Fraud Detection Predictive Analytics

Ishan Ambike

4/13/2022

Introduction:

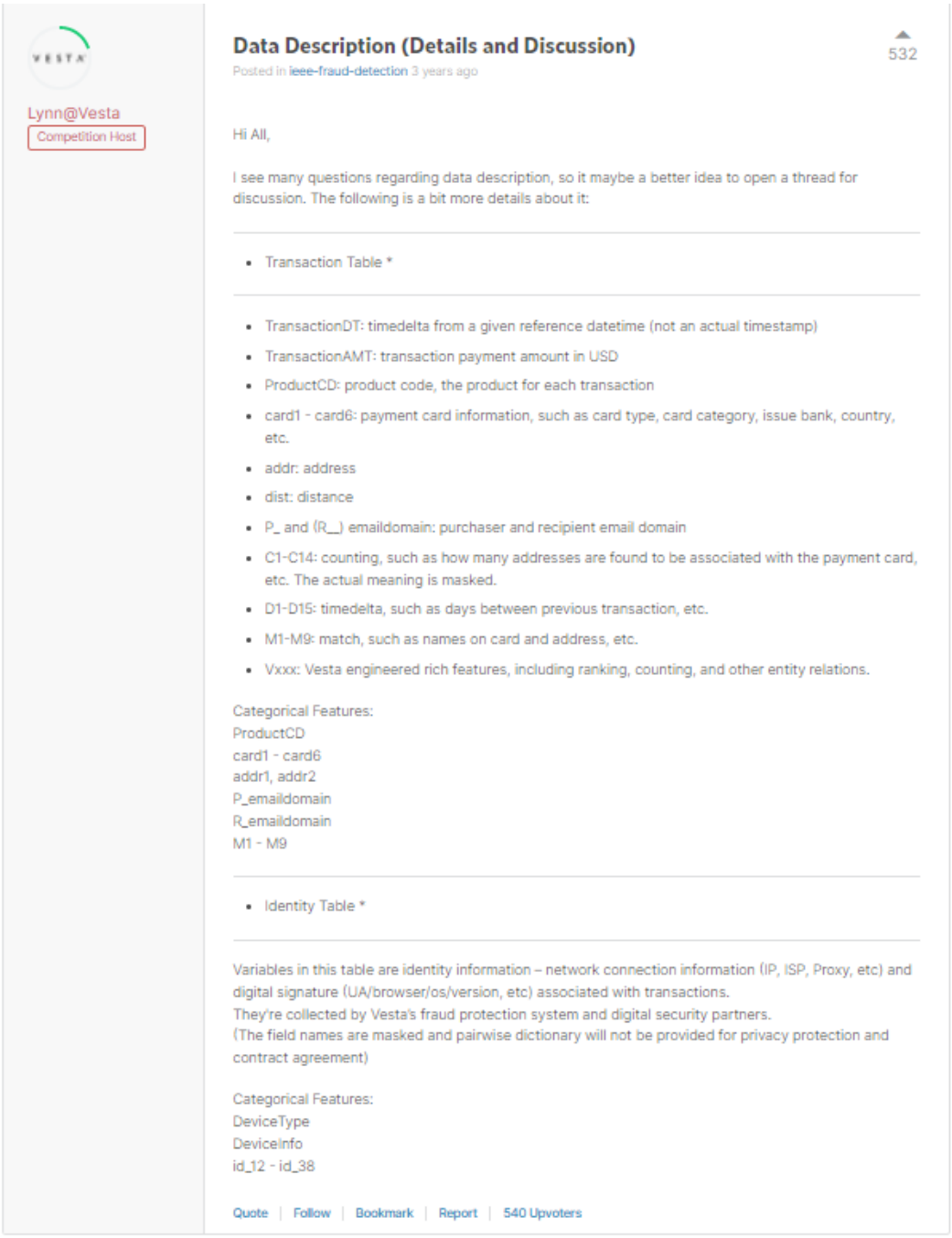
This problem involves predicting whether a transaction is a fraud based on data available about the transaction and the customer. The data was made available by Kaggle in the IEEE-CIS Fraud Detection competition hosted in 2019. The data for this problem was collected from the IEEE Computational Intelligence Society (IEEE-CIS) and Vesta Corporation. The aim of this problem is to predict fraud transactions correctly and also minimize false positives. To solve this problem I built and analyzed various Machine Learning models and found which model predicts fraudulent transactions most accurately.

