Analyzing small molecule inhibition of enzymes:
A preliminary machine learning approach towards drug lead generation

Pearl Philip

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Chemical Engineering

University of Washington

2017

Committee:

David A.C. Beck

Jim Pfaendtner

Program Authorized to Offer Degree:

Chemical Engineering

University of Washington

Abstract

Analyzing small molecule inhibition of enzymes:
A preliminary machine learning approach towards drug lead generation

Pearl Philip

Chair of the Supervisory Committee:
Professor David A.C. Beck
Chemical Engineering

This project is designed to create an implementation of quantitative structure-activity relationships (QSAR) models in Python for the prediction of inhibitory action of small-molecule drugs on the enzyme USP1 - an enzyme essential to DNA-repair in proliferating cancer cells. Molecular descriptors are calculated using PyChem and employed to characterize the properties of about 400,000 drug-like compounds from a high-throughput screening assay made available on PubChem. Multiple machine learning models are created on the training data using Scikit-learn and Theano after feature selection and processing, followed by a genetic algorithm to synthesize an ideal enzyme inhibitor to be tested for activity and use as a drug compound.

Higher error and poorer model fits can be attributed to multiple sources of error – measurement of activity using AC_{50}, imbalanced dataset in favor of molecules with zero inhibition, incomplete feature space, highly non-linear interactions between the enzyme and drug, and the attainment of local minima in hyperparameter optimization. Solutions have been suggested for each of these issues, and is proposed as a part of future work. The genetic

algorithm is used to synthesize a molecule in-silico and as the model prediction accuracy is increased, it can be pursued as a drug lead in clinical trials. This project provides a promising pipeline for future work in open-source molecular drug design and can be extended for use with other datasets and target species.

Table of Contents

List of Figures

List of Tables

Chapter 1
INTRODUCTION

1.1 MACHINE LEARNING IN THE CHEMICAL SCIENCES

Machine learning is the science of training computers to solve problems without being explicitly programmed to do so. The computer learns from data presented to it using algorithms and we thus impute to it a predictive and decision-making power. In the past decade, machine learning has given us spam filtering, economic forecasting, speech recognition, intelligent web search, and a greater understanding of the human genome[1].

In chemical engineering, machine learning is well-adapted to and developed for applications of process control in chemical plants, with artificial neural networks and reinforcement learning being the methods of choice[2]. Massive amounts of data are generated in academic and commercial chemical engineering settings that must be processed efficiently to make safe and informed decisions, and often hold more useful information than meets the eye. For example, manufacturing processes are highly complex and produce increasing amounts of data leading to a need for improved process monitoring, data analysis and fault detection[3]. Other examples of use of machine learning include prediction of boiling points,[4] carcinogenicities[5] and toxicities of substances[6]. It also has a long history in biomedical research, such as processing and diagnoses of cancers[7], drug design[8] and assessing biodegradability[9]. Smart, automated data analysis and pattern recognition are key to modern innovation.

Cheminformatics is the use of computational and informatics techniques to solve a variety of problems in chemistry and chemical engineering, particularly dealing with data collected from processes at the molecular scale. Unlike quantum chemistry or molecular dynamics and

simulation, which are based on physics, cheminformatics is intended to produce models that can usefully predict chemical and biological properties of compounds given the 2-D structure of a molecule. Cheminfomatics models use machine learning methods of regression, classification and clustering[10]. Support vector machines, nearest neighbors, random forest, multiple linear regression and artificial neural networks are some specific examples of learning models we use to train the computer to predict molecular activity. The accuracy of the prediction is retroactively used to optimize the model parameters for better performance, while seeking to avoid overfitting. The availability of molecular data from a wide chemical space, powerful computing technology, the use of a large and diverse selection of descriptors, and the development of sophisticated nonlinear machine learning algorithms have helped cheminformatics flourish since the 1990s.

1.2 QSAR AND ITS APPLICATIONS

Within the domain of cheminformatics, quantitative structure–activity relationship (QSAR) and quantitative structure–property relationship (QSPR) models are *in-silico* regression or classification models that find applications in chemical and biological sciences and engineering to relate the structural features of a compound or substance to its biological and physicochemical properties. QSAR models lay out a possible relationship between molecular structures and activity (or target properties) in a large set of biological or physicochemical data[11]. They then predict the properties of a new chemical, previously unknown to it. This predictive power can be captured to virtually analyze the properties of compounds before testing them in a laboratory. Some QSAR applications in chemical engineering are the identification of structural determinants for protein interactions with inorganic surfaces[12], cytolytic and antibiotic activity of peptides[13], molecular inhibition of enzymes[14], Log P_{liver} values for volatile organic compounds[15], and blood-brain barrier permeability.[16]

For an application, we may desire specific properties to be present in an ideal compound and, through traditional QSAR, we can derive the structural characteristics that it must possess to exhibit that property. The structural characteristics used to create the model are called features or descriptors, and a QSAR model could contain a few or even hundreds of descriptors, present in numerous combinations for the best prediction accuracy. Some examples of molecular descriptors are the sum of atomic charges, electric dipole moment, molecular weight, partial positive surface area, primary sp2 carbon count, average charge on carbonyl carbons, etc.[17] **Figure 1** traces how chemical data is gathered and features are defined to produce a dataset that is split for training and testing QSAR models. This is an iterative process as the workflow consists of multiple knobs that can be tuned for better prediction accuracies once a preliminary model has been created.

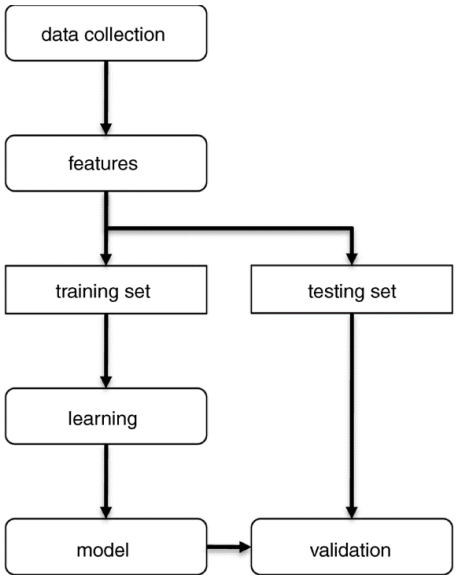


Figure 1: General workflow of developing a machine learning model to study QSAR

The success of machine learning in modern product development can be extended into laboratories across the world and help tackle complex data-intensive problems at every stage. Specifically, the field of drug design and testing can benefit immensely from such modeling. Cheminformatics is intertwined with drug discovery and therapeutics development, and thus,

bioactivity and ADMET (Absorption, Distribution, Metabolism, Excretion and Toxicity) properties are popularly approached by informatics techniques.[18] They can be used to understand pathways of drug action, design of new ones, predict any quantifiable physical property and screen libraries of chemicals.[19]

A pharmaceutical company keeps focus on the quality, safety, and efficacy of a marketed drug by conducting a variety of tests. Thus, the development process of a single drug is a considerable investment of time and money, often taking an average of 12 years for development and investments of about USD 868 million.[20,21] Large investment in traditional drug development is a result of low successful discovery rates. Estimates say that only 11% of developed drugs are approved by US and European boards.[22] The FDA claims that USD100 million could be saved while developing a single drug could be attained with a 10% prediction improvement, which is a game-changer for pharmaceutical economics.[23]

In-silico methods are increasingly being used to weed out potential failures in the pre-development process, and given time, strong statistical proof is bound to show that QSAR and related techniques shorten the life of a drug in a lab and decrease the funding required to make it consumer-ready.[8] The availability of high-throughput screening (HTS) data to the public in recent years allow research groups to collaborate over data by providing meaningful analysis and products from informationally-valuable results of experimental efforts. With the use of robots, detectors and software that control the whole process, a series of analyses of chemical compounds is conducted in a short time and the biological and chemical properties can be defined.

1.3 DATA SCIENCE FOR PREDICTION OF ENZYME INHIBITION

In this project, a quantitative high throughput screening (qHTS) dataset was procured for analysis from PubChem[24], made available by the University of Delaware and the NIH Chemical

Genomics Center in 2014. It records the inhibition activities of about 400,000 drug-like molecules against the deubiquitinating enzyme USP1.[25] Deubiquitinating enzymes (DUBs) are a class of enzymes that can cleave the isopeptide bond formed between the C-terminal carboxylate of ubiquitin and a lysine side-chain amino group on a target protein. Among them, ubiquitin specific proteases (USPs) form the largest DUB family. They are involved in chromatin remodeling, DNA repair, etc[26]. A model structure of the enzyme USP1 is depicted in **Figure 2**.

Human USPs are emerging as promising targets for pharmacological treatment because of their connection to several diseases, including prostate[27], colon and breast cancer[28], pediatric acute lymphoblastic leukemia[29], and familial cylindromatosis[30]. Given the upregulation of DNA repair in cancer cells, they recover well from damage sustained during cell division and chemotherapy. Among the human USPs, the deubiquitinating enzyme (DUB) USP1 occupies a special position as it is implicated in DNA damage response. One approach that could potentially cease this repair and cause cancer cells to die, is to inhibit USP1 from regulating the DNA repair pathways. The advantage of inhibiting USPs lies in the potential specificity of therapeutic intervention that can lead to better efficacy and reduce nonspecific side effects.

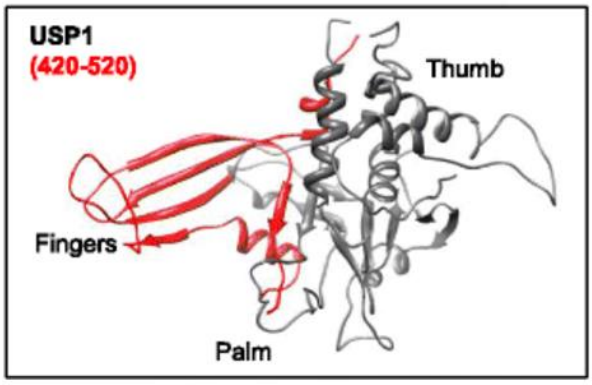


Figure 2: Modeled structure of USP1 catalytic domains using SWISS-MODEL. The Thumb, Palm and Fingers sub-domains are indicated. The UAF1-binding sites are highlighted in red (USP1 residues 420–520).[31]

To seek out better performing inhibitors for USP1, Wahi et. al used the dataset to create prediction models using the structural descriptors for the compounds.[32] The models "sniff" out molecules that could act as potential abrogators. They utilized four machine learning techniques - Naive Bayes, Random Forests, Sequential minimal optimization and J48 - and predicted inhibition in a test set with ~80% accuracy. In their work, owing to the larger size of the data file, they split it in to smaller parts for further processing. They used Mayachem tools for file splitting; PowerMV for statistical analysis, descriptor generation and molecular viewing; Perl scripts for splitting the training and testing data, Weka for feature selection and machine learning algorithms.

The primary set of descriptors used in their work were a 'best first guess' of three classes of descriptors, and ignored other classes of molecular features that could prove to be of statistical significance. Their work uses classification techniques and it would be interesting to explore the outcomes from a regression standpoint, since the data is amenable to both. While their work provides a performance comparison of the different methods, it doesn't provide us useful information about why the models perform as they do, which features have a statistically significant effect on the outcome and what the descriptor values for an ideal inhibitor might be. If the latter were described, pharmaceutical labs could pursue the actual synthesis and testing of that molecule, substance or one that is most like it. They also do not provide public access to their code or the assumptions in their methods, which hinders quick replicability and verification of their results. These problems are, unfortunately, commonplace in existing cheminformatics projects and can be addressed through intentional open-source development, analysis of well-performing molecules and descriptors, and a product-driven modeling process. Our project seeks to address each of these issues and ideas, to show that open-source development of cheminformatic models and resources can indeed provide quality computational results that serve as a starting point for laboratory work.

