Using Machine Learning to Predict Drug-Drug Interactions

Evaluation of Supervised Machine Learning Algorithms in the Ability to Predict DDIs

Hwa Chong Institution

Ng Jing Jie, Asher

Tang Wei Herng, Calen

Academies of Loudoun

Raghav Krisnaswamy

Shalini Panthangi

Abstract

Drug-drug interactions are an often overlooked aspect of the medical field which can have drastic implications. During the prescription and consumption of drugs, adverse drug reactions may result which have significant impacts on one's health. However, limitations in clinical trials mean that ADRs may only be detected when they happen after approval for clinical use. Hence, to assist in the prediction of DDIs, machine learning algorithms can be used to identify drugs with a high potential to have interactions. This project is a research collaboration between Hwa Chong Institution (HCI) and the Academies of Loudoun (ACL). Our project uses data from the DrugBank database, including Anatomical Therapeutic Classification codes and Simplified Molecular Line-Entry System codes, as well as the drug interactions. We obtained 2,770 drugs with ATC and SMILES codes as valid drugs for analysis. By extracting interactions of each type into an individual CSV file, we were able to analyse the drug properties of each drug, running KNN, Decision Tree regression and classification, Random Forest regression and classification, and naive Bayes to obtain a few models. We then ran various metrics on the models, finding that Decision Tree produces the best classification and regression model.

Contents

1 Introduction

1.1 Introduction

Drug-drug interactions (DDIs) are defined as a change in a drug's effect when two or more are co-administered. This often results in various outcomes, including that of increased or decreased action, and adverse drug reactions (ADRs). This project focuses on the latter, which can result in severe side effects that are potentially lethal. The challenge faced by the scientific community in this day and age is that many DDIs are only identified through clinical testing phases. However, in reality, there are far too many drugs to experimentally test. As a result, numerous cases of ADRs occur only after such medicines are administered. In 2019 alone, 36,847 cases of ADRs were recorded in Singapore, hence suggesting the urgent need to determine DDIs before drugs are approved.

Thus, the premise of this project: to make use of machine-learning algorithms to identify drugs with high potential of resulting in DDIs through the use of calculated probability, and the evaluation of said methods to determine its accuracy and area of feasibility. This research was a collaboration between Hwa Chong Institution (HCI) and the Academies of Loudoun (ACL).

1.2 Research Questions

The main aims of the research are as follows:

1. Construct a drug database based on therapeutic, chemical and interactive properties.
2. Perform machine learning algorithms for the prediction of drug-drug interactions.
3. Evaluate accuracy of each model through classification and regression metrics.

1.3 Literature Review

The following research paper done by Cheng and Zhao in 2014 is the main literature review that we will be utilising. They used a heterogeneous network-assisted interface (HNAI) framework to assist the prediction of DDIs. They used the DrugBank database of DDIs as their main source, using 6946 DDI pairs with 721 drugs in the database. By comparing similarities between these drug-drug pairs using properties such as the Anatomical Therapeutic Chemical classification system, chemical structural similarity from SMILES data, and genomic similarity, they constructed a network using DrugBank and the Therapeutic Target Database, as shown in Figure 1.1.

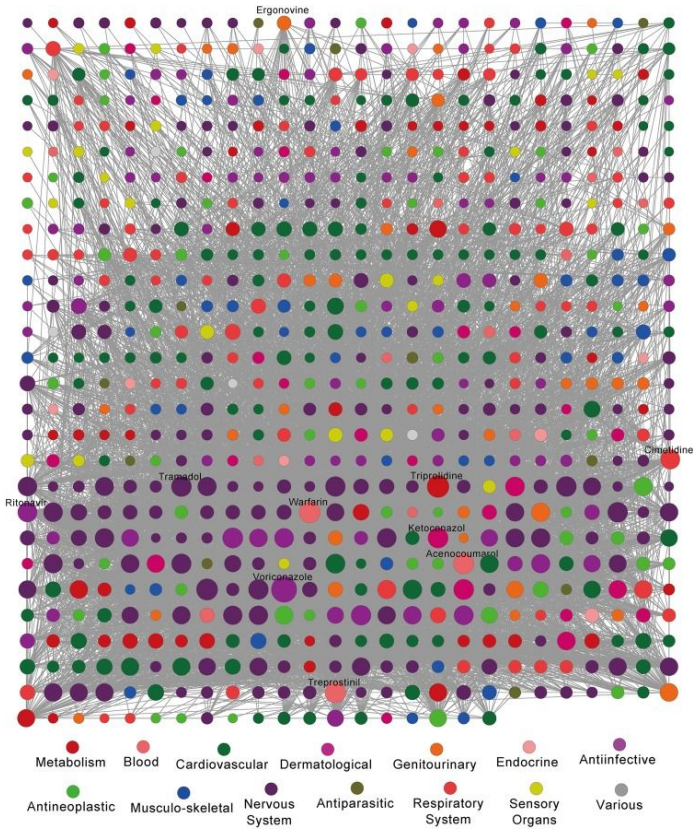

Figure 1.1: Drug-drug pair network from the HNAI framework.

