Fraud/Financial Mule Detection using Various Machine Learning Techniques

Rayani Venkat Sai Rithvik: EE21BTECH11043

Abstract—**In this project, we explore the domain of fraud detection. As a huge number of transactions take place every day, we need an automated method of identifying frauds to maintain law and order. In this problem statement, we need to identify mules who opened savings accounts with the sole purpose of defrauding others. In the dataset there are a lakh data points with over 170 features, with only 2% mules. As a part of data preprocessing, we use Multivariate Imputation by Chained Equations(MICE) and median/mode imputation to fill the missing values. We then use Principal Component Analysis(PCA) to reduce the number of features to 20. To counter the imbalance, we use undersampling and oversampling techniques. We then use k-Nearest Neighbors, Logistic Regression, Decision Trees, and XGBoost models for the normal dataset, undersampled dataset, and oversampled dataset. We see that undersampling worsened the performance of the models, and oversampling did not improve the performance by a great extent, and we analyse the causes for this. XGBoost performed the best with 89.2% precision with the normal dataset.**

I. Introduction

Fraud detection is the process of identifying scams and anomalous instances in financial transactions involving businesses, banks and other institutions. As the methods of transactions have advanced from mere currency notes to debit cards, net banking and UPI, fraudsters are coming up with different ways of scamming innocent people. Thus, to protect the citizens and enforce the law, fraud detection is a huge and urgent problem to solve. As a huge number of transactions happen every day, we need an automatic system for detecting fraud rather than a case-by-case handling of the problem. In this project, an attempt has been made to solve the following variation of fraud detection using machine learning techniques.

The problem statement is an IIT Bombay case study in association with IDFC First Bank. Bank A(name not specified in the problem statement) allows its customers to open their savings accounts through online mediums. All the steps required to open the account, including providing details and documentation, KYC and account funding) can be done online without visiting a physical branch. Considering the time it takes for the offline method, especially on busy days, the online process has improved the customer experience and fastened the process. However, money mules are taking advantage of the online method by creating accounts with Bank A and convincing gullible people to transfer money from their accounts to the mules' accounts. An automated transaction monitoring framework needs to be developed to detect the mules.

II. Related Work

In [1], the authors use different machine learning and deep learning techniques to detect fraud in the PaySim dataset. The thirteen techniques are as follows: MLP Repressor, Random Forest Classifier, Complement NB, MLP Classifier, Gaussian NB, Bernoulli NB, LGBM Classifier, Ada Boost Classifier, K Neighbors Classifier, Logistic Regression, Bagging Classifier, Decision Tree Classifier and Deep Learning. After balancing their dataset, the Bagging Classifier achieved the best metrics. This paper served as an overall survey of the techniques that could be tried, even though XGBoost was not tried in the paper.

[2] is a similar paper that uses the same dataset and focuses on the advantages of ensemble learning methods.

[3] is a standard Kaggle notebook that demonstrates the handling of imbalanced datasets using sampling techniques. The author uses logistic regression to classify the data.

In [4], the author uses Logistic Regression and Support Vector Machine(SVM) on the PaySim dataset.

III. Methodology

A. Dataset Description

There are 1,00,000 savings account details(henceforth referred to as data points) in the given dataset, along with a flag/target. The value of the target is 1 for mules and 0 for others. Each data point has 178 variables(henceforth referred to as features). For security and privacy purposes, the dataset does not mention most of the names of the features. The features are divided into the following types:

- 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.

B. Data Preprocessing

We first need to preprocess the raw dataset. There are various issues with the original dataset that need to be addressed, and they are as follows:

1) Missing Values: The biggest issue with the dataset is the fact that there are several missing values in the data points corresponding to features. As there are a huge number of features, we would need to apply Principal Component Analysis(PCA), and PCA cannot handle missing values. Only 19 features have no missing values, and 46 features have less than 10 missing values. 80 features have less than 10000 missing values, i.e., less than 10% of the data is not present in these features. 9 features have between 10% and 25% of the values missing, and 77 features have exactly 25794 values missing. 8 features have 40-60% of the values missing, and the remaining four features have more than 95% values missing.

As the number of values missing in the features is varied, different techniques have been used to handle the missing data. The features with more than 95% values missing have been dropped, and the missing values in the features with less than 10% have been filled with the median of the data, unless the feature name is mentioned(such as City Tier, Occupation, etc.), in which case the missing data is considered to be a value in itself(and replaced with -1 or 0 depending on the data), as the absence of the data can be informative in such cases.

