In-silico Methods for Drug Discovery: Applications of Molecular Dynamics, Drug Docking, and Machine Learning

by

Rajeev Jaundoo

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Biomedical Engineering

University of Alberta

Abstract

Drug discovery is a venture that is costly in both time and money. *In-silico* methods are a core part of biomedical research, from traditional tools such as drug docking and molecular dynamics to newer machine learning frameworks, all of which are more efficient in both time and cost compared to wholly experimental approaches. This thesis highlights 3 separate studies that reflect the past, present, and future of *in-silico* research, starting with the development of novel platelet activated ligands for the purpose of targeted drug delivery using traditional tools such as drug docking and simulated annealing. The second study demonstrated how machine learning was used for classification of drug activity (agonist, antagonist, or non-binder) towards a group of macromolecules all within the nuclear receptor family. Finally, the third study used machine learning in a regression task to predict bioelectric potentials and ion channel activity of a cellular network to serve as a replacement for another *in-silico* application, reducing the computational resources required for these predictions and providing the ability to scale to larger cell networks with ease. Development of newer, more advanced *in-silico* tools increases the accuracy of drug treatment predictions, leading to more effective therapeutics and lower rates of failure during pre-clinical as well as clinical phases. Not only that, but computational methods excel at drug repurposing, a process in which existing pharmaceuticals are used for indications not originally intended. Bioinformatic techniques and machine learning take advantage of the substantial

amount of biological, pharmacological, etc. data available to identify adverse and beneficial interactions that would otherwise be missed. Overall, *in-silico* techniques are performed in tandem with *in-vitro* and *in-vivo* experiments, used to validate computational predictions, during various phases of drug discovery and repurposing, making them a core part of biomedical research at large.

Preface

All *in-vitro* data from Chapter 1 was obtained from the CSTS Health Care company, including the FASTA sequences of all platelet activated ligands (PALs) outside of PAL1, which was obtained from Butterfield et al. (2019). All *in-silico* modeling was performed by myself in Chapter 1, and for all other chapters I performed all research and generated all results. The following is a list of publications related to this thesis.

Jaundoo, R., & Craddock, T. J. (2020). DRUGPATH: the drug gene pathway meta-database. *International Journal of Molecular Sciences, 21*(9), 3171.

Jaundoo, R., Bohmann, J., Gutierrez, G. E., Klimas, N., Broderick, G., & Craddock, T. J. (2020). Towards a Treatment for Gulf War Illness: A Consensus Docking Approach. *Military Medicine, 185*(Supplement_1), 554-561.

Jaundoo, R., Bohmann, J., Gutierrez, G. E., Klimas, N., Broderick, G., & Craddock, T. J. (2018). Using a consensus docking approach to predict adverse drug reactions in combination drug therapies for gulf war illness. *International Journal of Molecular Sciences, 19*(11), 3355.

Acknowledgments

First and foremost, I would like to express my gratitude towards both my supervisor, Professor Jack Tuszynski, as well as my co-supervisor, Professor Travis Craddock, for the mentoring and support throughout my studies. Additionally, I would like to thank Professor Russ Greiner for serving on my committee and lending his expertise in machine learning.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADP	Adenosine Diphosphate
APOL	Atomic Polarizabilities
AR	Androgen Receptor
ASG	2-Deoxy-2-Acetamido-Beta-D-Galactose-4-Sulfate
BAYRID	Bayesian Ridge
BEN	BioElectric Network
BETSE	Bioelectric Tissue Simulation Engine
CA^{2+}	Calcium Ion
CART	Classification and Regression Trees
CFL	Charge Free Ligand
CL^{-}	Chloride Ion
CSA	Chondroitin Sulfate A
CV	Cross-Validation
DECTRE	Decision Tree
DRUGPATH	Drug-Gene-Pathway
DUD-E	Database of Useful Decoys: Enhanced
ECFP	Extended-Connectivity Fingerprint
E_ELE	Electrostatic Potential Energy
ER	Estrogen Receptor
FCHARGE	Formal Charges
FN	False Negative
FP	False Positive
GAG	Glycosaminoglycan
GB/VI	Generalized Born Solvation Mode
GC4	4-Deoxy-beta-D-Glucopyranuronic Acid
GCU	Alpha-D-Glucopyranuronic Acid
GF	Growth Factor
GJ	Gap Junction
GR	Glucocorticoid Receptor
HS	Heparan Sulfate
ITC	Isothermal Titration Calorimetry
K^{+}	Potassium Ion
KNN	K-Nearest Neighbors
LBD	Ligand Binding Domain
LINREG	Linear Regression
LXR-BETA	Liver X Receptor Beta
MAE	Mean Absolute Error
MAPE	Mean Absolute Percent Error

MEDAE	Median Absolute Error
MD	Molecular Dynamics
ML	Machine Learning
MLP	Multi-Layer Perceptron
MOE	Molecular Operating Environment
MSE	Mean Square Error
MV	Millivolts
NA^+	Sodium Ion
NAIBAY	Naïve Bayes
NAMD	Nanoscale Molecular Dynamics
NEUNET	Neural Network
NMR	Nuclear Magnetic Resonance
PAL	Platelet Activated Ligand
PDB	Protein Data Bank
PF4	Platelet Factor 4
PR	Progesterone Receptor
QSAR	Quantitative Structure-Activity Relationship
R^2	Coefficient of Determination
RBF	Radial Basis Function
RANFOR	Random Forest
RMSD	Root Mean Square Distance
RMSE	Root Mean Square Error
ROC-AUC	Area Under the Receiver Operator Characteristics
SA	Simulated Annealing
SDF	Structure Data File
SMILES	Simplified Molecular-Input Line-Entry System
SUPLRN	Super Learner
SVM	Support Vector Machine
SVR	Support Vector Regression
TN	True Negative
TP	True Positive
TSV	Tab-Separated Value
USD	United States Dollars
VDW	van der Waals
V_{MEM}	Transmembrane Potential

Introduction

Computational, or *in-silico*, tools and methods have long been a part of biomedical research, starting as early as the 1960s with bioinformatic techniques being used for prediction of protein sequences and the explosion of popularity in the early 2000s of simulation-based approaches (Hagen, 2000; Noble, 2002). These methods tend to not only be faster and less costly compared to their *in-vivo* and *in-vitro* counterparts, but also allow for predictions and discoveries to be made by integrating the vast amount of existing biological and medical data available today (Durrant & McCammon, 2012; Ekins et al., 2007). The development of a novel pharmaceutical costs approximately $1-2 billion United States Dollars (USD) and takes anywhere from 10-15 years from start (target identification) to finish (clinical use), which involves target selection, compound screening, preclinical validation, and clinical trials (Chan et al., 2019; Sun et al., 2022). Preclinical validation includes *in-vivo* experiments involving animal models such as mice or rats that require maintenance including food and regular cleaning of the animals' living conditions, as well as the usage of anesthetics or tranquilizers during invasive procedures (Robinson et al., 2019). The time and rising costs of these animal models, estimated to be over $300,000 USD in 2019 (Van Norman, 2019), serve as another barrier for approval of a drug to market. For the pharmaceuticals that do reach clinical trials, costs continue to rise with the development of an

Alzheimer's drug for example costing $79 million USD for phase one, $141 million for phase 2, and $462 million for phases 3 and 4 (Cummings et al., 2021). Furthermore, there is an incredibly high failure rate for drug candidates; 90% of drugs that are able to make it to clinical trials fail during phases 1, 2, or 3, and this rate is even higher when candidates that fail preclinical stages are counted (Sun et al., 2022). The current drug development process can be summarized as being costly and time consuming considering the extremely high failure rate of drug candidates.

With this in mind, *in-silico* methods are playing an increasingly important role within drug design and discovery due to their efficiency in both cost and time, not to mention the ability to take advantage of scientific literature, health records, and the abundant amount of biomedical data available. For instance, the drug-gene-pathway (DRUGPATH) meta-database consolidates numerous databases such as the Food and Drug Administration's National Drug Code directory, ConsensusPathDB, and the Toxin and Toxin-Target Database among others in order to predict adverse drug interactions (Jaundoo & Craddock, 2020). *In-silico* systems biology approaches such as these are also useful for drug repurposing, where the off-target interactions of existing pharmaceuticals can be leveraged towards diseases or any application other than what was originally intended. Repurposing not only provides patients with novel treatments that may otherwise

not exist, but it additionally allows pharmaceutical companies to potentially save a therapeutic that has failed in one domain (Palve et al., 2021).

The overarching focus of this thesis was the use of various *in-silico* methods to perform drug discovery, specifically, utilizing techniques such as drug docking, molecular dynamics (MD), and machine learning (ML) to ultimately improve drug treatment predictions in diseases such as cancer. Developing more accurate computational tools and methods would not only lead to potentially more effective pharmaceuticals with less side effects, but it would also drastically reduce the costs of drug development because it would filter candidates destined to fail in later stages. The first chapter focuses on research performed using the more traditional MD and drug docking computational methods to model novel platelet activated ligands (PALs) for the purpose of targeted drug delivery. This involved building all structures, both receptor and ligands, in the Molecular Operating Environment (MOE) application, the use of MD to obtain a pose, or conformation, of the receptor likely to be found *in-vivo*, normalization of the predicted binding affinities of the docked ligands and experimental values obtained from the CSTS Health Care company (Toronto) for comparison, and evaluation of the *in-silico* results to determine how to best optimize PALs. The second chapter follows with the current state of the field transitioning to using ML as a supplement to these traditional methods, where ML models were trained to classify pharmaceuticals based on their chemical properties, allowing researchers to efficiently screen drug

libraries used during early-stage drug discovery for further drug docking and other types of computational modeling. Here, several different ML algorithms were trained in classifying pharmaceuticals that act as either activators, blockers, or non-binders for a set of receptors to determine the one best suited for the task. This entire process included data augmentation to boost the total number of pharmaceuticals within each class, using MOE to calculate hundreds of features for all pharmaceuticals, feature selection to discard irrelevant features, and training using the RapidMiner Studio application. Finally, the third chapter demonstrates how ML was utilized to predict properties from interactions that occur within cellular networks, replacing an existing *in-silico* application in the process. The Bioelectric Tissue Simulation Engine (BETSE) application (Pietak & Levin, 2016) was first used to simulate thousands of different cellular networks to generate a comprehensive amount of data regarding membrane potentials and ion channel flux. Next, ML was performed with a total of 8 different types of learners, including a meta-learner made up of 5 base learners. The trained ML models were then evaluated to determine the one(s) best suited for predicting ion channel concentrations as well as membrane potentials, providing a more efficient and scalable method compared to BETSE.

These 3 chapters provided an insight into both conventional and modern computational drug discovery, exhibiting their ability to integrate the existing knowledgebase of chemical, physical, and biomedical data to predict and model

various interactions (e.g., peptide-peptide) and other biological activity (e.g., ion channels). Not only that, but *in-silico* methods were used in collaboration with *in-vitro* experiments, allowing researchers in both domains to benefit.

Chapter 1: Using Platelet Activated Ligands for Targeted Drug Delivery

1.1 Introduction

Platelets are a special type of blood cell that act as first responders in tissue injury and play crucial roles in wound healing, cancer progression, and metastasis (Jurk & Kehrel, 2005; Schwarz et al., 2020). That being said, tumor cells also secrete platelet activators such as adenosine diphosphate (ADP) and thrombin in order to activate platelets (Palacios-Acedo et al., 2019). Once activated, platelets bind to these tumor cells and serve as a protective measure against shear stress and apoptosis by adhering to the site of action and creating a provisional stromal matrix both rich in sequestered growth factors (GFs) as well as angiogenesis-regulating proteins located in the platelets' alpha-granules, one of 3 types of secretory granules that contain proteins that facilitate the adhesive process among other functions (Italiano et al., 2008; Schwarz et al., 2020). This behavior of sequestering GFs and releasing them on tumor sites makes platelets the perfect carrier for anti-tumor therapeutics. Ideally, platelets would allow drugs to be delivered without any interaction with other organs, avoiding off-target interactions, better known as side effects.

Klement et al. (2009) had described previously that the sequestration of growth regulators in platelets within cancer conditions is a selective process and

occurs through the interaction of growth regulators with glycosaminoglycans (GAGs), which are sulphated, negatively charged polysaccharides located within the platelet alpha-granules (Zhang et al., 2020). Moreover, there are 2 main types of GAGs: heparan sulfate (HS) and chondroitin sulfate (L. Zhang, 2010). Binding and sequestration via GAG-binding domains preserves both the integrity and functions of the proteins, and additionally, prevents ligand activation and signaling (Italiano et al., 2008; Klement et al., 2009, 2015). This GAG-binding process inspired a strategy to develop PALs, which are peptides that mimic the GAG-binding domain and can be used to anchor a drug to a given GAG, ultimately leading to the sequestration of the drug within platelet alpha-granules without receptor binding. Platelet sequestration protects the drug from plasma proteases and degradation, extending the half-life to that of a platelet, approximately 4-7 days.

Off-target interactions have long plagued the field of drug discovery, leading to issues with efficacy and toxicity. Small molecule drugs are known to bind anywhere from 6-11 different targets minimum outside of their intended pharmacological target, and this number may be even larger due to the fact pharmaceutical companies only test a small subset of potential targets during preclinical trials (Rao et al., 2019). Consequently, when attempting to bring a drug candidate to market, one of the most common points of failure during testing is either high accumulation of the drug within off-target organs, or alternatively, poor

accumulation within the intended organs (Zhao et al., 2020). The usage of PALs for drug delivery would alleviate this issue, saving millions of dollars in costs as well as valuable research and development time.

In this chapter, peptides derived from a PAL FASTA sequence previously published by Butterfield et al. (2010) were built using the MOE application to model their interactions with the GAG chondroitin sulfate A (CSA). There was a total of 13 unique PALs modeled *in-silico*: PAL1 was the original high affinity peptide to CSA obtained from Butterfield et al. (2010), PAL2 was created by the CSTS Health Care company by scrambling the sequence of PAL1, and the 11 mutant peptides, known as PALs 1-11A, were created by CSTS Health Care through alanine mutagenesis. *In-vitro* experiments from CSTS Health Care confirmed the binding of both PAL1 and PAL2 to CSA, and furthermore, the conjugation of a large protein to these PALs did not prevent binding, leading to the possibility of anchoring a drug for the purpose of targeted drug delivery. The goal of this study was to establish an *in-silico* model that allows for the evaluation of the interactions between the PALs and CSA, which would allow for future optimization of these PALs and ultimately identify the one best suited for binding and transporting therapeutics. This would allow a given therapeutic to be attached to a PAL, which itself would bind to a GAG such as CSA, allowing it to be sequestered within platelet alpha-granules and then transported to a target site as in the case of tumor cell-induced platelet activation for example.

1.2 Methods

As an overview, MOE (version 2019.01) was used to generate the 3D structures of PAL1, PAL2, and the mutant PALs 1-11A from their FASTA sequences. Additionally, the FASTA sequence for a charge free ligand (CFL) was also obtained from CSTS Health Care, which served as a control during their *in-vitro* experiments as a weak binder to CSA. The Protein Data Bank (PDB) online database was used to obtain the template structure for CSA (PDB ID: 1C4S) as well as for the complete structure of platelet factor 4 (PF4; PDB ID: 1F9Q), a protein stored in platelet alpha-granules with a high natural affinity to GAGs such as chondroitin sulfate and HS (Kowalska et al., 2010). Once all 3D structures were built, an MD technique known as simulated annealing (SA) was used to determine the lowest energy pose, or conformation, of CSA assumed to be found *in-vivo* within the physiologically relevant tissues. Afterwards, PAL1, PAL2, PALs 1-11A, CFL, and PF4 were docked to this lowest energy conformation of CSA to identify their bound positions on the structure. Finally, the contribution of the individual forces involved in each of the bound ligands to CSA structures was evaluated.

Table 1.1: FASTA sequences of PAL1, PAL2, PALs 1-11A, and CFL.

Peptide	Amino Acid Sequence
PAL1	ERRIWFPYRRF
PAL2	RFRWPYRIREF
PAL1A	ARRIWFPYRRF
PAL2A	EARIWFPYRRF
PAL3A	ERAIWFPYRRF
PAL4A	ERRAWFPYRRF
PAL5A	ERRIAFPYRRF
PAL6A	ERRIWAPYRRF
PAL7A	ERRIWFAYRRF
PAL8A	ERRIWFPARRF
PAL9A	ERRIWFPYARF
PAL10A	ERRIWFPYRAF
PAL11A	ERRIWFPYRRA
CFL	EGGIWFPYGGF

1.2.1 Generating 3D Structures

The FASTA sequences for each of the PAL1, PAL2, PALs 1-11A, and CFL were imported into MOE, where their 3D structures were constructed using the Protein Builder tool that utilized chemical and physical properties of the amino acids to determine their *in-silico* structures. The *in-vivo* or *in-vitro* structures of the PALs and CFL were unknown as the short length of peptides such as these poses a challenge in obtaining stable structures experimentally (Aldas-Bulos & Plisson, 2023). CSA was built using the 1C4S structure as a starting point to generate a 34-unit alternating chain of 4-Deoxy-beta-D-glucopyranuronic acid (GC4; A), 2-Deoxy-2-Acetamido-beta-D-Galactose-4-Sulfate (ASG; B), alpha-D-Glucopyranuronic acid (GCU; C), and finally ASG again. 1C4S contained the 6

units GC4-ASG-GCU-ASG-GC4-ASG (A-B-C-B-A-B) and using MOE's Builder tool, this chain was continued until it reached a total of 34 units so it would reflect the *in-vitro* experimental conditions from CSTS Health Care; A-B-C-B-A-B-A-B-C-B-A-B-A-B-C-B-A-B-A-B-C-B-A-B-A-B-C-B-A-B-A-B-C-B-A-B-A-B-C-B. Once all structures were complete, they were then processed using the default settings of QuikPrep, a toolkit in MOE that performed energy minimization using a combination of the steepest descent, conjugate gradient, and truncated newton methods depending on whether the gradient was calculated as high, small, or reasonable respectively using the Amber ff10 force field. Moreover, QuikPrep added all hydrogens, polar and non-polar, and removed all waters. It is important to note that although CFL is neutral by design, it had to undergo this preparation and become charged, otherwise docking to CSA would have failed.

1.2.2 Simulated Annealing

MOE also served as an interface to the Nanoscale Molecular Dynamics (NAMD; version 2.13) application, which was used to perform all MD simulations. MD is a technique in which Newtonian physics is used to calculate simple approximations of atomic movement, which includes the forces originating from interactions between bonded and non-bonded atoms (Durrant & McCammon, 2011). Both chemical bonds and atomic angles are simulated via virtual springs, while non-bonded forces emerge from van der Waals (vdW) and electrostatic interactions,

modeled using the Lennard-Jones potential and Coulomb's law, respectively (Durrant & McCammon, 2011).

Due to the fact that CSA was built starting from a template structure, a technique known as SA was employed to remedy any optimization problems. This is a local search algorithm that increases the chance a structure will overcome conformations corresponding to the local minima on the potential energy surface, and instead obtain the global minima conformation (Gendreau & Potvin, 2010). SA involves first heating up a structure within a high temperature environment and then performing all subsequent simulations in gradually cooler environments to determine the structure's final conformation (Hatmal & Taha, 2017). SA was performed in this chapter by running 4 consecutive simulations, each with distinct temperature phases: I) 295.15K, II) 310.00K, III) 400.00K, and IV) 295.15K. Except for temperature, the parameters of each simulation were kept the same; see Table 1.2 for details. The default Amber ff10 force field was used, the pH was set to 7.35 to keep consistent with the previous *in-vitro* experiments of CSA from CSTS Health Care, and the structure was simulated within water. The temperature within each phase was also kept consistent throughout the entire simulation, meaning there were no heating up or cooling down periods, and no pressure was applied within any of the simulations.

Table 1.2: Parameters used during all phases of SA.

Parameter	Value
Timestep	2.0
pH	7.35
Steps per Cycle	25

1.2.3 Drug Docking

Drug docking is a technique that simulates the binding of a ligand, a drug for example, to a receptor such as a protein or other macromolecule. Docking involves 2 processes: sampling, where the conformational space, the possible 3D arrangement of atoms, of the ligand are explored so that the second process, scoring, can calculate the estimated binding affinity, or strength of the bond, of each bound conformation to the receptor (Crampon et al., 2022). There are a variety of sampling strategies, but here the triangle matcher algorithm was used that systematically aligns triplets of ligand atoms with alpha spheres on the receptor, which are triplets of receptor atoms identified during the site finder process to determine active sites (Chemical Computing Group, 2019a). The London δG scoring function was used to estimate binding affinity. Empirical methods such as London δG are a function of components such as hydrogen bonds as well as ionic, hydrophobic, and hydrophilic interactions among others, the values of which are determined using regression analysis on experimental data (Crampon et al., 2022).

Drug docking of PAL1, PAL2, mutant PALs 1-11A, CFL, and PF4 to CSA was performed using MOE. While induced fit docking was selected, a process where the side chains of both the receptor and the ligand are kept flexible during docking, CSA was instead kept entirely rigid while the ligands were kept flexible. This was due to issues with the MOE application labeling the entire CSA structure as a backbone, possibly stemming from converting the structure from MOE → NAMD (for SA) → MOE (for QuickPrep).

Furthermore, instead of specifying a binding region on CSA, all ligands were allowed to bind to any region on the receptor in a method known as blind docking (Hetényi & van der Spoel, 2002). The default parameters in MOE were used for docking except for the number of refinement poses, which was increased from 5 to 30 in order to increase the likelihood of obtaining the lowest energy conformation of each ligand. During docking, the placement process is used to generate poses of the docked ligand using the specified placement method, in this case triangle matcher. Once all poses were generated, the placement scoring function, London δG, was then utilized to estimate the binding affinity of each pose to the receptor.

Equation 1.1: The London δG scoring function utilized during the placement process of docking to estimate the binding energy of the ligand to the receptor. Here, c is the average gain or loss from rotational and translational entropy, E_{flex} represents the energy stemming from the loss of ligand flexibility, c_{HB} and c_M are ideals of hydrogen bond energy and metal ligation respectively, f_{HB} and f_M are both values within $[0, 1]$ that measure the geometric faults of hydrogen bonds and metal ligations respectively, and finally, D_i is a variable that represents the desolvation energy of atom i (Chemical Computing Group, 2019a).

$$\delta G = c + E_{flex} + \sum_{h-bonds} c_{HB} f_{HB} + \sum_{m-lig} c_M f_M + \sum_{atoms\ i} \delta D_i$$

After docking, the refinement process was performed to further improve the poses generated within the placement process. This involved the use of a molecular mechanics forcefield, specifically Amber10, and the final scoring of these refined poses was estimated using the Generalized Born solvation mode (GB/VI). Force field scoring functions such as GB/VI are derived from molecular force fields, in this case both MMFF94x and AMBER99, and calculate binding affinity using the sum of vdW interactions, electrostatics, and entropy terms (Chemical Computing Group, 2019a).

Equation 1.2: The GBVI/WSA δG forcefield-based scoring function that estimates the binding affinity of the docked ligand to the receptor. Here, c represents the average gain or loss from rotational and translational entropy, α and β are forcefield-dependent constants that were determined during training, E_{Coul} and E_{sol} represent the coulomb and solvent electrostatic terms respectively, E_{vdW} is the vdW forces, and $SA_{weighted}$ is the surface area (Chemical Computing Group, 2019b).

$$\delta G \approx c + \alpha\left[\frac{2}{3}(\delta E_{Coul} + \delta E_{sol}) + \delta E_{vdW} + \beta \delta SA_{weighted}\right]$$

Table 1.3: All parameters used during docking of PAL1, PAL2, PALs 1-11A, and PF4 to CSA.

Parameter	Value
Placement (Method)	Triangle Matcher
Placement (Score)	London δG
Placement (Poses)	30
Refinement (Method)	Induced Fit
Refinement (Score)	GBVI/WSA dG
Refinement (Poses)	30

Once MOE finished docking, Equation 1.3 below was used to normalize both the calculated binding affinities from MOE as well as the *in-vitro* experimental values from CSTS Health Care.

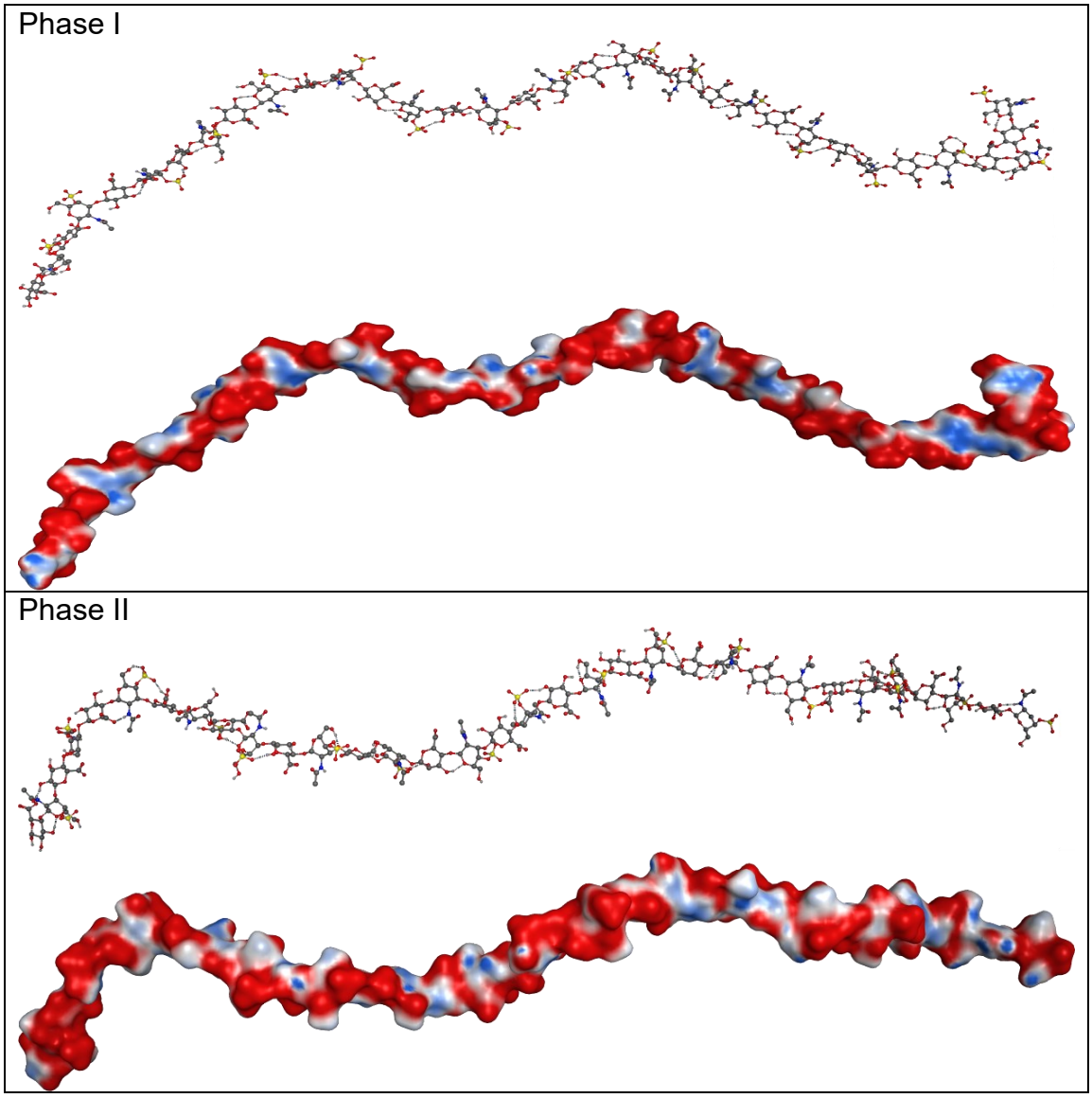
Equation 1.3: Formula used to normalize the binding affinities of both *in-silico* and *in-vitro* experimental values so they can be compared. ΔG is the binding affinity, K is the Boltzmann constant, T is the temperature, and $\overline{\Delta G}$ is the normalized value.

$$Z^{-1} = \sum_{i=1}^{N} exp\left(\frac{-\Delta G_i}{KT}\right)$$

$$\overline{\Delta G} = Z^{-1} \times \sum_{i=1}^{N}\left(\Delta G_i \times \exp\left(\frac{-\Delta G_i}{KT}\right)\right)$$

1.3 Results

Phase I SA of CSA (temperature: 295.15K) ran for a total of 12.97 ns, phase II (temperature: 310.00K) for 9.33 ns, phase III (temperature: 400.00K) for 10.04 ns, and the total simulation time for phase IV (temperature: 295.15K) was 10.47 ns. Figure 1.1 displays CSA at each phase, including the final phase IV structure that was used for drug docking which is color coded to display the electrostatic energy; red identifies areas that are more electronegative, white refers to neutral areas, and blue areas are more electropositive. The final phase IV CSA was shown to be mainly electronegative, and its conformation contained a small notch towards the center of an otherwise straight structure.


Phase I
Phase II

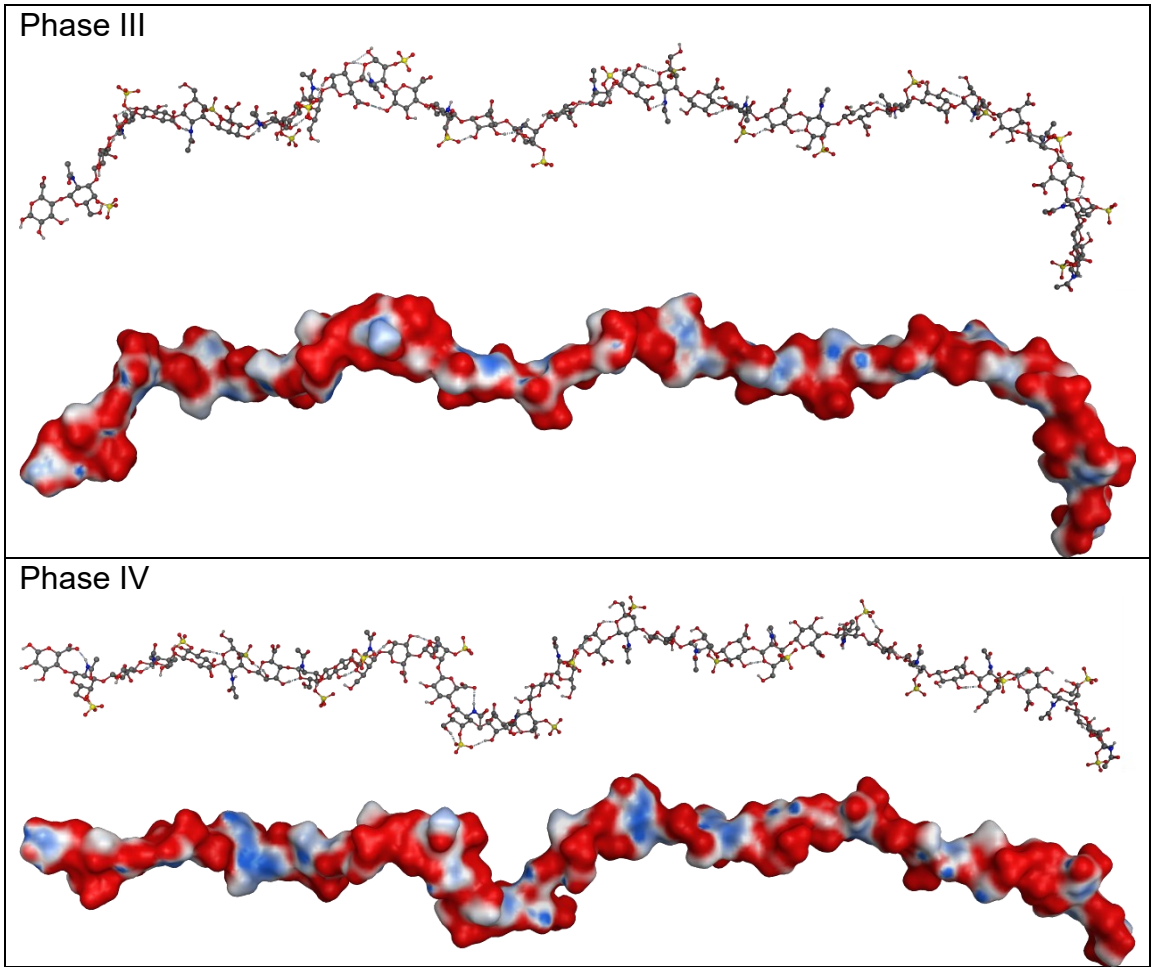


Figure 1.1: The structure of CSA after each of the 4 phases of SA, color coded to show its electrostatic energy. The color range indicates a net charge from -40 elementary charges (red) to 0 (white) to +40 elementary charges (blue).