Data:

The data provided is about transactions and user identity.

Transaction: Data about transaction amount, time, product, card details, address, email, Vesta engineered rich features, etc
Identity: Data about device, browser used, network connection, digital signature, etc.

Snapshot of data description provided by Vesta:



Lynn@Vesta

Competition Host

Data Description (Details and Discussion)

532

Posted in ieee-fraud-detection 3 years ago

Hi All,

I see many questions regarding data description, so it maybe a better idea to open a thread for discussion. The following is a bit more details about it:

- Transaction Table *

- TransactionDT: timedelta from a given reference datetime (not an actual timestamp)
- TransactionAMT: transaction payment amount in USD
- ProductCD: product code, the product for each transaction
- card1 - card6: payment card information, such as card type, card category, issue bank, country, etc.
- addr: address
- dist: distance
- P_ and (R__) emaildomain: purchaser and recipient email domain
- C1-C14: counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked.
- D1-D15: timedelta, such as days between previous transaction, etc.
- M1-M9: match, such as names on card and address, etc.
- Vxxx: Vesta engineered rich features, including ranking, counting, and other entity relations.

Categorical Features:
ProductCD
card1 - card6
addr1, addr2
P_emaildomain
R_emaildomain
M1 - M9

- Identity Table *

Variables in this table are identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.
They're collected by Vesta's fraud protection system and digital security partners.
(The field names are masked and pairwise dictionary will not be provided for privacy protection and contract agreement)

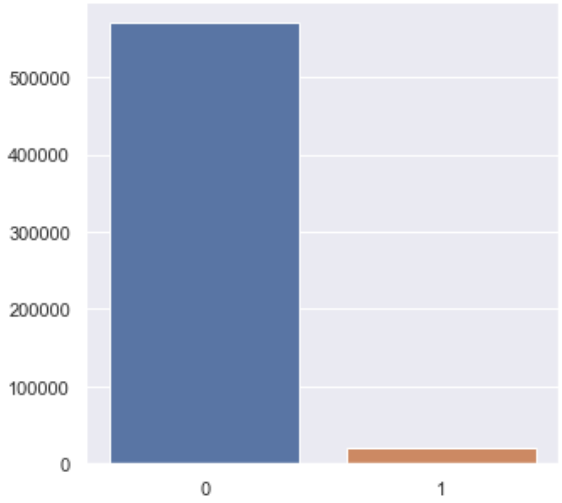
Categorical Features:
DeviceType
DeviceInfo
id_12 - id_38

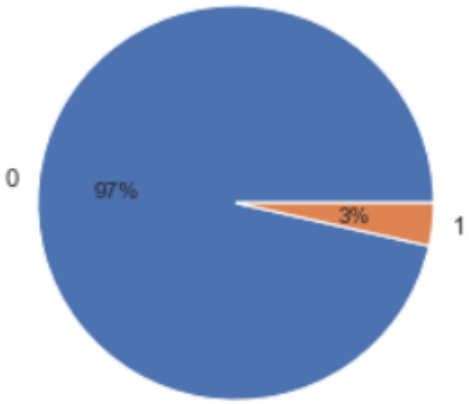
Quote | Follow | Bookmark | Report | 540 Upvoters

As this is raw data it contains many missing values it would be necessary to handle the missing values and also transform the data into a usable format for making predictions. Hence we would have to do EDA and feature engineering before preparing predictive models.

