CS-UY 4563: Machine Learning

Final Project Written Report

Credit Card Fraud Detection

11:00 AM Section

7 December 2022

Professor Linda N. Sellie

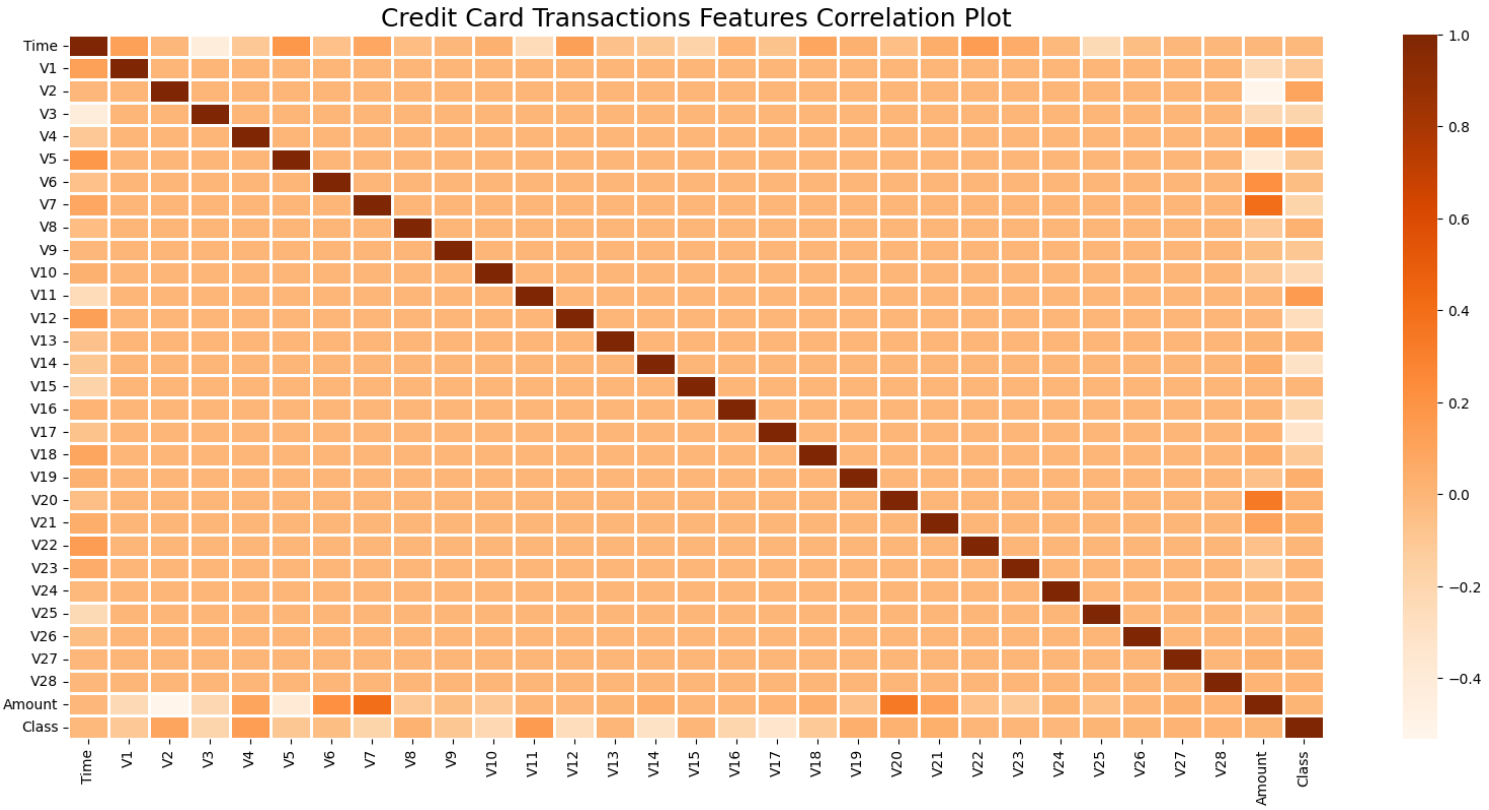
Sergey Hovhannisyan, Arnav Kanwal

Introduction

This project focuses on identifying fraudulent transactions using a dataset from Kaggle. The dataset includes credit card transactions performed by European cardholders in September 2013. We will preprocess the dataset to some degree as it is highly unbalanced; then, we will use Logistic Regression, Support Vector Machines, and Artificial Neural Networks to train on the preprocessed data tuning using hyperparameters to find an optimal model that would generalize fairly enough.

Preprocessing

We have 492 frauds out of 284,807 transactions in our dataset of transactions that took place over the course of two days. The dataset is quite skewed, with frauds making up 0.172% of all transactions in the positive class. It only has numeric input variables that have undergone PCA transformation. Unfortunately, the dataset does not offer the original characteristics and further context for the data due to confidentiality concerns; the dataset is “masked” applying PCA on the original dataset. The major components derived from PCA are features V1, V2,..., V28. The only features that have not been changed with PCA are "Time" and "Amount." The seconds that passed between each transaction and the dataset's initial transaction are listed in the feature "Time." After observing the data, we preprocessed the data by plotting the Correlation Matrix and reducing the number of features from 32 to 22 including the “Class” feature (Figure 1). Also, we randomly chose only 20,000 samples to reduce the computation time. The feature selection was implemented using the Chi-Square test. As the number of fraudulent transactions is extremely low, we practiced different techniques to balance the dataset such as (Undersampling, NearestMiss, etc.) and concluded that oversampling would be most appropriate. After splitting



