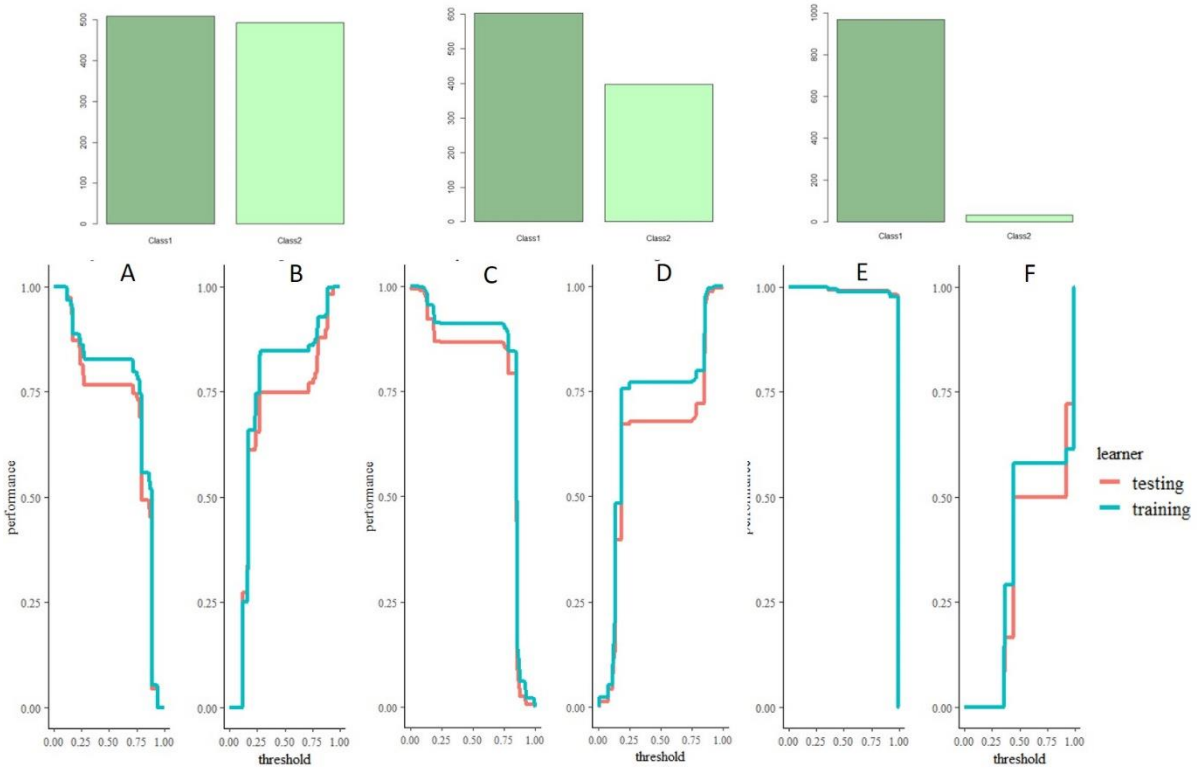


Figure 3.27 Class Imbalance Simulations

The figure shows model performance metrics with balanced classes (~1:1) (A) true positive rate (B) false-positive rate; with partially balanced classes (~1.5:1) (C) true positive rate (D) false-positive rate; with highly unbalanced classes (~31:1) (E) true positive rate (F) false-positive rate.

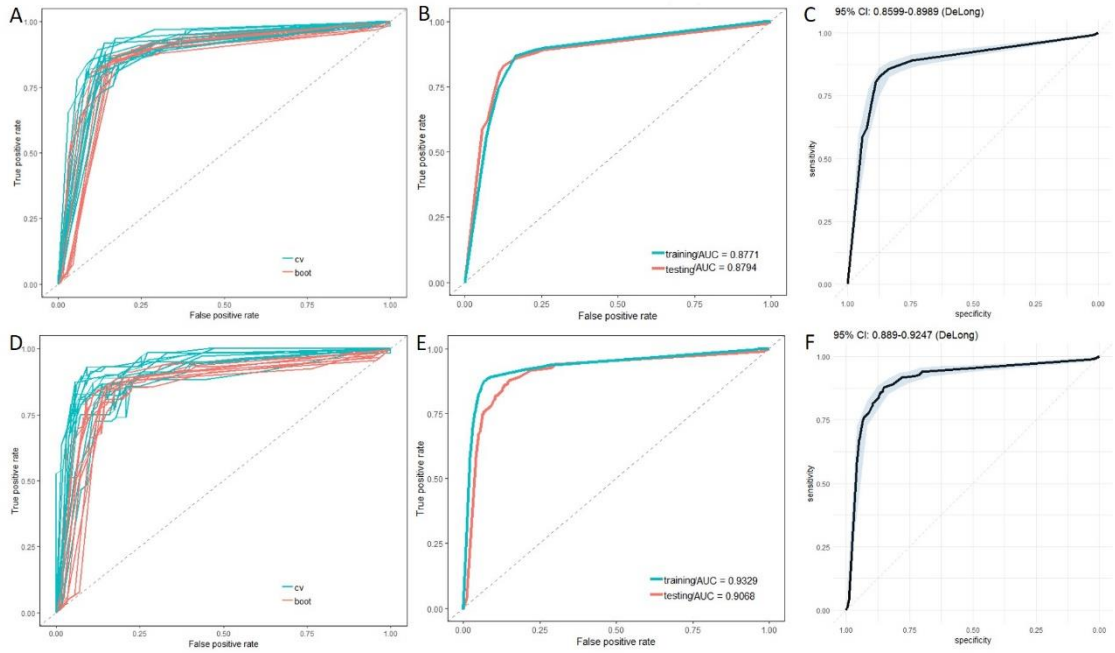


Figure 3.28 CART Hypertuned Model ROC Curves
ROC curves are plotted for *base* models (upper panel) and *hypertuned* models (lower panel) (A) Bootstrapped resampling and Cross-Validation (CV) (B) Testing vs. training data (C) Confidence Interval plot. (D) Bootstrapped resampling and Cross-Validation (CV) (E)Testing vs. training data (F) Confidence Interval plot.

3.11.3 Random Forest Models – OOB Error, Hypertuning, & ROC Curves

The random forest ensembles models are a collection of trees built by drawing random bootstrap resampling (sampling with replacements) from the JAK2 chemical descriptors training dataset. Besides, it also draws random sub-sampling of features to build the individual trees. As a general rule, roughly two-thirds of each draw is used for building the model and the other one-third is set aside as Out-Of-Bag (OOB) testing data until stable error estimates are reached. The class prediction for each of the testing samples is adjudicated by polling the votes from each of the sequentially aggregated trees and the class membership is decided by the majority vote. In these experiments, we have initially built 500 trees and followed the reduction in error, as called by the majority vote in each of these sequentially aggregated trees. While OOB error is the combined error of the two classes, the error of each class is also plotted. In the initial tree aggregates, the error is high but errors begin to stabilize approximately between 300 and 500 trees. The estimated OOB error is 8.18%. We also tested for the number of features (mtry) that is optimal in reducing the OOB errors and it turns out, that on an average it is 24 features in a randomly selected bootstrap sample, which is roughly the equivalent

to the square root of the total number of features. The OOB error as a function of mtry is in agreement with the earlier estimate of 8.18% (see Figure 3.29).

After building the *base models*, we tried to improve the model performance by leveraging the tunable parameters to build *hypertuned models*. Like in the earlier section, we set up a parameter tuning regimen for a computational grid search to obtain optimal values of ntree" (300 - 500), mtry (30 – 35), and nodesize (10 – 50). The search space is a cartesian joint of three tunable parameters that we are seeking to minimizes the mean misclassification error. Hypertuning experiments are computationally expensive and time-consuming. Often, finding the optimal combination is a non-trivial task. We evaluated the model performance gains by ROC curves, both on the training data and the unseen testing data. The AUC measurements yielded a value of 98.03%, on the testing set (see Figure 3.30), which is a significant improvement when compared with single tree rPART models on the same testing data, with an AUC value of 90.68% (see Figure 3.28). Prior to evaluating prediction performance on the unseen testing data, we validated the random forest model by bootstrap resampling and 10-fold cross-validation on the training data and plotted ROC curves for multi-point AUC estimates. We also developed sensitivity and selectivity curves with confidence interval bands for the general assessment of the robustness of these models.

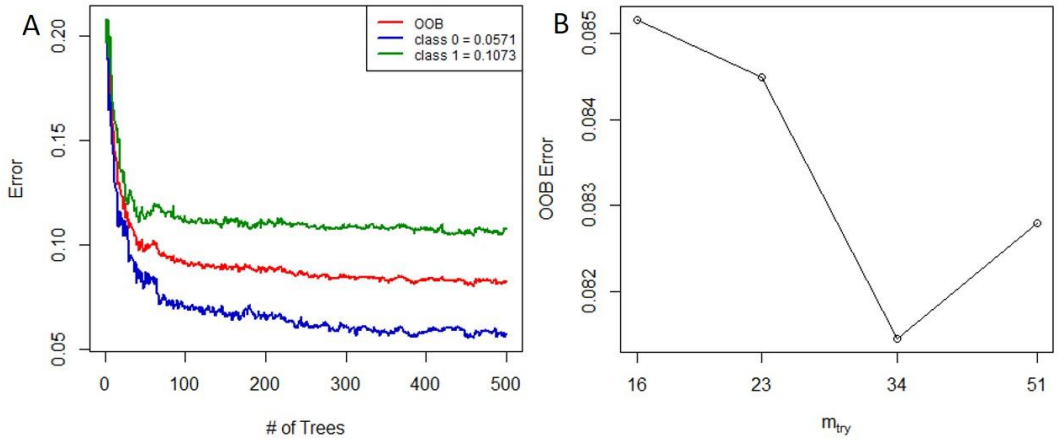


Figure 3.29 OOB Error Estimation and Model Tuning
(A) The errors are plotted for each of the two classes independently (green and blue lines and the combined class OOB (B) The plots show the optimal number of variables that are randomly selected in building the trees that give the smallest error.

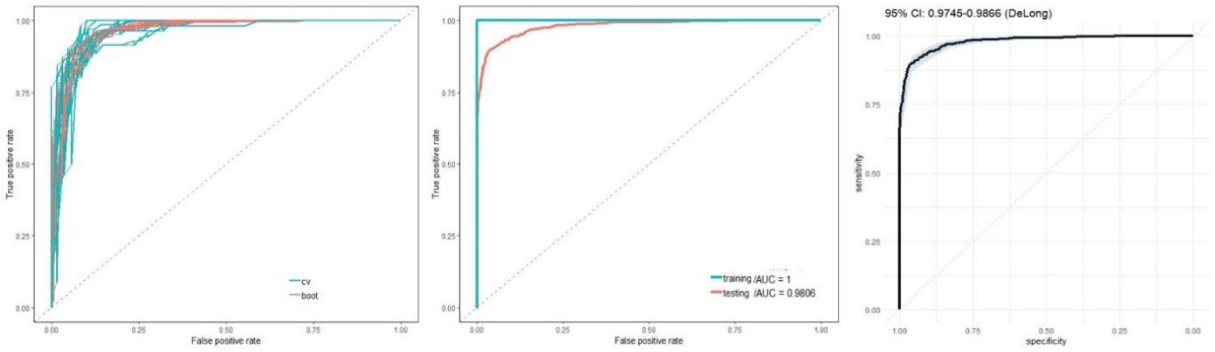


Figure 3.30 Random Forest Hypertuned Model ROC Curves
ROC curves are plotted for hypertuned models (A)Bootstrapped resampling and Cross-Validation (CV) (B) Testing vs. training data (C) Confidence Interval plot.

3.11.4 Random Forest Models – Variable Importance & Partial Dependence

Conceptually, in a binary decision tree, at each node a single predictor i_t is used to partition the data into two homogeneous groups. The chosen predictor is one that maximizes some measure of improvement $\hat{\imath}_t$, more precisely, the information gain. In the ensemble of individual decision trees, as in the random forest, the idea is extended to identifying such predictors and the improvement score is averaged over all the trees in the ensemble, which is called variable importance (VI). The index, Mean Decrease in Accuracy (MDA), utilizes the Out-of-Bag (OOB) samples to compute variable importance. For each tree, the predicted error on the OOB samples is recorded (misclassification error). And each predictor variable is permuted in the OOB samples and the predicted error is recorded once again. The difference between the two is then averaged over all the trees. The general equation is given as.

$VI_j = \frac{1}{ntree}\sum_{t=1}^{ntree} EP_{tj} - E_{tj}$ where EP is the permuted error of the predictor j and E is the base error

Thus, the Mean Decrease in Accuracy (MDA) measures the fraction of the misclassification errors in the model, when that predictor variable is removed (see Figure 3.31). In other words, the removal of P_VSA_LogP_5 can lead to a greater fraction of error compared to P_VSA_LogP_6. The larger the MDA value, the more important is the variable. On the other hand, the second measure Mean Decrease in Gini (MDG) is the total decrease in node impurities, averaged over all the trees. The most important predictor variable will have the largest Mean Decrease in Ginni (i.e., C.027) and conversely, lesser important predictors will have a smaller Mean Decrease in Ginni values (see Figure 3.31). Coincidentally, C.027 is the root node in

the single tree CART model (see Figure 3.25). Han and colleagues (Han et al., 2016) suggested a slight variation in the selection of variables in the random forest that combines MDA and MDG in a two-step procedure called the dichotomy method.

