Problem 4: IEEE-CIS Fraud Detection

Introduction

Credit card fraud is a rampant problem in the world. As payments go digital, this issue will only worsen as bad actors try and seize unsuspecting customers. The size of fraudulent transactions is such that a slight increase in fraud detection performance can lead to millions of dollars saved for the customers. There are several ways to predict whether a transaction is fraudulent. This paper will explore the different ways of predicting whether a transaction is fraudulent or not through machine learning algorithms.

Set-Up

Before reading the data, a few helper functions were used to help understand and manage the data and increase the interpretability of the models. Once the data was read, I merged the identity and transaction data sets for the train and test sets. Approximately 250,000 rows (50% of the dataset) were removed at random to reduce computational load. A 'reduce memory' function was used to reduce the memory of the data set by 67%.

Exploratory Data Analysis

A table containing counts and percentages of null values was created, which showed many columns with more than 80% of data missing. Then, to visualize data appropriately, the data was split into numerical and categorical features. My approach to this was relatively straightforward – examine sets of features in groups, check for null values, patterns, cross variant behavior, and prepare the data to run the model.

The number of fraudulent cases (target variable) was the first variable to be examined, as shown in chart 1. Fraudulent cases are relatively low relative to non-fraudulent transactions.

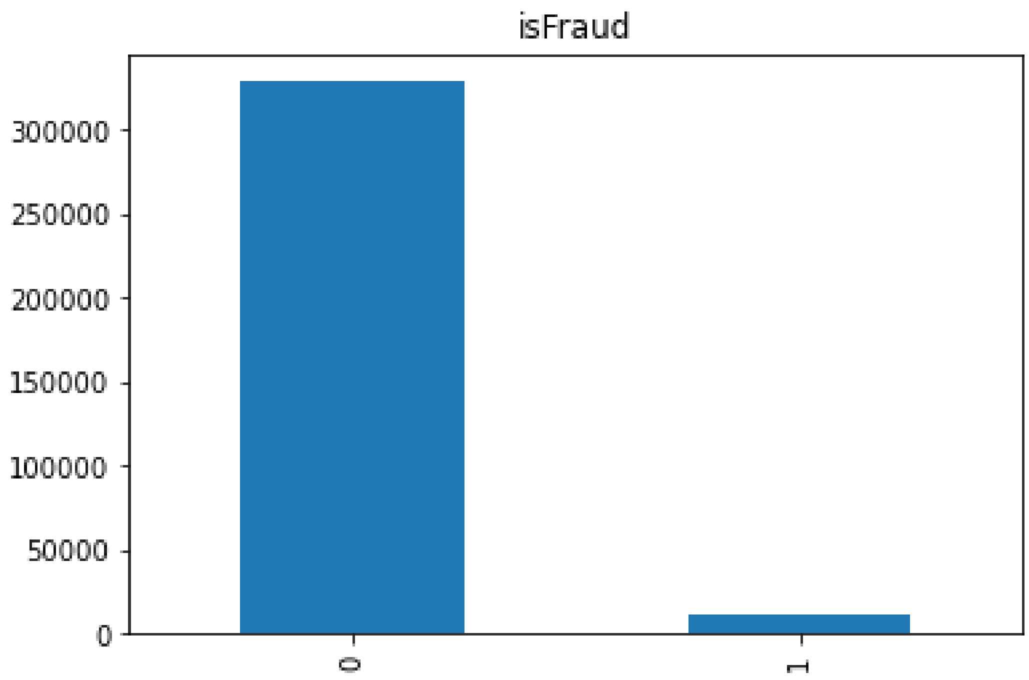


Chart 1

Numerical features related to ‘id’ were visualized – these features showed that several ‘id_xx’ columns have a few values that occur far more frequently than others. On the other hand, a couple of ‘id’ variables were distributed similarly, with high-frequency values at regular intervals.

