"Convolve Epoch 2 Documentation"

The following contains a detailed explanation of our approach towards Liability account monitoring for Money Mules

EDA

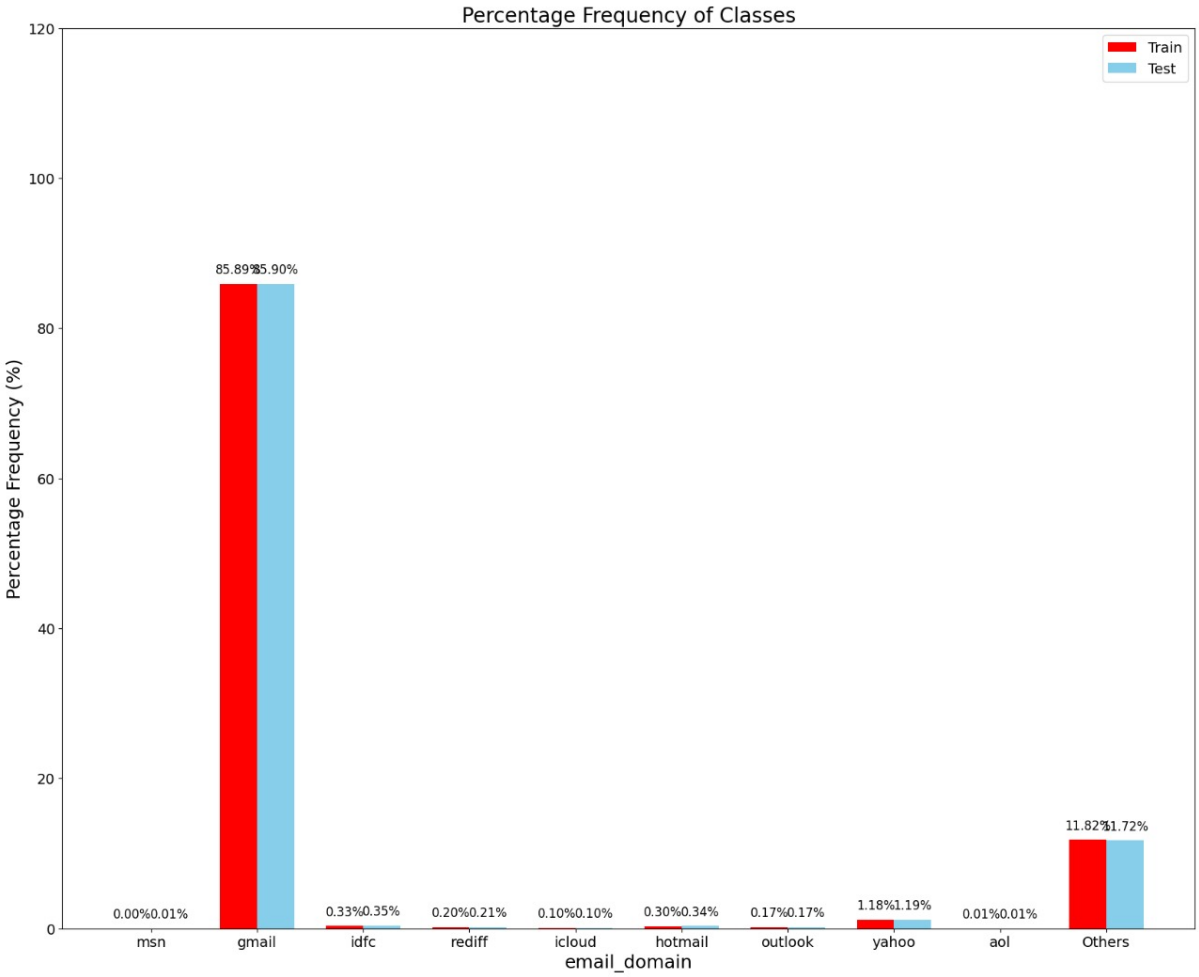
Exploratory Data Analysis plays a key role in understanding the dataset, identifying key features and filtering the ones that aren't required, understanding correlations between input features, or the lack thereof, handling outliers, missing values and datatype manipulations so as to make useful insights using ML models.

These are our key comprehensions from EDA

1. There are columns with only a single unique value and missing values represented by NaNs. We see that such columns do not contribute much to the target column and neither do they carry any information. So we drop these columns.

2. The account opening date column, represented as a string in a converted csv is extracted to give the day, month, year and day of the week on which the account was opened. This helps us extract useful information for clustering frauds based on account opening date.

3.There are categorical columns where there exist unexpected entries like "ZZ" in between numerical entries that might be representative of some kind of categories. We deal with this by replacing such entities with nan, so that they can be dealt with later./ with another number to treat it as another class.



4. We see for such columns that there is one class with a very high frequency of one class, while other classes lie in a minority. We see that one hot encoding such columns to 1 if it is that class and 0 if others improves performance, helps capture this class imbalance and tackles the issue of unseen data.

5. We assume that columns that contain a high percentage of missing values do not carry much information. So we drop those columns that have higher than 90% of NaN values.

6. The correlation matrix helps us visualise the relationships between multiple input features and those between the input features and the target column. Besides the usual left diagonal in the heatmap which represents correlation of a feature with itself is = 1, we also noticed high correlations between some features. This might be a representation of duplicates, i.e, when an input feature is a linear transformation of another, the correlation between these features will be

one. We drop such duplicates.

7. At the same time we also notice, with our custom one hot encoding of country code, it becomes a duplicate of demog_42, which implies demog_42 represents whether the transaction is Indian or foreign.

8. We propose a new method based on PCA to identify features that pose no meaning. Sum of Correlations. If the sum of correlations of a column with remaining columns is less than a certain threshold, we drop such columns as they do not carry much information.