the dataset into training, validation, and testing sets, we applied the "RandomOverSampling" transformation using the "imblearn" library on the training set and tested our models with the validation set.

Logistic Regression

The first model that we chose to experiment with was a logistic regression model to see if it could separate the data well, as detecting fraudulent credit card transactions is a classification problem. One of the key issues we addressed in the preprocessing stage was that our data was highly unbalanced with a relatively low number of samples that showed fraud, so we used an oversampling technique to help fix this discrepancy, but this increase in data can be highly expensive for testing feature transformations. So without any regularization, we tested a linear transformation and two polynomial transformations and achieved the following:

Dataset	Score with Polynomial Transformation Degree		
	1 (no transformation)	**2**	**3**
Train	0.981683	0.918086	0.5
Validation	0.990566	0.990566	0.9735

Table 1: Polynomial Transformation Results

While a transformation of degree 3 has a high test score and low train score, it is a clear indicator of overfitting. Based on the data, any transformation may cause some sort of overfitting because we already have enough features to test. Instead, we'll try different values of K in K-fold cross-validation in order to find the best choice of the hyperparameter C and also see if performance is better with different types of regularization.

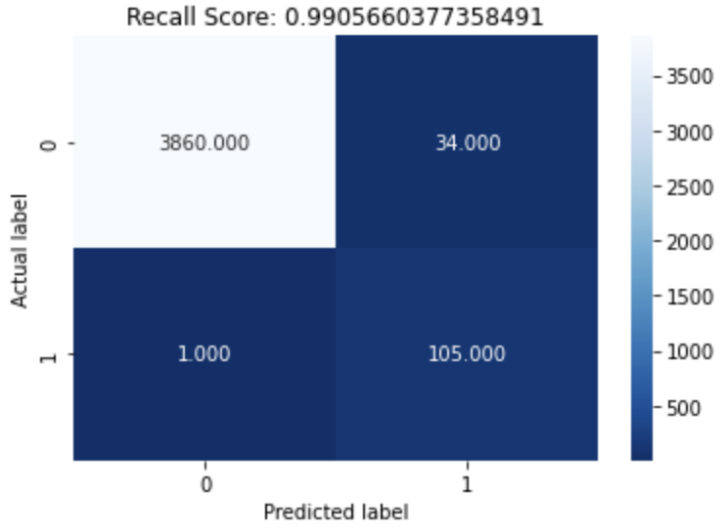
L1 Regularization		
K	**Best C**	**Recall**
3	0.1	0.98113
4	0.1	0.9816
5	0.1	0.97923
6	0.1	0.98113