After MD was completed, drug docking of PAL1, PAL2, mutant PALs 1-11A, CFL, and PF4 to the phase IV conformation of CSA resulted in 30 total docked poses for each of these ligands, where the lowest energy docked ligand was used to determine its preferred binding location on the CSA receptor. Most of

the ligands were bound around or to the central notch of the CSA structure, except for PAL9A that preferred the left-most side; see Figure 1.2.

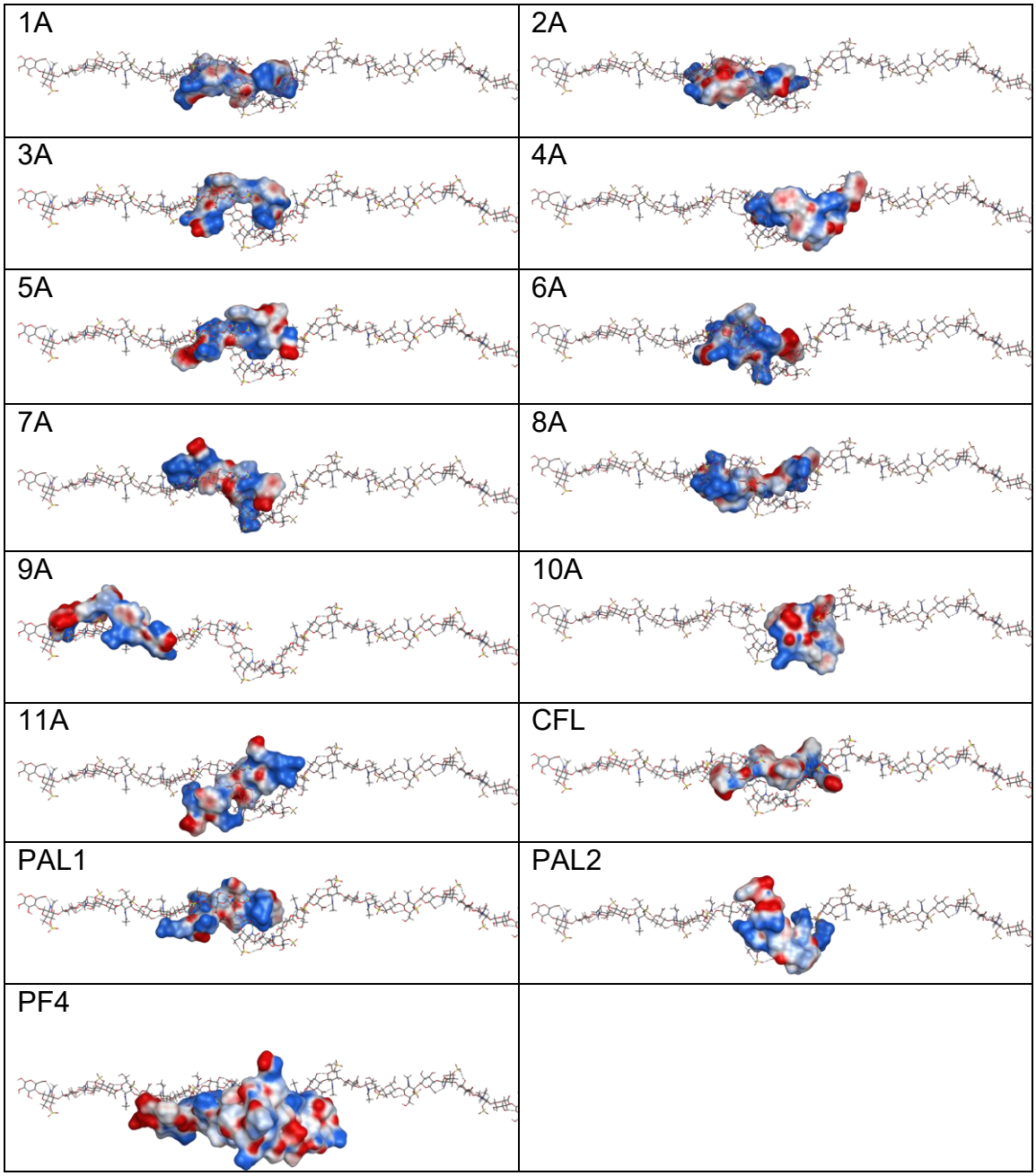


Figure 1.2: PAL1, PAL2, PALs 1-11A, CFL, and PF4 docked to the phase IV structure of CSA. Each ligand is color coded where red areas are more electronegative, white areas are neutral, and blue areas are more electropositive.

The normalized binding affinities from MOE and *in-vitro* experimental results are shown in Table 1.4. The Pearson R correlation between *in-vitro* and *in-silico* was approximately 0.50.

Table 1.4: Normalized scores of the *in-silico* docking results and *in-vitro* binding affinities obtained from CSTS Health Care. All units were originally in kcal/mol and are ordered from best/lowest to worst/highest *in-silico* binding affinity.

Ligand	$\overline{\Delta G}$ (*In-silico*)	$\overline{\Delta G}$ (*In-vitro*)
PF4	-17.48	N/A
PAL6A	-11.39	-8.23
PAL4A	-11.15	-5.39
PAL10A	-11.13	-6.15
PAL5A	-10.89	-6.32
PAL1	-10.84	-6.85
PAL8A	-10.74	-6.45
PAL2	-10.66	-6.88
PAL7A	-10.62	-3.32
PAL1A	-10.38	-4.48
PAL3A	-10.36	-7.14
PAL11A	-10.10	-4.60
PAL9A	-10.06	-6.18
PAL2A	-9.63	-5.41
CFL	-8.71	-4.33

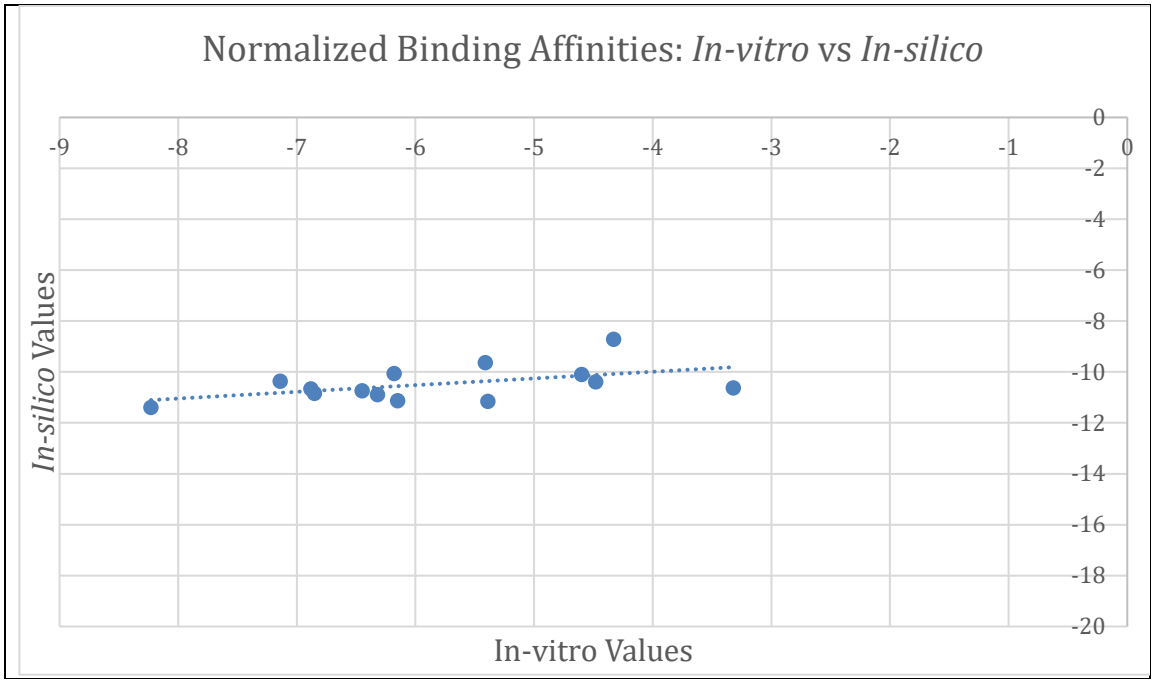


Figure 1.3: Graph displaying the *in-vitro* binding affinities of PAL1, PAL2, PALs 1-11A, and CFL to CSA on the x-axis while the corresponding *in-silico* docked values of each ligand were placed on the y-axis. Note that because CSTS Health Care did not perform an *in-vitro* binding of PF4 to CSA, this ligand was not shown. The Pearson R was 0.50.

Additionally, the ranking of both the *in-silico* and *in-vitro* results are shown below, ordered from lowest to highest binding affinity for comparison.

Table 1.5: The ranking of the docked and experimental ligands, ordered from best to worst. Note that the *in-vitro* binding of PF4 was not performed by CSTS Health Care, but is a natural high affinity ligand to GAGs and would therefore be the strongest binder.

Rank	*In-silico*	*In-vitro*
1	PF4	PF4*
2	PAL6A	PAL6A
3	PAL4A	PAL3A
4	PAL10A	PAL2
5	PAL5A	PAL1
6	PAL1	PAL8A
7	PAL8A	PAL5A
8	PAL2	PAL9A
9	PAL7A	PAL10A
10	PAL1A	PAL2A
11	PAL3A	PAL4A
12	PAL11A	PAL11A
13	PAL9A	PAL1A
14	PAL2A	CFL
15	CFL	PAL7A

The interacting charges between each ligand:CSA complex were also evaluated, and the total charge, defined as the sum of formal charges (FCharge), ranged between -1 and 7: PAL1A was 4, PALs 2-11A were within the 2-3 range, CFL was -1, and PF4 had a charge of 7. For electrostatic potential energy (E_ele) most of the ligands were within the -300 to -400 kcal/mol range, the greatest exceptions being CFL which was approximately -8 kcal/mol, and PF4 was approximately -786 kcal/mol. The sum of the atomic polarizabilities (apol) for each of the docked ligands were mostly between 238 and 247, CFL was in the 184 range, PAL1 and PAL2 were both approximately 252, and PF4 was approximately 1105.

1.4 Discussion

The SA phase IV structure of CSA was shown to uncoil from its phase I conformation, was mostly electronegative, and had a notch towards the center. During drug docking on the entire structure, this notch was found to be the preferred binding region for all ligands with the exception of PAL9A. The docking results showed PF4 had the highest affinity by far, with PALs 6A, 4A, and 5A trailing behind, and then PAL1 in fifth place. PAL2 was in seventh place, trailing behind PAL10A. As expected, CFL was in last place as a control ligand, which was consistent with the *in-vitro* results from CSTS Health Care. Electrostatic analysis using MOE confirmed that positively charged amino acid residues, mostly arginines, of the PALs formed ionic bonds with the negatively charged sulfate groups in CSA. That being said, the structure of the CSA:PAL1 model did not support an equal contribution of the 4 arginine side chains; instead, only arginines with spatial proximity to the sulfate groups on CSA promoted binding. These results suggest that modifying PAL to change the position of the arginines would allow it to better accommodate the sulfate groups on CSA during binding, improving the overall PAL:CSA affinity.

Comparison of the *in-vitro* and *in-silico* binding affinity rankings to CSA showed similarities; both PAL6A (rank: 2) and PAL11A (rank: 12) were consistently ranked between docking and experimental results, with PAL1 and PAL8A being within 1 rank of each another as well. The *in-silico* results showed

that CFL was the weakest binder as expected, however, *in-vitro* PAL7A was shown to be the worst binder to CSA. This was explained by CSTS Health Care stating that the proline being in the seventh position within PAL7A's FASTA sequence may have destabilized the structure or have positioned it in an unfavorable conformation that led to undetectable binding during isothermal titration calorimetry (ITC). Finally, PF4 was docked to CSA as a control for the *in-silico* results, and it met expectations as the strongest binder seeing as it has a natural affinity for negatively GAGs such as CSA.

1.5 Conclusion

The goal of this study was to develop a computational model of PALs obtained from CSTS Health Care derived from a high affinity PAL previously published by Butterfield et al. (2010). Establishing an *in-silico* model would allow for future research into PAL optimization for the purpose of targeted drug delivery with minimal to no off-target interactions. Here, the drug would be anchored to a PAL, which mimics the GAG binding domain, allowing it to be sequestered into the platelet alpha-granules and finally transported to the site of action. The PAL sequence obtained from Butterfield et al. (2010), known as PAL1, was their highest affinity peptide to CSA and it served as the base for the PAL2 as well as mutant PALs 1-11A sequences modeled within this study.

Multiple *in-silico* techniques were utilized in order to model and evaluate the interactions between the various PALs and CSA. First, 3D structures were built from the FASTA sequences of PAL1, PAL2, the 11 mutant PALs, and the control ligand, CFL. Moreover, the protein structure of PF4 was downloaded from the PDB in addition to a template structure for CSA that was used as a starting point for creating a 34-unit peptide. Next, a method known as SA was performed in 4 phases to obtain a low energy conformation of CSA that would most likely be found *in-vivo*. Finally, drug docking was performed between PAL1, PAL2, the mutant PALs, CFL, and PF4 to the phase IV conformation of CSA to evaluate both the electrostatic energies that contributed to binding as well as the rank of the ligands from strongest to weakest binders.

The computational methods used in this chapter, specifically homology modeling of CSA, MD/SA, and drug docking allowed for the evaluation of PAL:CSA binding. Furthermore, using the knowledge gained from this study, novel PALs can now be designed and tested *in-silico*, providing a more flexible and cost-efficient platform than a wholly experimental approach. In conclusion, the results show that PF4 and CFL had the highest and lowest affinity for CSA, which was expected. Evaluation of the electrostatic interactions between the PALs and CSA showed that the interactions between the negatively charged sulfate groups on CSA and the positively charged arginines on the PALs were responsible for binding. Future optimizations of PAL will include rearranging the position of the

arginines on PAL to better accommodate the sulfate groups on CSA, leading to a stronger binding affinity of the PAL:CSA complex.

The computational methods utilized in this chapter have been around for decades, the first instances of both drug docking as well as MD of biomolecules were performed in the late 1970s (Karplus, 2003; Amaro et al., 2018). Newer techniques have shifted towards using ML methods to make predictions regarding biological processes using data obtained from published studies, databases, and repositories. The following chapter explores how ML can be used to classify a given therapeutic's action (activator, blocker, or non-binder) towards a receptor, involving processes such as data augmentation, feature selection, and training and validation of the ML models.

Chapter 2: Supervised Machine Learning for Drug-Action Classification

2.1 Introduction

As discussed previously, there are various *in-silico* techniques used during the development of a novel therapeutic such as homology modeling, MD/SA, and drug docking. More recent advances in the field have shifted towards utilizing ML for various tasks including predicting both the binding affinity as well as the bound conformation of a ligand to a receptor (Yang et al., 2022; Isert et al., 2023), rapid screening of drug libraries for specific properties such as those that meet Lipinski's rule-of-5 for example (Cáceres et al., 2020), predicting the 3D structure of proteins from their amino acid sequences (e.g., AlphaFold), and so on. The availability of large amounts of biomedical data also allows researchers to focus on areas that have traditionally received relatively less attention, and in this chapter the focus was on drug-action prediction. A drug, defined here as any chemical that induces a biological response, can serve as an agonist (activator) or an antagonist (blocker) to a receptor such as a protein for instance (Neubig et al., 2003). More formally, an agonist is any drug that induces a biological response when binding to a protein, and antagonists reduce or block the action of any other drug, often agonists, by occupying the binding region on their target protein (Neubig et al., 2003).

The idea of 1-drug-1-target is no longer viable; complex diseases such as cancer involve numerous pathways and proteins, and pharmaceuticals designed to target multiple receptors have been shown to be more effective and efficient compared to those with high specificity to their targets (Kabir & Muth, 2022). This is collaborated in part by mouse knockout studies that have shown only 10% of all targetable genes are effective as individual targets (Kabir & Muth, 2022), or in other words, the majority of mouse genes need to be targeted in conjunction with other genes to be viable as a therapeutic target. That being said, whether its 1-drug-1-target or 1-drug-multiple-targets (polypharmacology), drug-action prediction towards a target is necessary to identify adverse drug interactions, better known as side effects, for a proposed therapy. These predictions are specifically important for performing as well as identifying potential side effects of drug repurposing, where an existing market-approved drug is used to treat a disease or indication not originally intended, especially when 2 or more pharmaceuticals are combined into a single treatment. Furthermore, predicting whether a drug would be an agonist or antagonist to a given target allows researchers to quickly filter drug libraries, often containing millions of chemicals, during early-stage drug discovery.

In this chapter, a series of ML models were trained to classify a given drug as an agonist, antagonist, or a non-binder, otherwise known as a decoy, to a set of receptors: the androgen (AR), estrogen (ER), glucocorticoid (GR), or

progesterone (PR) type 1 nuclear receptors. Only full agonists were considered because they are able to elicit the highest possible biological response from the receptor upon binding and activation, unlike partial and irreversible agonists (Pleuvry, 2004). For antagonists, competitive and non-competitive were considered valid because both types bind to the protein in the same manner, the main difference being that competitive antagonists can be displaced from the protein's binding region with a high enough concentration of an agonist, whereas non-competitive antagonists cannot (Pleuvry, 2004). Decoy drugs were defined as those that simply did not have any known affinity to the specified receptors.

2.1.1 Objectives

Supervised ML models were trained to predict whether a drug would be an agonist, antagonist, or decoy to each of the AR, ER, GR, or PR targets. There were 2 objectives for this thesis: 1) train separate ML models for each receptor with better classification accuracy, or performance, than baseline; defined as accuracy of simply classifying every drug as being part of the majority class, and 2) identify the ML model for each receptor with the best performance out of 5 popular learners: decision tree (DECTRE), naive Bayes (NAIBAY), neural net (NEUNET), random forest (RANFOR), and support vector machine (SVM).

Objective 1

The goal of the first objective was to develop a total of 5 ML models for each target: AR, ER, GR, and PR, which were chosen due to being well characterized in literature, meaning they have many known agonists, antagonists, and decoys available for training the DECTRE, NAIBAY, NEUNET, RANFOR, and SVM learners.

Objective 2

Once the ML models were trained for each receptor, the performance of each were compared to determine the best model for AR, ER, GR, and PR separately. The goal was to determine the learner with the best balance of performance and efficiency, specifically the time required for training, which would allow this method to be extended to other receptors beyond the ones utilized here.

2.1.2 Machine Learning

The overarching goal of ML is to utilize data in order to make predictions. In this chapter, supervised learning was used, a method where a training dataset consisting of labeled examples is provided to a learner so its underlying equations can determine the parameters that would allow it to accurately predict unseen data. The validation dataset, consisting of novel examples, is then used to assess the performance of the newly trained model (Dwork et al., 2015; El Naqa & Murphy, 2015; Learned-Miller, 2014). Each receptor had their own training and

validation datasets; a tab-separated value (TSV) file with the first column containing the drug names, and every subsequent column containing features, or characteristics about each drug such as its molecular weight, total number of aromatic rings, total number of atoms, and so on, with the exception of the final column that specified the label, or class each drug belonged to: agonist, antagonist, or decoy.

2.1.3 Related Work

Li et al. (2015) compared the performance of the NAIBAY, k-nearest neighbors (KNN), recursive partitioning, and SVM supervised ML models in identifying liver X receptor beta (LXR-beta) selective agonists from non-selective agonists. Using a training set of 176 compounds comprised of 69 selective and 107 non-selective agonists, a total of 324 trained models were developed using the abovementioned learners (Li et al., 2015). Note that the MOE and PaDEL-Descriptor applications were used to generate a total of 962 combined features for each agonist in the training dataset. The top 15 models, identified from an initial validation set of 58 compounds (22 selective, 36 non-selective), were assessed for a final time using a secondary validation dataset consisting of 73 selective LXR-beta agonists and 3 non-selective agonists, 76 novel compounds in total (Li et al., 2015). Of these 15 top models, 3 were able to classify selective and non-selective agonists in the secondary validation set with a performance greater than the baseline performance of 96.05%, calculated as the number of entries in the largest class

divided by the total number of entries. Overall, while SVM had the best performance over other models at 97.37%, the large skew towards selective LXR-beta agonists within the secondary validation set may not reflect the actual performance of the models in classifying selective versus non-selective agonists of this receptor.

Asako and Uesawa (2017) utilized supervised ML to predict ER agonists, specifically environmental pollutants that disrupt the endocrine system and lead to issues in reproduction and growth. In this study the authors developed a novel learner based on RANFOR, an ensemble learner that uses multiple DECTREs to make predictions, to identify chemicals that specifically target the ER ligand binding domain (LBD; Asako & Uesawa, 2017). The Tox21 Data Challenge 2014 library of 8,733 known ER-LBD binders was split 50:50 for training and validation, where a total of 4,071 features were generated for each chemical using a combination of the MOE 2013.08, MarvinView 6.0.0, and Dragon 6 applications. These features included a count of the various chemical groups (e.g., aromatic hydroxyls, phenol/enol/carboxyl OH), bond donor information (e.g., H-bond donor capacity), pH related features (e.g., lipophilicity under pH = 5.5), and molecular descriptors such as mass and surface area. Similar to Li et al. (2015) above, the first validation set was used for model selection while another consisting of 599 ER-LBD binding chemicals assessed the final performance of the single selected model (Asako & Uesawa, 2017). Here, the area under the receiver operator

characteristics (ROC-AUC) curve estimated that the performance of the selected model was 0.87, a metric in which a value of 1 signifies perfect classification accuracy and 0 indicates none (Asako & Uesawa, 2017).

Similarly, Russo et al. (2018) also focused on detection of endocrine disrupting environmental pollutants that bound to ER. Here, the Bernoulli NAIBAY, AdaBoost DECTRE, RANFOR, SVM, and deep NEUNET learners were trained using a dataset of 24,305 compounds amassed from several sources including Tox21 and CERAPP (Russo et al., 2018). In addition to chemical features such as molecular weight and the number of aromatic rings for instance, the molecular fingerprint of each chemical was also generated, which is a string of characters that describes the structure. For example, a variation of the Morgan algorithm is used to derive the extended-connectivity fingerprint (ECFP) of molecules such as butyramide for instance, where its ECFP of “-1708545601” describes the atoms, bonds, and connectivity of the structure (Rogers & Hahn, 2010). The performance of all models was evaluated using several scoring metrics including ROC-AUC, described above, as well as F1 score, precision, and recall, all of which utilize the terms: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). For example, when attempting to identify people who are sick with a disease, mistakenly classifying a healthy person as sick would be a FP while the inverse, classifying a sick person as healthy, would be a FN. In the same vein, correctly classifying a healthy person as healthy or a sick person as sick would be a TN and

TP respectively. The performance of all scoring metrics for each learner was normalized to $[0,1]$ and the mean was calculated to obtain the final score and subsequent ranking of each learner, from best to worst: RANFOR, deep NEUNET, SVM, Bernoulli NAIBAY, and AdaBoost DECTRE (Russo et al., 2018).

Equation 2.1: The equation for calculating precision (Russo et al., 2018).

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

Equation 2.2: The equation for calculating recall (Russo et al., 2018).

$$Recall = \frac{TP}{TP\ +\ FN}$$

2.2 Methods

An overview of the methods are shown in Figure 2.1. Published literature was first used to find agonists and antagonists for each receptor, while decoy drugs were obtained from the Database of Useful Decoys: Enhanced (DUD-E; Mysinger et al., 2012). Next, the agonists, antagonists, and decoys for every receptor were split in half to create a separate training and validation dataset for AR, ER, GR, and PR. In cases with an odd number of drugs, the remainder drug was placed into the training dataset. Due to the fact there were few agonists and antagonists available for each receptor (≤100 on average) compared to the number of decoys (>14,000), a process known as data augmentation was performed to generate various

conformations of the agonists and antagonists for each receptor, which only affected properties related to the 3D configuration of their atoms such as polarizability and dipole moments, to bolster these numbers. Once data augmentation was completed, the MOE application was then used to generate the properties, otherwise known as features, of each drug in the training dataset. This included features such as the number of aromatic rings, Lipinski's rule-of-5 violations, and rotatable bonds, as well as the potential and solvation energies among others. Appendix Table 1 lists all features.

Table 2.1: The total number of agonists, antagonists, and decoys obtained for each receptor.

Receptor	# Agonists	# Antagonists	# Decoys
AR	106	18	14,503
ER	62	28	20,818
GR	44	10	15,185
PR	45	7	15,814

RapidMiner Studio (version 9.6; Mierswa & Klinkenberg, 2020) was used to perform ML, where a process known as cross-validation (CV) was used to train each of the DECTRE, NAIBAY, NEUNET, RANFOR, and SVM learners. During each iteration of CV, feature selection was performed where the most useful, or relevant, features were kept and the rest were filtered so as not to be used during training, which reduced training time and potentially increased performance (Hall & Smith, 1998; Lee & Lee, 2006). Finally, the performance of all trained models in classifying novel agonist, antagonist, and decoy drugs within the validation dataset

was assessed. The RapidMiner Studio process used in this chapter is shown in Figure 2.2.

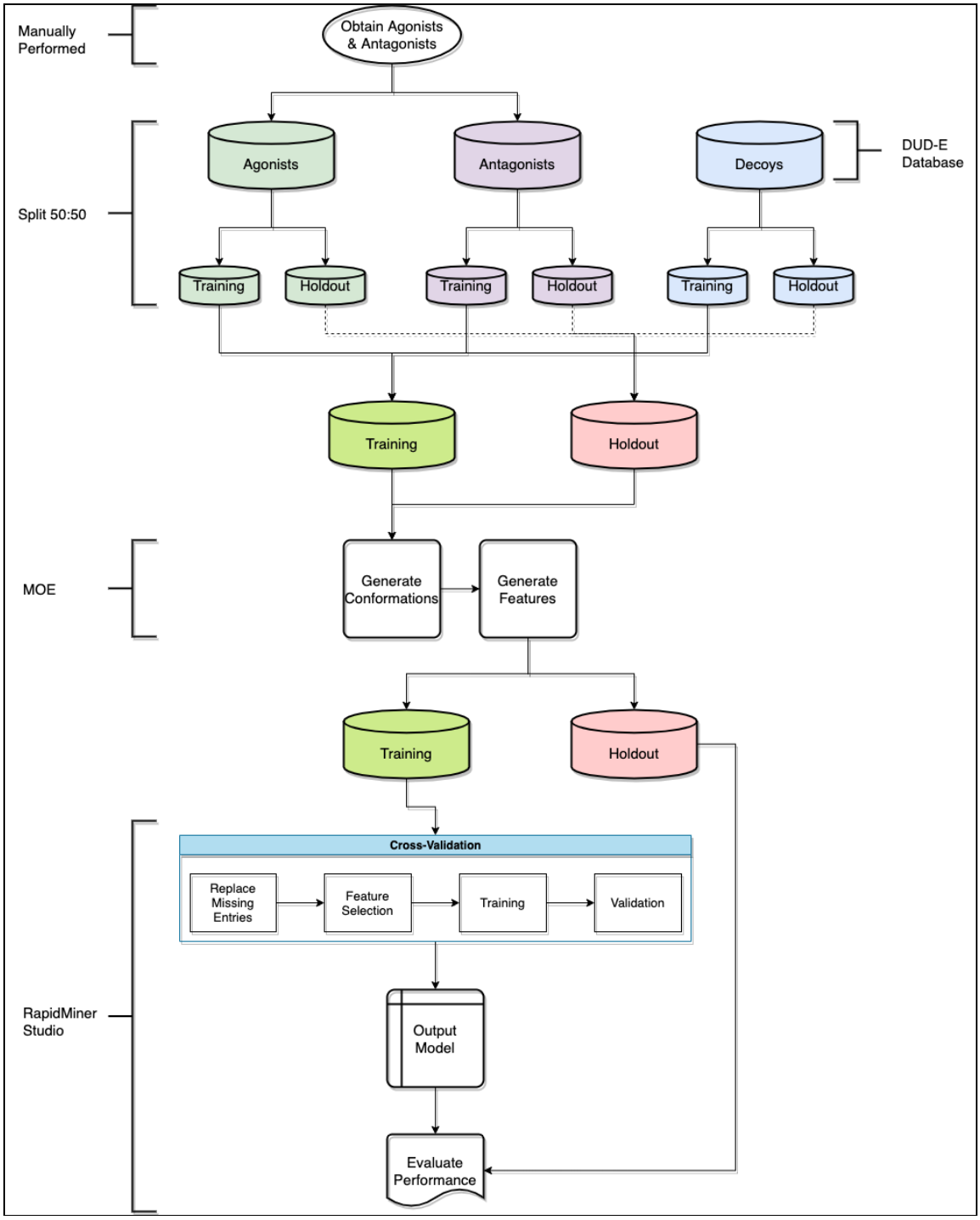


Figure 2.1: An overview of the ML workflow used.

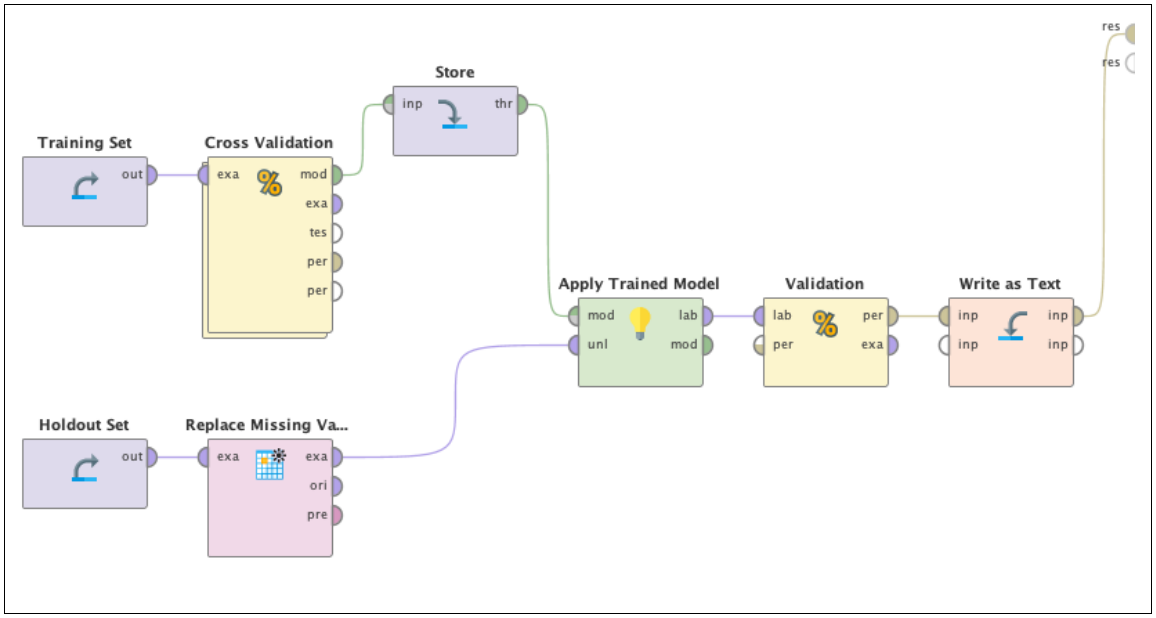


Figure 2.2: The RapidMiner Studio process used during ML.

The cost matrix is used to set the penalty for misclassification by the learner during training for a given class. In this chapter all classes were treated the same, meaning the penalty for misclassifying an agonist, antagonist, or decoy drug was exactly the same as the others; see Table 2.2. The reward for correctly classification was 1 while the penalty for misclassification was -1.

Table 2.2: The cost matrix used in this chapter.

	Agonist	Antagonist	Decoy
Agonist	1	-1	-1
Antagonist	-1	1	-1
Decoy	-1	-1	1

2.2.1 Compounds

Published literature as well as expert-curated databases such as the Protein Data Bank (PDB), DrugBank, ChEMBL, and ChemSpider were used to obtain agonists and antagonists for all targets: AR, ER, GR, and PR. The simplified molecular-input line-entry system (SMILES) string was downloaded for each drug, which represents the atoms and stereochemistry of a chemical structure (O'Boyle, 2012); see Figure 2.3. This format was chosen only because it was the most readily available option for all agonist and antagonist drugs.