They applied primarily classification algorithms in the framework, namely naive Bayes, decision tree, k-nearest neighbours and support vector machine, as well as logistic regression. Their models obtained a value for the area under the receiver operating curve (AUROC) of 0.67. The results obtained by their metrics are shown in Figure 1.2.

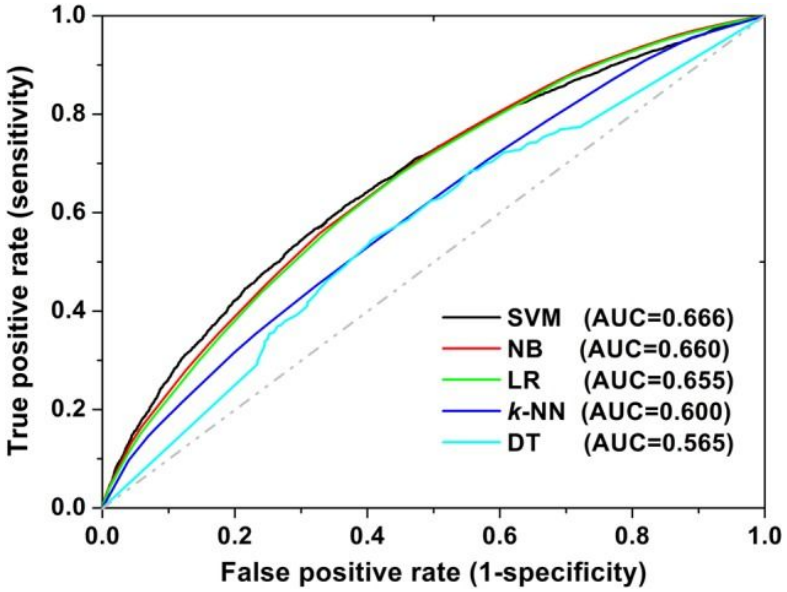

Figure 1.2: AUROC Scores for Classification Models in Predicting DDIs

This research paper demonstrates the feasibility of machine learning algorithms in the medical field, specifically in the prediction of DDIs. However, our project differs in a few areas. We will be using a different framework, utilising a database of individual drugs rather than drug-drug pairs. Also, their research only predicts the presence of DDIs and doesn't specify the type, while our project will analyse the drugs by interaction to identify any interactions between drugs for that specific type of interaction. As mentioned above, we will also be using different models, including random forest and decision tree regression.

1.4 Methodology

The following is a brief description of the steps taken during the research. We obtained the required information from the DrugBank database, which includes the Anatomical Therapeutic Classification (ATC) code, Simplified Molecular-Input Line-Entry System (SMILES) codes, as well as the DDIs proper. In order to extract the data from the dataset, which is in XML format, we will be using a Python package for the parsing and extraction of data. Python will also be used for the analysis of the data using models. For the identification of interaction types, we will be manually running through the dataset to find all possible interactions. The algorithms we will be using for the analysis are k-nearest neighbours (KNN) classification, decision tree (DT) regression and classification, random forest (RF) regression and classification, as well as naive Bayes classification. We will be using different metrics to evaluate the regression and classification models, with mean absolute error (MAE), root mean squared error (RMSE) and r-squared error for regression, and precision, recall, f1 score and balanced accuracy for classification.

2 Results

2.1 Research Question 1

The first research question was to construct a comprehensive database of various drug properties. In particular, this research looked at 3 main aspects in order to predict the DDIs of given drugs, which were classified as the chemical, therapeutic and interactive properties. The bulk of data was obtained from DrugBank, an open-source website that contained detailed drug data and drug target information. However, the dataset was formatted in eXtensible Markup Language (XML) with a total of 13,640 drugs, of which majority had data on required properties missing. Hence, Python was used to format the data by filtering out unused drugs while converting the dataset to a Comma-Separated Values (CSV) file for data analysis.

In order to evaluate the chemical property, we looked at the SMILES code of each drug. This is a form of line notation that describes the chemical structure of a species through short ASCII strings. In graph theory, the string is obtained by a depth-first traversal of the chemical graph for each given drug. Figure 2.1.1 displays the process of the SMILES code generation for Ciprofloxacin, starting from its structural formula.