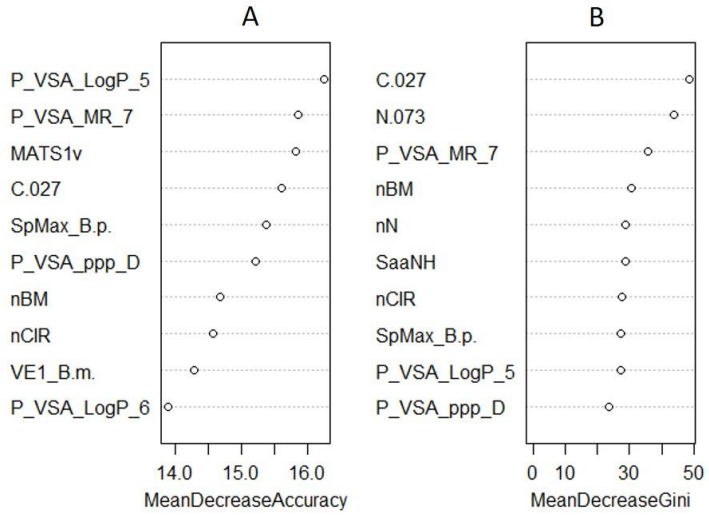


Figure 3.31 Random Forest Important Variables
The importance of these variables was determined by (A) Entropy and (B) Gini index measurements.

The Partial Dependence Plots (PDP) depicts the marginal effect of one or two features on the predicted outcome of the machine learning model (Friedman, 2001). In other words, the effect of an individual predictor variable, in total isolation of other predictor variables in the feature set on the predicted outcome. The function being plotted is defined as

$$f(x) = \log p_k(x) - \frac{1}{K}\sum_{j=1}^{k} \log p_j(x)$$

Where K is the number of classes, k is which class, and p_j is the proportion of votes for the class j.

In partial dependence plots, we show the relative logit contribution of the important variables derived earlier on the class probability in the random forest model (see Figure 3.32). For a subset of important variables, for example, the variable P_VSA_LogP_5 strongly predicts the Class '0' (inactive compounds) membership for values approximately below 0.75 and Class '1' (active compounds) membership above 0.75. Interestingly, there is a perfect *mirror symmetry* in the partial dependence curves for the two classes. This pattern of *mirror*

symmetry is repeated in all the cases examined for each of the predictor values in this subset of important variables.

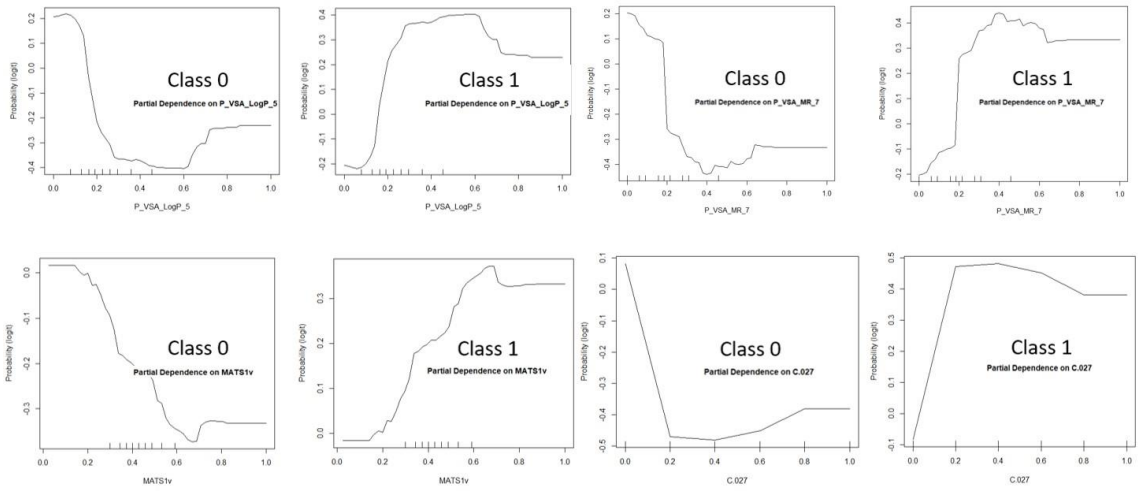


Figure 3.32 Partial Dependence Plots
The partial dependence plots for a subset of important variables for the two-class Random Forest classifier are shown.

3.11.5 Random Forest Ensemble Models – Distribution of Nodes & Tree Depth

The tree complexity of individual decision trees in the random forest depends on several hyperparameters that include minimum node size, maximum tree depth, the minimum number of samples to split

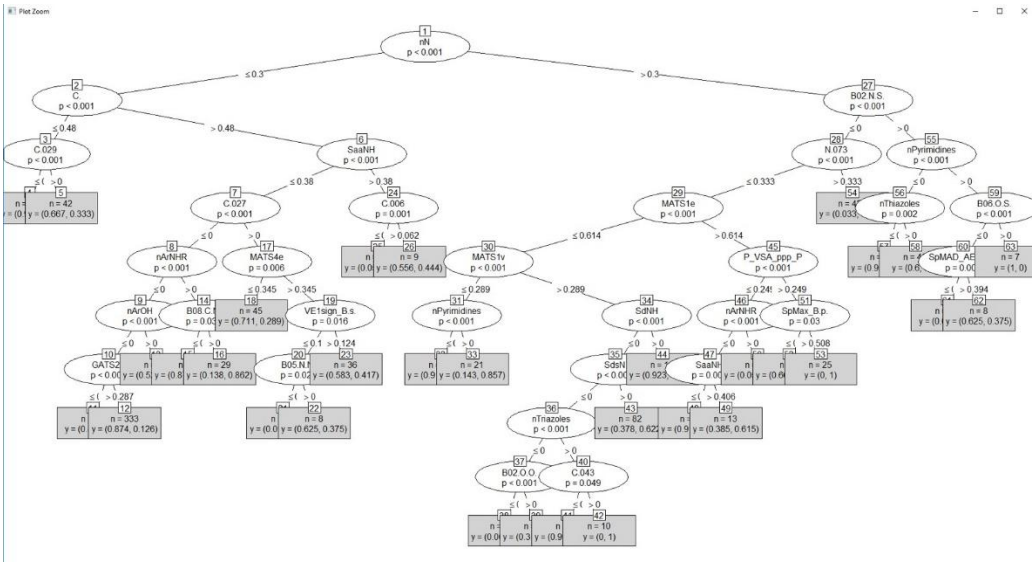


Figure 3.33 Random Forest Tree Structure
Here we show one of the representative trees in the ensemble collection with the root node, internodes, and terminal nodes leaf nodes.

on an internal node, the minimum number of samples at a leaf node, and decrease in node impurity among other properties. We show a representative tree structure with limited nodes in the random forest ensemble with a root node, internal nodes, and terminal nodes. (see Figure 3.33). From a collection of 500 trees, we estimated the distribution of the nodes, it ranged from 228 to 308. On average there are about 263 nodes per tree with +/- SD (12.76) and there are approximately 85 trees with 260-265 nodes per tree. On either extremes, there is a single tree with less than 230 nodes per tree and just three trees with more than 300 nodes per tree. We plotted the probability density function (PDF) over continuous node interval which gives a better insight into the shape of the distribution (see Figure 3.34). We also calculated the mean minimal depth and its mean. The smaller the value of the mean minimal depth of the tree it shows these nodes are higher up in the hierarchical distribution of the canonical tree structure. In other words, it is either the root node or a node closer to the root node (see Figure 3.34). The same subset of predictor variables listed in Figure 3.31 re-appear in Figure 3.34 and Figure 3.35 with the suggested different depths in the hierarchy.

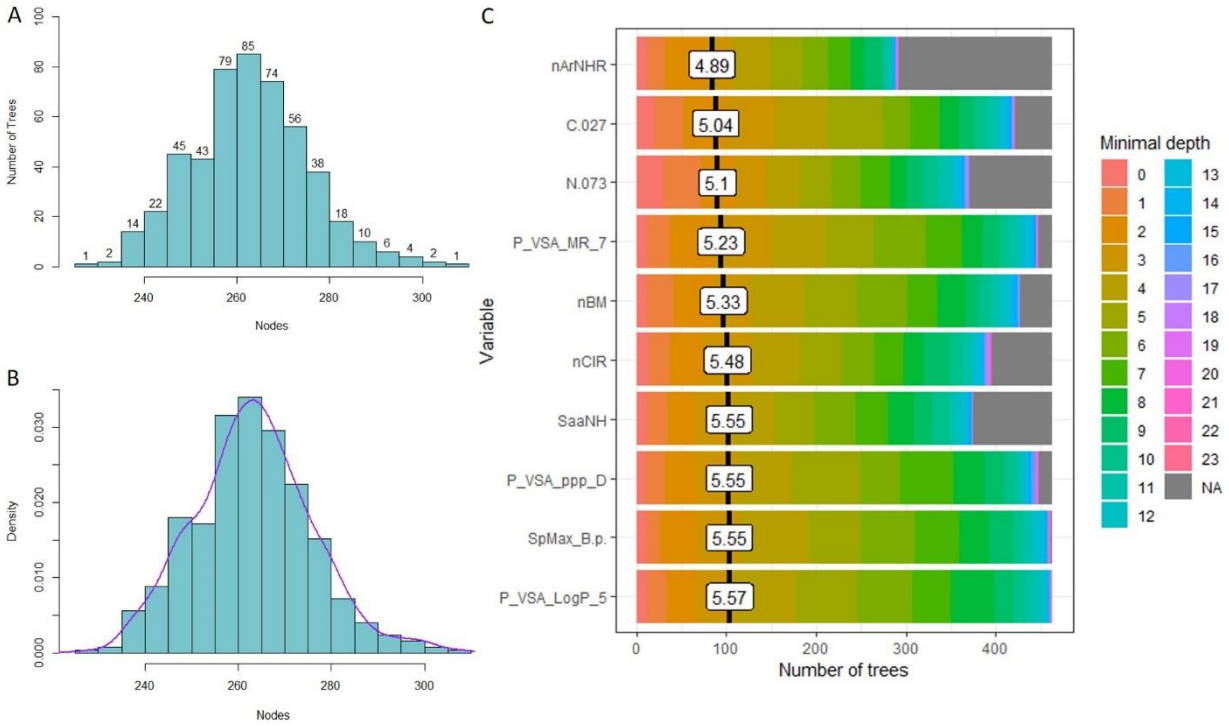


Figure 3.34 Random Forest Distribution of Nodes and Node Depth
The figure shows (A) the distribution of the nodes in the forest (B) the probability density plot (C) the distribution of the variables across the number of trees in the forest and at the depth they appear and their mean.

The Figure 3.34 (C) offers an interesting insight into three aspects – (1) the appearance of a predictor variable at the root node, (2) the appearance of predictor variable at different depths, (3) the appearance of predictor variable in the number of trees in the forest. While nArNHR is ranked above others it appears in far fewer trees (<275) in the forest. On the other hand, the predictor variable C.027 is ranked the next highest but appears in more trees (>450). The bottom three predictor variables in Figure 3.34 (C) appear almost close to 500 trees in the forest. It is important to consider the participation of a predictor variable in multiple splits within the growing tree, which is highlighted by inspecting the multi-way importance plot (see Figure 3.35). Therefore, predictor variables C.027, P_VSA_ppp_D, SpMax_B.p., and P_VSA_LogP_5 are considered significant.

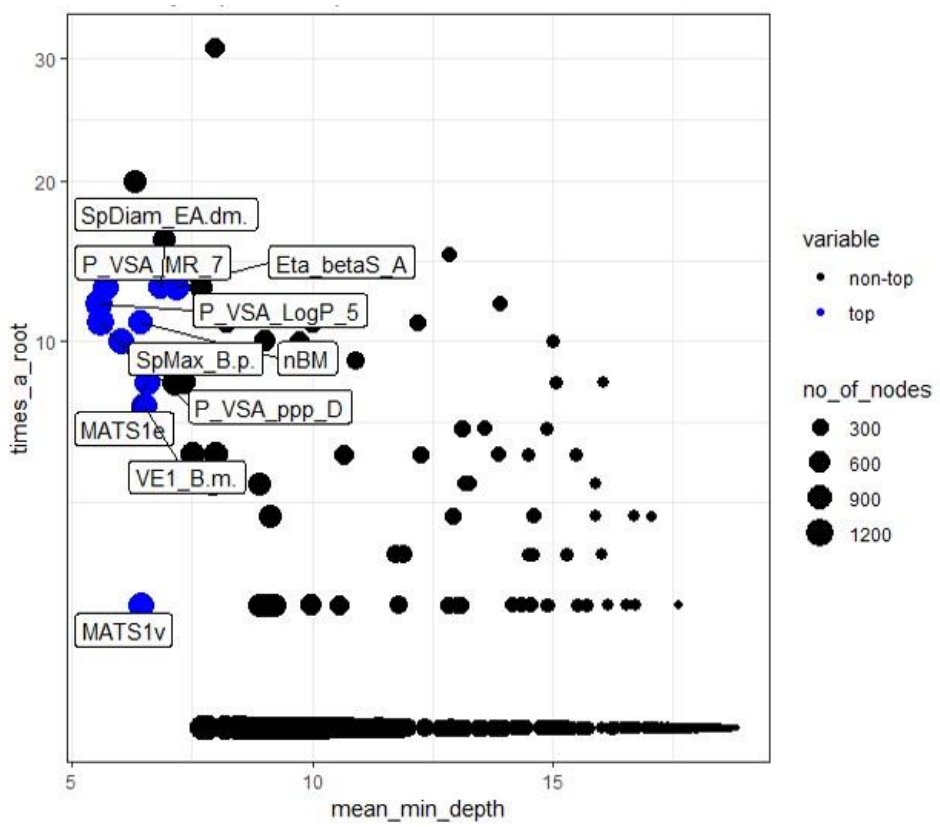


Figure 3.35 Multi-way Importance Plot
This is a plot of the most important variables that appear at the root node and their representation at various depths in all the trees in the ensemble.