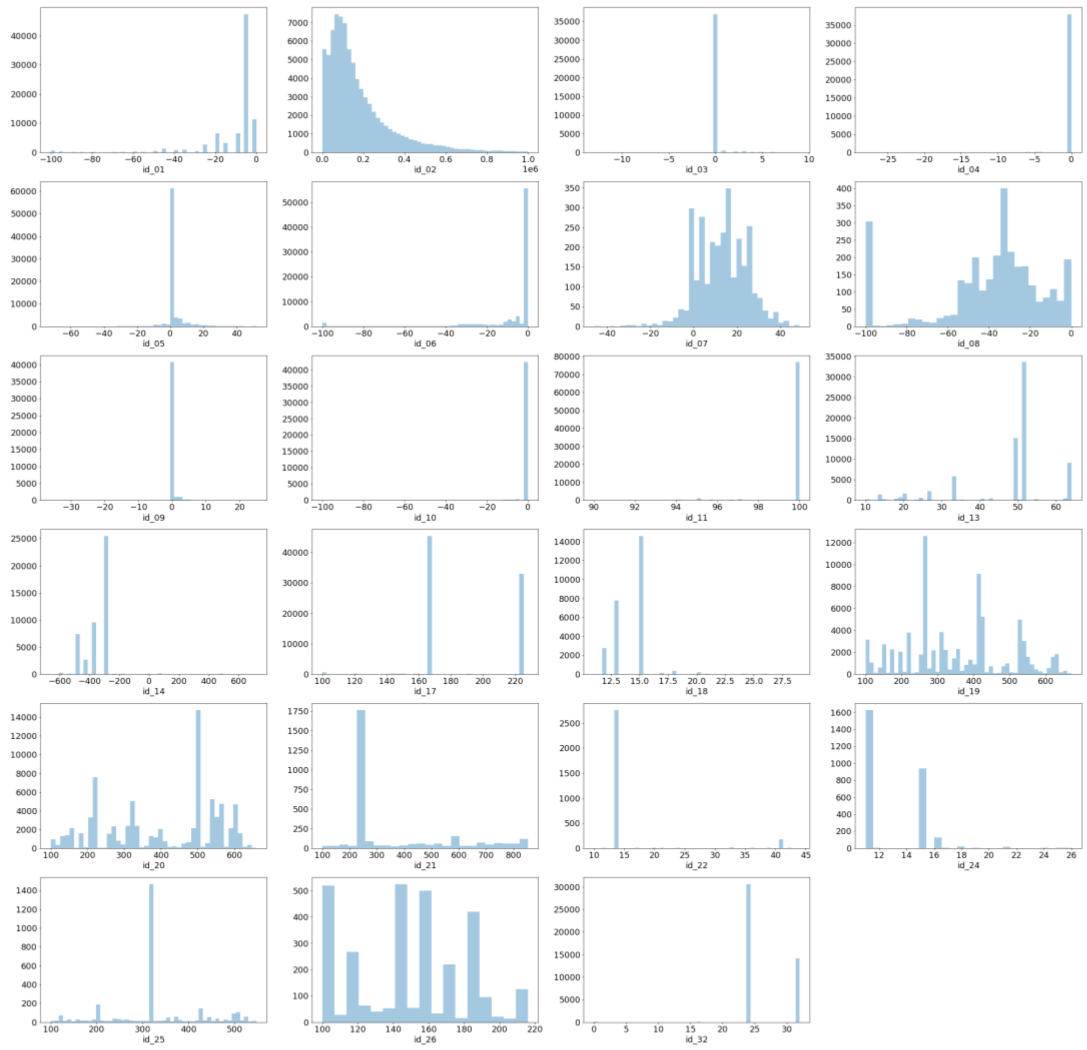


Chart 2

With categorical id features, the counts of each variable was illustrated to find the proportion of null values with relation to other unique values in each variable. All categorical id variables had an overwhelmingly large proportion of null values.

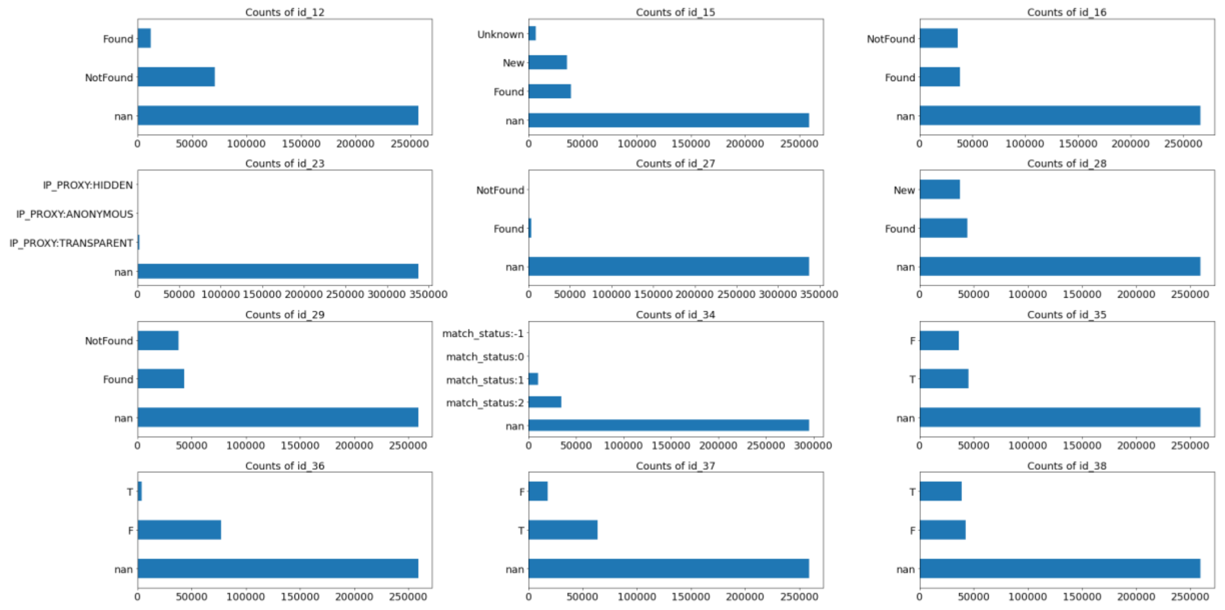


Chart 3

Devictype, Deviceinfo, and emaildomain were also examined (in chart 4) – the results were in line with what was expected. Windows system was most commonly used in transactions, desktops were used more than mobile for transactions, and Gmail was the most popular email address.

