Convolve 2.0 : Epoch 2

Team Tensile Tensors

January 2024

1 Introduction

The banking sector plays a crucial role in the economy; however, this vital role also makes banks susceptible to fraudulent activities, and one prevalent form of financial fraud involves the creation and operation of fraudulent bank accounts. Detecting and preventing fraudulent bank accounts is of utmost importance, as it helps mitigate financial losses, protects customers, and maintains the integrity of the banking system.
This paper examines the effectiveness and implementation of different machine learning techniques to develop an automated transaction monitor to identify fraudulent "mule" bank accounts.

2 Methodology

2.1 Data Pre-Processing

We pre-process the data and engineer useful features to clean, transform and reduce the given data for suitable analysis and model training.

2.1.1 Data Splitting

The data is subject to 60/40 train-test split. All the data pre-processing and feature engineering steps are done considering the values of the train set only in order to make sure data leakage doesn't occur.
However the same data engineering is done to the test set, considering the test set to be a black box (no value from the test set is considered for any data engineering step)
The validation data given is also subject to the same pre-processing before we forecast the target values corresponding to each of it's rows.

2.1.2 Data Cleaning

- **Imputing NaNs:** NaN values in every column of the training set were replaced by the average value of the corresponding column, while for the

test set, it was replaced by the median of the corresponding column of the training data.

- **K-beans discretization:** Columns with continuous values were discretized into different classes, as buckets and label-encode each bucket. This makes it easier for the model to train itself on the data and find hidden statistical relationships.
- **Label Encoding:** Text labels or categorical variables are converted into numerical representations. Each unique label is assigned a unique integer, thereby encoding the categorical variables into numerical format.

2.1.3 Data Reduction

- **Correlation with other columns:** The correlation of each column with every other column was checked. All the columns with high correlation($>$ 0.85) with the column being analyzed were eliminated.
- **Redundant Columns:** Certain columns had $>$95% of values to be NaN. These columns were determined to be not useful for unbiased predictions.
- **Unbalanced columns:** Some columns were found to have more 99% of their data to be of one class with no distinction in their relationship with the Target column. Such columns were decided to be of no use as they did not possess any distinct predictive power.
- **Chi square test:**
  - The **Chi-Square test** is a statistical method used to determine whether there is a significant association between categorical variables. Chi-square test can be used to determine whether two categorical variables are independent of each other or if there is a significant association between them.
  - The Chi-Square test was used to eliminate several features with little association with the target feature.
- **Recursive Feature Selection:**
  - **Recursive Feature Selection (RFE):** is a feature selection technique commonly used in machine learning. Its purpose is to select the most relevant features from a dataset by recursively considering smaller and smaller sets of features. By removing irrelevant or redundant features, RFE helps to reduce overfitting and noise in the data, leading to better generalization performance.
  - Several features were eliminated using RFE.