Chapter 2
MATERIALS AND METHODS

2.1 DATA COLLECTION AND FORMATTING

The data used in this project[24] was obtained from PubChem and consists of the molecular structures of 389,560 compounds in the form of simplified molecular-input line-entry system (SMILES)[33] and IUPAC International Chemical Identifier (InChI)[34] notation, their USP1-inhibition activities at various doses, and experimental variables. The SMILES and InChI formats describe the chemical species using a line notation of short ASCII strings, as shown in **Figure 3** for the molecule ciprofloxacin. SMILES strings can be imported by most molecule editors and converted back into 2-D and 3-D representations of the molecules.

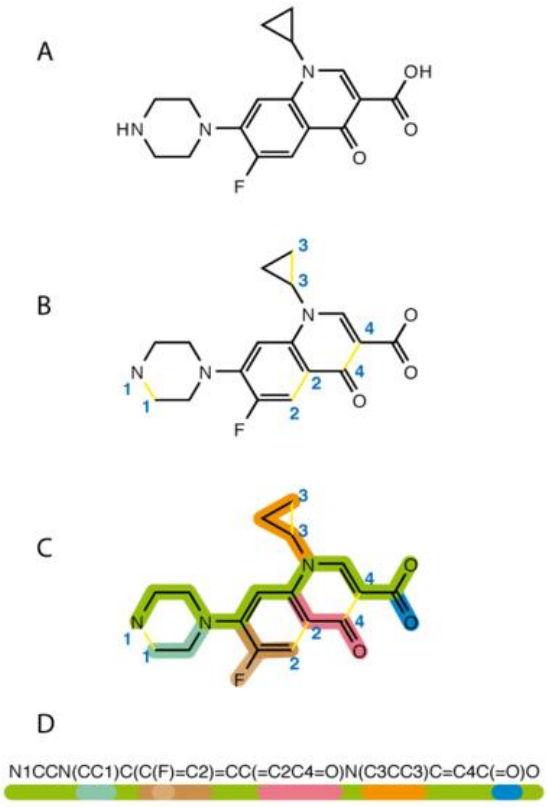


Figure 3: SMILES representation of the chemical compound ciprofloxacin, a fluoroquinolone antibiotic.[35]

In this dataset, compounds are first classified as having full titration curves, partial modulation, partial curve (weaker actives), single point activity (at highest concentration only), or inactive. Based on the single point activity at the reference concentration, an inhibition scale from 0 to 100 was developed and used to rank the compounds. For this assay, apparent inhibitors are ranked higher than compounds that showed apparent activation. For all inactive compounds, the PubChem activity score is 0. Active compounds have PubChem activity score between 40 and 100. Inconclusive compounds have PubChem activity score between 1 and 39.

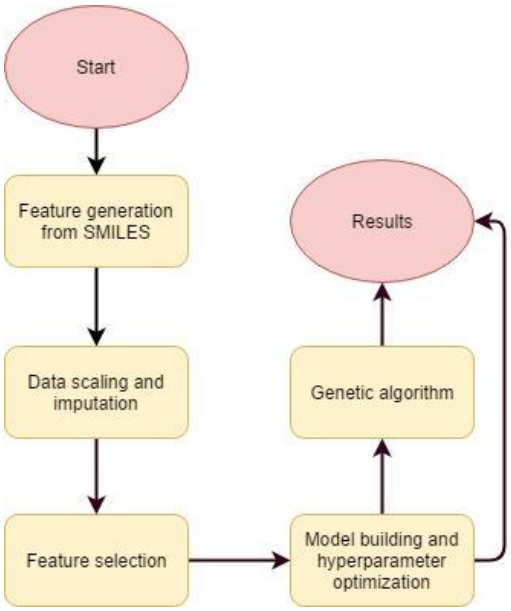
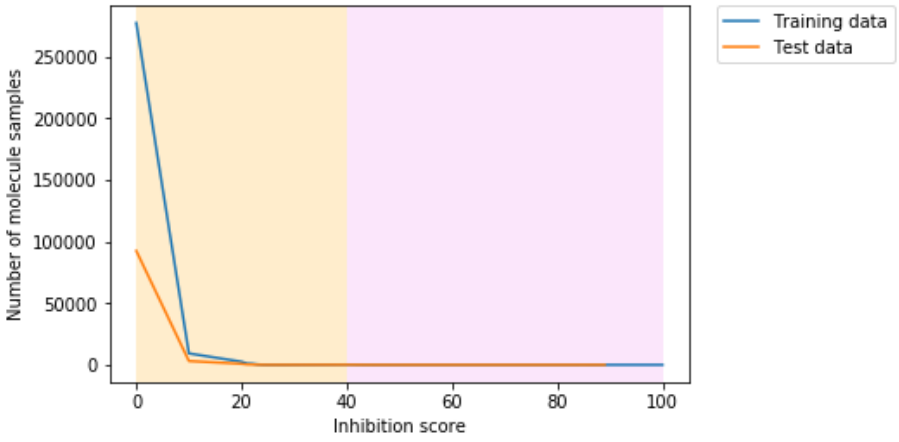


Figure 4:Flowchart showing the project pipeline.

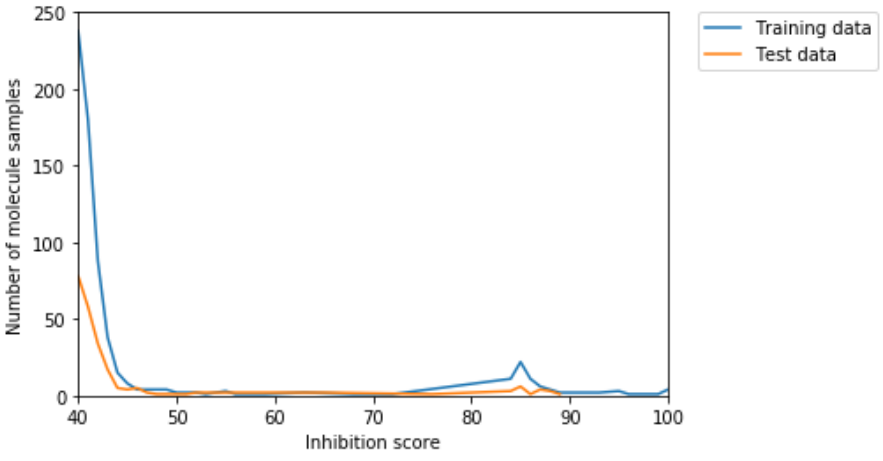
In our project, the data was processed and studied using Python 3.5, and the repository containing the scripts and the instructions to set up the environment for the entire Python project is located on GitHub[36]. The workflow in this project is shown in the flowchart above, in **Figure 4**. All the data and plots generated in this project are available on Amazon S3[37]. This was done to contribute to open resources for chemical engineering data analyses. In the 2016 version of the assay, there were two datasets available (now stored on Amazon S3[37]), one containing compound IDs and SMILES and InChI strings as a text file and the other containing compound IDs and

activity data. The two datasets are read in using Pandas v0.19.2 – a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. The activity data frame is then cleaned to remove unnecessary columns and header labels, and retain only the compound IDs and PubChem activity scores. The two data frames are joined over their compound IDs to form a data frame that contains only the SMILES strings and their respective activity scores.

Before working with the data, it is important to understand the distribution of inhibition scores in the dataset. **Figure 5a** shows the counts of samples with different inhibition scores (higher scores signifying better inhibition). It is immediately evident that inhibiting molecules are underrepresented in the dataset, and samples with zero inhibition score dominate the assay data. In **Figure 6a and 6b**, the sample counts for inhibiting molecules are less than 0.5% of total samples present in the training and test sets.

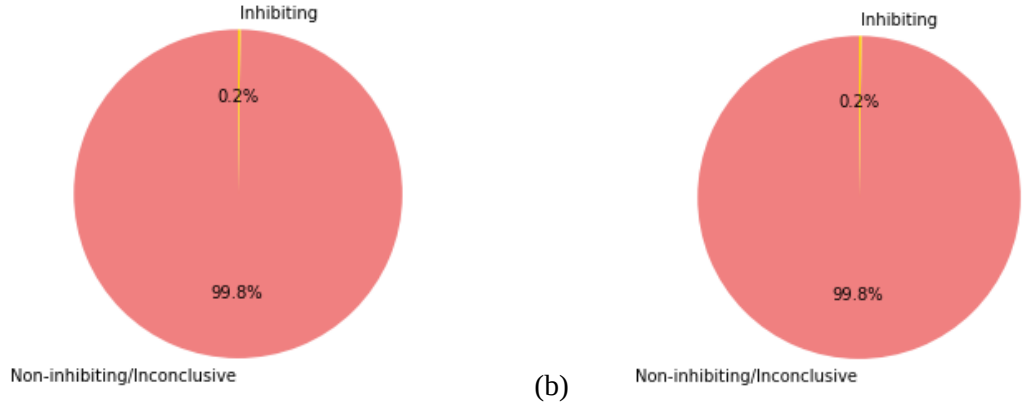


(a)



(b)

Figure 5: (a) Counts of samples of molecules in the complete training and test data. Yellow zone: Non-inhibiting molecules. Red zone: Inhibiting molecules. (b) Counts of samples of inhibiting molecules in the training and test data.



(a) (b)

Figure 6: Chart showing the percentage of samples that are confirmed inhibiting molecules and non-inhibiting/inconclusive molecules in: (a) The training set (b) The testing set.

This may adversely affect the extent to which the machine learning algorithms train on the inhibiting samples in favor of the non-inhibiting molecules which do not hold useful trends within their set because they all have zero inhibition. A popular approach to deal with this issue is to weight the samples towards the inhibiting molecules prior to implementing the designed workflow. Another approach used to explore and learn from the data is to only take the inhibiting molecules into consideration while building models and using those models to create a set of features using the genetic algorithm. In the project, the latter is used to avoid increased complexity of the

algorithm which may arise from the weighting mechanism. Though there is valuable information present in set of the non-inhibiting molecules, it is difficult to intelligently work with them given the ratio in which they are present in comparison to inhibiting molecules and not being able to apply appropriate knowledge-based filters to these molecules. As a result, both the entire dataset of 389,560 molecules and a reduced dataset of 19,927 molecules are passed through the workflow described in the sections to follow.

2.2 DESCRIPTOR GENERATION

To populate a working dataset with features representative of our molecule samples, molecular descriptors are calculated. According to Todeschini[17], a molecular descriptor is the result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment. Eleven classes of descriptors are computed using the SMILES strings and the Python-based package PyChem v1.0.[38,39] The instructions to set up this computing environment in Ubuntu exists in the GitHub repository[36]. The list of descriptors calculated are in **Appendix Table A.1**, totaling 566 descriptor values for each compound. **Figure 7** shows the transformation of the dimensions of the dataset during feature engineering.

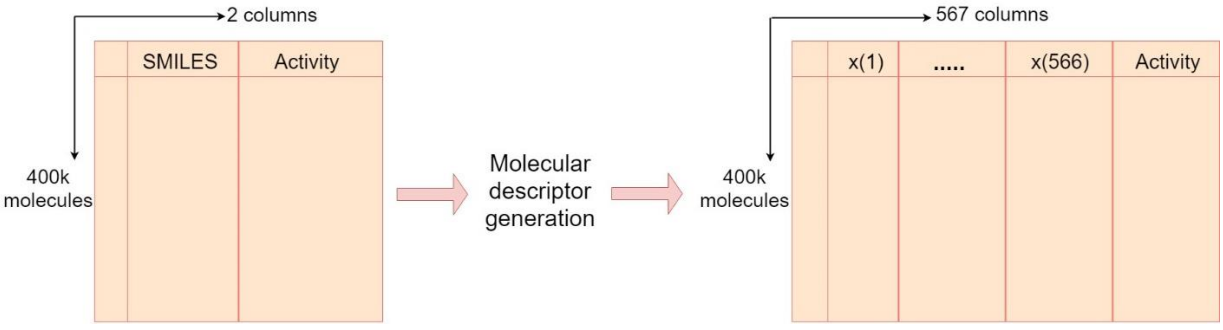


Figure 7: Dataset before and after the calculation of descriptors using PyChem

2.3 DATA PREPROCESSING AND FEATURE SELECTION

The best way to measure the ability of a predictive model to perform on future data is to try to simulate this eventuality. Since new data cannot be gathered easily, a portion of the available data is randomly selected and reserved to verify predictions. The portion that is not reserved and is used to develop models is called the training set, and the other portion used for predictive model assessment and model refinement is called the test set. The data is randomly split into arrays of train and test subsets in the ratio 3:1 using Scikit-learn. The assumptions made for this step are that:

- The data available for analytics fairly represents the physical process being modeled.
- The data generation and collection techniques used for the real-world process we wish to model are expected to remain relatively stable over time so that a well-constructed model using current laboratory standards is reasonably expected to perform adequately in the future.

