Credit Card Fraud Modeling

Brandon Miner and Konnor Payne

Introduction

The focus of this paper was to create a regression model to accurately predict whether a transaction is fraudulent from parameters including time, location, etc. The models were built on the Credit Card Fraud Detection 2023 dataset obtained from Kaggle.com by Nidula Elgiriyewithana (2), which contains information on European credit cardholders. The dataset consists of 31 parameters in which 28 were anonymized to protect the credit card holders’ information, however the 3 parameters that were made public are as follows; *ID:* a unique identifier for each transaction, *Amount:* the dollar amount of each transaction, and *Class:* a binary label for whether the transaction was fraudulent or not. The remainder of the anonymized parameters were given arbitrary labels of *V1* through *V28* which represent various attributes of the transactions such as the aforementioned time, location, etc. In total there were over 550,000 records contained in this dataset contributing to nearly 20 million data points.

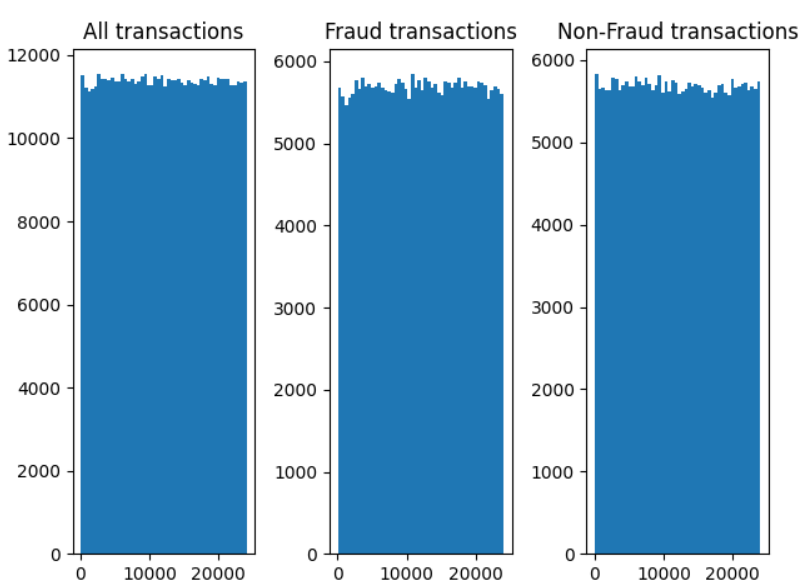
Since we are both statisticians we found this dataset of particular interest as we clearly saw the opportunity to build complex models on the given parameters to predict fraudulent transactions. We must build a labeling model to challenge this classification problem.

The first order of operations was to clean and learn more about the dataset to find any relevant trends or features to assist in the model selection. First a histogram of *Amount* was created to observe any distinguishing features. Then a dimensionality reduction technique was needed, in this case a Principal Component Analysis (PCA) was conducted and using the results from this analysis the insignificant parameters were dropped. After analyzing the data, the information was utilized to construct a series of models with a particular interest in accuracy, sparsity, and false negative rate.

Data Analysis

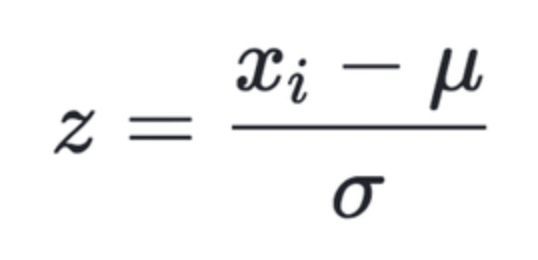
Before beginning any analysis the data needed to be cleaned. By reading the data using a pandas dataframe, the data was transferred to code seamlessly. The parameter, *ID*, was the first parameter dropped from the data frame because it was only an index for data points and therefore it was insignificant for modeling or to include in the data analysis.

The first approach of analysis was to evaluate the parameter *Amount,* as this was the only known parameter in the dataset when differentiating by fraudulence. A histogram of fraudulent, non-fraudulent, and all transactions (Figure 1) was created to examine the underlying trends in this stratification. At first look there were no apparent trends or signals in the histogram so adjustments were made to the range to get a closer look at the maximum values and determine if any features could be derived upon closer inspection. However, despite the alterations to the scale no discernable features could be deduced in the histograms across all transactions, fraudulent and non-fraudulent. More analysis was required to confirm, but this was the first instance where *Amount* showed signs of being insignificant for classifying a fraudulent transaction.