EDA:

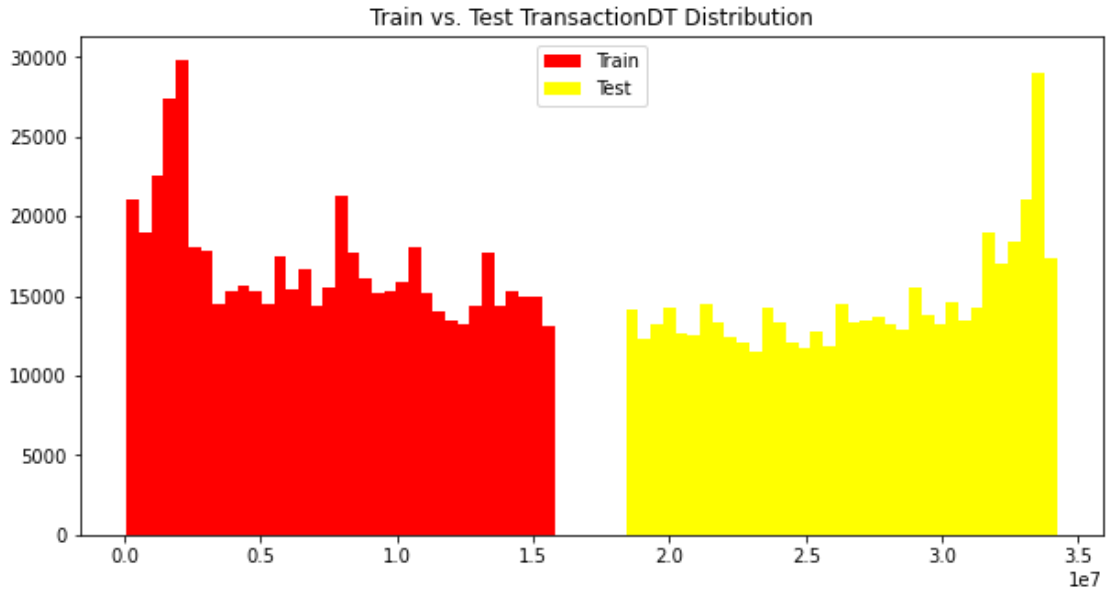
Analysis from isFraud column:





So from the above plots, we can see that most of the transactions are not labeled as fraud. Only 3% of the training transactions are fraud.

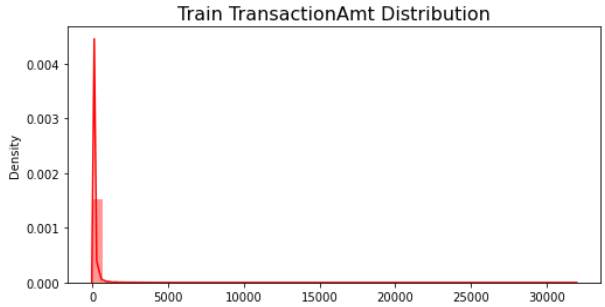
Train and Test TransactionDT analysis:

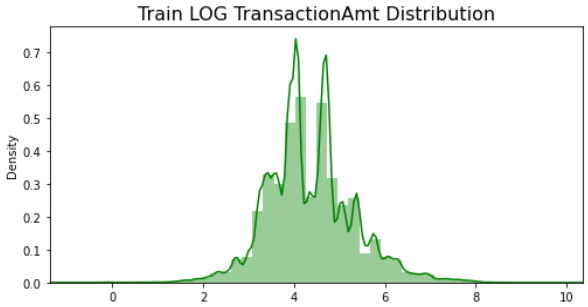


From the TransactionDT plot, we understand that train and test transactions are from different time periods and do not overlap.