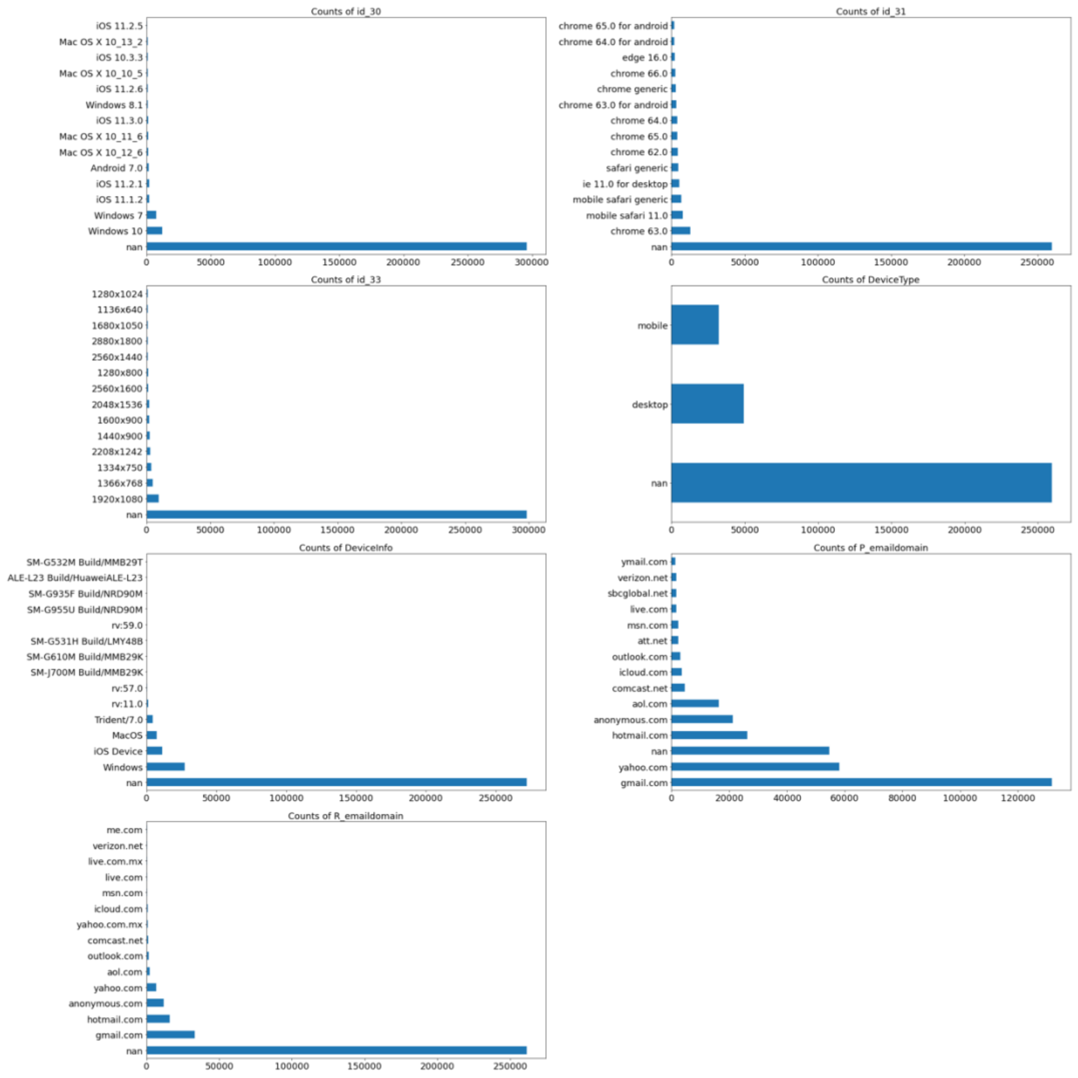


Chart 4

Next, the card variables were visualized. Out of the numerical card features, it seems like card3 and card5 have several missing values with a few non-null values that make up most of their data. 'Card1' and 2 seems to have a more uniform distribution (not normal), that has values across a large range. This tells us that some components of payment information were recorded more thoroughly than others.

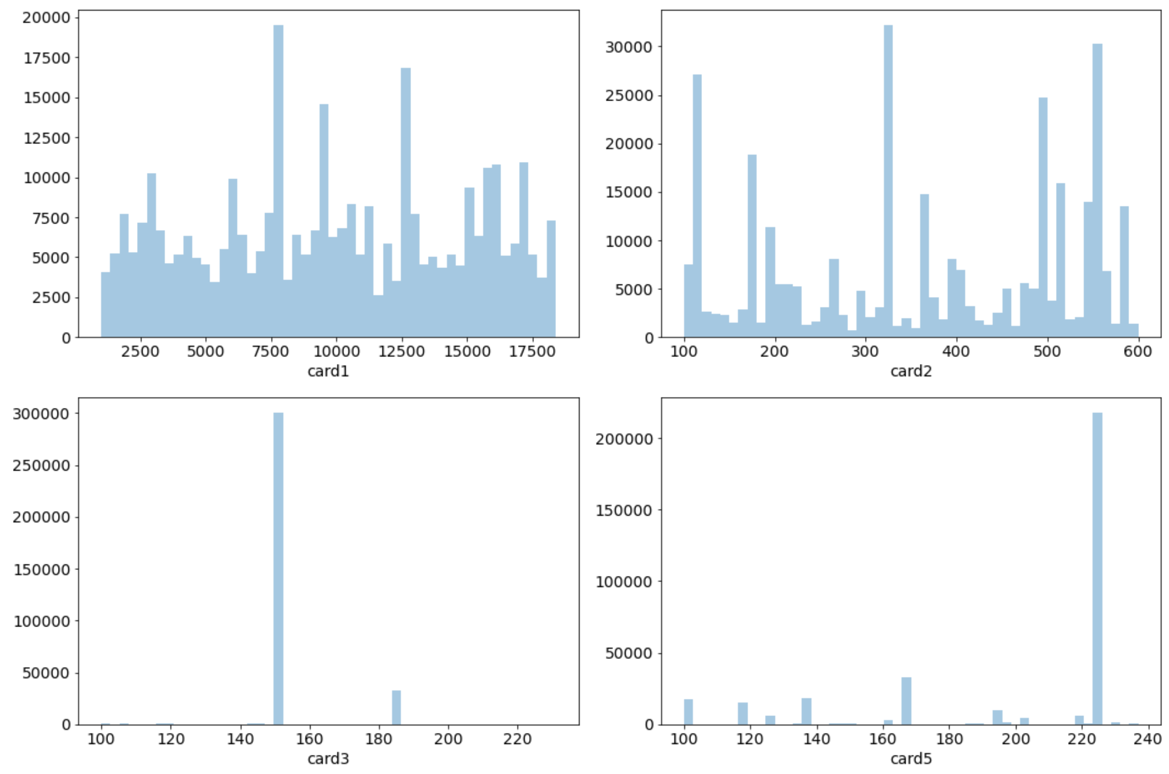


Chart 5

The categorical card variables show the type of credit card and the credit card company. I mapped a line showing the types of cards with high levels of fraudulent activities. It seems that Discover cards tend to have high levels of fraudulent transactions, with American Express having the least amount of fraudulent transactions. Credit cards were most at risk for fraud, whereas charge cards were one of the safest options.

