Brian Guo, Winston Yang

Dec 9, 2021

Intro to Machine Learning

Final Project Write Up

Introduction

For this final project, we are trying to predict the possibility of heart failure given a patient's records. By being able to predict heart failure with machine learning, doctors can achieve early intervention to prevent such deaths. In our dataset, there are 11 features which are Age, Sex, Chest Pain Type, Resting Blood Pressure, Cholesterol, Fasting Blood Sugar, Resting ECG, Maximum Heart Rate, Exercise Angina, Old Peak and ST_Slope. The output class is whether the patient has heart disease. The dataset has 918 observations of patients.

Feature Transformation

Due to some of our features not having numerical values, feature transformation must be used to represent these values. One hot encoding is used for these feature transformations. Sex is represented as two features: Male and Female with a correspoding 0 or 1 in its column, Chest Pain is represented by four features: TA or Typical Angina, ATA or Atypical Angina, NAP or Non-Anginal Pain, and ASY or Asympatomatic Pain, RestingECG is represented as three features: Normal, ST, and LVH or left ventricular hypertrophy, Exercise Angina is represented by two features: Yes and No, ST_Slope is represented by three features: Upward slope, flat slop, and downward slope.

In addition to these transformations, we used the standard scaler from the sklearn preprocessing package on our data.

Supervised Analysis

For our analysis, we chose to use Logistic Regression, SVM, and Neural Networks. For the Logistic Regression, we tried it with both L1 and L2 Regularization and by using a polynomial transformation on the data. For SVM, we tried linear, rbf, and polynomial kernels. For our neural networks, we used 3 different solvers: lbfgs, sgd, and adam. Then, we tried different different activation functions and different values for alpha, which affects the regularization term.

Results

Figure 1 below shows the results of running logistic regression on our data. We tried different C values for each model, and the best performing C value (highest test accuracy) is highlighted in green for each model. The best overall performing logistic regression model was the model with polynomially transformed data in conjunction with L2 Regularization. This model with C = 0.01 yielded the highest test accuracy of 0.881.

Logreg L1	C	Train Accuracy	Test Accuracy
	0.0001	0.533333	0.594059
	0.001	0.533333	0.594059
	0.01	0.821138	0.79868
	0.1	0.856911	0.851485
	1	0.860163	0.854785
	10	0.863415	0.854785
Logreg L2	**C**	**Train Accuracy**	**Test Accuracy**
	0.0001	0.629268	0.669967
	0.001	0.850407	0.874587
	0.01	0.860163	0.864686
	0.1	0.865041	0.861386
	1	0.865041	0.854785
	10	0.863415	0.854785
Logreg polynomial L1	**C**	**Train Accuracy**	**Test Accuracy**
	0.0001	0.533333	0.594059
	0.001	0.533333	0.594059
	0.01	0.821138	0.79868
	0.1	0.878049	0.864686
	1	0.902439	0.858086
	10	0.902439	0.858086
Logreg polynomial L2	**C**	**Train Accuracy**	**Test Accuracy**
	0.0001	0.757724	0.808581
	0.001	0.873171	0.877888
	0.01	0.891057	0.881188
	0.1	0.895935	0.867987
	1	0.900813	0.861386
	10	0.899187	0.858086