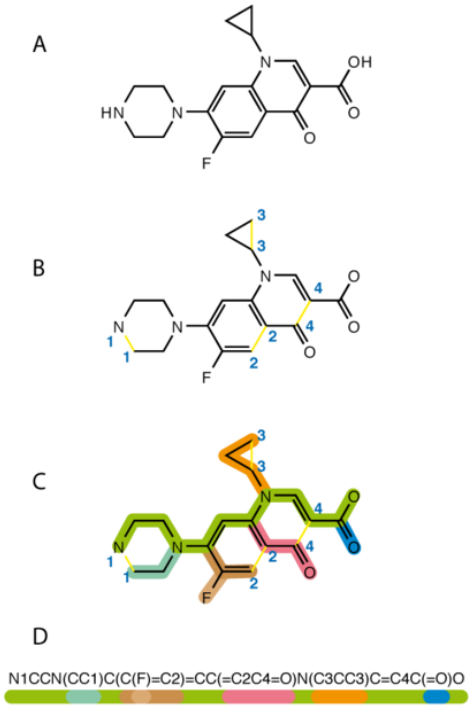

Figure 2.1.1: SMILES Generation Algorithm for Ciprofloxacin

Hence, we were able to compare the similarity of the SMILES codes through Tanimoto's coefficient, defined as the size of the intersection divided by the size of union sets, given by:

$$T(a,b) = \frac{N_c}{N_a + N_b - N_c}$$

where N is the number of attributes for objects a and b, and c is the intersecting set. This gives a similarity index of value 0 to 1 inclusive based on how similar the chemical structures of a pair of drugs are.

For the therapeutic properties of each drug, we looked at their ATC codes. This is a classification system for medicines based on their receptor sites and pharmacological properties. The ATC code system is divided into 5 levels of identification, with each code having only one parent code with the exception of the 14 codes on the first level of identification. Each level is classified in ascending order of a drug, anatomical main group, therapeutic subgroup, pharmacological subgroup, chemical subgroup and the last identifies the chemical substance. Table 2.1.1 provides an example of the classification of Metformin.

Table 2.1.1: Table of ATC Code Identification Levels for Metformin

A	Alimentary tract and metabolism (1st level, anatomical main group)
A10	Drugs used in diabetes (2nd level, therapeutic subgroup)
A10B	Blood glucose lowering drugs, excl. insulins (3rd level, pharmacological subgroup)
A10BA	Biguanides (4th level, chemical subgroup)
A10BA02	metformin (5th level, chemical substance)

To compare the similarity of ATC codes, we compared the first 3 characters for each pair of strings. Similar pairs were indicated with a value of 1, while pairs which did not match were denoted with 0, hence giving the therapeutic similarity.

Lastly, the interactive properties of each drug were investigated by examining the existing DDIs and their reacting drugs. Data on the DDIs provided in the DrugBank dataset was provided in the form of description strings (e.g. “The metabolism of Cyclosporine can be decreased when combined with Ziprasidone”). A Python code was used to iterate through the XML file and identify keywords for each interaction, resulting in 147 different types of interactions identified for the filtered data. A csv file was then created for each interaction, where the data was converted to a binary matrix. Whether there was an interaction present or not between a drug pair was denoted with a 1 or 0 respectively.

From the data preprocessing, we were able to obtain the following datasets. Table 2.1.2 shows a data sample of the combined table for drug id (DrugBank identification code was used), ATC code and SMILES code of the 2,770 filtered drugs.

Table 2.1.2: Table of Drug ID, ATC code and SMILES structure for Data Sample 1-20:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	DBID	atc	smiles										
2	DB00006	B01AE06	CC[C@H](C)[C@H](NC(=O)[C@H](CCC(O)=O)NC(=O)[C@H](CCC(O)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(										
3	DB00014	L02AE03	CC(C)C[C@H](NC(=O)[C@@H](COC(C)(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2										
4	DB00027	R02AB30	CC(C)C[C@@H](NC(=O)CNC(=O)[C@@H](NC=O)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(C)C)C(=O)N[C@@H](C(C)C)C(=										
5	DB00035	H01BA02	NC(=O)CC[C@@H]1NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CCSSC[C@H](NC(=O)[C@H]										
6	DB00050	H01CC02	CC(C)C[C@H](NC(=O)[C@@H](CCCNC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](CC1=C										
7	DB00080	J01XX09	CCCCCCCCCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(O)=O)C(=O)N[C@H]1										
8	DB00091	L04AD01	CC[C@@H]1NC(=O)[C@H]([C@H](O)[C@H](C)C\C=C\C)N(C)C(=O)[C@H](C(C)C)N(C)C(=O)[C@H](CC(C)C)N(C)C(=O)[C@H										
9	DB00104	H01CB02	[H][C@]1(NC(=O)[C@H](CCCCN)NC(=O)[C@@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H]((										
10	DB00106	L02BX01	CC(C)C[C@H](NC(=O)[C@@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)N(C)C(=O)[C@H](CO)NC(=O)[C@@H](CC1=CN										
11	DB00114	A11HA06	CC1=NC=C(COP(O)(O)=O)C(C=O)=C1O										
12	DB00115	B03BA01	[C@H]1([C@@H]2[C@H]([C@@H](N3C4=CC(=C(C)C)C=C4[N+]([Co-3]456(N7C8=C(C9=[N+]4C(C([C@@H]9CCC(N)=O)(C)C)										
13	DB00118	A16AA02	[H][C@](N)(CC[S+](C)C[C@@]1([H])O[C@@]([H])(N2C=NC3=C(N)N=CN=C23)[C@]([H])(O)[C@]1([H])O)C([O-])=O										
14	DB00121	A11HA05	[H][C@]12CS[C@@H](CCCCC(O)=O)[C@@]1([H])NC(=O)N2										
15	DB00123	B05XB03	NCCCC[C@H](N)C(O)=O										
16	DB00125	B05XB01	N[C@@H](CCCNC(N)=N)C(O)=O										
17	DB00126	G01AD03	[H][C@@]1(OC(=O)C(O)=C1O)[C@@H](O)CO										
18	DB00129	A05BA06	NCCC[C@H](N)C(O)=O										
19	DB00130	A16AA03	N[C@@H](CCC(N)=O)C(O)=O										
20	DB00134	V03AB26	CSCC[C@H](N)C(O)=O										