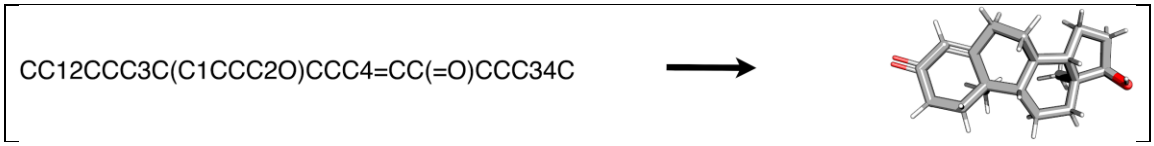


Figure 2.3: Testosterone SMILES string (left) and its corresponding 3-D structure (right).

Decoy compounds for each receptor were obtained from DUD-E (Mysinger et al., 2012), which served as training data for the decoy group. These decoys were in structure data file (SDF) format (Dalby et al., 1992) which can represent either the 2D or 3D coordinates of all atoms in a compound.

2.2.2 Data Augmentation

The limited number of agonists and antagonists obtained for each receptor, often less than 100 in total, created a risk of overfitting, where a model performs poorly

on novel data (e.g., the validation dataset) because the training data was not representative of the actual population (Mutasa et al., 2020). To alleviate this issue, a process known as data augmentation was performed where new entries are generated from existing ones (Lemley et al., 2017), and in this case, MOE was used to generate numerous conformations of each agonist and antagonist via the “Conformation Import” tool. This process was only performed for the agonists and antagonists since there were already thousands of decoys available for each receptor.

MOE generated conformations using 5 main steps. First, acids or bases that have previously been (de)protonated were corrected, and then filtering was performed where drugs that did not meet specific thresholds and/or violated certain rules such as Lipinski's rule-of-5 were removed (Chemical Computing Group, 2019a). Each drug was then broken into overlapping fragments where the conformation(s) of these individual fragments were determined using a stochastic conformational search (Chemical Computing Group, 2019a), the parameters of which are shown in Appendix Table 2.

Table 2.6: Total time used for conformation generation on the Compute Canada Graham platform. All times are rounded to the nearest minute (m) or hour (h).

	Time (Agonists)	Time (Antagonists)
AR	1h, 32m	5m
ER	19m	49m
GR	1h, 23m	2m
PR	29m	2m

Table 2.3: Total time used to perform conformation generation on the Compute Canada Graham platform. All times are rounded to the nearest minute (m) or hour (h).

	Time (Agonists)	Time (Antagonists)
AR	1h, 32m	5m
ER	19m	49m
GR	1h, 23m	2m
PR	29m	2m

The individual fragments of various conformations were put back together to create numerous conformations of the original drug, and the vdW energy of each conformation was calculated to filter those with bad contacts between atoms (Chemical Computing Group, 2019a). Finally, the strain energies were calculated and, along with the conformations, were written to the output file (Chemical Computing Group, 2019a).

Table 2.4: Total number of agonists within the training and validation datasets before and after conformation generation.

	Training		Validation	
	Before	After	Before	After
AR	53	2198	53	1579
ER	31	1440	31	1417
GR	22	4014	22	3437
PR	23	1434	22	799

Table 2.5: Total number of antagonists within the training and validation datasets before and after conformation generation.

	Training		Validation	
	Before	After	Before	After
AR	9	588	9	853
ER	14	2370	14	1337
GR	5	542	5	477
PR	4	584	3	258

Table 2.6: Total time used for conformation generation on the Compute Canada Graham platform. All times are rounded to the nearest minute (m) or hour (h).

	Time (Agonists)	Time (Antagonists)
AR	1h, 32m	5m
ER	19m	49m
GR	1h, 23m	2m
PR	29m	2m

2.2.3 Feature Generation

Features are properties or characteristics that describe a given agonist, antagonist, or decoy such as its molecular weight, number of hydrogen/carbon/nitrogen/etc. atoms, and potential energy that allows a trained model to differentiate between classes. MOE was used to calculate a total of 435

features for all agonists, antagonists, and decoys via the QuaSAR-Descriptor utility, which included (a) 2D (e.g., number of acidic, aromatic, H-bond donor, heavy, etc. atoms; number of single, rotatable, triple, etc. bonds; molecular weight; vdW volume and surface area; etc.), (b) internal 3D (i3D; e.g., total, potential, electronic, electrostatic, etc. energy; polar surface area; surface rugosity; etc.), (c) external 3D (x3D; e.g., dipole moment, principal moment of inertia, non-bonded interaction energy, etc.), and (d) protein (e.g., net charge; hydrophobicity; accessible surface area, volume, etc.) descriptors (Chemical Computing Group, 2019a).

2.2.4 Importing Data into RapidMiner Studio

Once the training and validation datasets for AR, ER, GR, and PR were finalized, they were imported into RapidMiner Studio. Figure 2.2 shows an overview of the ML process, which contains an operator named “Replace Missing Va…” that expands to "Replace Missing Values". During feature generation, some features were unable to be computed for certain drugs and consequently had blank or missing values, which posed an issue for learners such as NEUNET. To mitigate this, all entries with missing values were replaced with -1, a unique value that did not occur elsewhere. Other methods for handling missing values included the removal of the drug from the dataset as well as replacement using the mean or median value of the corresponding feature. Neither of these were ideal, as there was no guarantee that the feature associated with the missing value would have

been chosen during the feature selection process, leading to the removal of an otherwise valid drug. Furthermore, the use of -1 for only missing values avoided overlap between calculated and missing data unlike the median or mean, which may have led to inconsistencies between features. For example, in the training dataset for AR the mean number of carbon atoms was 18.4 and the median was 19, which would have conflicted with any drug weighing <216 Daltons (Da) or <228 Da respectively, seeing as a single carbon atom has a weight of 12.01 Da (de Laeter et al., 2003).

2.2.5 Machine Learning

Training of the DECTRE, NAIBAY, NEUNET, RANFOR, and SVM learners was completed using external k-fold CV where k = 10.

Table 2.7: Total number of agonists, antagonists, and decoys within the training and validation datasets for each receptor after data augmentation.

Receptor	Training	Validation
AR	10,036	9,687
ER	14,224	13,160
GR	12,148	11,509
PR	9,924	8,967

2.2.5.1 Cross-Validation

k-fold CV is a data resampling method that reduces the likelihood of overfitting by training and validating multiple models based on different subsets of the entire

training dataset. First, the training dataset X was first split into k equally sized subsets, $\{x_1, x_2, \ldots, x_k\}$, and the learner was then trained using k-1 subsets, $\{x_1, x_2, \ldots, x_{k-1}\}$, where the last subset, x_k, was used as the validation set. This process occurred k number of times such that each subset was used for validation exactly once. Finally, the performance of each model on their corresponding iteration, otherwise known as a fold, was averaged into a final score (Refaeilzadeh et al., 2009; Mierswa & Klinkenberg, 2020). Note that the output model produced by RapidMiner Studio was based on the average performance of each model within every fold. Stratified sampling was used where both the training and validation subsets used in each fold contained approximately the same proportion of classes as the entire training dataset (Mierswa & Klinkenberg, 2020). For instance, the PR training dataset was comprised of approximately 14.5% agonists, 5.9% antagonists, and 79.7% decoys, which was kept largely intact within each training and validation subset created during each fold of CV. Table 2.8 contains an example of 3-fold CV.

Table 2.8: Example of k-fold CV where k = 3 and the training dataset, X, is split into $\{x_1, x_2, x_3\}$ subsets. The final performance of the learner is an average of all scores over all folds.

Fold	Training	Validation	Score
1	$\{x_1, x_2\}$	$\{x_3\}$	s_1
2	$\{x_1, x_3\}$	$\{x_2\}$	s_2
3	$\{x_2, x_3\}$	$\{x_1\}$	s_3
	$Performance = \frac{s_1 + s_2 + s_3}{3}$		

2.2.5.2 Feature Selection

Feature selection is a process used to identify and remove irrelevant features from the training dataset, which leads to a reduction of noise and can increase performance by allowing the learner to more easily distinguish relationships between the remaining features and classes (Hall & Smith, 1998). The forward selection algorithm was used to perform feature selection, which occurred internally within each fold of k-fold CV before training. Starting from an empty set, $F = \{\}$, every feature was evaluated individually using 5-fold CV and the one that best classified agonists, antagonists, and decoys was added to the feature set, $F = \{f_1\}$ (Reif & Shafait, 2014). Next, F was coupled with every other feature (e.g., $\{f_1, f_2\}$, $\{f_1, f_3\}$, $\{f_1, f_4\}$, etc.) to again determine which combination yielded the best performance. Feature set F, for example $F = \{f_1, f_3\}$, was then continuously combined with all remaining features until either all features were evaluated, or the classification performance plateaued after a specified number of iterations, kept as the default value of 1 in this chapter (Reif & Shafait, 2014). The final set of features, $F = \{f_1, f_3, f_6, f_{87}, f_{103}, \dots\}$ for instance, were the ones kept while the rest of the features were discarded. The parameters used by RapidMiner Studio for feature selection can be found in Appendix Table 3.

In RapidMiner Studio the “Forward Selection” operator was a nested one containing the “Cross Validation” operator, which in turn was comprised of a training and validation phase; see Figure 2.4.

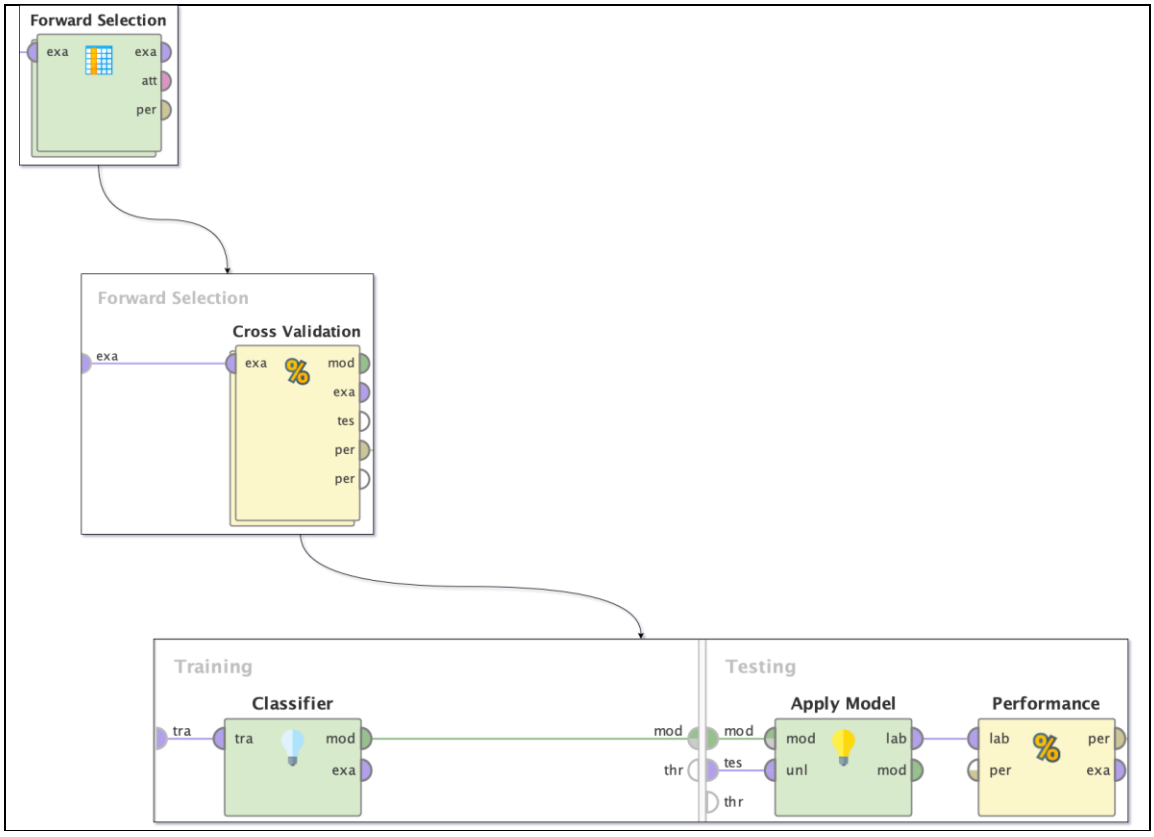


Figure 2.4: Architecture of the Forward Selection operator in RapidMiner Studio. Starting from the top, the operator "Forward Selection" contains the "Cross Validation" operator, which in turn was comprised of training and validation phases. The "Classifier" operator is one of DECTRE, NAIBAY, NEUNET, RANFOR, or SVM. The "exa" port refers to the example set, or input data, the "mod" port is the output model obtained from a given operator, "tes" is the test or validation subset for the current fold of CV, the "unl" port is for unlabeled data, "lab" is labelled data, and the "per" port is the estimated performance of the model.

2.2.5.3 Decision Tree

A DECTRE is a tree-like structure with a root, or the starting point for classification, that has attached branches, which represent decisions stemming from features, which contain many nodes, each of which route the query drug based on a specific

threshold for a given feature, that eventually terminate at leaves, where classification occurs (Mierswa & Klinkenberg, 2020); see Figure 2.5. For instance, in RapidMiner Studio the construction of a DECTRE from a training dataset starts with using the gain ratio criterion to select the best feature for splitting. All possible splits for every feature are first evaluated, for example the number of Lipinski rule-of-5 violations, $X = \{0,1,2,3\}$, and the number of carbon atoms, $Y = \{4, 5, 6, 7\}$. The midpoints of all feature values are then used as thresholds for splits (Mierswa & Klinkenberg), in this case $\{0.5, 1.5, 2.5\}$ and $\{4.5, 5.5, 6.5\}$ respectively, where the gain ratio of each threshold split (e.g., drugs with ≤0.5 Lipinski rule-of-5 violations versus those with >0.5, drugs with ≤4.5 number of carbon atoms versus those with >4.5, etc.) is calculated. The feature and threshold with the best or highest gain ratio is chosen as the first split, and then this process is recursively repeated until either all features have been evaluated or a stopping criterion, such as the maximum depth of the tree, is met (Mierswa & Klinkenberg, 2020; Mori, 2002).

Equation 2.3: The gain ratio of each potential split (S) for a given feature is evaluated to identify the one with the best, or highest, value. Here, d represents the drugs (agonist, antagonist, decoy) in each group (S_i), such as the ≤0.5 Lipinski rule-of-5 violations versus >0.5 groups for example, and $\text{freq}(d, S)$, S_i, and $|S_i|$ represent the total number of drugs within S and the ith group of S. All equations below were obtained from Mori (2002).

$$\text{gain\_ratio} = \frac{\text{gain}(d,S)}{\text{split\_info}(S)}$$
$$\text{gain}(d,C) = \text{entropy}(d,S) - \text{entropy}_\text{p}(d,S)$$
$$\text{entropy}(d,C) = -p(d\|S) \times log_2(p(d\|S))$$
$$p(d\|C) = \frac{\text{freq}(d,S)}{\|S\|}$$
$$\text{entropy}_\text{p}(d,C) = \sum_i \left(\frac{S_i}{S} \times \text{entropy}(d,S_i)\right)$$
$$\text{split\_info} = -\sum_i \left(\frac{\|S_i\|}{\|S\|} \times log_2\left(\frac{\|S_i\|}{\|S\|}\right)\right)$$

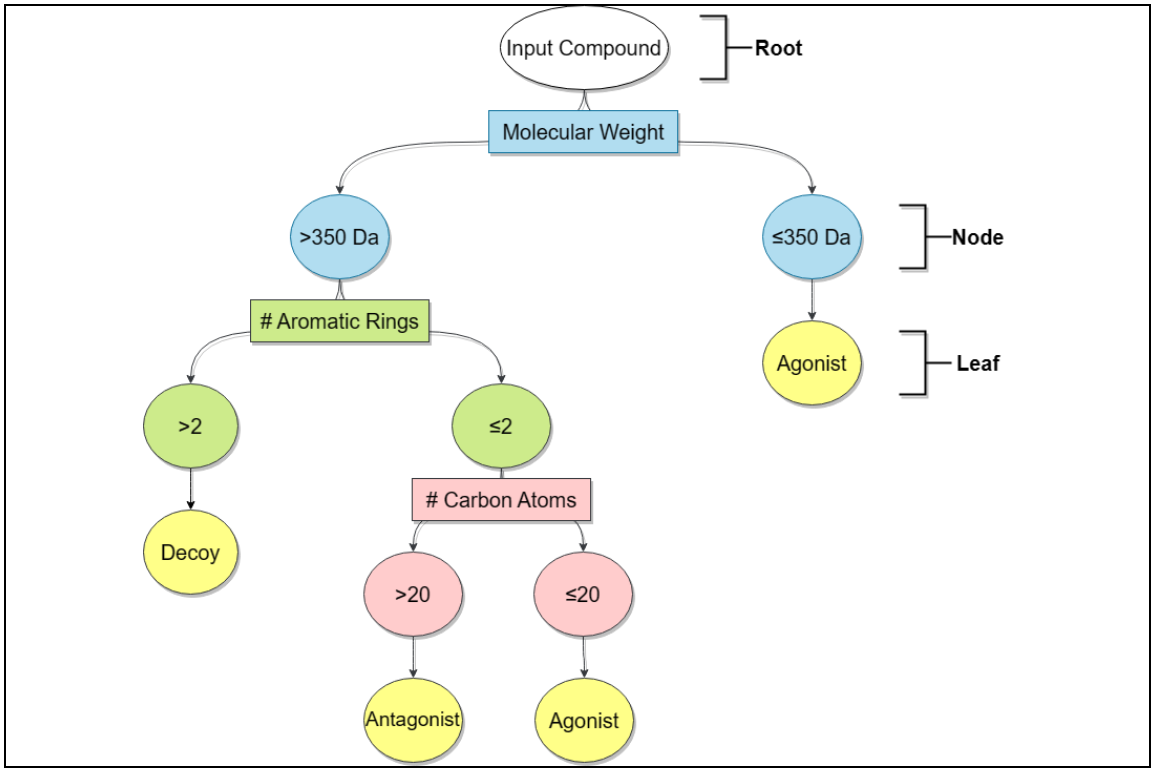


Figure 2.5: An example DECTRE where classification is performed based on the input drug's molecular weight, number of aromatic rings, and/or the total number of carbon atoms. For example, any drug that weighs more than 350 Da and contains more than 2 aromatic rings is classified as a decoy, while any drug ≤350 Da is classified as an agonist.

2.2.5.4 Naive Bayes

NAIBAY is based on Bayes' theorem with the assumption that all features within the training dataset are independent of one another (Zhang, 2004), see Equation 2.4. The trained model provides the prior, or probability, of each class within the training dataset, as well as the conditional probability, the probability of observing each feature within every class. Using the PR training dataset as an example, the priors are P(agonist) $= 0.14$, P(antagonist) $= 0.06$, and P(decoy) $= 0.80$, which was calculated by dividing the total number of entries within each class by the total

number of entries in the entire dataset. Additionally, the mean and standard deviation of each feature within each class was calculated since gaussian NAIBAY was used, which assumed the feature values all followed a normal or gaussian distribution (Pedregosa et al., 2011); see Equation 2.4G.

Equation 2.4: A) Bayes' theorem is the basis for the B) NAIBAY classifier that maps the relationship between a given class (y) and dependent feature vector ($\{x_1, \dots, x_n\}$). This equation can be simplified (C-E), where F) the estimated class is determined by calculating the product between the prior ($P(y)$) of each class and the likelihood ($P(x_i|y)$) of each feature, and the best or highest value is used for prediction. G) Gaussian NAIBAY was used, which defines the likelihood of a feature within a class. Equation A was obtained from Zhang (2004) and equations B-G were obtained from https://scikit-learn.org/stable/modules/naive_bayes.html (Pedregosa et al., 2011).

A)	$$P(A\|B) = \frac{P(A)P(B\|A)}{P(B)}$$
B)	$$P(y\|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n\|y)}{P(x_1, \dots, x_n)}$$
C)	$$P(x_1, \dots, x_n\|y) = P(x_i\|y)$$
D)	$$P(y\|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i\|y)}{P(x_1, \dots, x_n)}$$
E)	$$P(y\|x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i\|y)$$
F)	$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i\|y)$$
G)	$$P(x_i\|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

To classify a drug containing 30 carbon atoms and a molecular weight of 420 Da for example, the mean and standard deviation of these features for each class are used: $\mu_{\text{carbon atoms}}(\text{agonist}) = 24.13$, $\sigma_{\text{carbon atoms}}(\text{agonist}) = 3.04$, and so on. For each class, the product of the prior and likelihood of each feature occurring

in this class are calculated: $P(\text{agonist}|\text{carbon atoms} = 30) = P(\text{agonist}) \times P(\text{carbon atoms} = 30|\text{agonist}) = 0.14 \times 0.02 = 0.00$, $P(\text{antagonist}|\text{carbon atoms} = 30) = 0.06 \times 0.46 = 0.03$, etc., and the highest probability is used to determine the predicted class.

2.2.5.5 Neural Network

The NEUNET learner was inspired by the human brain, where it utilizes what are known as neurons to send and/or receive information to other neurons in the system and perform classification (Svozil et al., 1997; Mierswa & Klinkenberg, 2020). A NEUNET is comprised of 3 types of layers: an input layer with as many neurons as there are features, each representing a value for each feature, hidden layer(s) that processes input from all neurons within previous layers using weights as well as an activation function, and finally an output layer where classification is performed (Karsoliya, 2012; Mierswa & Klinkenberg, 2020). Structurally, every neuron in a given layer is connected to all neurons in the following layer (Karsoliya, 2012). Multiple hidden layers can be specified, but in this chapter a total of 2 hidden layers was used, the default value from RapidMiner Studio 9.6.

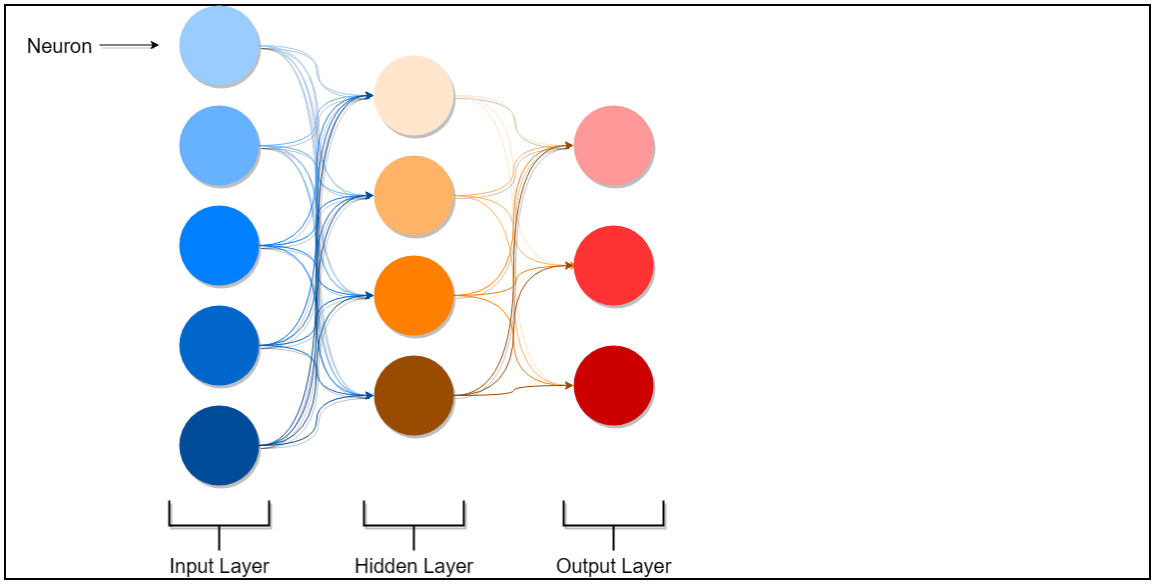


Figure 2.6: An overview of a NEUNET consisting of 1 hidden layer and 3 possible output classes.

For a training dataset with a total of m features, let the corresponding feature value set be $F = \{x_1, x_2, \ldots, x_m\}$. Note that RapidMiner Studio normalized all feature values to be $[-1,1]$ since the sigmoid activation function was used, which maps input values to $(0,1)$. This avoided potential issues such as correlations between features that would have otherwise been missed if each feature operated on their own scale (e.g., number of carbon atoms $= [18, 27]$, molecular weight $= [272.39, 521.44]$). Each one of these feature values was represented as one neuron within the input layer.

Equation 2.5: Equation used to normalize feature values for the NEUNET learner. x_i corresponds to the value of a given feature, and F_{min} and F_{max} represent the minimum and maximum values within feature set F respectively. This equation was taken from RapidMiner Studio https://github.com/rapidminer/rapidminer-studio/blob/7124551801923decfe2c0e077e329a1686087c12/src/main/java/com/rapidminer/operator/learner/functions/neuralnet/InputNode.java#L54.

$$\frac{x_i - F_{min}}{F_{max}}$$

Next, all neurons within the input layer are passed to each neuron in the first hidden layer, where their values are transformed using weights ($\{w_1, w_2, \dots w_m\}$) and a bias (b) that serves as a scalar: $w_1x_1 + w_2x_2 + \cdots + w_mx_m + b$, followed by the activation function shown in Equation 2.6 (Pedregosa et al., 2011). The weights and biases are initially set as random values but are later optimized during backpropagation (Mierswa & Klinkenberg, 2020).

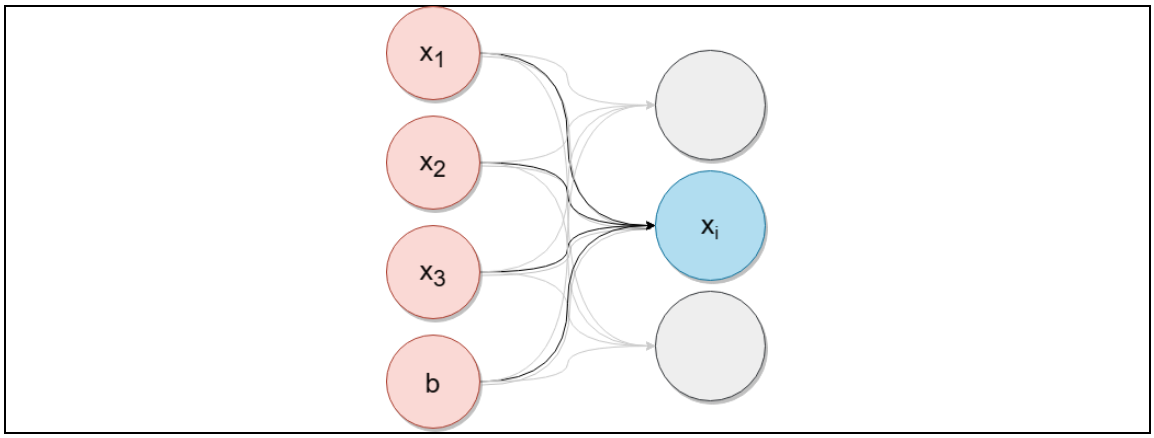


Figure 2.7: The transformation of neurons from a previous layer to the current one, for example from the input to a hidden layer, is performed by calculating the sum of the product between the weights ($\{w_1, w_2, w_3\}$) and value of each neuron ($\{x_1, x_2, x_3\}$) and the bias (b), followed by the use of an activation function (f): $x_i = f(w_1x_1 + w_2x_2 + w_3x_3 + b)$.

Equation 2.6: The sigmoid activation function (Erb, 1993; Svozil et al., 1997; Mierswa & Klinkenberg, 2020).

$$f(x) = \frac{1}{1 + e^{-x}}$$

The output layer receives data from the final hidden layer and utilizes an activation function, in this case the sigmoid function again, to perform classification (Mierswa & Klinkenberg, 2020). For example, a value $[0, 0.3]$ could refer to agonists, $[0.4, 0.6]$ for antagonists, and $[0.7, 1.0]$ for decoys. At this point classification has occurred for the first time, and the error function quantifies how far away the predicted value is from the ground truth.

To minimize error, a process known as backpropagation was used to optimize the weights and biases from all hidden layers, starting from the final and ending at the first (Mierswa & Klinkenberg, 2020). A gradient descent algorithm was used to calculate the direction of steepest ascent, or gradient, of the error from every weight and bias (Haji & Abdulazeez, 2021; Mierswa & Klinkenberg, 2020). Each weight was then adjusted to reduce the overall error of the network, or in other words, move in the opposite direction of the gradient (Haji & Abdulazeez, 2021). This process was repeated until either the specified number of training cycles, 200 by default, was reached or the error falls below the error epsilon threshold: 0.0001.

Equation 2.7: Error function used by RapidMiner Studio 9.6 for NEUNET, where y corresponds to the correct value, $\hat{y}$ refers to the predicted output value, and F_{max} is the maximum output value, 1 for the sigmoid function (Mierswa & Klinkenberg, 2020).

$$\frac{y - \hat{y}}{F_{max}}$$

2.2.5.6 Random Forest

RANFOR is an ensemble method which utilizes multiple DECTREs to perform prediction, with RapidMiner Studio utilizing 100 trees by default (Mierswa & Klinkenberg, 2020). Bootstrap sampling was used to build each DECTRE within the ensemble, a process where samples of the training dataset are randomly drawn with replacement, allowing them to be used again. Once all DECTREs were built, RANFOR determined the predicted class by selecting the one in which the majority of DECTREs predicted (Mierswa & Klinkenberg, 2020); see Figure 2.8.

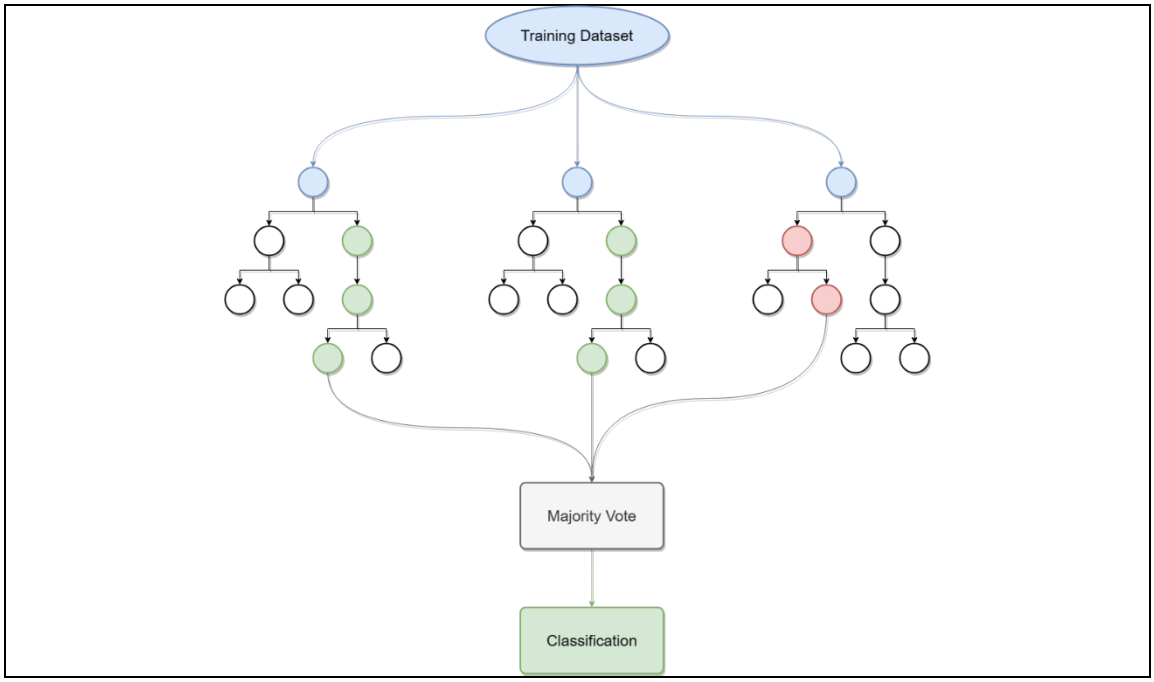


Figure 2.8: RANFOR uses multiple DECTREs to perform classification, where the majority vote is used. In this example, 2 DECTREs predict green while only one predicts red, making green the predicted output.