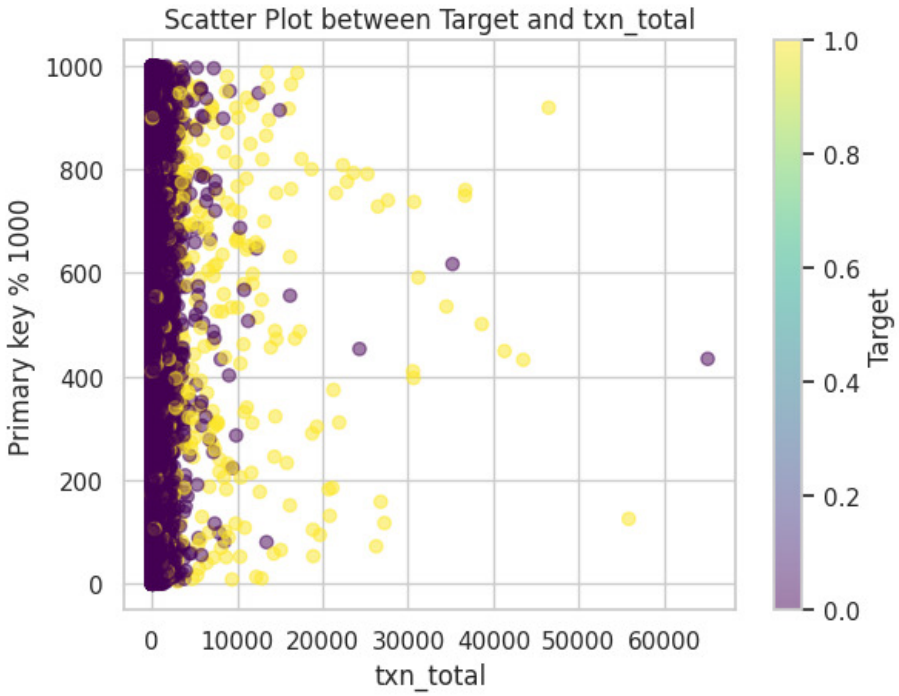

Figure 1: Plot showing total transaction of all members (Target color-coded)

2.1.4 Feature Engineering

Given the significant class imbalance in the provided training data, with a ratio of 1:49 between the minority and majority classes, it is imperative to employ various methods and models to ensure adequate sampling of the data. Failure to address this imbalance could lead to pronounced bias in the model predictions, favoring the majority class excessively. Hence, a strategic approach is devised to mitigate this issue and to promote a more equitable representation of both classes in the training dataset. In addition, conventional and unique methods are used to handle categorical and noisy data.

- **SMOTE:** Synthetic Minority Over-sampling Technique, is used to mitigate class imbalance in datasets encountered in machine learning tasks. Class imbalance can lead to biased models favouring the majority class and poor predictive performance for the minority class.
- **RandomOverSampler:** The Random Over Sampler is a method used to address the issue of class imbalance in datasets. Like SMOTE, it aims to rectify situations where one class is significantly underrepresented by artificially increasing the number of instances in the minority class.
- **RandomUnderSampler:** Random Under Sampler is used to address class imbalance in datasets as well, similar to the aforementioned methods.
- **Label Encoding:** Text labels or categorical variables are converted into numerical representations. Each unique label is assigned a unique integer, thereby encoding the categorical variables into numerical format.
- **Additive Noise Reduction:** Based on our analysis, we found a lot of columns to be a noisy version of themselves, while some share an almost perfect linear correlation. We used different visualization and statistical methods to back our findings. The addition of these columns helps in reducing their idiosyncratic noise, assuming the noise is Gaussian in nature,
- **Transaction Sum:** Based on our observations and analysis, we found that the Target variable does not share any significant relationship with each transaction value but rather quite a strong one with the total sum of transactions made in the given time window (ref. fig 1).
- **NaN counts:** Based on our analysis and testing, we found that the number of NaN values in columns containing individual transaction data shows a significant relationship towards the value of the Target variable. Our research found that mule accounts are engaged in much more frequent transactions than non-mule accounts.
  Testing our initial hypothesis on the training set confirms our hypothesis as **28%** of Non-Mules engage in frequent transactions (NaN count=0), while **77%** of the Mules are engaged in frequent transactions (NaN count=0)

- **Principal Component Analysis (PCA):** Principal Component Analysis (PCA) is a widely used technique in dimensionality reduction and data visualization. PCA transforms high-dimensional data into a lower-dimensional space while preserving as much of the original information as possible. It achieves this by identifying the directions, or principal components, along which the data varies the most.

All these methods are judiciously used to ensure the dataset is well-engineered for the models to perform the best.

2.2 Model Training

We have trained and analyzed the performance of the following machine-learning methods/models:

- Random Forest
- XGBoost
- CatBoost
- K - Nearest Neighbours
- Fully Connected Neural Network (MLP)

We use Bagging (an ensemble method) with each of the aforementioned machine learning models to further improve upon their performance as a significant class imbalance exists in the model (1:50) and there exists a lot of noise and redundancy in the dataset.