Table 2.1.3 displays a combined matrix for all drug interactions between the filtered drug pairs, from which individual binary matrices were extracted for every individual interaction, as shown in Table 2.1.4.

Table 2.1.3: Table of Combined DDIs for Data Sample 1-21

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	ID	DB00006	DB00014	DB00027	DB00035	DB00050	DB00080	DB00091	DB00104	DB00106	DB00114	DB00115	DB00118
2	DB00006	0	0	0	0	0	0	0	0	0	0	0	0
3	DB00014	0	0	0	Goserelin ı	0	Goserelin ı	Ciclosporir	The risk or	0	0	Goserelin ı	0
4	DB00027	0	0	0	0	0	0	The serum	The serum	0	0	0	0
5	DB00035	0	Goserelin ı	0	0	0	Desmopre	Ciclosporir	0	0	0	Desmopre	0
6	DB00050	0	0	0	0	0	0	0	0	0	0	0	0
7	DB00080	0	Goserelin ı	0	Desmopre	0	0	Ciclosporir	0	0	0	Daptomyc	0
8	DB00091	0	Ciclosporir	The serum	Ciclosporir	0	Ciclosporir	0	The serum	0	0	Ciclosporir	0
9	DB00104	0	The risk or	The serum	0	0	0	The serum	0	0	0	0	0
10	DB00106	0	0	0	0	0	0	0	0	0	0	0	0
11	DB00114	0	0	0	0	0	0	0	0	0	0	0	0
12	DB00115	0	Goserelin ı	0	Desmopre	0	Daptomyc	Ciclosporir	0	0	0	0	0
13	DB00118	0	0	0	0	0	0	0	0	0	0	0	0
14	DB00121	0	0	0	0	0	0	0	0	0	0	0	0
15	DB00123	0	0	0	0	0	0	0	0	0	0	0	0
16	DB00125	0	0	0	0	0	0	0	0	0	0	0	0
17	DB00126	0	0	The serum	0	0	0	The serum	The serum	0	0	0	0
18	DB00129	0	0	0	0	0	0	0	0	0	0	0	0
19	DB00130	0	0	0	0	0	0	0	0	0	0	0	0
20	DB00134	0	0	0	0	0	0	0	0	0	0	0	0
21	DB00136	0	0	0	0	0	0	The metab	0	0	0	0	0

Table 2.1.4: Table of Binary Matrix for DDI Headache for Data Sample 1-21:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	ID	DB00006	DB00014	DB00027	DB00035	DB00050	DB00080	DB00091	DB00104	DB00106	DB00114	DB00115	DB00118
2	DB00006	0	0	0	0	0	0	0	0	0	0	0	0
3	DB00014	0	0	0	0	0	0	0	0	0	0	0	0
4	DB00027	0	0	0	0	0	0	0	0	0	0	0	0
5	DB00035	0	0	0	0	0	0	0	0	0	0	0	0
6	DB00050	0	0	0	0	0	0	0	0	0	0	0	0
7	DB00080	0	0	0	0	0	0	0	0	0	0	0	0
8	DB00091	0	0	0	0	0	0	0	0	0	0	0	0
9	DB00104	0	0	0	0	0	0	0	0	0	0	0	0
10	DB00106	0	0	0	0	0	0	0	0	0	0	0	0
11	DB00114	0	0	0	0	0	0	0	0	0	0	0	0
12	DB00115	0	0	0	0	0	0	0	0	0	0	0	0
13	DB00118	0	0	0	0	0	0	0	0	0	0	0	0
14	DB00121	0	0	0	0	0	0	0	0	0	0	0	0
15	DB00123	0	0	0	0	0	0	0	0	0	0	0	0
16	DB00125	0	0	0	0	0	0	0	0	0	0	0	0
17	DB00126	0	0	0	0	0	0	0	0	0	0	0	0
18	DB00129	0	0	0	0	0	0	0	0	0	0	0	0
19	DB00130	0	0	0	0	0	0	0	0	0	0	0	0
20	DB00134	0	0	0	0	0	0	0	0	0	0	0	0
21	DB00136	0	0	0	0	0	0	0	0	0	0	0	0

2.2 Research Question 2

The six algorithms used in this project are k-nearest neighbours (KNN), decision tree (DT) regression and classification, random forest (RF) regression and classification, and Naive Bayes. These 6 models were split between HCI and ACL, with HCI taking KNN and both DT models, while ACL takes naive Bayes and both RF models.