2.2.5.7 Support Vector Machine

The SVM learner implemented in RapidMiner Studio was based on the LIBSVM library (Chang & Lin, 2019). The radial basis function (rbf) kernel was used to perform classification, a non-linear kernel that maps training data onto an n-dimensional space, where n represents the total number of features, to determine the optimal hyperplane that maximizes the margin between the closest points of each class, or simply put, best separates 2 classes from one another (Mierswa & Klinkenberg, 2020; Pedregosa et al., 2011); see Figure 2.9.

Equation 2.8: The goal of SVM is to determine the weights (w) and bias (b) that maximizes the margin between the closest points of both classes, which is performed by minimizing $w^T w$ (Pedregosa et al., 2011). Additionally, a penalty (ζ) is applied for misclassification with C serving as a scalar that determines the strength of the penalty (Pedregosa et al., 2011). Equations obtained from https://scikit-learn.org/stable/modules/svm.html#svc (Pedregosa et al., 2011).

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

In order to handle multiple classes, a one-against-one approach was used where $\frac{n(n-1)}{2}$ total SVM models were generated, n being the total number of classes, and the majority vote from all SVMs was used to perform prediction (Chang & Lin, 2019), similar to RANFOR. In this chapter, a total of 3 models were generated: 1) agonist versus antagonist, 2) agonist versus decoy, and 3) antagonist versus decoy.

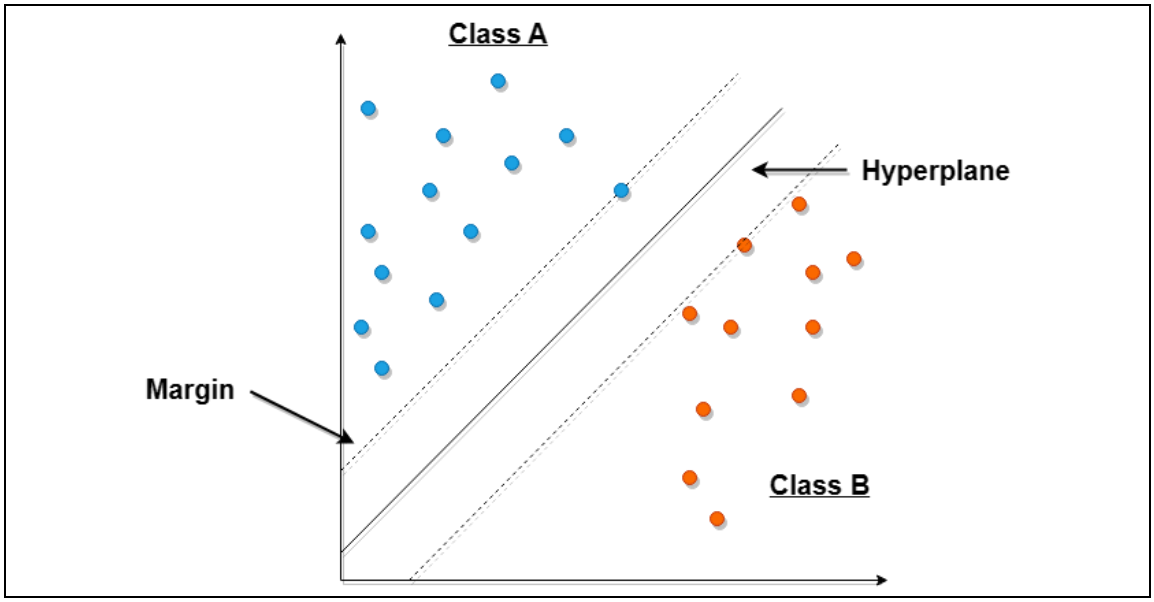


Figure 2.9: SVM constructs a hyperplane that separates 2 classes from one another, where the maximum size of the margins is based on the closest points from each class. Here, both class A (blue) and B (orange) have one point on the margin of their respective side while all other points fall behind the calculated margins.

2.2.5.8 Performance Assessment

Completion of CV generated a trained ML model for each learner, where its performance in classifying agonist, antagonist, and decoy drugs was assessed using the validation dataset. Note that each receptor's dataset was split in half for the training dataset, used during CV to train each learner, and the validation dataset, which comprised of novel examples never seen by any of the models.

2.3 Results

The performance of DECTRE, NAIBAY, NEUNET, RANFOR, and SVM learners for each receptor's dataset are shown below, which also includes the selected features and total runtime as well. However, in order to properly assess the performance of these 5 classifiers, the baseline percentages for each receptor are shown in Table 2.9. The baseline percentage is defined as the total number of entries in the largest class divided by the total number of entries across all classes. In the ER validation dataset for example, there are 1,417 agonists, 1,337 antagonists, and 10,405 decoys. Since decoys are the largest class, the baseline percentage would be $\frac{\text{decoys}}{\text{agonists + antagonists + decoys}}$ or $\frac{10405}{1417 + 1337 + 10405}$, which is approximately 0.79. This means that classifying all drugs as decoys, regardless of their features, will result in performance of 79%. Consequently, the performance of all models should always be better than baseline, otherwise its classification accuracy is no better than simply placing all queries into the largest class.

Table 2.9: The baseline percentages for all receptors. The decoy class contains the largest number of entries over all classes for all receptors, so it was used to calculate these percentages.

Receptor	Baseline Percentage
AR	75%
ER	79%
GR	66%
PR	88%

The following sections display each of the DECTRE, NAIBAY, NEUNET, RANFOR, and SVM learners' results on the validation set for each of AR, ER, GR, and PR. This includes the precision (defined in Equation 2.1) and recall (defined in Equation 2.2) metrics.

2.3.1 Performance: AR

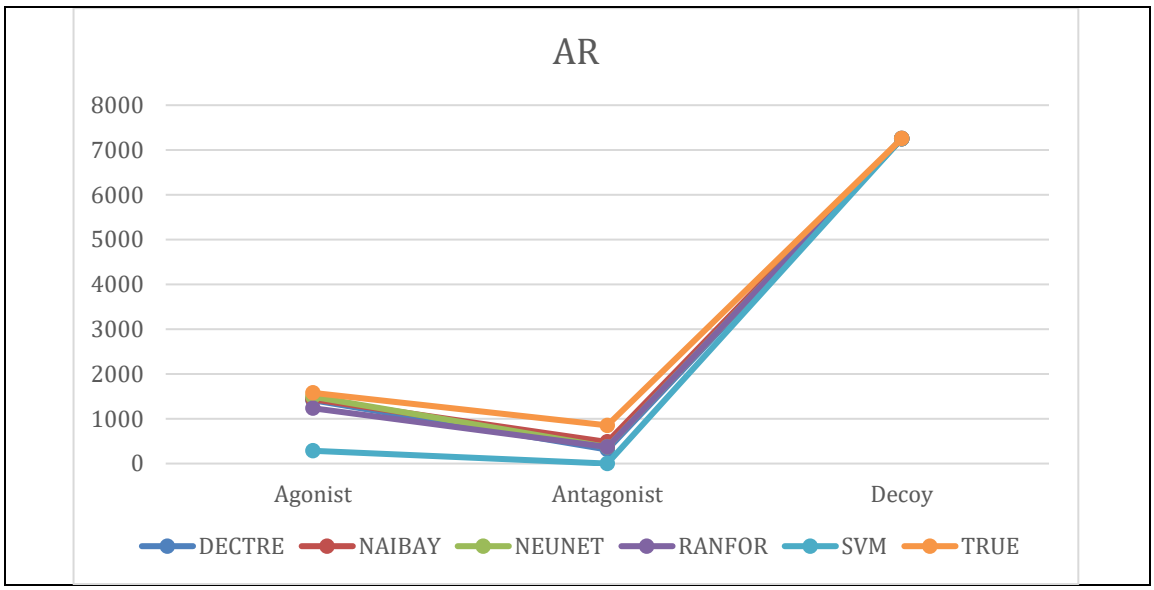


Figure 2.10: The performance of each model compared to the ground truth (TRUE) in classifying agonists, antagonists, and decoys for AR. All models had similar performance in classifying decoys, while SVM had more FNs for agonist and antagonist prediction in comparison.

Table 2.10: DECTRE. Run-time: 3 minutes 33 seconds. Performance: 92.73%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	1415	239	0	85.55%
Pred. Antag.	85	313	0	78.64%
Pred. Decoy	79	301	7254	95.02%
Recall	89.61%	39.69%	100.00%	

Table 2.11: NAIBAY. Run-time: 1 minute 9 seconds. Performance: 94.64%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	1434	243	2	85.41%
Pred. Antag.	67	482	1	87.64%
Pred. Decoy	78	128	7251	97.24%
Recall	90.82%	56.51%	99.96%	

Table 2.12: NEUNET. Run-time: 6 hours 34 minutes 37 seconds. Performance: 94.09%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	1492	218	0	87.25%
Pred. Antag.	84	370	2	81.14%
Pred. Decoy	3	265	7252	96.44%
Recall	94.49%	43.38%	99.97%	

Table 2.13: RANFOR. Run-time: 2 hours 21 minutes 18 seconds. Performance: 91.41%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	1232	184	2	86.88%
Pred. Antag.	347	370	0	51.60%
Pred. Decoy	0	299	7252	96.04%
Recall	78.02%	43.38%	99.97%	

Table 2.14: SVM. Run-time: 1 day 16 hours 47 minutes 47 seconds. Performance: 77.83%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	286	0	1	99.65%
Pred. Antag.	0	0	0	00.00%
Pred. Decoy	1293	853	7253	77.17%
Recall	18.11%	00.00%	99.99%	

2.3.2 Performance: ER

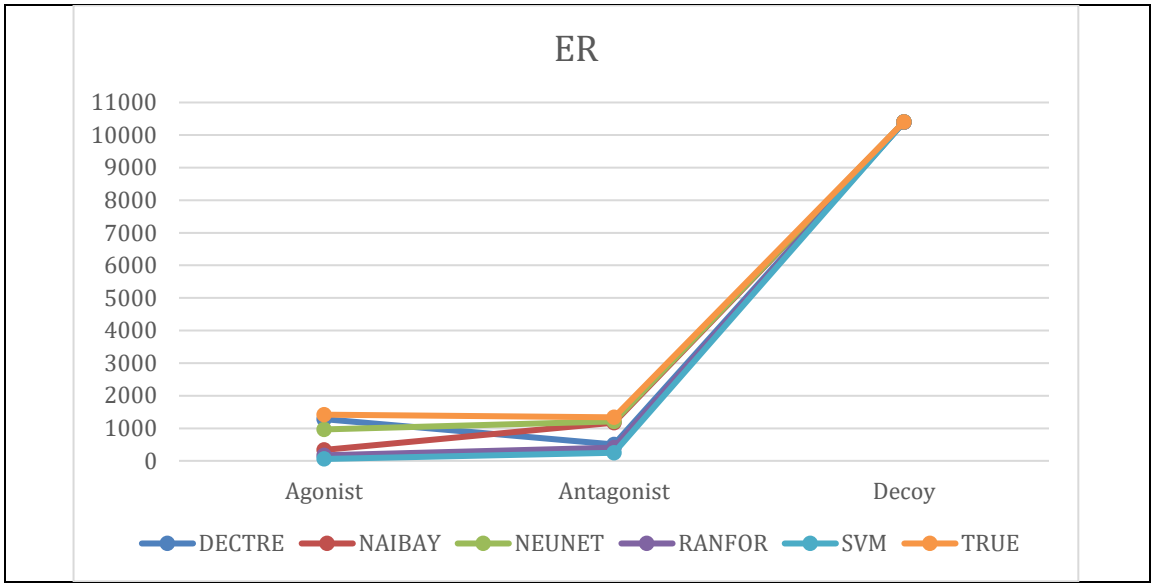


Figure 2.11: The performance of each model compared to the ground truth (TRUE) in classifying agonists, antagonists, and decoys for ER. All models had similar performance to TRUE in classifying decoys, while DECTRE, RANFOR, and SVM both had more FNs for the antagonist class. NAIBAY and NEUNET had similar performance to TRUE in regard to antagonist prediction, while DECTRE was overlapping with TRUE for agonist prediction.

Table 2.15: DECTRE. Run-time: 13 minutes 39 seconds. Performance: 92.61%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	1283	405	1	75.96%
Pred. Antag.	122	504	4	80.00%
Pred. Decoy	12	428	10400	95.94%
Recall	90.54%	37.70%	99.95%	

Table 2.16: NAIBAY. Run-time: 2 minutes 25 seconds. Performance: 90.51%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	336	55	0	85.93%
Pred. Antag.	1081	1176	7	51.94%
Pred. Decoy	0	106	10398	98.99%
Recall	23.71%	87.96%	99.93%	

Table 2.17: NEUNET. Run-time: 21 hours 38 minutes 12 seconds. Performance: 95.59%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	965	40	0	96.02%
Pred. Antag.	452	1213	4	72.68%
Pred. Decoy	0	84	10401	99.20%
Recall	68.10%	90.73%	99.96%	

Table 2.18: RANFOR. Run-time: 8 hours 1 minute 19 seconds. Performance: 83.43%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	171	36	0	82.61%
Pred. Antag.	1198	406	4	25.25%
Pred. Decoy	48	895	10401	91.69%
Recall	12.07%	30.37%	99.96%	

Table 2.19: SVM. Run-time: 15 days 12 hours 40 minutes 35 seconds. Performance: 81.39%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	58	5	0	92.06%
Pred. Antag.	2	250	3	98.04%
Pred. Decoy	1357	1082	10402	81.01%
Recall	04.09%	18.70%	99.97%	

2.3.3 Performance: GR

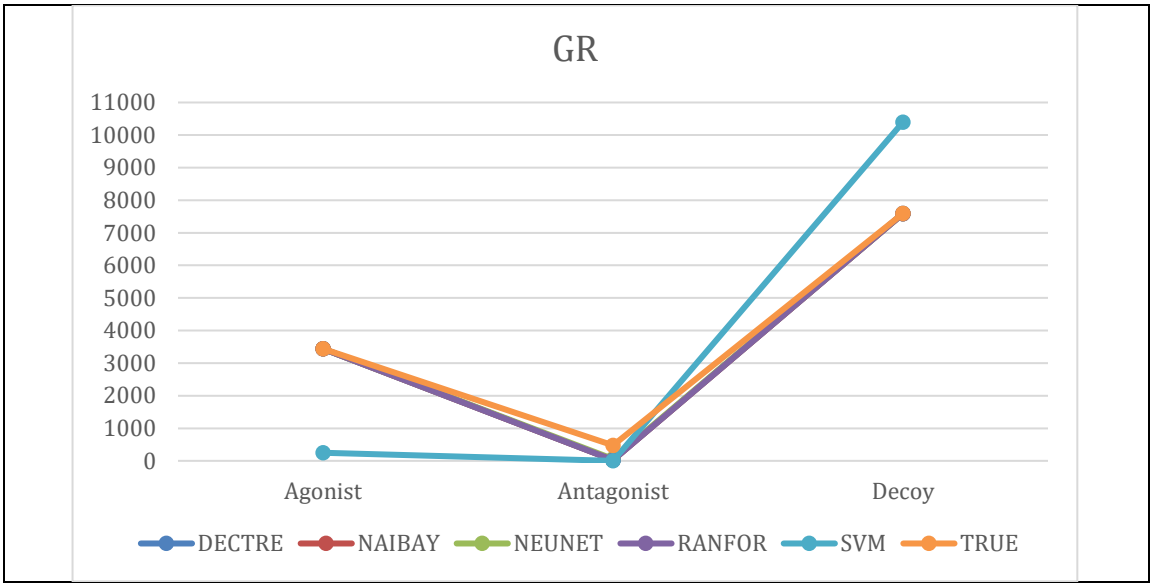


Figure 2.12: The performance of each model compared to the ground truth (TRUE) in classifying agonists, antagonists, and decoys for GR. DECTRE, NAIBAY, and NEUNET overlapped with TRUE for agonists and decoys, while all models were shown to have similar performance in antagonist prediction. SVM had more FNs for the agonist class, and inversely, more FPs for decoys.

Table 2.20: DECTRE. Run-time: 1 minute 35 seconds. Performance: 96.08%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	3437	16	0	99.54%
Pred. Antag.	0	26	0	100.00%
Pred. Decoy	0	435	7594	94.58%
Recall	100.00%	05.45%	100.00%	

Table 2.21: NAIBAY. Run-time: 1 minute 6 seconds. Performance: 96.02%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	3437	283	1	92.37%
Pred. Antag.	0	26	6	81.25%
Pred. Decoy	0	168	7587	97.83%
Recall	100.00%	05.45%	99.91%	

Table 2.22: NEUNET. Run-time: 8 hours 46 minutes 1 second. Performance: 96.32%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	3437	0	6	99.83%
Pred. Antag.	0	59	0	100.00%
Pred. Decoy	0	418	7588	94.78%
Recall	100.00%	12.37%	99.92%	

Table 2.23: RANFOR. Run-time: 4 hours 20 minutes 41 seconds. Performance: 96.03%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	3437	16	6	99.36%
Pred. Antag.	0	26	0	100.00%
Pred. Decoy	0	435	7588	94.58%
Recall	100.00%	05.45%	99.92%	

Table 2.24: SVM. Run-time: 13 days 22 hours 40 minutes 27 seconds. Performance: 80.93%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	252	0	6	97.67%
Pred. Antag.	0	0	1	00.00%
Pred. Decoy	1165	1337	10398	80.60%
Recall	17.78%	00.00%	99.93%	

2.3.4 Performance: PR

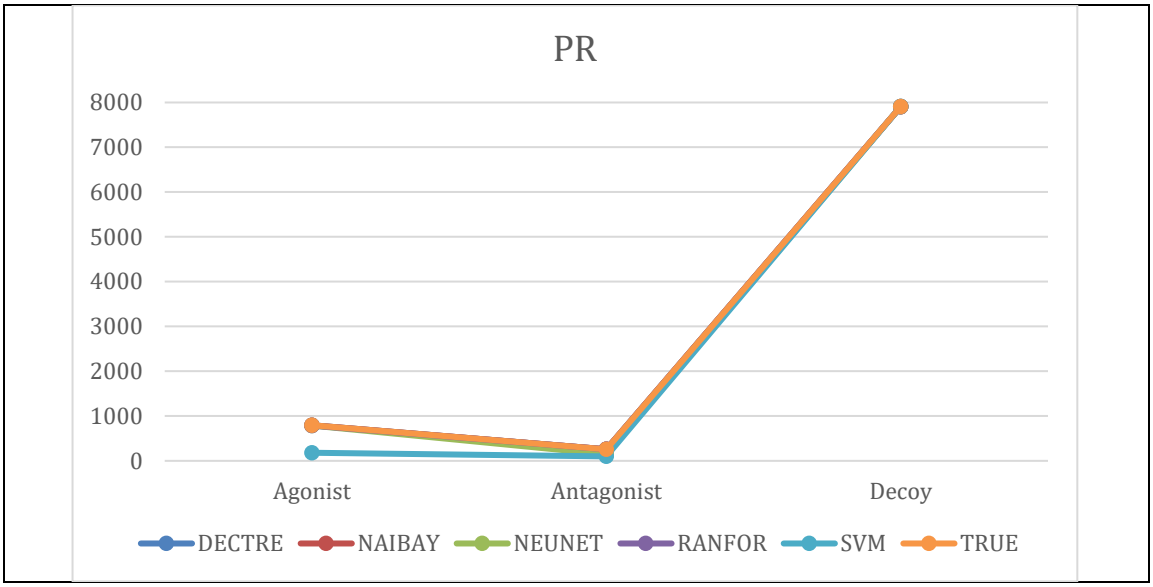


Figure 2.13: The performance of each model compared to the ground truth (TRUE) in classifying agonists, antagonists, and decoys for PR. Most of the models overlapped with TRUE on all classes, the exception being SVM with more FNs for agonists and antagonists.

Table 2.25: DECTRE. Run-time: 2 minutes 58 seconds. Performance: 99.92%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	793	0	0	100.00%
Pred. Antag.	6	258	1	97.36%
Pred. Decoy	0	0	7908	100.00%
Recall	99.25%	100.00%	99.99%	

Table 2.26: NAIBAY. Run-time: 1 minute. Performance: 99.91%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	795	0	3	99.62%
Pred. Antag.	0	258	1	99.61%
Pred. Decoy	4	0	7905	99.95%
Recall	99.50%	100.00%	99.95%	

Table 2.27: NEUNET. Run-time: 4 hours 14 minutes 59 seconds. Performance: 98.64%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	789	112	0	87.57%
Pred. Antag.	6	146	0	96.05%
Pred. Decoy	4	0	7909	99.95%
Recall	98.75%	56.59%	100.00%	

Table 2.28: RANFOR. Run-time: 1 hour 19 minutes 57 seconds. Performance: 99.87%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	789	0	1	99.87%
Pred. Antag.	6	258	1	97.36%
Pred. Decoy	4	0	7907	99.95%
Recall	98.75%	100.00%	99.97%	

Table 2.29: SVM. Run-time: 6 days 2 hours 55 minutes 33 seconds. Performance: 91.36%.

	True Agonist	True Antag.	True Decoy	Precision
Pred. Agonist	184	0	0	100.00%
Pred. Antag.	46	100	2	67.57%
Pred. Decoy	569	158	7907	91.58%
Recall	23.03%	38.76%	99.97%	

2.3.5 Summary

A summary of each learner's performance on every receptor is shown in Table 2.30 while Table 2.31 displays the total time it took each learner to complete training. In regard to classification performance, NEUNET was shown to be the most performant while SVM was least accurate in classifying agonists, antagonists, and decoys. For total run-time, the NAIBAY classifier was fastest while SVM was the slowest.

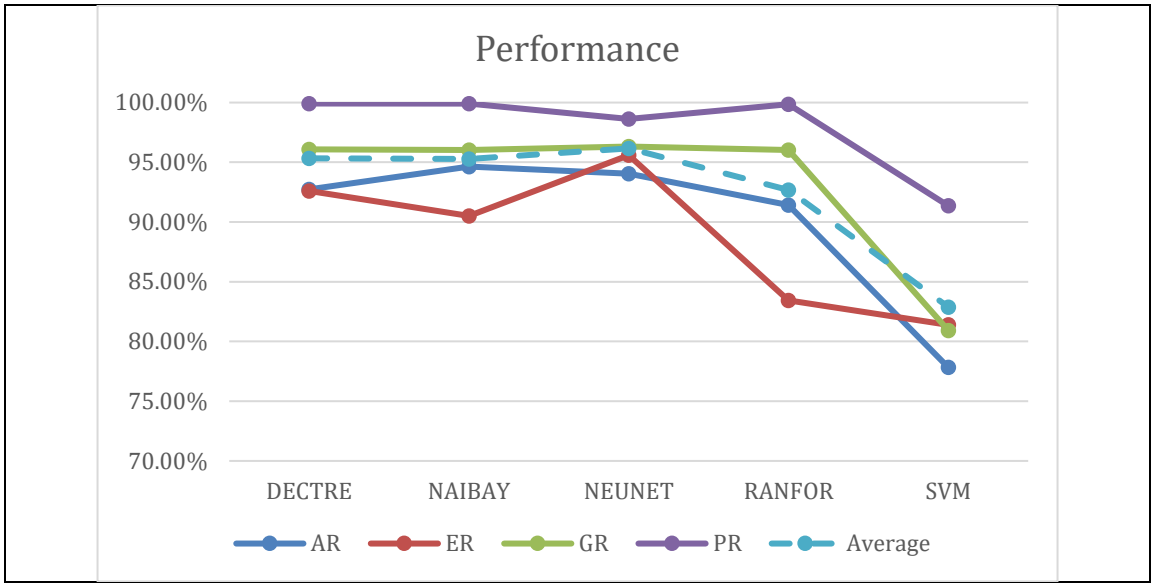


Figure 2.14: The performance of all models over all receptors, with the average performance of each model on AR, ER, GR, and PR shown in teal with dashed lines. All models performed best on PR while the worst performance of DECTRE, NAIBAY, and RANFOR were all on ER. Both NEUNET and SVM had their lowest performance on AR.

Table 2.30: Performance of all learners on all receptors.

	DECTRE	NAIBAY	NEUNET	RANFOR	SVM
AR	92.73%	94.64%	94.04%	91.41%	77.83%
ER	92.61%	90.51%	95.59%	83.43%	81.39%
GR	96.08%	96.02%	96.32%	96.03%	80.93%
PR	99.92%	99.91%	98.64%	99.87%	91.36%
Average	95.34%	95.27%	96.16%	92.69%	82.88%

Figure 2.15: The total training/run-time (in seconds) of the models, with and without the inclusion of SVM, over all receptors. The average time of each model on AR, ER, GR, and PR are shown in teal with dashed lines.

Table 2.31: Total run-time of all learners on all receptors. The format used is day:hour:minute:second.

	DECTRE	NAIBAY	NEUNET	RANFOR	SVM
AR	00:00:03:33	00:00:01:09	00:06:34:47	00:02:21:18	01:16:47:47
ER	00:00:13:39	00:00:02:25	00:21:38:12	00:08:01:19	15:12:40:35
GR	00:00:01:35	00:00:01:06	00:08:46:01	00:04:20:41	13:22:40:27
PR	00:00:02:58	00:00:01:00	00:04:14:59	00:01:19:57	06:02:55:33
Average	00:00:05:27	00:00:01:25	00:10:18:30	00:04:03:49	09:07:46:06

Table 2.32: Average class precision of the agonist, antagonist, and decoys classes for all learners on all receptors.

	Agonist	Antagonist	Decoy
AR	88.95%	59.80%	92.38%
ER	86.52%	65.58%	93.37%
GR	97.75%	76.25%	92.47%
PR	97.41%	91.59%	98.29%
Average	92.66%	73.31%	94.13%

Table 2.33: Average class recall of the agonist, antagonist, and decoys classes for all learners on all receptors.

	Agonist	Antagonist	Decoy
AR	74.21%	36.59%	99.98%
ER	39.70%	53.09%	99.95%
GR	83.56%	05.74%	99.94%
PR	83.86%	79.07%	99.98%
Average	70.33%	43.62%	99.96%

Table 2.34: Average class precision of all learners on all receptors.

	DECTRE	NAIBAY	NEUNET	RANFOR	SVM
AR	86.40%	90.10%	88.28%	78.17%	58.94%
ER	83.97%	78.95%	89.30%	66.52%	90.37%
GR	98.04%	90.48%	98.20%	97.98%	59.42%
PR	99.12%	99.73%	94.52%	99.06%	86.38%
Average	91.88%	89.82%	92.58%	85.43%	73.78%

Table 2.35: Average class recall of all learners on all receptors.

	DECTRE	NAIBAY	NEUNET	RANFOR	SVM
AR	75.43%	82.43%	79.28%	73.79%	39.37%
ER	76.06%	70.53%	86.26%	47.47%	40.92%
GR	68.48%	68.45%	70.76%	68.46%	39.24%
PR	99.75%	99.82%	85.11%	99.57%	53.92%
Average	79.93%	80.31%	80.35%	72.32%	43.36%

2.4 Discussion

Overall, NEUNET had the best average classification performance of the 5 learners, while RANFOR was second best, NAIBAY was third, DECTRE was fourth, and SVM was the worst. Using a 2-tailed paired samples t-test where any value with $p < 0.05$ was considered to be significant, the performance of NEUNET over all receptors was not statistically significant compared to RANFOR ($p = 0.33$), NAIBAY ($p = 0.58$), or DECTRE ($p = 0.43$). Similarly, there was no significant difference between DECTRE and NAIBAY ($p = 0.94$), DECTRE and RANFOR ($p = 0.31$), or NAIBAY and RANFOR ($p = 0.22$). Only the performance of SVM was found to be significant compared to DECTRE ($p = 0.00$), NAIBAY ($p = 0.01$), NEUNET ($p = 0.01$), and RANFOR ($p = 0.04$). In terms of class precision, a measure of the number of drugs classified as an agonist, antagonist, or decoy that were correctly identified as such, the antagonist class had the worst/lowest percentage. In other words, drugs classified as antagonists were more likely to be FPs compared to the other classes. In the AR validation set for

instance, approximately 40% of all drugs classified as antagonists were actually agonists or decoys; see Table 2.32. Class recall is the number of drugs correctly classified as an agonist, antagonist, or decoy, accounting for FNs. Regarding average class recall (Table 2.33), decoys had the highest percentage while the antagonists had the lowest, meaning 99.96% of all decoys over all receptors were correctly identified for instance. Furthermore, NEUNET was shown to have the highest average class recall, with DECTRE following, then NAIBAY, RANFOR in fourth, and SVM with the lowest. Examining each class individually, Table 2.32 and Table 2.33 show that all 5 learners had difficulty in predicting antagonists, which may be due to a number of factors; for one, there were less antagonists available for all receptors than both the agonist and antagonist classes, which may have prevented proper training. Additionally, the differences between agonists and antagonists may be very subtle or inconsistent, making differentiation between the 2 arduous. Regarding baseline, the accuracy of every classifier on all receptors were (often significantly) above the baseline percentages shown in Table 2.9, demonstrating the viability of this method in classifying the agonists, antagonists, and decoys of AR, ER, GR, and PR.

While Li et al. (2015) concluded that the SVM classifier was best suited for predicting agonists of LXR-beta, our results show that SVM did not provide any advantages over the other classifiers used; in fact, on ER and GR it required a vast amount of computational resources to train, an issue no other learner faced.

Moreover, Li et al. (2015) trained their models to discriminate between selective and non-selective agonists using a training set comprised of 234 entries with a validation set of 58, and a total of 962 features were calculated using both the MOE and PaDEL-Descriptor programs. In comparison, this chapter specified 3 classes and the training and validation datasets were kept as balanced as possible, with only MOE being used to calculate a total of 435 features for each agonist/antagonist/decoy. Additionally, Li et al. (2015) did not use CV for training, but rather generated a total of 324 models from the SVM, NAIBAY, recursive partitioning, and KNN learners, and then used another validation set of 76 selective and non-selective agonists to determine the top 10 best performing models. While the performance of the models produced by Li et al. (2015) was ≥90%, similar to the results in this study, the validation set in Li et al. (2015) was small compared to the hundreds of examples used here which provided a more comprehensive evaluation of classification performance. Feature selection was also another major difference, where the forward algorithm was used during each iteration of CV to identify the best set of features, but Li et al. (2015) kept features that were correlated with the selective ratio values of all LXR agonists, defined as the IC50 of a given agonist to LXR-alpha divided by the IC50 of the agonist to LXR-beta (Li et al., 2015). Although this method of feature selection is based on experimental values, it does require this data to either be available in published literature or from the experimenters themselves, which is not always the case.

Asako and Uesawa (2017) developed their own classifier based on the RANFOR learner to predict environmental pollutants that act as ER agonists and affect the endocrine system. The training set was comprised of 8,733 chemicals, which was split 50:50 into training and validation datasets (Asako & Uesawa, 2017). Similar to Li et al., Asako and Uesawa's (2017) method was to generate numerous models and evaluate their performance using a final validation set, made up of 599 chemicals in this case, to identify the best ones. There was a total of 4,071 features calculated using the Dragon, MOE, and Marvin applications, and the most important ones were determined using a combination of each feature's split ranking, or the number of times this feature was used when partitioning the DECTREs that make up the RANFOR, as well as the chi-squared value of the feature's likelihood ratio, otherwise known as a G2 value (Asako & Uesawa, 2017). Overall, Asako and Uesawa's (2017) final model achieved a ROC-AUC value of 87% by generating a total of 1,050 models and then identifying the most performant one through a series of evaluations. In this chapter, RANFOR had an average classification accuracy of 83.43% for ER, its lowest percentage over all receptors; see Table 2.18.

Russo et al. (2018) utilized Bernoulli NAIBAY, AdaBoost DECTRE, RANFOR, SVM, and deep NEUNET learners to also identify endocrine disrupting ER agonists. Russo et al.'s training set was comprised of 24,305 compounds while the validation set was made up of 227 compounds, and a total of 197 features

were used during training, which included descriptors such as number of carbons, number of aromatic rings, etc. as well as a molecular fingerprint from the ChemoTyper program. Russo et al. (2018) used 5-fold CV for training each learner, and the best performing learner was RANFOR, followed by deep NEUNET, SVM, Bernoulli NAIBAY, and then AdaBoost DECTRE (Russo et al., 2018). In comparison, the results from this chapter showed that the rank of best to worst classifier for ER was NEUNET, DECTRE, NAIBAY, RANFOR, and then SVM. The difference in rankings may be due to a few factors; for one, Russo et al. (2018) focused specifically on endocrine disrupting agonists only, while the goal of this chapter was to classify all types of agonists. Russo et al. also used multiple training sets from different sources such as Tox21, ChEMBL, and CERAPP, where separate models were generated for each dataset, which varied in terms of ratio of active to inactive agonists among other things. Moreover, Russo et al. (2018) used a larger training dataset with a validation set that was about 0.93% its size, whereas in this chapter, the training and validation datasets were kept balanced.