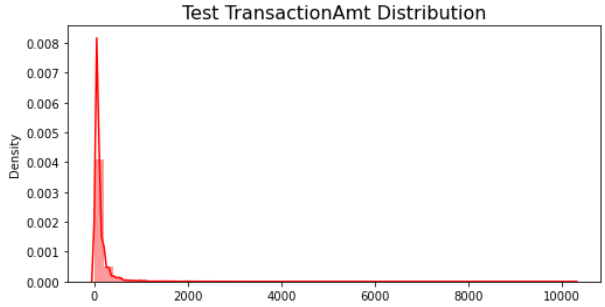
Analysis of transaction amount:

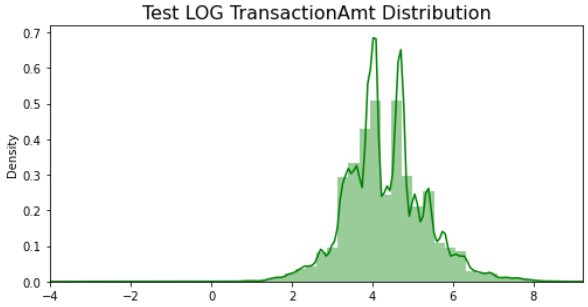
Train:





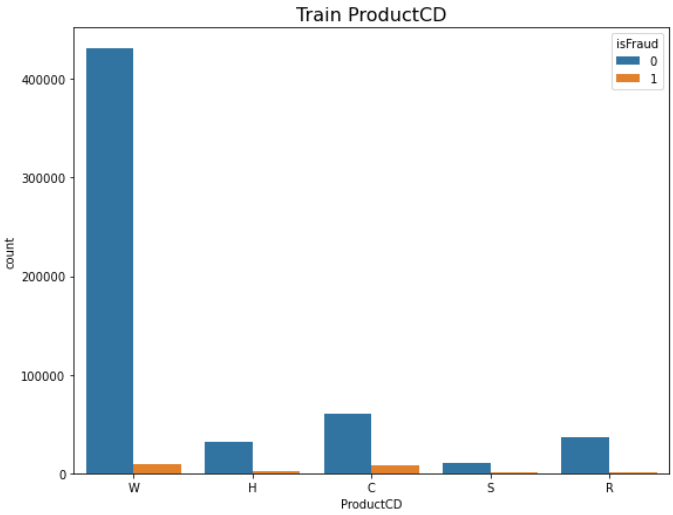
Test:

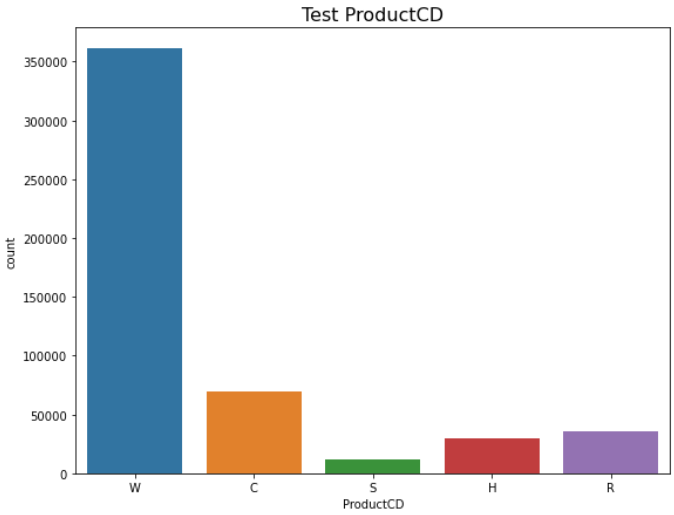




As TransactionAmt doesn’t follow the bell curve, we can use log transformation to normalize the data. It removes or reduces the skewness of our data which makes the data symmetric.

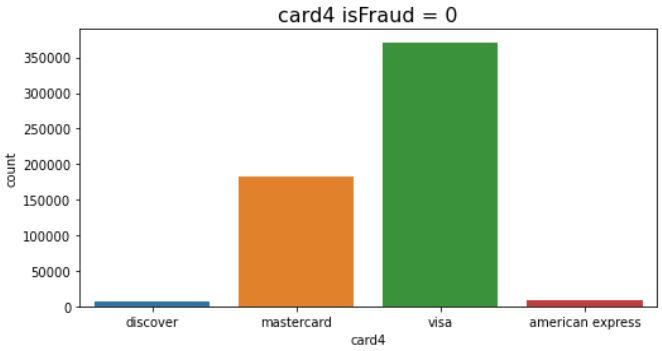
ProductCD:

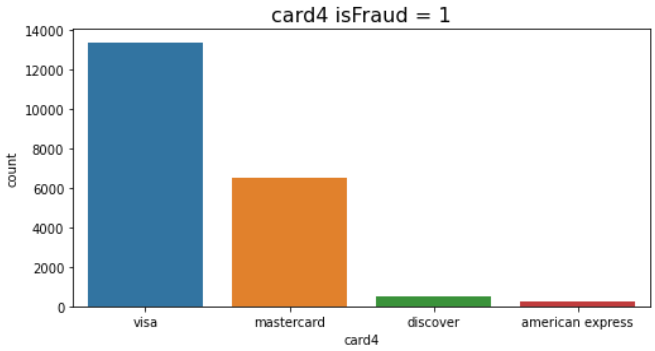




From the train data, we can observe that product W has the most transactions, followed by product C. We see a similar pattern in test data as well.

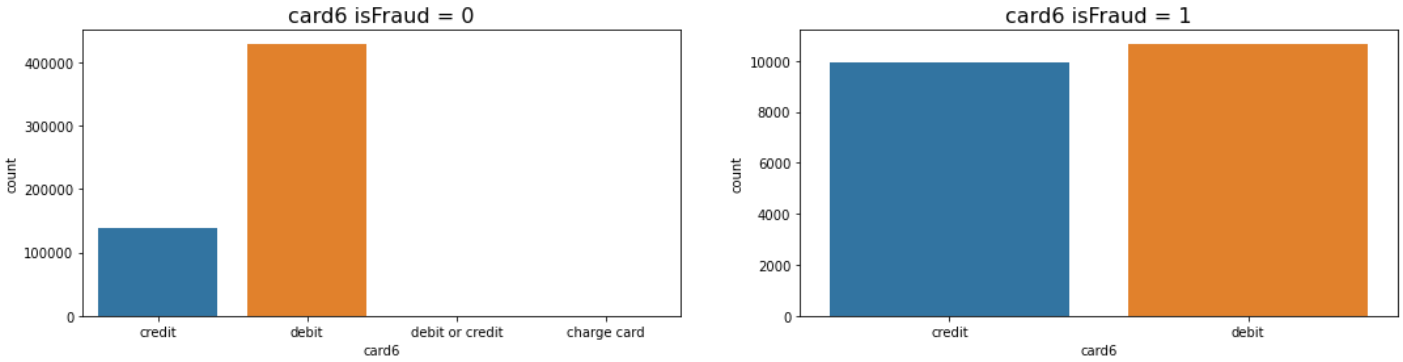
Card4:





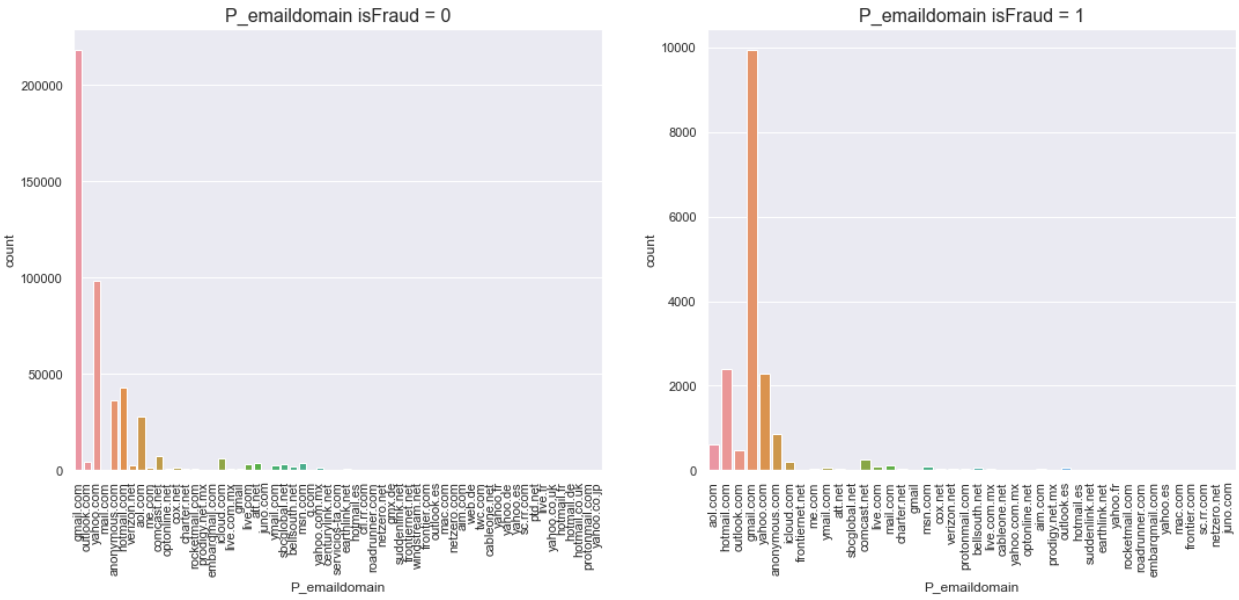
Visa cards have the most transactions followed by MasterCard. The number of frauds also follows the same pattern among cards.