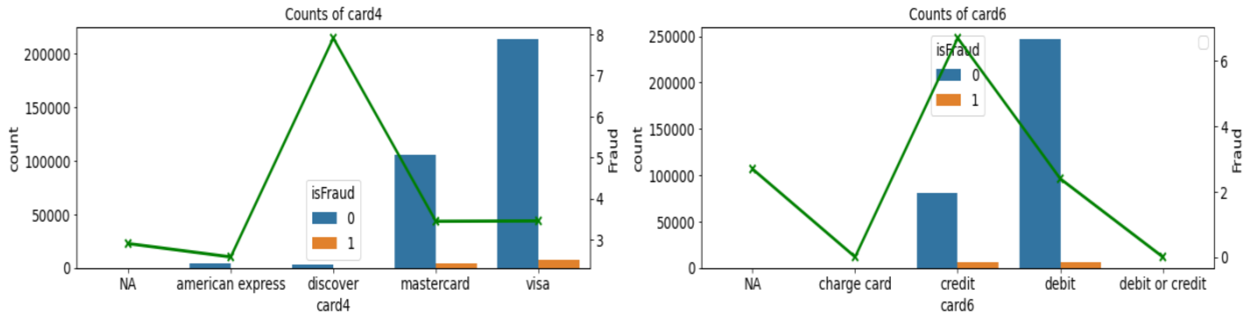


Chart 6

The 'Mx' features were visualized to understand their distributions. Then, a fraud line was mapped onto them to see what values exhibited high levels of fraud (proportionate to its own set of non-fraudulent data). Since the meaning of these variables is masked, it's challenging to interpret what any of these variables suggest. However, some of these variables have a large proportion of missing values. Since these variables represent whether certain pieces of data matched, such as names on cards, etc., it seems reasonable that most of these variables showed higher levels of fraud when value = F (False), which may mean that the data did not match.

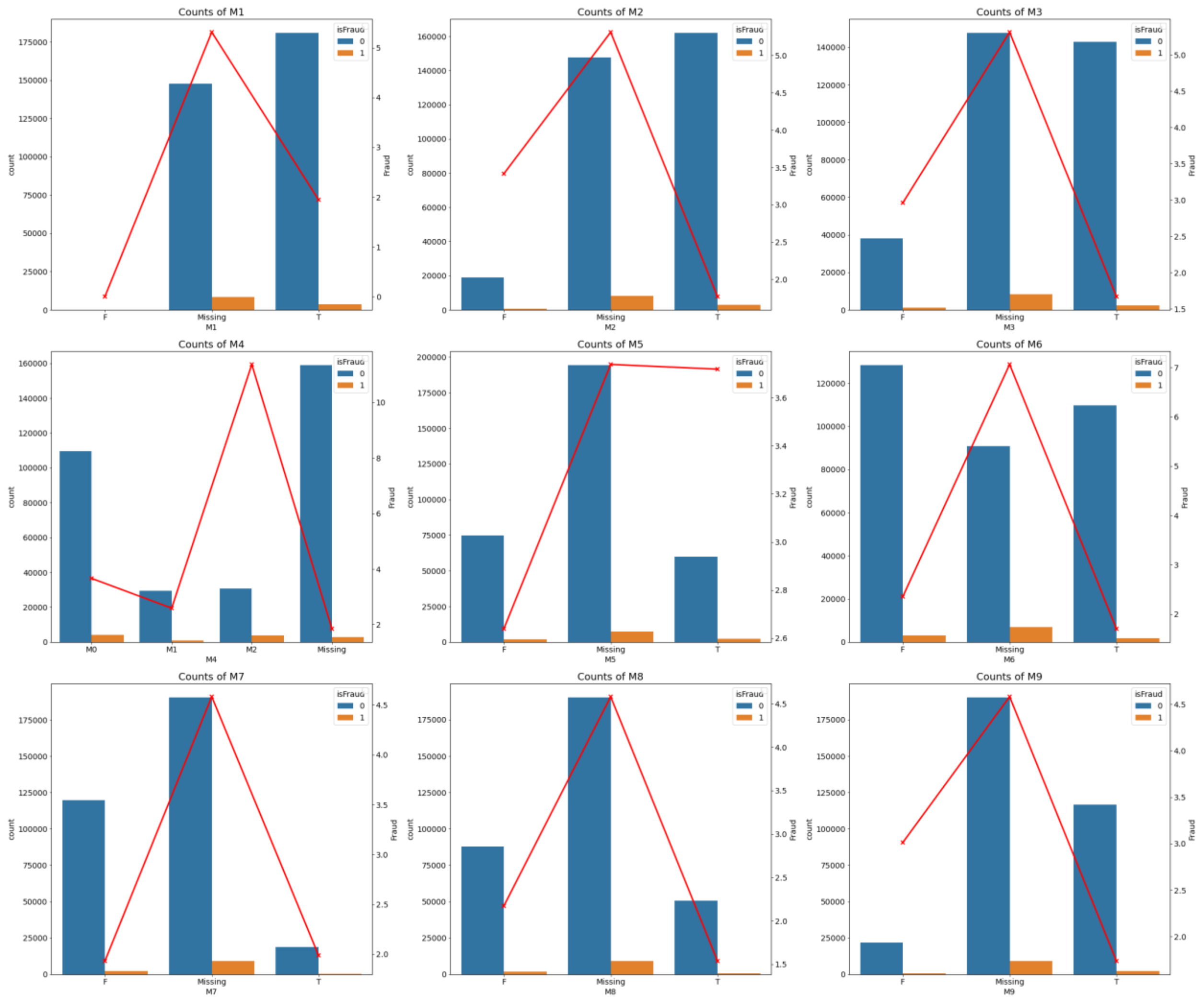


Chart 7

I wanted to plot the transaction amounts, but a log transformation to normalize the variable seemed appropriate due to how skewed they were. I compared the variable across the train and test dataset. The variable in both sets seems to be distributed quite similarly.

