Machine Learning Approaches to Drug Sensitivity Prediction

CSCI 5523 Fall 2020 Final Project Report

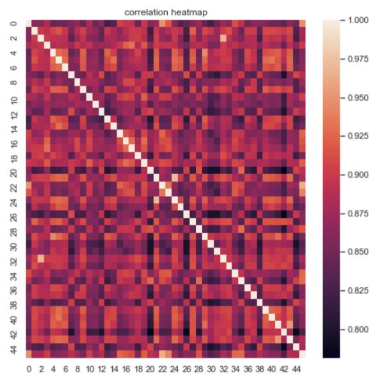
Yu Fang, Yu Han, Yixuan Wang, Xianjian Xie, Xiang Zhang

Introduction

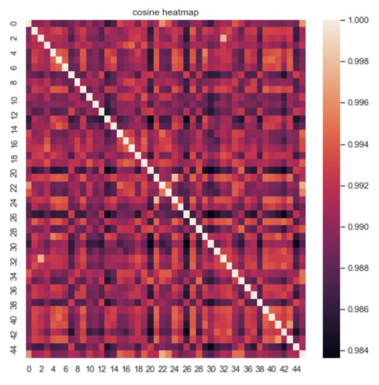
Predicting drug response to assist creating the best treatment strategy from genomic information is an essential goal of precision medicine. We have noticed the collaborative effort between the National Cancer Institute (NCI) and the Dialogue on Reverse Engineering Assessment and Methods (DREAM) project which provides a platform [1] for us to develop drug sensitivity prediction algorithms on a genomic profiling data set measured in human breast cancer cell lines. We mainly focused on selecting interesting drug targets and applying various supervised learning approaches to predict drug sensitivity for a panel of cell lines based on gene expression profiles, during which we also explored feature engineering and dimension reduction techniques.

Data Analysis

This dataset consists of 46 cancer cell lines, each with 18,637 attributes. The first five columns are a binary value indicating the cell's reactions to the target drugs ('1' corresponds to active, and '0' corresponds to inactive), and the rest of the columns are the gene expression profiles for 18,632 genes. This is a dataset with a small number of samples and high-dimensional features. The dataset analysis consists of two sections: data visualization and feature analysis. We first transposed the dataset to analyze how the drugs perform in each cell. In this case, we deployed heatmaps to visualize the correlation and cosine similarities between the cells. It turned out that the cell lines have a high level of similarities regarding the gene expression.