Classification in this project assigns either a 1 (DDI) or 0 (no DDI) to each drug based on the model, while regression assigns a value between 0 and 1 inclusive indicating the percentage chance of an interaction. We will be using multivariate regression for our regression models.

KNN is a classification model that uses the data of the k closest data points. As can be seen below in Fig. 2.2.1, we ran KNN on values of k from 1 to 40, determining the best k value to be 5. This means that for the 5 most similar drugs to the baseline drug, a plurality vote is taken. If at least 3 of the 5 most similar drugs have an interaction with a certain drug, the baseline drug is determined to have an interaction. Conversely, if a majority of the 5 closest drugs do not interact with a certain drug, the baseline drug does not have an interaction.

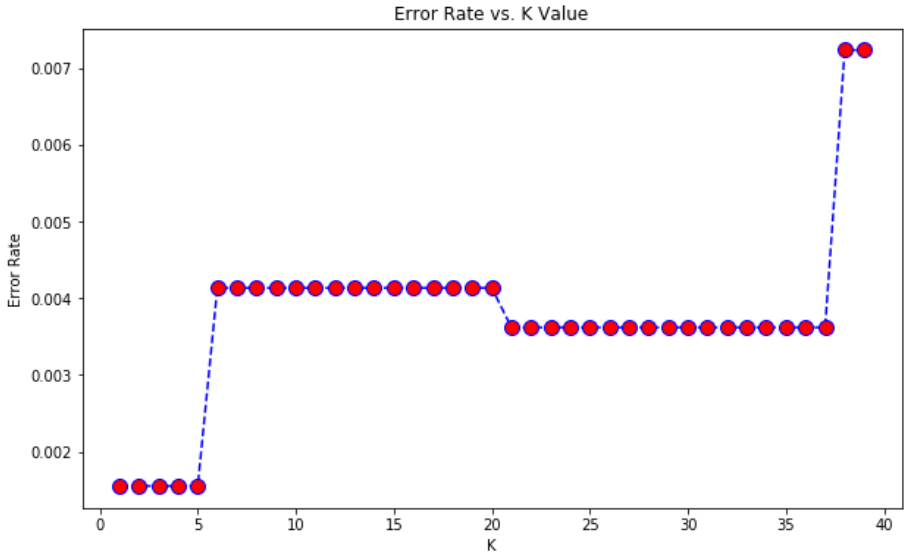

Figure 2.2.1: Graph of error rate against k value

DT makes use of classification trees to either classify or determine the percentage chance of an event. In most DT algorithms, including the one that is used in this project, the nodes of the tree are determined by how evenly the data can be split. At each node, a true/false question is asked of every data point with the data being grouped accordingly. The final decision tree is then used to determine the percentage chance from the leaf nodes.

RF, similarly to DT, uses classification trees, with the primary difference being that RF, as its name suggests, uses randomly determined nodes and a user-defined number of trees. This has the function of minimising overfitting of the dataset, as the

decision tree has the potential to become too deep and hence result in overfitting of the dataset.

Naive Bayes, as its name suggests, makes use of Bayes' Theorem in order to find the percentage chance of the occurrence of an event. It is considered naive as it assumes the independence of all input variables, which may not always be the case. However, it is still a fairly accurate model in most situations.

2.3 Research Question 3

For the evaluation of the 6 prediction models, 4 classification and 3 regression metrics were implemented. For classification, the metrics used include precision, recall, accuracy and f1-score, which are all derived from the confusion matrix that lists the number of true positives (TP), false positives (FP) true negatives (TN) and false negatives (FN). For regression, the metrics used include MAE, RMSE and R-squared error.

In classification, precision is defined as the number of true positives divided by the total number of elements labelled under the positive class, given by:

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

Recall, on the other hand, is defined as the number of true positives divided by the total number of elements that actually belong to the positive class, given by:

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

Accuracy is used to evaluate the ability of the precision model to label each class correctly. We used an alternate form known as balanced accuracy that normalizes the true positive and true negative predictions, for our dataset which had a majority of '0' predictions. Balanced accuracy is given by:

$$\text{Balanced accuracy} = \frac{TPR + TNR}{2}$$

where *TPR* is the true positive rate and *TNR* is the true negative rate. Finally, f1-score is used as a combined evaluation of precision and recall metrics by taking the harmonic mean, thereby penalizing extreme values. F1-score is given by:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

For the regression metrics, MAE measures the distance between test values and predicted values, given by:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |x_i - x|$$

Similarly, RMSE finds the distance between test and predicted values, but provides larger weightage to extreme values and outliers, given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2}$$

Finally, we looked at the R-squared value of each regression model, which is also known as the coefficient of determination. This measures how well the model explains the output values, given by:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

where SS_{res} is the sum of squared residuals and SS_{tot} is the total sum of squared results.