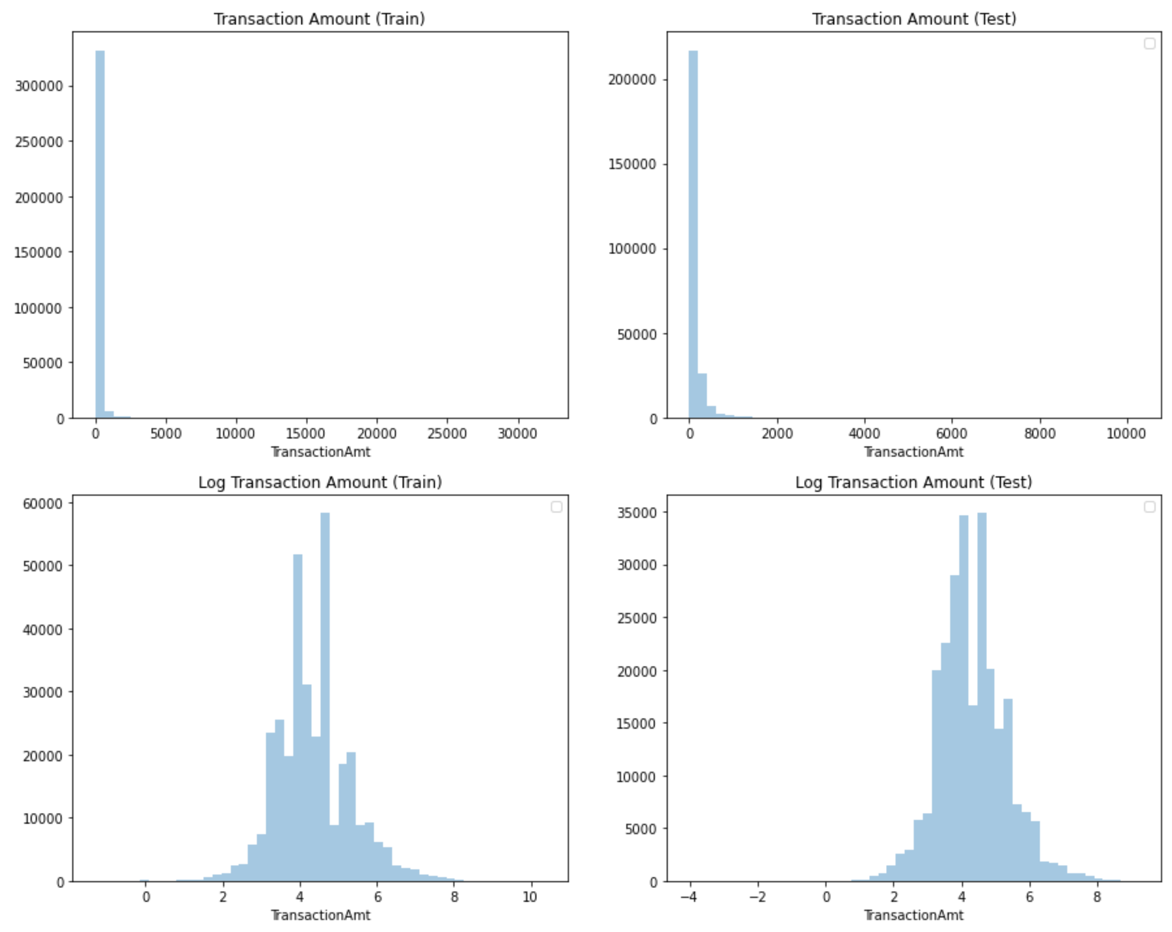


Chart 8

Next, the product codes were examined. These variables represent not just products but also services. It seems that category ‘C’ has a high proportion of fraudulent transactions. I mapped across Transaction amounts, and it is evident that fraudulent transactions occurred on higher value transactions for most categories.