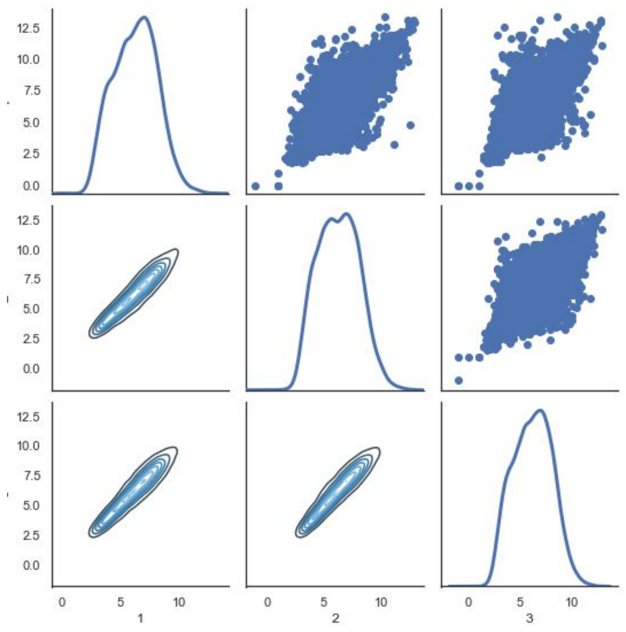


Correlation

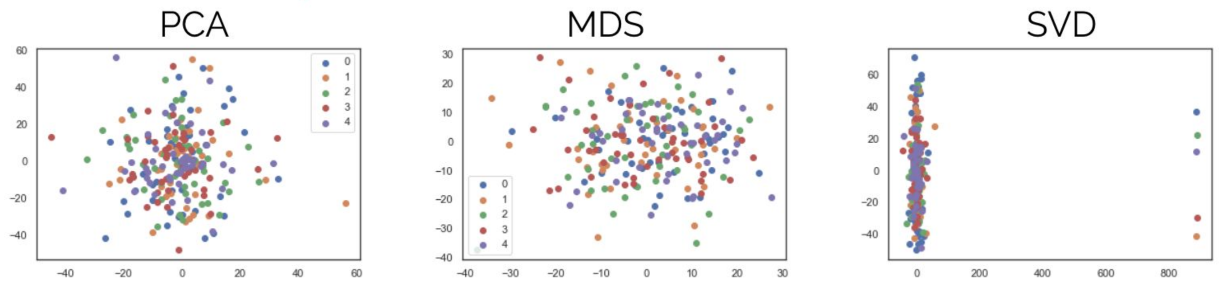


Cosine

Then, we had a closer look at the pairwise relationships by utilizing the jointplot and pairgrid functions. In this case, different functions on the upper and lower triangles of the plot are used to get further knowledge.



In regards to feature analysis, because there are over 18,000 features in the dataset, a few classic feature reduction analysis methods were used: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and Multidimensional scaling (MDS). For each approach, we reduced the dimensionality to be 46 and visualized the results by plotting the first five components.



Methods

Before training the model, model performance metrics should be introduced first, which are usually used to compare different models. All of the performances are based on the following four important values.

- TP: True positive, both actual and prediction are positive
- FP: False positive, actual negative and prediction positive
- FN: False negative, actual positive and prediction negative
- TN: True negative, both actual and prediction are negative

Using these 4 basic values, the following metrics can be calculated:

- Accuracy: (TP+TN)/(TP+TN+FP+FN), measuring the model prediction accuracy.
- ROC Curve: ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The method was developed for operators of military radar receivers, which is why it is so named. The ROC curve is created by plotting the true positive rate (TP/(TP+FN)) against the false positive rate (FP/(TN+FP)) at various threshold settings. This curve can compare different models directly according to the graphical plot.
- AUC value: The area under the ROC curve is called AUC. Its value is between 0 and 1.

The larger of these metrics the better models would be. In our analysis, ROC and AUC will be key measures for us to compare the model performance.

Regression

➢ *Step 1: Dimension Reduction - Principal components analysis (PCA)*

The data only has 46 records, but has more than 18,000 features, so the first job should be done is dimension reduction. PCA method is used in the analysis from sklearn. After setting the 99% target explained variance ratio, there would be more than 40 components. It is also too large. We found that the ratio after 10 would be very small, so 10 components will be the final choice. Re-train the PCA model with 10 components, the total explained variance ratio is 55%.

➢ *Step 2: Modeling*

Now, using the PCA results and different target variables, several models will be trained separately.

Another method that will be used in the analysis is cross validation, which is also called CV for short. The data set is divided into *k* subsets, Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. If *k* is the number of total records, this would be called Leave-One-Out cross validation, which we adopted in some cases.

❑ 2-1 Linear Regression Model

Model 1 is the linear regression model, and 5 models were trained on the data. We did k-fold on training data and got the optimal model parameter. We then test the generalization of the optimal model on the test set.

❑ 2-2 Logistic Regression Model

Model 2 is the logistic regression model, and 5 models were trained on the data. Logit function was used in Logistic Regression to measure the relationship between the target variable and independent variables.

- ❏ 2-3 LASSO Model

Model 3 is the LASSO model, which is based on the traditional regression model with L1 regularization. The L1 regularization adds a penalty equivalent to the absolute magnitude of regression coefficients and tries to minimize them, which allows some coefficients to be exactly zero. Thus, the LASSO model performs feature selection and returns a final model with lower number of parameters.

- ❏ 2-4 ElasticNet Model

Model 4 is the ElasticNet model. Elastic net uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve on the regularization of statistical models. It has 4 parameters called "alpha", "normalize", "l1_ratio" and "max_iter". Cross validation is also used.

- ❏ 2-5 Ridge Model

Model 5 is about the ridge regression. It is similar to the LASSO model, the only difference is that Ridge uses L2 regularization and LASSO uses L1 regularization and all coefficients won't be exactly zero, they will be close to zero but will never be zero. The function in the ElasticNet model can also be used with L1 ratio is always 0.