**Figure 1.** Histograms of *Amount* of each transaction, separated by all transactions, fraudulent, and non-fraudulent transactions.

A dimensionality reduction technique was required to further the analysis of the dataset and calculate the optimal number of parameters for the model. This would also help reduce the computation complexity and prevent crashes. A principal component analysis (PCA) using the singular value decomposition (SVD) method was chosen for this step. The SVD method was selected over the eigenvalue decomposition method as the latter requires a square matrix to perform in which this dataset is not formatted in that way (9). In order to perform the PCA the dataset was standardized using Equation 1 where *z* is the standardized value, is the individual value, μ is the mean, and σ is the standard deviation.

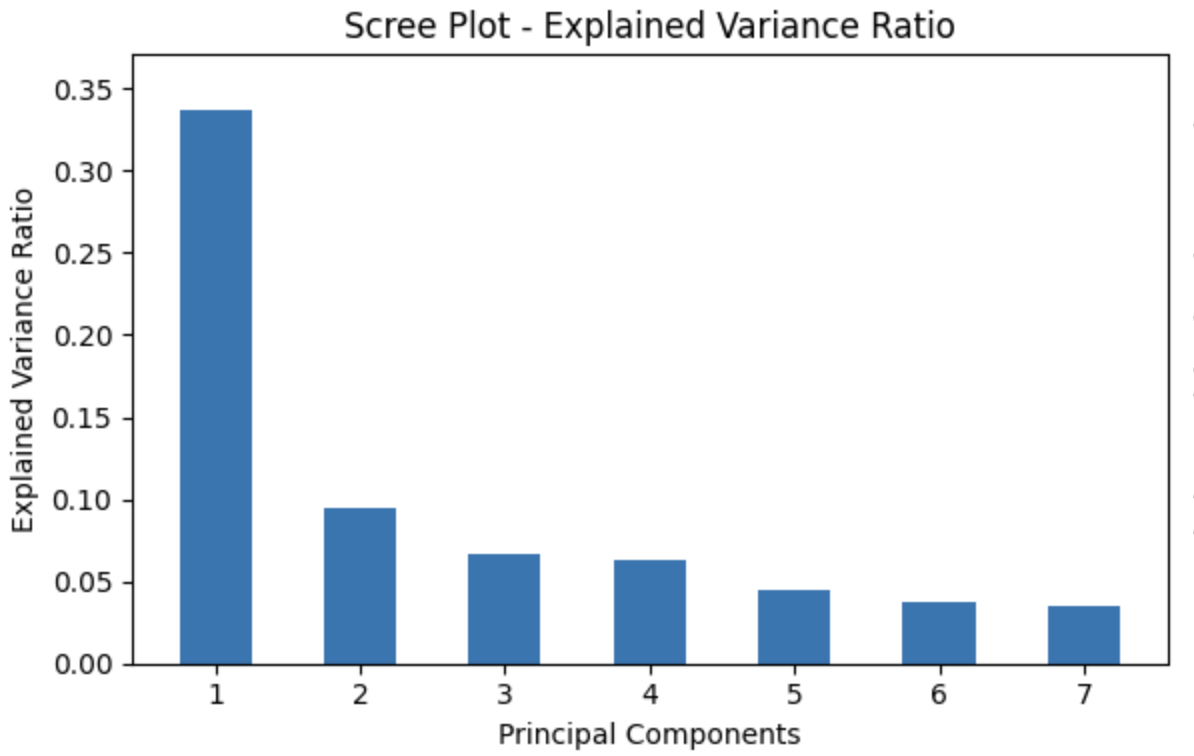
(**Equation 1**)

Equation 1 was used to ensure that parameters composed of large ranges did not dominate over the parameters of smaller ranges in the dataset (10). Then with the new standardized dataset the SVD was performed as described in Equation 2 where *A* is the transformed matrix, *U* is a *m* by *m* matrix of the orthonormal eigenvectors of *A*, ∑ is a *n* by *n* diagonal matrix of the singular values which are the square roots of the eigenvalues of *A*, and is the transpose of a *n* by *n* matrix containing the orthonormal vectors of *A.*