The data frames are checked for rows containing missing, NaN and infinite values. These are replaced with zeros, since the data is centered around zero and it is the most abundant value in the dataset. The data is standardized so that the individual features have zero mean and unit variance. In practice, we often ignore the shape of the distribution and just transform the data to center it by removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator less responsive to other features. The training data is fit and standardized using a standard scaler from Scikit-learn's preprocessing module, and this fit is used to transform the test set as well. This order of events ensures that training set provides a baseline for comparison of trends in new and unseen data. The test dataset

may contain just one data point that we want to classify, which makes calculating the mean and standard deviation of the test data impossible. That is an intuitive case to show why we keep and use the training data parameters for scaling the test set. The target data is not scaled here since the unit of measurement of inhibition does not suffer from poor resolution.

Of the hundreds of descriptors that have been extracted, a few will have a substantial effect on the activity of the small molecule. The process of selection of relevant features from an available set through automated or domain knowledge based methods is called feature selection. It improves the speed, interpretability and accuracy of subsequent learning and prediction processes. The Python library Scikit-learn v0.18.1 is used to implement a random forest algorithm to perform feature space reduction. Since our target values are continuous between 0 and 100, only regression methods are used on this dataset. The algorithms are trained on the training set and the features that are preserved are used to reduce the test set as well.

A random forest is a meta estimator that fits several decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. It is commonly used because the tree-based strategies used by random forests naturally ranks features by how well they improve the purity of the node, leading to a decrease in impurity over all trees. Nodes with the greatest decrease in impurity happen at the start of the trees, while nodes with the least decrease in impurity occur at the end of trees. By pruning trees below a node, we can create a subset of the most important features. It is implemented using the random forest regression ensemble in Scikit-learn. To study the effects of the number of features used on the predictive performance of the machine learning models, different numbers of the most relevant features are selected using this algorithm: 25, 50, 75, 100, 150, 200, 250 and 300 features.

2.4 CHOOSING AND BUILDING MODELS

The algorithms we use in this project to model the data include linear regression, artificial neural networks, decision trees, ridge regression, Bayesian ridge regression, lasso and random forest regression. They are all created using Scikit-learn v0.18.1 except for neural nets which are implemented using nolearn v0.6.0.

Linear regression is used in this work to form a baseline model – if a straight-line fit is the simplest possible way to explain the quantitative structure-activity relationships. Though highly inaccurate, it serves as a performance plumb line, meaning that the prediction accuracies of all other models used must at least be as good as the accuracy provided by linear regression.

Neural nets are algorithms inspired by biological neurons for learning complex non-linear and multivariate relationships by functioning as a "black box" that is fed a set of input data and produces some desired output. A neural net approximates the complex functions governing QSARs, without any explicit knowledge of these mathematical expressions. They contain layers made up of non-linear summing elements called nodes that are interconnected between layers with weighted connection lines, where each layer receives its input from the previous layer and the output of each node feeds the nodes of the next layer. The weights are adjusted iteratively, forming highly accurate and computationally efficient networks. In this work, a single neural network was designed for each batch of features to predict all the features at once using two dense hidden layers instead of one based on findings by Hough[40]. In his work, a single hidden layer could not adequately approximate the relationships, and predictions resulted in unacceptably low R^2 values. The Python package nolearn v0.6.0 was used to design the neural network architecture and train it[41].

Ridge regression models have linear least squares loss functions and L2 regularization. Instead of just minimizing the residual sum of squares we also have a penalty term on the feature coefficients, equivalent to square of the magnitude of coefficients. Like ridge regression, lasso regression performs L1 regularization, adding a penalty equivalent to square of the magnitude of coefficients. In both cases, if the coefficients take on large values, the optimization function is penalized to drive parameter values closer to zero and reduce penalty. The slope of the prediction is much more stable and the variance in the line itself is greatly reduced, in comparison to that of the standard linear regression. They are used in this project due to numerous predictors, where fitting the full model without penalization will result in large prediction intervals. These methods place tight constraints on the parameters in the linear model. Ridge regression does not zero out coefficients and uses all the features in the model, while lasso does both parameter shrinkage and variable selection automatically. Compared to the ordinary least squares estimator, in Bayesian ridge regression, the coefficient weights are slightly shifted toward zeros, which stabilizes them. It enables the study of the posterior distributions of the coefficient estimates and ease of interpretation, and enhances flexibility in model design even though its solution takes more time to arrive at.

Random forest regression was previously used to select features and is used again to create a predictive model. Along with the decision trees algorithm which works using the same principle – except for the presence of just one decision-making tree instead of a forest of trees – a random forest regression model is built using the training data. They predict the value of a target variable by learning simple decision rules inferred from the data features, and deciding about the testing data at each node of the tree. They are simple to understand and interpret, governed by Boolean logic (making it a “white-box” model), and can be easily visualized. Decision trees are, however, prone to overlearning, which is solved using a forest of trees. Mechanisms such as setting the

minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem. Support vector regression (SVR) is a learning method that is effective in high dimensional spaces. The model produced by SVR depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. For our project, a linear kernel is used for the cost function.

Models have been tuned after being checked for suitability for this application. In machine learning, hyperparameter optimization is the process of choosing a set of hyperparameters for a learning algorithm to optimize the predictive accuracy on a test data set. It ensures that the model does not overfit its data by tuning, while still learning the inputs well. Scikit-learn's GridSearchCV is a popular method for hyperparameter optimization that tries the model using every combination of values in distributions of values given for each parameter, but we chose to use RandomizedSearchCV instead to reduce the computational expense that comes with a grid search. A fixed number of parameter settings are randomly sampled from the specified parameter distributions. Optimization was used to determine the maximum depth of the trees and the minimum samples required for a node split for the random forest. In this project, hyperparameter optimization is used to determine the hyperparameters for decision trees, neural networks, random forest regression and lasso. Optimization was used to sample over each parameter space 20 times and determine the maximum depth of the trees (over a range of 1 to 15 for the length from root to leaf) and the minimum samples required for a split (between 2 and 15 samples) for decision trees and random forests; the number of nodes in the two hidden layers (between 3 and 30 nodes per layer) for the neural network; and the L1 regularization constant alpha for lasso (between 0.0001 and 1). The chosen values are used to train and fit the model on the data.

The measures of model performance used include the mean absolute error, mean square error, median absolute error, the R^2 score and the explained variance score. The predicted values of inhibition scores for each compound in the test set are also obtained.

2.5 GENETIC ALGORITHMS FOR DRUG LEAD GENERATION

This section of the project is conceptually novel and forms a link between cheminformatics and the pharmaceutical process for drug design. In pharmaceutical development, the design of new molecules can be based on desired properties like better binding, potency or precision of action, achievable through evolutionary techniques. For this process, two things must be known to some extent: the properties that are desired and how they relate to a molecule's structure. The structure–activity relationship is needed to both determine the necessary properties and build a molecule that has those properties as shown in **Figure 8**, where the connection between molecular structure and properties is depicted.

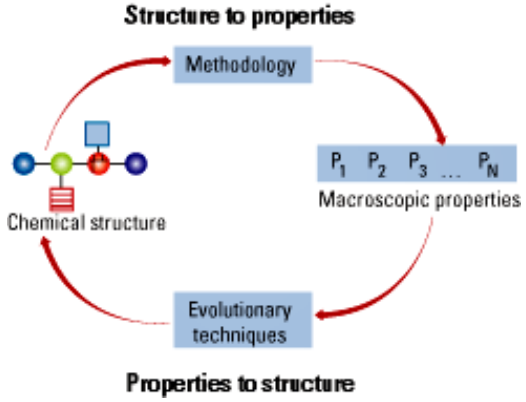


Figure 8: The forward and reverse flow of data in property prediction and compound design respectively.[42]

The first stage is determining the properties of a molecule based on its structure. This has been achieved by building machine learning models that describe QSAR relationships. The second stage is building a structure based on properties known to be important and determine the structure that will have them. This is not as straightforward as the first part though it depends on it.

Computers can be instructed to mutate and combine virtual molecules to produce new molecules from various combinations of existing molecules using evolutionary survival of the fittest, removing the less-fit candidates. This works by combining two functions to perform the evolution process: genetic algorithms and fitness functions.

A genetic algorithm (GA) produces mutations and crossover on the best data, like sexual reproduction, to produce new and random generations of data. The fitness function rounds out the evolutionary system by testing all the data and eliminating poor performers, just like the survival of the fittest. In humans, the genetic material in DNA is represented as base pairs, where strings of three base pairs makes a codon that specifies the amino acids within a protein. By analogy, in the design of drug compounds, symbols like N could represent a nitrogen atom, or COOH a carboxylic acid group in the molecular structure. Further, using larger units like numerical molecular descriptors instead of atoms, for use as 'genetic' data, reduces the complexity and speed of computation.

Parents	abcdefg	uvwxyz
Crossover point	abc*defg*	uvw*xyz*
Children	abc*xyz*	uvw*defg*

Figure 9: Genetic algorithms work by crossing over linear strings of data.

The GA causes either mutation, crossover, or both. Mutation is the random changing of individual molecular descriptors. For example, a carbon count of five could be changed to three, and the new molecule would then be measured by the fitness function. Crossover is analogous to sexual reproduction of a cruder format. The GA randomly cuts the string of two different individuals in two, as seen in **Figure 9**. The half strings switch to create two children, each with half from each parent.

The children and the parents are then evaluated by the fitness function: the function ranks the molecules; the poorest inhibiting compounds are removed; and the successful progeny and parents go on to be manipulated again by the GA. With the proper governing equations generated by machine learning models, data can be tested to determine the best-performing member of any set using the principle that the fittest remain and the weak become extinct. The fitness function in this project is the absolute error between the maximum inhibition score of 100 and the predicted inhibition score of the molecule whose properties have been generated by the genetic algorithm. The inhibition score is predicted using the trained machine learning models from the previous section of this project. 100 descriptors were calculated using the genetic algorithm using the previously trained random forest regression model fit on 100 descriptors.

The key to any genetics-based solution is a good fitness function which must be robust since the randomly generated initial molecules rarely make chemical sense. The entire process is usually limited by the speed of the fitness function which can be computationally expensive as they might rely on a complex trained QSAR model. A flowchart of the process of a genetic algorithm is shown in **Figure 10**, which is pseudocode for the Python script used in this work.