SVM

We applied the Support Vector Machines classifier to the dataset in a leave-one-out manner after the pre-processing of z-score standardization. Since the SVM can be interpreted as finding hyperplane boundaries for the two classes of data points, we are hoping that by performing standardizations, we are able to suffer less from the noise in the data. We applied the SVM with linear, polynomial and the Gaussian radial basis function kernel with different regularization parameters. We found that SVM is performing very well on predicting drug *Methylglyoxol* with an accuracy of 0.82 and drug *Everolimus* with 0.76, but relatively poorly on other drugs. In addition, since this dataset has a numerous number of attributes, we presume that the reason for SVM not performing ideally may be because of this. So we further conducted an experiment with Principal Component Analysis applied to the dataset before feeding it into the SVM; however, this contributes to worse results. So eventually we presented the results without PCA.

Decision Tree

Decision tree does not need standardization because it just determines a boundary for each feature but does not think about the relationship between the different features. We directly used the decision tree classifier to fit training data sets after eliminating the records with missing classification labels. Decision tree performs not too badly on the first four drugs, but performs poorly on the last drug. Increasing depth and features does not help, which is likely caused by interacting attributes in the data set.

Random Forest
Random forest, as an estimator that fits a number of decision tree classifiers on this data set, indeed improves the predictive accuracy because these classifiers protect each other from their individual errors, showing the power of ensemble methods.

KNN
After preprocessing the data with z-score standardization, we firstly applied K-Nearest Neighbor towards the high-dimensional data trying different metrics. It works well on the drug *Methylglyoxol*, getting an accuracy of 0.85 and precision of 0.88. However, the performances on the other 4 drugs are not that high, which is likely caused by the irrelevant attributes adding noise to the proximity measure and the redundant attributes biasing the proximity measure towards certain attributes. Therefore, to address this problem, we utilized the results after dimensionality reduction but no significant improvement was observed. So we would still present the results from the first attempt without dimensionality reduction.

Neural Networks
After preprocessing the data with z-score standardization, we firstly applied a three-layer Neural Network towards the high-dimensional data trying different levels of regularization and batch sizes. However, unlike our expectation that it would outperform the other algorithms, this Neural Network only achieves an average level of performance, and it predicts extremely badly on the drug *4-HC*. This might result from overfitting and unconvergence in global minimum.

We will further discuss our insights developed from all of the observations in the **Discussion**.

Results

The performance of each model we built for all of the 5 drugs has been collected as shown below, together with the optimal hyper-parameters we got from cross-validation.

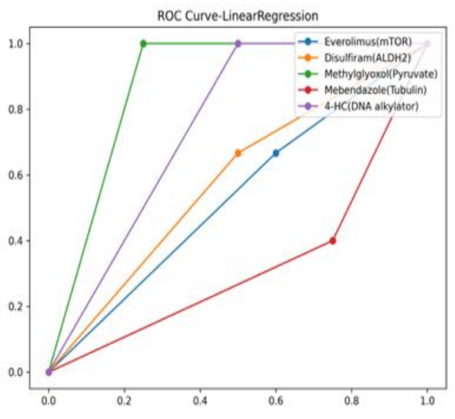
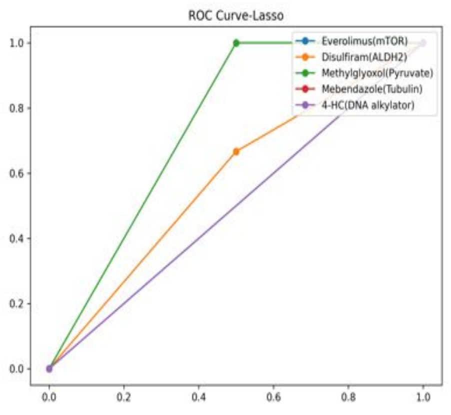
Everolimus	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Hyper-parameter**
SVM	0.76	0.71	0.83	0.77	0.64	{'C': 1.0, 'kernel': 'sigmoid'}
Linear Regression	0.5	0.40	0.67	0.50	0.533	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Logistic Regression	0.5	0.40	0.67	0.50	0.533	{'C': 0.5, 'max_iter': 100, 'penalty': 'l2'}
Lasso Regression	0.375	0.38	1.00	0.55	0.5	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Elastic Net	0.5	0.40	0.57	0.50	0.533	{'alpha': 0.1, 'l1_ratio': 0, 'max_iter': 1000, 'normalize': False}
Ridge Regression	0.5	0.40	0.67	0.50	0.533	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Decision Trees	0.68	0.71	0.63	0.67	0.68	{'max_depth':7 'max_features': 13}
Random Forest	0.76	0.81	0.68	0.74	0.61	{'max_depth':3 'max_features': 16}
KNN	0.76	0.71	0.83	0.77	0.67	{'metric': 'correlation', 'n_neighbors': 8, 'weights': 'distance'}
Neural Network	0.76	0.76	0.72	0.74	0.76	{'alpha': 1, 'batch_size': 32, 'hidden_layer_sizes': (100, 100, 100)}