The datasets from both Asako and Uesawa (2017) and Russo et al. (2018) were comprised of mostly or entirely endocrine disrupting ER agonists, a subset of all ER agonists involving toxic chemicals such as environmental pollutants. These many thousands of agonists were left out of this chapter to ensure the trained models were able to identify all types of agonists rather than skewing towards toxic ones. In regard to feature selection, forward selection is a greedy

method, meaning it only kept the best k feature subset(s) from previous iterations (Reif & Shafait, 2014), where $k = 1$ in this chapter. While only keeping the single best subset from each iteration sped up computation time, a disadvantage is that a better feature subset may have been overlooked (Reif & Shafait, 2014). Moreover, the 435 features included metrics such as HOMO and LUMO energies, which describe the highest occupied molecular orbital and the lowest occupied molecular order respectively, whose values such as -6.5 or -5.4 may not seem meaningful at first glance. That being said, HOMO and LUMO energies can in fact be used to compare and contrast agonists, antagonists, and decoys because the same algorithm was used to calculate the values, allowing drug X's HOMO energy of -6.5 to be compared to drug Y's HOMO energy of -5.4 for instance. A truism within the field is “garbage in, garbage out”, referring to the fact a ML model can only ever be as good as the data used to train it. While there are numerous databases and resources on pharmaceuticals such as DrugBank or Therapeutic Target DB for instance, data augmentation in the form of conformation generation was required since only a small number of druggable agonists and antagonists were found for AR, ER, GR, and PR. The biggest limitation of this study was that a significant portion of the agonist and antagonist classes were required to be generated *in-silico* instead of being obtained from experimental results within literature. Nevertheless, even if 2 or more conformations were detected as the same drug it would not have contaminated the performance assessment since the

training and validation datasets were completely separated before data augmentation was performed. In other words, all potential duplicates for a given drug would either exist in the training or validation dataset, but not in both at the same time.

2.5 Conclusion

In this chapter, the area of drug-action prediction was explored, where the DECTRE, NAIBAY, NEUNET, RANFOR, and SVM supervised ML models were trained to classify compounds as either an agonist, antagonist, or decoy for each of the AR, ER, GR, and PR proteins. The main objective was to determine a given drug's behavior towards a receptor, which has uses in polypharmacology such as prediction of side effects for a given drug therapy as well as identifying targets for drug repurposing. Additionally, drug-action prediction allows for easier filtering of drug libraries for computational simulations such as drug docking and MD. The secondary objective was to compare the performance of all 5 models for each receptor to determine the one best suited for this task. The results showed that all models were able to exceed the baseline accuracies for every learner on each receptor, demonstrating that these models performed better than simply placing all drugs into the largest class. Additionally, no statistical difference was found between any of the top 3 models: NEUNET and DECTRE ($p = 0.43$), NEUNET and NAIBAY ($p = 0.58$), and NAIBAY and DECTRE ($p = 0.94$), meaning none of

these models were significantly different to one another in classifying agonist, antagonist, and decoy drugs. That being said, the training time for both DECTRE and NAIBAY was measured in minutes for all receptors, far less compared to NEUNET, which took approximately 4 hours (on PR) at the least and 21.5 hours (on ER) at most.

Future work will include improving the accuracy of classifying antagonist compounds, which was found to be generally poor in comparison to the agonists and decoys. This can be accomplished by obtaining more antagonist compounds using natural language processing to automatically review published literature as well as generating more features for the compounds, which may provide enough information to the given classifiers to properly differentiate antagonists from the other classes. The paradigm shift from traditional *in-silico* tools to ML allows researchers to utilize the vast amounts of biomedical data available to accomplish tasks such as drug-action prediction that was performed in this chapter. Overall, the considerable benefit of having trained ML models is that any given drug can be classified within seconds as an agonist, antagonist, or decoy of AR, ER, GR, or PR without needing to perform further modeling (e.g., QSAR) or use costly and/or computationally intensive software.

The next chapter also involves the usage of ML, this time to investigate bioelectric signaling of cells. This work was based on data generated by BETSE, an application that utilizes matrix-based differential equations to model the

connections between cells and ion channel activity among other cellular processes. While the ML models generated in this chapter were independently created as standalone classifiers, the regression-based ML models in the next chapter were trained to replace the main functionalities of BETSE, allowing predictions to be made more efficiently as well as on larger cell networks without requiring the use of a supercomputer or cluster.

Chapter 3: Replacing Bioelectric Dynamics Modeling using Regression-based Machine Learning

3.1 Introduction

Previously, *in-silico* tools such as MD/SA, used to obtain the lowest energy conformation of CSA, and drug docking, a process that simulates the binding of a ligand to a target, were used to optimize PALs for the purpose of targeted drug delivery. These types of methodologies have been around for decades and are an essential part of the current drug discovery process. However, recent computational methods have moved towards ML in order to take advantage of the vast amounts of biomedical data available, for example, training classifiers to determine whether a drug is an agonist, antagonist, or decoy to each of 4 receptors as shown in the preceding chapter. That all being said, there are few traditional or modern tools that focus on modeling bioelectricity, which is used by cells to communicate with one another and is involved with numerous processes including cell proliferation, programmed cell death or apoptosis, and tumor suppression (Pietak & Levin, 2016; Srivastava et al., 2021). Bioelectrical dynamics of non-neuronal cells can be explored *in-silico* using the BETSE program (Pietak & Levin, 2016), which models gap junction (GJ) and ion channel activity of all cells

within a network to predict the cytosolic and extracellular concentrations of sodium (Na^+), potassium (K^+), chloride (Cl^-) and calcium (Ca^{2+}) ions, as well as each cell's transmembrane potential (V_{mem}), or the difference in electrical potential between the cytosol and extracellular medium (Pietak & Levin, 2016; Pietak & Levin, 2017). V_{mem} is used by cells to form networks or circuits with one another, where they can regulate stem cell differentiation, the development and repair of organs and limbs, and the growth of tumors, as well as accomplish tasks such as the opening and closing voltage-gated ion channels (Pietak & Levin, 2016; Silver & Nelson, 2018). The V_{mem} of a given cell, measured in millivolts (mV), affects, and is in turn affected by, the ion concentration. For instance, if a cell contains more negative chloride ions (Cl^-) than positive sodium ions (Na^+), the V_{mem} will be more polarized and vice versa. Furthermore, V_{mem} plays a role in diseases such as cancer, where studies have found that tumor cells are generally more depolarized compared to healthy ones, and not only that, but *in-vitro* experiments have demonstrated that returning these depolarized cancer cells back to a healthy V_{mem} can prevent and even reverse tumorigenesis in some cases (Srivastava et al., 2021; Tuszynski et al., 2017).

Bioelectric medicine is currently United States Food and Drug Administration approved for treatment of Parkinson's disease, epilepsy, chronic pain, and depression. For these cases an electrode is surgically implanted into the target area within a patient, such as the brain, spinal cord, or on a nerve, and it is

programmed to send a specific pattern of electrical pulses to stimulate the target for treatment (Lee et al., 2020). As more research is being done in bioelectricity and cellular networks in general, *in-silico* methods must also keep pace. On that front, BETSE has been validated to be able to predict both the V_{mem} and intracellular ion concentrations of Xenopus oocytes with only <10% difference from experimental values using a simulation consisting of 35 cells in total (Pietak & Levin, 2016), and furthermore, the ability of BETSE to model ion channel activity can be leveraged for drug discovery as well. Ion channels are remodeled in cancer cells to enhance the cancer's ability to proliferate; in both breast and prostate cancers the Ca^{2+} and Na^{+} channels are upregulated, while in thyroid cancer K^{+} channels are downregulated (Haworth & Brackenbury, 2019; Peters et al., 2017; Wang et al., 2018). The membrane diffusion constants as well as initial extracellular and intracellular concentrations of Na^{+}, K^{+}, Cl^{-}, and Ca^{2+} can be modified for a given BETSE simulation, allowing researchers to examine how different values affect the overall ion concentration and V_{mem} of the network at various times.

3.1.1 BETSE

BETSE takes an input configuration file containing parameters such as initial extracellular and intracellular Na^{+}, K^{+}, Cl^{-}, and Ca^{2+} concentrations as well as diffusion constants, total simulation time, temperature and pressure of the network, and so on. The output provides the average V_{mem} of the entire network

as well as each cell at every timestep (e.g., 1 second, 2 seconds, etc.), including the average ion concentrations of Na^+, K^+, Cl^-, and Ca^{2+} for the entire network. To model the bioelectric signaling of a cellular network, BETSE generates an irregular Voronoi diagram-based cell grid and places it on top of the environmental grid, made up of evenly spaced homogenous square cells (Pietak & Levin, 2016). See Figure 3.1 below.

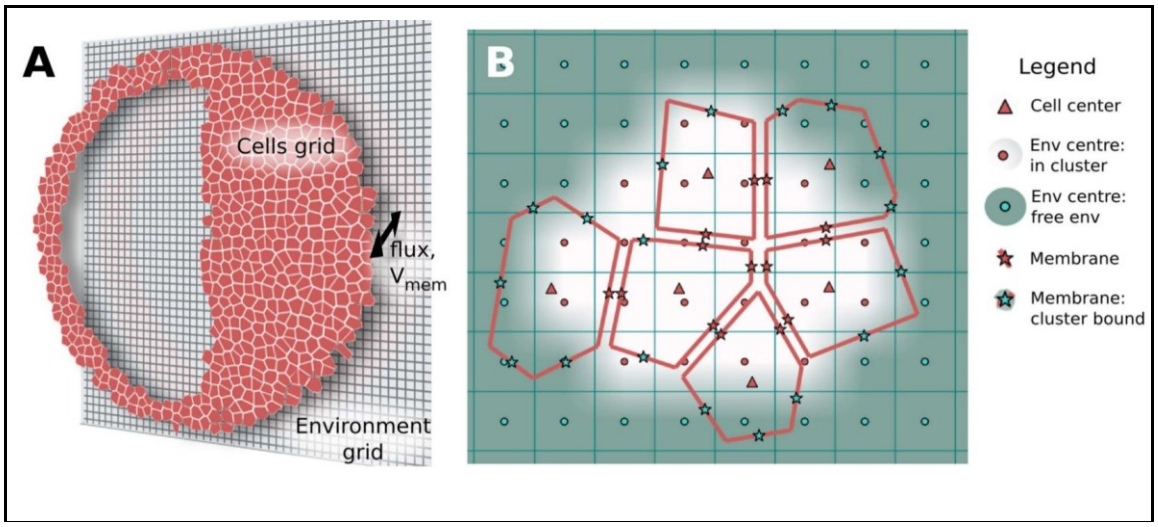
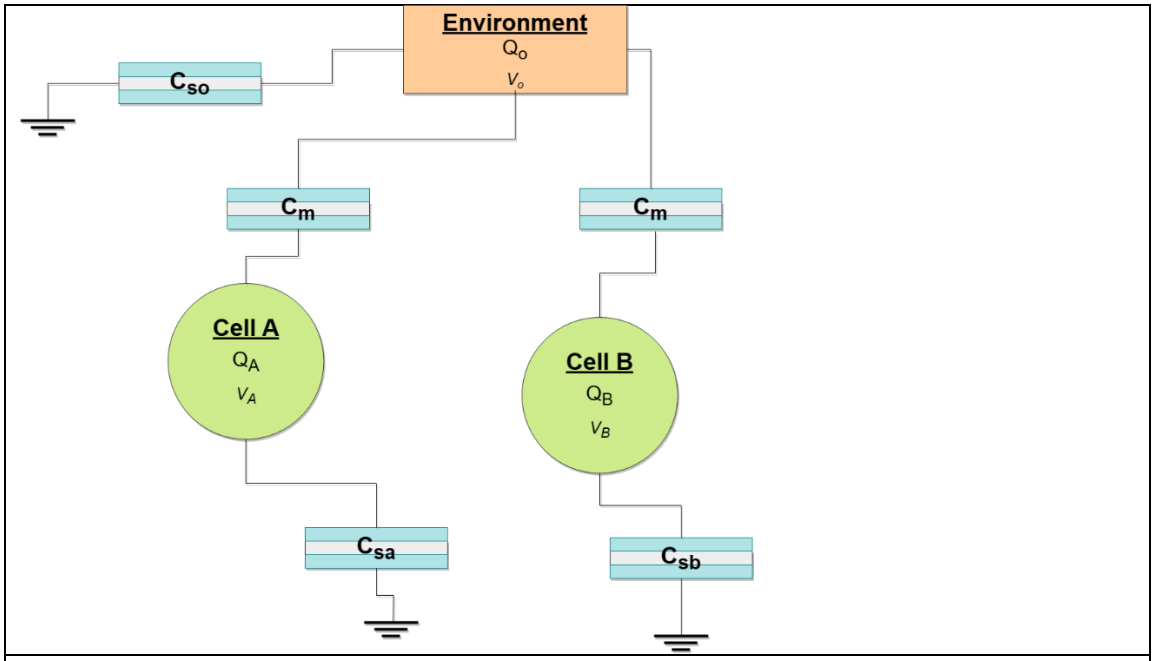


Figure 3.1: A) BETSE generates an irregular Voronoi diagram-based cell grid to place cells in while the environment grid underneath is represented by evenly spaced homogeneous squares (Pietak & Levin, 2016). The size of the environment grid, specified in the input configuration file, determines the total number of cells. B) The "Cell center" within each cell, represented by a red ▲, is where properties such as ion concentration and intracellular voltage are defined. Additionally, the membrane perimeter for each cell is divided into segments, where the midpoint for each membrane segment (represented by a blue ★) is where V_{mem} is calculated. This image was obtained without modification from Figure 2 in Pietak & Levin (2016) which was published under the terms of the Creative Commons Attribution License (CC-BY); https://creativecommons.org/licenses/by/4.0.

The center point of each cell on the cell grid, and also of each square on the environmental grid (see Figure 3.1B), are where differential equations are applied and solved during a simulation to calculate scalar properties that are defined only by their magnitude such as ion concentration, charge, and voltage, as well as vector properties that are defined by both their magnitude and direction such as electric field (e.g., across a GJ) and mass flux (e.g., ion flux) between neighboring cells for instance (Pietak & Levin, 2016). Every cell on the cell grid also has its own volume as well as perimeter, which represents the cell membrane that can further be separated into individual membrane segments, each having their own properties such as V_{mem}. Moreover, each of these cell membrane segments are connected to a neighboring cell membrane segment via a GJ. The ion concentration, intracellular charge, and intracellular voltage properties are defined on both the cell center points and midpoints of cell membrane segments on the cell grid, and also on the center points of each square on the environmental grid (Pietak & Levin, 2016). Other properties including V_{mem}, ion flux, and electric field are defined on cell membrane midpoints on the cell grid as well as on the midpoints between the center points of 2 neighboring squares on the environmental grid (Pietak & Levin, 2016).



$$Q_A = C_m(V_A - V_o) + C_{sa}V_A$$

$$Q_B = C_m(V_B - V_o) + C_{sb}V_B$$

$$Q_o = C_m(V_o - V_A) + (V_o - V_B) + C_{so}V_o$$

Figure 3.2: Diagram of a 2-cell network adopted from Figure 4C of Pietak & Levin (2016) that models the electrical system used to calculate V_{mem} between cells in BETSE, where $V_{mem} = V_{intra} - V_{extra}$. Q_A and Q_B represent the total ionic charges of their respective cells, which is calculated in part using the inner (V_A, V_B) and environmental (V_o) voltages, corresponding to V_{intra} and V_{extra}. C_m represents the capacitance, which determines how much charge can be stored (Kaiser, 1992/2012), of the cellular membranes between each cell and the extracellular environment. Finally, the self-capacitance, defined as the amount of charge stored on each cell and the environment per applied unit voltage, is represented by C_{sa}, C_{sb}, and C_{so} respectively (Pietak & Levin, 2016).

BETSE uses the Nernst-Planck equation to estimate the concentration of ions transporting through GJs while ion pumps are modeled using Michaelis-Menten enzyme kinetic relations. Additionally, V_{mem} is calculated using the

Maxwell Capacitance Matrix (Pietak & Levin, 2016); see Figure 3.2 and Equation 3.1.

Equation 3.1: The Maxwell Capacitance matrix is used to relate the set of net ionic charges (Q) and their corresponding voltages (V). A) If the voltages (V_A, V_B, V_o) are known, then charges are calculated using its product with the Maxwell Capacitance matrix, which includes the self-capacitances of each cell (C_{sa}, C_{sb}) and the environment (C_{so}), as well as their membranes (C_m). B) This also works vice versa, where the voltages are calculated using the product of the inverse Maxwell Capacitance matrix and the net cellular and environmental ionic charges. All equations were obtained from Pietak & Levin (2016), Equation 19.

A)	$$\begin{bmatrix} Q_A \\ Q_B \\ Q_o \end{bmatrix} = \begin{bmatrix} C_m + C_{sa} & 0 & -C_m \\ 0 & C_m + C_{sb} & -C_m \\ -C_m & -C_m & 2C_m + C_{so} \end{bmatrix} \begin{bmatrix} V_A \\ V_B \\ V_o \end{bmatrix}$$ $$\bar{Q} = M\bar{V}$$
B)	$$\bar{V} = M_{\text{inv}}\bar{Q}$$

The gradient, divergence, and the Laplace differential operators are defined by BETSE on both the environment and cell grid to calculate the scalar and vector properties. The gradient operator is used to calculate the change of scalar properties over a given space, such as the Ca^{2+} concentration between neighboring cells, and on the cell grid there are 3 different types: 1) cell to environment, 2) cell to cell, and 3) intracellular (Pietak & Levin, 2016). First, trans-membrane gradients involve scalar properties (ion concentration, voltage, etc.) that are calculated using a nearest-neighbor interpolation scheme where

midpoints of each cell membrane segment interface with the center points of squares on the environmental grid (Pietak & Levin, 2016). These types of gradients are used to exchange information between a cell and its local environment, where a weighing function is used to assign the mole transfer between cell and environmental grids for a given mass flux (Pietak & Levin, 2016). See Equation 3.2.

Equation 3.2: Trans-membrane gradient for a scalar property s (e.g., ion concentration) defined on the environmental point j and cell membrane midpoint k with membrane thickness d_{mem} on the cell grid. This equation was obtained from Pietak & Levin (2016), Supplementary Equation 4.

$$\nabla s_{jk} = \frac{(s_j - s_k)}{d_{mem}}$$

Second, membrane midpoints are also used for inter-cellular gradients on the cell grid, allowing BETSE to calculate how scalar properties change between neighboring cells, see Equation 3.3.

Equation 3.3: Inter-cellular gradient for a scalar property s between the center point a of cell x and center point b on neighboring cell y that are separated by distance d_{ab} on the cell grid. Both t_{gj_x} and t_{gj_y} represent the tangent vectors of the gap junctions between the respective cells. These equations were obtained from Pietak & Levin (2016), Supplementary Equations 2 and 3.

$$F_{ab} = \frac{(s_b - s_a)}{d_{ab}}$$

$$\nabla s_{ab} = F_{ab} t_{gj_x} + F_{ab} t_{gj_y}$$

Finally, intra-membrane gradients calculate the lateral change of a scalar property on the cell grid, the Na^+ ion channel concentration for example, between points on an individual cellular membrane (Pietak & Levin, 2016), see Equation 3.4.

Equation 3.4: Intra-membrane gradient for a scalar property s between vertices p and q on an individual cell that are separated by distance d_{pq} on the cell grid. This equation was obtained from Pietak & Levin (2016), Supplementary Equation 5.

$$\nabla s_{pq} = \frac{(s_q - s_p)}{d_{pq}}$$

On the environmental grid, x-axis and y-axis gradients were calculated using the central difference formula to estimate first derivatives in space for points defined on the jth row and the kth column on the grid with uniform spacing, d_{grid}, between each square (Pietak & Levin, 2016); Equation 3.5.

Equation 3.5: X-axis and y-axis gradients on the environmental grid with points defined on the j^{th} row and k^{th} column on a grid with d_{grid} spacing. This equation was obtained from Pietak & Levin (2016), Supplementary Equations 10 and 11.

$$\frac{\delta s(x_j, y_i)}{\delta x} \Rightarrow \frac{s(x_{j+1}, y_i) - s(x_{j-1}, y_i)}{2d_{grid}}$$

$$\frac{\delta s(x, y)}{\delta y} \Rightarrow \frac{s(x_j, y_{i+1}) - s(x_j, y_{i-1})}{2d_{grid}}$$

The divergence operator measures the amount of outflow flow of a vector from every point over space, such as the flux of K^+ ions from a cell for instance; See Equation 3.6, which is similar to the equation used to calculate the divergence on the environmental grid as well (Pietak & Levin, 2016).

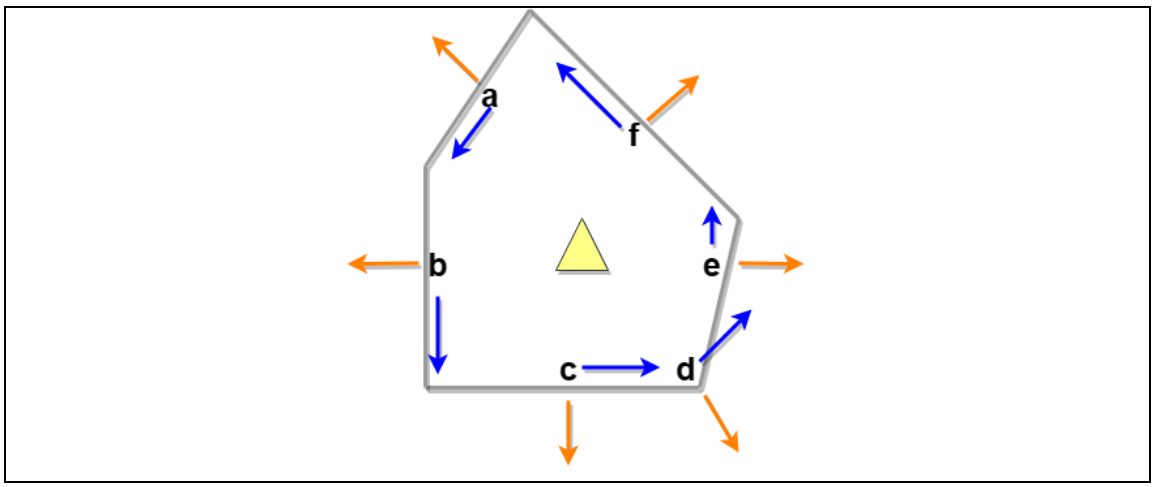


Figure 3.3: The normal and tangent vectors of each membrane segment (a-f) for a given cell. Normal unit vectors are represented by the orange arrows while the tangent vectors are shown using blue arrows. The cell midpoint is represented by the yellow triangle. This figure was based on supplementary Figures 3A and 3D from Pietak & Levin (2016).

Equation 3.6: Divergence of a given vector property P for cell c on the cell grid. The property (P) defined at the midpoint (m; P_m) on a given cell membrane segment is multiplied with the unit vector of P normal to the membrane, n_{mem_m}, and the cell membrane's surface area, σ_{mem_m}, to obtain the net flux for that membrane segment. Next, the net flux of every membrane segment is summed, and the result is divided by the cell volume, vol_{cell_c}. This equation was adapted from Pietak & Levin (2016), Supplementary Equation 6.

$$\nabla \times P_c = \frac{\Sigma(P_m \times n_{mem_m} \sigma_{mem_m})}{vol_{cell_c}}$$

The Laplacian is the divergence of a given scalar property, or in other words, the outward flow of ion concentration or voltage for instance to other points on the environment and cell grids; see Figure 3.4, Equation 3.7, and Equation 3.8.

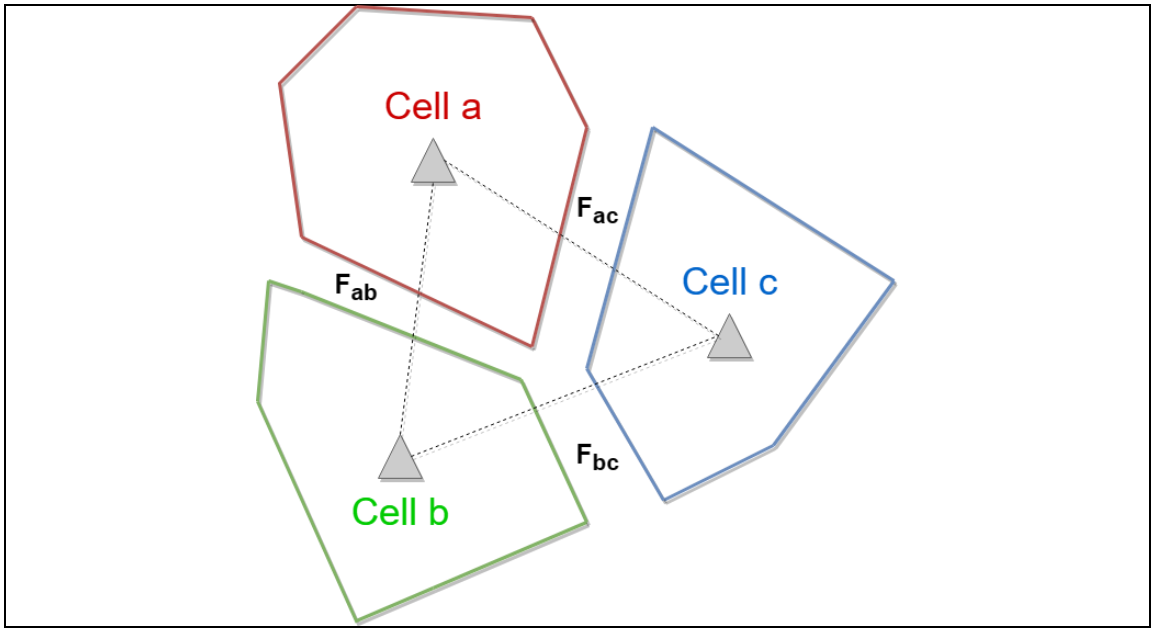


Figure 3.4: A group of 3 cells used to describe the Laplacian operator, adopted from supplementary Figure 4 of Pietak & Levin (2016). There are a total of 3 gradient fluxes: 1) F_{ab} between cell a and cell b, 2) F_{ac} between cell a and cell c, and 3) F_{bc} between cell b and cell c.

Equation 3.7: The Laplacian of a scalar property s at cell a from Figure 3.4 for example, begins with calculating the gradient fluxes F_{ab} and F_{ac} between neighboring cells b and c respectively. This involves the normal unit vectors of each flux from their corresponding cell to the neighboring one (e.g., $n_{x_{ab}}, n_{y_{ac}}$) as well as the surface area between the shared membrane between cell a and cell b (σ_{ab}), and cell a and cell c (σ_{ac}). These equations were obtained from Pietak & Levin (2016), Supplementary Equation 7.

$$F_{ab} \cdot n_{ab}\sigma_{ab} = (\frac{s_b - s_a}{d_{ab}})\sigma_{ab}n_{x_{ab}} + (\frac{s_b - s_a}{d_{ab}})\sigma_{ab}n_{y_{ab}})$$
$$F_{ac} \cdot n_{ac}\sigma_{ac} = (\frac{s_c - s_a}{d_{ac}})\sigma_{ac}n_{x_{ac}} + (\frac{s_c - s_a}{d_{ac}})\sigma_{ac}n_{y_{ac}})$$

Equation 3.8: Divergence of scalar property s at cell a on the cell grid is calculated by summing the components obtained from Equation 3.7 and dividing the result by vol_a, the volume of cell a. These equations were obtained from Pietak & Levin (2016), Supplementary Equation 8.

$$\nabla^2 s_a = As_a + Bs_b + Cs_c$$
$$A = \frac{-2\sigma_{ab}(n_{x_{ab}} + n_{y_{ab}})\, d_{ac} - 2\sigma_{ac}(n_{x_{ac}} + n_{y_{ac}})\, d_{ab}}{d_{ac}d_{ab}vol_a}$$
$$B = \frac{\sigma_{ab}(n_{x_{ab}} + n_{y_{ab}})}{d_{ab}vol_a}$$
$$C = \frac{\sigma_{ac}(n_{x_{ac}} + n_{y_{ac}})}{d_{ac}vol_a}$$

For the environmental grid, BETSE calculates the Laplacian in a similar manner to the process for the cell grid described above (Pietak & Levin, 2016), see Equation 3.9.

Equation 3.9: The equation BETSE uses to calculate the Laplacian of a scalar property s defined on the midpoint (i, j) of a given square on the environmental grid with d_{grid} length spacing between all squares. Defined in Pietak & Levin (2016), Supplementary Equation 12.

$$\nabla^2 s(x,y) = \left(\frac{s_{i+1,j} - s_{ij}}{d_{grid}}\right) + \left(\frac{s_{i-1,j} - s_{ij}}{d_{grid}}\right) + \left(\frac{s_{i,j+1} - s_{ij}}{d_{grid}}\right) + \left(\frac{s_{i,j-1} - s_{ij}}{d_{grid}}\right)$$

Overall, BETSE utilizes matrix-based differential equations to calculate V_{mem} and various other equations to determine scalar (e.g., ion concentration, voltage) and vector (e.g., mass flux and electric field) properties during a simulation. These methods can require a considerable amount of computational resources depending on parameters such as the size of the environment grid, specified simulation time, timestep, and so on, all of which limit the network that can be modeled.

3.1.2 Objectives

Due to the computational resources required for BETSE, especially with larger simulations, the goal of this study was to develop ML models that replace the main functions of BETSE. This was accomplished through 3 objectives:

1) Predict the V_{mem} of the entire cellular network at a given time.
2) Predict the V_{mem} of each individual cell within the cellular network at a given time.

3) Predict the ion concentrations of Na^+, K^+, Cl^-, and Ca^{2+} of the entire cellular network at a given time.

For all objectives the goal was to be able to predict the target value(s) at a particular time within the BETSE simulation, measured in seconds. More specifically, the goal of objective 1 was to predict the average V_{mem} of the entire cellular network, objective 2 focused on predicting the V_{mem} of each individual cell, and objective 3 aimed to predict average ion concentrations of the entire network, which has applications in drug discovery of ion agonists and antagonists for instance. Supervised learning was performed using training and validation datasets comprised of output from BETSE, with performance being assessed using multiple performance metrics: coefficient of determination (R^2), mean square error (MSE), mean absolute error (MAE), and median absolute error (MEDAE). Ideally, training and validation would include only experimental *in-vitro* or *in-vivo* data stemming from cell networks in a variety of conditions including up/down-regulated levels of $Na^+/K^+/Cl^-/Ca^{2+}$, differing number of cells in each experiment, and assorted membrane diffusion constants of $Na^+/K^+/Cl^-/Ca^{2+}$, but this was not available in published literature.

3.1.3 Related Work

Manicka and Levin (2019) developed BioElectric Network (BEN), an *in-silico* application used to simulate bioelectric signaling and functions as a minimal

version of BETSE. BEN models bioelectric communication between non-neuronal cells, providing a better understanding of basal cognition and synthetic biology as well as aiding in the development of regenerative medicine (Manicka & Levin, 2019). As an overview, BEN utilizes a layered approach where cells in each layer have roles; the first and last layers are the input and output layers respectively, while each of the middle layers perform a variety of functions such as the sensory layer that obtains signals directly from the environmental grid and sends it to the inter-neuron layers (Manicka & Levin, 2019). BEN models a variety of biological entities, including GJs that connect any 2 cells, electrodiffusion, a Na-K pump, and the Na^+, K^+, and Cl^- ions. Similar to BETSE, cells are placed on top of an environmental grid where electrodiffusion is calculated via the Nernst-Planck equation while transmembrane ion flux is calculated using the Goldman-Hodgkin-Katz equation (Manicka & Levin, 2019). Furthermore, chemical-gated ion channels are modeled using a sigmoid function that maps the concentration of the relevant signaling molecule to the maximum permeability of its corresponding ion channel. BEN has 2 types of learnable parameters: weight and bias. Weight regulates the effect V_{mem} has on the network as well as determines the voltage-gating of GJs, where lower values diminish V_{mem} and signal concentration within a cell while positive values increase them. Bias is responsible for regulating signal flux, specifically determining the threshold where signal concentration switches direction of change (Manicka & Levin, 2019). Unfortunately, *in-vitro* validation was

unable to be performed because there are no current methods in which generic cells can be used to quantify how non-neural tissues process information in the context of basal cognition (Manicka & Levin, 2019).