With the above metrics, we were able to obtain the following results. The prediction models were split between HCI and ACL where k-NN and Decision Tree were done by HCI while Naive Bayes and Random Forest were done by ACL. In each of the tests, a certain drug was removed from the dataset, after which the models were trained using the remaining data to predict the interactions of the removed drug. A test set was then used containing the removed drug to evaluate the accuracy of each model.

The first test was done using Pamidronic acid (DB00282) for the interaction: gastrointestinal ulceration. This test evaluated the ability of the prediction models to correctly predict if an interaction was present for an unknown drug. Table 2.3.1 displays the metric scores for the classification models, and Table 2.3.2 displays the metric scores for the regression models.

Table 2.3.1: Classification Model Metric Scores for DB00282, Gastrointestinal Ulceration

	Precision	Recall	F1-Score	Balanced Accuracy
k-NN	0.9948595012727	0.9948320413436	0.9946161677744	0.9206349206349
Decision Tree	1.0	1.0	1.0	1.0
Random Forest	0.9989664082687	0.9989675113548	0.9989583487703	0.9841269841269
Naive Bayes	0.9994834800531	0.9994832041343	0.9994812059490	0.9920634920634

Table 2.3.2: Regression Model Metric Scores for DB00282, Gastrointestinal Ulceration

	Mean Absolute Error	Root Mean Squared Error	R-Squared
Decision Tree	0.0	0.0	1.0
Random Forest	0.00158139534	0.02046235593	0.98670688339

It was found that most models had relatively high ability to accurately predict the interactions with regards to DB00282 and gastrointestinal ulceration. It was noted that Decision Tree classification and regression were able to predict with full accuracy in this instance.

The next test investigated the ability for the models to predict which drug interacted with Lidocaine (DB00281) for gastrointestinal ulceration. This drug was selected as it did not contain any DDIs with other drugs that caused gastrointestinal

ulceration as an ADR. Hence, this test was used to evaluate if the prediction models could correctly predict if a drug had no interactions with any other in the dataset. Table 2.3.3 displays the metric scores for the classification models, while Table 2.3.4 displays the metric scores for the regression models..

Table 2.3.3: Classification Model Metric Scores for DB00281, Gastrointestinal Ulceration

	Precision	Recall	F1-Score	Balanced Accuracy
k-NN	1.0	1.0	1.0	1.0
Decision Tree	1.0	1.0	1.0	1.0
Random Forest	1.0	1.0	1.0	1.0
Naive Bayes	1.0	1.0	1.0	1.0

Table 2.3.4: Regression Model Metric Scores for DB00281, Gastrointestinal Ulceration

	Mean Absolute Error	Root Mean Squared Error	R-Squared
Decision Tree	0.0	0.0	1.0
Random Forest	0.0	0.0	1.0

From the metric scores, it was found that all prediction models were able to predict with full accuracy if a drug had no interactions with the remaining drugs in the training

dataset. This suggests possible use in eliminating drugs from a list that requires clinical testing, thereby facilitating quicker analysis.

The final test was done using Cyclosporine (DB00091) for the interaction: liver damage. In the dataset retrieved only two other drugs produced liver damage as an ADR after being co-administered with DB00091. This test was used to describe the ability of the models to predict well when given little pre-existing information in the training set. The metric scores obtained are shown in Table 2.3.5 for classification models, and Table 2.3.6 for regression models.

Table 2.3.5: Classification Model Metric Scores for DB00091, Liver Damage

	Precision	Recall	F1-Score	Balanced Accuracy
k-NN	0.9948320413436	0.9896907904840	0.9922547562624	0.5
Decision Tree	0.9927648578811	0.9896801405599	0.9912200992848	0.4989610389610
Random Forest	0.9927648578811	0.9896801405599	0.9912200992848	0.4989610389610
Naive Bayes	0.0832041343669	0.9820187256551	0.1464012924338	0.4397402597402

Table 2.3.6: Regression Model Metric Scores for DB00091, Liver Damage

	Mean Absolute Error	Root Mean Squared Error	R-Squared
Decision Tree	0.0080103359173	0.0852492924338	-0.413555194805
Random Forest	0.0077843578057	0.0786524755808	-0.203250353631

It was found that all classification models performed poorly for accuracy and all regression models had low R-squared scores, indicating that such predictive models are limited by the amount of information given the training dataset. Upon closer inspection, most models predicted that the drug had no interactions entirely, hence giving a true positive rate of 0. It can thus be concluded that while the prediction models possess the ability to identify DDIs for unknown drugs, there is still a large room of improvement for largely untested domains.

3 Conclusion

3.1 Summary of Results

In the first research question, a dataset containing ATC and SMILES code was created using the DrugBank database. The Tanimoto coefficient was then used to evaluate the similarity values between drug pairs. For each interaction type, a separate dataset containing the interactions in the form of a binary matrix was created. Based on the datasets created in the first research question, the KNN, DT regression and classification, RF regression and classification, and naive Bayes algorithms were used in the second research question. In the third research question, several metrics were used on both the regression and classification algorithms, and it was found that for classification algorithms, DT was the best algorithm, followed by RF, naive Bayes and KNN. For regression algorithms, DT was the best algorithm followed by RF. However, the prediction models are only accurate when substantial information on pre-existing drug interactions is supplied, or when predicting drugs without interactions.