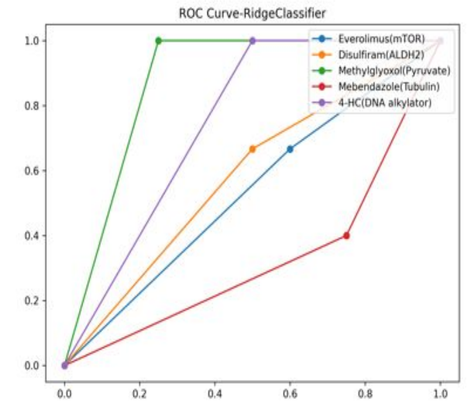
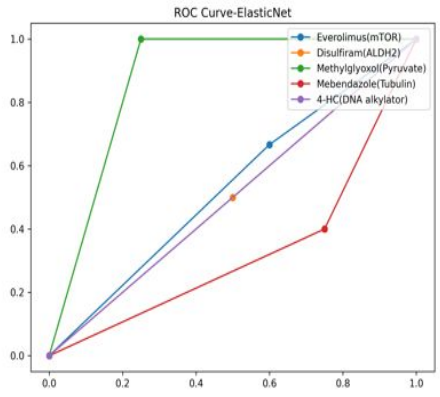
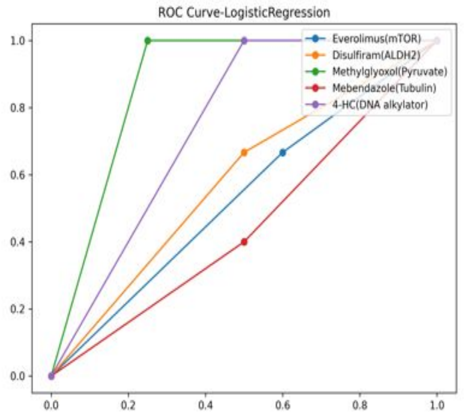
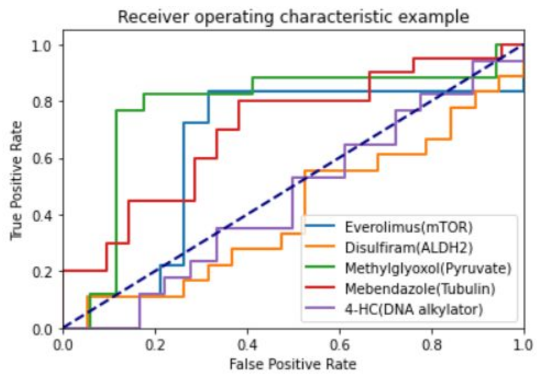
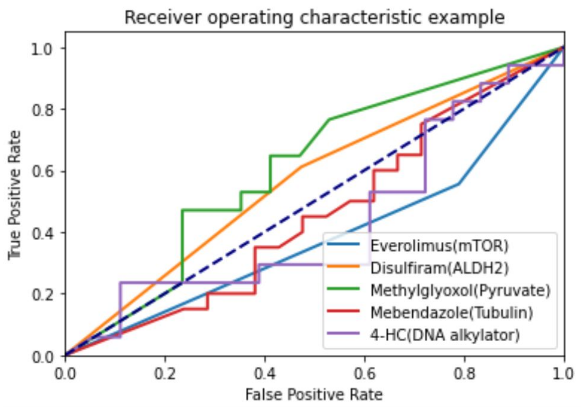
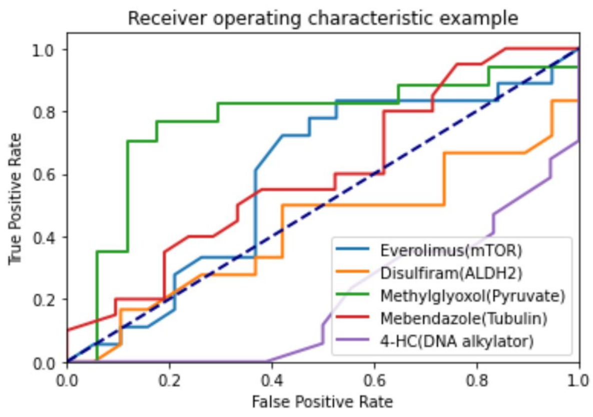
Disulfiram	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Hyper-parameter**
SVM	0.59	0.59	0.56	0.57	0.41	{'C': 0.1, 'kernel': 'linear'}
Linear Regression	0.625	0.80	0.67	0.73	0.583	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Logistic Regression	0.625	0.80	0.67	0.73	0.583	{'C': 1, 'max_iter': 100, 'penalty': 'l2'}
Lasso Regression	0.625	0.80	0.67	0.73	0.583	{'alpha': 0.1, 'max_iter': 100, 'normalize': False}
Elastic Net	0.5	0.75	0.50	0.60	0.5	{'alpha': 0.1, 'l1_ratio': 0.3, 'max_iter': 100, 'normalize': True}
Ridge Regression	0.625	0.80	0.67	0.73	0.583	{'alpha': 0.1, 'max_iter': 100, 'normalize': False}
Decision Trees	0.62	0.63	0.63	0.63	0.61	{'max_depth':2 'max_features': 17}
Random Forest	0.57	0.57	0.63	0.60	0.53	{'max_depth':7 'max_features': 17}
KNN	0.76	0.71	0.83	0.77	0.69	{'metric': 'correlation', 'n_neighbors': 6, 'weights': 'distance'}
Neural Network	0.78	0.81	0.72	0.76	0.77	{'alpha': 0.1, 'batch_size': 8, 'hidden_layer_sizes': (100, 100, 100)}