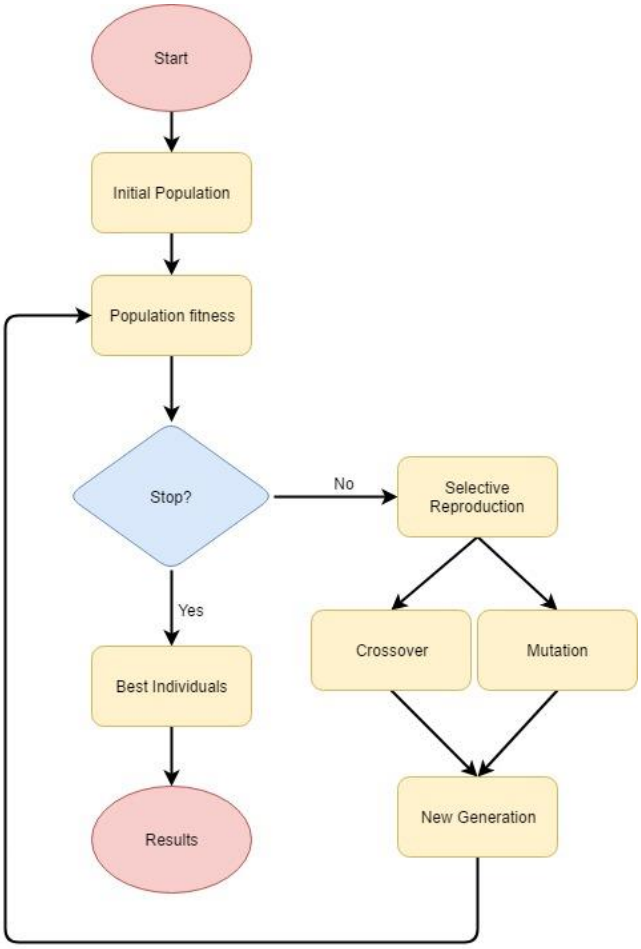


Figure 10: Flowchart of a genetic algorithm.

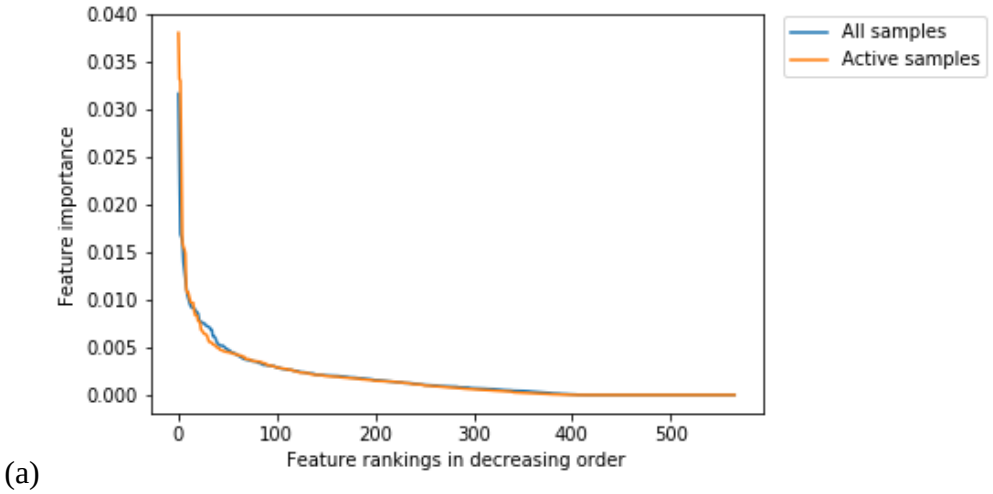
Other computational techniques have been used in the past to develop molecular structures. Strictly computational or rule-based suffer from combinatorial complexity of the search space, design knowledge acquisition difficulties, nonlinear structure–property correlations, and problems incorporating higher-level chemical and biological knowledge[42]. The advantage of genetic algorithms is that the computer is a tool not just for modeling and understanding these highly non-linear relationships, but also randomly developing new structures and determining whether those structures serve a specific purpose. When combined with the domain knowledge of the practicality of the generated structures, genetic algorithms can be a powerful tool for drug lead generation. The

code used to implement the genetic algorithm in this project is a modification of a previous general purpose Python GA script by Larson[43].

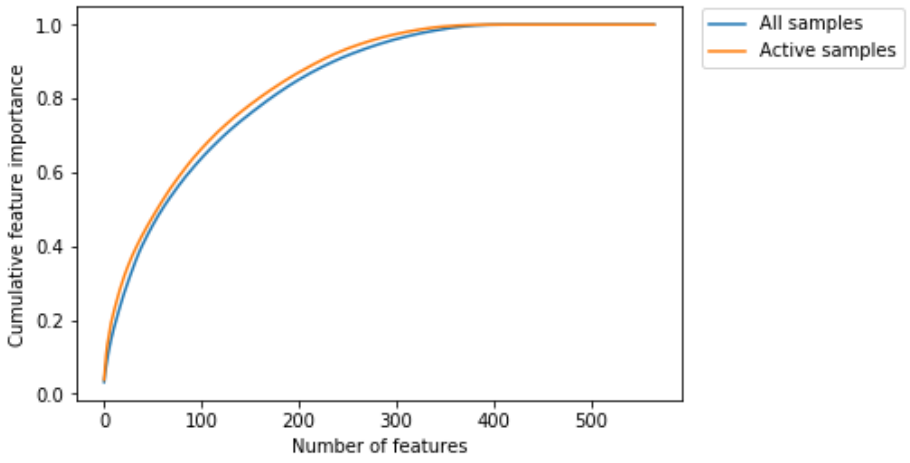
Chapter 3

RESULTS AND DISCUSSION

This section looks through the results of the methods used in the project as listed in the previous section, analyzes those results and their impact, seeks sources of errors and ways to address them for better work in the future.



(a)



(b)

Figure 11: (a) Random forest feature importance plot with features ranked in decreasing order of importance. (b) Random forest cumulative feature importance plot with features ranked in decreasing order of importance.

Random forest's impurity based ranking is typically aggressive in the sense that there is a sharp drop-off of scores after the best ranking features, as shown in **Figure 11(a)**. The feature

importance scores must add up to one when using the complete dataset as seen in **Figure 11(b)** where the sum becomes when all the features are added. Most of the variance in the model is thus explained by approximately the top 300 features. Using this estimate for the total number of relevant features, subsets of the training set and test dataset were created – the random forest regression algorithm ranked different numbers of the most relevant 25, 50, 75, 100, 150, 200, 250 and 300 features – to study the effect of the number of features on the predictive accuracies of the machine learning models. To counter the sudden drop in feature importance scores, stability selection methods like LassoCV can be used, which produce a smooth decrease in scores.

The values for the optimized hyperparameters and results from the feature selection process are available in the Jupyter notebooks in the GitHub repository. After the training procedure, each machine learning model is evaluated based on how well it predicts the activity of the test set compounds. To achieve this, each model is provided input data from the test set, which none of the models had previously been exposed to, and model predictions are compared with the expected test set outputs. The results are tabulated in **Table 1** and the measures of model performance used include the:

- Mean absolute error (MAE): The average of the absolute differences between the true and predicted values. The best possible value is zero.

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| .$$

- Mean square error (MSE): The average of the squares of the difference between the true and predicted values. The best possible value is zero.

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

- Median absolute deviation (MedAE): The median of the absolute values of the differences between the true and predicted values. The best possible value is zero.

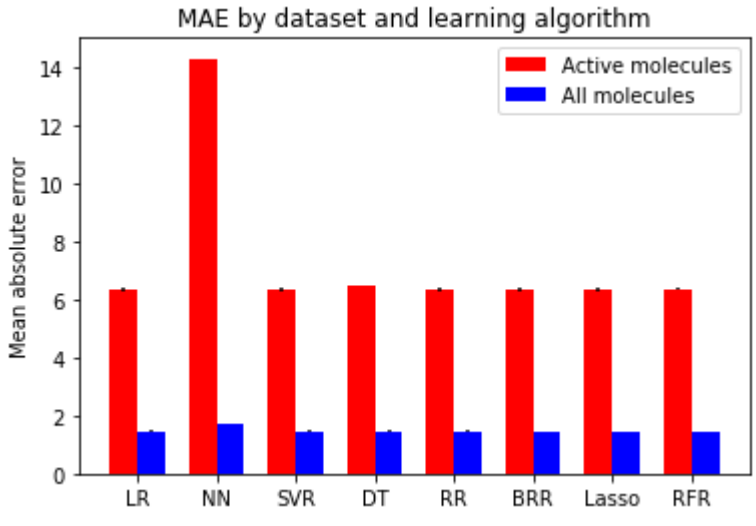
$$\text{MedAE}(y, \hat{y}) = \text{median}(\mid y_1 - \hat{y}_1 \mid, \ldots, \mid y_n - \hat{y}_n \mid).$$

- R^2 score: The coefficient of determination as a regression score function. The best possible score is one and it can be negative if the model is arbitrarily worse. A constant model that always predicts the expected value of y, independent of the input values, has a R^2 score of zero.

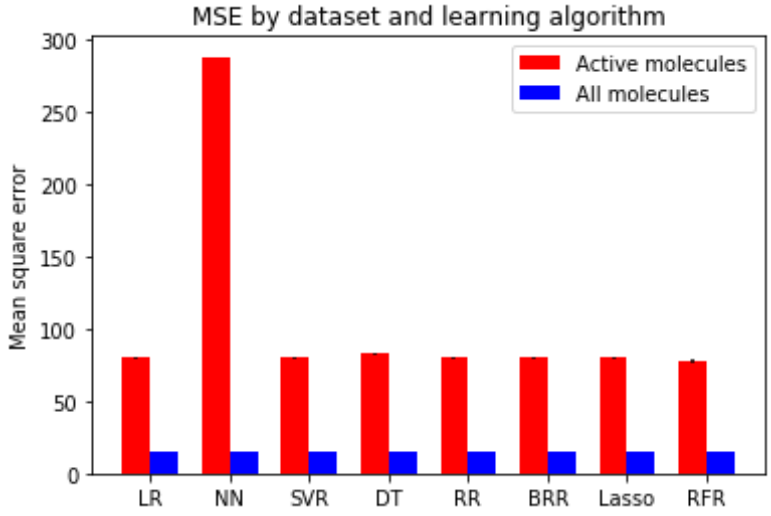
$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2}$$

- Explained variance score (EVS): A measure of the proportion to which a model explains the dispersion in a data set. The best possible score is one, with lower scores for poorer models.

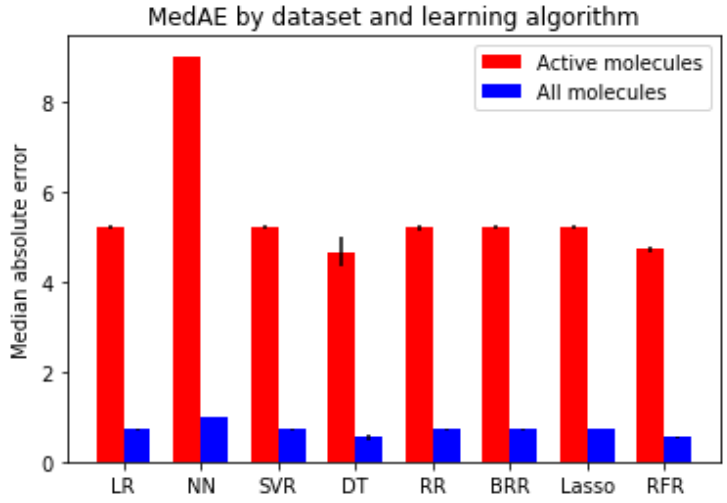
$$\texttt{explained\_variance}(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$



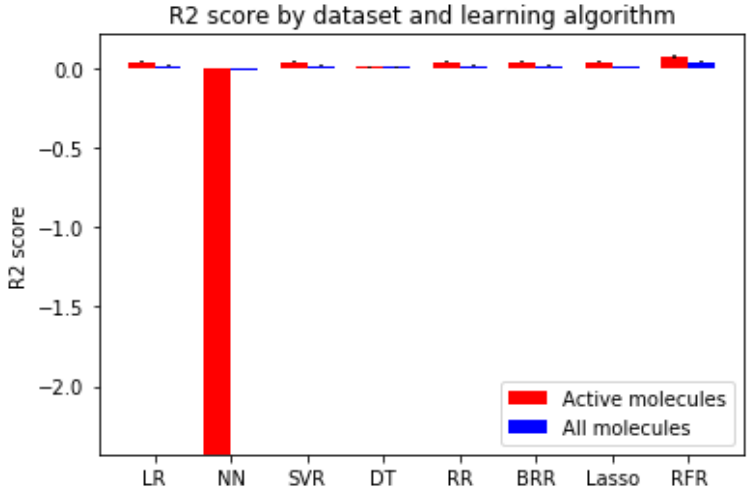
(a)