For the remaining features, an imputation technique called Multivariate Imputation by Chained Equations(MICE) is used to closely identify the relationship between the data points to predict the missing values in each feature accurately. An initial imputation is done to fill all the values with a simple technique such as mean, mode or median. After this, one point is removed and is predicted using a machine learning model with the other filled data points as input. This is iterated through for all the data points, and for multiple cycles, with each cycle consisting of all the missing values being imputed using the method.

2) Various Datatypes: The data points in the dataset are of different datatypes depending on the features. Most of the features have integer-type(int) data, but a few have boolean(bool) values. A few features such as ’os’, ’City tier’, ’occupation’, etc. have string(str) datatypes, and the ’account opening date’ has the TimeSeries datatype. These features have been dealt with separately before imputing the missing values. The boolean type values are converted to 1 and 0(True and False, respectively). With string type values, a unique ID was associated with each string and the string data point is replaced with the ID. The ’account opening date’ is divided into ’date’, ’day’ and ’month’.

3) Number of Features - Dimensionality Reduction: After the previous two steps, we have 175 features. Handling data with such a huge number of features is difficult, so we perform Principal Component Analysis(PCA) after normalizing the data to pick the features that have the highest variance. PCA identifies the hyperplane that lies closest to the data, and projects the data onto it. To pick the number of features we require, we visualise the amount of variance explained(cumulative explained variance) by the number of features. We then pick the number of features corresponding to the point where the plot starts flattening(’elbow’ point). In our case(Fig. 1), we see that the curve flattens at 25 features, and thus, we apply PCA and reduce the number of features to 25.

4) Dataset Imbalance: The dataset is highly imbalanced in favour of non-fraudulent data points, with only 2% of the 1,00,000 data points belonging to mules. This means that

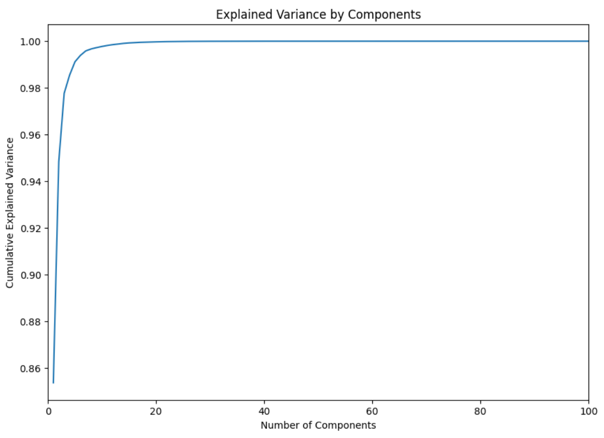

Fig. 1: Explained Variance by components

the machine-learning models can overfit and predict every input as non-mule, which is detrimental to our solution. There are a few ways of trying to make the dataset balanced. Undersampling is a method which removes data points from the majority class to balance the dataset, and oversampling is a method which adds synthetic data points to the minority class which are similar to the original data points. Both were done using the imblearn library in Python. Oversampling was done using SMOTE(Synthetic Minority Over-sampling Technique), which generates points on the line/plane connecting the points belonging to the minority class. While both can ensure balance in the dataset, they have their own disadvantages. Oversampling can make the model overfit the data as the model can learn the algorithm with which the synthetic points were generated, whereas undersampling can remove crucial information from the dataset.

C. Model(s) Description

The following models were used in training and testing the data, with the hyperparameters picked using GridSearchCV, which is explained further in ’Training Paradigm’.

1) k-Nearest Neighbors(kNN): k-Nearest Neighbors is a model that stores all the training data-points in a vector space with their labels. For each test data-point, it computes the distance from all the training data-points to identify its K nearest neighbors and takes a majority vote to predict the label for the test point. Here, K is a hyperparameter that is tuned.

2) Logistic Regression(LR): Logistic Regression is a model that fits a sigmoid curve according to the data. It predicts the probability of a point being in the positive class, and if it is above 0.5, the label is predicted as the positive class. C is a regularization hyperparameter which penalizes the objective function to prevent overfitting. A smaller C is indicative of higher regularization.

3) Decision Trees(DT): Decision Tree is a hierarchical structure consisting of nodes(decisions), branches(possible outcomes), and leaves(class labels). During training, the model learns the thresholds and decisions to be taken for each data point. During testing, the data point traverses through

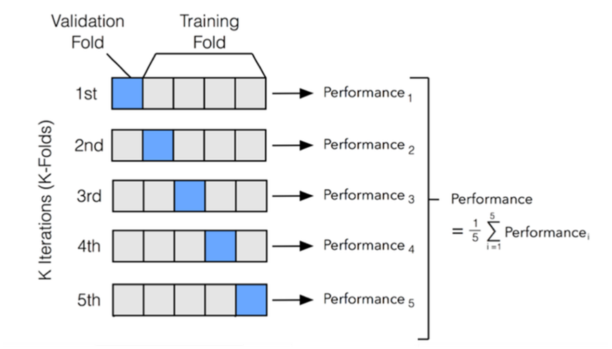

Fig. 2: 5-fold Cross-Validation[5]

the tree and reaches a leaf which is the label predicted. The number of levels is a hyperparameter to be tuned.