Table 2: Logistic Regression Results (K-Fold and L1 Regularization)

L2 Regularization		
K	**Best C**	**Recall**
3	1000	0.990566
4	0.1	0.969003
5	0.1	0.98911
6	10	0.981132

Table 3: Logistic Regression Results (K-Fold and L2 Regularization)

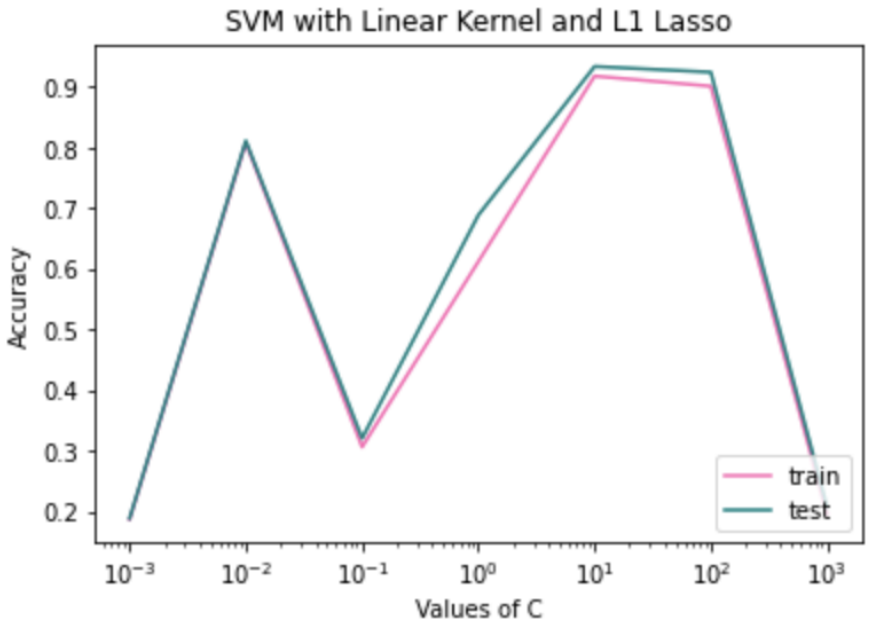
From the above results, we find that the best choice for logistic regression is with C = 1000 and adding L2 regularization to our model to prevent overfitting. By fitting the model and predicting the test set, we can produce the following confusion matrix:

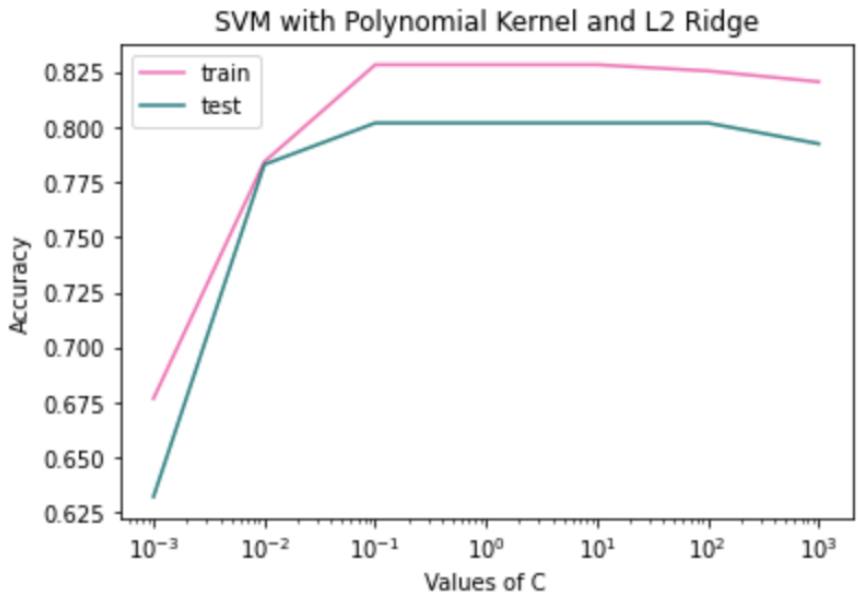


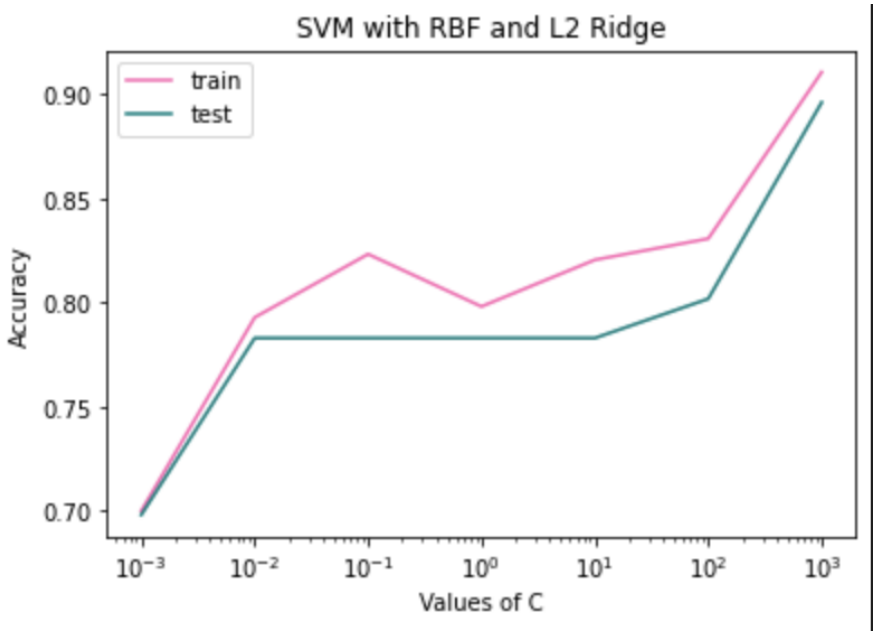
From this matrix, we can compute statistics such as precision of 75.5%, recall and accuracy of 99.1%, and an F1-score of 85.7%. As mentioned in the preprocessing stage, we want to look at recall instead of accuracy due to our unbalanced data and ensure that our model can detect fraudulent transactions. While our precision is low, it is better to err on the side of caution and assume that some transactions may be fraudulent when they are not. We can also assume that we did not need any feature transformations because our data was already linearly separable and we don't want to overfit. With higher values of K we also expect there to be a higher bias in our data, thus with a value of K = 3, we get our best test recall score of 0.99056.

Support Vector Machine (SVM)

Following logistic regression, the next model we chose to experiment with was the SVM. From our previous results, we know our data already has a high amount of features and that feature transformation are extremely costly. With this in mind, we chose to test a linear kernel with L1 regularization, a polynomial kernel of degree 2 with L2 regularization, and a Gaussian RBF kernel with L2 regularization. We then choose to test each of these versions of SVM with the following hyperparameter choices: C = [0.001, 0.01, 0.1, 1, 10, 100, 1000]. Each graph below depicts the output of running these models and shows the difference in recall between the train and validation sets.







Combining the results from all the graphs, the best-performing SVM using a linear kernel with L1 regularization (lasso regression). We can see that a polynomial kernel performed the worst with the highest recall score being around 83%, and using RBF, was able to achieve a recall score of 89% with the test data. Examining all three graphs, we can see that the training recall scores seem to outperform the test recall scores when using RBF and polynomial

transformations. A primary cause for this could again be overfitting as the data does not require complex transformations and doing so will result in high variance and low bias. In addition, we already normalized our data in the preprocessing stage which would indicate another reason for making the RBF kernel (which is normally the go-to standard for SVM) less effective than using a linear SVM. Thus, our final result is a hyperparameter choice of C = 10 with a train recall of 0.918 and a test recall of 0.934.