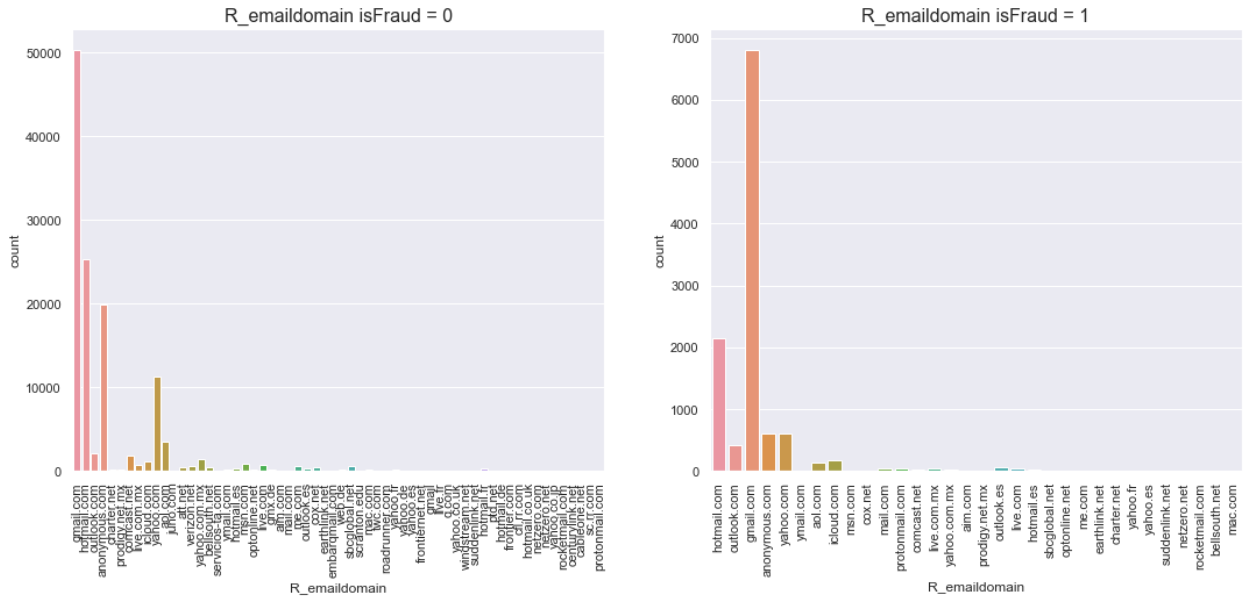
Card 6:



As expected more fraud transactions were committed on debit transactions than credit transactions.

P_emaildomain and R_emaildomain:





Most frauds were committed on Gmail followed by HotMail and yahoo.

Feature Engineering:

Transforming DeviceInfo column:

DeviceInfo contains raw data about the devices used. Using this information I have created a new column ‘device_name’ which has the name of the device manufacturer. This reduces the categorical values about the devices being used.