3.11.6 Random Forest - Variable Interactions

Breiman and colleagues (Breiman and Cutler, 2001) suggested that in the random forest the operating definition of interaction used is that variables m and k interact if a split on one variable, say m, in a tree makes a split on k either systematically less possible or more possible. The implementation used is based on the Gini values g(m) for each tree in the forest. These are ranked for each tree and for each two variables, then the absolute difference of their ranks is averaged over all trees.

This number is also computed under the hypothesis that the two variables are independent of each other and the latter is subtracted from the former. A large positive number implies that a split on one variable inhibits a split on the other and conversely. We plot such pair-wise variable interactions after enumerating the splits in individual trees and across all the trees in the forest (see Figure 3.36) taking advantage of the library functions (Paluszynska et al., 2020). The results suggest that a split on one of the predictor variables has

systematically increased the splits of the cognate pair. Wright and collaborator (Wright et al., 2016) suggested, using extensive simulation studies, that caution is warranted in over interpretations of variable interactions because it is difficult to differentiate between the real interaction and marginal effects.

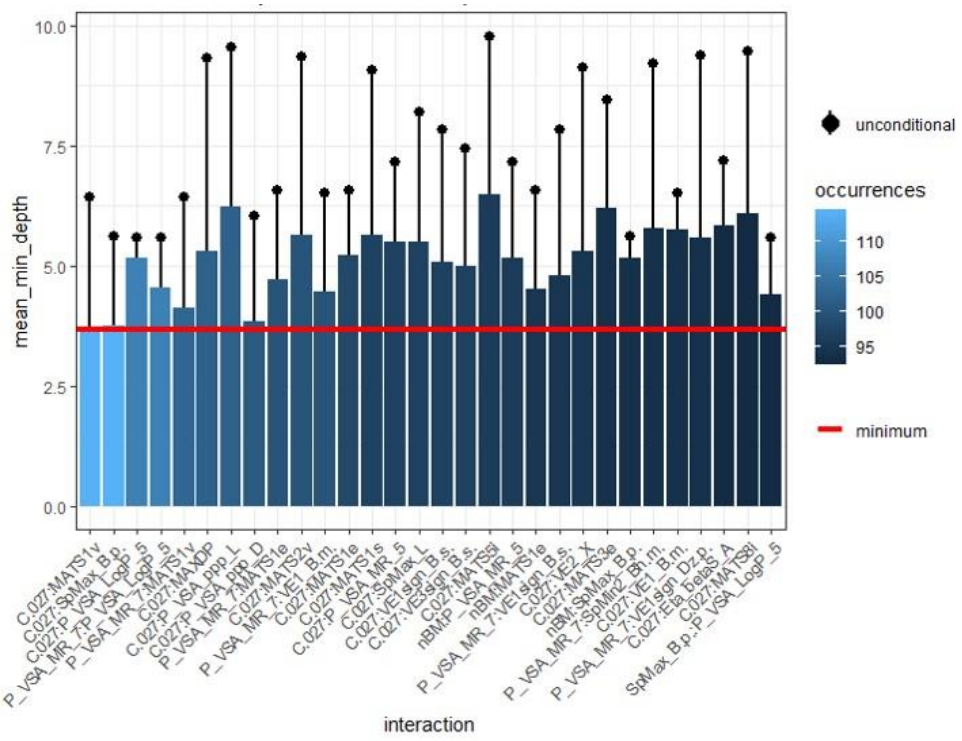


Figure 3.36 Random Forest Model Feature Interactions
The plot shows the feature depth and feature interactions.

3.11.7 XGBoost Ensemble Models - Trees, Validation, and Evaluation,

The XGBoost ensembles models are the special case of boosting techniques where a collection of trees are built iteratively by tracing the decrease in misclassification error on the training data, as an ensemble of weak prediction models until the error values are stabilized after certain epochs (see Figure 3.37). We developed XGBoost ensemble models using the JAK2 dataset. Initially, we began with a large ensemble of 1000 trees, progressively reduced it to 50 trees, and recorded the errors. The misclassification error was found to stabilize between 50 and 1000 trees, which guides model hypertuning.

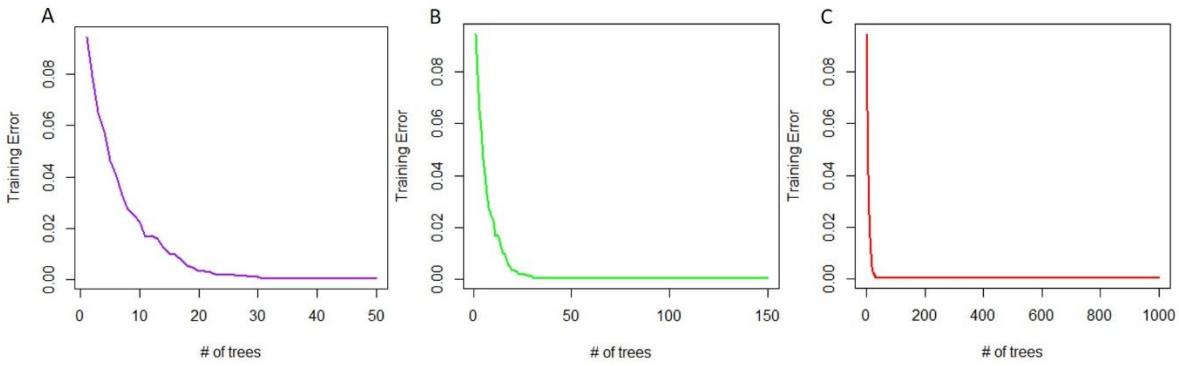


Figure 3.37 XGBoost Models Training Error
The figure shows the trace of the progressive lowering of training errors in ensembles of different epochs (A) 50 trees (B) 150 trees (C) 1000 trees.

Prior to evaluating prediction performance on the unseen testing data, we validated the XGBoost models by bootstrap resampling and 10-fold cross-validation on the training data and plotted ROC curves for multi-point AUC estimates (see Figure 3.38) After building the XGBoost *base models*, we tried to improve the model performance by leveraging the tunable parameters to build *hypertuned models*.

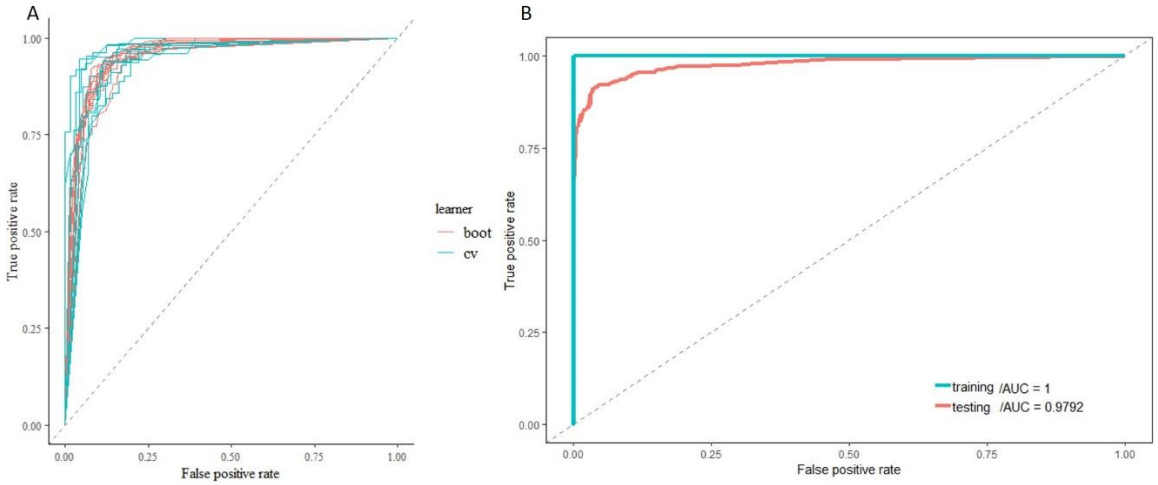


Figure 3.38 XGBoost Hypertuned Model ROC Curves
ROC curves are plotted for (A)Bootstrapped resampling and Cross-Validation (B) Testing vs. training data.

Like in the earlier sections pertaining to decision trees, we systematically set up a parameter tuning regimen for a computational grid search to obtain optimal values of the number of epochs; it involved the following

parameter ranges: nrounds (100, - 500), mtry (30 – 35), tree depth: max_depth (1– 10), learning rate: eta (0.1 – 0.5), and L2 regularization: λ (-1 – 0). The search space is a cartesian joint of four tunable parameters that we are seeking to minimizes the mean misclassification error. Besides, we computed standard model performance metrics – accuracy, false-positive rate, false-negative rate, and misclassification error (data not shown). Hypertuning experiments are computationally expensive and time-consuming. Often, finding the optimal combination is a non-trivial task. We evaluated the model performance gains by ROC curves, both on the training data and the unseen testing data, the AUC measurements yielded a value of 97.92%, which is slightly less than random forest models 98.03%, on the testing set (see Figure 3.38, also Figure 3.30 for comparison).

3.11.8 Benchmarking Different Machine Learning Models

In general, when evaluating different machine learning algorithms, it is imperative to provide empirical evidence that illustrates the behavior of any given algorithm compared to other competing algorithms. In the field of machine learning and statistics, such empirical studies are called benchmark experiments. In benchmark experiments, different learning methods are applied in tandem to a dataset to compare and rank an algorithm with respect to one or more performance measures. We conducted some benchmark experiments using the *mlr* framework (Bischl et al., 2016).

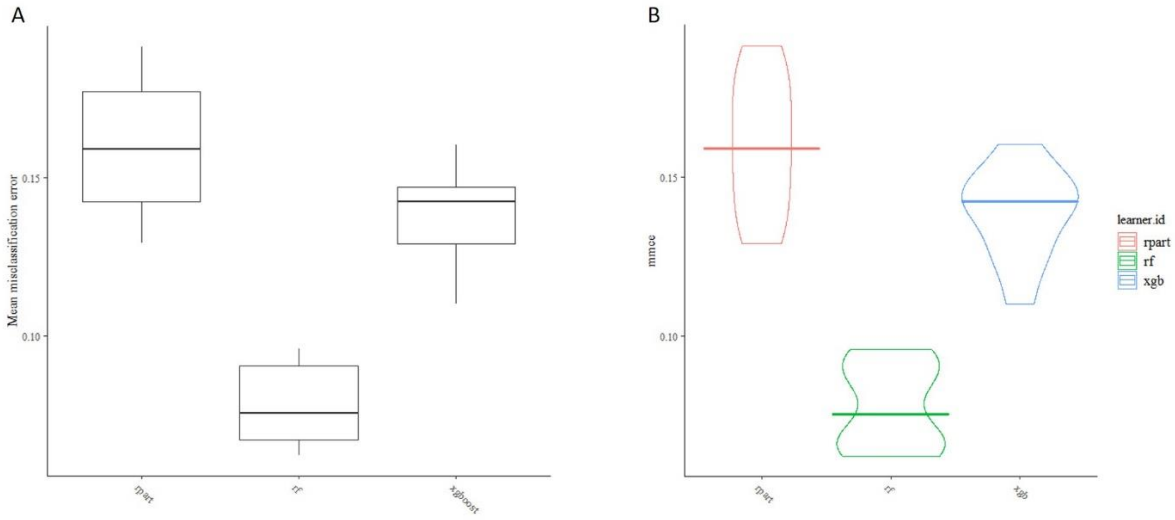


Figure 3.39 Comparison of Decision Tree Models
The plot shows the mean misclassification errors of the three models being compared – Single Tree (rPART), Ensemble Random Forest, and XGBoost in two formats (A) Box plots (B) Violin plots.