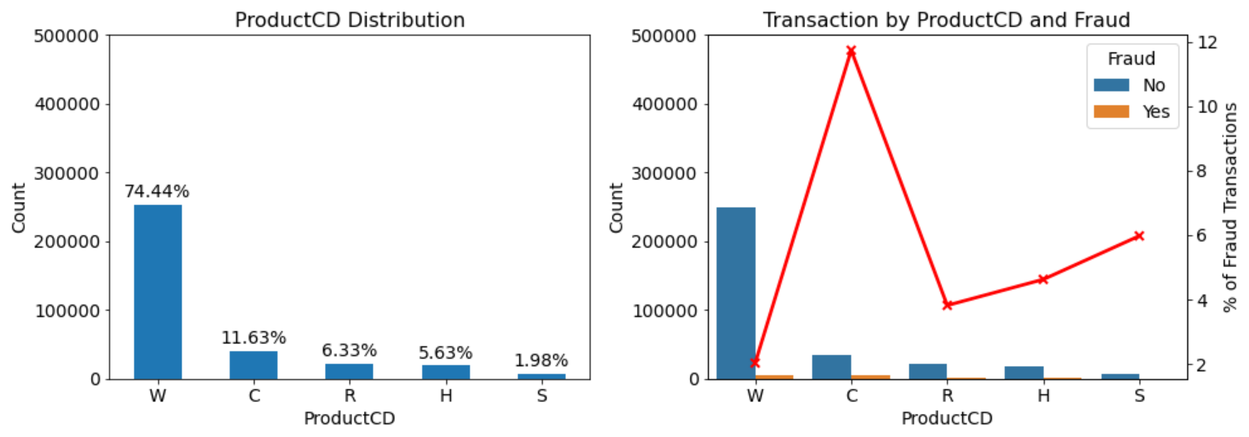


Chart 9

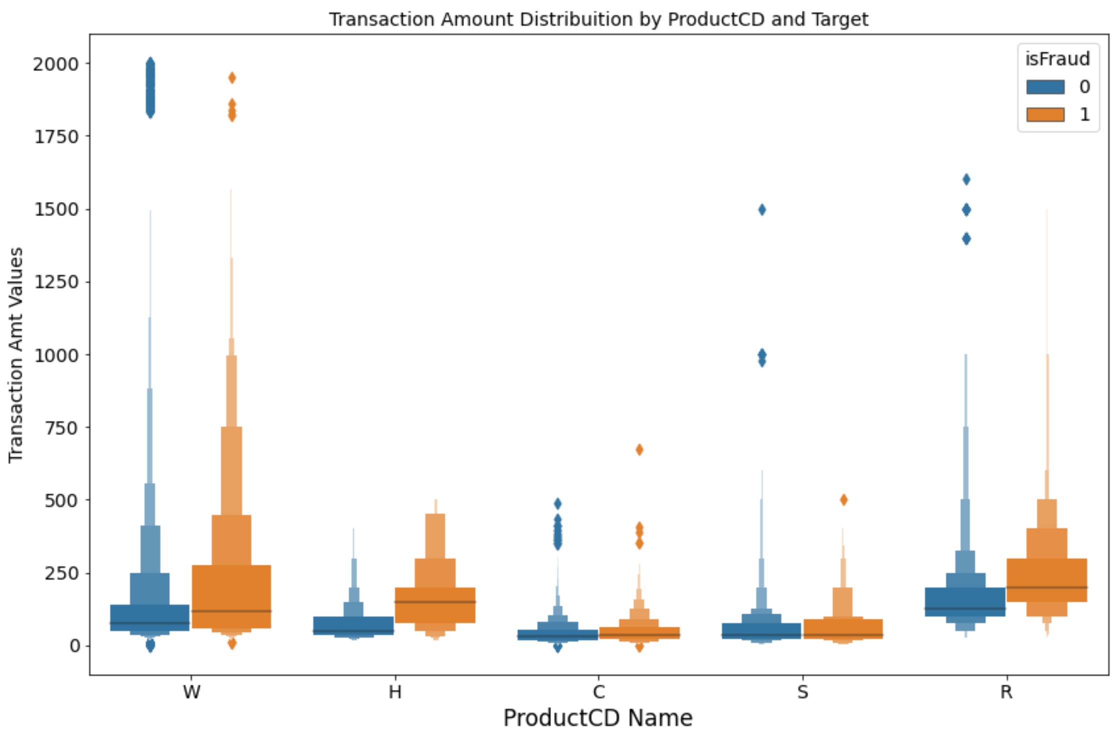


Chart 10

The dataset contains a variable TransactionDT, a time delta from a given reference (it is not a timestamp). This variable can tell us the distribution of the data across time.

It seems that the train data and test data are taken from two separate periods, as there is no overlap in the two distributions. The lack of overlap may affect the method of cross validation.

For the last part of EDA, I examined the counts of email domains against fraudulent activity. It seems that protonmail has a disproportionately high number of fraudulent transactions. Gmail appears to be the best option for users, given the very low proportion of fraudulent transactions.