(**Equation 2)**

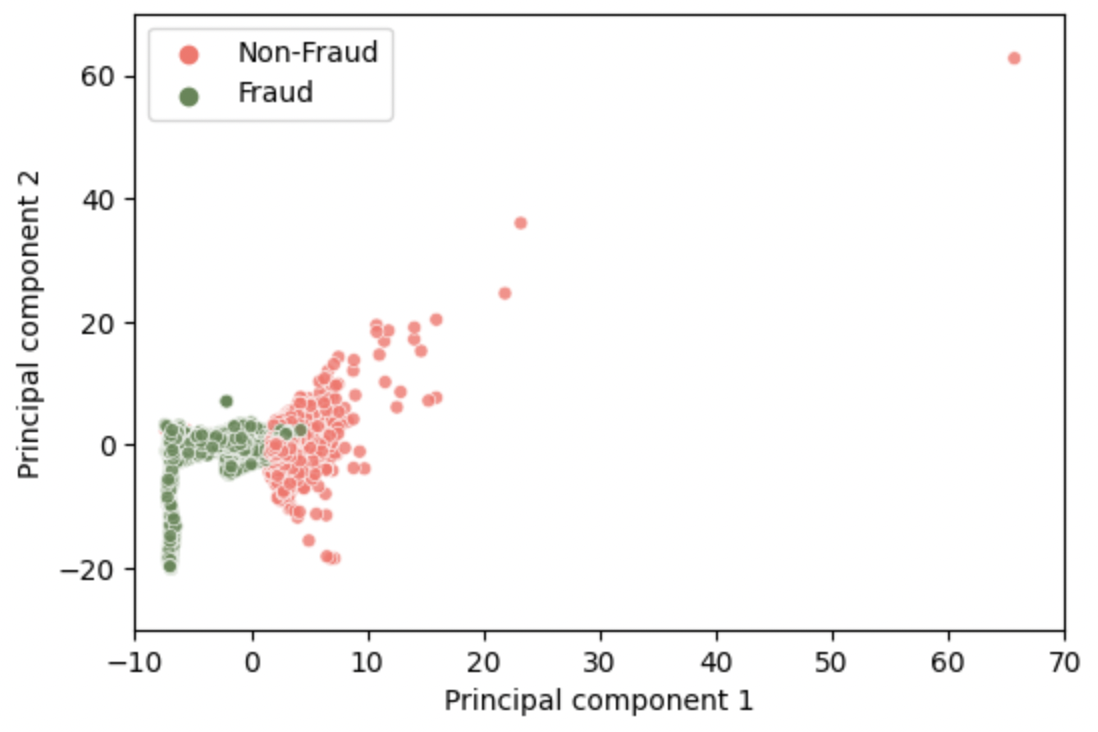
By using Equation (2) the transformed matrix contains the directions with the highest variances respectively (2). By looking at the parts of the SVD we can analyze the eigenvectors that correspond to the largest respective eigenvalue with the same dimensionality of the new feature space (3). Significant values were considered to be greater than 1 and those with lower values did not contribute enough variance to the rest of the dataset to be considered in the analysis. With the remaining eigenvalues they were sorted into descending order and chose the eigenvectors that corresponded to their respective largest eigenvalues in the new feature space (9). The new projection matrix is composed of the eigenvectors with the largest eigenvalues, which are linear combinations of the 29 parameters called principal components (PC) obtained from *U*.

To determine which principal components most significantly affect the dataset the estimated variance of each principal component was needed. Estimated variance was calculated by dividing the eigenvalues of each component by the sum of the eigenvalues (14). The estimated variance for the top 7 principal components were plotted on a scree plot (Figure 2). In total there were 29 principal components, but as shown in Figure 1 the dropoff for explained variance quickly degrades past PC1, so only a portion was shown. For further analysis only PC1 and PC2 were selected in which they comprised 33.69% and 9.50% of the total variance respectively. Despite only explaining 43.19% of the total variance the other PCs do not significantly contribute to the total variance nor do the parameters significantly correlate to them, so they were exempt from further analysis.