The framework offers a standardized structure, syntax, environment, and methods to accomplish such a task. While there are other frameworks/toolkits, for example, caret (Kuhn et al., 2016), CORElearn (Robnik-Sikonja and Savicky, 2020), rattle(Williams et al., 2020), ipred(Peters et al., 2019), and rminer (Cortez, 2020), all of which in some form or other support classification and regression tasks but none provide the mlr's generic wrapper mechanism, which makes the task of comparison facile. In the following experiments, using a common dataset, JAK2, we selected three machine learning algorithms from the decision tree family and evaluated a single performance metric – mean misclassification error (see Figure 3.39). The random forest was ranked with the lowest mean misclassification error, followed by XGBoost and CART models. Like-wise we extended the benchmark experiments to include all algorithms that we implemented, models developed and evaluated in the present study (see Figure 3.40).

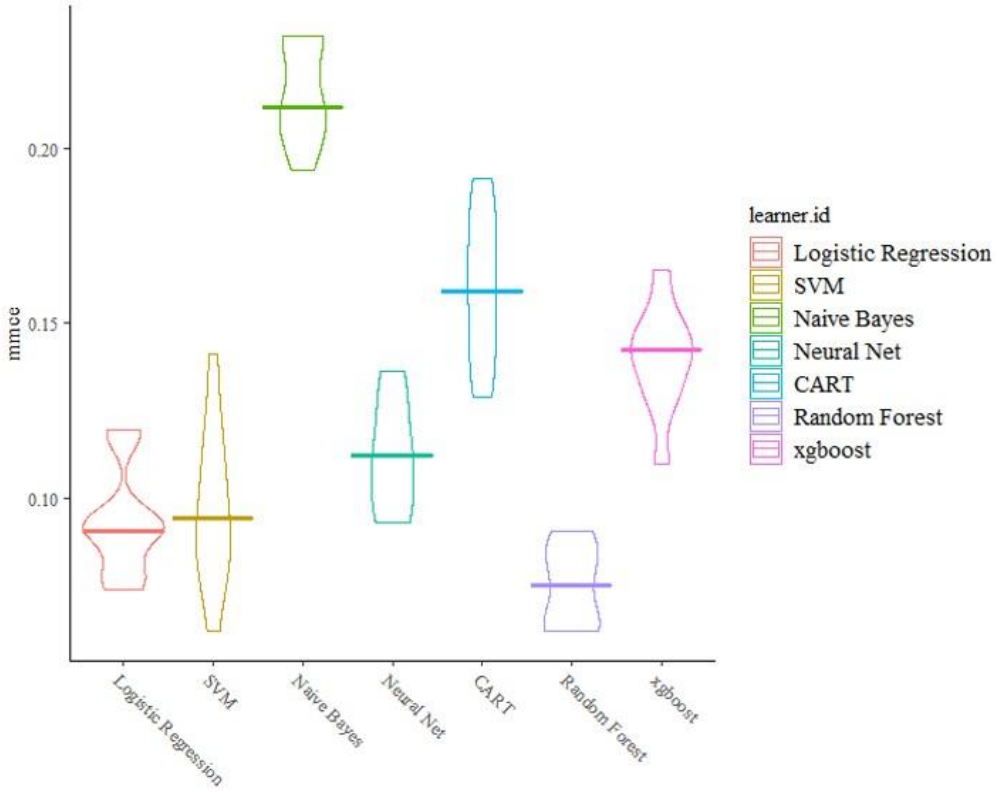


Figure 3.40 Benchmarking Machine Learning Models
The violin plot shows the mean misclassification errors of the Machine Learning Algorithms being compared for the same dataset under identical conditions – Logistic Regression (LR), Support Vector Machines (SVM), Naïve Bayes (NB), Artificial Neural Nets (ANN), Classification and Regression Trees (CART), Random Forest (RF), and XGBoost (XGB).

Interestingly, ensemble learning methods seem to outperform other learners. The rank order is as follows: Random Forest was ranked the highest with the lowest mean misclassification error, and Naïve Bayes was ranked the lowest with the highest misclassification error, while the rest of them were stacked between them in rank order. It must be mentioned that the entire dataset was included in the cross-validation resampling, with no attempt to tune the individual models. Logistic regression is relatively robust (or insensitive) to hyperparameter tuning while we have clearly demonstrated that hypertuning in the case of support vector machines, artificial neural nets, and decision tree ensembles yielded better model performance.

Chapter 4: Discussion

"You are your deepest desire; as is your desire, so is your intent; as is your intent, so is your will; as is your will, so is your deed; as is your deed, so is your destiny." - Upanishads

4.1 Systems Thinking in Disease and Drug Discovery

Although the present study is framed from a systems perspective in understanding biological complexity, human diseases, and drug discovery. It must be emphasized that the critical focus of this study is on *evaluating machine learning algorithms* in the early discovery phase of profiling oncology drug candidates. Looking through the systems thinking lens offers a rich repertoire of possibilities for interrogating human diseases and injecting innovations in modern drug development campaigns.

As we alluded earlier, reductionism and systems thinking have been competing ideas that have dominated scientific discourse over the years. The biological systems are complex and have emergent properties. It must be emphasized, the understanding or prediction of their behavior is not a trivial task, that it can not be accomplished by simply analyzing their constituents. The constituents of a complex system interact in a myriad ways, including positive feed-back, negative feed-back, feed-forward that leads to dynamic features that cannot be predicted satisfactorily by linear mathematical models alone that disregard cooperativity and non-additive effects (Aderem and Smith, 2004). At least in the area of cancer/oncology, there is an increasing call for a 'systems' approach. Incorporating the concepts of 'emergence', 'systems', thermodynamics', 'chaos', and fractals into a single integrated framework for oncogenesis. On some fundamental principles, cancer is recognized as a dynamic complex system, emerging at the level of a 'functional tissue unit' and is associated with multiscale causality at different levels in the hierarchy concomitantly (Sigston and Williams, 2017).

We now know that most diseases are caused by defects not in a single gene, but by several genes and/or gene products (Mitchell, 2012). Network biology offers a broad framework to orient towards systems thinking, as the basis for human diseases. Its importance is emerging with the availability of data on protein-protein interactions, DNA-protein interactions, and RNAi in disease etiology. For example, the prescription drugs, tamoxifen, bicalutamide, and all-trans retonic acid target interactions between DNA and transcription factors (estrogen, androgen, and retinoic acid receptors), Azacitine targets DNA methyltransferases, while vorinostat and romidepsin target histone deacetylase. There are also descriptions of how cells are organized networks of metabolic and signal transduction cascades. There are many– 'omics' foci, including genomics, epigenomics, transcriptomics, proteomics, lipidomics, metabolome, and the glycome, as well as the exposome. They are a

collection of all the genes, epigenetic modifications, transcription products, proteins, lipids, metabolites, and glycans (carbohydrates) in a cell or an organism (Baker, 2013; Smith et al., 2015)

Network-based drug discovery has been proposed to identify all the nodes that affect a disease network and display higher-order phenotypes (Schadt et al., 2009). While it may be tempting to undertake a large-scale compound screening to identify drugs that restore hemostasis to such perturbed networks, the prospects for success is somewhat limited. On the other hand, the approach of polypharmacology offers some advantages, where a limited number of target nodes in a canonical signaling network in cancer/oncogenesis, are simultaneously interrupted at multiple points to short-circuit receptor-mediated biological signals and attenuate down-stream cellular processes to achieve the desired therapeutic outcomes (Boran and Iyengar, 2010). Machine learning modeling will play a key role in this enablement. At a much broader level, the synergies of combining *systems thinking* and *machine learning* offer some radical prescription for innovation across the entire pharmaceutical value chain from R&D, clinical trials, and commercialization that will have a significant impact on delivering next-generation medicines.

4.2 Machine Learning in Oncology Drug Discovery

In the present study, we demonstrate how different machine learning algorithms can be used to mine and model large volumes of potentially druggable compounds in the early phase of compound screening for drug discovery, with particular emphasis on oncology drug candidates. Although PubMed and ChEMBL are extraordinary resources in the public domain, they suffer from some minor data quality issues. Therefore, we accessed the resources funded and developed by AstraZeneca Sweden. The ExCAPE database, which provides a more standardized form of chemical structures, activity annotations, and target identifiers, covering a large chemical and target space harvesting both public domain resources and in-house repositories (Sun et al., 2017).

The standard small-molecule drug discovery campaign starts by screening large molecular libraries against specific protein targets and assaying them for their potential binding, biochemical readouts, and setting up phenotypic screens. Testing these large compound libraries with specific biological activities is very expensive and time-consuming. Building predictive machine learning models for such large-scale screening activities can potentially save resources, time, and financial overheads. The adoption of computational approaches in drug discovery work-flow offers an attractive alternative (Lo et al., 2018).

Our focus and efforts were directed at prosecuting large-scale molecular libraries to identify potential oncology drug candidates for a subset of experimentally validated oncology targets by machine learning approaches. Multiple bioassays directed to a given target were pooled. To our knowledge, this is the first attempt at such

a large-scale aggregation of bioassays. We extracted feature sets for each of these compound collections by computing chemical descriptors and chemical fingerprints.

Standard data mining preliminaries for machine learning modeling included checking for data distributions, missing values, balanced classes, removal of duplicates, removal of highly correlated predicted variables, data normalizations, evaluating sparse matrices to speed computations.

Often one of the basic flaws in building machine learning models is overlooking the fundamental prerequisite, i.e., data normalization and class imbalance. Normalization helps in scaling the data, for training stability, and improving the model performance. Besides, several machine learning algorithms, for example, logistic regression, support vector machines (SVM), and artificial neural nets use optimization techniques like *gradient descent* and *stochastic gradient descent* for faster convergence to local minima. It's well known that decision trees i.e., boosting (random forest models) or bagging (XGBoost models) are less sensitive to normalizations (Bishop, 2006; Borkin et al., 2019). In our experience range transform, Box-cox transform (Box and Cox, 1964), and Yoe-Johnson transform (Yeo and Johnson, 2000) more or less were alike in model performance, while Z-transform was slightly inferior. On the contrary, a study involving testing different normalization techniques with the Nearest Neighbor Classifier, reports that Z-transform and Pareto scaling performed better compared to mean-centered, variable stability scaling and other techniques (Singh and Singh, 2019).

In a typical large-scale screening of molecular libraries, only a small fraction of the compounds being screened turn out to be active binders. Naturally, such skewed class distribution leads to building some, if not all, erroneous models. However, many of these concerns were addressed in ExCAPE data by rebalancing with inactive molecules (Sun et al., 2017) and in the preprocessing phase, we corrected the situation, by applying some routine resampling techniques to attain approximate class balance (Kuhn and Johnson, 2013). In simulation experiments with Classification and Regression Tree (CART) models we developed, we provide visual proof on how class imbalance affects sensitivity and specificity measures at different probability thresholds. One of the most widely used techniques for correcting the class imbalance in data is the Synthetic Minor Oversampling Technique (SMOTE) (Chawla et al., 2002). Zakharov's paper examines several approaches, in great detail, regarding correcting for class imbalance and makes an argument that probabilistic models like Naïve Bayes are more tolerant in dealing with unbalanced classes (Zakharov et al., 2014).

Extracting features sets, especially chemical fingerprints, can lead to producing very large matrices ($>$ 10,000 columns), with zeros in several columns, because we are performing binary encoding of certain substructural features (e.g., aromatic groups, carbonyl groups, hydrogen acceptors, and donors, etc) in a molecule. By

contrast, chemical descriptor feature sets are typically dense matrices with predominantly populated with non-zero values, which are usually continuous measurements derived from chemical graph invariants. Such matrices with mostly zero values (i.e., no data), when performing numerical operations involving several zeros lead to increased time complexity of matrix operations. Thus, there arose an interest in sparse matrices for chemical fingerprints, because its exploitation can lead to enormous computational savings (Brownlee, 2018; Press et al., 2007). We generated sparse matrices to build logistic regression and support vector machine (SVM) models. While sparsity and robustness are closely related, the exact relationship and subsequent trade-offs are not always transparent. For example, complex penalties like LASSO are often motivated by sparsity considerations, yet the success of these methods is also driven by their robustness (Copenhaver, 2018).

The collinearity diagnostics of feature space is critical in evaluating the redundancy in the information content of features. Pruning features, which are referred to as dimensionality reduction leads to the building and evaluation of *full* and *reduced* models, especially when the feature space is vast, as in computing fingerprints. We demonstrate that reduced logistic models are stable over a range of feature space via recursion in evaluating model accuracy. Similar feature engineering approaches were used for building artificial neural networks (ANN) chemisorption models for catalyst design in chemical synthesis (Li et al., 2017). Automated methods of selection of features, for example, molecular descriptors using random forest over ad-hoc methods have been suggested (Cano et al., 2017). Eklund and colleagues by evaluating different feature selection methods (for example, wrapper, RelefF, MARS, and elastic nets) found no evidence that a particular feature selection method was well-suited for a particular learner (Eklund et al., 2014).