3.2 Methods

As an overview, BETSE simulations with randomized parameters were first run to generate data for the training and validation datasets. Next, the scikit-learn Python library (Pedregosa et al., 2011) was then used to train the Bayesian ridge (BAYRID), DECTRE, KNN, linear regression (LINREG), multi-level perceptron (MLP), RANFOR, and support vector regression (SVR) learners. Additionally, the mlens Python module (Flennerhag, 2018) was used to implement the super learner (SUPLRN) as defined by van der Laan et al. (2007), which is an ensemble method that incorporates various base learners to perform regression. Finally, the performance of each model on the validation dataset was estimated using the MAE, MEDAE, MSE, and R^2 metrics.

3.2.1 BETSE

The parameters of a given BETSE simulation are defined using a YAML formatted configuration file, which included total initialization and simulation time, Na^+, K^+, Cl^-, and Ca^{2+} ion concentrations, diffusion constants, environment grid size, timestep, and so on. To build models that can perform in a variety of conditions,

Python scripts were first written to generate configuration files with randomized values for the parameters shown in Table 3.1. Note that the total simulation time was set to 100 seconds for every configuration file, ensuring all simulations were simulated for the same amount time to provide an equivalent amount of data. Additionally, BETSE performed an initialization run before the actual simulation phase to generate the cell cluster and bring the network to cell resting V_{mem}.

Next, these configuration files were used to run BETSE simulations on the Compute Canada supercomputing platform, where a total of 14,891 BETSE runs were used for objective 1 to predict the V_{mem} of an entire cellular network. For objectives 2 and 3, a total of 26,924 BETSE simulations were run to provide additional data. There were 2 types of output files BETSE produced: 1) the "ExportedData.csv" file provided the average V_{mem} of the entire cellular network at each time within the simulation (e.g., 1 second, 2 seconds, etc.) as well as the average ion concentrations for Na^+, Ca^{2+}, Cl^-, and K^+ within each cell among other parameters such as membrane permeabilities, and 2) the "Vmem2D_TextExport.csv" files provided the x and y coordinates of each cell within the network along with their V_{mem} at every given time point.

Table 3.1: List of all BETSE configuration options that were randomized.

Option	Description
comp grid size	Size of the environment grid
cytosolic Na+ concentration	Intracellular sodium ion concentration
cytosolic K+ concentration	Intracellular potassium ion concentration
cytosolic Cl- concentration	Intracellular chloride ion concentration
cytosolic Ca2+ concentration	Intracellular calcium ion concentration
Dm_Na	Transmembrane diffusion constant for the sodium ion
Dm_K	Transmembrane diffusion constant for the potassium ion
Dm_Cl	Transmembrane diffusion constant for the chloride ion
Dm_Ca	Transmembrane diffusion constant for the calcium ion
extracellular Na+ concentration	Extracellular sodium ion concentration
extracellular K+ concentration	Extracellular potassium ion concentration
extracellular Cl- concentration	Extracellular chloride ion concentration
extracellular Ca2+ concentration	Extracellular calcium ion concentration
time step	Time step for the initialization phase
time step	Time step for the simulation phase

3.2.2 Machine Learning

ML was completed using the scikit-learn program, using the mlens module that implements the SUPLRN framework defined by van der Laan et al. (2007). A total of 8 learners were used: BAYRID, DECTRE, KNN, LINREG, MLP, RANFOR, SVR, and additionally, a SUPLRN was composed of 5 base learners: DECTRE, LINREG, RANFOR, MLP, and SVR.

3.2.2.1 Linear Regression

In general, regression learners attempt to find a linear relationship between the target values (e.g., V_{mem}) and the feature values (e.g., intracellular/extracellular ion concentrations). In scikit-learn this learner was implemented as ordinary least squares LINREG, where the feature weights, otherwise known as coefficients, were calculated to minimize the sum of squares between the model's predicted values and the ground truth values from the training dataset (Pedregosa et al., 2011). Note that there was no offset used.

Equation 3.10: For a training dataset X and a total of p features with coefficients $w = (w_1, w_2, \ldots, w_p)$, LINREG attempts to find the coefficients that lead to the minimum sum of squares between the predicted (Xw) and ground truth values. This equation was taken from the scikit-learn user guide https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares (Pedregosa et al., 2011).

$$min_w ||Xw - y||_2^2$$

3.2.2.2 Bayesian Ridge

Bayesian regression techniques utilize probabilistic models and allow for the inclusion of regularization parameters, which are used to penalize the complexity of a model and consequently helps protect against overfitting (Pedregosa et al., 2011). Model complexity is linked to the number of features used during training as well as their magnitudes, measured by weights. For non-Bayesian regression, penalties such as ridge regularization, otherwise known as the L2 penalty, utilize

the sum of squares of the coefficient array w to calculate complexity, $||w||_2^2$, with a positive correlation between the L2 penalty value and model complexity (Kernbach & Staartjes, 2021). Ridge regression attempts to determine the feature weights, w, that minimizes both the sum of squares between the model's predicted and ground truth values, as well as the sum of squares of the complexity parameter α, specified by the user, and the estimated feature weights array (Pedregosa et al., 2011); see Equation 3.11.

Equation 3.11: Ridge regression is LINREG with ridge regularization, or the L2 penalty, on the right-hand side. The L2 penalty uses α as a complexity parameter that controls the amount of shrinkage for the coefficient array w (Pedregosa et al., 2011). As α increases the coefficients become smaller, leading to more robust models (Pedregosa et al., 2011). It is important to note that a model can be underfit if α becomes too large and the weights shrink too much. This equation was taken from the scikit-learn user guide https://scikit-learn.org/stable/modules/linear_model.html#regression (Pedregosa et al., 2011).

$$min_w ||Xw - y||_2^2 + \alpha||w||_2^2$$

On the other hand, BAYRID estimates a probability distribution for w with precision λ^{-1}. The ground truth values, y, are assumed to be Gaussian distributed around the training data with their applied feature weights, Xw (Pedregosa et al., 2011); see Equation 3.12. Bayesian regression in general has the advantage of being able to adapt to the training data since parameters such as α and λ, both used for regularization, are both estimated during learning using the training data (Pedregosa et al., 2011).

Equation 3.12: A) Bayesian regression generates a probability distribution for the feature weights (w), with α serving as a regularization parameter. B) The prior for w is given by a spherical Gaussian distribution, while gamma distributions are used for priors over both α and λ (Pedregosa et al., 2011). This equation was taken from the scikit-learn user guide https://scikit-learn.org/stable/modules/linear_model.html#bayesian-ridge-regression (Pedregosa et al., 2011).

A)	$p(y \mid X, w, \alpha) = N(y \mid Xw, \alpha)$
B)	$p(w \mid \lambda) = N(w \mid 0, \lambda^{-1} I_p)$

3.2.2.3 Decision Tree

DECTREs utilize an upside-down tree-like hierarchy, and simply put, work as a flowchart where complex decisions are broken into numerous smaller, simpler decisions that are easier to interpret (Xu et al., 2005). The root node is the starting point and contains all data, the internal nodes are where binary decisions based on certain criteria or thresholds are made, and finally, terminal or leaf nodes are where the target values are predicted (Xu et al., 2005). While there are multiple algorithms available for generating DECTREs, scikit-learn specifically implements a performance optimized version of the classification and regression trees (CART) algorithm (Pedregosa et al., 2011).

The process of generating a regression tree starts with the data Q_m located at node m with a total of n_m samples. For a given feature j and a threshold t_m, the

candidate split $\theta = (j, t_m)$ partitions the data into left (Q_m^{left}) and right (Q_m^{right}) subsets (Pedregosa et al., 2011). The quality of this split is then evaluated using a loss function, in this case MSE, to determine the best candidate. This entire process is performed recursively for every internal node until all samples at node m have been evaluated, or the maximum depth has been reached, $n_m < min_{samples}$, where $min_{samples}$ is defined as the minimum number of samples required for a split to take place (Pedregosa et al., 2011).

Equation 3.13: The equations used to partition the data Q_m located at node m into A) left and B) right subsets. For regressor trees the quality of this split is evaluated using the MSE loss function to determine whether it will be chosen. These equations were taken from the scikit-learn user guide https://scikit-learn.org/stable/modules/tree.html#mathematical-formulation (Pedregosa et al., 2011).

A)	$Q_m^{left}(\theta) = \{(x, y) \mid x_j \leq t_m\}$
B)	$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$

Equation 3.14: Equation of the MSE, used to determine the best candidate split θ. The variable $\bar{y}$ is the predicted value, y is the ground truth value, and $n_{samples}$ is the total number of data samples available. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error (Pedregosa et al., 2011).

$$MSE(y, \bar{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \bar{y}_i)^2$$

3.2.2.4 K-Nearest Neighbors

KNN is a non-parametric learner that utilizes a distance metric to determine the K number of closest points for a given input, x_i, and calculates its value based on these neighbors (Kramer, 2013; Pedregosa et al., 2011). In KNN regression, the predicted value for x_i is calculated using the mean value of its K-nearest neighbors (Kramer, 2013), where $K = 5$ in this chapter. Additionally, the Euclidean distance metric with L2 normalization was used to determine the neighboring points (Pedregosa et al., 2011).

Equation 3.15: Given the training dataset $\{(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i)\}$, the value for a given input, x_i, is calculated using the mean of its K-nearest neighboring points, represented by set $N_K(x_i)$. The variable y_i is the ground truth value in which the neighboring points are closest to. This equation was adapted from Kramer (2013), Equation 2.4.

$$F_{KNN}(x_i) = \frac{1}{K} \sum_{i \in N_K(x_i)} y_i$$

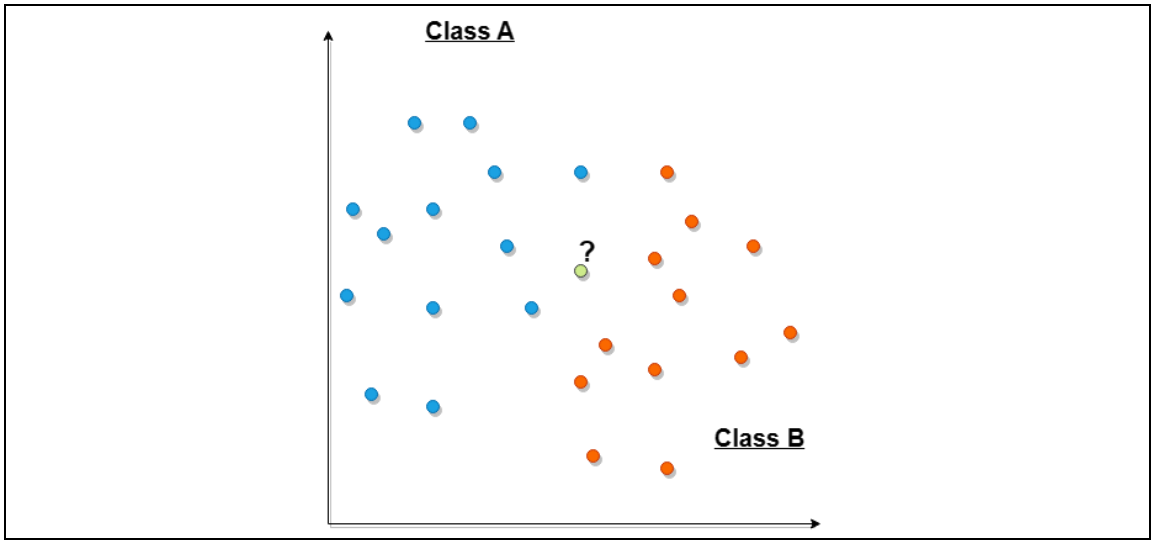


Figure 3.5: KNN plots all training data on an n-dimensional plot, where n is the total number of features. In the above example, $n = 2$ and there are a total of 2 classes: A (blue) and B (orange). The green point with the ? above it represents an input whose value will be predicted from the K nearest neighbors closest to it, measured using the Euclidean distance metric.

3.2.2.5 Multi-Layer Perceptron

The MLP learner was inspired by the human brain, utilizing neurons, or nodes that contain data such as feature values, to both relay and receive information to/from neurons in other layers in order to make a prediction. As an overview, the MLP learns the function $f(\cdot): R^m \rightarrow R^o$ given the features $X = \{x_1, x_2, x_3, \ldots, x_m\}$, where m and o represent the number of dimensions for the input and output sets respectively (Pedregosa et al., 2011). There are a total of 3 different types of layers: input, hidden, and output. The input layer is comprised of neurons that each contain a feature value x_i that is passed onto the hidden layer(s), where each neuron in a given hidden layer applies a set of weights to the neurons from the

previous one such that $w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_mx_m + b$, where b is a bias that serves as a scalar (Pedregosa et al., 2011). This is followed with the use of an activation function, in this case the rectified linear unit function $f(x) = max(0, x)$ that returns zero for any negative x or the value of x as-is if it is positive (Pedregosa et al., 2011). These values from the final hidden layer are then passed to the output layer, where the predicted output value is given.

However, the process does not end here; backpropagation is used to determine the optimal weights and biases that minimizes the MSE between the predicted and ground truth values (Pedregosa et al., 2011). Backpropagation starts by utilizing a loss function, in this case MSE, to quantify how far away the predicted value from the output layer is from the ground truth. This loss value is then used to calculate the amount of error each neuron within every hidden layer contributes, starting from the final layer and working towards the first (Cilimkovic, 2015). Afterwards, the weights of all neurons are updated using the adam solver, a stochastic gradient-based optimizer, which minimizes the overall MSE of the model (Cilimkovic, 2015; Pedregosa et al., 2011). The performance of the model is then checked again using the loss function, and backpropagation is repeated until either the maximum number of iterations is met, or the loss does not improve by a given tolerance for a specified number of iterations (Cilimkovic, 2015; Pedregosa et al., 2011).

Equation 3.16: A) An example function for f that contains one hidden layer. The variable W_1 represents the weights of all feature values within the input layer, W_2 are the weights of all feature values within the hidden layer, and b_1 and b_2 are the bias values for the hidden and output layers respectively. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/neural_networks_supervised.html#mathematical-formulation (Pedregosa et al., 2011). B) The activation function g which is the rectified linear unit function scikit-learn uses by default for MLP regression. This equation was obtained from the scikit-learn documentation for the MLP Regressor learner https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (Pedregosa et al., 2011).

A)	$f(x) = W_2 g(W_1^T x + b_1) + b_2$
B)	$g(z) = max(0, z)$

3.2.2.6 Random Forest

RANFOR is an ensemble method that utilizes multiple DECTREs to perform prediction. Each DECTRE is trained using sub-datasets generated from a process known as bootstrap sampling, where samples of the training dataset are randomly drawn with replacement, allowing them to be used again. Section 3.2.2.3 describes how DECTREs are built. Moreover, for each individual DECTRE, the best split was determined from a randomly selected feature, and the quality of the split was measured using the MSE metric (Pedregosa et al., 2011). Once all DECTREs have been built, RANFOR calculates the average of the ensemble to determine the predicted value (Pedregosa et al., 2011).

3.2.2.7 Support Vector Regression

SVR, specifically epsilon-SVR, utilizes the hyperparameter ε to define what is known as the epsilon-tube, a hyperplane with a width of a single ε deviation on top and bottom (Pedregosa et al., 2011). The hyperplane is generated such that predicted values should not go beyond an ε deviation from the ground truth values, however, the slack variables ζ and ζ^* are used as penalties for those that fall above and below the epsilon-tube respectively (Pedregosa et al., 2011; Zhang & O'Donnell, 2020). Fundamentally, SVR aims to generate the narrowest possible epsilon-tube with all predicted values lying inside. In this chapter, SVR with the rbf kernel was used.

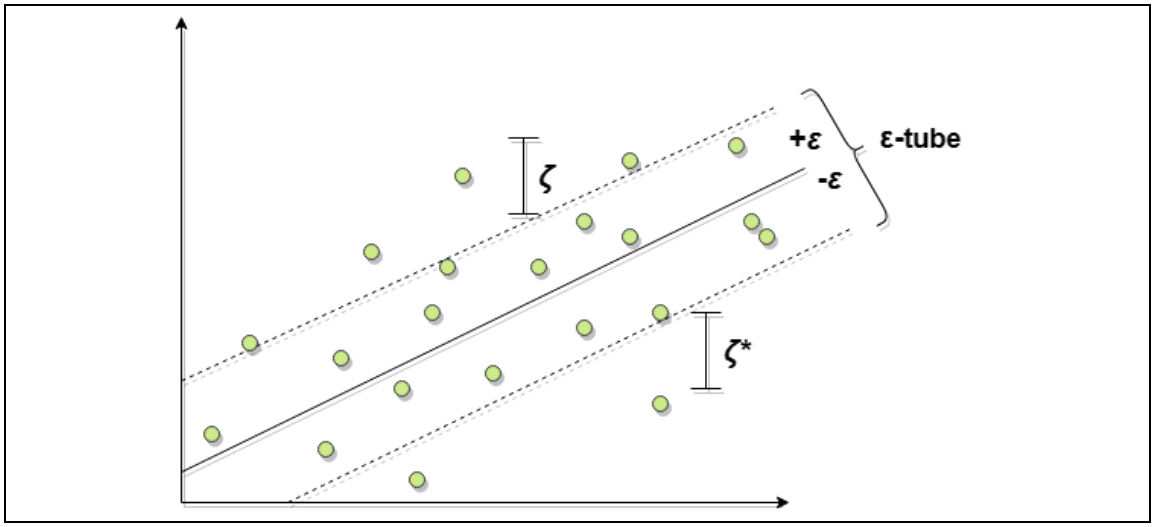


Figure 3.6: SVR constructs a hyperplane that best approximates the relationship between the input features and ground truth values. Predicted values should be at most $\pm\varepsilon$ deviations above/below the hyperplane, although the slack variables, ζ and ζ*, exist to penalize those that fall outside the epsilon-tube.

Equation 3.17: Given the training values $\{x_1, \ldots, x_n\}$ and ground truth values $\{y_1, \ldots, y_n\}$, SVR calculates the optimal weights (w) and bias (b) that generates the narrowest possible epsilon-tube with the smallest possible slack variables above (ζ) and below (ζ^*) the tube (Pedregosa et al., 2011). C is a hyperparameter used for regularization, which helps protect against overfitting. These equations were taken from the scikit-learn user guide https://scikit-learn.org/stable/modules/svm.html#svr (Pedregosa et al., 2011).

$$min_{w,b,\zeta_i,\zeta_i^*}[\frac{1}{2}w^T w + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*)]$$

3.2.2.8 Super Learner

The SUPLRN is an ensemble method that utilizes multiple learners to perform prediction, and in this thesis the DECTRE, LINREG, RANFOR, SVR, and MLP base learners were chosen. The process of generating a SUPLRN model was first outlined by van der Laan et al. (2007), and an overview of the entire process is shown in Figure 3.7.

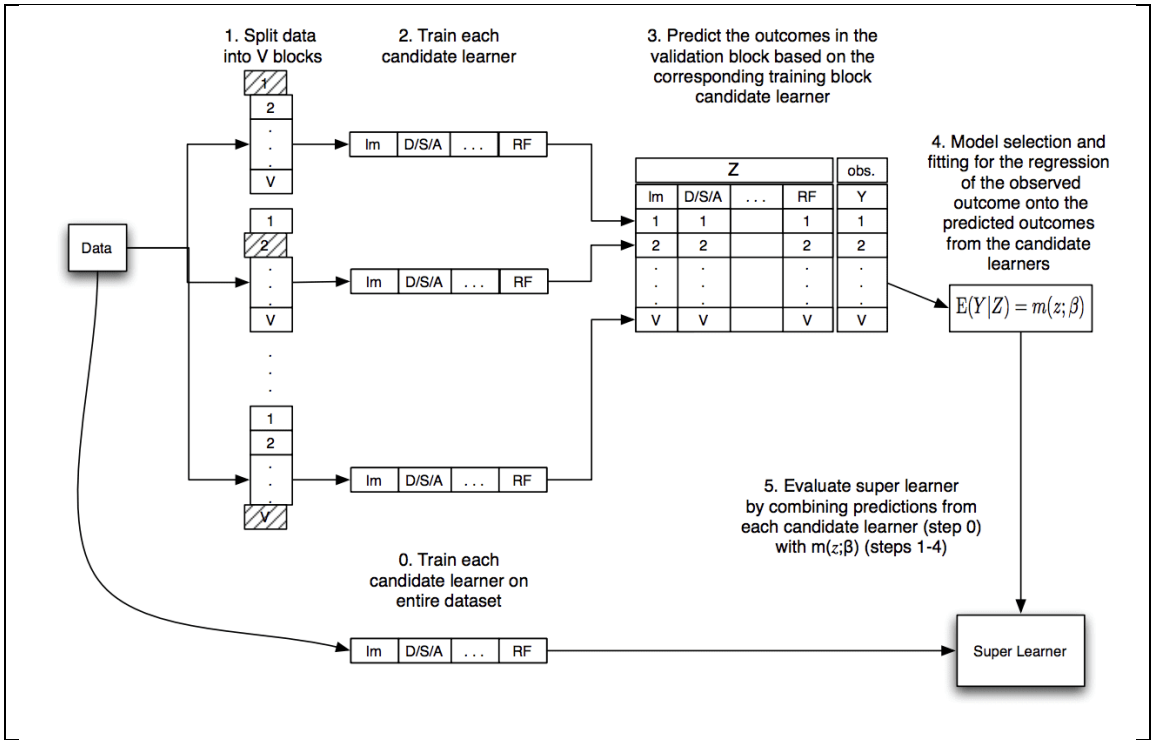


Figure 3.7: SUPLRN as defined by van der Laan et al. (2007). The learners were first trained on the entire training dataset and then set aside. k-fold CV was then performed for each learner, where their performance on each fold was stored in the Z matrix. A meta-learner was then used on the Z matrix to calculate the weight of each learner for the optimal combination that either maximized a scoring function or minimized a loss function, depending on which was chosen. Finally, these weights were then applied to the learners trained on the entire training set originally, which produced the learned model. Image taken directly from van der Laan et al. (2007), Figure 1.

First, the set of all learners, denoted by L, were trained on the entire training dataset. Next, external CV was performed for every learner using the R^2 performance metric, which has an output range of $(-\infty, 1]$. A score of 1 indicates the independent variables (feature values) explain all the variance of the dependent variable (V_{mem} for instance), while a score of 0, considered a soft lower

bound, indicates none of the variance between the feature values and V_{mem} or ion concentrations are explained by the model. In other words, for any score between 0 and 1 there is some percentage of variance explained by the trained model, while any value ≤0 indicates the model is unable to perform prediction with any degree of accuracy (Chicco et al., 2021).

In the case of the SUPLRN, the performance of every learner on each fold was placed into what is known as a Z-matrix with dimensions k by L, corresponding to the total number of folds used during CV, 10 in this case, by the total number of base learners, which was 5. Once CV was completed, a meta-learner, LINREG, was used to calculate the weights of each learner within the Z-matrix that best optimized the overall CV performance. The calculated weights determined how much input each learner has in the trained model; for instance, if SVR has a weight of 0.2, DECTRE has 0.3, and MLP has 0.7, then 70% of the SUPLRN prediction will be done by MLP, 20% by DECTRE, and 20% by SVR. Finally, the last step was to apply these weights to the learners initially trained on the entire training set, and the SUPLRN model was generated (van der Laan et al., 2007).

3.2.2.9 Validation

Performance was assessed using the validation dataset where the MAE, otherwise known as the L1 loss function, MSE or L2 loss function, median absolute error (MEDAE), and the coefficient of determination, or R^2, performance metrics

were all utilized. MSE's usage of the squared error between each predicted and ground truth value leads to the magnification of outliers, a problem that other metrics do not share. Both MAE and MEDAE handle outliers far better in comparison, with MEDAE being especially robust (Pedregosa et al., 2011) due to its usage of the median rather than taking into account all values like the other metrics. R^2 measures how well the independent variables, in this case the features, explain the amount of variance within the model (Pedregosa et al., 2011). While R^2 has an output value range of $(-\infty, 1]$, anything ≤0 signifies the model explains none of the variance between the independent and dependent variables, making 0 the soft lower boundary (Chicco et al., 2021). Consequently, the output range of R^2 can be considered $[0,1]$, where all negative values were simplified to 0 without losing meaning (Chicco et al., 2021).

Equation 3.18: MAE, otherwise known as the L1 loss metric, is calculated by summing the absolute value of the difference between the ground truth (y) and the predicted values ($\hat{y}$) for all samples in the validation set (n). The goal is to minimize the loss, so the best possible value is 0. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-error (Pedregosa et al., 2011).

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$

Equation 3.19: MSE, otherwise known as the L2 loss metric, is calculated by summing the squared difference between the ground truth value (y) and the predicted values ($\hat{y}$) for all samples in the validation set (n). The goal is to minimize the loss, so the best possible value is 0. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error (Pedregosa et al., 2011).

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2$$

Equation 3.20: MEDAE is determined by finding the median of a set of values, each of which correspond to the difference between the ground truth (y) and predicted ($\hat{y}$) values for all samples within the validation set, the total number being n. The best possible value is 0. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-error (Pedregosa et al., 2011).

$$MEDAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|)$$

Equation 3.21: R^2 is calculated by subtracting from 1 the sum of the squared difference between all ground truth (y) values from the predicted ($\hat{y}$) values, divided by the sum of the squared difference between the ground truth and the mean ($\bar{y}$) of all validation values. The total number of entries within the validation set is represented by n. The best possible value is 1. This equation was obtained from the scikit-learn user guide https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score (Pedregosa et al., 2011).

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

3.3 Results

3.3.1 Objective 1

Table 3.2: Objective 1 performance of all models over all metrics. Highlighted entries signify the model with the best performance for that metric/column.

Model	MAE	MSE	MEDAE	R2
BAYRID	21.91	936.46	16.60	0.57
DECTRE	2.35	67.00	0.00	0.97
KNN	2.25	56.10	0.00	0.97
LINREG	21.91	936.46	16.60	0.57
MLP	4.24	64.72	1.40	0.97
RANFOR	2.07	48.84	0.00	0.98
SVR	6.49	123.32	3.44	0.94
SUPLRN	3.15	53.81	0.76	0.98

Table 3.3: Objective 1 R^2 performance of each learner within the SUPLRN on 10-fold CV.

Learner	Score	Standard Deviation
DECTRE	0.97	0.00
LINREG	0.57	0.00
MLP	0.96	0.00
RANFOR	0.98	0.00
SVR	0.94	0.00

3.3.2 Objective 2

For objectives 2 and 3 a total of 26,817 BETSE simulations were used.

Table 3.4: Objective 2 performance of all models over all metrics. Highlighted entries signify the model with the best performance for that metric/column. The following parameters were modified from their defaults: MLP's max_iter was set to 99,999 from 200; SVR's max_iter was set to 75,000 from -1 (no limit); Within the SUPLRN: DECTRE's max_depth was set to 50 from None (no limit); RANFOR's max_depth was set to 50 from None (no limit); SVR's max_iter was set to 50,000; MLP's max_iter was set to 3,000.

Model	MAE	MSE	MEDAE	R2
BAYRID	3.56E+124	1.27E+249	3.56E+124	0.00
DECTRE	0.91	2880.39	0.01	0.31
KNN	4.99	103.28	1.70	0.92
LINREG	1.55E+125	3.93E+250	1.27E+125	0.00
MLP	9.60	3050.36	5.98	0.27
RANFOR	16.59	3191.34	16.07	0.23
SVR	34.07	4291.04	34.66	0.00
SUPLRN	5.46	2946.66	2.39	0.29

Table 3.5: Objective 2 R^2 performance of each learner within the SUPLRN on 10-fold CV.

Learner	Score	Standard Deviation
DECTRE	0.86	0.00
LINREG	0.38	0.00
MLP	0.87	0.00
RANFOR	0.92	0.00
SVR	0.30	0.02

3.3.3 Objective 3

Table 3.6: Objective 3 performance of all models over all metrics. Highlighted entries signify the model with the best performance for that metric/column. The following parameters were modified from their defaults: MLP's max_iter was set to 99,999 from 200; SVR's max_iter was set to 250,000 from -1 (no limit). Within the SUPLRN: MLP's max_iter was set to 3,000; RANFOR's max_depth was set to 50 from None (no limit); SVR's max_iter was set to 250,000.

Model	MAE	MSE	MEDAE	R2
BAYRID	2.88	19.45	1.98	0.97
DECTRE	0.22	0.90	0.06	1.00
KNN	0.65	3.33	0.06	1.00
LINREG	2.88	19.45	1.98	0.97
MLP	1.57	6.58	0.91	0.99
RANFOR	0.20	0.57	0.03	1.00
SVR	3.25	19.64	2.80	0.97
SUPLRN	0.17	0.47	0.04	1.00

Table 3.7: Objective 3 R^2 performance of each learner within the SUPLRN on 10-fold CV.

Learner	Score	Standard Deviation
DECTRE	1.00	0.00
LINREG	0.97	0.00
MLP	0.99	0.00
RANFOR	1.00	0.00
SVR	0.98	0.00

3.4 Discussion

ML models corresponding to BAYRID, DECTRE, KNN, LINREG, MLP, RANFOR, SVR, and SUPLRN were trained to complete 3 separate objectives. Objective 1 focused on predicting the average V_{mem} of an entire network of cells, objective 2

was to predict the V_{mem} of each individual cell within the network, and finally, objective 3 was to predict the average ion concentration for Na^+, K^+, Cl^-, and Ca^{2+} of the entire cellular network. The SUPLRN was comprised of the DECTRE, LINREG, MLP, RANFOR, and SVR learners. Note that a 2-tailed paired samples t-test was used to compare the models from each objective, where any value with $p < 0.05$ was considered significant.

For objective 1, LINREG and BAYRID, both linear learners, were shown to have the worst performance of all models. Conversely, RANFOR had the best performance over all metrics, where DECTRE and KNN matched its MEDAE score, and SUPLRN had the same R^2 score. The t-test between RANFOR and DECTRE ($p = 0.30$) showed no significant difference between the models, while RANFOR and KNN ($p = 0.00$) as well as RANFOR and SUPLRN ($p = 0.00$) demonstrated that the performance of these models was significantly different. The same was found between DECTRE, KNN, and SUPLRN, $p = 0.00$ for every pair. Outside of the linear learners, SVR was also found to be particularly sensitive to outliers as shown by its MSE score. Additionally, the performance of each base learner within the SUPLRN regarding 10-fold CV are shown in Table 3.3. Here, the performance of the trained models within the SUPLRN are reflective of their individual counterparts (shown in Table 3.2), with DECTRE and RANFOR performing best and LINREG being the worst.

In objective 2, DECTRE had the best MAE and MEDAE scores while KNN was the most performant model over MSE and R^2; see Table 3.5. The performance of these models was not found to be significantly different ($p = 0.36$) to each other. BAYRID and LINREG were again the worst performing models over all metrics, demonstrating linear learners were not suited for V_{mem} prediction using BETSE data. Unlike objective 1 previously, the R^2 scores of the models within the SUPLRN did not reflect the ranking of their individual counterparts, with RANFOR having the best performance on 10-fold CV, SVR performing the worst after LINREG, and MLP and DECTRE's scores being within 0.01 of each other.

Finally, in objective 3 the SUPLRN was found to be most performant over the MAE and MSE metrics, RANFOR had the best MEDAE score (by 0.01 over SUPLRN), and SUPLRN, RANFOR, DECTRE, and KNN all had equivalent R^2 values. Here, the performance of the SUPLRN was found to be significant compared to each of the RANFOR ($p = 0.00$), DECTRE ($p = 0.00$), and KNN ($p = 0.00$) models. Moreover, there was no significant statistical difference between the performance of DECTRE and RANFOR ($p = 0.11$), while all other 2-tailed paired samples t-tests between DECTRE and KNN ($p = 0.01$) as well as RANFOR and KNN ($p = 0.04$) were significant. While BAYRID and LINREG continued to lag behind the others, in this objective SVR performed the worst over all metrics except for R^2, where all 3 of these models had the same score. Within the SUPLRN, LINREG was in last place, within 0.01 of SVR, which itself was the same

amount away from MLP, and DECTRE and RANFOR obtained perfect R^2 scores over 10-fold CV.

