**CONVOLVE 2.0 DOCUMENTATION**

**INTRODUCTION**

**Problem Statement and Objective:**

The problem at hand is to develop a framework for Bank A to proactively monitor transactions done by customers through their Savings Accounts in order to identify potential money mules. Money mules are individuals who exploit online banking facilities to defraud others by convincing them to transfer money to accounts opened by the mules. The objective of the project is to use machine learning algorithms to predict the probability that a given account is a mule based on various account-level attributes, demographic information, transaction history, and other relevant factors.

**Overview of the Approach:**

1. Understanding Data and EDA (Exploratory Data Analysis)
2. Preprocessing
3. Feature selection
4. Dimensionality Reduction
5. Model Selection and Training
6. Prediction on the Validation Data

**DATA DESCRIPTION**

The **development data** contains 1,00,000 savings account details with 178 feature columns consisting of-

Primary key – The index of the accounts

Target – 0 or 1 accounts identified as mules have target=1.

Account level attributes like account opening date

Demographic attributes of the customer (e.g. income, occupation, city tier etc.)

Transaction history of customer ( includes credits / debits of specific types over a period of time)

Other attributes like product holding with the Bank, app / web logins etc.

Some variable names have been provided. For other variables, demographic attributes start with “demog\_”, transaction attributes start with “txn\_” and the rest start with “others\_”.

The **validation data** consists of 50,000 savings account details with the same set of input variables but without “Target”.

The feature columns contained 147 columns with float values, 18 columns with int values, 12 with categorical or object entries and a datetime column.

**EXPLORATORY DATA ANALYSIS (EDA)**

We performed EDA on columns whose description was available namely, account opening date, country code, income, city tier, occupation, os and the email domains.

**Account Opening Date-**

We separated this column into 3 separate columns containing day, month and year. Then we analysed the distribution of mules with respect to the day of the week over a period of 4 months ( 2022-12-01 to 2023-03-31).

**Days-**

We observe that mule accounts openings were relatively less on weekends compared to the weekdays.

**Months-**

More mule accounts were opened in the month of December whereas March had the least number of mule account openings.

**Country Code-**

Since most of the userbase belonged to India,hence all of the mule accounts were found to be Indian.

**Income-**

Majority of the mule accounts had an income of less than 5 lacs.

**City Tier-**

It was observed that majority portion of the mules belonged to rural areas followed by the Tier 1 city.

**Occupation-**

Most of the mule accounts were opened by individuals who called themselves as self employed.

**OS-**

Almost all of the mule account holderswere Android Users.

**Email Domain-**

Most of the mule account users used gmail as their email domain.

**PREPROCESSING**

**Handling Missing Values:**

The dataset was found to have a lot of feature columns containing missing values.

We dropped the 4 columns with more than 95,000 empty values out of 1,00,000.

For the rest features, the empty values of columns having **categorical data** were filled with the **mode** of the data while those with **non-categorical data** were filled with their respective **mean**.

**Dropping Unnecessary Columns:**

We navigated through the features to get the idea of unique values in each column. It was found that 11 feature columns contained a **single unique value**. Since they would not contribute anything to the model they were dropped.

The account\_opening\_date feature column had a datetime dtype so we converted it to day month and year columns with dtype int. Since the month & year columns were minimally correlated with the target column, we dropped them.

We also checked for any **duplicate rows** but found none.

**Encoding Categorical Variables:**

Since the categorical features contained large number of unique values, thus we used **label encoder** instead of one hot encoder.

We now have 163 features in our dataset.

**FEATURE SELECTION**

Following the conventional **correlation matrix would not help us** in finding the correlation of the feature columns with the target column since we have quite an imbalanced dataset with only 2000 mules out of 1,00,000. So, we used **mutual information for feature selection** due to its ability to capture non-linear relationships, information gain for prediction, **robustness to imbalance**, and consideration of feature independence.

Further, we got a list of feature columns with **mutual information less than 0.001** that were 28 in number and were dropped.

**DIMENSIONALITY REDUCTION**

**Dimensionality Reduction:**

We used **Standard Scaler** to scale the data to unit variance and applied PCA**(Principal Component Analysis)** for dimensionality reduction. On setting a cumulative explained variance of 0.90, we get 68 principal components.

