Finding The Best Machine Learning Algorithm to Detect Fraud in Mobile Money.

Project idea:

In the year 2020, $4 billion was lost across the world due to fraud in mobile money. Due to these substantial losses, it is critical to look into the situation in order to prevent fraud in the future. This project aims to provide more insight into fraud and find which machine learning algorithm works best in order to detect and prevent such frauds in the future. Five algorithms will be used in this project including Logistic Regression, K-nearest Neighbor, Naive Bayesian, XGBoost, and Random Forests.

Description of the dataset: Source: https://www.kaggle.com/datasets/ealaxi/paysim1

This dataset contains 30 days of mobile money transactions that are simulated based on original transaction logs from the mobile money service provided in an African country using PaySim software. It has 6,362,620 transactions (rows) and 11 columns that are described as shown in table 1.

Table 1: all attributes and type of data in the dataset

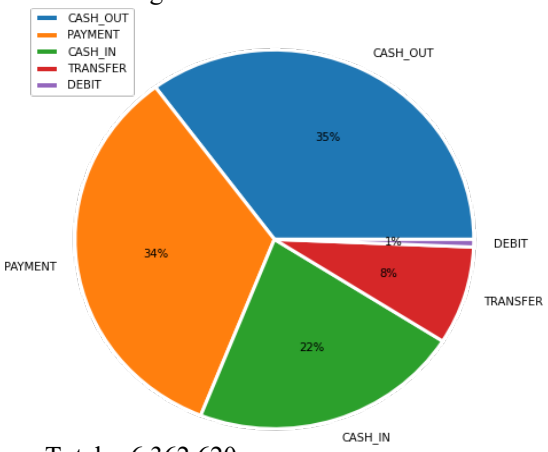
Attribute name	Description	Data type
Step	Total 744 steps; each step represents one hour of time	int
Transaction type	cash in, cash out, debit, payment, transfer	string
Amount	Amount of transaction in local currency	float
nameOrig	Customer code who began the transaction	string
oldbalanceOrg	Balances of the sender before the transaction of the money	float
newbalanceOrig	Balances of the sender after the transaction of the money	float
nameDest	Customer code who received the money	string
oldbalanceDest	Balances of the recipient before the transaction of the money	float
newbalanceDest	Balances of the recipient after the transaction of the money	float
isFraud	Transfers done by the fraudulent agents inside the simulation	int
isFlaggedFraud	Will be set when the amount of transaction crosses 200,000	int

1. Data exploration:

1.1. Six types of transactions were explored. Most transactions are cash out and payment accounting for 35% and 34% respectively according to figure 1. Half of the total

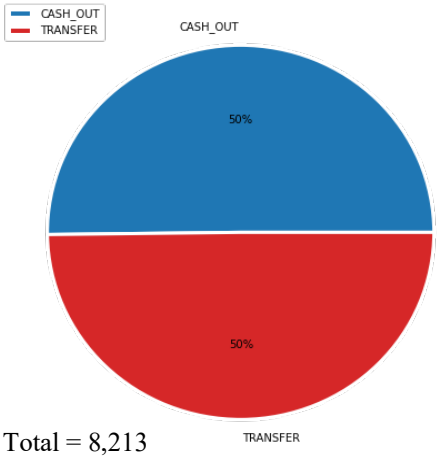
amount of fraudulent transactions, 4116, comes from cash-out transactions, and the other half stems from transfer transactions, 4097 according to figure 2. There was no fraud in other types of transactions at all.

Figure 1: Total Transactions



Total = 6,362,620

Figure 2: Fraudulent transactions



Total = 8,213

1.2. The box plot and bar chart were used to visualize the amount and numbers of fraudulent transactions. As a result, according to figure 3, there are 16 fraudulent transactions that have an amount of 0, so assuming that fraud did not actually happen, they were excluded from the analysis. Thus, fraudulent transactions will be 8,197. Moreover, using box plots in figure 4, it was found that 50% of fraudulent transactions have amounts ranging from 127,091.33 to 1,517,771.48. There are also many extreme amounts of fraudulent transactions making this distribution skewed to the right.