2.3 Model Selection

Finally, we compare the following model evaluation metrics with respect to the test set for all the models to select the best-performing one for our use:

- F1 Score
- Balanced Accuracy
- ROC-AUC

2.4 Predictions for Validation Data

Using our best performing trained model, we predict the probability of an account being a mule for all the accounts (rows) in the validation dataset.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 2: The formulation for the F1 score

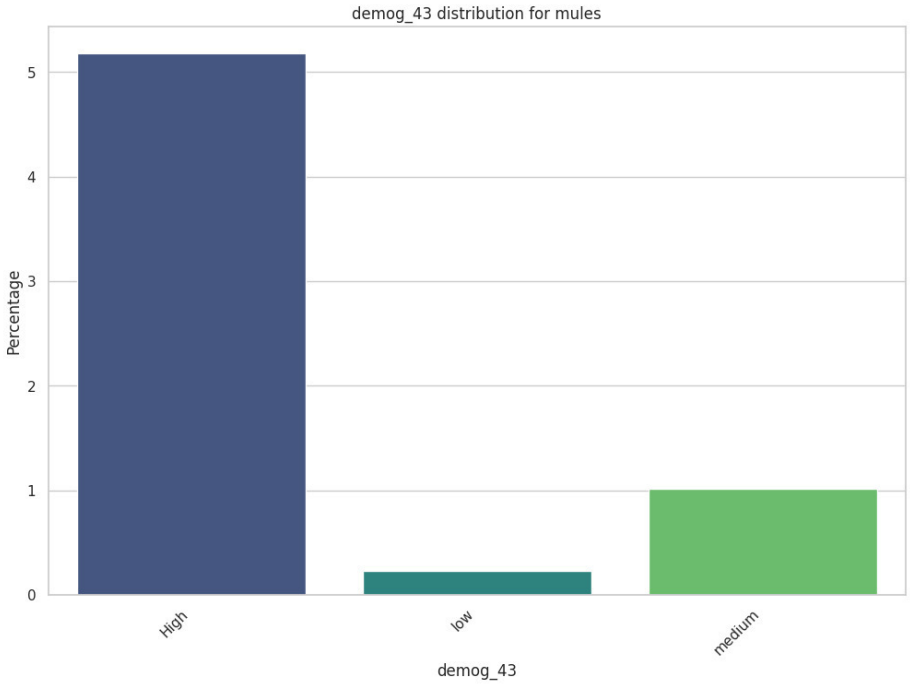

Figure 3: demog_43 distribution for Mule accounts

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 4: The formulation for Balanced Accuracy

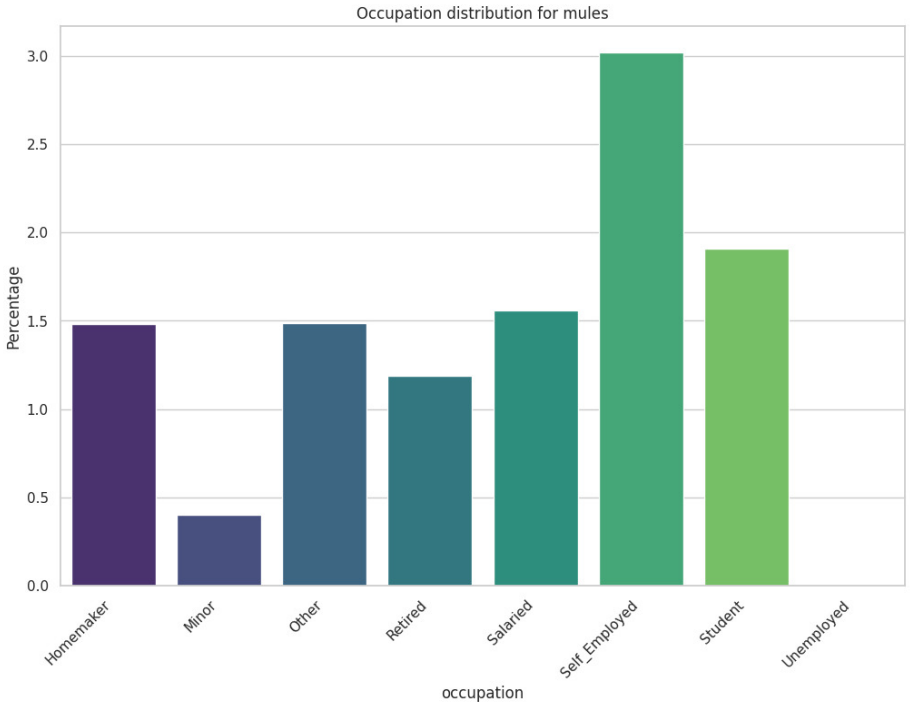

Figure 5: Distribution of Occupation (for Mules)