We plotted the correlation of all the 68 principal components with the target column as well as their mutual information values.

**MODELS USED FOR TRAINING**

**Splitting into training, testing and validation data:**

Using train\_test\_split, we split our development data into 20% testing data and 80% temporary training data which is further split into 60% training and 20% validation data.

**Model Selection:**

We used different binary classification models for training on the development data namely-

1. XGB Classifier
2. Random Forest Classifier
3. K-Nearest Neighbours (KNN)
4. Support Vector Machines (SVM)

**XGB Classifier-**

Known for its high performance and efficiency, it is capable of capturing complex relationships in the data as well as prevents overfitting due to built-in regularization techniques. It is particularly useful when dealing with high dimensional datasets as in our case.

**Random Forest Classifier-**

It is an ensemble learning method that combines multiple decision trees to make predictions. It can handle class imbalance well that is quite evident inour case where we have 2000 mules and 98000 non mule accounts. It can be fine-tuned using its hyperparameters.

**K-Nearest Neighbours-**

It makes predictions based on the majority class of its k-nearest neighbours. However, it might not perform that well for large and unbalanced datasets since it requires calculating distances between the query points and all training samples.

**Support Vector Machine-**

Effective in high-dimensional spaces, it works well with imbalanced data and aims to find the hyperplane that maximizes the margin between classes leading to a more robust and generalizable solution.

**METRICS USED FOR MODEL SELECTION**

**Classification Report:**

The main components of a classification report typically include precision, recall, F1-score, and support for each class.

**Precision-**

It measures the proportion of true positive predictions out of all positive predictions made by the model.

**Recall-**

Also known as sensitivity or true positive rate, it measures the proportion of true positive predictions out of all actual positives in the dataset.

**F1 Score-**

It is the harmonic mean of precision and recall, providing a single score that balances both measures.

**Support-**

Support is the number of actual occurrences of each class in the dataset, indicating how many samples are in each class.

**Average Precision Score:**

It is a metric used to evaluate the performance of binary classification models, particularly in scenarios where the class distribution is imbalanced. It measures the area under the precision-recall curve, which plots the trade-off between precision and recall for different thresholds of a classifier.

It is important in money mule account detection, where correctly identifying mules (recall) while minimizing false positives (precision) is crucial.

**Balanced Accuracy Score:**

It is calculated as the arithmetic mean of the recall (sensitivity) of each class, and it provides a balanced assessment of the model's ability to correctly predict both the majority and minority classes.

It is a useful metric for evaluating our model because it provides a balanced assessment of the model's performance across both classes, which is essential in our imbalanced dataset.

**Cohen Kappa Score:**

It is a statistic that measures inter-rater agreement for categorical items. It is generally used when there are two raters (or in the context of machine learning, two classifiers) who classify items into mutually exclusive categories.

It accounts for class imbalance, focuses on agreement beyond chance, considers the costs of different types of errors, and provides a single interpretable measure of performance.

**Cross Validation Score:**

This technique is used for assessing the performance of a predictive model. It involves splitting the dataset into multiple subsets, called folds, and then training the model on a subset of the data while using the remaining subsets for validation.

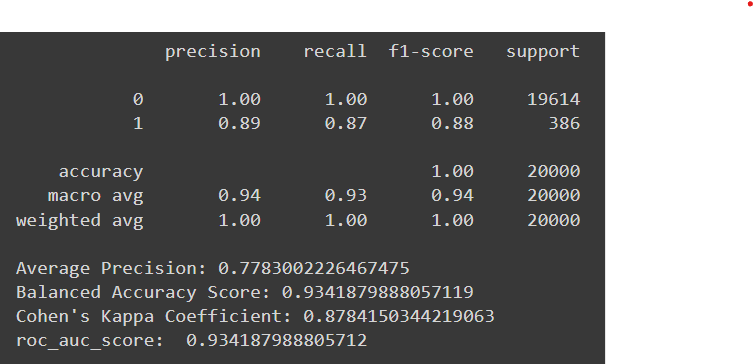
We used Stratified K-fold for splitting it into 5 folds. It helps in reducing the risk of overfitting.