9. Bool value handling:

If columns contain boolean values, we convert true and false observations to ones and zeros respectively.

10. Encoding Categorical columns
We see that there are many categorical columns. We one hot encode those columns that have less than 8 unique classes, and use a Target Encoder for the other categorical columns to handle sparse matrices and the dreaded curse of dimensionality.

One hot encoder: One Hot Encoding is a technique to encode categorical data to numerical ones. One hot encoding creates new (binary) columns, indicating the presence of each possible value from the original data.For eg.

Animal	Age
Cat	3.2
Dog	4.3
Parrot	12.1
Cat	5.1

Animal_Cat	Animal_Dog	Animal_Parrot	Age
1	0	0	3.2
0	1	0	4.3
0	0	1	12.1
1	0	0	5.1

Target Encoder: Target encoding replaces a categorical value by a blend of the probability (or expected value) of the target given the category with the target probability (or expected value) over the entire training set.

$$encoding = \frac{\text{Count of target 1}}{\text{Total occurrences}}$$

https://towardsdatascience.com/dealing-with-categorical-variables-by-using-target-encoder-a0f1733a4c69

11. We finally use a Standard Scaler to normalise all the columns of the modified df.

Standard Scaler: StandardScaler() will normalise the features i.e. each column of X, INDIVIDUALLY, so that each column/feature/variable will have mu = 0 and $\sigma = 1$.

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} (x_i)$$

and standard deviation:

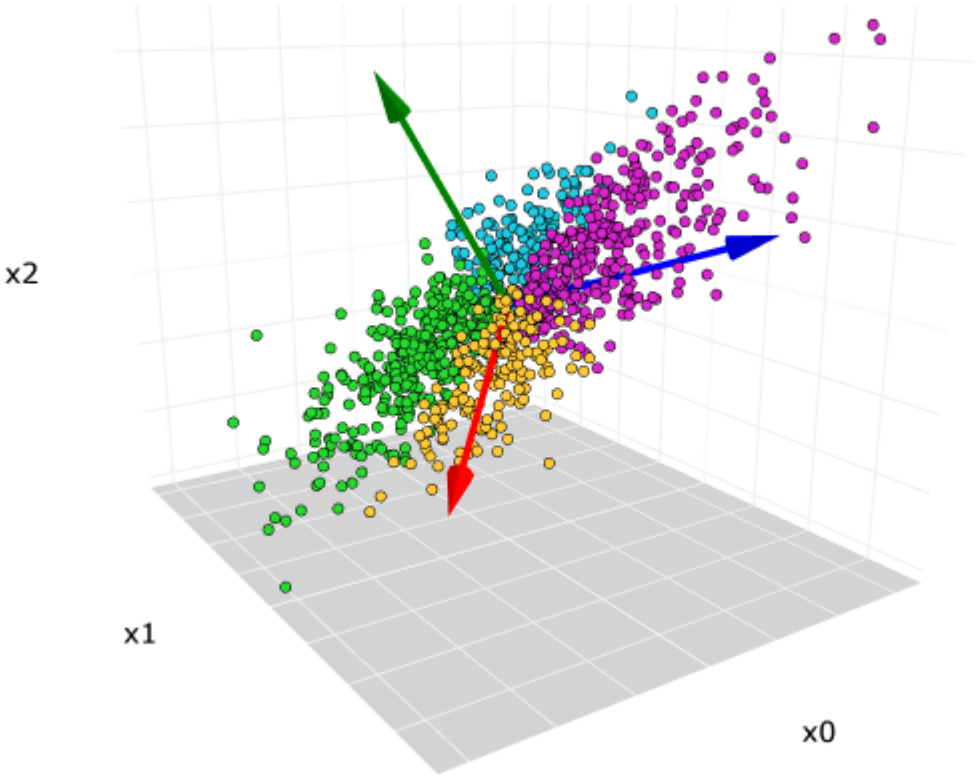
$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}$$

https://stackoverflow.com/questions/40758562/can-anyone-explain-me-standardscaler

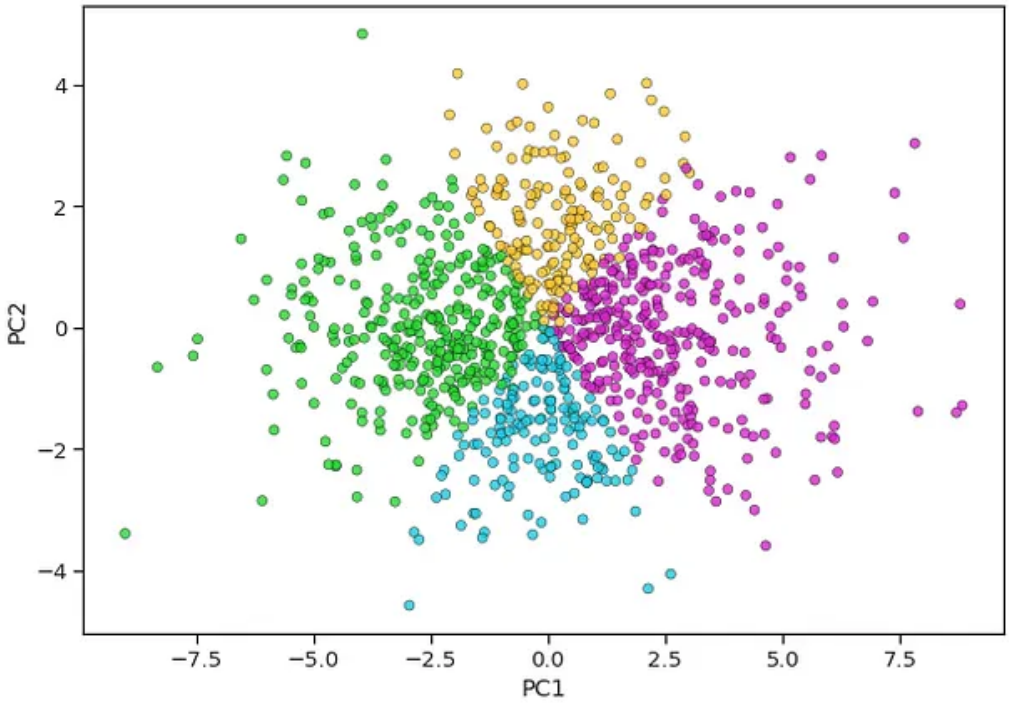
PCA:

Principal Component Analysis, commonly referred to as PCA, is a dimensionality reduction algorithm commonly used to decrease the number of input features while capturing most of the information effectively.

PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.



For the following distribution, we see that most data can be represented using 2 dimensions, i.e, a plane passing through it as seen below.



The PCA components in this case become the primary axes defining that plane about which the variance is least.

https://youtu.be/TJdH6rPA-TI?si=1XDHFUoWYFW3XyQD

Using the above PCA components, we try the following 4 classifiers:

RFC Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3354727

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Gradient Boosting Gradient boosting works by ensembling many decision trees in order to perform regression or classification. However, unlike random forest, gradient boosting grows trees sequentially, iteratively growing trees based on the residuals of the previous tree. Doing so allows gradient boosting to focus on particularly tricky observations and yields an extraordinarily powerful ensemble of trees.

XGBC XGBoost is an implementation of gradient-boosting decision trees. XGBoost stands for extreme gradient Boosting. It is an ensemble learning

method that combines the predictions of multiple weak models to produce a stronger prediction.

https://github.com/dmlc/xgboost/blob/82d846bbeb83c652a0b1dff0e3519e67569c4a3d/doc/index.rst

LGBMC LightGBM uses a novel technique called Gradient-Based One-Side Sampling (GOSS). It is a technique where a subset of the data is selected to use when training a gradient boosting model. It works by first sorting the training data by the gradients of the loss function with respect to the current model, and then selecting a subset of the data based on the magnitude of these gradients.

https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

https://github.com/microsoft/LightGBM

HGBC It is a machine learning technique designed to accelerate the training of gradient boosted trees. It achieves this by leveraging histograms and integer-based data structures, rather than relying on sorted continuous values which is done in traditional gradient boosting algorithms.

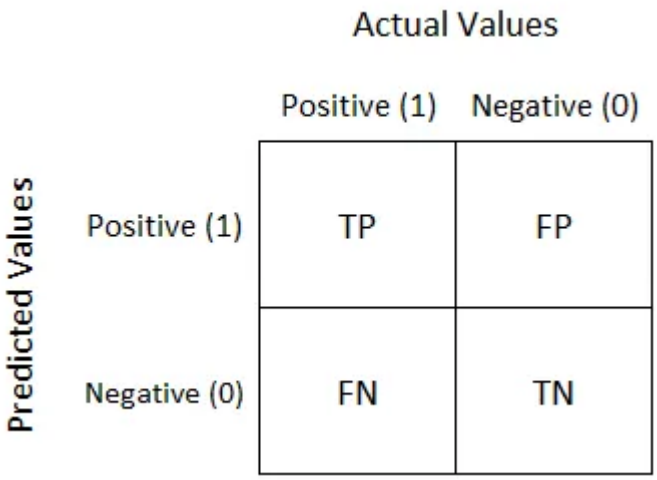
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html

Dataset Manipulation

A balanced dataset, in a binary classification problem, contains a 50:50 split of both the target columns. Balancing becomes necessary in case of anomaly detection, where there is a very small number of instances of one class. It makes training a model easier because it helps prevent the model from becoming biased towards one class.

Metrics of Measurement

The Confusion Matrix



What is True Positive, False Positive, True Negative, False Negative

True Positive is when a model predicts target = 1 correctly.

False Positive is when a model predicts target = 0 incorrectly.

True Negative is when a model predicts target = 0 correctly.

False Negative is when a model predicts target = 1 incorrectly.

Accuracy : The number of classes predicted correctly by the model.

$$\textit{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

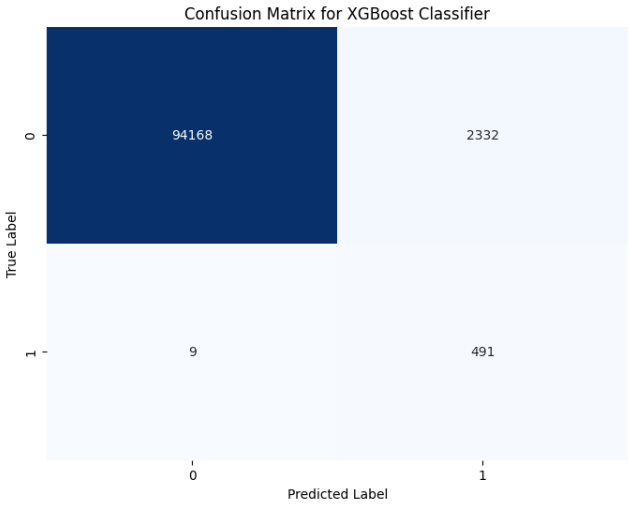
$$\textbf{F - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

Why F-1Score?

Accuracy is used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial.