We demonstrate how model regularization leads to its generalizability on test data, we show this phenomenon in two cases we examined, namely, logistic regression and artificial neural nets (ANN) models. After all, regularization is a penalty against model complexity. In logistic regression Elastic net regularizer performs slightly better than LASSO (L1) and ridge (L2) under AUC, we also trace the coefficient shrinkage paths as we perform these regularizations. In artificial neural net models, neuron drop regularization stacked above LI and L2 regularizations in preventing model *overfitting*. A combination of least absolute shrinkage and selection operator (LASSO) with Bayesian Regularization feed-forward artificial neural network (LASS-BR-ANN) was used as a new approach to quantitative structure-activity relationship (QSAR) studies (Mozafari et al., 2020). Altman's group showed that shallow representation learning improves the accuracy of L1 regularized logistic regression (LASSO) using curated drug screening data (Rensi and Altman, 2017). Using imaging recognition data for classification, neural dropout regularizer was found to be more effective than L2-norm for complex neural networks (Phaisangittisagul, 2016). This is particularly relevant to phenotypic screening in drug discovery.

SVM with non-linear Radial Basis Function (RBF) kernel produces superior model performance as evaluated by ROC curves. Similar kernel behavior was reported for kernel evaluation of C-aryl glucoside SGLT2 inhibitors (Prasoona et al., 2013) and with the vibrational signal data (Elangovan et al., 2011), In a different study in image recognition experiments SVM with exponential Laplacian kernels seem to outperform others (Fadel et al., 2016). Unexpectedly, the sigmoidal kernel, in our experience consistently failed across all the datasets tested. This was a bit of a conundrum because we were unable to provide a clear interpretation. Providentially, this explanation was provided in one of Vapnik's original publication, which we unwittingly overlooked, that the sigmoidal kernel may not be positive semi-definite (PSD) for certain values of the parameters a and r. When K is not PSD the expression of projection in the higher dimensional space is not satisfied and the primal-dual relationship does not exist (Vapnik, 1995). Frohlich and others have shown that the adoption of new kernel methods, which they developed, can achieve generalization performance compared to a classical model with a few descriptors which are *apriori* known to be relevant to the problem (Fröhlich et al., 2006; Prasoona et al., 2013).

Naïve Bayes is a simple robust classifier based on the naïve assumption of predictor variable independence. Though it produced reasonable models with our datasets, the presumption of predictor variable independence is either violated or tolerated, because we clearly demonstrate predictor variable interaction with similar data building non-parametric random forest models. Therefore, we cannot adequately resolve the issue without the fallacy of equivocation. However. there are numerous computational procedures for probing such interactions in low-dimensional data. For example, Friedman's H-statistic deals with two basic cases, first, two-way interactions which tell to what extent two features in a model interact with each other; second, a total interaction measure that tells us, whether and to what extent a feature interacts in the model with all other features (Friedman and Bogdan, 2008). Other approaches include a proposal by Hooker (Hooker, 2004), wherein the method measures the loss associated with the projection of the prediction function into a space of additive models, which closely corresponds with the functional ANOVA decomposition; a well-developed construct in statistics.

Artificial Neural Networks (ANN) have generated valuable Quantitative Structure-Activity/Property relationships (QSAR/QSPR) models for a wide variety of small molecules and material properties. However, the debate about the choice between shallow (single hidden layer) and deep neural networks (multiple hidden layers), with respect to QSAR predictions is still open to interpretation. Our results show that ANN models with multiple hidden layers, to a point, scored better both on the accuracy and loss function measurements, at least on this particular dataset. However, the performance eroded beyond a certain number of layers. This

is slightly inconsistent, if not a dramatic departure from the universal approximation theorem, which states that a single layer with finite neurons is capable of universal approximation of any continuous function when given appropriate weights (Csáji, 2001). On the contrary, multilayer neural networks explored in the earlier QSAR studies found no advantage over single-layer ANN models (Nakama, 2011). The response surface discontinuities correspond to so-called "activity cliffs", an important issue observed in drug design and optimization. Activity cliffs occur when small changes in the drug molecule result in large changes in biological activity (Maggiora, 2006). While this is in part, attributed to the descriptor representation of a critical substituent that participates in the chemistry that confers the biological activity, for instance, via a salt bridge, as a proton donor, through electron delocalization, as the case may be. In general, non-linear mapping algorithms like neural networks, random forest, and support vector machines perform better at ameliorating such response surfaces (Winkler and Le, 2017).

The authoritative paper by LeCun, Bengio, and Hinton reviews the whole field of deep neural networks and its application across disparate domains ranging from speech recognition, visual object recognition, genomics to drug discovery (LeCun et al., 2015). The power of DNN is in its sophisticated representational learning and adjusting the internal parameters in reducing the differentiable loss function compared to multilayer neural networks. Using Kaggle drug-like molecule datasets, two groups make interesting claims. Ma (Ma et al., 2015) showed that DNN routinely makes better prospective predictions than random forest RF scoring on the root-mean-square error. Winkler (Winkler and Le, 2017) based upon the dispersion measure, standard error of prediction (SEP) showed that shallow and deep neural networks performed differently on different targets.

We found in our experiments that ANN models compiled with ADAM optimizer (Kingma and Ba, 2017) delivered superior models compared to Stochastic Gradient Descent, which is the most recognizable first-order optimizer in the learners space. Choi and colleagues in their paper on empirical comparison of optimizers for deep learning suggest that hyperparameter search space is the single most important factor in optimizer ranking (Choi et al., 2020). Interestingly, in an analysis of gradient descent-based optimization algorithms on convoluted neural networks on image classification datasets, Nesterov-accelerated Adaptive Moment Estimation (NADAM) performed the best in terms of convergence, accuracy, and loss function (Dogo et al., 2018).

Tuning learning rates is critical in updating the weight parameter, our results with ANN models show that small values of learning rates led to greater model stability. Smith (Smith, 2017) proposed autonomous methods in optimal learning rates by bouncing values between shifting bounds that exhibit multiple modes of decay from fixed to exponential. Stochastic Gradient Descent with warm Restarts (SGDR), is another

approach that is predicated on an aggressive annealing schedule (i.e., where finding approximate global optimum is more important than finding the precise local optimum) combined with periodic restarts with the original learning rates (Loshchilov and Hutter, 2017).

We built both single tree classification and regression tree (CART) and multi-tree ensemble methods, namely, random forest (RF) and XGBoost. In general, decision trees are more robust and less sensitive to feature scaling, outliers, and high-dimensional data. The intuition is that the tree builds hyper rectangles in the data space. For each dimension, an edge of a hyper rectangle is defined by a point. Since the point only depends on the *relative* position, the tree will be the same regardless of whatever monotonic transformation is applied to any feature.

To take full advantage of decision tree models, we set up hypertuning regimens for each of the models. Hypertuning is a non-trivial task because we are setting up cartesian joints of each of the available parameters which are computationally very expensive and then evaluate performance gains. When training models, often, interventional bootstrap resampling cross-validations provide some insights into the trajectories of performance gains, to guide us in tuning model parameters. Bootstrap is an alternative to asymptotic approximation for carrying out inference. While delivering high performance continues to be the goal, model *interpretability* is equally desirable. For example, knowing which features are most predictive of the outcome in RF models. Therefore, feature importance becomes a useful interpretation tool. We show results using two different approaches, namely, the Mean Decreases in Accuracy (MDA) and Mean Decrease in Ginni. The former measures the model accuracy by random permutations of the variables on the Out-of-Bag (OOB) samples, while the latter measures the decrease in impurity. We observed that there is an overlap (>70%) in the features selected between the two procedures, as one would expect, however, the rank order is different. Mean Decrease in Accuracy simply means the proportion of samples that will be incorrectly classified if that feature was expunged from the model. On the other hand, the Mean Decrease in Ginni suggests the positive reduction in impurity associated with that feature mapping. Likewise, there is an overlap between feature importance between the RF models and XGBoost models (data not shown). Interestingly, mapping the distribution of variables of trees in the forest and tree depth reveal that highly ranked variables are close to the root, while not ruling out the possibility of they being reused, several times in the multiple tree splits.

Breiman and Cutler, who pioneered the statistical underpinnings of random forest agree the Gini decrease, is often consistent with the permutation method (Breiman and Cutler, 2001). It is generally believed that the Gini decrease is biased in favor of features with multiple splits. In a recent paper, Nembrini and colleagues proposed a method that debiases the impurity-based variable importance measures, which is fast, flexible, and

robust with minimal computational overheads (Nembrini et al., 2018). Co-dependent features tend to share importance, at least when permutation importance is applied to RF models. Strobl and colleagues (Strobl et al., 2008) proposed *conditional permutation importance* using the feature space created by node splitting during tree construction. Parr and colleagues argue, that most practitioners conflate feature importance measures with feature impact, an isolated effect of an explanatory variable on the response variable. In their paper, they give a mathematical definition of feature impact and importance derived from partial dependence curves, that operate directly on the data (Parr et al., 2020; Parr and Wilson, 2020). Model-agnostic versions of feature importance are a bold new proposal, that can be applied to an entire class of prediction models simultaneously (Fisher et al., 2019). Unlike RF models, generalized linear models (GLMs) rely on model parameters i.e., model coefficients, which are poor proxies for feature importance.

On a related note, it is a perfectly reasonable argument to make that producing stable and immutable motifs of feature sets, can be used effectively as *signatures* for application in diagnosis of disease or biomarker discovery in drug development. Sparse hyperplane classifiers were suggested as a statistical methodology for binary classification and relevant feature identification in a single pass, compared to the prevailing multistep filter-wrapper strategy, to extract *signatures* in transcript profiling studies in biomarker identification and toxicogenomics (Bhattacharyya et al., 2004; Natsoulis, 2005).

The random forest has been often claimed to uncover variable interaction effects. However, if and when such interactions can be differentiated from marginal effects remains unclear (Wright et al., 2016). Although we detect interactions in our experiments, it is unclear if these are direct interactions or merely marginal effects. Besides, it is limited to pair-wise interaction. Basu and colleagues building on random forest (RFs) and random intersection trees (RITs) have developed the iterative random forest algorithm (iRF) to detect stable, higher-order interactions with the same order of computational cost as RF (Basu et al., 2018).

Finally, we performed benchmarking experiments with decision trees i.e., classification and regression trees (CART) with multi-tree ensemble models. Further, we extended the benchmarking experiments to include other algorithms, namely, support vector machines (SVMs), naïve Bayes, artificial neural nets (ANN) for a side-by-side comparison. Overall, we conclude that the random forest delivered the lowest misclassification error. In other words, when rank-ordered, random forest models were adjudicated to the higher end and Naïve Bayes occupied the lower end of the stack.

Chapter 5: Conclusion

"Do not follow the path may lead. Go instead where there is no path and leave a trail." - Ralph Waldo Emerson

5.1 Algorithms for Modeling in Drug Discovery

This study is focused on building robust predictive models for identifying drug candidates for a select subset of validated protein targets in oncogenic pathways using pooled high-throughput bioassay datasets. For each of these compound collections, we extracted feature sets by computing the chemical descriptors and chemical fingerprints. Here we demonstrate the success of building such predictive models with different machine learning algorithms and comparing their model performance. The algorithms range from logistic regression, support vector machines, naïve Bayes, to the more sophisticated Artificial Neural Nets (ANN), single-tree classification and regression tree, and multi-tree ensemble models that include random forest (RF) and XGBoost.

We systematically built the entire modeling work-flow pipeline, starting with the compound acquisition, through multiple steps of preprocessing, developing *base* models, cross-validation, model regularization, hypertuning models, and final validation. We find that normalization and class balance leads to building more stable models. We find that feature that engineering and colinearity diagnostics help in reducing model redundancy. We find that model regularization limits the risk of *overfitting* and allows greater generalizability. We find model hypertuning to be one of the critical steps in building predictive models. Finally, we benchmarked each of these machine learning algorithms and ranked their performance on the metric - their mean misclassification error. The random forest models produced the lowest mean misclassification error. This study shows the feasibility of developing prototype predictive models for large-scale deployment in the production environment.

5.2 Limitations

While this study provides sufficient evidence of the power of *in-silico* predictive models in profiling compound libraries in predicting their biological activity for the selection and nomination of potential drug candidates, with the implied goal of bypassing expensive and often time-consuming experimental steps involved in lead identification and lead optimization in drug discovery. However, caution must be warranted about the limitations of this study. Firstly, the incredible vastness of the chemical space imposes challenges for screening

druggable compounds. Theoretical studies have estimated the organic chemical space to lie anywhere between 10^{29} to 10^{63} molecules (Bohacek et al., 1996). Even though every compound library is enriched with a rich structural diversity of chemical scaffolds, it is unlikely to meet the requirements of the demands of the vastness of chemical space of every molecular entity in terms of atoms, bonds, bond angles, rotatable bonds, ring structures, functional groups, and stereochemistry considerations.

However, it is believed that the complementarity requirements of the molecular geometry of the protein *active site* and the cognate ligand interaction will set theoretical limits on the chemical space, which will make it a lot more amenable to modeling. Secondly, the bulk of the work was done with two-dimensional (2D) chemical descriptors and Extended Connectivity FingerPrint (ECFP), this precludes us from accounting for the 3D structural patterns and the range of conformations a molecule may adopt in solution or during protein binding. Thirdly, even if a chemical library is structurally diverse, the constraints imposed by the small number of validated synthetic chemical reactions available for chemical synthesis will be rate-limiting to produce sufficient yields for scaling (Walters, 2019).