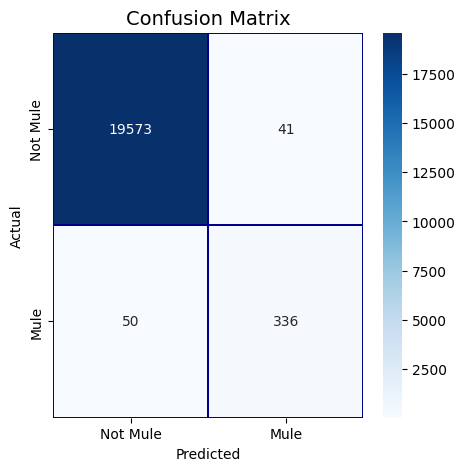
**ROC AUC Score:**

It measures the ability of a model to discriminate between positive and negative classes across different thresholds.

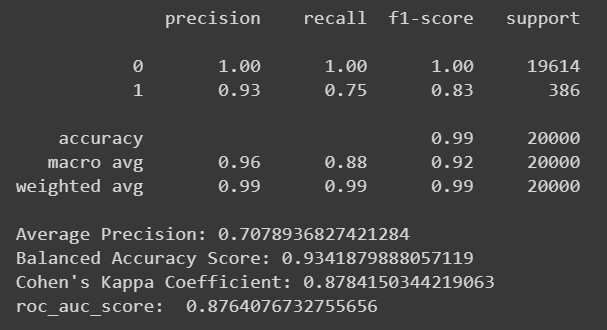
**MODEL ANALYSIS**

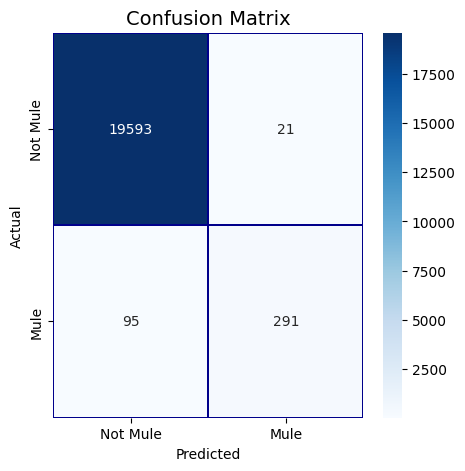
1. **XGB Classifier:**

****

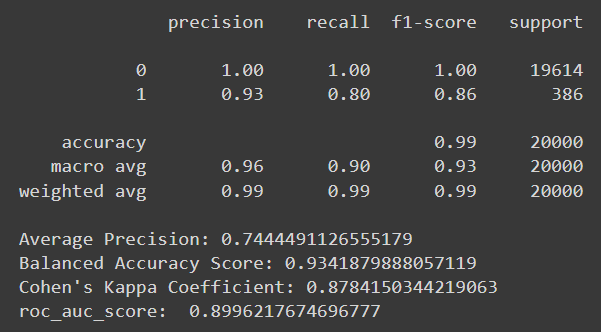


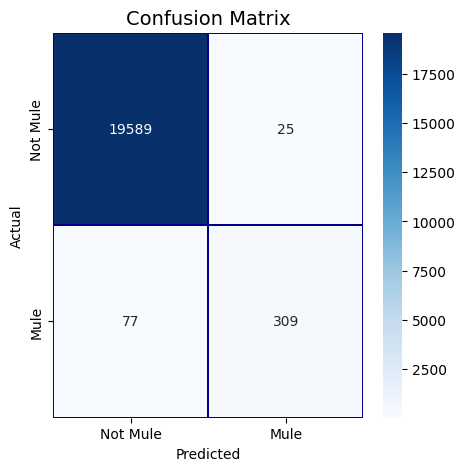
1. **Random Forest Classifier:**

****

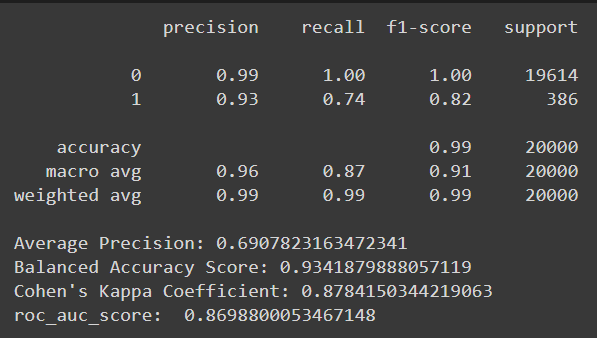


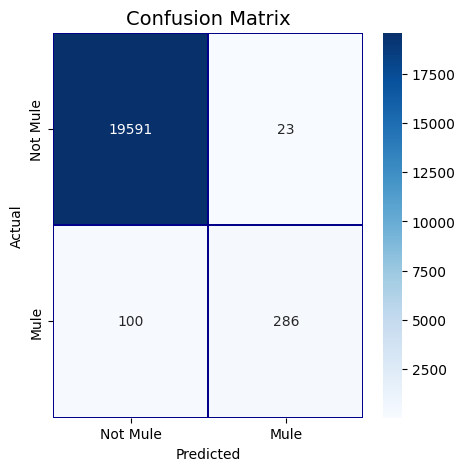
1. **K-Nearest Neighbours:**

****

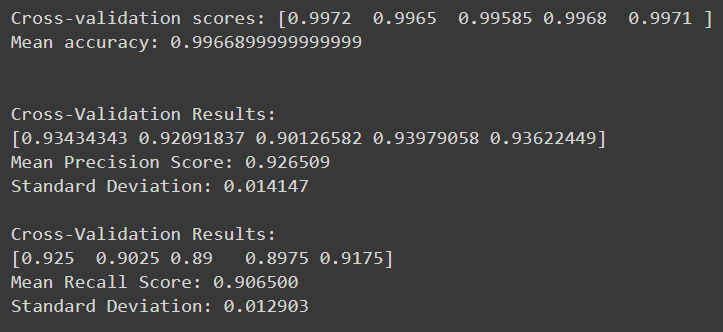


1. **Support Vector Machine:**