Neural Network

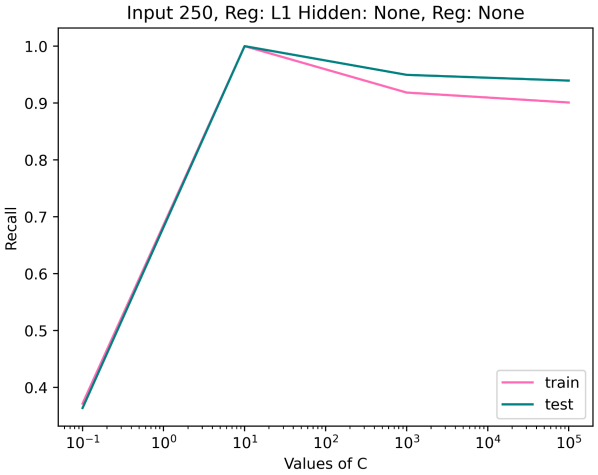
For Artificial Neural Network architecture, we are going to find the optimal neural network structure and parameters by observing the Recall score while changing the number of layers, the number of neurons on each layer, the regularization type for each layer, and tunning the C hyperparameter. As our dataset is imbalanced and has too many samples, which causes extremely high computation time, we experimented with separate models and picked the ones with the closest success. So, we are going to try 6 different models for which we will try 4 different hyperparameters and train having epochs = 2. The input layer will have a variation of 250 and 200 neurons that will have Lasso and Ridge regularizations accordingly. We will practice 2 hidden layers with neurons size of 100 and 150 and will apply Lasso and Ridge regularization accordingly to each hidden layer. We will also train our model with 2 layer structure by only using the 2 input layers mentioned above. The models will be trained with the following hyperparameters:

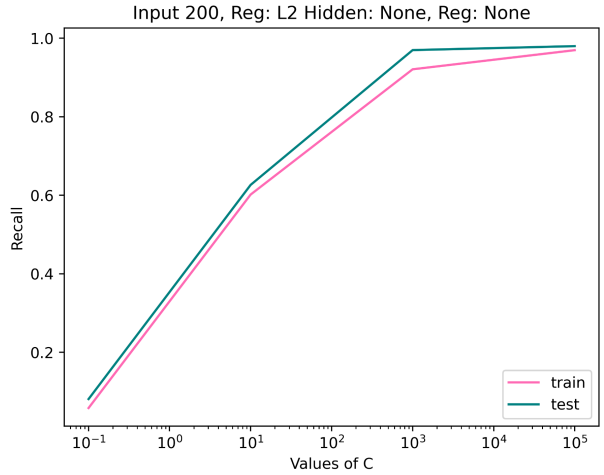
C = [0.1, 10, 1000, 100000]. The recall score is recorded for all models and hyperparameters in Figure 2.

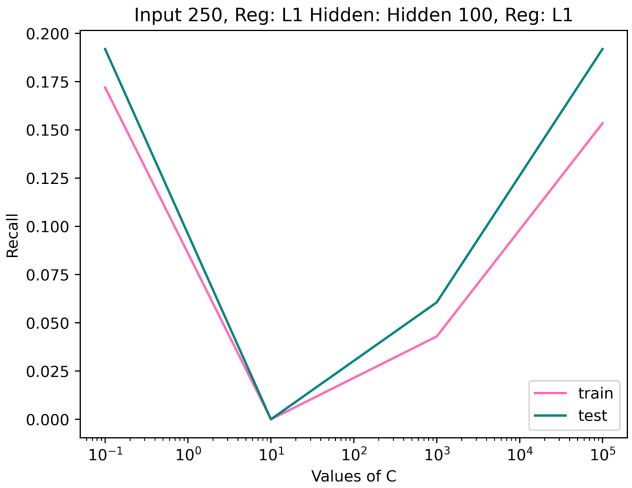
Model S_1,S_2 Reg_1,Reg_2	C			
	0.1	10	1000	100000
200, None L2 None	0.08	0.63	0.97	0.98
200, 100 L2 L1	1.00	0.00	1.00	0.90
200, 150 L2 L2	0.19	0.99	0.11	0.94
250, None L1 None	0.36	1.00	0.95	0.94
250, 100 L1 L1	0.19	0.00	0.06	0.19
250, 150 L1 L2	1.00	1.00	1.00	0.02