Figure 1: Results from Logistic Regression

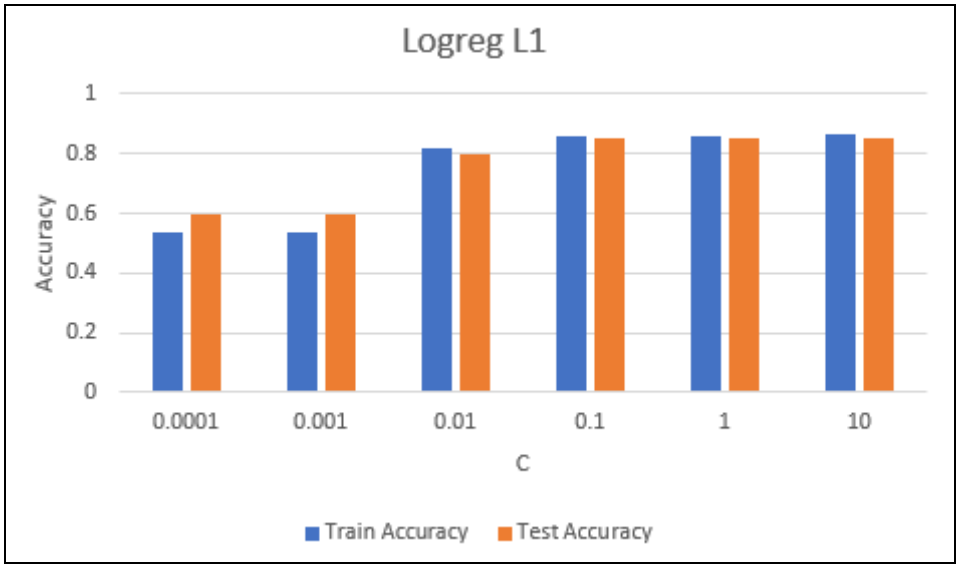

Figure 2: Logistic Regression with L1 Regularization Graph

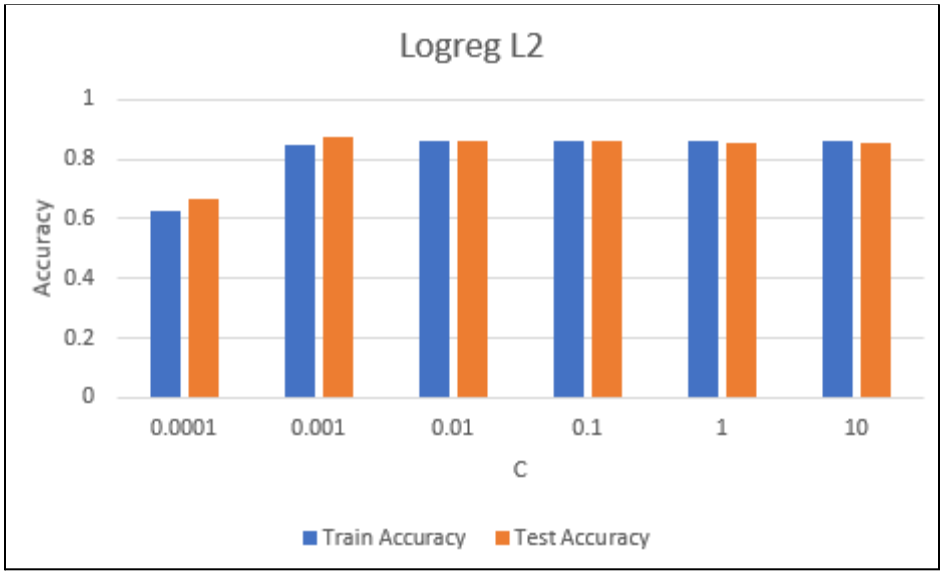

Figure 3: Logistic Regression with L2 Regularization Graph

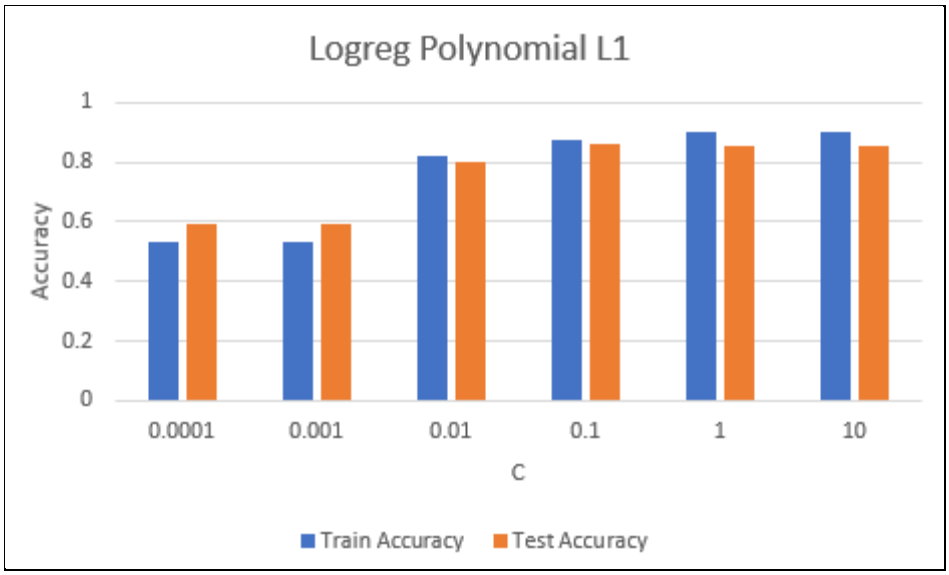

Figure 4: Logistic Regression, polynomial transform, L1 regularization

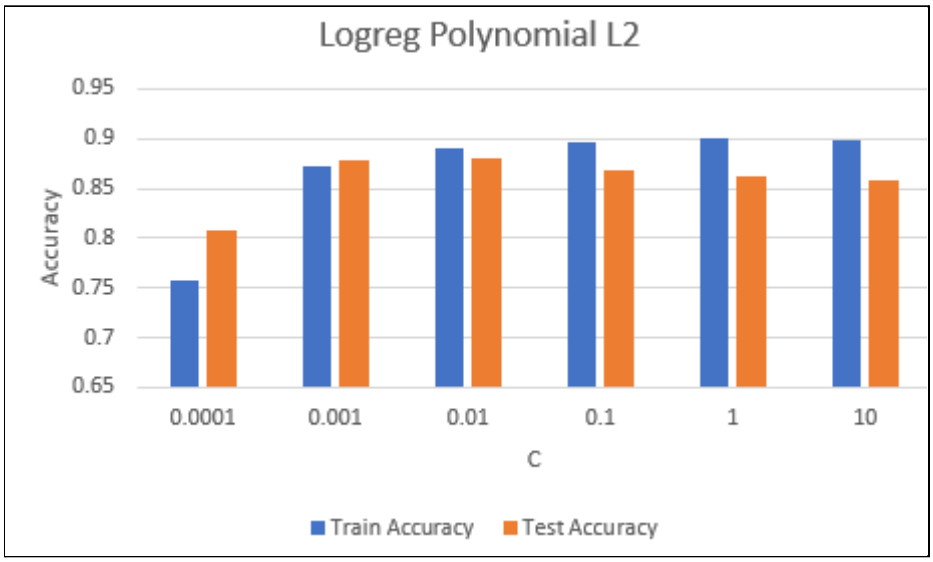

Figure 5: Logistic Regression, polynomial transform, L1 regularization

Figures 2-5 show the results for each model in a graph form. The different C values are plotted along the x-axis while the y-axis shows the training accuracies in blue and the test accuracies in orange.

SVM Linear Results	C	Train Accuracy	Test Accuracy
	0.0001	0.554472	0.551155
	0.001	0.871545	0.854785
	0.01	0.869919	0.851485
	0.1	0.869919	0.848185
	1	0.873171	0.844884
	10	0.873171	0.844884
	100	0.873171	0.844884
	1000	0.873171	0.844884
SVM rbf Results	**C**	**Train Accuracy**	**Test Accuracy**
	0.0001	0.554472	0.551155
	0.001	0.554472	0.551155
	0.01	0.554472	0.551155
	0.1	0.868293	0.851485
	1	0.912195	0.848185
	10	0.962602	0.818482
	100	0.988618	0.80198
	1000	1	0.782178
SVM Polynomial Results	**C**	**Train Accuracy**	**Test Accuracy**
	0.0001	0.554472	0.551155
	0.001	0.554472	0.551155
	0.01	0.55935	0.554455
	0.1	0.869919	0.828383
	1	0.920325	0.848185
	10	0.95935	0.821782
	100	0.986992	0.782178
	1000	1	0.775578

Figure 6: Results from SVM

Figure 6 shows the results of running SVM on our data. We tried different C values for each model, and the best performing C value (highest test accuracy) is highlighted in green for each model. The highest test accuracy was 0.855, achieved by the linear function kernel SVM with C = 0.001, but the rbf and polynomial kernels came very close, so it's hard to definitively pick one over the others.