4) XGBoost(XGB): eXtreme Gradient Boosting(xGBoost) is a powerful ensemble learning technique that uses gradient boosting. It combines multiple weak learners to strongly learn the features and predict the labels. Each weak learner(usually decision trees) is used to correct the errors of the previous learners by fitting the new learner to the previous errors. This is called gradient boosting, which minimizes the error. Like decision trees, the number of levels in the tree, and the number of learners are a couple of hyperparameters.

D. Training Paradigm

The dataset is split in the train-test ratio 80:20. The models are trained using supervised learning, i.e., the labels of the data are available to us. The hyperparameters of each model are tuned using GridSearchCV. A grid consisting of possible hyperparameters is created, and for each set, the training data is split into 5 folds/parts. The model with a particular set of hyperparameters is trained on 4 folds, and is validated on the fifth fold. This is done 5 times, with each fold being the validation set for different times. This is repeated until all sets of hyperparameters are tested on all the folds and the best set of hyperparameters is chosen.

The scikit-learn library[6] was used for imputation, PCA and the different ML models. The imbalanced-learn library[7] was used for undersampling and oversampling the data.

IV. Experimental Results

The following metrics are used to analyze the results of the model:

- Accuracy: Ratio of correct predictions and total number of predictions
- Precision: Ratio of true positives and total positives predicted
- Recall/Sensitivity: Ratio of true positives to the actual number of positives
- F1-Score: Harmonic mean of precision and recall.
- Confusion Matrix: Visual representation of correct and wrong predictions.

There are three types of averaging typically used:

- Micro-average: Calculated by considering the total number of true positives, true negatives, false positives and false negatives for all classes. Equal importance is given to each sample regardless of the class it belongs to.
- Macro-average: Calculated by taking the average of the respective metric independently calculated for each class. Each class has equal importance regardless of the number of samples in the class.
- Weighted-average: Calculated with the average weights being the class size, with higher weight for larger classes.

As the dataset is overwhelmingly imbalanced, we mention weighted averaging and simple averaging metrics.

Metric	**kNN**	**LR**	**DT**	**XGB**
Accuracy	0.993	0.989	0.991	0.994
Precision	0.875	0.74	0.784	0.892
Recall	0.744	0.645	0.754	0.788
F1-Score	0.804	0.691	0.769	0.836
Precision (Weighted)	0.993	0.988	0.991	0.994
Recall (Weighted)	0.993	0.988	0.991	0.994
F1-Score (Weighted)	0.993	0.988	0.991	0.994

TABLE I: Performance Metrics: Without Sampling

Metric	**kNN**	**LR**	**DT**	**XGB**
Accuracy	0.973	0.977	0.963	0.971
Precision	0.412	0.462	0.339	0.398
Recall	0.974	0.961	0.977	0.984
F1-Score	0.579	0.624	0.504	0.567
Precision (Weighted)	0.988	0.988	0.987	0.988
Recall (Weighted)	0.973	0.977	0.963	0.971
F1-Score (Weighted)	0.973	0.977	0.963	0.971

TABLE II: Performance Metrics: With Under-Sampling

Metric	**kNN**	**LR**	**DT**	**XGB**
Accuracy	0.991	0.980	0.980	0.993
Precision	0.716	0.493	0.495	0.791
Recall	0.842	0.958	0.958	0.894
F1-Score	0.774	0.651	0.652	0.839
Precision (Weighted)	0.991	0.989	0.989	0.994
Recall (Weighted)	0.991	0.980	0.980	0.993
F1-Score (Weighted)	0.991	0.980	0.980	0.993

TABLE III: Performance Metrics: With Over-Sampling

The best-performing model for oversampling and normal sampling is XGBoost. The confusion matrices for XGBoost are shown. Further analysis is written in the 'Conclusions and Future Work' section.