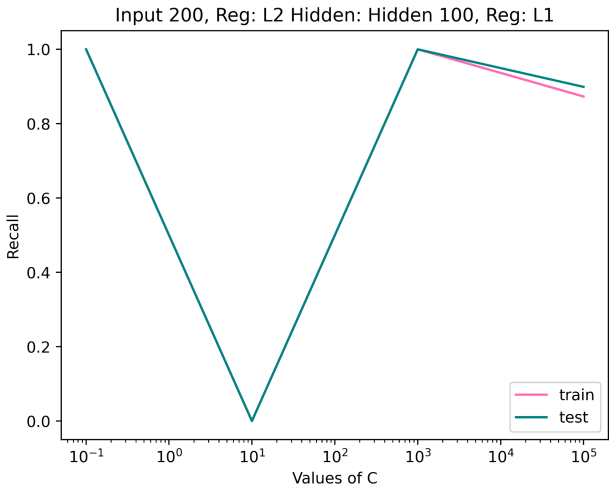
Figure 2

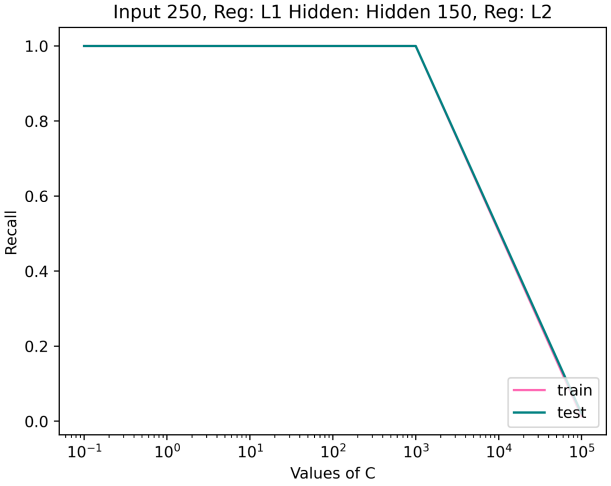
As the dataset is highly imbalanced and the test set contains only about 100 fraudulent transactions, we sometimes got overfitting (marked red); however, the model with "steady" results was 2 layer architecture with an input layer having 200 neurons and using Ridge Regularization where C is 100,000. This was our pick model with neural networks. After observing the models, we came to the conclusion that adding hidden layers only made the models less accurate and less predictable. Also, we noticed better accuracy as the C value increased. The Recall vs C diagrams are the following:

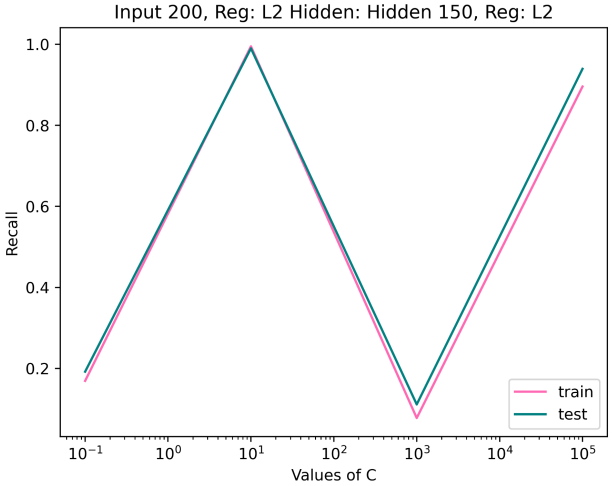

Input 250, Reg: L1 Hidden: None, Reg: None
Recall
Values of C
train
test


Input 200, Reg: L2 Hidden: None, Reg: None
Recall
Values of C
train
test


Input 250, Reg: L1 Hidden: Hidden 100, Reg: L1
Recall
Values of C
train
test


Input 200, Reg: L2 Hidden: Hidden 100, Reg: L1
Recall
Values of C
train
test


Input 250, Reg: L1 Hidden: Hidden 150, Reg: L2
Recall
Values of C
train
test


Input 200, Reg: L2 Hidden: Hidden 150, Reg: L2
Recall
Values of C
train
test

Moreover, we tried sigmoid activation function for both input and hidden layers; however, due to incredibly poor performance, we excluded them from model training to make it short so we could run on GitHub Codespace. Therefore, we used ReLU activation function to ensure better recall and accuracy scores and faster computation.

Conclusion

The best results obtained from each of the models: logistic, support vector machine, and neural network models are displayed in table X. The best-performing model was logistic regression using L2 regularization and no additional feature transformations.

	Logistic Regression, linear transform, L2 regularization	**SVM, linear kernel, L1 Regularization**	**Neural Network, 200, None L2, None**
Recall	**0.991**	**0.934**	**0.98**

From what we found above, we can see that our data is very linearly separable and proves to be a simple classification problem with no difficult boundaries to work around. For logistic regression, we first observed the results of different feature transformations without any sort of regularization or penalty and already saw how expensive these transformations were on both running our algorithms on the sheer number of samples that our data contained as well as in terms of precision, recall, and accuracy scores of the data. Then, we added regularization into the model with K-fold cross-validation, which allowed us to find a hyperparameter choice of C = 1000 and a recall score of 0.99056, which was the highest out of all other models.

Regarding SVM, we tried various transformations with L1 and L2 regularization and avoided testing without penalty altogether. Due to our data already having had PCA applied and

then oversampling the number of fraudulent transactions, cutting features, and scaling our data, transformations like RBF also did not perform as well as a linear SVM, again indicating that our data is linearly separable and should not be overcomplicated. With these different variations, the highest recall we were able to achieve was with C = 10 and a score of 0.934.

When we tried neural network models to find nonlinear decision boundaries with about 200 input layers, the model performed well, however, adding new layers or increasing the number of neurons only worsened the performance in the sense of recall score. Hence, we can conclude that the "optimal" ANN model for the preprocessed dataset was 200 input neurons with L2 (Ridge) regularization where the C value was 100,000. Also, as the dataset was highly imbalanced, both overfitting and underfitting were unavoidable due to creating new data with random oversampling and trying it on only 100 samples in the validation set.

Across all three models, a common pattern that occurred was that complex feature transformations resulted in lower test recalls and higher train recalls, which is a classic indicator of overfitting. This kind of issue means introducing low bias and high variance, which is not good practice, especially when trying to solve a classification problem like detecting fraudulent credit card transactions. By using data with unsupervised analysis (PCA) and cutting features with low contribution, the model's weights were able to explain much of the variance within the data. Although logistic regression was able to perform the best, a natural extension of this project would be to collect further data on credit card transactions until we have developed a generous amount of non-simulated transactions that were fraudulent. By using a more balanced dataset, we may be able to produce better results with SVM and Neural Networks or find a more complicated boundary that isn't necessarily linearly separable like the data that we worked with in this project.

Works Cited

"Credit Card Fraud Detection," *Machine Learning Group - ULB*, Kaggle, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Fawcett, Tom, "Learning from Imbalanced Classes," *Silicon Valley Data Science,* 16 Nov. 2017, https://www.svds.com/tbt-learning-imbalanced-classes/

Galarnyk, Michael, "Logistic Regression using Python (scikit-learn)," *Towards Data Science,* 13 Sep 2017, https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a