Overall, the results show linear models did not perform well in any of the objectives, while tree-based models such as DECTRE and RANFOR performed well in V_{mem} and ion concentration prediction within objectives 1 and 3. In objective 2, the performance of DECTRE and KNN was found to not be significantly different, meaning either model would be appropriate for predicting the V_{mem} of each individual cell within a network. On average, the performance of all models was worse in objective 2 compared to the others; for example, the mean MSE score was approximately 2743.85 (excluding BAYRID and LINREG) compared to 285.84 in objective 1 and 8.80 in objective 3, and the mean R^2 score was 0.25 in objective 2 compared to 0.87 and 0.99 for objectives 1 and 3 respectively. Moreover, these averages also demonstrate that the models tended to perform best in objective 3.

3.5 Conclusion

The goal of this study was to utilize ML to replicate the functionality of BETSE, a program that simulates bioelectric signaling of cellular networks via GJs, the activity of the Na^+, K^+, Cl^-, and Ca^{2+} ion channels, and the V_{mem} of each cell within the network. Bioelectricity is used in a myriad of ways; for instance, cells use V_{mem} to communicate and form networks with one another as well as to regulate

processes such as the growth of tumors, stem cell differentiation, and the control of voltage-gated ion channels to name a few (Pietak & Levin, 2016; Silver & Nelson, 2018). The ability to configure parameters in BETSE such as membrane diffusion constants as well as the initial extracellular and intracellular ion concentrations of Na^+, K^+, Cl^-, and Ca^{2+}, all of which ultimately affect V_{mem}, allow for a variety of cellular networks to be modeled. This is useful because diseases such as cancer up/downregulate ion channels in order to proliferate (Haworth & Brackenbury, 2019), while Tuszynski et al. (2017) has shown *in-vitro* that bringing cancer-state depolarized cells back to a healthy-state V_{mem} can prevent and even reverse tumorigenesis in some cases.

All models were trained and validated using data generated from BETSE, where the configuration files had randomized values for specific parameters (e.g., size of the environmental grid). This is one of the limitations of this study, seeing as the trained ML models can only ever be as good as BETSE, which itself simulates *in-vivo* conditions. In other words, all trained models were 2 degrees away from reality; the degree of error from ML to BETSE, and then the degree of error of BETSE from *in-vivo* conditions. Furthermore, scikit-learn does not always handle large data efficiently, often requiring the entire training dataset to be loaded into memory during training. This can lead to memory usage in the hundreds of gigabytes depending on the learner, number of features, and number of examples in the dataset, and while the “partial_fit” method is available for incremental

learning, this only covers a small subset of all available learners (Buitinck et al., 2013).

In conclusion, 8 ML models were trained for each of the 3 objectives and their performance was compared to determine the best model for each task. Here, RANFOR was the most performant over all MAE, MEDAE, MSE, and R^2 metrics in objective 1, the performance of DECTRE and KNN, each with the best score in half of all scoring metrics, was found to not be statistically significant in objective 2, and SUPLRN had the best MAE and MSE scores along with an equivalent R^2 score to DECTRE, KNN, and RANFOR in objective 3. Future work in bioelectric signal modeling will incorporate other parameters of cellular networks such as temperature and pressure, as well as more fine-grained properties of cells such as the surface area of GJs for instance. It may also be possible to obtain experimental data from published literature as bioelectric signaling becomes more of a focus. Similar to the trained classifiers in Chapter 2, used to determine whether a given drug is an agonist, antagonist, or decoy to the AR, ER, GR, and PR nuclear receptors, the advantage of ML is that it provides a cost-efficient method of prediction in context of both time and computational resources. The models generated in this chapter allow numerous cellular networks, each with varied parameters, to be predicted while only requiring a fraction of the computational power needed to run a single BETSE simulation. This additionally has the advantage of being able to scale to larger networks that BETSE would not

be able to handle due to limitations on computer memory. These advantages apply to BEN as well, a more minimalistic version of BETSE developed by Manicka and Levin (2019) that also utilizes differential equations for estimating processes such as ion flux or electrodiffusion.

Discussion and Conclusions

In this thesis, various applications of *in-silico* drug discovery were explored, starting with classical methodologies such as drug docking and MD/SA. While these techniques have been around for decades, they are still the go-to for designing novel therapeutics, as in the case PAL modeling shown in Chapter 1. This study was performed in collaboration with CSTS Health Care to develop novel PALs that bind pharmaceuticals, for example a cancer drug, and allow for precision drug delivery to tissues via platelet sequestration. The 3D structures of PAL and PAL-derived PALs were generated using their FASTA sequences, while the receptor, CSA, was created *in-silico* using a template structure to build from. A 4-phase SA technique was then used on CSA to obtain its lowest energy conformation, followed with blind docking of the PALs to this structure. Electrostatic analysis of the docked ligands revealed that the arginines on PAL interfered with sulfate groups on CSA during binding. Future development and optimizations of PAL will include relocation of these amino acids to better accommodate CSA, ultimately leading to better binding affinity of the PAL:CSA complex.

Chapter 2 involved the use of ML to train a series of classifiers that determined whether a given drug was an agonist, antagonist, or decoy to the AR, ER, GR, and PR nuclear receptors. This chapter highlighted how scientific literature can be leveraged to build new tools while also mitigating a pitfall of ML,

not having enough data for training and validation. This issue was remedied using data augmentation, which generated additional agonists and antagonists for each receptor using existing ones as templates. However, this solution added a layer of uncertainty because the generated structures may not reflect actual agonists and antagonists, which could have degraded the performance of the trained models. A total of 5 models were compared, where NAIBAY had the best performance on AR, NEUNET for ER and GR, and the classification accuracy of DECTRE and NAIBAY were within 0.01% of each other for PR. A 2-tailed paired samples t-test showed no significance between NEUNET, which had the highest average accuracy over all receptors, and DECTRE, NAIBAY, and RANFOR. While these models were found to be statistically significant to SVM, which had the worst performance over all receptors, the DECTRE, NAIBAY, and RANFOR models were not statistically significant to one other.

Chapter 3 focused on modeling bioelectricity, an area that has historically received relatively little attention. The BETSE application, developed by Pietak and Levin (2016), simulates GJs and ion channels of cells to predict the cytosolic and extracellular concentrations of Na^+, K^+, Cl^-, and Ca^{2+} ions, which in turn provides the V_{mem} of each cell within the network. Cells utilize V_{mem} to form circuits with one another, allowing them to regulate a host of processes including the growth of tumors and the opening and closing of voltage-gated ion channels. Using experimentally obtained membrane ion permeabilities as well as

extracellular Na^+, K^+, and Cl^- of Xenopus oocytes, BETSE was shown to predict the V_{mem} and intracellular ion concentrations with <10% difference to their *in-vitro* values (Pietak & Levin, 2016). That being said, a given BETSE simulation can be time consuming and computationally expensive, especially when simulating larger networks. ML models were generated based on BETSE data to predict 1) the V_{mem} of a cellular network, as well as 2) each individual cell within the network, and additionally, 3) the average Na^+, K^+, Cl^-, and Ca^{2+} concentrations. Similar to Chapter 2, the training data here was completely based on *in-silico* results which presented an extra layer of uncertainty compared to using experimental data directly. Tree-based models such as DECTRE and RANFOR performed well over these objectives, while BAYRID and LINREG, both linear models, did not. In particular, the performance of DECTRE was found to not be significantly different to either RANFOR or KNN in objectives 1 and 2, where these models were among the top performers. In objective 3 RANFOR marginally had the best MEDAE value over the SUPLRN model, which had the lowest MAE and MSE scores and also tied with DECTRE, KNN, and RANFOR in the R^2 scoring metric. Here, the performance between DECTRE and RANFOR were not statistically significant either.

Advances in computational power and storage in addition to the development of numerous learning algorithms have led to an immense growth in the usage of ML for biomedical research. In this thesis ML was used for both

classification and regression tasks, providing a cost and resource effective method for drug activity as well as bioelectric and ion channel concentration prediction. Within the pharmaceutical industry, ML has been integrated into various stages of drug discovery including target selection and validation, lead discovery, and even preclinical and clinical development through biomarker and drug response/pharmacological predictions (Patel et al., 2020; Vamathevan et al., 2019). While newer ML methodologies have many benefits, existing *in-silico* tools are still a critical part of the drug discovery process and likely will not be replaced for quite some time due to the lack of data, especially high quality or expert curated, in many areas. Chapter 3 was a prime example of this issue, where large amounts of experimentally validated bioelectric data were not readily available, requiring the use of BETSE instead.

To date there have been over 70 therapeutics brought to market with computational techniques playing a variety of roles in each (Sabe et al., 2021). In the case of Imatinib for example, drug docking of various chemical libraries was performed against Bcr-Abl tyrosine kinase for lead identification. Furthermore, drug docking was also utilized during the development of Selinexor to identify exportin-1 inhibitors, and for Vaborbactam, both drug docking as well as MD were performed to determine beta-lactamase inhibitors (Dhillon, 2018; Sabe et al., 2021; Syed, 2019). Moreover, the exorbitant cost and time needed to develop a therapeutic from scratch and bring it to market has led to the process of drug

repurposing as an alternative, with a few examples being sildenafil, originally designed for angina and then repurposed to treat erectile dysfunction, mifepristone, which was first approved for abortion and found later to work as a treatment for Cushing's syndrome, and topiramate, whose original indication of epilepsy was later expanded to include migraines as well (Akhoon et al., 2019). Bioinformatic and ML techniques are especially able to take advantage of protein, genomic, pharmacological, etc. databases to identify novel interactions for the objective of drug repurposing. DRUGPATH for instance incorporates several expert-curated resources to provide a meta-database that maps interactions between drugs, genes, targets, and biological pathways to predict adverse drug interactions (Jaundoo & Craddock, 2020). Likewise, comboFM is a ML framework developed by Julkunen et al. (2020) that utilizes a factorization machine, used for non-linear learning of large data, to determine viable drug combinations and dosages for preclinical studies such as cancer cell lines.

Future work will involve the additional use of ML to replace more *in-silico* tools within drug discovery such as NAMD for instance, which was used in Chapter 2 to perform MD. Here, natural language processing could be utilized to extract experimental data from published literature about various protein families, and additionally, databases containing already performed MD simulations such as MoDEL (T. Meyer et al., 2010) would be used for training and validation. A similar process could be used to generate ML models that predict the docked pose(s) of

a given drug to a receptor, with the addition of integrating chemical and physical properties of pharmaceuticals including molecular weight, number of H-bond donors and acceptors, and water solubility for example to increase performance. As more *in-vivo* and *in-vitro* data become available, *in-silico* tools will be able to integrate this data to produce increasingly accurate models. This may allow computational models to eventually replace mammalian animal studies, which are costly and have various issues including a lack of reliability between various animal species and strains, poor translation of animal data to human trials, and ethical concerns such as prolonged conditions of pain and suffering as well as the requirement to euthanize animals after experimentation (Freires et al., 2016; Robinson et al., 2019).

In conclusion, *in-silico* tools and methods are a core part of modern-day biomedical research at large, allowing researchers to model and simulate biological processes that would otherwise be costly, time consuming, or improbable *in-vitro* or *in-vivo*. For instance, conventional *in-silico* methods such as molecular docking and MD have been shown to play valuable roles, both past and present, in drug discovery as well as repurposing. Additionally, the vast amount of biomedical data available in databases, repositories, and published literature enables ML and informatics methods to predict ligand:target affinity, protein structure and dynamics, and moreover, identify various drug-drug, drug-target, and protein-protein interactions that would otherwise be unknown. The accuracy

of *in-silico* models and predictions will continue to increase as the field progresses, leading to reduced costs and faster development within the drug discovery process, more effective therapeutics and treatments with fewer side effects, and ultimately, better outcomes for patients.

Bibliography

Akhoon, B. A., Tiwari, H., & Nargotra, A. (2019). In silico drug design methods for drug repurposing. In *In Silico Drug Design* (pp. 47–84). Elsevier. http://dx.doi.org/10.1016/b978-0-12-816125-8.00003-1

Aldas-Bulos, V. D., & Plisson, F. (2023). Benchmarking protein structure predictors to assist machine learning-guided peptide discovery. *Digital Discovery*, *2*(4), 981–993. https://doi.org/10.1039/d3dd00045a

Amaro, R. E., Baudry, J., Chodera, J., Demir, Ö., McCammon, J. A., Miao, Y., & Smith, J. C. (2018). Ensemble docking in drug discovery. *Biophysical Journal*, *114*(10), 2271–2278. https://doi.org/10.1016/j.bpj.2018.02.038

Asako, Y., & Uesawa, Y. (2017). High-Performance prediction of human estrogen receptor agonists based on chemical structures. *Molecules*, *22*(4), 675. https://doi.org/10.3390/molecules22040675

Ben-Hur, A., & Weston, J. (2009). A user's guide to support vector machines. In *Methods in Molecular Biology* (pp. 223–239). Humana Press. http://dx.doi.org/10.1007/978-1-60327-241-4_13

Berrar, D. (2019). Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology* (pp. 542–545). Elsevier. http://dx.doi.org/10.1016/b978-0-12-809633-8.20349-x

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R.,

Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013, September 1). *API design for machine learning software: Experiences from the scikit-learn project*. arXiv.Org. https://arxiv.org/abs/1309.0238

Butterfield, K. C., Caplan, M., & Panitch, A. (2010). Identification and sequence composition characterization of chondroitin sulfate-binding peptides through peptide array screening. *Biochemistry*, *49*(7), 1549–1555. https://doi.org/10.1021/bi9021044

Cáceres, E. L., Tudor, M., & Cheng, A. C. (2020). Deep learning approaches in predicting ADMET properties. *Future Medicinal Chemistry*, *12*(22), 1995–1999. https://doi.org/10.4155/fmc-2020-0259

Cervera, J., Alcaraz, A., & Mafe, S. (2016). Bioelectrical signals and ion channels in the modeling of multicellular patterns and cancer biophysics. *Scientific Reports*, *6*(1). https://doi.org/10.1038/srep20403

Chan, H. C. S., Shan, H., Dahoun, T., Vogel, H., & Yuan, S. (2019). Advancing drug discovery via artificial intelligence. *Trends in Pharmacological Sciences*, *40*(8), 592–604. https://doi.org/10.1016/j.tips.2019.06.004

Chang, C.-C., & Lin, C.-J. (2019). *LIBSVM: A Library for Support Vector Machines*. https://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf.

Chemical Computing Group. (2019a). *MOE User Guide*.

Chemical Computing Group. (2019b). Molecular Operating Environment (MOE). *2019.01*.

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, *7*, e623. https://doi.org/10.7717/peerj-cs.623

Cilimkovic, M. (2015). Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown*, *15*(1).

Crampon, K., Giorkallos, A., Deldossi, M., Baud, S., & Steffenel, L. A. (2022). Machine-learning methods for ligand–protein molecular docking. *Drug Discovery Today*, *27*(1), 151–164. https://doi.org/10.1016/j.drudis.2021.09.007

Cummings, J. L., Goldman, D. P., Simmons-Stern, N. R., & Ponton, E. (2021). The costs of developing treatments for Alzheimer's disease: A retrospective exploration. *Alzheimer's & Dementia*, *18*(3), 469–477. https://doi.org/10.1002/alz.12450

Dalby, A., Nourse, J. G., Hounshell, W. D., Gushurst, A. K. I., Grier, D. L., Leland, B. A., & Laufer, J. (1992). Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences*, *32*(3), 244–255. https://doi.org/10.1021/ci00007a012

de Laeter, J. R., Böhlke, J. K., De Bièvre, P., Hidaka, H., Peiser, H. S., Rosman, K. J. R., & Taylor, P. D. P. (2003). Atomic weights of the elements.

Review 2000 (IUPAC Technical Report). *Pure and Applied Chemistry*, *75*(6), 683–800. https://doi.org/10.1351/pac200375060683

Dhillon, S. (2018). Meropenem/Vaborbactam: A review in complicated urinary tract infections. *Drugs*, *78*(12), 1259–1270. https://doi.org/10.1007/s40265-018-0966-7

Durrant, J. D., & McCammon, J. A. (2011). Molecular dynamics simulations and drug discovery. *BMC Biology*, *9*(1). https://doi.org/10.1186/1741-7007-9-71

Durrant, J. D., & McCammon, J. A. (2012). AutoClickChem: Click chemistry in Silico. *PLoS Computational Biology*, *8*(3), e1002397. https://doi.org/10.1371/journal.pcbi.1002397

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., & Roth, A. (2015). Generalization in adaptive data analysis and holdout reuse. *Advances in Neural Information Processing Systems*, *28*.

Ekins, S., Mestres, J., & Testa, B. (2007). In silico pharmacology for drug discovery: Methods for virtual ligand screening and profiling. *British Journal of Pharmacology*, *152*(1), 9–20. https://doi.org/10.1038/sj.bjp.0707305

El Naqa, I., & Murphy, M. J. (2015). What is machine learning? In *Machine Learning in Radiation Oncology* (pp. 3–11). Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-18305-3_1

Erb, R. J. (1993). Introduction to backpropagation neural network computation. *Pharmaceutical Research*, *10*(2), 165–170. https://doi.org/10.1023/A:1018966222807

Freires, I. A., Sardi, J. de C. O., de Castro, R. D., & Rosalen, P. L. (2016). Alternative animal and non-animal models for drug discovery and development: Bonus or burden? *Pharmaceutical Research*, *34*(4), 681–686. https://doi.org/10.1007/s11095-016-2069-z

Gendreau, M., & Potvin, J. Y. (Eds.). (2010). *Handbook of metaheuristics* (Vol. 2, p. 9). Springer.

Ghahramani, Z. (2003). Unsupervised Learning. In *Summer school on machine learning* (pp. 72–112). Springer.

Goodall, G. J., & Wickramasinghe, V. O. (2020). RNA in cancer. *Nature Reviews Cancer*, *21*(1), 22–36. https://doi.org/10.1038/s41568-020-00306-0

Hagen, J. B. (2000). The origins of bioinformatics. *Nature Reviews Genetics*, *1*(3), 231–236. https://doi.org/10.1038/35042090

Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of Optimization Techniques Based on Gradient Descent Algorithm: A Review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, *18*(4), 2715–2743.

Hall, M. A., & Smith, L. A. (1998). *Practical feature subset selection for machine learning* (C. McDonald, Ed.; pp. 181–191). Springer.

Hatmal, M. M., & Taha, M. O. (2017). Simulated annealing molecular dynamics and ligand–receptor contacts analysis for pharmacophore modeling. *Future Medicinal Chemistry*, *9*(11), 1141–1159. https://doi.org/10.4155/fmc-2017-0061

Haworth, A. S., & Brackenbury, W. J. (2019). Emerging roles for multifunctional ion channel auxiliary subunits in cancer. *Cell Calcium*, *80*, 125–140. https://doi.org/10.1016/j.ceca.2019.04.005

Hetényi, C., & van der Spoel, D. (2002). Efficient docking of peptides to proteins without prior knowledge of the binding site. *Protein Science*, *11*(7), 1729–1737. https://doi.org/10.1110/ps.0202302

Isert, C., Atz, K., & Schneider, G. (2023). Structure-based drug design with geometric deep learning. *Current Opinion in Structural Biology*, *79*, 102548. https://doi.org/10.1016/j.sbi.2023.102548

Italiano, J. E., Jr, Richardson, J. L., Patel-Hett, S., Battinelli, E., Zaslavsky, A., Short, S., Ryeom, S., Folkman, J., & Klement, G. L. (2008). Angiogenesis is regulated by a novel mechanism: Pro- and antiangiogenic proteins are organized into separate platelet α granules and differentially released. *Blood*, *111*(3), 1227–1233. https://doi.org/10.1182/blood-2007-09-113837

Jaundoo, R., & Craddock, T. J. A. (2020). DRUGPATH: The drug gene pathway meta-database. *International Journal of Molecular Sciences*, *21*(9), 3171. https://doi.org/10.3390/ijms21093171

Julkunen, H., Cichonska, A., Gautam, P., Szedmak, S., Douat, J., Pahikkala, T., Aittokallio, T., & Rousu, J. (2020). Leveraging multi-way interactions for systematic prediction of pre-clinical drug combination effects. *Nature Communications*, *11*(1). https://doi.org/10.1038/s41467-020-19950-z

Jurk, K., & Kehrel, B. E. (2005). Platelets: Physiology and biochemistry. *Seminars in Thrombosis and Hemostasis*, *31*(04), 381–392. https://doi.org/10.1055/s-2005-916671

Kabir, A., & Muth, A. (2022). Polypharmacology: The science of multi-targeting molecules. *Pharmacological Research*, *176*, 106055. https://doi.org/10.1016/j.phrs.2021.106055

Kaiser, C. J. (2012). *The Capacitor Handbook* (1st ed.). Springer. (Original work published 1992)

Karplus, M. (2003). Molecular dynamics of biological macromolecules: A brief history and perspective. *Biopolymers*, *68*(3), 350–358. https://doi.org/10.1002/bip.10266

Karsoliya, S. (2012). Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. *International Journal of Engineering Trends and Technology*, *3*(6), 714–717.

Klement, G. L., Yip, T.-T., Cassiola, F., Kikuchi, L., Cervi, D., Podust, V., Italiano, J. E., Wheatley, E., Abou-Slaybi, A., Bender, E., Almog, N., Kieran, M. W., & Folkman, J. (2009). Platelets actively sequester angiogenesis

regulators. *Blood*, *113*(12), 2835–2842. https://doi.org/10.1182/blood-2008-06-159541

Klusowski, J. M. (2019). Analyzing CART. *arXiv*, *1906.10086*. https://doi.org/10.48550/arXiv.1906.10086

Kowalska, M. A., Rauova, L., & Poncz, M. (2010). Role of the platelet chemokine platelet factor 4 (PF4) in hemostasis and thrombosis. *Thrombosis Research*, *125*(4), 292–296. https://doi.org/10.1016/j.thromres.2009.11.023

Kramer, O. (2013). K-Nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors* (pp. 13–23). Springer. http://dx.doi.org/10.1007/978-3-642-38652-7_2

Kramer, S., Widmer, G., Pfahringer, B., & de Groeve, M. (2000). Prediction of ordinal classes using regression trees. In *Lecture Notes in Computer Science* (pp. 426–434). Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-39963-1_45

Kumar, A., & Jain, M. (2020). *Ensemble learning for AI developers*. Apress. http://dx.doi.org/10.1007/978-1-4842-5940-5

Learned-Miller, E., G. (2014). *Introduction to supervised learning*.

Lee, C., & Lee, G. G. (2006). Information gain and divergence-based feature selection for machine learning-based text categorization. *Information*

Processing & Management, *42*(1), 155–165. https://doi.org/10.1016/j.ipm.2004.08.006

Lee, S. K., Jeakins, G. S., Tukiainen, A., Hewage, E., & Armitage, O. E. (2020). Next-Generation bioelectric medicine: Harnessing the therapeutic potential of neural implants. *Bioelectricity*, *2*(4), 321–327. https://doi.org/10.1089/bioe.2020.0044

Lemley, J., Bazrafkan, S., & Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, *5*, 5858–5869. https://doi.org/10.1109/access.2017.2696121

Manicka, S., & Levin, M. (2019). Modeling somatic computation with non-neural bioelectric networks. *Scientific Reports*, *9*(1). https://doi.org/10.1038/s41598-019-54859-8

Meyer, D. (2009). *Support Vector Machines: The Interface to libsvm in package e1071*.

Meyer, T., D'Abramo, M., Hospital, A., Rueda, M., Ferrer-Costa, C., Pérez, A., Carrillo, O., Camps, J., Fenollosa, C., Repchevsky, D., Gelpí, J. L., & Orozco, M. (2010). MoDEL (molecular dynamics extended library): A Database of atomistic molecular dynamics trajectories. *Structure*, *18*(11), 1399–1409. https://doi.org/10.1016/j.str.2010.07.013

Mierswa, I., & Klinkenberg, R. (2020). *RapidMiner Studio (version 9.6)*.

Mori, T. (2002). Information gain ratio as term weight. *Proceedings of the 19th International Conference on Computational Linguistics*. http://dx.doi.org/10.3115/1072228.1072246

Mutasa, S., Sun, S., & Ha, R. (2020). Understanding artificial intelligence based radiology studies: What is overfitting? *Clinical Imaging*, *65*, 96–99. https://doi.org/10.1016/j.clinimag.2020.04.025

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics*, *18*(6), 275–285. https://doi.org/10.1002/cem.873

Mysinger, M. M., Carchia, M., Irwin, John. J., & Shoichet, B. K. (2012). Directory of useful decoys, enhanced (DUD-E): Better ligands and decoys for better benchmarking. *Journal of Medicinal Chemistry*, *55*(14), 6582–6594. https://doi.org/10.1021/jm300687e

Neubig, R. R., Spedding, M., Kenakin, T., & Christopoulos, A. (2003). International Union of Pharmacology Committee on Receptor Nomenclature and Drug Classification. XXXVIII. Update on terms and symbols in quantitative pharmacology. *Pharmacological Reviews*, *55*(4), 597–606. https://doi.org/10.1124/pr.55.4.4

Noble, D. (2002). The rise of computational biology. *Nature Reviews Molecular Cell Biology*, *3*(6), 459–463. https://doi.org/10.1038/nrm810

O'Boyle, N. M. (2012). Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *Journal of Cheminformatics*, *4*(1), 1–14. https://doi.org/10.1186/1758-2946-4-22

Palve, V., Liao, Y., Remsing Rix, L. L., & Rix, U. (2021). Turning liabilities into opportunities: Off-target based drug repurposing in cancer. *Seminars in Cancer Biology*, *68*, 209–229. https://doi.org/10.1016/j.semcancer.2020.02.003

Patel, L., Shukla, T., Huang, X., Ussery, D. W., & Wang, S. (2020). Machine learning methods in drug discovery. *Molecules*, *25*(22), 5277. https://doi.org/10.3390/molecules25225277

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(85), 2825–2830.

Peters, A. A., Jamaludin, S. y N., Yapa, K. T. D. S., Chalmers, S., Wiegmans, A. P., Lim, H. F., Milevskiy, M. J. G., Azimi, I., Davis, F. M., Northwood, K. S., Pera, E., Marcial, D. L., Dray, E., Waterhouse, N. J., Cabot, P. J., Gonda, T. J., Kenny, P. A., Brown, M. A., Khanna, K. K., … Monteith, G. R. (2017). Oncosis and apoptosis induction by activation of an

overexpressed ion channel in breast cancer cells. *Oncogene*, *36*(46), 6490–6500. https://doi.org/10.1038/onc.2017.234

Phillips, J. C., Hardy, D. J., Maia, J. D. C., Stone, J. E., Ribeiro, J. V., Bernardi, R. C., Buch, R., Fiorin, G., Hénin, J., Jiang, W., McGreevy, R., Melo, M. C. R., Radak, B. K., Skeel, R. D., Singharoy, A., Wang, Y., Roux, B., Aksimentiev, A., Luthey-Schulten, Z., … Tajkhorshid, E. (2020). Scalable molecular dynamics on CPU and GPU architectures with NAMD. *The Journal of Chemical Physics*, *153*(4), 044130. https://doi.org/10.1063/5.0014475

Pietak, A., & Levin, M. (2016). Exploring instructive physiological signaling with the bioelectric tissue simulation engine. *Frontiers in Bioengineering and Biotechnology*, *4*. https://doi.org/10.3389/fbioe.2016.00055

Pietak, A., & Levin, M. (2017). Bioelectric gene and reaction networks: Computational modelling of genetic, biochemical and bioelectrical dynamics in pattern regulation. *Journal of The Royal Society Interface*, *14*(134), 20170425. https://doi.org/10.1098/rsif.2017.0425

Pleuvry, B. J. (2004). Receptors, agonists and antagonists. *Anaesthesia & Intensive Care Medicine*, *5*(10), 350–352. https://doi.org/10.1383/anes.5.10.350.52312

Polley, E. C., & van der Laan, M. J. (2010). Super learner in prediction. *Collection of Biostatistics Research Archive*. https://doi.org/https://biostats.bepress.com/ucbbiostat/paper266

Rao, M. S., Gupta, R., Liguori, M. J., Hu, M., Huang, X., Mantena, S. R., Mittelstadt, S. W., Blomme, E. A. G., & Van Vleet, T. R. (2019). Novel computational approach to predict off-target interactions for small molecules. *Frontiers in Big Data*, *2*. https://doi.org/10.3389/fdata.2019.00025

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-Validation. In *Encyclopedia of Database Systems* (pp. 532–538). Springer US. http://dx.doi.org/10.1007/978-0-387-39940-9_565

Reif, M., & Shafait, F. (2014). Efficient feature size reduction via predictive forward selection. *Pattern Recognition*, *47*(4), 1664–1673. https://doi.org/10.1016/j.patcog.2013.10.009

Robinson, N. B., Krieger, K., Khan, F. M., Huffman, W., Chang, M., Naik, A., Yongle, R., Hameed, I., Krieger, K., Girardi, L. N., & Gaudino, M. (2019). The current state of animal models in research: A review. *International Journal of Surgery*, *72*, 9–13. https://doi.org/10.1016/j.ijsu.2019.10.015

Rogers, D., & Hahn, M. (2010). Extended-Connectivity fingerprints. *Journal of Chemical Information and Modeling*, *50*(5), 742–754. https://doi.org/10.1021/ci100050t

Russo, D. P., Zorn, K. M., Clark, A. M., Zhu, H., & Ekins, S. (2018). Comparing multiple machine learning algorithms and metrics for estrogen receptor binding prediction. *Molecular Pharmaceutics*, *15*(10), 4361–4370. https://doi.org/10.1021/acs.molpharmaceut.8b00546

Sabe, V. T., Ntombela, T., Jhamba, L. A., Maguire, G. E. M., Govender, T., Naicker, T., & Kruger, H. G. (2021). Current trends in computer aided drug design and a highlight of drugs discovered via computational techniques: A review. *European Journal of Medicinal Chemistry*, *224*, 113705. https://doi.org/10.1016/j.ejmech.2021.113705

Schwarz, S., Gockel, L. M., Naggi, A., Barash, U., Gobec, M., Bendas, G., & Schlesinger, M. (2020). Glycosaminoglycans as tools to decipher the platelet tumor cell interaction: A focus on p-selectin. *Molecules*, *25*(5), 1039. https://doi.org/10.3390/molecules25051039

Silver, B. B., & Nelson, C. M. (2018). The Bioelectric Code: Reprogramming Cancer and Aging From the Interface of Mechanical and Chemical Microenvironments. *Frontiers in Cell and Developmental Biology*, *6*. https://doi.org/10.3389/fcell.2018.00021

Srivastava, P., Kane, A., Harrison, C., & Levin, M. (2021). A meta-analysis of bioelectric data in cancer, embryogenesis, and regeneration. *Bioelectricity*, *3*(1), 42–67. https://doi.org/10.1089/bioe.2019.0034

Sun, D., Gao, W., Hu, H., & Zhou, S. (2022). Why 90% of clinical drug development fails and how to improve it? *Acta Pharmaceutica Sinica B*, *12*(7), 3049–3062. https://doi.org/10.1016/j.apsb.2022.02.002

Suthaharan, S. (2016). Support vector machine. In *Machine Learning Models and Algorithms for Big Data Classification* (pp. 207–235). Springer US. http://dx.doi.org/10.1007/978-1-4899-7641-3_9

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, *39*(1), 43–62. https://doi.org/10.1016/s0169-7439(97)00061-0

Syed, Y. Y. (2019). Selinexor: First global approval. *Drugs*, *79*(13), 1485–1494. https://doi.org/10.1007/s40265-019-01188-9

Tuszynski, J., Tilli, T. M., & Levin, M. (2017). Ion channel and neurotransmitter modulators as electroceutical approaches to the control of cancer. *Current Pharmaceutical Design*, *23*(32), 4827–4841. https://doi.org/10.2174/1381612823666170530105837

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., & Zhao, S. (2019). Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, *18*(6), 463–477. https://doi.org/10.1038/s41573-019-0024-5

van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner. *Statistical Applications in Genetics and Molecular Biology*, *6*(1). https://doi.org/10.2202/1544-6115.1309

Van Norman, G. A. (2019). Limitations of animal studies for predicting toxicity in clinical trials. *JACC: Basic to Translational Science*, *4*(7), 845–854. https://doi.org/10.1016/j.jacbts.2019.10.008

Wang, J., Lu, Z., Wu, C., Li, Y., Kong, Y., Zhou, R., Shi, K., Guo, J., Li, N., Liu, J., Song, W., Wang, H., Zhu, M., & Xu, H. (2018). Evaluation of the anticancer and anti-metastasis effects of novel synthetic sodium channel blockers in prostate cancer cells in vitro and in vivo. *The Prostate*, *79*(1), 62–72. https://doi.org/10.1002/pros.23711

Xu, M., Watanachaturaporn, P., Varshney, P., & Arora, M. (2005). Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, *97*(3), 322–336. https://doi.org/10.1016/j.rse.2005.05.008

Yang, C., Chen, E. A., & Zhang, Y. (2022). Protein–Ligand docking in the machine-learning era. *Molecules*, *27*(14), 4568. https://doi.org/10.3390/molecules27144568

Zhang, H. (2004). The Optimality of Naive Bayes. *The Florida AI Research Society*.