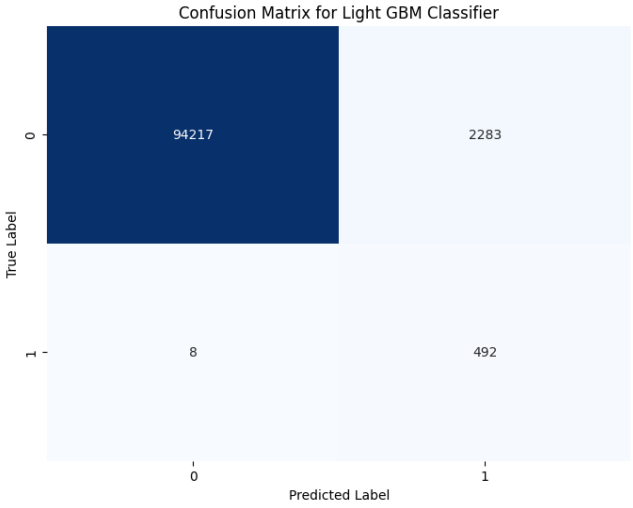
Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in the above case.

For balanced dataset



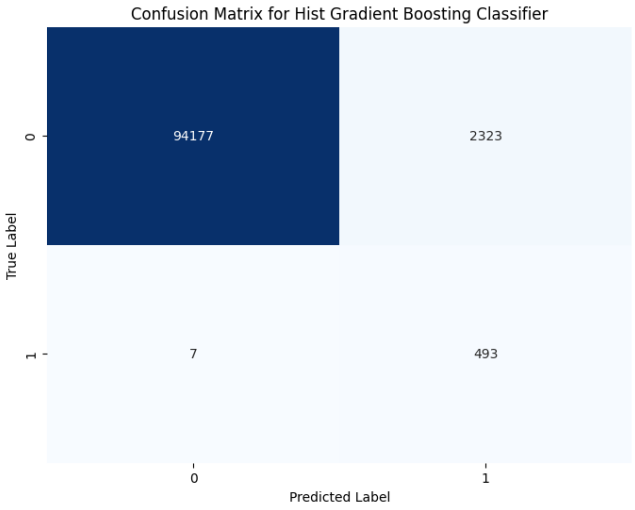
F1 Score for XGBoost Classifier: 0.9842

Accuracy for XGBoost Classifier: 0.9759



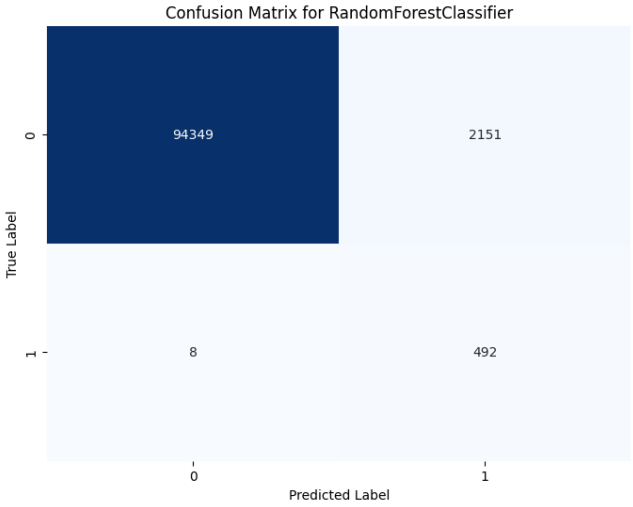
F1 Score for Light GBM Classifier: 0.9844

Accuracy for Light GBM Classifier: 0.9764



F1 Score for Hist Gradient Boosting Classifier: 0.9842

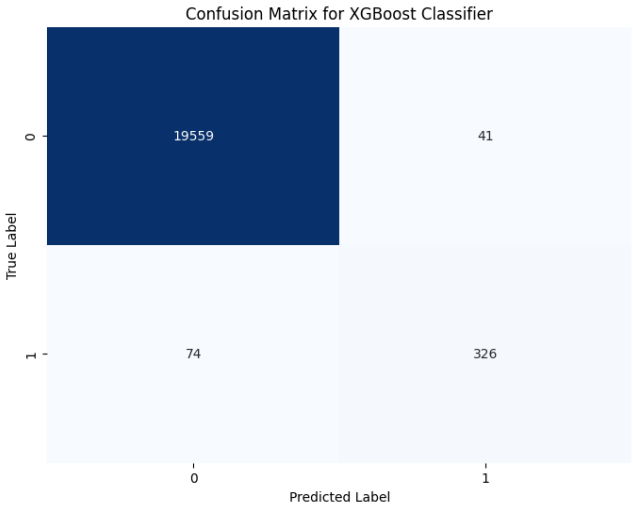
Accuracy for Hist Gradient Boosting Classifier: 0.9760



F1 Score for RandomForestClassifier: 0.9852

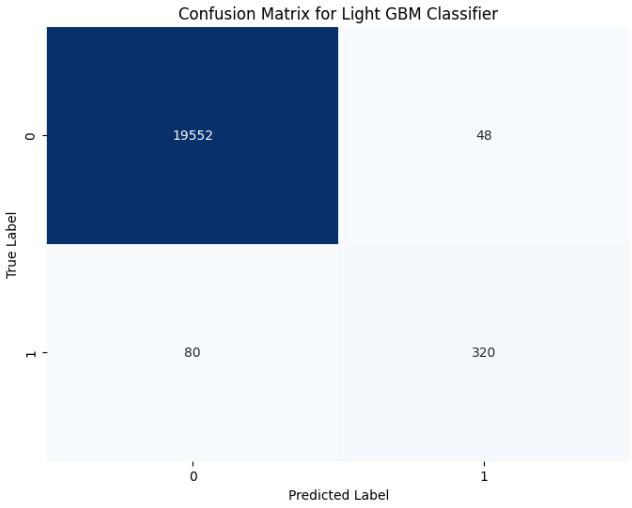
Accuracy for RandomForestClassifier: 0.9777

For unbalanced model:



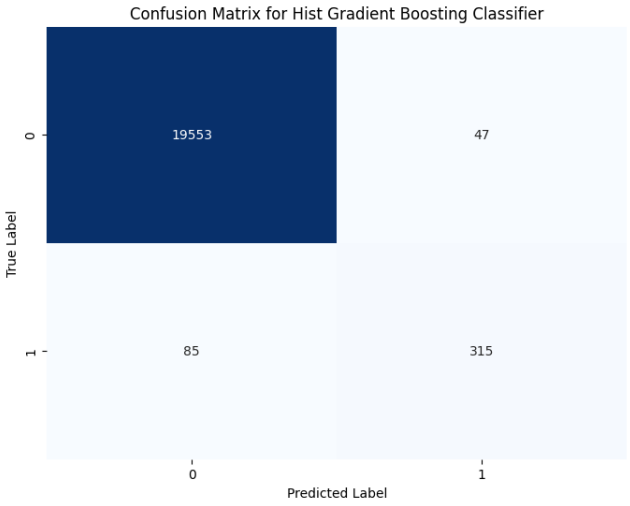
F1 Score for XGBoost Classifier: 0.9941

Accuracy for XGBoost Classifier: 0.9942



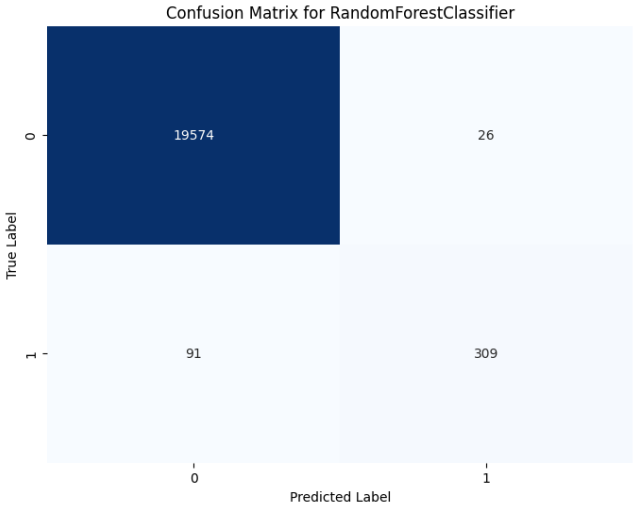
F1 Score for Light GBM Classifier: 0.9935

Accuracy for Light GBM Classifier: 0.9936



F1 Score for Hist Gradient Boosting Classifier: 0.9932

Accuracy for Hist Gradient Boosting Classifier: 0.9934



F1 Score for RandomForestClassifier: 0.9939

Accuracy for RandomForestClassifier: 0.9941

Using the metrics mentioned above, we see that **Random Forest Classifier** performs best on a balanced dataset and is effectively able to detect mules, while **XGBoost Classifier** is effectively able to capture the intricacies of details to classify transactions that are not frauds.

We propose the final model as follows:

Let BM denote the Random forest classifier fitted on a balanced dataset, and UBM denote the XGBoost classifier fitted on an unbalanced dataset.

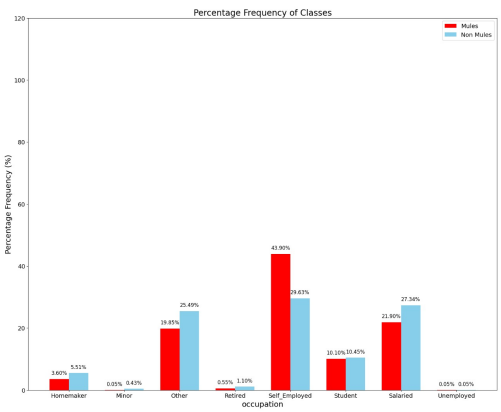
If both BM and UBM predict 1, target = 1

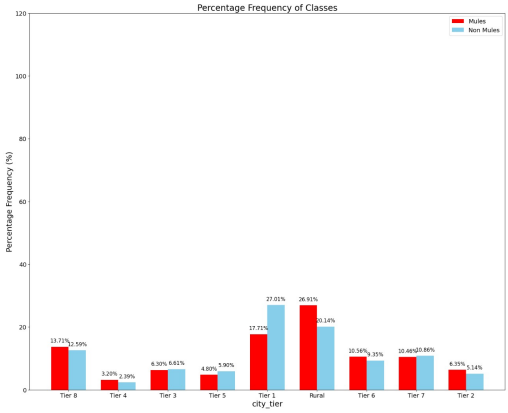
If both BM and UBM predict 0, target = 0

If predictions of BM and UBM differ, target = 1/0 depending on the confidence with which either model makes a prediction.

However, for the probability column, we return the probability with which the selected model predicts target class=1, as a convention mentioned.

