Electronic transactions play a major role in our modern world. Debit and credit card technology has improved how business and shopping is done worldwide. Using digital payment can save customers and business time, reduce risk of losing money, and greater control over expenses. For all the benefits that come with electronic payment, one of the constant battles for patrons and businesses alike is fraudulent card use. This risk can be mitigated by private parties keeping their information secure, but this is not always 100% effective.

Businesses benefit from data recorded from electronic transactions to mitigate the chance of fraudulent activity. As an example we perform a case study using the Fraud Detection dataset from Kaggle ('https://www.kaggle.com/mlg-ulb/creditcardfraud#creditcard.csv') which includes credit card transactions as samples. The dataset has 31 columns. These include the time of the transaction, the amount, whether it was fraud or not, and many dimensionless columns of converted sensitive data. With this data we may compare how fraud transactions compare to non-fraud transactions in multiple ways. We will explore how the time the transaction takes place can effect when fraud takes place, the statistical distributions between fraud and non-fraud, and how the features of the dataset may be connected. These insights should allow us to build a model that detects whether a transaction can be flagged as fraudulent.

In the accompanying Jupyter Notebook, we share our analysis of this dataset. Using the Python Pandas, Seaborn, and Scikit-Learn libraries we are able to gain valuable insights into the dataset. Due to the low number of fraud samples we have an unbalanced classification problem. The fraud samples make up only 0.17% of our dataset. When looking at how much each charge was, the fraud charges tend to be less than a value of $1,000. The number of fraud samples also varies by time of day, as there are specific hours fraudulent charges spike in frequency. We found the dimensionless columns of the dataset to be uncorrelated, but a select few did correlate with the transactions amount. One of the more useful discoveries was found in comparing the empirical cumulative distribution for the fraud/non-fraud classes for each feature. Doing so revealed many of the features have visual differences in their statistical distributions between the two classes.

The rest of the analysis is a binary classification problem with uneven distributions. The two classes are fraud, or non-fraud. Since there is little information on the background of many of the columns, we choose not to normalize or preprocess the distributions of the features before fitting a model. In this example we will take an approach with a Random Forest Classifier. After parameter tuning using Grid Search and Cross Validation techniques, it was found that 160 trees in the forest with a maximum depth of 8 gave the best score. The analysis also searched over a select few oversampling parameters. The scoring method was done using Receiver Operator Characteristic Area Under Curve (ROC AUC). This method helps test the true detection and false positive rates, which is beneficial with uneven classes due to the lack of sensitivity in other binary classification scoring methods (i.e. accuracy or log-loss). Our final method had a successful ROC AUC scoring of 0.94, where the highest score is 1. It missed 12 fraud samples out of 102, and misclassified 27 non-fraud out of approximately 57,000. Finally, we tested feature importance by running experiments on our best model while removing a single feature at a time. In this way, we are able to see what features improve or decrease our ROC AUC scoring. Many of the features may be dropped, while the three most important features under this model are V14, V4, and Amount.

We were successful at building an analysis pipeline with a model that had strong ability at classifying fraud or non-fraud samples, using a Random Forest Classifier for our model. It attained an ROC AUC score of 0.94. Given more time or resources we would be able to further explore the feature set and parameter space of our forest using extensive grid searches. Other future directions would also include testing various resampling methods to improve the class balance.