3.2 Applications

This project serves as a proof of concept for the use of machine learning in the prediction of DDIs and the medical field as a whole. Particularly, it can assist in

identifying drug pairs that do not need to be clinically tested to confirm predictions or have a low chance of having any interactions. Most importantly, however, it can be used to identify drugs pairs that have a high chance of having an interaction, such that they can be clinically tested to either confirm or disprove the prediction. This can help to minimise interactions that are only detected when a patient experiences the ADRs associated with that interaction.

3.3 Limitations

The data was fairly limited and the database was not very comprehensive with the properties, with most of the over 13,000 drugs in the dataset not having both ATC and SMILES codes, leaving us with just 2,770 drugs to work with. Some of the interaction types also had very few interactions, which resulted in the algorithms having great difficulty in creating an accurate model due to the small sample size. The limited pre-existing research on the use of machine learning to predict DDIs also caused limitations in the potential methods in comparing drug similarities, increasing the difficulty of the project.

3.4 Further Extensions

Given that all the algorithms used in this project are supervised, the use of unsupervised algorithms such as clustering and anomaly detection could give us different

results that may or may not be more accurate. Deep learning could also be implemented, which can potentially yield more accurate results. We could also extend the dataset to include drug-receptor and drug-protein interactions to be more comprehensive.

References

A History of Bayes' Theorem. (2013, February 16). Retrieved from http://georgemaciunas.com/exhibitions/knowledge-as-art-chance-computability-and-improving-education-thomas-bayes-alan-turing-george-maciunas/thomas-bayes/a-history-of-bayes-theorem/

Assessing the Fit of Regression Models. (2020, January 16). Retrieved from https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/

Brid, R. (2018, October 26). Decision Trees - A simple way to visualize a decision. Retrieved from https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Brownlee, J. (2019, May 21). Difference Between Classification and Regression in Machine Learning. Retrieved from https://machinelearningmastery.com/classification-versus-regression-in-machine-learning

Decision Tree Classification in Python. (n.d.). Retrieved August 06, 2020, from https://www.datacamp.com/community/tutorials/decision-tree-classification-python

Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined. (n.d.). Retrieved from https://pathmind.com/wiki/accuracy-precision-recall-f1

Koehrsen, W. (2018, January 17). Random Forest in Python. Retrieved from https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

Narkhede, S. (2019, May 26). Understanding AUC - ROC Curve. Retrieved from

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Yiu, T. (2019, August 14). Understanding Random Forest. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Appendix

Appendix A: Data Collection Code

```
import xml.etree.ElementTree as ET
import csv

tree = ET.parse("REDACTED")
root = tree.getroot()

with open('REDACTED', 'w', newline='', encoding="utf-8") as csvfile:

    fw = csv.writer(csvfile)
    headers = ["DBID", "atc" , "smiles"]
    fw.writerow(headers)

    for drug in root:
        atcode = ""
        smilescode = ""

        if drug.find('{http://www.drugbank.ca}drugbank-id').text in
drugList:
            id = drug.find('{http://www.drugbank.ca}drugbank-id').text

            for atccode in drug.find("{http://www.drugbank.ca}atc-codes"):
                atcode = atccode.attrib['code']
                break

            if drug.find('{http://www.drugbank.ca}calculated-properties'):
                for property in
drug.find('{http://www.drugbank.ca}calculated-properties'):
                    if property.find('{http://www.drugbank.ca}kind').text ==
"SMILES":
                        smilescode =
property.find('{http://www.drugbank.ca}value').text

            data = [id, atcode, smilescode]
            fw.writerow(data)
```

```
import xml.etree.ElementTree as ET
import csv

#Read Dataset - Change file path
tree = ET.parse("REDACTED")
root = tree.getroot()
headers = ["ID"]

headers.extend(druglist)

with open('REDACTED', 'w', newline='', encoding="utf-8") as csvfile:
    fw = csv.writer(csvfile)
    fw.writerow(headers)

    for drug in root:

        if drug.find('{http://www.drugbank.ca}drugbank-id').text in
druglist:
            id = drug.findall('{http://www.drugbank.ca}drugbank-id')[0].text
            data = [id]

            for i in range(len(druglist)):
                data.append(0)

            for drugInteraction in
drug.find("{http://www.drugbank.ca}drug-interactions"):
                tempId =
drugInteraction.find("{http://www.drugbank.ca}drugbank-id").text
                if tempId in druglist:
                    tempInt =
drugInteraction.find("{http://www.drugbank.ca}description").text

                    if data[headers.index(tempId)] == 0:
                        data[headers.index(tempId)] = tempInt
                    else:
                        data[headers.index(tempId)] += "; " + tempInt

            fw.writerow(data)
```