Insights drawn

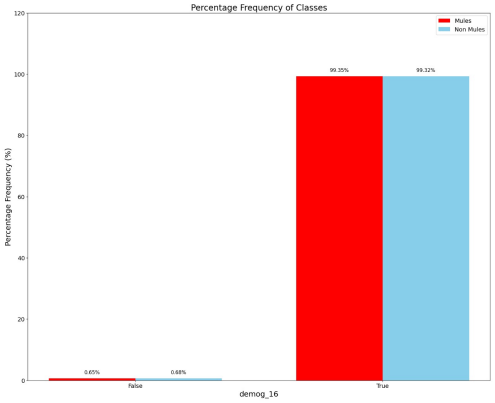


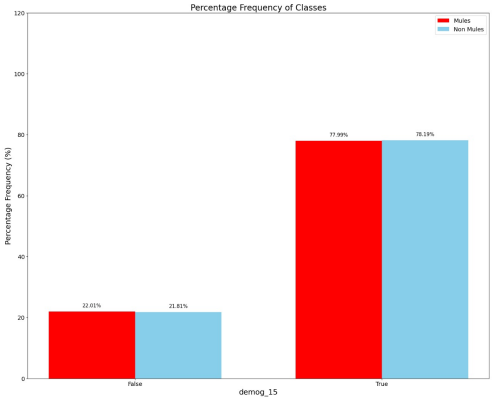


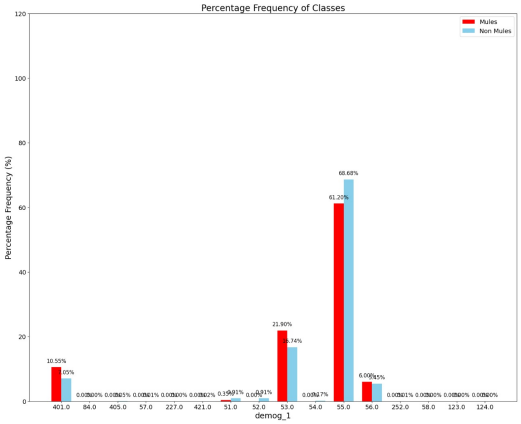
43.9% of the mules are self employed, while 21.9% of the mules are salaried and 19.85% of the mules are working other jobs.

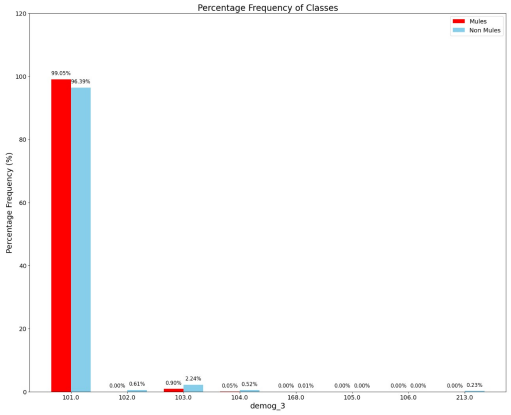
26.91% of the mules live in rural areas, while 17.71% of the mules stay in Tier 1 cities and 13.71% of the mules occupy Tier 8 cities.

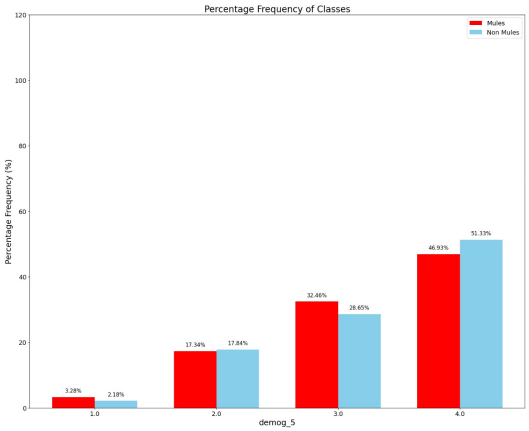
Mules tend to follow the same trends as non mules in most categories,

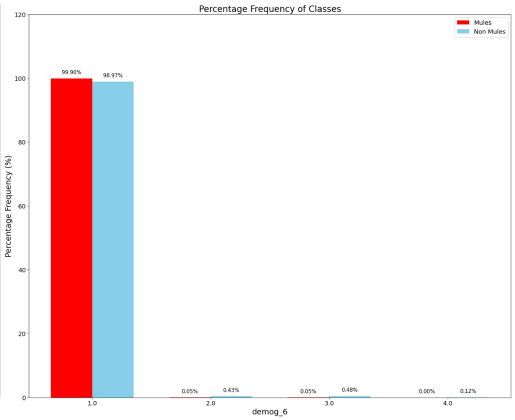


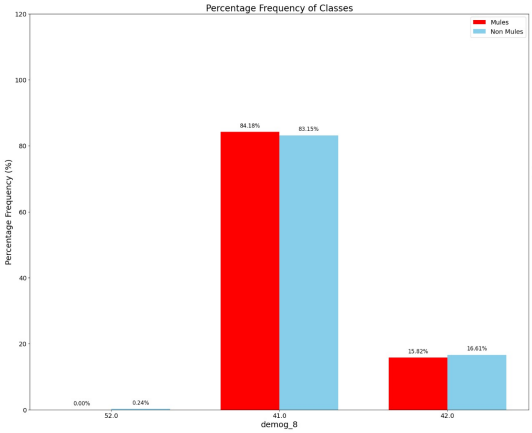


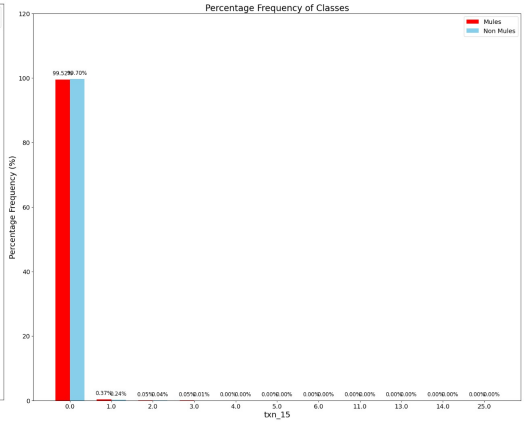

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
demog_1
401.0
84.0
405.0
57.0
227.0
421.0
51.0
52.0
53.0
54.0
55.0
56.0
252.0
58.0
123.0
124.0
10.55%
7.05%
21.90%
16.74%
61.20%
68.68%
6.00%
5.45%