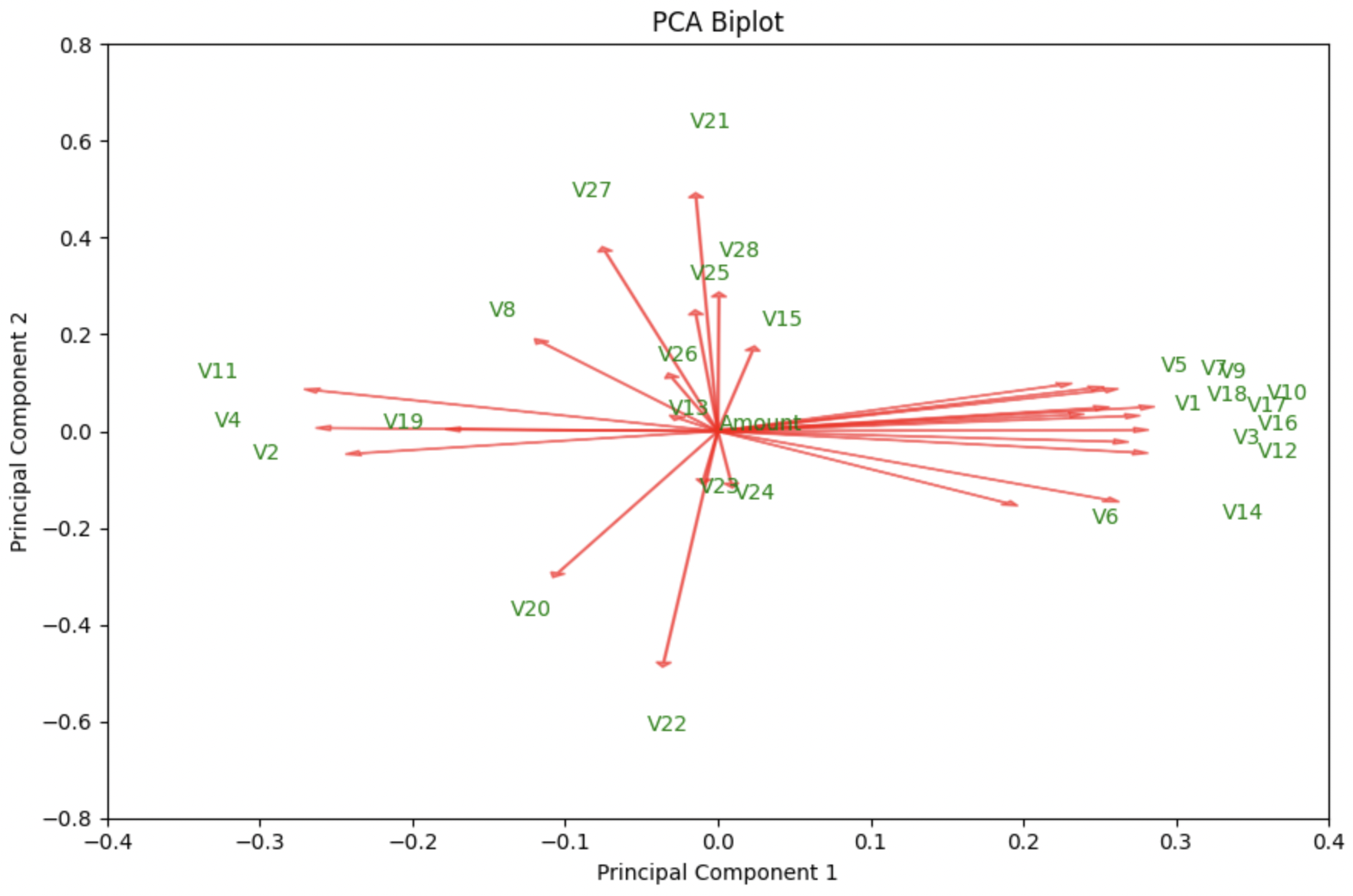
**Figure 2**. Scree plot of the eigenvalues with their respective PC values.

A scatter plot of the transactions’ principal component scores were plotted on PC1 and PC2 that were differentiated by fraudulent and non-fraudulent transactions (Figure 3). As seen in Figure 3 there was a clear distinction between the two groups near the vertical axis on 0 along PC1 that nearly separates all of the fraudulent and non-fraudulent transactions. This implied that the parameters that have a high correlation (positive or negative) with PC1 were the most important to predicting whether a transaction was fraudulent or not.