MSE by dataset and learning algorithm
Mean square error
Active molecules
All molecules
300
250
200
150
100
50
0
LR
NN
SVR
DT
RR
BRR
Lasso
RFR

(b)


MedAE by dataset and learning algorithm
Median absolute error
Active molecules
All molecules
8
6
4
2
0
LR
NN
SVR
DT
RR
BRR
Lasso
RFR

(c)


R2 score by dataset and learning algorithm
R2 score
Active molecules
All molecules
0.0
−0.5
−1.0
−1.5
−2.0
LR
NN
SVR
DT
RR
BRR
Lasso
RFR

(d)

(e)

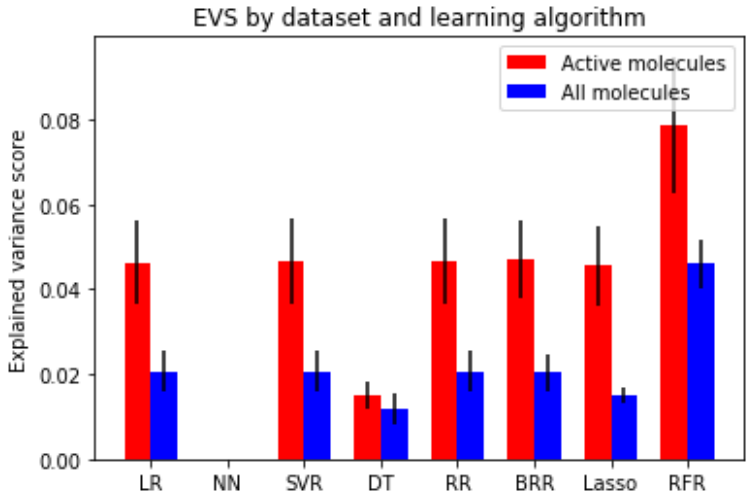


Figure 12: Model performance results across different learning algorithms and datasets. (a) Mean absolute error (b) Mean square error (c) Median absolute error (d) R2 score (e) Explained variance score.

In **Figure 12** and **Appendix Table A.1**, there isn't a high level of accuracy provided by the models. Though the MAE, MSE and MedAE values may be considered moderate accuracy, the R^2 and EVS scores provide vital information about the fit. For MSE alone is not a good measure of the fit of a model, since multiple fits can produce the same MSE as shown in **Figure 13**. The Anscombe's quartet comprises of four datasets that have nearly identical simple descriptive statistics, yet appear very different when graphed. They were constructed in 1973 by the statistician Francis Anscombe to demonstrate both the importance of graphing data before analyzing it and the effect of outliers on statistical properties. MSE will be a bad measure in any case except the first set. In the second set, the data is nonlinear; in the third, the fit does not capture outliers well; and in the fourth the fit is very poor.

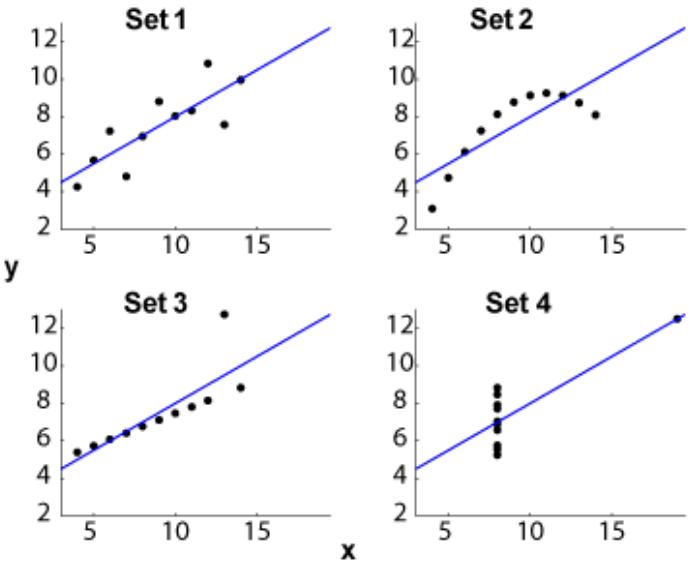


Figure 13: Anscombe's quartet.

The R^2 scores are very low, which indicate an ill fit. In the case of neural networks, the R^2 scores are negative which signifies an opposite slope of fit. The mean of the test set might be very different from the mean of the training set during cross-validation, inducing a much higher incurred squared error of prediction, resulting in a negative R^2 score. The low EVS scores show that only a small percentage of our data is explained by the models. Between the entire sample set and the active sample set, the EVS scores are double for the latter, supporting our hypothesis that useful trends can be learnt in the active set of molecules. The MAE differences between the complete set and the active set is interesting, since the MAE values for the latter is about 4 times larger than the former. The reason for this is that most the molecules in the complete set have activities of zero, which increases the probability of a zero-value prediction. Thus, the MAE and MSE values are small compared to the active set where there is a high variance in the activities of the training set and as result, the test set.

There are a few reasons for the generally low accuracies of prediction. The regression accuracies depend on the chemical relationships between the enzyme and the drug which are highly non-linear relationships, not adequately explained simply by the molecular descriptors of the drug.

QSAR techniques function well on a small scale, determining properties of specific regions but failing to produce an accurate global description of the molecule. If functional areas that might participate in binding are identified with QSAR properties, then those regions can be weighed heavily in determining the molecule's activity. This is a pharmacophore and can be used to characterize the whole molecule. Pharmacophore models attempt to combine some of the advantages of QSAR techniques with the idea of identifying substituents.

With regards to the topic of data collection, the measure used to rank the molecules as inhibitors or non-inhibitors is the AC_{50} – the concentration for half-maximal activity derived from the Hill equation model. It is the most common potency measure applied in pharmacological research and toxicity testing. The AC50 parameter estimates the concentration at which a chemical produces the half the maximum response along a sigmoidal curve. Incorporating domain knowledge into the curve fitting process can improve agreement between AC50 estimates for sigmoidal curves, however, it is not possible to know the underlying shape of the concentration-response relationship before conducting an experiment and real biological responses may be represented by complex response patterns. Applying individualized curve fitting procedures can be useful for characterizing screening results which, in the high-throughput setting, can be restrictively laborious and result in extensive data loss while still compromising on the repeatability of process. It is not unusual for AC50 estimates to be accompanied by large standard errors even when one or both asymptotes can be defined for the concentration-response relationships. This issue has been studied by Shockley[44] and suggestions have been made for usage of other measures of potency like the Point of Departure of the weighted entropy score which describes the average information content in a probability distribution, and can be used to describe the extent and uniformity of response in a concentration-response profile.

Another reason for low accuracies is that the data is skewed to favor training on non-inhibiting molecules and can't learn well enough from inhibiting molecules. Though in this project, the solution of analyzing the active and inactive molecules separately was implemented, it would be worthwhile exploring other techniques to deal with imbalanced sets, that could provide better representation in the models. The method of adding sample weights can be tried, though it would be more useful to increase the number of active samples through intentional data collection and then train the models better.

In addition, the uncalculated features – numbering to over 700 descriptors – could explain the QSAR much better, and they can be computed to take into consideration the valuable information in them. During feature selection, independent variables that are useless by themselves might be useful together. Understanding combinatory feature relevance could bring into the picture the intricate interdependencies of the molecular descriptors. Applying a priori knowledge-based filters to the feature set and samples collected is an intensive manual process that goes against the purpose of machine learning but could increase the likelihood of accurate predictions. The feature selection process is a crucial part of preprocessing and greatly affects the training success of the models. By fixing the issues we faced in calculating and filtering these descriptors, the model accuracies and completeness can be increased.

Another cause for poorly trained models is that the random search over the hyperparameter space could result in local minima. An exhaustive grid search can search for the global solution and overcome the use of bad hyperparameters that influence learning tendencies. This is known to be a source of error in our project because depths of decision trees have sometimes been found to be one, which is impractical for effective training. This can be conducted through increase computational resources and smarter pipelines suited to multiprocessing.

The genetic algorithm was used to virtually synthesize a drug lead with an optimal activity score and the accuracy of its property values is a strong function of the predictive accuracy of the trained machine learning model powering the algorithm. Since our model performance can be greatly improved, the results from the genetic algorithm can be honed until we obtain realistic molecular properties that can be studied for viability from a chemistry perspective. A noteworthy point here is that this is a working open-source Python implementation of genetic algorithms for molecular drug design, and provides an intuitive framework for further computational research using this technique. The code produced in this project is the first publicly available implementation of the PyChem package, and provides detailed instructions on how to setup the environment for the descriptor calculations.

Chapter 4
CONCLUSIONS

This chapter summarizes the research described in this thesis and its significance, and provides recommendations for future work.

4.1 SUMMARY AND RESEARCH SIGNIFICANCE

The code used to perform machine learning and inhibitor molecule design is meant to be general, and applicable to a new set of molecules for a different property of interest. An open-source software was created to take in molecular data, extract its features, filter the best features, create models, optimize them, and use them to synthesize lead molecules that exhibit desirable properties. Despite the low model accuracies and poor fits, the sources of error have been enumerated and suggestions have been made– proprietary data collection techniques tailored to the needs of research, improved techniques to measure enzyme activity, ensuring a balanced dataset with adequate representation of diverse samples, a complete feature generation process that takes into consideration the mechanics of the inhibition process, improved feature selection through the inclusion of domain knowledge-based reasoning, an exhaustive hyperparameter optimization process that searches for the global minimum avoiding local minima.

Although the results of this project show many shortcomings of the present techniques, advances in 3-D QSAR are leading to better characterization of molecules and calculation of their properties. Increases in computer power allow more data points to be generated for surface properties, making the current field of functions closer estimations of macro-properties than ever before. With the right combination of good data, techniques and computational power, the potential for such work is immense in the development of novel and effective drugs against various target diseases. By making our methodology and practices freely available to the public, we seek

to further openness and transparency in chemical engineering research - an accelerator for scientific progress. This project has provided a promising pipeline that can serve as a guideline for future work in the field of cheminformatics.

4.2 FUTURE WORK

The following steps can be pursued to improve the work completed in this project:

i. The software was created to be used more widely, and applicable to a new set of molecules for a different property of interest. We can further the code to accept inputs of public and proprietary datasets and generalize the software for extended use.

ii. Design a graphical user interface (GUI) using a web-based Flask application, with a backend Python framework.

iii. The uncalculated features – numbering to over 700 – could explain the QSAR much better, and they can be computed to take into consideration the valuable information that might be in them. By fixing the issues we faced in calculating these descriptors, the model accuracies and completeness can be increased.

iv. Depending on the nature of the dataset and interpretability of feature spaces, an intuition and knowledge-based preliminary feature reduction mechanism will enhance the quality of model inputs.

v. Optimize the model creation process using powerful machines that can search over a finer hyperparameter space to reach the global minimum and avoid local minima.

vi. The molecular properties obtained from the genetic algorithm can be tested for practicality, and if it is not practical, some knowledge-based constraints can be placed on the genetic algorithm and fitness function.

vii. Add unit tests to the package to write faster and more robust code of higher quality; improve code definition, design and re-use (both code and tests can be migrated to a new project). This also helps write larger chunks of code through test-driven development, providing on-the-go sanity checks.