5.3 Future Work

Statistical and machine learning approaches to predict drug-to-target relationships would be vastly improved with the development, incorporation, and adoption of small-molecule 3D information through better molecular representation for the application of predictive modeling. Some of the early efforts in this direction are the ideas around Extended Three-Dimensional FingerPrint (E3FP) which is expected to encode the 3D representation of molecular conformations, which would be a logical extension of Extended Connectivity FingerPrint (ECFP) which captures the small-molecule topology patterns (Axen et al., 2017). Ash and Fourches (Ash and Fourches, 2017; Fourches and Ash, 2019) approached the problem of extracting 3D information via molecular dynamics. For instance, using chemical descriptors computed from molecular dynamics (MD) trajectories. The promise and prospects of next-generation ML-driven MD-QSAR predictive modeling for drug discovery look particularly enticing. Besides, advances in algorithmic techniques and exponential computing, hardware enhancements are accelerating the development of AutoML which promises to deliver the end-to-end automated ML models that can be incorporated into drug discovery work-flow with minimal human intervention (Dixon et al., 2016).

5.4 Future Vision

On a broader scale, the prospects of application of machine learning and artificial intelligence across multiple domains of pharmaceutical drug discovery are almost limitless, ranging from computational chemistry,

molecular modeling (MD) simulations, phenotypic screening, digital pathology, proteomics profiling, digital radiology, connected lab, connected health, digital clinical trials, predictive maintenance in drug manufacturing, and drug supply chain logistics.

In conclusion, it must be emphasized that it is imperative to examine problems and seek solutions through *systems thinking* framework to overcome the burden of cognitive dissonance in problem-solving, especially in dealing with complex biological systems and sub-system interactions – in understanding the etiology, progression, and burden of human diseases, to offer meaningful therapeutic interventions. We make no pretense to offer all the solutions, we are merely raising awareness.

References

Aderem, A., Smith, K.D., 2004. A systems approach to dissecting immunity and inflammation. Seminars in Immunology, Toll Receptor Families Structure and Function 16, 55–67. https://doi.org/10.1016/j.smim.2003.10.002

Akutsu, T., Nagamochi, H., 2013. COMPARISON AND ENUMERATION OF CHEMICAL GRAPHS. Computational and Structural Biotechnology Journal 5, e201302004. https://doi.org/10.5936/csbj.201302004

Alex Smola and S.V.N. Vishwanathan, 2008. Introduction to Machine Learning. Cambridge Press.

Alm, E., Arkin, A., 2003. Biological networks. Current Opinion in Structural Biology 13, 193–202. https://doi.org/10.1016/S0959-440X(03)00031-9

Arnold, R.D., Wade, J.P., 2015. A Definition of Systems Thinking: A Systems Approach. Procedia Computer Science 44, 669–678. https://doi.org/10.1016/j.procs.2015.03.050

Ash, J., Fourches, D., 2017. Characterizing the chemical space of ERK2 kinase inhibitors using descriptors computed from molecular dynamics trajectories. Journal of chemical information and modeling 57, 1286–1299.

Axen, S.D., Huang, X.-P., Cáceres, E.L., Gendelev, L., Roth, B.L., Keiser, M.J., 2017. A simple representation of three-dimensional molecular structure. J Med Chem 60, 7393–7409. https://doi.org/10.1021/acs.jmedchem.7b00696

Baker, M., 2013. Big biology: The 'omes puzzle. Nature News 494, 416. https://doi.org/10.1038/494416a

Baker, N.A., Sept, D., Joseph, S., Holst, M.J., McCammon, J.A., 2001. Electrostatics of nanosystems: application to microtubules and the ribosome. Proc. Natl. Acad. Sci. U.S.A. 98, 10037–10041. https://doi.org/10.1073/pnas.181342398

Balaban, A.T., 1985. Applications of graph theory in chemistry. Journal of Chemical Information and Modeling 25, 334–343. https://doi.org/10.1021/ci00047a033

Basu, S., Kumbier, K., Brown, J.B., Yu, B., 2018. Iterative random forests to discover predictive and stable high-order interactions. Proceedings of the National Academy of Sciences 115, 1943–1948. https://doi.org/10.1073/pnas.1711236115

Bhattacharyya, C., Grate, L.R., Jordan, M.I., Ghaoui, L.E., Mian, I.S., 2004. Robust sparse hyperplane classifiers: application to uncertain molecular profiling data. Journal of Computational Biology 11, 1073–1089.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., Jones, Z.M., 2016. mlr: Machine Learning in R. Journal of Machine Learning Research 17, 1–5.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning, Information science and statistics. Springer, New York.

Bohacek, R.S., McMartin, C., Guida, W.C., 1996. The art and practice of structure-based drug design: a molecular modeling perspective. Med Res Rev 16, 3–50. https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6

Bonchev, D., Rouvray, D.H., 1991. Chemical Graph Theory: Introduction and Fundamentals. CRC Press.

Boran, A.D., Iyengar, R., 2010. Systems approaches to polypharmacology and drug discovery. Curr Opin Drug Discov Devel 13, 297–309.

Borkin, D., Némethová, A., Michaľčonok, G., Maiorov, K., 2019. Impact of Data Normalization on Classification Model Accuracy. Sciendo 27, 79–84. https://doi.org/10.2478/rput-2019-0029

Bottou, L., 2012. Stochastic Gradient Descent Tricks, in: Montavon, G., Orr, G.B., Müller, K.-R. (Eds.), Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 421–436. https://doi.org/10.1007/978-3-642-35289-8_25

Box, G.E.P., Cox, D.R., 1964. An Analysis of Transformations. Journal of the Royal Statistical Society. Series B (Methodological), 26, 211–252.

Breiman, L., 2001. Randon Forests. Machine Learning 45, 5–32.

Breiman, L., 1996a. Bagging predictors. Machine Learning 24, 123–140. https://doi.org/10.1007/BF00058655

Breiman, L., 1996b. Bias, Variance, and Arcing Classifiers.

Breiman, L., Cutler, A., 2001. Random forests - classification description [WWW Document]. URL https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#features (accessed 9.24.20).

Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1993. Classification and Regression Trees. Chapman & Hall/CRC, New York, NY.

Brownlee, J., 2018. A Gentle Introduction to Sparse Matrices for Machine Learning [WWW Document]. Machine Learning Mastery. URL https://machinelearningmastery.com/sparse-matrices-for-machine-learning/ (accessed 10.22.20).

Bruce, P., Bruce, A., 2017. Practical Statistics for Data Scientists. O'Reilly Media, Inc., Sebastopol, CA 95472.

Burges, C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, 121–161.

Cano, G., Garcia-Rodriguez, J., Garcia-Garcia, A., Perez-Sanchez, H., Benediktsson, J.A., Thapa, A., Barr, A., 2017. Automatic selection of molecular descriptors using random forest: Application to drug discovery. Expert Systems with Applications 72, 151–159. https://doi.org/10.1016/j.eswa.2016.12.008

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 1–27. https://doi.org/10.1145/1961189.1961199

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 321–357. https://doi.org/10.1613/jair.953

Chemical Fingerprints - ChemAxon Documentation [WWW Document], n.d. URL https://docs.chemaxon.com/display/docs/Chemical_Fingerprints.html (accessed 5.12.20).

Chen, H., Kogej, T., Engkvist, O., 2018. Cheminformatics in Drug Discovery, an Industrial Perspective. Molecular Informatics 37, 1800041. https://doi.org/10.1002/minf.201800041

Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. ACM, pp. 785–794. https://doi.org/10.1145/2939672.2939785

Chevillard, F., Kolb, P., 2015. SCUBIDOO: A Large yet Screenable and Easily Searchable Database of Computationally Created Chemical Compounds Optimized toward High Likelihood of Synthetic Tractability. J Chem Inf Model 55, 1824–1835. https://doi.org/10.1021/acs.jcim.5b00203

Choi, D., Shallue, C.J., Nado, Z., Lee, J., Maddison, C.J., Dahl, G.E., 2020. On Empirical Comparisons of Optimizers for Deep Learning. arXiv:1910.05446 [cs, stat].

Chollet, F., 2015. Keras [WWW Document]. URL https://github.com/fchollet/keras

Christie, B.D., Leland, B.A., Nourse, J.G., 1993. Structure searching in chemical databases by direct lookup methods. Journal of Chemical Information and Modeling 33, 545–547. https://doi.org/10.1021/ci00014a004

Cook, D.J., Holder, L.B., 2007. Mining Graph Data. John Wiley & Sons, Inc.

Copenhaver, M.S., 2018. Sparsity and robustness in modern statistical estimation. MIT, Cambridge, US.

Cortez, P., 2020. R Package - rminer [WWW Document]. URL https://cran.r-project.org/web/packages/rminer/rminer.pdf (accessed 10.15.20).

Cramer, J.S., 2002. The origins of Logistic Regression (Timbergen Institute Discussion Paper). University of Amsterdam and Tinbergen Institute, Amsterdam.

Csáji, B.C., 2001. Approximation with Artificial Neural Networks. Faculty of Sciences Eötvös Loránd University Hungary.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signal Systems 2, 303–314.

Dalke, A., 2019. The chemfp project. Journal of Cheminformatics 11, 76. https://doi.org/10.1186/s13321-019-0398-8

Daylight Theory: Fingerprints [WWW Document], n.d. URL https://www.daylight.com/dayhtml/doc/theory/theory.finger.html (accessed 5.12.20).

Dehmer, M., Emmert-Streib, F., Tripathi, S., 2013. Large-Scale Evaluation of Molecular Descriptors by Means of Clustering. PLoS ONE 8, e83956. https://doi.org/10.1371/journal.pone.0083956

Deisenroth, M.P., Faisal, A.A., Ong, C.S., 2020. Mathematics for Machine Learning, 1st ed. Cambridge University Press. https://doi.org/10.1017/9781108679930

Dixon, S.L., Duan, J., Smith, E., Von Bargen, C.D., Sherman, W., Repasky, M.P., 2016. AutoQSAR: an automated machine learning tool for best-practice quantitative structure–activity relationship modeling. Future Medicinal Chemistry 8, 1825–1839. https://doi.org/10.4155/fmc-2016-0093

Dogo, E.M., Afolabi, O.J., Nwulu, N.I., Twala, B., Aigbavboa, C.O., 2018. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. IEEE, pp. 92–99. https://doi.org/10.1109/CTEMS.2018.8769211

Durant, J.L., Leland, B.A., Henry, D.R., Nourse, J.G., 2002. Reoptimization of MDL Keys for Use in Drug Discovery. Journal of Chemical Information and Computer Sciences 42, 1273–1280. https://doi.org/10.1021/ci010132r

Eklund, M., Norinder, U., Boyer, S., Carlsson, L., 2014. Choosing Feature Selection and Learning Algorithms in QSAR. Journal of Chemical Information and Modeling 54, 837–843. https://doi.org/10.1021/ci400573c

Elangovan, M., Sugumaran, V., Ramachandran, K.I., Ravikumar, S., 2011. Effect of SVM kernel functions on classification of vibration signals of a single point cutting tool. Expert Systems with Applications 38, 15202–15207. https://doi.org/10.1016/j.eswa.2011.05.081

Ernesto, E., Bonchev, D., 2014. Chemical Graph Theory, in: Grossman, J.L., Yellen, J., Zhang, P. (Eds.), Handbook of Graph Theory. CRC Press, Boca Raton, FL, pp. 1538–1558.

Fadel, S., Ghoniemy, S., Abdallah, M., Abu, H., 2016. Investigating the Effect of Different Kernel Functions on the Performance of SVM for Recognizing Arabic Characters. International Journal of Advanced Computer Science and Applications 7. https://doi.org/10.14569/IJACSA.2016.070160

Falbel, D., Allaire, J., Chollet, F., Tang, Y., Van Der Bijl, W., Studer, M., Keydana, S., 2020. R Interface to "Keras" [WWW Document]. URL https://keras.rstudio.com

Fechner, N., Hinsellman, G., Wegner, J.K., 2010. Molecular Descriptors, in: Faulon, J.-L., Bender, A. (Eds.), Handbook of Cheminformatics Algorithms. CRC Press Chapman & Hall Book, pp. 89–144.

Fisher, A., Rudin, C., Dominici, F., 2019. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. Journal of Machine Learning Research 20, 1–81.

Fourches, D., Ash, J., 2019. 4D- quantitative structure–activity relationship modeling: making a comeback. Expert Opinion on Drug Discovery 14, 1227–1235. https://doi.org/10.1080/17460441.2019.1664467

Freund, Y., Schapire, R.E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences 55, 119–139. https://doi.org/10.1006/jcss.1997.1504

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software 33. https://doi.org/10.18637/jss.v033.i01

Friedman, J.H., 2001. GREEDY FUNCTION APPROXIMATION: A GRADIENT BOOSTING MACHINE. The Annals of Statistics 29, 1189–1232.

Friedman, J.H., Bogdan, P., 2008. Predictive learning via rule ensembles. The Annals of Applied Statistics.

Fröhlich, H., Wegner, J.K., Sieker, F., Zell, A., 2006. Kernel Functions for Attributed Molecular Graphs – A New Similarity Based Approach To ADME Prediction in Classification and Regression. QSAR & Combinatorial Science 25, 317–326.