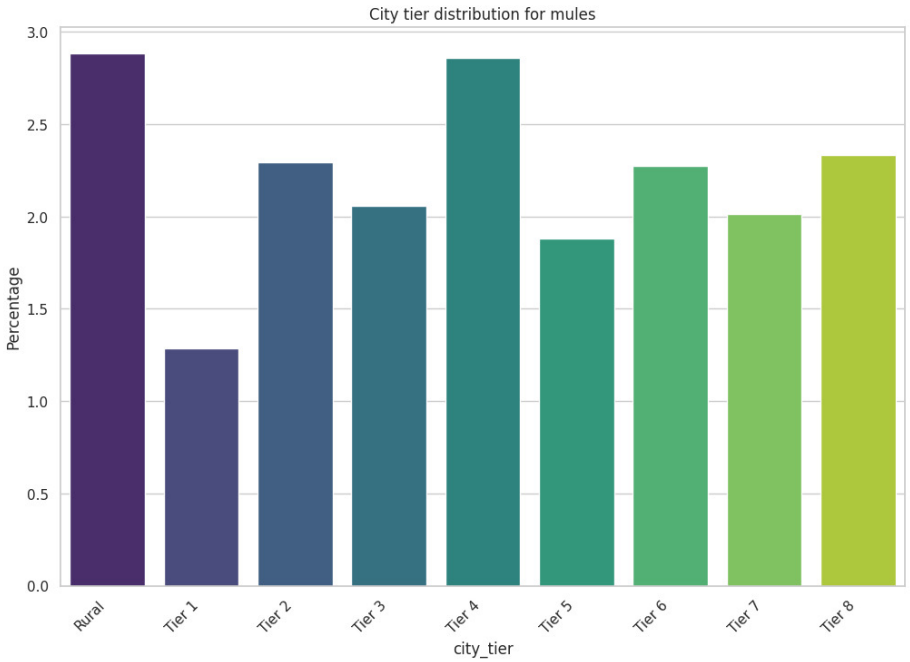

Figure 6: Distribution of City-tier (for Mules)