V. Conclusions and Future Work

A. Observations

It is important to note that accuracy is not the best metric to look at for this dataset, as a high accuracy can be achieved even by predicting every test case as non-mule. Thus, we take precision as the main metric to see which model performed the best. We see that XGBoost is the best model when trained on the data without sampling and over-sampling. This is consistent with [2], which claimed that the ensemble learning

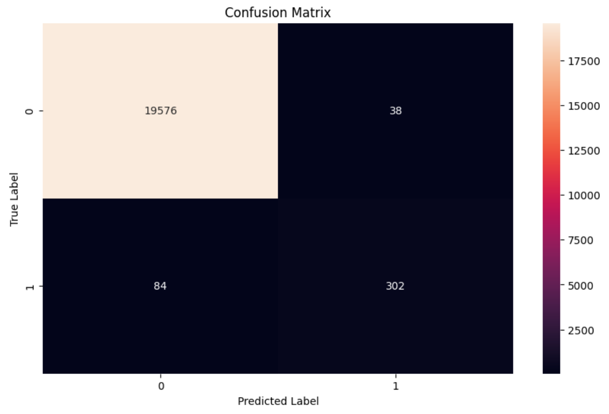

Fig. 3: Confusion matrix using XGBoost: Without Sampling

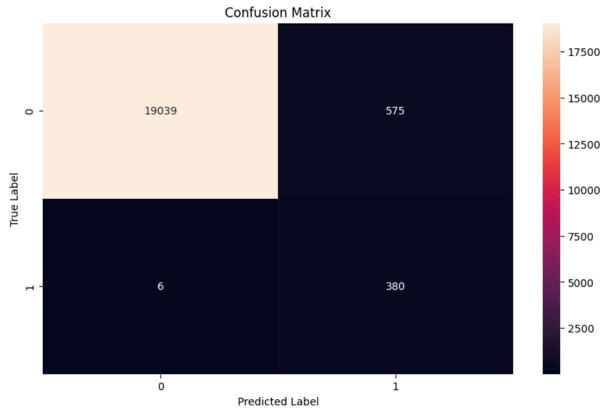

Fig. 4: Confusion matrix using XGBoost: Under-Sampling

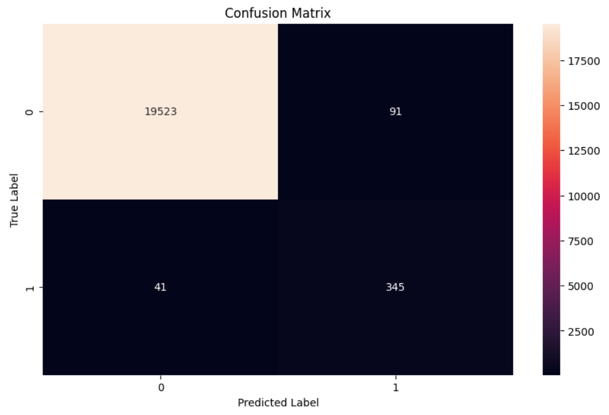

Fig. 5: Confusion matrix using XGBoost: Over-Sampling

methods perform better on such tasks. However, for under-sampling, Logistic Regression seems to perform the best.

It is to be noted that under-sampling and over-sampling did not improve the performance of the models. With over-sampling, more non-mules were predicted as mules, while with under-sampling, more mules were predicted as non-mules. This means that with under-sampling we lost valuable information and with over-sampling the model overfit the data. However, in this particular problem, predicting false negatives are more dangerous than predicting false positives, because of which we can discard the under-sampling technique. The performances of over-sampling and normal sampling are comparable in terms of weighted precision, but over-sampling only increases the computation time for similar metrics.

B. *Future Scope*

We can try more imputation techniques to improve the precision, such as using kNN Imputer or maintaining columns indicating the data points whose values have been synthesised. Running the models with different 'datasets' formed by the imputation techniques can help in achieving more precision, as no particular idea or technique fits all the machine learning or data science problems. We can also try other ensemble learning techniques or deep learning techniques, such as deep neural networks or CNNs, for the same reason. [8] is a survey of different machine learning techniques with advantages and disadvantages listed, which serves as a handy resource for deciding which techniques to try next.

VI. References/Resources

1) Megdad, Mosa M. M. ; Abu-Naser, Samy S. & Abu-Nasser, Bassem S. (2022). Fraudulent Financial Transactions Detection Using Machine Learning. International Journal of Academic Information Systems Research (IJAISR) 6 (3):30-39.
2) Bhakta, Susmit & Ghosh, Sangita & Sadhukhan, Bikash. (2023). Credit Card Fraud Detection Using Machine Learning: A Comparative Study of Ensemble Learning Algorithms. 10.1109/ICSCC59169.2023.10335075.
3) Credit Fraud: Dealing with Imbalanced Datasets by Janio Martinez Bachmann
4) Oza-aditya, Aditya. "Fraud Detection using Machine Learning." (2018).
5) Cross Validation Image
6) Scikit-learn Library
7) Imbalance-learn Library
8) Zojaji, Zahra, Reza Ebrahimi Atani, and Amir Hassan Monadjemi. "A survey of credit card fraud detection techniques: data and technique oriented perspective." arXiv preprint arXiv:1611.06439 (2016).
9) Github repository for this project