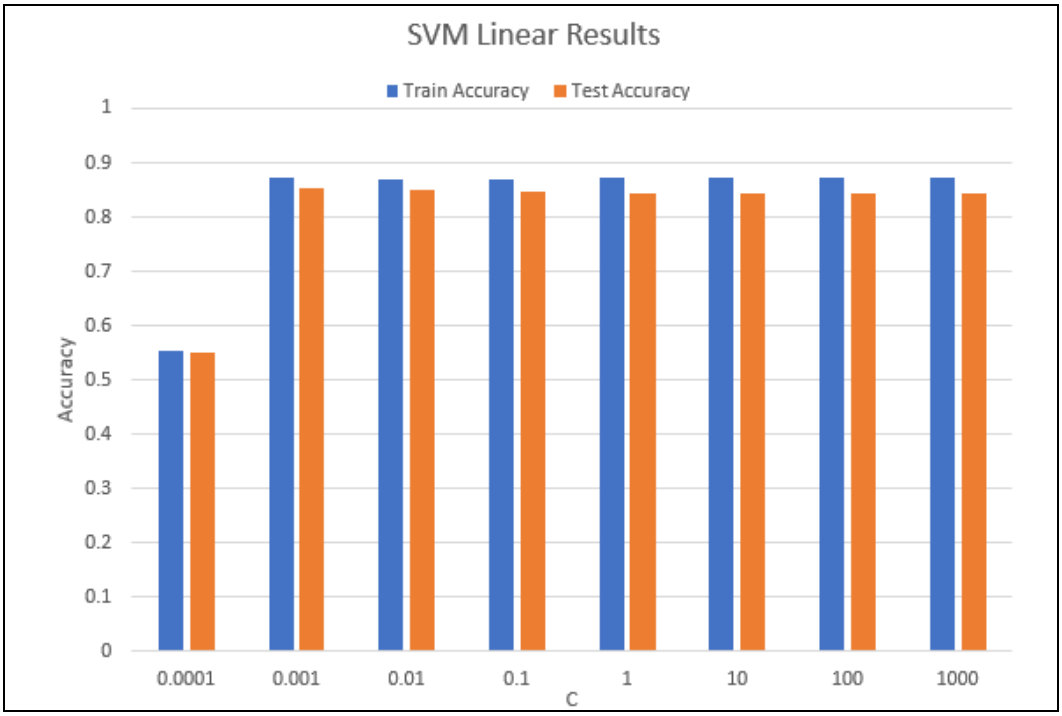

Figure 7: SVM Linear Kernel Results

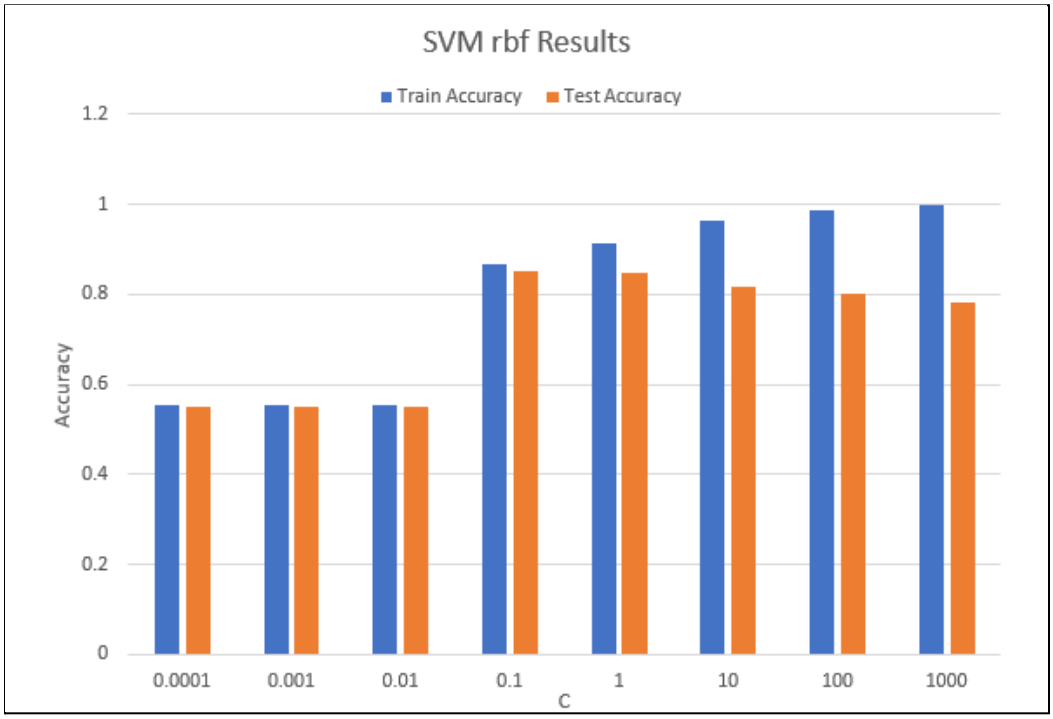

Figure 8: SVM rbf Kernel Results

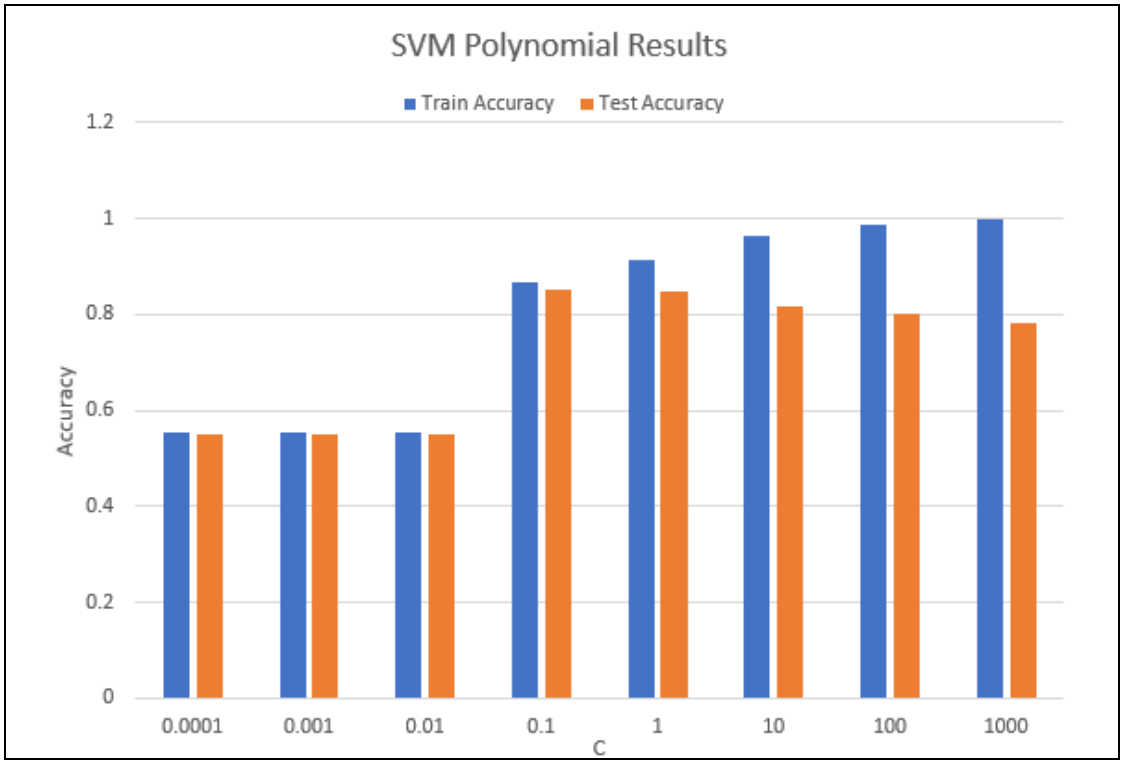

Figure 9: SVM Polynomial Kernel Results

Figures 7-9 show the results of the 3 SVM models in graph forms. Just like the logistic regression graphs, the different C values are plotted along the x-axis while the y-axis shows the training accuracies in blue and the test accuracies in orange.

Solver Used	Train Accuracy	Test Accuracy
lbfgs	0.837615176	0.813421342
sgd	0.689376694	0.688338834
adam	0.856747967	0.847194719
adam + logistic activation function	0.862601626	0.860066007
adam + logistic activation function + different alpha	0.863631436	0.856875688

Figure 10: Results from Neural Network

Figure 10 shows the results of running neural networks on our data. The highest test accuracy was 0.86, achieved by using the adam solver in combination with a logistic activation function. However, both the adam solver on its own and the adam solver with the logistic activation function and a smaller alpha value came very close. So, it could just be the inherent

randomness which pushed this particular model over the others. Overall, all of the models we tried using the adam solver performed very close to each other with a high test accuracy.