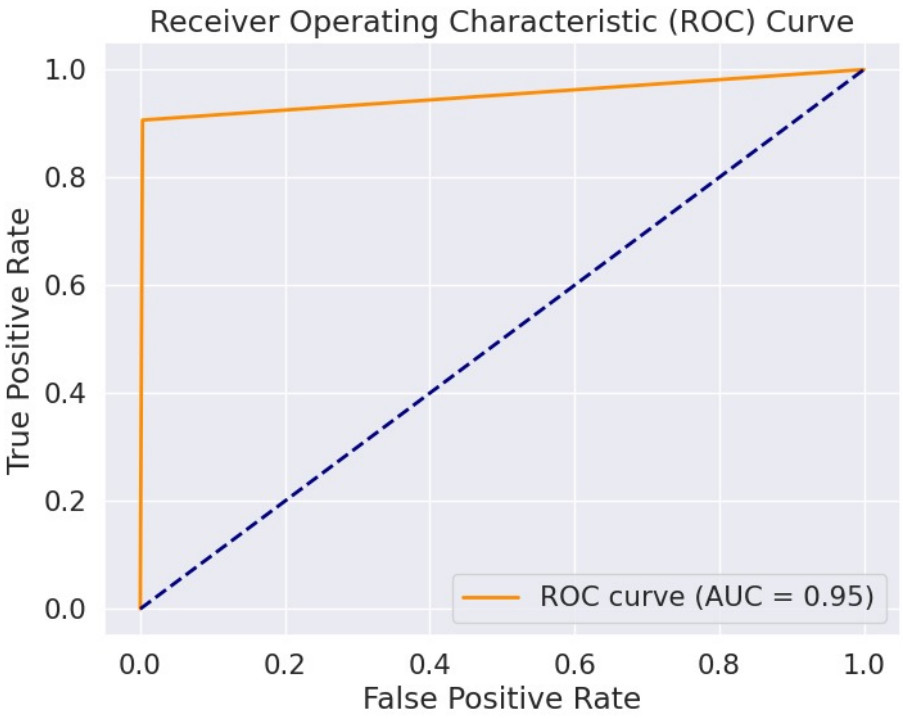

Figure 7: Receiver Operating Characteristic (ROC) Curve

Model (with Bagging)	F1 Score	Balanced Accuracy	ROC - AUC
CatBoost	0.8794	**0.9497**	**0.9982**
Random Forest	**0.8812**	0.9485	0.9972
XGBoost	0.8747	0.9478	0.9983
Fully Connected Neural Network	0.8730	0.9264	0.9948
K-Nearest Neighbours	0.7140	0.9545	0.9712

3 Results

3.1 Performance of Models

3.2 Model Evaluation and Selection

CatBoost with Bagging (bootstrap aggregation) is the best performing model as it has the highest values for:

- Balanced Accuracy
- ROC - AUC

However, Random Forest with Bagging has very similar scores for the aforementioned evaluation metrics, apart from marginally outperforming the CatBoost + Bagging model according to the F1 score.

4 Inference and Discussion

Among all the models considered, Bootstrap Aggregation (Bagging) of CatBoost gave us the best results with the model evaluation parameters:

- **F1 Score** = 0.8794
- **Balanced Accuracy** = 0.9497
- **ROC - AUC** = 0.9982

The same model was used to generate the prediction values of the validation set.

4.1 Interesting Observations

- Mule accounts have relatively higher volume of transactions (Total transactions)

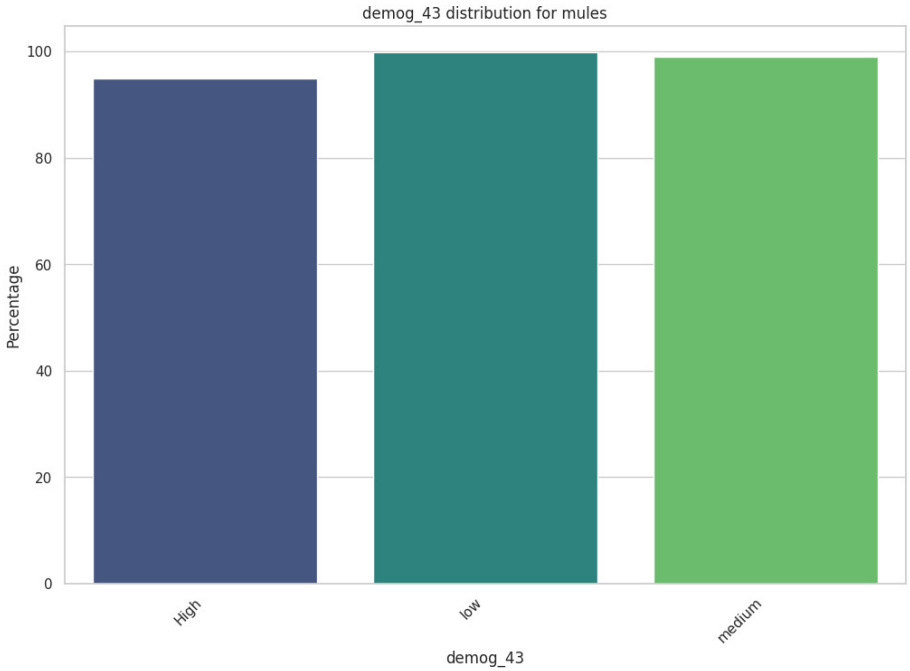

Figure 8: demog_43 distribution for Non-Mule accounts

- The accounts with demography apart from India have very low correlation with the target; that is, non-Indians are less likely to be mules
- Demographic features show bad correlation with the target, however they improve the model indicating the presence of a non-linear relationship between the two.
- Mule accounts have a much lower count of NaN values amongst the transactions (**23%**) compared to non-mule accounts (**72%**)
- Most of the mule accounts have **'Self-employed'** as their occupation.
- **demog_43** has three classes: **high**, **medium** and **low** - which correspond very well to the likelihood of an account being a mole