Appendix B: Data Analysis Code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors
from rdkit import RDConfig
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit.Chem import PandasTools as PandasTools
from rdkit import DataStructs
from rdkit.Chem.Subshape import
SubshapeBuilder,SubshapeAligner,SubshapeObjects

########################################################################

url = "REDACTED"

inputdata = ["DB00091", "L04AD01",
"CC[C@@H]1NC(=O)[C@H]([C@H](O)[C@H](C)C\C=C\C)N(C)C(=O)[C@H](C(C)C)N(C)C(=O)
[C@H](CC(C)C)N(C)C(=O)[C@H](CC(C)C)N(C)C(=O)[C@@H](C)NC(=O)[C@H](C)NC(=O)[C@
H](CC(C)C)N(C)C(=O)[C@@H](NC(=O)[C@H](CC(C)C)N(C)C(=O)CN(C)C1=O)C(C)C"]


dataset = pd.read_csv(url, names=names, skiprows=[0])
interdata = pd.read_csv("REDACTED", names=names2, skiprows=[0])

dataset = pd.concat([dataset, interdata], axis=1, sort=False)
del dataset['ID']
dataset = dataset.drop(dataset[dataset.DBID == inputdata[0]].index)
move_col = dataset.pop(inputdata[0])
dataset.insert(1, inputdata[0], move_col)

dataset.head()
atcsimarr = []
for colname, colval in dataset.iteritems():
    if colname == "atc":
        for i in colval:
            counter = 0
            if inputdata[1][0] == i[0]:
                if inputdata[1][1:3] == i[1:3]:
                    counter = 1
```

```
            atcsimarr.append(counter)

ser = pd.Series(atcsimarr)
dataset.insert(2, "atcSim", ser)
del dataset['atc']

dataset.head()
```

```
df = pd.read_csv("REDACTED")
molecules = df.smiles.apply(Chem.MolFromSmiles)

mols_fps=[]
for x in range(0,len(molecules)):
    if not molecules[x]:
        continue
    mols_fps.append(AllChem.GetMorganFingerprint(molecules[x],2))


ref = Chem.MolFromSmiles(inputdata[2])
ref = AllChem.GetMorganFingerprint(ref, 2)

sim_ref = DataStructs.BulkTanimotoSimilarity(ref, mols_fps)

ser = pd.Series(sim_ref)
dataset.insert(3, "smileSim", ser)
del dataset['smiles']
dataset.head()
```

```
dataset = dataset.dropna()
dataset = dataset.reset_index(drop=True)

X = dataset.iloc[:, 2:].values
y = dataset.iloc[:, 1].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)

import sklearn.metrics as metrics
accuracy = metrics.balanced_accuracy_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test, y_pred, average='weighted')
precision = metrics.recall_score(y_test, y_pred, average='weighted')
recall = metrics.precision_score(y_test, y_pred, average='weighted')
print("Accuracy:", accuracy)
print("F1_Score:", f1_score)
print("Precision:", precision)
print("Recall:", recall)

conf = metrics.confusion_matrix(y_test, y_pred)
print(conf)
```

```
from sklearn import tree
classifier = tree.DecisionTreeClassifier()
classifier = classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

import sklearn.metrics as metrics
accuracy = metrics.balanced_accuracy_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test, y_pred, average='weighted')
precision = metrics.recall_score(y_test, y_pred, average='weighted')
recall = metrics.precision_score(y_test, y_pred, average='weighted')
print("Accuracy:", accuracy)
print("F1_Score:", f1_score)
print("Precision:", precision)
print("Recall:", recall)

conf = metrics.confusion_matrix(y_test, y_pred)
print(conf)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=100)
classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test)

import sklearn.metrics as metrics
accuracy = metrics.balanced_accuracy_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test, y_pred, average='weighted')
precision = metrics.recall_score(y_test, y_pred, average='weighted')
recall = metrics.precision_score(y_test, y_pred, average='weighted')
```

```
print("Accuracy:", accuracy)
print("F1_Score:", f1_score)
print("Precision:", precision)
print("Recall:", recall)

conf = metrics.confusion_matrix(y_test, y_pred)
print(conf)
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

import sklearn.metrics as metrics
accuracy = metrics.balanced_accuracy_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test, y_pred, average='weighted')
precision = metrics.recall_score(y_test, y_pred, average='weighted')
recall = metrics.precision_score(y_test, y_pred, average='weighted')
print("Accuracy:", accuracy)
print("F1_Score:", f1_score)
print("Precision:", precision)
print("Recall:", recall)

conf = metrics.confusion_matrix(y_test, y_pred)
print(conf)
```

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

import sklearn.metrics as metrics
mae = metrics.mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
r2 = metrics.r2_score(y_test, y_pred)
print("Mean Absolute Error:", mae)
print("Root Mean Squared Error:", rmse)
print("R-Squared:", r2)
```

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
```

```
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)

import sklearn.metrics as metrics
mae = metrics.mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
r2 = metrics.r2_score(y_test, y_pred)
print("Mean Absolute Error:", mae)
print("Root Mean Squared Error:", rmse)
print("R-Squared:", r2)
```