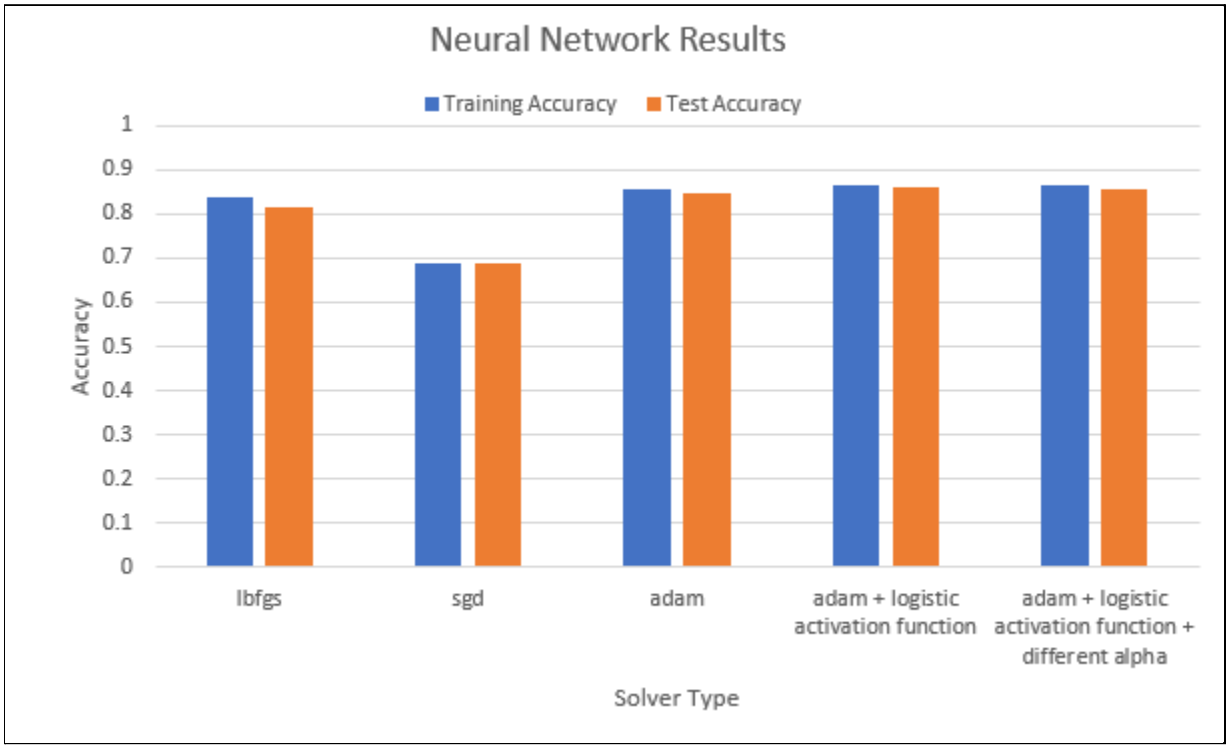

Figure 11: Neural Network Results, graphed

Figure 11 shows the results of running neural networks on our data. As with the previous graphs, the training accuracy is shown in blue and the test accuracy in orange. From the graph, it's clear that all 3 of the models using the adam solver came very close in terms of performance, and the lbfgs solver was just a bit behind.

Analysis

Out of all of the models we tested, the logistic regression model with polynomially transformed data and L2 regularization with C = 0.01 yielded the highest test accuracy of 0.881. However, many of the other models came close, with many yielding test accuracies around 0.85.

When modifying the C values in logistic regression, our training accuracy increases when the C value increases meaning we are increasing the variance while decreasing our bias. This will cause our model to start overfitting the noise; therefore, usually a C value near the median is the best model for logistic regression. Although our training accuracy decreases after our optimal C value, it does not decrease drastically showing that the model is not overfitting much.

The data yielded from running SVM suggests that there may have been overfitting in our tests. As you can see in Figures 8 & 9, the higher C values yielded training accuracies which were much higher than the test accuracies. We hypothesize that the one hot encoding we used may have been a major factor. Our one hot encoding resulted in 9 extra features. This expressiveness can be useful to have but can also lead to overfitting. To prevent this, we could try to add more regularization to our models in the future. We also thought about using ordinal encoding to reduce the number of features, but it didn't make sense to do so with most of the categorical data we had.

Variance was definitely visible in our results as well. For all of our models, the accuracies yielded were slightly different for each run. While not a huge difference each time, the results were different. This was especially apparent in the neural network runs, probably due to the random starting weights of the model each time. To reduce the overall variance, we could try doing multiple runs of each model and then taking an average of the accuracies of the different runs. Furthermore, our relatively small dataset (918 examples) further contributed to the variance. If we found more data to use and run our tests on, then the variance would be less of an issue.