****



We observe that XGB Classifier performs the best on the training data. Therefore, we checked the cross validation score and plotted the learning curve for the xgb classifier model.



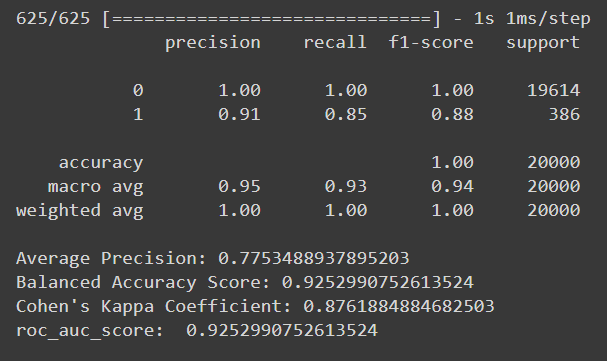
We also used grid search to find the best hyperparameters for our model.

Class weights were assigned to the different classes to address the imbalance and improve the performance of our model.

**DEEP LEARNING APPROACH**

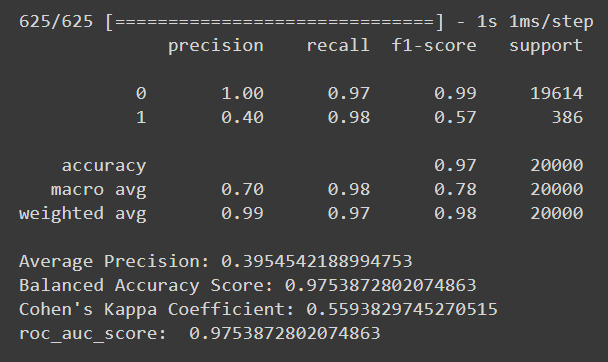
**Deep Neural Network-**

We created a sequential model with multiple dense layers with relu activation for the initial ones and sigmoid for the output layer along with a dropout layer to train on the training data.