BIBLIOGRAPHY

1. Libbrecht MW, Noble WS. Machine learning applications in genetics and genomics. *Nat Rev Genet*. 2015;16(6):321-332. http://dx.doi.org/10.1038/nrg3920.

2. Bhat N, McAvoy TJ. Use of neural nets for dynamic modeling and control of chemical process systems. *Comput Chem Eng*. 1990;14(4):573-582. doi:http://dx.doi.org/10.1016/0098-1354(90)87028-N.

3. Yang H, Park M, Cho M, Song M, Kim S. A system architecture for manufacturing process analysis based on big data and process mining techniques. In: *Big Data (Big Data), 2014 IEEE International Conference on*. ; 2014:1024-1029. doi:10.1109/BigData.2014.7004336.

4. Nigsch F, Bender A, Van Buuren B, Tissen J, Nigsch E, Mitchell JBO. Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization. *J Chem Inf Model*. 2006;46(6):2412-2422. doi:10.1021/ci060149f.

5. Tan NX, Rao HB, Li ZR, Li XY. Prediction of chemical carcinogenicity by machine learning approaches. *SAR QSAR Environ Res*. 2009;20(1-2):27-75. doi:10.1080/10629360902724085.

6. Pella A, Cambria R, Riboldi M, et al. Use of machine learning methods for prediction of acute toxicity in organs at risk following prostate radiotherapy. *Med Phys*. 2011;38(6):2859-2867. doi:10.1118/1.3582947.

7. Cruz JA, Wishart DS. Applications of Machine Learning in Cancer Prediction and Prognosis. *Cancer Inform*. 2006;2:59-77. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2675494/.

8. Lima AN, Philot EA, Goulart Trossini GH, Barbour Scott LP, Maltarollo VG, Honorio KM. Use of machine learning approaches for novel drug discovery. *Expert Opin Drug Discov*. 2016;11(3):225-239. doi:10.1517/17460441.2016.1146250.

9. Gamberger D, Sekušak S, Medven Ž, Sabljić A. Application of Artificial Intelligence in Biodegradation Modelling. In: Peijnenburg WJGM, Damborský J, eds. *Biodegradability Prediction*. Dordrecht: Springer Netherlands; 1996:41-50. doi:10.1007/978-94-011-5686-8_5.

10. Beck DAC, Carothers JM, Subramanian VR, Pfaendtner J. Data science: Accelerating innovation and discovery in chemical engineering. *AIChE J*. 2016;62(5):1402-1416. doi:10.1002/aic.15192.

11. Nantasenamat C, Isarankura-Na-Ayudhya C, Prachayasittikul V. Advances in computational methods to predict the biological activity of compounds. *Expert Opin Drug Discov*. 2010;5(7):633-654. doi:10.1517/17460441.2010.492827.

12. Ozboyaci M, Kokh DB, Corni S, Wade RC. Modeling and simulation of protein-surface interactions: achievements and challenges. *Q Rev Biophys*. 2016;49:e4. doi:10.1017/S0033583515000256.

13. Zhou X, Li Z, Dai Z, Zou X. QSAR modeling of peptide biological activity by coupling support vector machine with particle swarm optimization algorithm and genetic algorithm. *J Mol Graph Model*. 2010;29(2):188-196. doi:10.1016/j.jmgm.2010.06.002.

14. Jamal S, Scaria V. Cheminformatic models based on machine learning for pyruvate kinase inhibitors of Leishmania mexicana. *BMC Bioinformatics*. 2013;14:329. doi:http://dx.doi.org/10.1186/1471-2105-14-329.

15. Palomba D, Martïnez MJ, Ponzoni I, Dïaz MF, Vazquez GE, Soto AJ. QSPR models for predicting log pliver values for volatile organic compounds combining statistical methods and domain knowledge. *Molecules*. 2012;17(12):14937-14953. doi:10.3390/molecules171214937.

16. Suenderhauf C, Hammann F, Huwyler J. Computational prediction of blood-brain barrier permeability using decision tree induction. *Molecules*. 2012;17(9):10429-10445. doi:10.3390/molecules170910429.

17. Todeschini R, Consonni V. *Molecular Descriptors for Chemoinformatics*.; 2010. doi:10.1002/9783527628766.

18. van de Waterbeemd H, Gifford E. ADMET in silico modelling: towards prediction paradise? *Nat Rev Drug Discov*. 2003;2(3):192-204. doi:10.1038/nrd1032.

19. Schwaighofer A, Schroeter T, Blanchard SM and G. How Wrong Can We Get? A Review of Machine Learning Approaches and Error Bars. *Comb Chem High Throughput Screen*. 2009;12(5):453-468. doi:http://dx.doi.org/10.2174/138620709788489064.

20. Kraljevic S, Stambrook PJ, Pavelic K. Accelerating drug discovery. *EMBO Rep*. 2004;5(9):837-842. doi:10.1038/sj.embor.7400236.

21. Adams C, Brantner V V. Estimating the cost of new drug development: is it really $802m? *Health Aff*. 2006;25(2):420-428. doi:10.1377/hlthaff.25.2.420.

22. Kola I, Landis J. Can the pharmaceutical industry reduce attrition rates? *Nat Rev Drug Discov*. 2004;3(August):1-5. doi:10.1038/nrd1470.

23. Group BC. A Revolution in R&D: How Genomics and Genetics Will Affect Drug Development Costs and Times. In: *PAREXEL's Pharmaceutical R&D Statistical Sourcebook*. ; 2003.

24. NIH Chemical Genomics Center. Inhibitors of USP1/UAF1: Primary Screen. PubChem. https://pubchem.ncbi.nlm.nih.gov/bioassay/743255. Published 2014.

25. National Center for Biotechnology Information. USP1 - ubiquitin specific peptidase 1 (human). PubChem. https://pubchem.ncbi.nlm.nih.gov/target/gene/USP1#section=Top. Published 2016.

26. Kirkin V, Dikic I. Role of ubiquitin- and Ubl-binding proteins in cell signaling. *Curr Opin Cell Biol*. 2007;19(2):199-205. doi:10.1016/j.ceb.2007.02.002.

27. Priolo C, Tang D, Brahamandan M, et al. The isopeptidase USP2a protects human prostate cancer from apoptosis. *Cancer Res*. 2006;66(17):8625-8632. doi:10.1158/0008-5472.CAN-06-1374.

28. Popov N, Wanzel M, Madiredjo M, et al. The ubiquitin-specific protease USP28 is required for MYC stability. *Nat Cell Biol*. 2007;9(7):765-774. http://dx.doi.org/10.1038/ncb1601.

29. De Pittà C, Tombolan L, Dell'Orto MC, et al. A leukemia-enriched cDNA microarray platform identifies new transcripts with relevance to the biology of pediatric acute lymphoblastic leukemia. *Haematologica*. 2005;90(7):890-898. doi:10.3324/haematol.11263.

30. Kovalenko A, Chable-Bessia C, Cantarella G, Israël A, Wallach D, Courtois G. The tumour suppressor CYLD negatively regulates NF-kappaB signalling by deubiquitination. *Nature*. 2003;424(6950):801-805. doi:10.1038/nature01802.

31. Olazabal-Herrero A, García-Santisteban I, Rodríguez JA. Structure-function analysis of USP1: insights into the role of Ser313 phosphorylation site and the effect of cancer-associated mutations on autocleavage. *Mol Cancer*. 2015;14:33. doi:10.1186/s12943-015-0311-7.

32. Wahi D, Jamal S, Goyal S, et al. Cheminformatics models based on machine learning approaches for design of USP1/UAF1 abrogators as anticancer agents. *Syst Synth Biol*. 2015;9(1-2):33-43. doi:10.1007/s11693-015-9162-1.

33. SMILES - A Simplified Chemical Language.

34. Heller SR, McNaught A, Pletnev I, Stein S, Tchekhovskoi D. InChI, the IUPAC International Chemical Identifier. *J Cheminform*. 2015;7:23. doi:10.1186/s13321-015-0068-4.

35. Fdardel. SMILES string representation of a chemical compound ciprofloxacin. https://commons.wikimedia.org/wiki/File:SMILES.png. Published 2007.

36. Philip P, Avadhoot R, A.C. Beck D. USP-inhibition|05.30.17. 2017. doi:10.5281/zenodo.800713.

37. Philip P. Data for USP-Inhibition project. https://s3-us-west-2.amazonaws.com/pphilip-usp-inhibition/data. Published 2016.

38. PyChem. https://code.google.com/archive/p/pychem/downloads. Published 2013.

39. Cao D-S, Xu Q-S, Hu Q-N, Liang Y-Z. ChemoPy: freely available python package for computational biology and chemoinformatics. *Bioinformatics*. 2013;29(8):1092-1094. http://dx.doi.org/10.1093/bioinformatics/btt105.

40. Hough BR. Computational approaches and tools for modeling biomass pyrolysis. 2016.

41. Nouri D. dnouri/nolearn: scikit-learn compatible neural network library. https://github.com/dnouri/nolearn. Published 2016.

42. Venkatasubramanian V, Chan K, Caruthers JM. Genetic Algorithmic Approach for Computer-Aided Molecular Design. In: *Computer-Aided Molecular Design*. Vol 589. ACS Symposium Series. American Chemical Society; 1995:28-396. doi:doi:10.1021/bk-1995-0589.ch028.

43. Larson W. Genetic Algorithms: Cool Name & Damn Simple. 2009. https://lethain.com/genetic-algorithms-cool-name-damn-simple/.

44. Shockley KR. Estimating Potency in High-Throughput Screening Experiments by Maximizing the Rate of Change in Weighted Shannon Entropy. *Sci Rep*. 2016;6:27897. doi:10.1038/srep27897.

APPENDIX

Table A.1: List of descriptors that can be calculated using PyChem. Those that have been calculated and those that have not been calculated (to date) are highlighted blue and red, respectively.

Constitutional descriptors (30)		
1	Weight	Molecular weight
2	nhyd	Count of hydrogen atoms
3	nhal	Count of halogen atoms
4	nhet	Count of hetero atoms
5	nhev	Count of heavy atoms
6	ncof	Count of F atoms
7	ncocl	Count of Cl atoms
8	ncobr	Count of Br atoms
9	ncoi	Count of I atoms
10	ncarb	Count of C atoms
11	nphos	Count of P atoms
12	nsulp	Count of S atoms
13	noxy	Count of O atoms
14	nnitro	Count of N atoms
15	nring	Number of rings
16	nrot	Number of rotatable bonds
17	ndonr	Number of H-bond donors
18	naccr	Number of H-bond acceptors
19	nsb	Number of single bonds
20	ndb	Number of double bonds
21	ntb	Number of triple bonds
22	naro	Number of aromatic bonds
23	nta	Number of all atoms
24	AWeight	Average molecular weight
25-30	PC1 - PC6	Molecular path counts of length 1-6

Topological descriptors (35)		
1	W	Weiner index
2	AW	Average Weiner index
3	J	Balaban's J index
4	Thara	Harary number
5	Tsch	Schiultz number
6	Tigdi	Graph distance index
7	Platt	Platt number
8	Xu	Xu index

9	Pol	Polarity number
10	Dz	Pogliani index
11	Ipc	Ipc index
12	BertzCT	BertzCT
13	GMTI	Gutman molecular topological index based on simple vertex degree
14-15	ZM1-ZM2	Zagreb index with order 1-2
16-17	MZM1-MZM2	Modified Zagreb index with order 1-2
18	Qindex	Quadratic index
19	diametert	Largest value in the distance matrix
20	radiust	radius based on topology
21	petitjeant	Petitjean based on topology
22	Sito	the logarithm of the simple topological index by Narumi
23	Hato	harmonic topological index proposed by Narnumi
24	Geto	Geometric topological index by Narumi
25	Arto	Arithmetic topological index by Narumi
26	ISIZ	Total information index on molecular size
27	TIAC	Total information index on atomic composition
28	DET	Total information index on distance equality
29	IDE	Mean information index on distance equality
30	IVDE	Total information index on vertex equality
31	Sitov	Logarithm of the simple topological index by Narumi
32	Hatov	Harmonic topological index proposed by Narnumi
33	Getov	Geometric topological index by Narumi
34	Gravto	Gravitational topological index based on topological distance
35	GMTIV	Gutman molecular topological index based on valence vertex degree(log10)