Gaulton, A., Hersey, A., Nowotka, M., Bento, A.P., Chambers, J., Mendez, D., Mutowo, P., Atkinson, F., Bellis, L.J., Cibrián-Uhalte, E., Davies, M., Dedman, N., Karlsson, A., Magariños, M.P., Overington, J.P., Papadatos, G., Smit, I., Leach, A.R., 2017. The ChEMBL database in 2017. Nucleic Acids Research 45, D945–D954. https://doi.org/10.1093/nar/gkw1074

Geysen, H.M., Meloen, R.H., Barteling, S.J., 1984. Use of peptide synthesis to probe viral antigens for epitopes to a resolution of a single amino acid. Proc. Natl. Acad. Sci. U.S.A. 81, 3998–4002. https://doi.org/10.1073/pnas.81.13.3998

Glass, L., Mackey, M., 1988. From Clocks to Chaos: The Rhythms of Life.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, Cambridge, US.

Guha, R., Willighagen, E., 2012. A Survey of Quantitative Descriptions of Molecular Structure. Curr Top Med Chem 12, 1946–1956.

Hahsler, M., 2017. R Code for Comparing Decision Boundaries of Different Classifiers [WWW Document]. URL https://michael.hahsler.net/SMU/EMIS7332/R/viz_classifier.html (accessed 9.9.20).

Han, H., Guo, X., Yu, H., 2016. Variable selection using Mean Decrease Accuracy and Mean Decrease Gini based on Random Forest, in: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS). Presented at the 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 219–224. https://doi.org/10.1109/ICSESS.2016.7883053

Hansen, L.K., Salamon, P., 1990. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 993–1001. https://doi.org/10.1109/34.58871

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Seconf Edition. ed, Springer Series in Statistics. Springer New York.

Hastie, T., Tibshirani, R., Friedman, J., n.d. The Elements of Statistical Learning - Data mining, Inference and Prediction 764.

Hinselmann, G., Rosenbaum, L., Jahn, A., Fechner, N., Zell, A., 2011. jCompoundMapper: An open source Java library and command-line tool for chemical fingerprints. Journal of Cheminformatics 3, 3. https://doi.org/10.1186/1758-2946-3-3

Hinton, G.E., Osindero, S., Teh, Y.-W., 2006. A Fast Learning Algorithm for Deep Belief Nets. Neural Computation 18, 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580 [cs].

Hoerl, A.E., Kennard, R.W., 1970. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics 12, 55. https://doi.org/10.2307/1267351

Hoffman, P., Grinstein, G., Marx, K.A., Grosse, I., Stanley, E., 1997. DNA Visual and Analytic Data Mining. Proceeding of IEEE Visualization 437–442.

Hoffman, P., Grinstein, G., Pinkney, D., 1999. Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations, in: New Paradigms in Information Visualization (NPIV). ACM Press, pp. 9–16. https://doi.org/10.1145/331770.331775

Honaker, J., King, G., Blackwell, M., 2011. Amelia II}: A Program for Missing Data. Journal of Statistical Software 1–47.

Hooker, G., 2004. Discovering additive structure in black box functions. ACM Press, p. 575. https://doi.org/10.1145/1014052.1014122

Hornik, K., 1991. Approximation Capabilities of Muitilayer Feedforward Networks. Neural Networks 4, 251–257.

Hosoya, H., 1988. On some counting polynomials in chemistry. Discrete Applied Mathematics 19, 239–257. https://doi.org/10.1016/0166-218X(88)90017-0

Houghten, R.A., 1985. General method for the rapid solid-phase synthesis of large numbers of peptides: specificity of antigen-antibody interaction at the level of individual amino acids. Proc Natl Acad Sci U S A 82, 5131–5135.

Ideker, T., Galitski, T., Hood, L., 2001. A NEW APPROACH TO DECODING LIFE: Systems Biology. Annual Review of Genomics and Human Genetics 2, 343–372. https://doi.org/10.1146/annurev.genom.2.1.343

Ilic, A., Ilic, M., 2017. On Some Algorithms for Computing Topological Indices of Chemical Graphs. MATCH Commun. Math. Comput. Chem. 78, 665–674.

Jeliazkova, N., Jeliazkov, V., 2011. AMBIT RESTful web services: an implementation of the OpenTox application programming interface. Journal of Cheminformatics 3. https://doi.org/10.1186/1758-2946-3-18

Junge, M.R.J., Dettori, J.R., 2018. ROC Solid: Receiver Operator Characteristic (ROC) Curves as a Foundation for Better Diagnostic Tests. Global Spine Journal 8, 424–429. https://doi.org/10.1177/2192568218778294

Khalifeh, M.H., Yousefi-Azari, H., Ashrafi, A.R., 2009. The first and second Zagreb indices of some graph operations. Discrete Applied Mathematics 157, 804–811. https://doi.org/10.1016/j.dam.2008.06.015

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B.A., Thiessen, P.A., Yu, B., Zaslavsky, L., Zhang, J., Bolton, E.E., 2019. PubChem 2019 update: improved access to chemical data. Nucleic Acids Research 47, D1102–D1109. https://doi.org/10.1093/nar/gky1033

Kingma, D.P., Ba, J., 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs].

Klein, R., Lindell, S.D., 2013. Combinatorial Chemistry Library Design, in: Audenaert, D., Overvoorde, P. (Eds.), Plant Chemical Biology. John Wiley & Sons, Inc, Hoboken, NJ, pp. 40–63. https://doi.org/10.1002/9781118742921.ch2.2

Kode - Chemoinformatics [WWW Document], n.d. URL https://chm.kode-solutions.net/ (accessed 12.30.20).

Koenker, R., Ng, P., 2003. **SparseM**: A Sparse Matrix Package for *R*. Journal of Statistical Software 8. https://doi.org/10.18637/jss.v008.i06

Kohavi, R., 1995. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence 2, 1137-1145.

Kohavi, R., Wolpert, D.H., 1996. Bias Plus Variance Decomposition for Zero-One Loss Function 9.

Kowalczyk, A., 2017. Support Vector Machines - Succintly. Syncfusion, Morrisville, NC, US.

Kuhn, M., Johnson, K., 2013. Applied Predictive Modeling. Springer New York, New York, NY. https://doi.org/10.1007/978-1-4614-6849-3

Kuhn, M., Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C., 2016. Caret: Classification and Regression Training. R package version 6.0-71. [WWW Document]. URL https://CRAN.R-project.org/package=caret (accessed 7.16.19).

Kumar, V., Krishna, S., Siddiqi, M.I., 2015. Virtual screening strategies: Recent advances in the identification and design of anti-cancer agents. Methods, Virtual Screening 71, 64–70. https://doi.org/10.1016/j.ymeth.2014.08.010

Landrum, G., 2012. Fingerprints in the RDKit 23.

Lawerence, N., 2015. Probabilistic Classification: Naive Bayes [WWW Document]. Neil Lawrence's Talks. URL http://inverseprobability.com/talks/notes/naive-bayes.html (accessed 9.7.20).

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. https://doi.org/10.1038/nature14539

Li, Z., Ma, X., Xin, H., 2017. Feature engineering of machine-learning chemisorption models for catalyst design. Catalysis Today 280, 232–238. https://doi.org/10.1016/j.cattod.2016.04.013

Liaw, A., Wiener, A., 2018. Package'randomForest' [WWW Document]. URL https://cran.r-project.org/web/packages/randomForest/index.html

Lin, H.-T., Lin, C.-J., 2003. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods. Neural Computation 3, 1–32.

Little, R.J.A., Rubin, D.B., 2020. Statistical Analysis with Missing Data, 3rd Edition. ed. John Wiley & Sons, Inc.

Liu, R., Li, X., Lam, K.S., 2017. Combinatorial Chemistry in Drug Discovery. Curr Opin Chem Biol 38, 117–126. https://doi.org/10.1016/j.cbpa.2017.03.017

Lo, Y.-C., Rensi, S.E., Torng, W., Altman, R.B., 2018. Machine learning in chemoinformatics and drug discovery. Drug Discovery Today 23, 1538–1546. https://doi.org/10.1016/j.drudis.2018.05.010

Loshchilov, I., Hutter, F., 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv:1608.03983 [cs, math].

Ma, J., Sheridan, R.P., Liaw, A., Dahl, G.E., Svetnik, V., 2015. Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships. Journal of Chemical Information and Modeling 55, 263–274. https://doi.org/10.1021/ci500747n

Maggiora, G.M., 2006. On Outliers and Activity CliffsWhy QSAR Often Disappoints. Journal of Chemical Information and Modeling 46, 1535–1535. https://doi.org/10.1021/ci060117s

Majka, {Michal, 2020. naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R, R package version 0.9.7 [WWW Document]. URL https://CRAN.R-project.org/package=naivebayes

Marsland, S., 2014. MACHINE LEARNING An Algorithmic Perspective, 2nd ed. Chapman & Hall/CRC.

Marx, K.A., O'Neil, P., Hoffman, P., Ujwal, M., 2003. Data Mining the NCI Cancer Cell Line Compound GI50 Values: Identifying Quinone Subtypes Effective Against Melanoma and Leukemia Cell Classes 16.

Mauri, A., Consonni, V., Pavan, M., Todeschini, R., 2006. DRAGON SOFTWARE: AN EASY APPROACH TO MOLECULAR DESCRIPTOR CALCULATIONS. MATCH Commun. Math. Comput. Chem. 56, 237–248.

McCarthy, J., Marx, K.A., Hoffman, P., Gee, A., G., O'Neil, P., Ujwal, M., Hotchkiss, J., 2004. Applications of machine learning and high-dimensional visualization in cancer detection, diagnosis, and management. Ann N Y Acad Sci. 1020, 239–262.

McCullagh, P., Nelder, J.A., 1983. Generalized Linear Models. Monographs on Statistics and Applied Probability. Chapman & Hall, London.

McGregor, M.J., Pallai, P.V., 1997. Clustering of Large Databases of Compounds: Using the MDL "Keys" as Structural Descriptors. J. Chem. Inf. Comput. Sci. 37, 443–448. https://doi.org/10.1021/ci960151e

MDL Keyset Technology [WWW Document], n.d. URL https://www.3dsbiovia.com/products/pdf/keys-to-keyset-technology.pdf (accessed 5.13.20).

Meyer, D., 2017. Support Vector Machines - The Interface to libsvm in package e1071 [WWW Document].

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2019. R package - e1071 [WWW Document]. URL https://CRAN.R-project.org/package=e1071

Mihalić, Z., Trinajstić, N., 1992. A graph-theoretical approach to structure-property relationships. Journal of Chemical Education 69, 701. https://doi.org/10.1021/ed069p701

Mitchell, K.J., 2012. What is complex about complex disorders? Genome Biol 13, 237. https://doi.org/10.1186/gb-2012-13-1-237

Mitchell, T.M., 1997. Machine Learning, McGraw-Hill series in computer science. McGraw-Hill, New York.

Morde, V., 2019. XGBoost Algorithm: Long May She Reign! [WWW Document]. Medium. URL https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (accessed 7.18.20).

Morgan, H.L., 1965. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. Journal of Chemical Documentation 5, 107–113. https://doi.org/10.1021/c160017a018

Mozafari, Z., Arab Chamjangali, M., Arashi, M., 2020. Combination of least absolute shrinkage and selection operator with Bayesian Regularization artificial neural network (LASSO-BR-ANN) for QSAR studies using functional group and molecular docking mixed descriptors. Chemometrics and Intelligent Laboratory Systems 200, 103998. https://doi.org/10.1016/j.chemolab.2020.103998

Nahida, S., Swathi, K., Sandeep, C.V., Kalyan, P.V.P., Harish, G.S., 2020. The Effect of SVM Kernel Functions on Heart Disease Dataset. International Journal of Advanced Science and Technology 29, 7.

Nakama, T., 2011. Comparisons of Single- and Multiple-Hidden-Layer Neural Networks, in: Liu, D., Zhang, H., Polycarpou, M., He, H. (Eds.), Lecture Notes in Computer Science. Springer, Berlin, Heidelberg.

Natsoulis, G., 2005. Classification of a large microarray data set: Algorithm comparison and analysis of drug signatures. Genome Research 15, 724–736. https://doi.org/10.1101/gr.2807605

Nayak, A., Stojmenovic, I. (Eds.), 2008. Handbook of Applied Algorithms_ Solving Scientific, Engineering, and Practical Problems. Wiley Interscience.

Nembrini, S., König, I.R., Wright, M.N., 2018. The revival of the Gini importance? Bioinformatics 34, 3711–3718. https://doi.org/10.1093/bioinformatics/bty373

Nesterov, Y., 2018. Lectures on Convex Optimization. Springer.

Ng, A.Y., 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. ACM Press, p. 78. https://doi.org/10.1145/1015330.1015435

O'Boyle, N.M., Hutchison, G.R., 2008. Cinfony – combining Open Source cheminformatics toolkits behind a common interface. Chemistry Central Journal 2. https://doi.org/10.1186/1752-153X-2-24

Oltvai, Z.N., Barabási, A.-L., 2002. Systems biology. Life's complexity pyramid. Science 298, 763–764. https://doi.org/10.1126/science.1078563

Open Babel [WWW Document], n.d. URL http://openbabel.org/wiki/ (accessed 5.16.20).