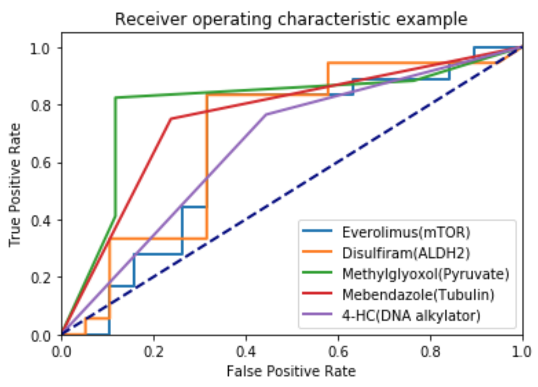
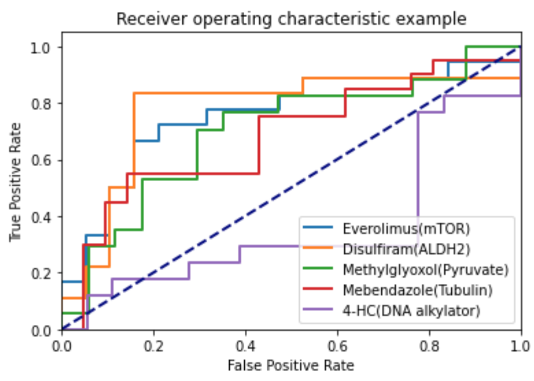
Methylglyoxol	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Hyper-parameter**
SVM	0.82	0.82	0.82	0.82	0.77	{'C': 0.5, 'kernel': 'sigmoid'}
Linear Regression	0.857	0.75	1.00	0.86	0.875	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Logistic Regression	0.857	0.75	1.00	0.86	0.875	{'C': 0.5, 'max_iter': 100, 'penalty': 'l2'}
Lasso Regression	0.714	0.6	1.00	0.75	0.75	{'alpha': 0.1, 'max_iter': 500, 'normalize': False}
Elastic Net	0.857	0.75	1.00	0.86	0.875	{'alpha': 1, 'l1_ratio': 0.3, 'max_iter': 100, 'normalize': False}
Ridge Regression	0.857	0.75	1.00	0.86	0.875	{'alpha': 0.1, 'max_iter': 100, 'normalize': False}
Decision Trees	0.65	0.65	0.65	0.65	0.64	{‘max_depth’:2 ‘max_features’: 17}
Random Forest	0.79	0.81	0.76	0.79	0.75	{‘max_depth’:3 ‘max_features’: 10}
KNN	0.85	0.82	0.88	0.85	0.80	{'metric': 'euclidean', 'n_neighbors': 3, 'weights': 'uniform''}
Neural Network	0.71	0.71	0.71	0.71	0.70	{'alpha': 0.1, 'batch_size': 8, 'hidden_layer_sizes': (100, 100, 100)}