Transforming id_31 column:

id_31 has information about the browser being used. I have created a new column ‘lastest_browser’ where 1 will be set for the latest browsers.

Transforming P_emaildomain and R_emaildomain:
For these columns, I have created separate columns to contain the domain bin and suffix.

Log transforming TransactionAmt:
As discussed in the EDA part, TransactionAmt is log-transformed to follow normal distribution.

Finding decimals in TransactionAmt:

As per the findings of developers in the discussion section, the number of decimal places can tell whether the transaction amount is converted to USD from a different currency. Hence I have found this value and stored it in ‘TransactionAmt_decimal’.

Handling missing values:
For columns with large missing values (>200,000 values missing), I have dropped the column.
For the remaining columns, I have handled the missing values with the most frequent value(mode).

To prepare the data for prediction I have created dummy variables of the categorical values.

Model building and comparison:

For predictions, I will be using 4 different models and compare the accuracy and time required by the models. The models used are:

- RandomForestClassifier
- Neural Networks
- XGBClassifier
- LGBMClassifier

I will be using the best model among these for the final prediction.

RandomForestClassifier:

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Summary of prediction using RandomForestClassifier:

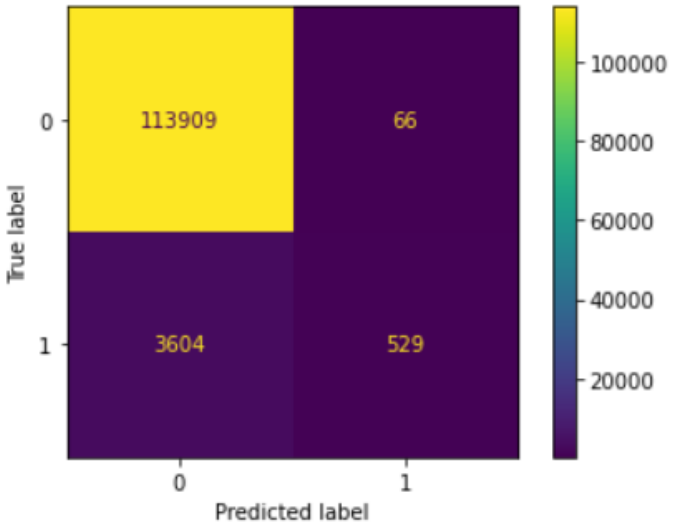
```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.98    113975
         1.0       0.89      0.13      0.22      4133

    accuracy                           0.97    118108
   macro avg       0.93      0.56      0.60    118108
weighted avg       0.97      0.97      0.96    118108

Accuracy:
0.9689267450130389

Confusion Matrix:
```



Neural Networks:

Neural Networks are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Hypertuning NN using RandomizedSearchCV:

```
RandomizedSearchCV(cv=5,
                   estimator=<keras.wrappers.scikit_learn.KerasClassifier object at 0x00000207C1736910>,
                   n_jobs=4,
                   param_distributions={'Neurons_Trial': [5, 10],
                                        'Optimizer_Trial': ['adam', 'rmsprop'],
                                        'batch_size': [10, 20],
                                        'epochs': [10, 20]},
                   scoring='f1')
```

Best parameters returned by RandomizedSearchCV:

```
random_search.best_params_
```

```
{'epochs': 20,
 'batch_size': 20,
 'Optimizer_Trial': 'adam',
 'Neurons_Trial': 10}
```

I have built a NN classifier using these best parameters.