Oprea, T., Allu, T., Fara, D., Curpan, R., Halip, L., Bologa, C., 2007. Lead-like, Drug-like or "Pub-like": How different are they? Journal of computer-aided molecular design 21, 113–9. https://doi.org/10.1007/s10822-007-9105-3

Paluszynska, A., Biecek, P., Jiang, Y., 2020. Package - randomForestExplainer [WWW Document]. URL https://cran.r-project.org/web/packages/randomForestExplainer/randomForestExplainer.pdf (accessed 9.28.20).

Parr, T., Wilson, J.D., 2020. Technical Report: Partial Dependence through Stratification. arXiv:1907.06698 [cs, stat].

Parr, T., Wilson, J.D., Hamrick, J., 2020. Nonparametric Feature Impact and Importance. arXiv:2006.04750 [cs, stat].

Peters, A., Hothorn, T., Ripley, B.D., Therneau, T., Atkinson, B., 2019. R package - ipred [WWW Document]. URL https://cran.r-project.org/web/packages/ipred/ipred.pdf (accessed 10.15.20).

Phaisangittisagul, E., 2016. An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network. IEEE, pp. 174–179. https://doi.org/10.1109/ISMS.2016.14

Portugués, E.G., 2020. Notes for Predictive Modeling [WWW Document]. URL https://bookdown.org/egarpor/PM-UC3M/glm-deviance.html (accessed 8.11.20).

Prasoona, R.K., Jyoti, A., Mukesh, Y., Nishant, S., Anuraj, N.S., Shobha, J., 2013. Optimization of Gaussian Kernel Function in Support Vector Machine aided QSAR studies of C-aryl glucoside SGLT2 inhibitors. Interdisciplinary Sciences: Computational Life Sciences 5, 45–52. https://doi.org/10.1007/s12539-013-0156-y

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 2007. Numerical recipes: The art of scientific computing, 3rd ed. Cambridge University Press, New York, NY, US.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Quinlan, J.R., 1986. Induction of decision trees. Machine Learning 1, 81–106. https://doi.org/10.1007/BF00116251

Randić, M., 2001. Novel Shape Descriptors for Molecular Graphs. Journal of Chemical Information and Computer Sciences 41, 607–613. https://doi.org/10.1021/ci0001031

Raschka, S., 2019. Regularization [WWW Document]. URL https://sebastianraschka.com/pdf/lecture-notes/stat479ss19/L10_regularization_slides.pdf

Raschka, S., 2014a. About Feature Scaling and Normalization [WWW Document]. Dr. Sebastian Raschka. URL https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (accessed 9.10.20).

Raschka, S., 2014b. Naive Bayes and Text Classification Intro + Theory [WWW Document]. Dr. Sebastian Raschka. URL https://sebastianraschka.com/Articles/2014_naive_bayes_1.html (accessed 9.10.20).

Rensi, S.E., Altman, R.B., 2017. Shallow Representation Learning via Kernel PCA Improves QSAR Modelability. Journal of Chemical Information and Modeling 57, 1859–1867. https://doi.org/10.1021/acs.jcim.6b00694

Riahi, S., Pourbasheer, E., Ganjali, M.R., Norouzi, P., 2009. Support Vector Machine-Based Quantitative Structure-Activity Relationship Study of Cholesteryl Ester Transfer Protein Inhibitors. Chemical Biology & Drug Design 73, 558–571. https://doi.org/10.1111/j.1747-0285.2009.00800.x

Robnik-Sikonja, M., Savicky, P., 2020. R Package - CORElearn [WWW Document]. URL https://cran.r-project.org/web/packages/CORElearn/CORElearn.pdf (accessed 10.15.20).

Rogers, D., Hahn, M., 2010. Extended-Connectivity Fingerprints. Journal of Chemical Information and Modeling 50, 742–754. https://doi.org/10.1021/ci100050t

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65, 386–408. https://doi.org/10.1037/h0042519

Ruddigkeit, L., van Deursen, R., Blum, L.C., Reymond, J.-L., 2012. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. J. Chem. Inf. Model. 52, 2864–2875. https://doi.org/10.1021/ci300415d

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning Internal Representations by Error Propagation,. Nature 323, 533–536.

Schadt, E.E., Friend, S.H., Shaywitz, D.A., 2009. A network view of disease and compound screening. Nature Reviews Drug Discovery 8, 286–295. https://doi.org/10.1038/nrd2826

Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. The Annals of Statistics 26, 1651–1686. https://doi.org/10.1214/aos/1024691352

Schulz, K.-P., 1992. M. Johnson, G. Maggiora (Eds.): Concepts and Applications of Molecular Similarity, Wiley Interscience, New York, Chichester 1990. ISBN 0-471175-7, 393 Seiten. Preis: £ 51.35. Berichte der Bunsengesellschaft für physikalische Chemie 96, 1087–1087. https://doi.org/10.1002/bbpc.19920960825

Sejdinovic, D., 2015. Statistical Data Mining and Machine Learning [WWW Document]. URL http://www.stats.ox.ac.uk/~sejdinov/teaching/sdmml15/materials/HT15_lecture12-nup.pdf

Shannon, C.E., 1948. A Mathematical Theory of Communication. The Bell System Technical Journal 27, 379–423.

Shaw, D., Grossman, J.P., Bank, J., Batson, B., Butts, J., Chao, J., Deneroff, M., Dror, R., Even, A., Fenton, C., Forte, A., Gagliardo, J., Gill, G., Greskamp, B., Ho, C., Ierardi, D., Iserovich, L., Kuskin, J., Larson, R., Young, C., 2015. Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, 41–53. https://doi.org/10.1109/SC.2014.9

Sigston, E.A.W., Williams, B.R.G., 2017. An Emergence Framework of Carcinogenesis. Front Oncol 7. https://doi.org/10.3389/fonc.2017.00198

Singh, D., Singh, B., 2019. Investigating the impact of data normalization on classification performance. Applied Soft Computing 105524. https://doi.org/10.1016/j.asoc.2019.105524

Singh, P., Mhaka, A.M., Christensen, S.B., Gray, J.J., Denmeade, S.R., Isaacs, J.T., 2005. Applying linear interaction energy method for rational design of noncompetitive allosteric inhibitors of the sarco- and endoplasmic reticulum calcium-ATPase. J. Med. Chem. 48, 3005–3014. https://doi.org/10.1021/jm049319a

Smith, G.P., 1985. Filamentous fusion phage - novel expression vectors that display cloned antigens on the virion surface. Science 228, 1315–1317.

Smith, L.N., 2017. Cyclical Learning Rates for Training Neural Networks. arXiv:1506.01186 [cs].

Smith, R.E., Tran, K., Richards, M., 2015. Systems Thinking for the Medicinal Chemists. 25.

Spackman, K., 1989. Signal detection theory: Valuable tools for evaluating inductive learning. Proceedings of the Sixth International Workshop on Machine Learning 160–163.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting 1929–1958.

Steinbeck, C., Hoppe, C., Kuhn, S., Floris, M., Willighagen, R.G. and E.L., 2006. Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics. Current Pharmaceutical Design 12, 2111–2120.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., Zeileis, A., 2008. Conditional variable importance for random forests. BMC Bioinformatics 9, 307. https://doi.org/10.1186/1471-2105-9-307

Sun, J., Jeliazkova, N., Chupakin, V., Golib-Dzib, J.-F., Engkvist, O., Carlsson, L., Wegner, J., Ceulemans, H., Georgiev, I., Jeliazkov, V., Kochev, N., Ashby, T.J., Chen, H., 2017. ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics. Journal of Cheminformatics 9. https://doi.org/10.1186/s13321-017-0203-5

The PubChem Project [WWW Document], 2017. URL https://pubchem.ncbi.nlm.nih.gov/ (accessed 6.20.17).

Theodoridis, S., Koutroumbas, K., 2009. Pattern Recognition. Elsevier.

Therneau, T., Atkinson, B., port, B.R. (producer of the initial R., maintainer 1999-2017), 2019. rpart: Recursive Partitioning and Regression Trees [WWW Document]. URL https://CRAN.R-project.org/package=rpart (accessed 10.12.20).

Tibshirani, R., 1996. Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological) 58, 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Todeschini, R., Consonni, V., 2008. Handbook of molecular descriptors. John Wiley & Sons.

Trinajstic, N., 1992. Chemical Graph Theory. Routledge. https://doi.org/10.1201/9781315139111

Ujwal, M., Hoffman, P., Marx, K.A., 2007. A Machine Learning Approach to Pharmacological Profiling of the Quinone Scaffold in the NCI Database: A Compound Class Enriched in Those Effective Against Melanoma and Leukemia Cell Lines. IEEE BIBE Proceedings 1, 456–463.

van der Greef, J., McBurney, R.N., 2005. Rescuing drug discovery: in vivo systems pathology and systems pharmacology. Nature Reviews Drug Discovery 4, 961–967. https://doi.org/10.1038/nrd1904

Van Regenmortel, M.H.V., 2004. Reductionism and complexity in molecular biology. EMBO reports 5, 1016–1020. https://doi.org/10.1038/sj.embor.7400284

Vapnik, V., 1998. Statistical Learning Theory. Wiley Interscience.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, Berlin.

Vapnik, V., 1979. Estimation of Dependences Based on Empirical Data. Springer-Verlag New York Inc.

Walters, W.P., 2019. Virtual Chemical Libraries: Miniperspective. Journal of Medicinal Chemistry 62, 1116–1124. https://doi.org/10.1021/acs.jmedchem.8b01048

Ward, M., Grinstein, G., Keim, D., 2010. Interactive Data Visualization: Foundations, Techniques, and Applications. A K Peters/CRC Press.

Weininger, D., 2002. Combinatorics of Small Molecular Structures, in: von RaguéSchleyer, P. (Ed.), In Encyclopedia of Computational Chemistry; John Wiley & Sons, Ltd.

Weininger, D., Weininger, A., Weininger, J.L., 1989. SMILES. 2. Algorithm for generation of unique SMILES notation. Journal of Chemical Information and Modeling 29, 97–101. https://doi.org/10.1021/ci00062a008

Wickham, H., 2009. ggplot2: Elegant graphics for data analysis2009. Springer-Verlag New York.

Williams, G., Culp, M.V., Cox, E., Nolan, A., White, D., Medri, D., Forest), A.W. (OOB A. for R., Ripley (print.summary.nnet), B., plots), J.M. (ggpairs, RevoScaleR/XDF), S.T. (initial, RevoScaleR/XDF), D.P.C. (initial, RevoScaleR/XDF), D.M.V. (initial, RevoScaleR/XDF), M.C. (initial, xgboost), F.Z. (initial, chart), C.C. (risk plot on risk, 2020. rattle: Graphical User Interface for Data Science in R [WWW Document]. URL https://CRAN.R-project.org/package=rattle (accessed 10.15.20).

Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B., 2017. The Marginal Value of Adaptive Gradient Methods in Machine Learning, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 4148–4158.

Wilson, R.J., 1996. Introduction to graph theory, 4. ed.,. ed. Prentice Hall, Harlow.

Winkler, D.A., Le, T.C., 2017. Performance of Deep and Shallow Neural Networks, the Universal Approximation Theorem, Activity Cliffs, and QSAR. Molecular Informatics 36, 1600118. https://doi.org/10.1002/minf.201600118

Wright, M.N., Ziegler, A., König, I.R., 2016. Do little interactions get lost in dark random forests? BMC Bioinformatics 17. https://doi.org/10.1186/s12859-016-0995-8

Wu, J., 2020. Support Vector Machines (SVM) [WWW Document]. URL https://cs.nju.edu.cn/wujx/teaching/07_SVM.pdf

Yeo, I.-K., Johnson, Richard.A., 2000. A New Family of Power Transformations to Improve Normality or Symmetry. Biometrika 87, 954–959.

Zakharov, A.V., Peach, M.L., Sitzmann, M., Nicklaus, M.C., 2014. QSAR Modeling of Imbalanced High-Throughput Screening Data in PubChem. Journal of Chemical Information and Modeling 54, 705–712. https://doi.org/10.1021/ci400737s

Zaki, M.J., Meira, Jr, W., 2014. Data Mining and Analysis: Fundamental Concepts and Algorithms, 1st ed. Cambridge University Press, New York, NY. https://doi.org/10.1017/CBO9780511810114

Zhang, A., Lipton, Z.C., Li, M., Smola, A.J., 2020. Dive into Deep Learning.

Zhao, S., Iyengar, R., 2012. Systems Pharmacology: Network Analysis to Identify Multiscale Mechanisms of Drug Action. Annual Review of Pharmacology and Toxicology 52, 505–521. https://doi.org/10.1146/annurev-pharmtox-010611-134520

Zheng, A., Casari, A., 2018. Feature Engineering for Machine Learning. O'Reilly Media, Inc., Sebastopol, CA 95472.

Zhou, Z.-H., 2012. Ensemble Methods - Foundations and Algorithms. Chapman & Hall/CRC, Boca Raton, FL.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67, 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x