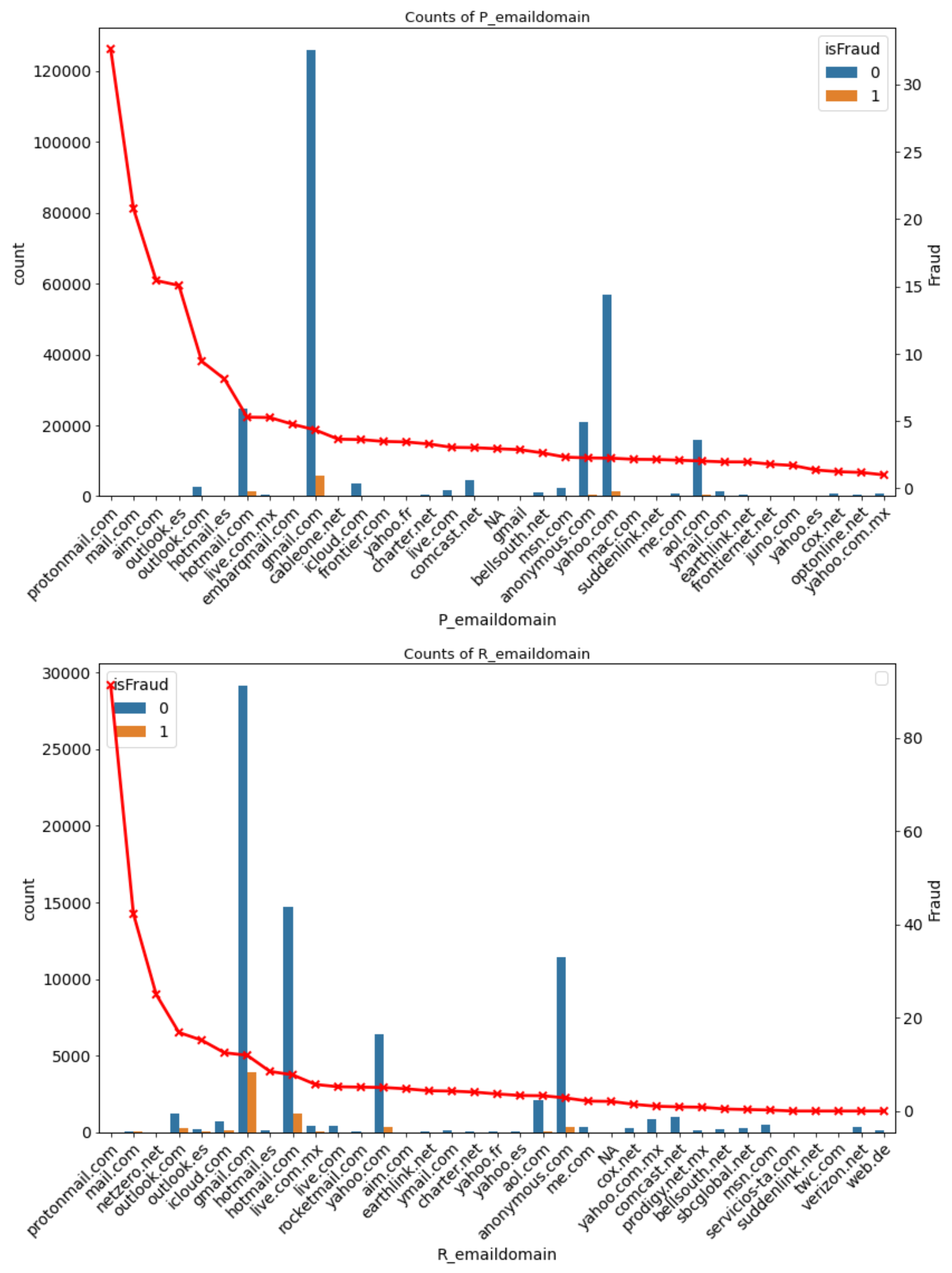


Chart 11

Preparing Data for Modelling

To make the data sets model-ready, a few changes were needed. To start off, the email strings were split, and hyphens were replaced with underscores. Due to many columns containing a majority of null values, I dropped all columns where null values were >50%. Additionally, I dropped columns that only consisted of a single unique value since this would not affect predicting a fraudulent transaction. A simpler model allows for faster computation times, improves interpretability, and prevents overfitting.

To address a large number of columns with missing values, I retrieved numerical and categorical columns for both datasets. Before imputing the variables, I checked for infinity values and replaced them with nan. Doing this beforehand would help remove all nan values. The categorical missing values were replaced with 'na.' Predictive models generally require two inputs, the dataset as the x value and the target variable as the y variable. The train and test data sets were split accordingly and sorted the dataset based on the transaction delta time. The TransactionID, and TransactionDT variables were dropped from the X value data sets as placeholders and identifiers.

Building Models

Since missing values still need to be imputed in the numerical columns, a pipeline seemed like the right approach to building a model. A pipeline allows us to transform data as well as run a model in that order. Additionally, they make the workflow more straightforward to understand. Since the categorical values are objects in string form, they are not machine-readable at the moment. To prepare the categorical variables, the data was encoded using a OneHotEncoder, which assigns integer values in place of categorical values and replaces each of those integers with binary values. The pipeline comes in handy here as the OneHotEncoder can be used within the pipeline, along with a StandardScaler (standardizes data such that mean is 0 and standard deviation is 1) for the numerical columns.

Random Forest Classifier

Random Forest Classifier (RFC) was used for the first model. The RFC is a classification algorithm consisting of several decision trees. It uses bagging and feature randomness to create uncorrelated decision trees, which ultimately improves model performance (predictive capabilities). Although RFC handles missing data well, I preprocessed the data as a conservative way to ensure that the model runs smoothly. The data is available for use across a range of different models. Once the model was run, it was fitted to the training data set and using the test dataset, the target variable was predicted. To understand which features had the most significant impact on predicting whether a transaction is fraudulent or not, the feature importances (scores assigned to features based on their importance in predicting the target variable).