**Figure 3.** Scatter plot of the transactions’ PC scores projected onto PC1 and PC2 that are differentiated by fraudulent and non-fraudulent transactions.

Using PC1 and PC2 a PCA Biplot (Figure 4) was constructed with all the parameters included to determine the type of correlations between the principal components and the parameters for model building. The arrows directly represent the correlation to both PCs in which direction and magnitude indicate the type of relationship and the strength of it. The larger the arrow the stronger the correlation and the angle of the arrow determines how much of the correlation is negative or positive and how it is divided between the two principal components. Arrows that were parallel to PC1 have the strongest correlation to this principal component, thus inversely the arrows that were perpendicular to PC1 have little to no correlation with it. Considering the trends observed earlier from Figure 3 the arrows with the highest correlation to PC1, whether negative or positive, were selected for model building. The parameters with a positive correlation included *V1, V3, V5, V6, V7, V9, V10, V12, V14, V16, V17, V18* and the parameters with a negative correlation included *V2, V4, V11, V19*. Other parameters such as *V8* or *V20* did not reach suitable magnitudes or directions to be considered for further use. With the 12 positive parameters and the 4 negative parameters the model was constructed with 16 of the original 29 parameters, resulting in the use of 55% of the original dataset. Of all the parameters in Figure 3 the one with the least interaction with PC1 and PC2 was *Amount*, which was located at the origin, confirming the conclusions made from Figure 1 that *Amount* was insignificant for predicting fraudulent transactions.



**Figure 4.** Biplot of the PC scores of each variable in the dataset projected onto PC1 and PC2.

Modeling

It is important to acknowledge that in theory, including all parameters *V1* through *V28* would improve the model’s prediction accuracy, but the increase would have been negligible as well as dramatically increase the computational load. In fact, minimizing the computational load is one of the goals for the model. One of the most insightful ways this was achieved was to make sure the model had sparsity.

While model accuracy is integral to a useful and meaningful model, having a model run as quickly, efficiently, and as simply as possible is also crucial to the model’s success. For a model to meet these criteria, it must have sparsity. A sparse model is one that does not include unimpactful parameters in the prediction calculation. What defines a parameter as “unimpactful” varies for each new dataset that is being modeled. For the selected credit card dataset, we want no smaller than the third decimal place. For most models, the original dataset it was trained/tested on will contain parameters that do not have a considerable effect on the accuracy of predictions, such as *Amount*. Adding sparsity to a model means deleting these parameters from the computation. This improves both runtime and memory load of the model. Simplicity also leads to easier understanding of the model and parameters for people to learn about the dataset that was modeled.

All models constructed in this project utilized a 80% training, 20% testing split of the data as per following general training/testing split convention. All models were created using linear regression as its basis and thus the formula for all models constructed are in linear form as seen below (Equation 3) where is a matrix which contains the known parameters which are used to make a prediction using the model, is the predicted variable, and is the vector with the coefficients of the input parameters. In our model, p is 16 with being the *V’s* we kept after PCA; y is 0 or 1 for the fraud classifier. We transformed the model’s output to fit said binary classifiers by transforming the outputs above or equal to 0.5 to 1 and outputs less than 0.5 to 0. Mathematically, the output was restricted to the interval [0,1].

(**Equation 3)**

To understand how the models were constructed, a breakdown of linear regression is required (7). Also known as least-squares regression, linear regression is based on minimizing the formula below (Equation 4). in this case represents the training parameters which were used to train the model for prediction, and represents the target training data we fit the model to. The -Norm component is essentially the Sum of Squared Errors, and where least-squares regression gets its name. By minimizing this equation, we get a line of best fit which is our model. Note that is not a matrix to each row of the dataset, but instead is only a vector. The error is minimized through that change in across the entire training set.

(**Equation 4**)

We can improve the time complexity by utilizing ridge regression. Ridge regression adds a component to the original linear regression minimized formula as seen below (Equation 5) (11). The second component is known as the regularization term where is a constant which we choose. Note that when , the formula simplifies to linear regression. When tested with the credit card fraud dataset, ridge regression models performed equally as well as linear regression in terms of accuracy and true/false positive/negative rates. The key difference between the two sets of models is that ridge regression performs roughly twice as fast, and thus has a smaller time complexity.