****

**Undersampling-**

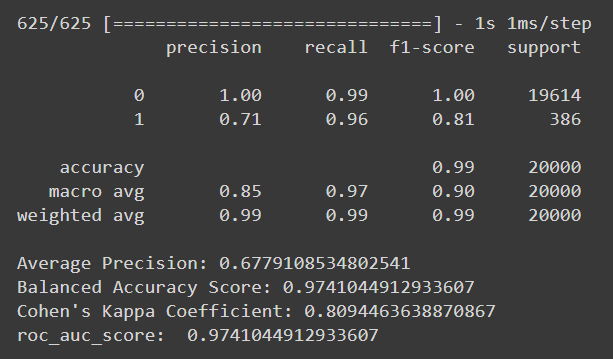
To address the class imbalance in our training data we undersampled the majority class to be equal to the minority class and trained the neural network model over it.



On undersampling, we observe a significant rise in the recall value but at the cost of precision.

**Oversampling-**

To address the class imbalance in our training data we oversampled the minority class using **SMOTE**, and trained over deep learning neural network model over it.

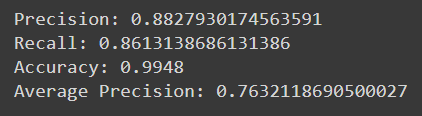
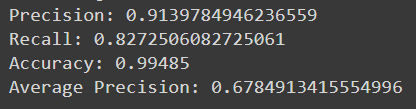


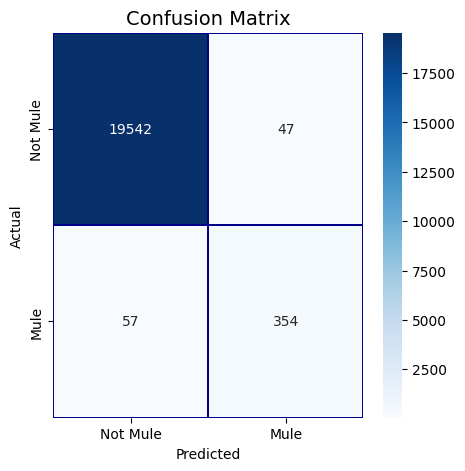
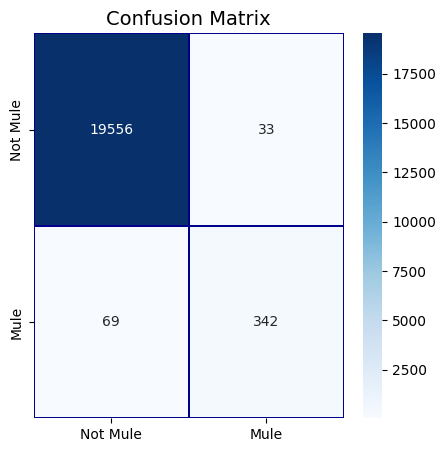
On oversampling, we observe a significant rise in the recall value but at the cost of precision.

Thus, the neural network performs better on the testing data without handling the class imbalances. It works on par with the XGB Classifier too.

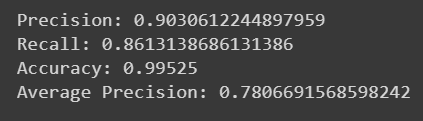
We finally check our model’s performance on the unseen 20% validation data that we created by splitting.

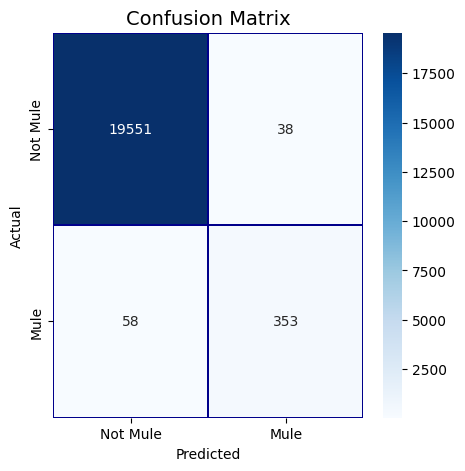
**XGB Classifier Neural Network Model**

** **

**ENSEMBLE OF THE TOP TWO PERFORMING MODELS**

****



We observe that the **ensemble model** performs well on the validation data and the number of false positives reduces without effecting the number of false negatives. Thus, we use this for predicting the probabilities on the given validation dataset with 50,000 account details.

**RESULTS ON THE VALIDATION DATASET**

The Ensemble of the top performing models managed to identify 518 mules out of 50,000.