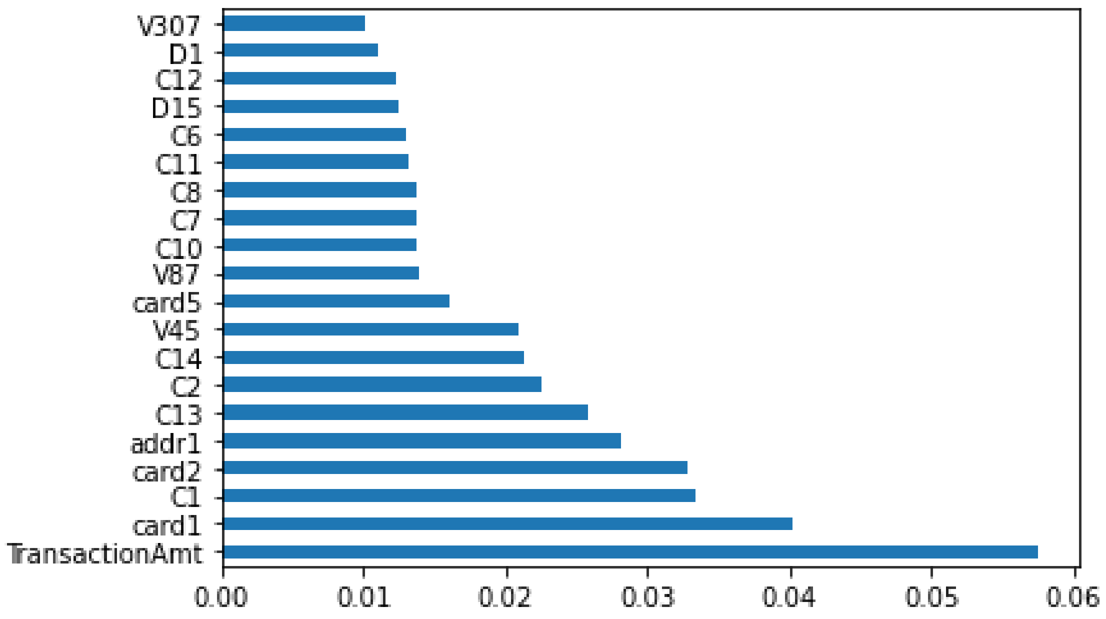


Chart 11

As the chart suggests, the transaction amount is the most important feature in predicting fraud. Card 1 & 2 seem to be important features alongside addr1 and some C, D, and V variables.

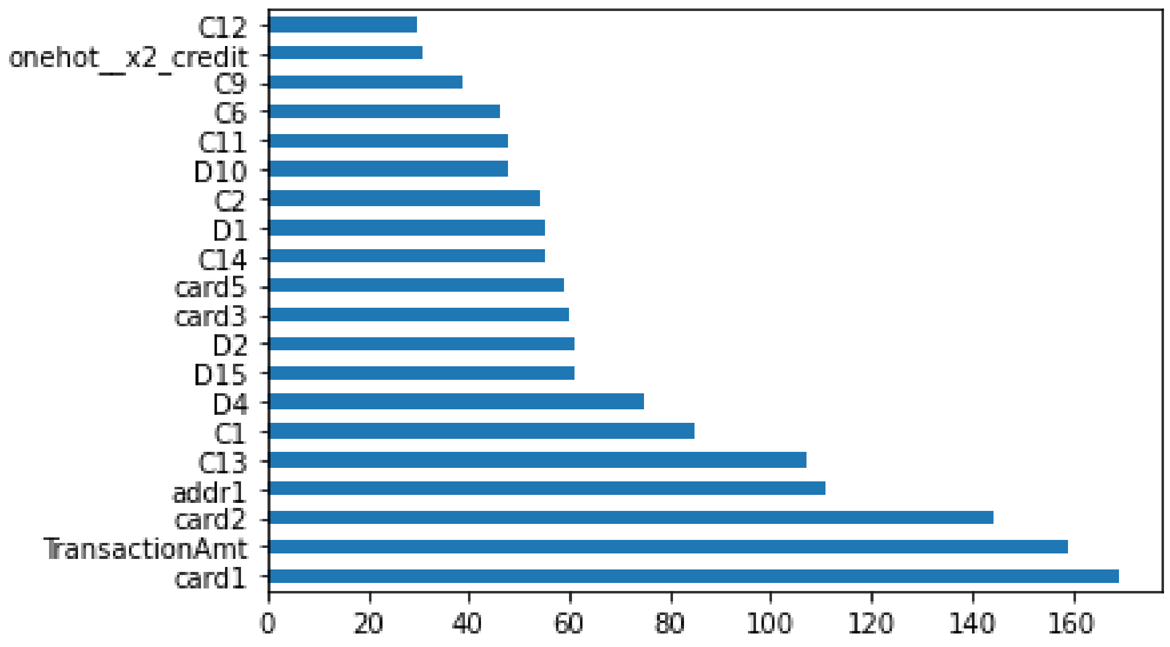
The test data set was run through the model to predict whether a transaction was fraudulent or not. The model was then cross-validated to measure the model's performance. An AUC score (Area
Under Curve) was calculated, which is a measure of separability.

AUC Score of RFC: 0.8477

A higher AUC score means that the model is better able to predict true positives and true negatives. This score was calculated as a way to compare different models' performance.

Light Gradient Boosting Machine

The Light Gradient Boosting Machine (LGBM) is an algorithm that also deploys decision trees but uses an ensemble method where the next tree reduces the error produced by the previous tree. It is far quicker than other boosting algorithms as it uses a leaf-split method instead of a level split method. This model is great for our case as it handles large datasets with ease. That being said, Boosting algorithms tend to overfit the input data, so it is important to keep that in mind while evaluating the performance of the model. Our approach to running the model was similar to that of the Random Forest Classifier. After fitting the data and predicting fraud vs. not fraud in the test data, the most important features of the model were extracted.



Based on the chart, card1 seems to be the most important, with TransactionAmt a close second. However, the order of the features has changed, which could be attributed to the different methods of predicting fraudulent transactions between the Random Forest Classifier and the LGBM algorithm.
The model was cross-validated, and the AUC score was calculated.

AUC Score of LGBM: 0.9041

Model Evaluation

Since the AUC score of the LGBM model - AUC Score of LGBM: 0.9041
- is higher than the AUC score of the RFC model - AUC Score of RFC: 0.8477 – it is fair to say that the LGBM model has done a better job in predicting fraudulent transactions. That being said, it is also very likely that the model is overfitted to this dataset and may not perform well on another dataset.

Conclusion

The predictive power of the algorithms used in this paper illustrates the possibilities across different domains. The Light Gradient Boost Model did a good job predicting fraud. Still, another classifier, such as the Random Forest, might be better suited for long-term usage across several data sets, as it prevents overfitting.

In future work, I would like to use a few more models and run each model with a higher number of trees to improve model performance and cross-validate based on a time split. In this case, the number of trees were severely limited due to time constraints. Additionally, an exhaustive EDA would be beneficial in understanding the relationships in the dataset on a granular level.

https://www.kaggle.com/c/ieee-fraud-detection