Zhang, L. (2010). Glycosaminoglycan (GAG) biosynthesis and gag-binding proteins. In *Progress in Molecular Biology and Translational Science* (pp. 1–17). Elsevier. http://dx.doi.org/10.1016/s1877-1173(10)93001-9

Zhang, X., Lin, L., Huang, H., & Linhardt, R. J. (2019). Chemoenzymatic synthesis of glycosaminoglycans. *Accounts of Chemical Research*, *53*(2), 335–346. https://doi.org/10.1021/acs.accounts.9b00420

Zhao, Z., Ukidve, A., Kim, J., & Mitragotri, S. (2020). Targeting strategies for tissue-specific drug delivery. *Cell*, *181*(1), 151–167. https://doi.org/10.1016/j.cell.2020.02.001

Appendix

Appendix Table 1: All 435 features generated from MOE. The code represents the name of the feature, the class denotes whether the given feature is a 2D, internal 3d (i3D) or external 3d (x3D) type.

Code	Class	Description
AM1_dipole	i3D	Dipole moment
AM1_E	i3D	Total energy (kcal/mol)
AM1_Eele	i3D	Electronic energy (kcal/mol)
AM1_HF	i3D	Heat of formation (kcal)
AM1_HOMO	i3D	HOMO energy (eV)
AM1_IP	i3D	Ionization potential (eV)
AM1_LUMO	i3D	LUMO energy (eV)
apol	2D	Sum of atomic polarizabilities
ASA	i3D	Water accessible surface area
ASA+	i3D	Positive accessible surface area
ASA-	i3D	Negative accessible surface area
ASA_H	i3D	Total hydrophobic surface area
ASA_P	i3D	Total polar surface area
ast_fraglike	2D	Astex Fragment-like Test
ast_fraglike_ext	2D	Astex Fragment-like Test (Extended)
ast_violation	2D	Astex Fragment-like Violation Count
ast_violation_ext	2D	Astex Fragment-like Violation Count (Extended)
a_acc	2D	Number of H-bond acceptor atoms
a_acid	2D	Number of acidic atoms
a_aro	2D	Number of aromatic atoms

a_base	2D	Number of basic atoms
a_count	2D	Number of atoms
a_don	2D	Number of H-bond donor atoms
a_donacc	2D	Number of H-bond donor + acceptor atoms
a_heavy	2D	Number of heavy atoms
a_hyd	2D	Number of hydrophobic atoms
a_IC	2D	Atom information content (total)
a_ICM	2D	Atom information content (mean)
a_nB	2D	Number of boron atoms
a_nBr	2D	Number of bromine atoms
a_nC	2D	Number of carbon atoms
a_nCl	2D	Number of chlorine atoms
a_nF	2D	Number of fluorine atoms
a_nH	2D	Number of hydrogen atoms
a_nI	2D	Number of iodine atoms
a_nN	2D	Number of nitrogen atoms
a_nO	2D	Number of oxygen atoms
a_nP	2D	Number of phosphorus atoms
a_nS	2D	Number of sulfur atoms
balabanJ	2D	Balaban averaged distance sum connectivity
BCUT_PEOE_0	2D	PEOE Charge BCUT (0/3)
BCUT_PEOE_1	2D	PEOE Charge BCUT (1/3)
BCUT_PEOE_2	2D	PEOE Charge BCUT (2/3)
BCUT_PEOE_3	2D	PEOE Charge BCUT (3/3)
BCUT_SLOGP_0	2D	LogP BCUT (0/3)

BCUT_SLOGP_1	2D	LogP BCUT (1/3)
BCUT_SLOGP_2	2D	LogP BCUT (2/3)
BCUT_SLOGP_3	2D	LogP BCUT (3/3)
BCUT_SMR_0	2D	Molar Refractivity BCUT (0/3)
BCUT_SMR_1	2D	Molar Refractivity BCUT (1/3)
BCUT_SMR_2	2D	Molar Refractivity BCUT (2/3)
BCUT_SMR_3	2D	Molar Refractivity BCUT (3/3)
bpol	2D	Difference of bonded atom polarizabilities
b_1rotN	2D	Number of rotatable single bonds
b_1rotR	2D	Fraction of rotatable single bonds
b_ar	2D	Number of aromatic bonds
b_count	2D	Number of bonds
b_double	2D	Number of double bonds
b_heavy	2D	Number of heavy-heavy bonds
b_max1len	2D	Maximum single-bond chain length
b_rotN	2D	Number of rotatable bonds
b_rotR	2D	Fraction of rotatable bonds
b_single	2D	Number of single bonds
b_triple	2D	Number of triple bonds
CASA+	i3D	Charge-weighted positive surface area
CASA-	i3D	Charge-weighted negative surface area
chi0	2D	Atomic connectivity index (order 0)
chi0v	2D	Atomic valence connectivity index (order 0)
chi0v_C	2D	Carbon valence connectivity index (order 0)
chi0_C	2D	Carbon connectivity index (order 0)

chi1	2D	Atomic connectivity index (order 1)
chi1v	2D	Atomic valence connectivity index (order 1)
chi1v_C	2D	Carbon valence connectivity index (order 1)
chi1_C	2D	Carbon connectivity index (order 1)
chiral	2D	Number of chiral centers
chiral_u	2D	Number of unconstrained chiral centers
DASA	i3D	Absolute difference in surface area
DCASA	i3D	Absolute difference in charge-weighted areas
dens	i3D	Mass density (AMU/A^3)
density	2D	Mass density (AMU/A**3)
diameter	2D	Largest vertex eccentricity in graph
dipole	i3D	Dipole moment
dipoleX	x3D	Dipole moment (X)
dipoleY	x3D	Dipole moment (Y)
dipoleZ	x3D	Dipole moment (Z)
E	i3D	Potential Energy
E_ang	i3D	Angle Bend Energy
E_ele	i3D	Electrostatic energy
E_nb	i3D	Non-bonded energy
E_oop	i3D	Out-of-plane Energy
E_rele	x3D	Electrostatic Interaction Energy
E_rnb	x3D	Non-bonded Interaction Energy
E_rsol	x3D	Solvation Correction Difference
E_rvdw	x3D	Van der Waals Interaction Energy
E_sol	i3D	Solvation energy

E_stb	i3D	Stretch-bend energy
E_str	i3D	Bond stretch energy
E_strain	i3D	E minus energy of local minimum
E_tor	i3D	Torsion energy
E_vdw	i3D	Van der Waals energy
FASA+	i3D	Fractional positive accessible surface area
FASA-	i3D	Fractional negative accessible surface area
FASA_H	i3D	Fractional hydrophobic surface area
FASA_P	i3D	Fractional polar surface area
FCASA+	i3D	Fractional charge-weighted positive surface area
FCASA-	i3D	Fractional charge-weighted negative surface area
FCharge	2D	Sum of formal charges
GCUT_PEOE_0	2D	PEOE Charge GCUT (0/3)
GCUT_PEOE_1	2D	PEOE Charge GCUT (1/3)
GCUT_PEOE_2	2D	PEOE Charge GCUT (2/3)
GCUT_PEOE_3	2D	PEOE Charge GCUT (3/3)
GCUT_SLOGP_0	2D	LogP GCUT (0/3)
GCUT_SLOGP_1	2D	LogP GCUT (1/3)
GCUT_SLOGP_2	2D	LogP GCUT (2/3)
GCUT_SLOGP_3	2D	LogP GCUT (3/3)
GCUT_SMR_0	2D	Molar Refractivity GCUT (0/3)
GCUT_SMR_1	2D	Molar Refractivity GCUT (1/3)
GCUT_SMR_2	2D	Molar Refractivity GCUT (2/3)
GCUT_SMR_3	2D	Molar Refractivity GCUT (3/3)
glob	i3D	Molecular globularity

h_ema	2D	Sum of EHT acceptor strengths
h_emd	2D	Sum of EHT donor strengths
h_emd_C	2D	Sum of EHT carbon donor strengths
h_logD	2D	Octanol/water distribution coefficient (pH=7)
h_logP	2D	Octanol/water partition coefficient
h_logS	2D	Log solubility in water
h_log_dbo	2D	Sum of log (1 + d-bond orders)
h_log_pbo	2D	Sum of log (1 + p-bond orders)
h_mr	2D	Molar Refractivity
h_pavgQ	2D	Average total charge (pH=7)
h_pKa	2D	Acidity (pH=7)
h_pKb	2D	Basicity (pH=7)
h_pstates	2D	Entropic state count (pH=7)
h_pstrain	2D	Protonation state strain energy (pH=7)
Kier1	2D	First kappa shape index
Kier2	2D	Second kappa shape index
Kier3	2D	Third kappa shape index
KierA1	2D	First alpha modified shape index
KierA2	2D	Second alpha modified shape index
KierA3	2D	Third alpha modified shape index
KierFlex	2D	Molecular flexibility
lip_acc	2D	Lipinski Acceptor Count
lip_don	2D	Lipinski Donor Count
lip_druglike	2D	Lipinski Druglike Test
lip_violation	2D	Lipinski Violation Count

logP(o/w)	2D	Log octanol/water partition coefficient
logS	2D	Log Solubility in Water
MNDO_dipole	i3D	Dipole moment
MNDO_E	i3D	Total energy (kcal/mol)
MNDO_Eele	i3D	Electronic energy (kcal/mol)
MNDO_HF	i3D	Heat of formation (kcal)
MNDO_HOMO	i3D	HOMO energy (eV)
MNDO_IP	i3D	Ionization potential (eV)
MNDO_LUMO	i3D	LUMO energy (eV)
mr	2D	Molar refractivity
mutagenic	2D	Mutagenicity
nmol	2D	Number of molecules
npr1	i3D	Normalized PMI ratio (1) (pmi1 / pmi3)
npr2	i3D	Normalized PMI ratio (2) (pmi2 / pmi3)
opr_brigid	2D	Oprea Rigid Bond Count
opr_leadlike	2D	Oprea Leadlike Test
opr_nring	2D	Oprea Ring Count
opr_nrot	2D	Oprea Rotatable Bond Count
opr_violation	2D	Oprea Violation Count
PC+	2D	Total positive partial charge
PC-	2D	Total negative partial charge
PEOE_PC+	2D	Total positive partial charge
PEOE_PC-	2D	Total negative partial charge
PEOE_RPC+	2D	Relative positive partial charge
PEOE_RPC-	2D	Relative negative partial charge

PEOE_VSA+0	2D	Total positive 0 vdw surface area
PEOE_VSA+1	2D	Total positive 1 vdw surface area
PEOE_VSA+2	2D	Total positive 2 vdw surface area
PEOE_VSA+3	2D	Total positive 3 vdw surface area
PEOE_VSA+4	2D	Total positive 4 vdw surface area
PEOE_VSA+5	2D	Total positive 5 vdw surface area
PEOE_VSA+6	2D	Total positive 6 vdw surface area
PEOE_VSA-0	2D	Total negative 0 vdw surface area
PEOE_VSA-1	2D	Total negative 1 vdw surface area
PEOE_VSA-2	2D	Total negative 2 vdw surface area
PEOE_VSA-3	2D	Total negative 3 vdw surface area
PEOE_VSA-4	2D	Total negative 4 vdw surface area
PEOE_VSA-5	2D	Total negative 5 vdw surface area
PEOE_VSA-6	2D	Total negative 6 vdw surface area
PEOE_VSA_FHYD	2D	Fractional hydrophobic vdw surface area
PEOE_VSA_FNEG	2D	Fractional negative vdw surface area
PEOE_VSA_FPNEG	2D	Fractional polar negative vdw surface area
PEOE_VSA_FPOL	2D	Fractional polar vdw surface area
PEOE_VSA_FPOS	2D	Fractional positive vdw surface area
PEOE_VSA_FPPOS	2D	Fractional polar positive vdw surface area
PEOE_VSA_HYD	2D	Total hydrophobic vdw surface area
PEOE_VSA_NEG	2D	Total negative vdw surface area
PEOE_VSA_PNEG	2D	Total polar negative vdw surface area
PEOE_VSA_POL	2D	Total polar vdw surface area
PEOE_VSA_POS	2D	Total positive vdw surface area

PEOE_VSA_PPOS	2D	Total polar positive vdw surface area
petitjean	2D	(diameter - radius) / diameter
petitjeanSC	2D	(diameter - radius) / radius
PM3_dipole	i3D	Dipole moment
PM3_E	i3D	Total energy (kcal/mol)
PM3_Eele	i3D	Electronic energy (kcal/mol)
PM3_HF	i3D	Heat of formation (kcal)
PM3_HOMO	i3D	HOMO energy (eV)
PM3_IP	i3D	Ionization potential (eV)
PM3_LUMO	i3D	LUMO energy (eV)
pmi	i3D	Principal moment of inertia
pmi1	i3D	Principal moment of inertia (1)
pmi2	i3D	Principal moment of inertia (2)
pmi3	i3D	Principal moment of inertia (3)
pmiX	x3D	Principal moment of inertia (X)
pmiY	x3D	Principal moment of inertia (Y)
pmiZ	x3D	Principal moment of inertia (Z)
pro_app_charge	Protein	Protein Charge at Debye Length
pro_asa_hph	Protein	Hydrophilic Surface Area
pro_asa_hyd	Protein	Hydrophobic Surface Area
pro_asa_vdw	Protein	Accessible Surface Area (Water Probe)
pro_coeff_280	Protein	Extinction coefficient at 280 nm
pro_coeff_diff	Protein	Diffusion Coefficient
pro_coeff_fric	Protein	Frictional Coefficient
pro_debye	Protein	Debye Screening Length

pro_dipole_moment	Protein	Protein Dipole Moment
pro_eccen	Protein	Protein Eccentricity
pro_helicity	Protein	Protein Helix Ratio
pro_henry	Protein	Henry's Function f(ka)
pro_hyd_moment	Protein	Hydrophobicity Moment
pro_mass	Protein	Protein Mass in kDa
pro_mobility	Protein	Protein Mobility
pro_net_charge	Protein	Protein Net Charge
pro_patch_cdr_hyd	Protein	Area of hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_1	Protein	Area of largest hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_2	Protein	Area of 2 largest hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_3	Protein	Area of 3 largest hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_4	Protein	Area of 4 largest hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_5	Protein	Area of 5 largest hydrophobic protein patch(es) near CDRs
pro_patch_cdr_hyd_n	Protein	Count of hydrophobic protein patch(es) near CDRs
pro_patch_cdr_ion	Protein	Area of ionic protein patch(es) near CDRs
pro_patch_cdr_ion_1	Protein	Area of largest ionic protein patch(es) near CDRs
pro_patch_cdr_ion_2	Protein	Area of 2 largest ionic protein patch(es) near CDRs
pro_patch_cdr_ion_3	Protein	Area of 3 largest ionic protein patch(es) near CDRs
pro_patch_cdr_ion_4	Protein	Area of 4 largest ionic protein patch(es) near CDRs
pro_patch_cdr_ion_5	Protein	Area of 5 largest ionic protein patch(es) near CDRs

pro_patch_cdr_ion_n	Protein	Count of ionic protein patch(es) near CDRs
pro_patch_cdr_neg	Protein	Area of negative protein patch(es) near CDRs
pro_patch_cdr_neg_1	Protein	Area of largest negative protein patch(es) near CDRs
pro_patch_cdr_neg_2	Protein	Area of 2 largest negative protein patch(es) near CDRs
pro_patch_cdr_neg_3	Protein	Area of 3 largest negative protein patch(es) near CDRs
pro_patch_cdr_neg_4	Protein	Area of 4 largest negative protein patch(es) near CDRs
pro_patch_cdr_neg_5	Protein	Area of 5 largest negative protein patch(es) near CDRs
pro_patch_cdr_neg_n	Protein	Count of negative protein patch(es) near CDRs
pro_patch_cdr_pos	Protein	Area of positive protein patch(es) near CDRs
pro_patch_cdr_pos_1	Protein	Area of largest positive protein patch(es) near CDRs
pro_patch_cdr_pos_2	Protein	Area of 2 largest positive protein patch(es) near CDRs
pro_patch_cdr_pos_3	Protein	Area of 3 largest positive protein patch(es) near CDRs
pro_patch_cdr_pos_4	Protein	Area of 4 largest positive protein patch(es) near CDRs
pro_patch_cdr_pos_5	Protein	Area of 5 largest positive protein patch(es) near CDRs
pro_patch_cdr_pos_n	Protein	Count of positive protein patch(es) near CDRs
pro_patch_hyd	Protein	Area of hydrophobic protein patch(es)
pro_patch_hyd_1	Protein	Area of largest hydrophobic protein patch(es)
pro_patch_hyd_2	Protein	Area of 2 largest hydrophobic protein patch(es)
pro_patch_hyd_3	Protein	Area of 3 largest hydrophobic protein patch(es)
pro_patch_hyd_4	Protein	Area of 4 largest hydrophobic protein patch(es)
pro_patch_hyd_5	Protein	Area of 5 largest hydrophobic protein patch(es)
pro_patch_hyd_n	Protein	Count of hydrophobic protein patch(es)
pro_patch_ion	Protein	Area of ionic protein patch(es)
pro_patch_ion_1	Protein	Area of largest ionic protein patch(es)
pro_patch_ion_2	Protein	Area of 2 largest ionic protein patch(es)

pro_patch_ion_3	Protein	Area of 3 largest ionic protein patch(es)
pro_patch_ion_4	Protein	Area of 4 largest ionic protein patch(es)
pro_patch_ion_5	Protein	Area of 5 largest ionic protein patch(es)
pro_patch_ion_n	Protein	Count of ionic protein patch(es)
pro_patch_neg	Protein	Area of negative protein patch(es)
pro_patch_neg_1	Protein	Area of largest negative protein patch(es)
pro_patch_neg_2	Protein	Area of 2 largest negative protein patch(es)
pro_patch_neg_3	Protein	Area of 3 largest negative protein patch(es)
pro_patch_neg_4	Protein	Area of 4 largest negative protein patch(es)
pro_patch_neg_5	Protein	Area of 5 largest negative protein patch(es)
pro_patch_neg_n	Protein	Count of negative protein patch(es)
pro_patch_pos	Protein	Area of positive protein patch(es)
pro_patch_pos_1	Protein	Area of largest positive protein patch(es)
pro_patch_pos_2	Protein	Area of 2 largest positive protein patch(es)
pro_patch_pos_3	Protein	Area of 3 largest positive protein patch(es)
pro_patch_pos_4	Protein	Area of 4 largest positive protein patch(es)
pro_patch_pos_5	Protein	Area of 5 largest positive protein patch(es)
pro_patch_pos_n	Protein	Count of positive protein patch(es)
pro_pI_3D	Protein	Structure-based pI Prediction
pro_pI_seq	Protein	Sequence-based pI Prediction
pro_r_gyr	Protein	Radius of Gyration
pro_r_solv	Protein	Hydrodynamic Radius
pro_sed_const	Protein	Sedimentation Constant
pro_volume	Protein	Protein Volume
pro_zdipole	Protein	Zeta Dipole Moment

pro_zeta	Protein	Zeta potential at Debye Length
pro_zquadrupole	Protein	Zeta Quadrupole Moment
Q_PC+	2D	Total positive partial charge
Q_PC-	2D	Total negative partial charge
Q_RPC+	2D	Relative positive partial charge
Q_RPC-	2D	Relative negative partial charge
Q_VSA_FHYD	2D	Fractional hydrophobic vdw surface area
Q_VSA_FNEG	2D	Fractional negative vdw surface area
Q_VSA_FPNEG	2D	Fractional polar negative vdw surface area
Q_VSA_FPOL	2D	Fractional polar vdw surface area
Q_VSA_FPOS	2D	Fractional positive vdw surface area
Q_VSA_FPPOS	2D	Fractional polar positive vdw surface area
Q_VSA_HYD	2D	Total hydrophobic vdw surface area
Q_VSA_NEG	2D	Total negative vdw surface area
Q_VSA_PNEG	2D	Total polar negative vdw surface area
Q_VSA_POL	2D	Total polar vdw surface area
Q_VSA_POS	2D	Total positive vdw surface area
Q_VSA_PPOS	2D	Total polar positive vdw surface area
radius	2D	Smallest vertex eccentricity in graph
reactive	2D	Reactivity
rgyr	i3D	Radius of gyration
rings	2D	Number of rings
RPC+	2D	Relative positive partial charge
RPC-	2D	Relative negative partial charge
rsynth	2D	Synthetic Feasibility

SlogP	2D	Log Octanol/Water Partition Coefficient
SlogP_VSA0	2D	Bin 0 SlogP (-10 ,-0.40]
SlogP_VSA1	2D	Bin 1 SlogP (-0.40,-0.20]
SlogP_VSA2	2D	Bin 2 SlogP (-0.20, 0.00]
SlogP_VSA3	2D	Bin 3 SlogP (0.00, 0.10]
SlogP_VSA4	2D	Bin 4 SlogP (0.10, 0.15]
SlogP_VSA5	2D	Bin 5 SlogP (0.15, 0.20]
SlogP_VSA6	2D	Bin 6 SlogP (0.20, 0.25]
SlogP_VSA7	2D	Bin 7 SlogP (0.25, 0.30]
SlogP_VSA8	2D	Bin 8 SlogP (0.30, 0.40]
SlogP_VSA9	2D	Bin 9 SlogP (0.40,10]
SMR	2D	Molar Refractivity
SMR_VSA0	2D	Bin 0 SMR (0.000,0.110]
SMR_VSA1	2D	Bin 1 SMR (0.110,0.260]
SMR_VSA2	2D	Bin 2 SMR (0.260,0.350]
SMR_VSA3	2D	Bin 3 SMR (0.350,0.390]
SMR_VSA4	2D	Bin 4 SMR (0.390,0.440]
SMR_VSA5	2D	Bin 5 SMR (0.440,0.485]
SMR_VSA6	2D	Bin 6 SMR (0.485,0.560]
SMR_VSA7	2D	Bin 7 SMR (0.560,10]
std_dim1	i3D	Standard dimension 1
std_dim2	i3D	Standard dimension 2
std_dim3	i3D	Standard dimension 3
TPSA	2D	Topological Polar Surface Area (A**2)
VAdjEq	2D	Vertex adjacency information (equal)

VAdjMa	2D	Vertex adjacency information (mag)
VDistEq	2D	Vertex distance equality index
VDistMa	2D	Vertex distance magnitude index
vdw_area	2D	Van der Waals surface area (A**2)
vdw_vol	2D	Van der Waals volume (A**3)
vol	i3D	Van der Waals volume
VSA	i3D	Van der Waals surface area
vsa_acc	2D	VDW acceptor surface area (A**2)
vsa_acid	2D	VDW acidic surface area (A**2)
vsa_base	2D	VDW basic surface area (A**2)
vsa_don	2D	VDW donor surface area (A**2)
vsa_hyd	2D	VDW hydrophobe surface area (A**2)
vsa_other	2D	VDW other surface area (A**2)
vsa_pol	2D	VDW polar surface area (A**2)
vsurf_A	i3D	Amphiphilic moment
vsurf_CP	i3D	Critical packing parameter
vsurf_CW1	i3D	Capacity factor at -0.2
vsurf_CW2	i3D	Capacity factor at -0.5
vsurf_CW3	i3D	Capacity factor at -1.0
vsurf_CW4	i3D	Capacity factor at -2.0
vsurf_CW5	i3D	Capacity factor at -3.0
vsurf_CW6	i3D	Capacity factor at -4.0
vsurf_CW7	i3D	Capacity factor at -5.0
vsurf_CW8	i3D	Capacity factor at -6.0
vsurf_D1	i3D	Hydrophobic volume at -0.2

vsurf_D2	i3D	Hydrophobic volume at -0.4
vsurf_D3	i3D	Hydrophobic volume at -0.6
vsurf_D4	i3D	Hydrophobic volume at -0.8
vsurf_D5	i3D	Hydrophobic volume at -1.0
vsurf_D6	i3D	Hydrophobic volume at -1.2
vsurf_D7	i3D	Hydrophobic volume at -1.4
vsurf_D8	i3D	Hydrophobic volume at -1.6
vsurf_DD12	i3D	vsurf_EDmin1, vsurf_EDmin2 distance
vsurf_DD13	i3D	vsurf_EDmin1, vsurf_EDmin3 distance
vsurf_DD23	i3D	vsurf_EDmin2, vsurf_EDmin3 distance
vsurf_DW12	i3D	vsurf_EWmin1, vsurf_EWmin2 distance
vsurf_DW13	i3D	vsurf_EWmin1, vsurf_EWmin3 distance
vsurf_DW23	i3D	vsurf_EWmin2, vsurf_EWmin3 distance
vsurf_EDmin1	i3D	Lowest hydrophobic energy
vsurf_EDmin2	i3D	2nd lowest hydrophobic energy
vsurf_EDmin3	i3D	3rd lowest hydrophobic energy
vsurf_EWmin1	i3D	Lowest hydrophilic energy
vsurf_EWmin2	i3D	2nd lowest hydrophilic energy
vsurf_EWmin3	i3D	3rd lowest hydrophilic energy
vsurf_G	i3D	Surface globularity
vsurf_HB1	i3D	H-bond donor capacity at -0.2
vsurf_HB2	i3D	H-bond donor capacity at -0.5
vsurf_HB3	i3D	H-bond donor capacity at -1.0
vsurf_HB4	i3D	H-bond donor capacity at -2.0
vsurf_HB5	i3D	H-bond donor capacity at -3.0

vsurf_HB6	i3D	H-bond donor capacity at -4.0
vsurf_HB7	i3D	H-bond donor capacity at -5.0
vsurf_HB8	i3D	H-bond donor capacity at -6.0
vsurf_HL1	i3D	First hydrophilic-lipophilic balance
vsurf_HL2	i3D	Second hydrophilic-lipophilic balance
vsurf_ID1	i3D	Hydrophobic integy moment at -0.2
vsurf_ID2	i3D	Hydrophobic integy moment at -0.4
vsurf_ID3	i3D	Hydrophobic integy moment at -0.6
vsurf_ID4	i3D	Hydrophobic integy moment at -0.8
vsurf_ID5	i3D	Hydrophobic integy moment at -1.0
vsurf_ID6	i3D	Hydrophobic integy moment at -1.2
vsurf_ID7	i3D	Hydrophobic integy moment at -1.4
vsurf_ID8	i3D	Hydrophobic integy moment at -1.6
vsurf_IW1	i3D	Hydrophilic integy moment at -0.2
vsurf_IW2	i3D	Hydrophilic integy moment at -0.5
vsurf_IW3	i3D	Hydrophilic integy moment at -1.0
vsurf_IW4	i3D	Hydrophilic integy moment at -2.0
vsurf_IW5	i3D	Hydrophilic integy moment at -3.0
vsurf_IW6	i3D	Hydrophilic integy moment at -4.0
vsurf_IW7	i3D	Hydrophilic integy moment at -5.0
vsurf_IW8	i3D	Hydrophilic integy moment at -6.0
vsurf_R	i3D	Surface rugosity
vsurf_S	i3D	Interaction field area
vsurf_V	i3D	Interaction field volume
vsurf_W1	i3D	Hydrophilic volume at -0.2

vsurf_W2	i3D	Hydrophilic volume at -0.5
vsurf_W3	i3D	Hydrophilic volume at -1.0
vsurf_W4	i3D	Hydrophilic volume at -2.0
vsurf_W5	i3D	Hydrophilic volume at -3.0
vsurf_W6	i3D	Hydrophilic volume at -4.0
vsurf_W7	i3D	Hydrophilic volume at -5.0
vsurf_W8	i3D	Hydrophilic volume at -6.0
vsurf_Wp1	i3D	Polar volume at -0.2
vsurf_Wp2	i3D	Polar volume at -0.5
vsurf_Wp3	i3D	Polar volume at -1.0
vsurf_Wp4	i3D	Polar volume at -2.0
vsurf_Wp5	i3D	Polar volume at -3.0
vsurf_Wp6	i3D	Polar volume at -4.0
vsurf_Wp7	i3D	Polar volume at -5.0
vsurf_Wp8	i3D	Polar volume at -6.0
Weight	2D	Molecular weight (CRC)
weinerPath	2D	Weiner path number
weinerPol	2D	Weiner polarity number
zagreb	2D	Zagreb index

Appendix Table 2: Parameters and their corresponding values used during the conformational search of each compound's fragments.

Parameter	Value	Description
mmFailureLimit	30	"Maximum number of tries the algorithm has to generate a new conformation before stopping" (Chemical Computing Group, 2019a)
mmSuperposeRMSD	0.15	"If the root mean square distance (RMSD) between 2 conformations is less than or equal to this value, the conformations are considered to be the same" (Chemical Computing Group, 2019a)
mmGradientTestMM	0.01	"If the RMS gradient falls below this value then energy minimization will stop" (Chemical Computing Group, 2019a)
mmStrainLimit	7	"Any conformation whose strain energy is above this value is rejected" (Chemical Computing Group, 2019a)

Appendix Table 3: Parameters used in RapidMiner Studio for feature selection. Note that all settings were kept to the defaults.

Parameter	Value
selection direction	forward
limit generations without improval	true
generations without improval	1
limit number of generations	false
keep best	1
maximum number of generations	10
normalize weights	true
use local random seed	false
local random seed	1992
user result individual selection	false
show population plotter	false
plot generations	10
constraint draw range	false
draw dominated points	true
maximal fitness	Infinity

Appendix Table 4: Parameters used in RapidMiner Studio for the DECTRE learner. Note that all settings were kept to the defaults.

Parameter	Value
criterion	gain_ratio
maximal depth	10
apply pruning	true
confidence	0.1
apply prepruning	true
minimal gain	0.01
minimal leaf size	2
minimal size for split	4
number of prepruning alternatives	3

Appendix Table 5: Parameters used in RapidMiner Studio for the NAIBAY classifier. Note that all settings were kept to the defaults.

Parameter	Value
laplace correction	true

Appendix Table 6: Parameters used in RapidMiner Studio for the NEUNET learner. Note that all settings were kept to the defaults.

hidden layers	2
training cycles	200
learning rate	0.01
momentum	0.9
decay	false
shuffle	true
normalize	true
error epsilon	1.0E-4
use local random seed	false
local random seed	1992

Appendix Table 7: Parameters used in RapidMiner Studio for the RANFOR learner. Note that all settings were kept to the defaults.

Parameter	Value
number of trees	100
criterion	gain_ratio
maximal depth	10
apply prepruning	false
minimal gain	0.01
minimal leaf size	2
minimal size for split	4
number of prepruning alternatives	3
apply pruning	false
confidence	0.1
random splits	false
guess subset ratio	true
subset ratio	0.2
voting strategy	confidence vote
use local random seed	false
local random seed	1992
enable parallel execution	true

Appendix Table 8: Parameters used in RapidMiner Studio for the SVM (LibSVM) algorithm. Note that all settings were kept to the defaults.

svm type	C-SVC
kernel type	rbf
degree	3
gamma	0.0
coef0	0.0
C	0.0
nu	0.5
cache size	80
epsilon	0.001
p	0.1
class weights	list
shrinking	true
calculate confidences	false
confidences for multiclass	true