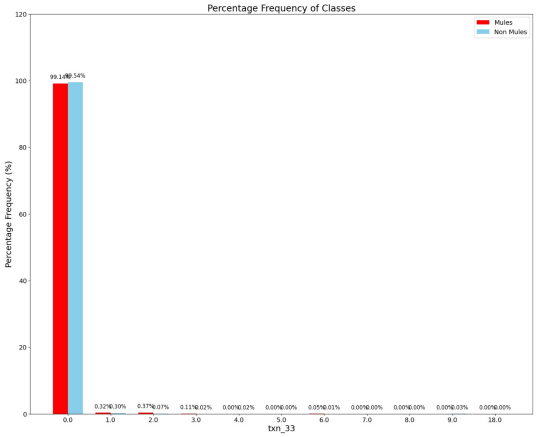

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
demog_3
101.0
102.0
103.0
104.0
168.0
105.0
106.0
213.0
99.05%
94.39%
0.00%
0.61%
0.90%
2.24%
0.05%
0.52%
0.00%
0.01%
0.00%
0.00%
0.00%
0.00%
0.00%
0.23%

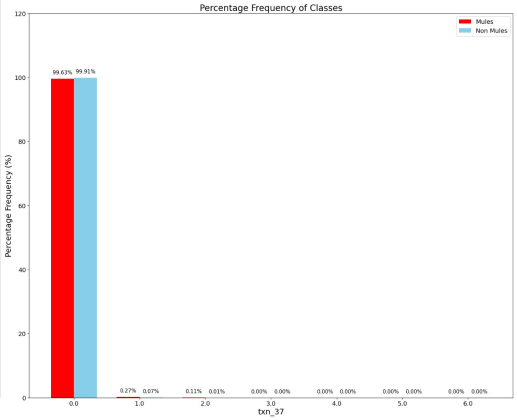

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
demog_5
1.0
2.0
3.0
4.0
3.28%
2.18%
17.34%
17.84%
32.46%
28.65%
46.93%
51.33%

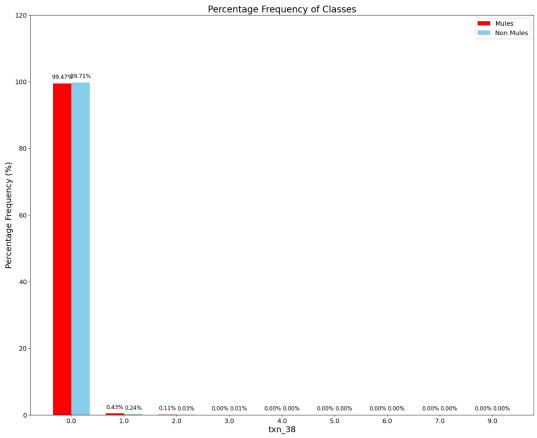

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
demog_6
1.0
2.0
3.0
4.0
99.90%
98.97%
0.05%
0.43%
0.05%
0.48%
0.00%
0.12%

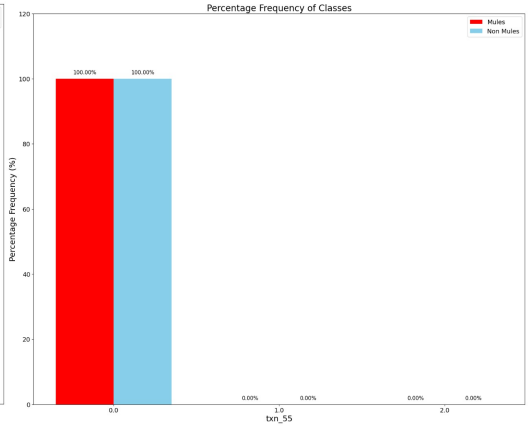

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
demog_8
52.0
41.0
42.0
0.00%
0.24%
84.18%
83.15%
15.82%
16.61%

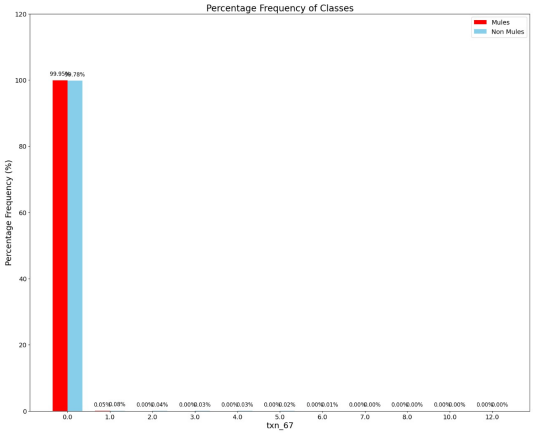

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_15
0.0
1.0
2.0
3.0
4.0
5.0
6.0
11.0
13.0
14.0
25.0
99.57%
99.70%
0.37%
0.24%
0.05%
0.04%
0.05%
0.01%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%
0.00%


Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_33


Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_37


Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_38

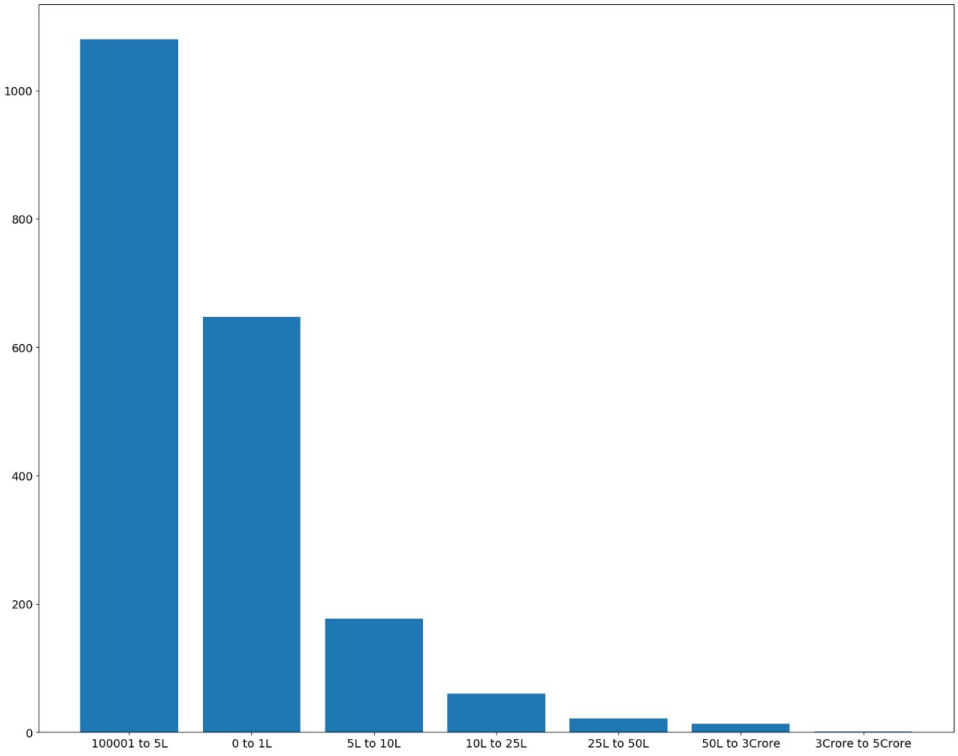

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_55

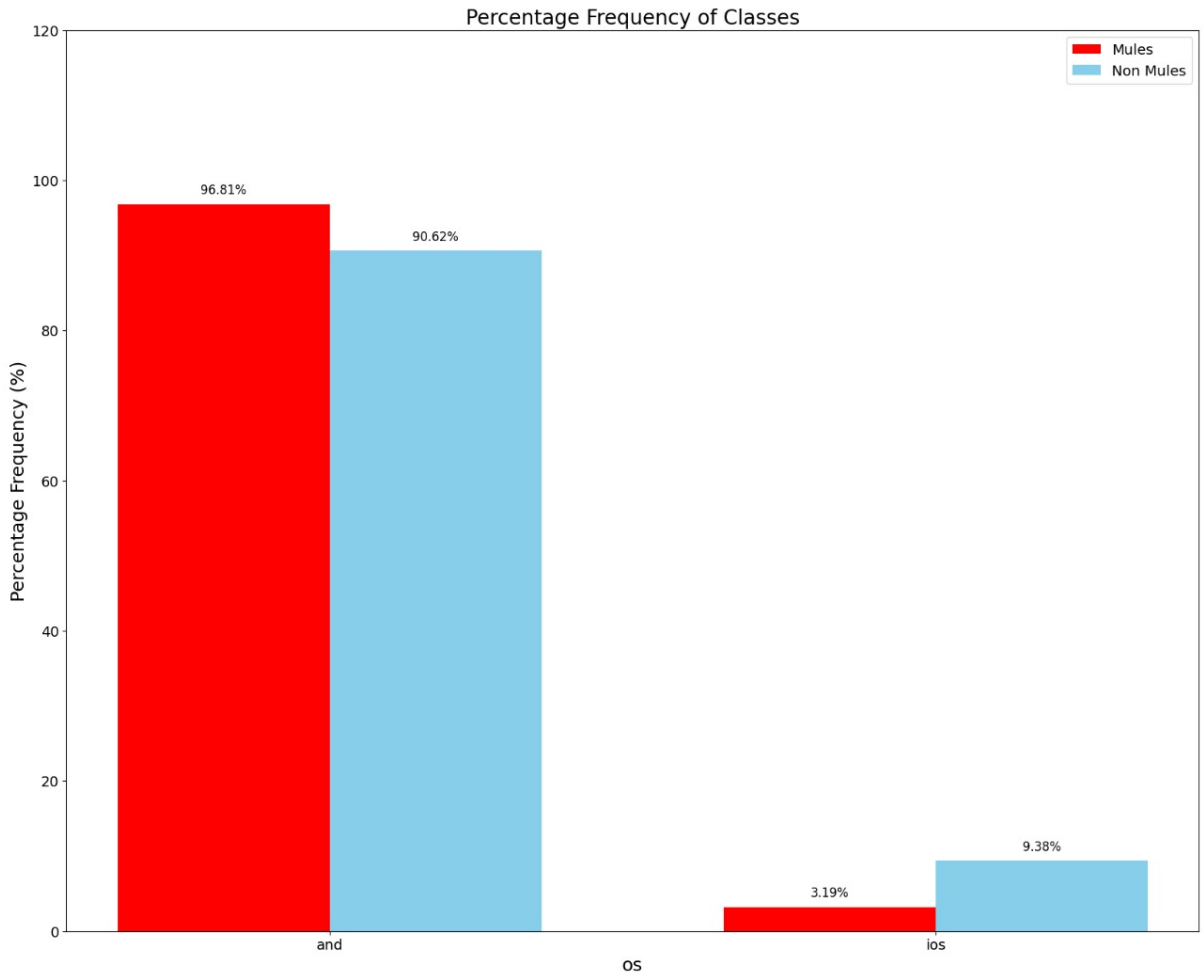

Percentage Frequency of Classes
Mules
Non Mules
Percentage Frequency (%)
txn_67

The following input features affect PCA components the most

txn_11 7.179229 txn_75 6.991520 txn_76 6.981462 txn_12 6.869490 txn_73 6.664859

The following bar graph represents the income bracket of mules





Most mules use android based devices

% of mules in train data, % of mules predicted.

The training data has 2K mules in 100K rows, i.e, 2% of the accounts are mules.

Applying the algorithm explained above, we conclude that the validation data on which we are predicting has around 2.33% of mule accounts.