(**Equation 5**)

The LASSO Regression adds another component to the minimized ridge regression formula as seen below (Equation 6) (6). The variable has a secondary use case besides that of regularization. is still a constant which we manually change, but for LASSO regression, adjusting affects the sparsity of the model. The component which specifically adds sparsity to the model is the ending component, the -norm. The -norm promotes sparsity by creating zeros which in-turn change various parameters into zeros. is a constant which we choose. Similar to ridge regression, is simplified to linear regression. As we increase , the accuracy goes down, but model sparsity increases. Finding the balance between the two is important for the model to maintain accuracy and efficiency.

(**Equation 6**)

Starting with the linear regression model, we had a mean accuracy of 94.7%. This accuracy was the same for ridge regression as well. For LASSO regression, we found that we can maintain above a 90% mean prediction accuracy with 𝜆 = 0.4. The false negative rate was calculated for each variation in the model by comparing the *Class* labels to the output labels to determine the percentage of output that was mislabeled as non-fraud. The maximized sparsity model has a mean false negative rate of 11.2%. False negative rate was calculated by finding the number of correct predictions and dividing it by total number of negatives in testing data to get true negative rate, and subtracting the true negative value from one. We then looked to minimize false negative rates for all three models.

Since the maximum sparse model has a 𝜆 value of 0.4, we looked at the interval with increments. We also limited the threshold to 2 decimal points. The findings are in the table below (Table 1).

| 𝜆 value | Threshold | False Negative Rate | Accuracy |
| --- | --- | --- | --- |
| 0 (Linear Regression) | 0.23 | 2.1% | 90.8% |
| 0.1 | 0.31 | 2.4% | 90.8% |
| 0.2 | 0.39 | 3.5% | 90.7% |
| 0.3 | 0.46 | 5.4% | 90.6% |
| 0.4 | 0.50 | 11.3% | 93.1% |

**Table 1.** Table of LASSO regression model mean accuracy and False Negative Rates with fixed intervals of 𝜆 values where mean accuracy is above 90% and False Negative Rate is minimized with respective threshold value which returned each false negative rate.

Here we can see a negative relationship between 𝜆 value, or sparsity, and false negative rate. At this point we have multiple models all of similar if not equal accuracy with varying degrees of sparsity and false negative rate. Depending on what is more prioritized (false negative rate or sparsity), we have a model which suits the needs of banks.

An important note is that the process by which averages were found for the models were constructed crudely. The training and testing subsets were constructed randomly and thus the models were trained and tested on randomized splits of the data. A more uniform and methodical approach that could be implemented is k-fold cross-validation. This method separates the total data into even segments where the number of segments can be considered as the proportion of the data you want to use to test the model (1). In the case of the current models, the segments would contain 20% of the data set with a total of 5 segments. 5 iterations of models would be created and accuracy averaged over all iterations. Each iteration would have a new testing segment. This allows us to capture the average model accuracy over the entire data set without the need of looking at over 10 or even 20 iterations to come to a similar if not equal result for finding mean accuracy of the models. Finding a way to do this to find the rates of true/false positives/negatives would be the second step to efficiently capture all the details of interest in the models.

Discussion

One of the first insights from the data analysis was the insignificance of *Amount* in detecting credit card fraud. Based on the histograms of Figure 1 showing a lack of distinguishable features and the shortest arrow of Figure 4 the amount of money in a transaction was deemed completely irrelevant to our model building. This conclusion contradicts our original assumptions before starting this study, in which we theorized fraudulent transactions would show a normal distribution of *Amount* where a certain range would contain the highest rate of fraudulent transactions. However, we cannot conclude that transaction amount limits or automatic flags for amount are useless in other scenarios. In fact, it might be because banks require further validation from the credit card holder with transactions of high amounts that *Amount* has no predictive quality in the dataset. If further information on the systems in place from the sample banks were provided, we could come to a more confident claim about *Amount* in the context of fraudulent transactions.