Connectivity descriptors (44)		
1_11	0χpv - 10χpv	Valence molecular connectivity Chi index for path order 0-10
12	3χvc	Valence molecular connectivity Chi index for three cluster
13	4χvc	Valence molecular connectivity Chi index for four cluster
14	4χvpc	Valence molecular connectivity Chi index for path/cluster
15-18	3χvCH - 6χvCH	Valence molecular connectivity Chi index for cycles of 3-6

19-29	0χp - 10χp	Simple molecular connectivity Chi indices for path order 0-10
30	3χc	Simple molecular connectivity Chi indices for three cluster
31	4χc	Simple molecular connectivity Chi indices for four cluster
32	4χpc	Simple molecular connectivity Chi indices for path/cluster
33-36	3χCH - 6χCH	Simple molecular connectivity Chi indices for cycles of 3-6
37	mChi1	mean chi1 (Randic) connectivity index
38	knotp	the difference between chi3c and chi4pc
39	dchi0	the difference between chi0v and chi0
40	dchi1	the difference between chi1v and chi1
41	dchi2	the difference between chi2v and chi2
42	dchi3	the difference between chi3v and chi3
43	dchi4	the difference between chi4v and chi4
44	knotpv	the difference between chiv3c and chiv4pc

Kappa descriptors (7)		
1	1κα	Kappa alpha index for 1 bonded fragment
2	2κα	Kappa alpha index for 2 bonded fragment
3	3κα	Kappa alpha index for 3 bonded fragment
4	phi	Kier molecular flexibility index
5	1κ	Molecular shape Kappa index for 1 bonded fragment
6	2κ	Molecular shape Kappa index for 2 bonded fragment
7	3κ	Molecular shape Kappa index for 3 bonded fragment

Basak descriptors (21)		
1	IC0	Information content with order 0 proposed by Basak
2	IC1	Information content with order 1 proposed by Basak
3	IC2	Information content with order 2 proposed by Basak
4	IC3	Information content with order 3 proposed by Basak
5	IC4	Information content with order 4 proposed by Basak
6	IC5	Information content with order 5 proposed by Basak
7	IC6	Information content with order 6 proposed by Basak
8	SIC0	Structural information content with order 0 proposed by Basak
9	SIC1	Structural information content with order 1 proposed by Basak

10	SIC2	Structural information content with order 2 proposed by Basak
11	SIC3	Structural information content with order 3 proposed by Basak
12	SIC4	Structural information content with order 4 proposed by Basak
13	SIC5	Structural information content with order 5 proposed by Basak
14	SIC6	Structural information content with order 6 proposed by Basak
15	CIC0	Complementary information content with order 0 proposed by Basak
16	CIC1	Complementary information content with order 1 proposed by Basak
17	CIC2	Complementary information content with order 2 proposed by Basak
18	CIC3	Complementary information content with order 3 proposed by Basak
19	CIC4	Complementary information content with order 4 proposed by Basak
20	CIC5	Complementary information content with order 5 proposed by Basak
21	CIC6	Complementary information content with order 6 proposed by Basak

E-state descriptors (245)		
1	S1	Sum of E-State of atom type: sLi
2	S2	Sum of E-State of atom type: ssBe
3	S3	Sum of E-State of atom type: ssssBe
4	S4	Sum of E-State of atom type: ssBH
5	S5	Sum of E-State of atom type: sssB
6	S6	Sum of E-State of atom type: ssssB
7	S7	Sum of E-State of atom type: sCH3
8	S8	Sum of E-State of atom type: dCH2
9	S9	Sum of E-State of atom type: ssCH2
10	S10	Sum of E-State of atom type: tCH
11	S11	Sum of E-State of atom type: dsCH
12	S12	Sum of E-State of atom type: aaCH
13	S13	Sum of E-State of atom type: sssCH
14	S14	Sum of E-State of atom type: ddC
15	S15	Sum of E-State of atom type: tsC

16	S16	Sum of E-State of atom type: dsC
17	S17	Sum of E-State of atom type: aasC
18	S18	Sum of E-State of atom type: aaaC
19	S19	Sum of E-State of atom type: ssssC
20	S20	Sum of E-State of atom type: sNH3
21	S21	Sum of E-State of atom type: sNH2
22	S22	Sum of E-State of atom type: ssNH2
23	S23	Sum of E-State of atom type: dNH
24	S24	Sum of E-State of atom type: ssNH
25	S25	Sum of E-State of atom type: aaNH
26	S26	Sum of E-State of atom type: tN
27	S27	Sum of E-State of atom type: sssNH
28	S28	Sum of E-State of atom type: dsN
29	S29	Sum of E-State of atom type: aaN
30	S30	Sum of E-State of atom type: sssN
31	S31	Sum of E-State of atom type: ddsN
32	S32	Sum of E-State of atom type: aasN
33	S33	Sum of E-State of atom type: ssssN
34	S34	Sum of E-State of atom type: sOH
35	S35	Sum of E-State of atom type: dO
36	S36	Sum of E-State of atom type: ssO
37	S37	Sum of E-State of atom type: aaO
38	S38	Sum of E-State of atom type: sF
39	S39	Sum of E-State of atom type: sSiH3
40	S40	Sum of E-State of atom type: ssSiH2
41	S41	Sum of E-State of atom type: sssSiH
42	S42	Sum of E-State of atom type: ssssSi
43	S43	Sum of E-State of atom type: sPH2
44	S44	Sum of E-State of atom type: ssPH
45	S45	Sum of E-State of atom type: sssP
46	S46	Sum of E-State of atom type: dsssP
47	S47	Sum of E-State of atom type: sssssP
48	S48	Sum of E-State of atom type: sSH
49	S49	Sum of E-State of atom type: dS
50	S50	Sum of E-State of atom type: ssS
51	S51	Sum of E-State of atom type: aaS
52	S52	Sum of E-State of atom type: dssS
53	S53	Sum of E-State of atom type: ddssS
54	S54	Sum of E-State of atom type: sCl
55	S55	Sum of E-State of atom type: sGeH3

56	S56	Sum of E-State of atom type: ssGeH2
57	S57	Sum of E-State of atom type: sssGeH
58	S58	Sum of E-State of atom type: ssssGe
59	S59	Sum of E-State of atom type: sAsH2
60	S60	Sum of E-State of atom type: ssAsH
61	S61	Sum of E-State of atom type: sssAs
62	S62	Sum of E-State of atom type: sssdAs
63	S63	Sum of E-State of atom type: sssssAs
64	S64	Sum of E-State of atom type: sSeH
65	S65	Sum of E-State of atom type: dSe
66	S66	Sum of E-State of atom type: ssSe
67	S67	Sum of E-State of atom type: aaSe
68	S68	Sum of E-State of atom type: dssSe
69	S69	Sum of E-State of atom type: ddssSe
70	S70	Sum of E-State of atom type: sBr
71	S71	Sum of E-State of atom type: sSnH3
72	S72	Sum of E-State of atom type: ssSnH2
73	S73	Sum of E-State of atom type: sssSnH
74	S74	Sum of E-State of atom type: ssssSn
75	S75	Sum of E-State of atom type: sI
76	S76	Sum of E-State of atom type: sPbH3
77	S77	Sum of E-State of atom type: ssPbH2
78	S78	Sum of E-State of atom type: sssPbH
79	S79	Sum of E-State of atom type: ssssPb
80-158	Smax1-Smax79	Maximum of E-State value of specified atom type
159-237	Smin1-Smin79	Minimum of E-State value of specified atom type
238	Shev	The sum of the EState indices over all non-hydrogen atoms
239	Scar	The sum of the EState indices over all C atoms
240	Shal	The sum of the EState indices over all Halogen atoms
241	Shet	The sum of the EState indices over all hetero atoms
242	Save	The sum of the EState indices over all non-hydrogen atoms divided by the number of non-hydrogen atoms
243	Smax	The maximal Estate value in all atoms
244	Smin	The minimal Estate value in all atoms
245	DS	The difference between Smax and Smin

Burden descriptors (64)		
1_16	bcutm1-16	Burden descriptors based on atomic mass

17-32	bcutv1-16	Burden descriptors based on atomic volumes
33-48	bcute1-16	Burden descriptors based on atomic electronegativity
49-64	bcutp1-16	Burden descriptors based on polarizability

Autocorrelation descriptors (96)		
1_8	ATSm1-ATSm8	Moreau-Broto autocorrelation descriptors based on atom mass
9_16	ATSv1-ATSv8	Moreau-Broto autocorrelation descriptors based on atomic van der Waals volume
17-24	ATSe1-ATSe8	Moreau-Broto autocorrelation descriptors based on atomic Sanderson electronegativity
25-32	ATSp1-ATSp8	Moreau-Broto autocorrelation descriptors based on atomic polarizability
33-40	MATSm1-MATSm8	Moran autocorrelation descriptors based on atom mass
41-48	MATSv1-MATSv8	Moran autocorrelation descriptors based on atomic van der Waals volume
49-56	MATSe1-MATSe8	Moran autocorrelation descriptors based on atomic Sanderson electronegativity
57-64	MATSp1-MATSp8	Moran autocorrelation descriptors based on atomic polarizability
54-72	GATSm1-GATSm8	Geary autocorrelation descriptors based on atom mass
73-80	GATSv1-GATSv8	Geary autocorrelation descriptors based on atomic van der Waals volume
81-88	GATSe1-GATSe8	Geary autocorrelation descriptors based on atomic Sanderson electronegativity
89-96	GATSp1-GATSp8	Geary autocorrelation descriptors based on atomic polarizability

Charge descriptors (25)		
1_4	QHmax, QCmax, QNmax, QOmax	Most positive charge on H, C, N, O atoms
5_8	QHmin, QCmin, QNmin, QOmin	Most negative charge on H, C, N, O atoms
9_10	Qmax, Qmin	Most positive and negative charge in a molecule
11_15	QHSS, QCSS, QNSS, QOSS, QaSS	Sum of squares of charges on H, C, N, O and all atoms
16-17	Mpc, Tpc	Mean and total of positive charges
18-19	Mnc, Tnc	Mean and total of negative charges
20-21	Mac, Tac	Mean and total of absolute charges
22	Rpc	Relative positive charge

23	Rnc	Relative negative charge
24	SPP	Submolecular polarity parameter
25	LDI	Local dipole index

Molecular property descriptors (6)		
1	MREF	Molar refractivity
2	logP	LogP value based on the Crippen method
3	logP2	Square of LogP value based on the Crippen method
4	TPSA	Topological polarity surface area
5	UI	Unsaturation index
6	Hy	Hydrophilic index

MOE-type descriptors (60)		
1	TPSA	Topological polar surface area based on fragments
2	LabuteASA	Labute's Approximate Surface Area
3_14	SLOGPVSA	MOE-type descriptors using SLogP contributions and surface area contributions
15-24	SMRVSA	MOE-type descriptors using MR contributions and surface area contributions
25-38	PEOEVSA	MOE-type descriptors using partial charges and surface area contributions
39-49	EstateVSA	MOE-type descriptors using Estate indices and surface area contributions
50-60	VSAEstate	MOE-type descriptors using surface area contributions and Estate indices

Geometric descriptors (12)		
1	W3DH	3-D Wiener index based geometrical distance matrix (including Hs)
2	W3D	3-D Wiener index based geometrical distance matrix (Not including Hs)
3	Petitj3D	Petitjean Index based on molecular geometrical distance matrix
4	GeDi	The longest distance between two atoms (geometrical diameter)
5	grav1	Gravitational 3D index
6	rygr	Radius of gyration
7	Harary3D	The 3D-Harary index
8	AGDD	The average geometric distance degree
9	SEig	The absolute eigenvalue sum on geometry matrix
10	SPAN	The span R