Summary of prediction using Neural Networks:

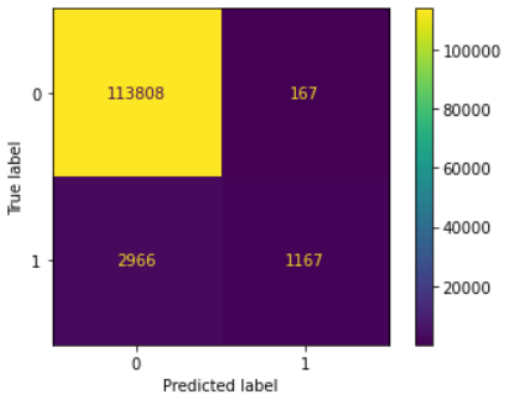
```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.99    113975
         1.0       0.87      0.28      0.43      4133

    accuracy                           0.97    118108
   macro avg       0.92      0.64      0.71    118108
weighted avg       0.97      0.97      0.97    118108

Accuracy:
0.973473431096962

Confusion Matrix:
```



XGBClassifier:

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

Summary of prediction using XGBClassifier:

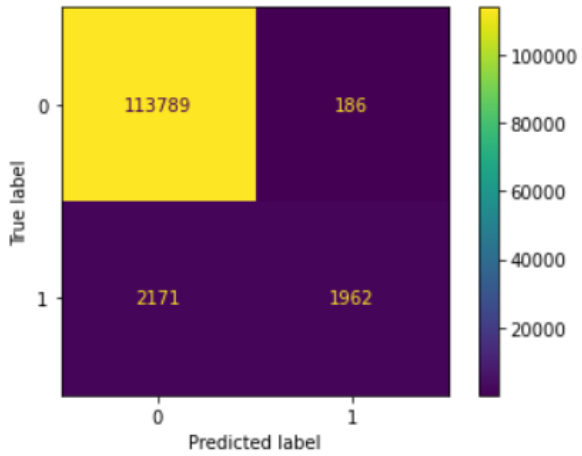
```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99    113975
         1.0       0.91      0.47      0.62      4133

    accuracy                           0.98    118108
   macro avg       0.95      0.74      0.81    118108
weighted avg       0.98      0.98      0.98    118108

Accuracy:
0.9800436888271751

Confusion Matrix:
```



LGBMClassifier:

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

Summary of prediction using LGBMClassifier:

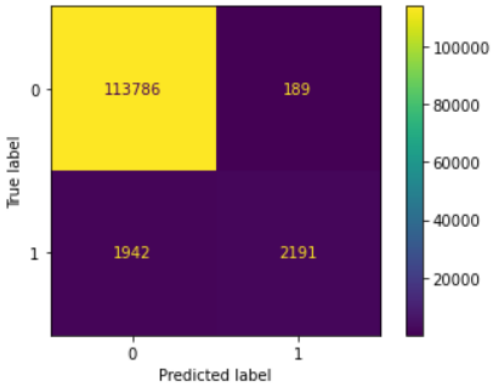
```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.98      1.00      0.99    113975
         1.0       0.92      0.53      0.67      4133

    accuracy                           0.98    118108
   macro avg       0.95      0.76      0.83    118108
weighted avg       0.98      0.98      0.98    118108

Accuracy:
0.9819571917228299

Confusion Matrix:
```



Selecting the best model and final prediction:

	Model Name	Accuracy	Time
0	LGBMClassifier	0.981957	48.911907
1	XGBClassifier	0.980044	277.468931
2	ANN	0.973473	1333.601733
3	RandomForestClassifier	0.968927	120.808890

As seen from the table above, LGBMClassifier has the best accuracy among the 4 models. Also, it is the fastest of all. Hence I will be using LGBMClassifier to make the final prediction.

Hypertuning LGBMClassifier using Optuna:

To find the most optimal parameters to increase accuracy, I am hyper tuning LGBMClassifier. Optuna is an open-source hyperparameter optimization framework to automate hyperparameter search(https://optuna.org/).

But best model returned by Optuna has an accuracy 96.4% which is less than initial model, I will go ahead with the initial model for the final prediction.

Conclusion:

From this fraud detection problem, I was able to work and understand various machine learning predictive models. To do predictions, first, the data needed to be transformed hence EDA and feature engineering were needed. Many of the features needed to be transformed to get the best prediction accuracy. After preparing the data then I created four different models and compared them. LGBMClassifier turned out to be the best predictor and also the fastest. I tried hyper-tuning but didn't get good results. More analysis can be done on hyper tuning to improve the model.