Unfortunately the anonymized parameters were understandably unknown to us, but this does not prevent insight into the significance of the models. The source of the dataset might know, specifically banks, which provided the information know the actual parameters behind the anonymized parameters. The set of models would be useful for these banks as a detection/flagging system to halt transactions showing parameters of fraud. Halted transactions can still be completed after additional confirmation is completed since the probability of a false positive is nonzero. A common practice currently is to call the credit card holder about the flagged transaction to confirm the authenticity. The models are flexible in that the probability of fraud output is converted to the binary classification of 0 (non-fraudulent) and 1(fraudulent) and the threshold for rounding the fraudulent probability can be altered. This fraudulent threshold can be adjusted by a bank to fit the desired balance of model accuracy and false negative rate. Considering that a bank is entrusted with the security of the client’s funds the threshold for fraudulent detection the false negative rate will be minimized to the lowest possible rates while matinting a high accuracy.

Appendix

Our work code

<https://github.com/konpayne/MATH-596-Final-Project-Payne-and-Miner/blob/main/MATH_596_Project_Brandon_Miner_Konnor_Payne.ipynb>

Same code, just backended and made neat:

<https://github.com/Branflakes333/credit_card_fraud_code>

Bibliography

1. Brownlee, Jason. “A Gentle Introduction to K-Fold Cross-Validation.” *MachineLearningMastery.Com*, 3 Oct. 2023, machinelearningmastery.com/k-fold-cross-validation/.
2. Elgiriyewithana, Nidula. “Credit Card Fraud Detection Dataset 2023.” *Kaggle*, 18 Sept. 2023, www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023/data
3. Hui, Jonathan. “Machine Learning - Singular Value Decomposition (SVD) & Principal Component Analysis (PCA).” *Medium*, Medium, 10 Jan. 2023, jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491.
4. Joe, Pearapon, and Kevin Glynn. “What Does the ‘Fit’ Method in Scikit-Learn Do?” *Stack Overflow*, 9 May 2021, stackoverflow.com/questions/45704226/what-does-the-fit-method-in-scikit-learn-do.
5. josh\_gray, and tuomastik. “Train\_test\_split() Error: Found Input Variables with Inconsistent Numbers of Samples.” *Data Science Stack Exchange*, 6 July 2017, datascience.stackexchange.com/questions/20199/train-test-split-error-found-input-variables-with-inconsistent-numbers-of-sam.
6. “Lasso (Statistics).” *Wikipedia*, Wikimedia Foundation, 17 Dec. 2023, en.wikipedia.org/wiki/Lasso\_(statistics).
7. “Linear Regression.” *Wikipedia*, Wikimedia Foundation, 7 Dec. 2023, en.wikipedia.org/wiki/Linear\_regression.
8. Oyedeji, Oluwafemi. “Eda and Classification (Credit Card Fraud).” *Kaggle*, Kaggle, 5 Oct. 2023, www.kaggle.com/code/demolaoyedeji/eda-and-classification-credit-card-fraud.
9. Pramoditha, Rukshan. “Singular Value Decomposition vs Eigendecomposition for Dimensionality Reduction.” *Medium*, Towards Data Science, 22 Mar. 2023, towardsdatascience.com/singular-value-decomposition-vs-eigendecomposition-for-dimensionality-reduction-fc0d9ac24a8e.
10. Raschka, Sebastian. “Principal Component Analysis.” *Sebastian Raschka, PhD*, 27 Jan. 2015, sebastianraschka.com/Articles/2015\_pca\_in\_3\_steps.html.
11. “Ridge Regression.” *Wikipedia*, Wikimedia Foundation, 7 Nov. 2023, en.wikipedia.org/wiki/Ridge\_regression.
12. the\_cat\_lady, and jarvis. “How Can I Hide Cell Contents in Google Colaboratory?” *Stack Overflow*, 3 Mar. 2018, stackoverflow.com/questions/49090476/how-can-i-hide-cell-contents-in-google-colaboratory.
13. tooty44, and o-90. “How Do I Create Test and Train Samples from One Dataframe with Pandas?” *Stack Overflow*, 10 June 2012, stackoverflow.com/questions/24147278/how-do-i-create-test-and-train-samples-from-one-dataframe-with-pandas.
14. Whitfield, Brennan. “A Step-by-Step Explanation of Principal Component Analysis (PCA).” *Built In*, 29 Mar. 2023, builtin.com/data-science/step-step-explanation-principal-component-analysis.
15. yemu, and smci. “How Do I Get the Row Count of a Pandas DataFrame?” *Stack Overflow*, 1 May 1959, stackoverflow.com/questions/15943769/how-do-i-get-the-row-count-of-a-pandas-dataframe.