11	ASPAN	The average span R
12	MEcc	The molecular eccentricity

CPSA descriptors (30)		
1	ASA	Solvent-accessible surface areas
2	MSA	Molecular surface areas
3	PNSA1	Partial negative area
4	PPSA1	Partial positive area
5	PNSA2	Total charge weighted negative surface area
6	PPSA2	Total charge weighted positive surface area
7	PNSA3	Total charge weighted negative surface area
8	PPSA3	Total charge weighted positive surface area
9	DPSA1	Difference in charged partial surface area
10	DPSA2	Difference in total charge weighted partial surface area
11	DPSA3	Difference in atomic charge weighted surface area
12	FNSA1	Fractional charged partial negative surface areas
13	FNSA2	Fractional charged partial negative surface areas
14	FNSA3	Fractional charged partial negative surface areas
15	FPSA1	Fractional charged partial positive surface areas
16	FPSA2	Fractional charged partial positive surface areas
17	FPSA3	Fractional charged partial positive surface areas
18	WNSA1	Surface weighted charged partial negative surface areas
19	WNSA2	Surface weighted charged partial negative surface areas
20	WNSA3	Surface weighted charged partial negative surface areas
21	WPSA1	Surface weighted charged partial positive surface areas
22	WPSA2	Surface weighted charged partial positive surface areas
23	WPSA3	Surface weighted charged partial positive surface areas
24	TASA	Total hydrophobic surface area
25	PSA	Total polar surface area
26	FrTATP	The fraction between TASA and TPSA
27	RASA	Relative hydrophobic surface area
28	RPSA	Relative polar surface area
29	RNCS	Relative negative charge surface area
30	RPCS	Relative positive charge surface area

WHIM descriptors (70)		
1_14	WHIM1-WHIM14	Unweighted WHIM descriptors
15-28	WHIM15-WHIM28	WHIM descriptors based on atomic mass
29-42	WHIM29-WHIM42	WHIM descriptors based on Sanderson Electronegativity
43-56	WHIM43-WHIM56	WHIM descriptors based on VDW Volume
57-70	WHIM57-WHIM70	WHIM descriptors based on Polarizability

MoRSE descriptors (210)		
1_30	MoRSEU1-30	Unweighted 3-D MoRse descriptors
31-60	MoRSEC1-30	3-D MoRse descriptors based on atomic charge
61-90	MoRSEM1-30	3-D MoRse descriptors based on atomic mass
91-120	MoRSEN1-30	3-D MoRse descriptors based on atomic number
121-150	MoRSEP1-30	3-D MoRse descriptors based on atomic polarizability
151-180	MoRSEE1-30	3-D MoRse descriptors based on atomic Sanderson electronegativity
181-210	MoRSEV1-30	3-D MoRse descriptors based on atomic van der Waals volume

RDF descriptors (180)		
1_30	RDFU1-30	Unweighted radial distribution function (RDF) descriptors
31-60	RDFC1-30	Radial distribution function (RDF) descriptors based on atomic charge.
61-90	RDFM1-30	Radial distribution function (RDF) descriptors based on atomic mass
91-120	RDFP1-30	Radial distribution function (RDF) descriptors based on atomic polarizability
121-150	RDFE1-30	Radial distribution function (RDF) descriptors based on atomic electronegativity
151-180	RDFV1-30	Radial distribution function (RDF) descriptors based on atomic van der Waals volume

Fragment/Fingerprint-based descriptors		
1	FP2	(Topological fingerprint) A Daylight-like fingerprint based on hashing molecular subgraphs
2	MACCS	(MACCS keys) Using the 166 public keys implemented as SMARTS
3	E-state	79 E-state fingerprints or fragments

4	FP4	307 FP4 fingerprints
5	Atom Pairs	Atom Pairs fingerprints
6	Torsions	Topological torsion fingerprints
7	Morgan/Circular	Fingerprints based on the Morgan algorithm

Table A.2: Performance metrics of models using all molecule samples and active molecule samples.

a) Mean absolute error (MAE) for all molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	1.464	1.463	1.464	1.462	1.465	1.466	1.47	1.473
Neural Network	1.678	1.678	1.678	1.678	1.678	1.678	1.678	1.678
Linear SVR	1.464	1.463	1.464	1.462	1.465	1.466	1.47	1.473
Decision Tree	1.451	1.47	1.47	1.47	1.47	1.456	1.456	1.448
Ridge Regression	1.464	1.463	1.464	1.462	1.465	1.466	1.47	1.473
Bayesian Ridge Regression	1.464	1.463	1.463	1.462	1.463	1.464	1.465	1.466
Lasso	1.471	1.463	1.464	1.468	1.462	1.463	1.461	1.464
Random Forest regression	1.413	1.398	1.42	1.416	1.431	1.42	1.416	1.42

b) Mean absolute error (MAE) for active molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	6.405	6.384	6.357	6.337	6.276	6.268	6.253	6.262
Neural Network	14.275	14.275	14.275	14.275	14.275	14.275	14.275	14.275
Linear SVR	6.405	6.384	6.356	6.337	6.276	6.267	6.247	6.256
Decision Tree	6.463	6.489	6.463	6.465	6.465	6.463	6.463	6.465
Ridge Regression	6.405	6.384	6.356	6.337	6.276	6.266	6.247	6.256
Bayesian Ridge Regression	6.406	6.384	6.355	6.345	6.304	6.288	6.273	6.269
Lasso	6.413	6.413	6.378	6.345	6.278	6.268	6.281	6.346
Random Forest regression	6.334	6.294	6.312	6.344	6.367	6.345	6.374	6.374

c) Mean square error (MSE) for all molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	15.164	15.13	15.115	15.081	15.042	15.018	14.98	14.964
Neural Network	15.349	15.349	15.349	15.349	15.349	15.349	15.349	15.349
Linear SVR	15.164	15.13	15.115	15.081	15.042	15.018	14.98	14.961
Decision Tree	15.075	15.241	15.241	15.241	15.241	15.176	15.176	15.202
Ridge Regression	15.164	15.13	15.115	15.081	15.042	15.018	14.98	14.962

Bayesian Ridge Regression	15.164	15.13	15.115	15.084	15.05	15.022	14.994	14.972
Lasso	15.211	15.142	15.153	15.174	15.125	15.13	15.118	15.144
Random Forest regression	14.608	14.504	14.783	14.692	14.752	14.631	14.689	14.728

d) Mean square error (MSE) for active molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	81.188	80.801	80.604	80.264	79.336	79.221	79.143	79.341
Neural Network	287.633	287.633	287.633	287.633	287.633	287.633	287.633	287.633
Linear SVR	81.188	80.801	80.603	80.262	79.338	79.200	79.077	79.188
Decision Tree	82.855	82.251	82.855	82.384	82.384	82.855	82.855	82.384
Ridge Regression	81.188	80.801	80.603	80.262	79.340	79.198	79.072	79.190
Bayesian Ridge Regression	81.126	80.629	80.306	80.174	79.538	79.328	79.102	79.059
Lasso	81.201	80.956	80.385	80.212	79.348	79.195	79.193	79.776
Random Forest regression	76.645	75.487	76.521	78.174	78.872	76.120	77.208	79.215

e) Median absolute error (MedAE) for all molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	0.726	0.72	0.72	0.713	0.709	0.706	0.711	0.711
Neural Network	1	1	1	1	1	1	1	1
Linear SVR	0.726	0.72	0.72	0.713	0.709	0.706	0.711	0.711
Decision Tree	0.49	0.547	0.547	0.547	0.547	0.49	0.49	0.632
Ridge Regression	0.726	0.72	0.72	0.713	0.709	0.706	0.711	0.711
Bayesian Ridge Regression	0.726	0.721	0.722	0.713	0.709	0.708	0.707	0.709
Lasso	0.737	0.726	0.727	0.734	0.719	0.719	0.717	0.723
Random Forest regression	0.553	0.527	0.546	0.552	0.581	0.542	0.542	0.556

f) Median absolute error (MedAE) for active molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	5.184	5.257	5.243	5.275	5.186	5.150	5.155	5.153
Neural Network	9.000	9.000	9.000	9.000	9.000	9.000	9.000	9.000
Linear SVR	5.184	5.256	5.243	5.274	5.177	5.150	5.131	5.156
Decision Tree	4.650	5.423	4.650	4.409	4.409	4.650	4.650	4.409
Ridge Regression	5.184	5.256	5.246	5.276	5.180	5.155	5.115	5.147
Bayesian Ridge Regression	5.189	5.261	5.258	5.299	5.220	5.203	5.148	5.150

Lasso	5.170	5.184	5.242	5.300	5.191	5.186	5.179	5.244
Random Forest regression	4.655	4.651	4.667	4.770	4.787	4.716	4.695	4.748

g) R^2 score for all molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Neural Network	-0.003	-0.003	-0.003	-0.003	-0.003	-0.003	-0.003	-0.003
Linear SVR	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Decision Tree	0.02	0.009	0.009	0.009	0.009	0.013	0.013	0.012
Ridge Regression	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Bayesian Ridge Regression	0.014	0.016	0.017	0.019	0.021	0.023	0.025	0.027
Lasso	0.011	0.016	0.015	0.013	0.017	0.016	0.017	0.015
Random Forest regression	0.05	0.057	0.039	0.045	0.041	0.049	0.045	0.042

h) R^2 score for active molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	0.032	0.036	0.039	0.043	0.054	0.055	0.056	0.054
Neural Network	-2.430	-2.430	-2.430	-2.430	-2.430	-2.430	-2.430	-2.430
Linear SVR	0.032	0.036	0.039	0.043	0.054	0.056	0.057	0.056
Decision Tree	0.012	0.019	0.012	0.018	0.018	0.012	0.012	0.018
Ridge Regression	0.032	0.036	0.039	0.043	0.054	0.056	0.057	0.056
Bayesian Ridge Regression	0.033	0.038	0.042	0.044	0.051	0.054	0.057	0.057
Lasso	0.032	0.035	0.041	0.043	0.054	0.056	0.056	0.049
Random Forest regression	0.086	0.100	0.087	0.068	0.059	0.092	0.079	0.055

i) Explained variance score (EVS) for all molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Neural Network	0	0	0	0	0	0	0	0
Linear SVR	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Decision Tree	0.02	0.009	0.009	0.009	0.009	0.013	0.013	0.012
Ridge Regression	0.014	0.016	0.017	0.019	0.022	0.024	0.026	0.027
Bayesian Ridge Regression	0.014	0.016	0.017	0.019	0.021	0.023	0.025	0.027
Lasso	0.011	0.016	0.015	0.013	0.017	0.016	0.017	0.015

Random Forest regression	0.05	0.057	0.039	0.045	0.041	0.049	0.045	0.042

j) Explained variance score (EVS) for active molecule samples

Number of features	**25**	**50**	**75**	**100**	**150**	**200**	**250**	**300**
Linear Regression	0.032	0.037	0.039	0.043	0.054	0.055	0.056	0.054
Neural Network	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Linear SVR	0.032	0.037	0.039	0.043	0.054	0.056	0.057	0.056
Decision Tree	0.012	0.019	0.012	0.018	0.018	0.012	0.012	0.018
Ridge Regression	0.032	0.037	0.039	0.043	0.054	0.056	0.057	0.056
Bayesian Ridge Regression	0.033	0.039	0.042	0.044	0.052	0.054	0.057	0.057
Lasso	0.032	0.035	0.041	0.044	0.054	0.056	0.056	0.049
Random Forest regression	0.086	0.100	0.088	0.068	0.060	0.093	0.080	0.056