Mebendazole	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Hyper-parameter**
SVM	0.63	0.63	0.60	0.62	0.72	{'C': 1.0, 'kernel': 'linear'}
Linear Regression	0.333	0.40	0.40	0.40	0.325	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Logistic Regression	0.444	0.50	0.40	0.44	0.450	{'C': 0.5, 'max_iter': 100, 'penalty': 'l2'}
Lasso Regression	0.444	0.00	0.00	0.00	0.5	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Elastic Net	0.333	0.40	0.40	0.40	0.325	{'alpha': 1, 'l1_ratio': 0, 'max_iter': 100, 'normalize': False}
Ridge Regression	0.333	0.40	0.40	0.40	0.325	{'alpha': 0.1, 'max_iter': 100, 'normalize': False}
Decision Trees	0.61	0.60	0.71	0.65	0.61	{‘max_depth’:7 ‘max_features’: 17}
Random Forest	0.54	0.54	0.62	0.58	0.57	{‘max_depth’:3 ‘max_features’: 14}
KNN	0.76	0.75	0.75	0.75	0.76	{'metric': 'correlation', 'n_neighbors': 1, 'weights': 'uniform'}
Neural Network	0.59	0.57	0.60	0.59	0.68	{'alpha': 1, 'batch_size': 8, 'hidden_layer_sizes': (100, 100, 100)}

4-HC	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Hyper-parameter**
SVM	0.40	0.39	0.41	0.40	0.46	{'C': 1.0, 'kernel': 'linear'}
Linear Regression	0.714	0.60	1.00	0.75	0.75	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Logistic Regression	0.714	0.60	1.00	0.75	0.75	{'C': 1, 'max_iter': 100, 'penalty': 'l2'}
Lasso Regression	0.429	0.43	1.00	0.60	0.5	{'alpha': 0.1, 'max_iter': 100, 'normalize': True}
Elastic Net	0.429	0.43	1.00	0.60	0.5	{'alpha': 0.1, 'l1_ratio': 1, 'max_iter': 100, 'normalize': True}
Ridge Regression	0.714	0.60	1.00	0.75	0.75	{'alpha': 0.1, 'max_iter': 100, 'normalize': False}
Decision Trees	0.34	0.33	0.28	0.30	0.34	{'max_depth':9 'max_features': 16}
Random Forest	0.37	0.40	0.44	0.42	0.28	{'max_depth':5 'max_features': 17}
KNN	0.66	0.62	0.76	0.68	0.66	{'metric': 'correlation', 'n_neighbors': 1, 'weights': 'uniform'}
Neural Network	0.37	0.33	0.29	0.31	0.36	{'alpha': 1, 'batch_size': 32, 'hidden_layer_sizes': (100, 100, 100)}

Linear Regression	Lasso
	

Ridge	Elastic Net
	
Logistic Regression	SVM
	
Decision Trees	Random Forest
	

KNN	Neural Networks
	

Discussion

Regression models:

- Linear Regression may not be the optimal model as there may be no simple linear relationship so the model suffers from multicollinearity, autocorrelation and heteroskedasticity.
- Logistic Regression can deal with probability of the event. As we change the cut-off, the prediction result will also change.
- Lasso Regression penalizes the absolute size of the regression coefficients and shrinks some of the correlated coefficients to zero (exactly zero), which certainly helps in feature selection.
- Elastic Net is a hybrid of Lasso and Ridge Regression techniques. It is trained with L1 and L2 prior as regularizer. Elastic Net is useful when there are multiple features which are correlated. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.
- The assumption of Ridge regression is the same as least squared regression except that normality is not to be assumed. Ridge regression shrinks the value of coefficients but doesn't reaches zero, which suggests no feature selection feature

SVM

Theoretically, SVM works well in both accuracy and algorithm efficiency with high-dimensional data space, especially in cases where the number of dimensions outnumbered the number of samples. The reason that SVM performance on our dataset is not very ideal for some labels may be because of the attributes outnumbered the sample size to a large extent, and the data contains noises that SVM is not able to handle. Also, we are unclear about the spatial positioning of the sample points, but it is very likely that there are no clear margins for the positive and negative classes concerning each label. All of the above notions are possible to influence the performance of SVM on our dataset.

Decision Trees
Compared to other algorithms implemented in this project, decision trees require less effort for data preparation during pre-processing. A decision tree does not require normalization and scaling of data. Our data set contains a large number of features, most of which are redundant for determining a label for a specified drug. Decision trees have advantages handling these redundant features. However, the results of decision trees tell us that there should be some interacting attributes.

Random Forest
Random forest leverages the power of multiple decision trees, so it indeed performs better than decision trees in our project. However, random forest is a black box model which is hard to know what is happening. In our project, we can at best try some different parameters combinations and random seeds to improve performance.

Neural Networks
Although neural networks can handle redundant and irrelevant attributes, due to its sensitivity to noise in training data, we believe that the high dimensional feature space should contain a lot of noise and thus cause the model to be overfitting and yielding unsatisfactory results, even with regularization. On the other hand, we have found that it was hard for the optimization to find a good local/global minima and converge, which may result from over-simplicity of the network we were using. However, considering the noise, it may not help a lot even if we construct more complex neural networks.

Conclusion
We have finished the drug sensitivity prediction tasks for 5 selected drugs, contributing to the NCI-DREAM challenge as a virtual participant. From the results, we have found that each machine learning technique performs differently on each drug and requires different strategies to improve. Besides, to deal with the high-dimensional feature space, we mainly implemented regularization and dimension reduction to help avoid overfitting and select important features. Particularly, certain drugs could yield poor performance even when the data was standardized or the dimension got reduced. For such certain drugs whose standards are hard to fit with a good model, efforts are needed in either processing the data more intelligently or building a more robust model that performs better.

Contribution
Each member contributed equally to this project. More specifically, Yixuan Wang was responsible for data analysis. Yu Fang, Yu Han, Xianjian Xie, and Xiang Zhang investigated the different machine learning models respectively. This report was written by collaborative efforts.

Job Division	
Yu Fang	SVM
Yu Han	Linear/Logistic/Lasso/Elastic Net/Ridge Regression
Yixuan Wang	Data Analysis, Dimension Reduction
Xianjian Xie	Decision Tree, Random Forest
Xiang Zhang	KNN, NN

Code Availability

The code for all the implementation and analysis in our project can be found at this Github repository: https://github.com/Karashan/CSCI5523_F20_Drug_Sensor

Reference

1. Costello, J. C. et al. A community effort to assess and improve drug sensitivity prediction algorithms. Nat. Biotechnol. 32, 1202–1212 (2014).
2. Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.
3. Shlens, J. (2014). A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100.
4. Cox, M. A., & Cox, T. F. (2008). Multidimensional scaling. In Handbook of data visualization (pp. 315-347). Springer, Berlin, Heidelberg.
5. Cortes, C. & Vapnik, V. (1995). Support Vector Networks. *Machine Learning*, 20, 273-297.
6. Hastie, Tibshirani and Friedman (2008). The Elements of Statistical Learning (2nd edition). Springer-Verlag.
7. Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879
8. Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep Learning". Nature. 521 (7553): 436–444. Bibcode:2015Natur.521..436L. doi:10.1038/nature14539