Figure 3: Number of fraudulent transactions with different amount

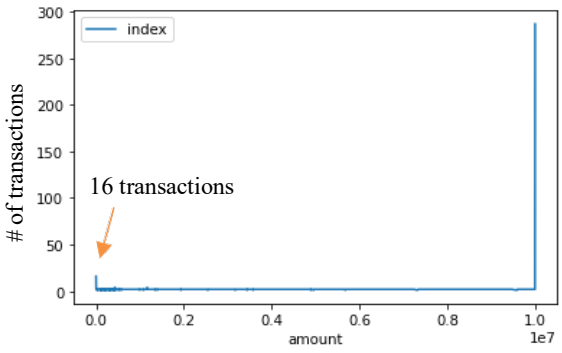
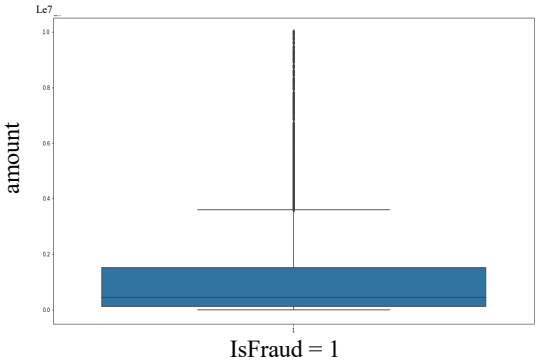


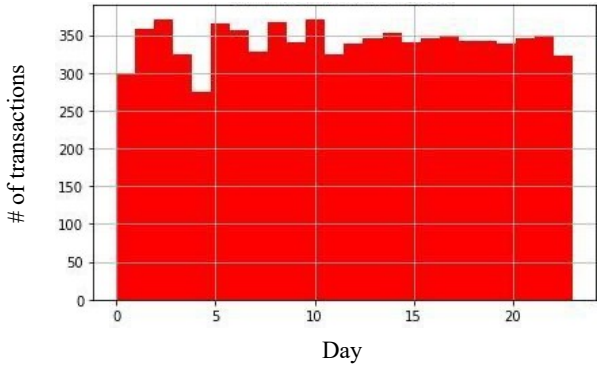
Figure 4: Amount of transactions



1.3. The nameOrg and nameDest have two types of relations, one is customer to customer (CC) and another one is customer to merchant (CM). Number of fraudulent transactions is type CC, which has total transaction of 8197, whereas type CM has 0.

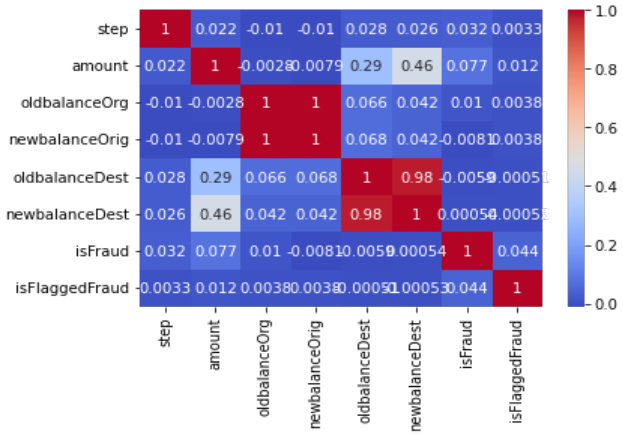
1.4. Time of fraud occurrence: for transactions with isFraud = 1, the frequency of fraudulent transactions occurrence does not change much each day as shown in figure 5.

Figure 5: Number of fraudulent transactions each day



1.5. Correlations between variables: correlation matrix was used, and the result showed that the isFlaggedFraud variable is least related to the other variables. Also, there is a good relation between amount and newbalanceDest and between amount and oldbalanceDest.

Figure 6: Correlation matrix



2. Data cleaning:

2.1. Missing data

Data has been checked for missing values using the isnull() method of pandas. No missing data was found.

2.2. Variables

2.2.1. After examining the 'isFlaggedFraud' variable, it will be dropped because it is hardly correlated with any other variables and fails to fulfill its purpose, which is to set a flag when transaction amount is more than 200,000.

2.2.2. A new attribute (nameOrigDest) is derived from nameOrig and nameDest where 1 represents CC and 0 represents CM. nameOrigDest will be used in the model and two original attributes are dropped.

2.2.3. Attributes in the model will include step, type of transaction, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDest, newbalanceDest, isFraud, and nameOrigDest.

2.3. Dealing with imbalance data

Since the data is imbalanced, supervised machine learning algorithms will mostly perform well to classify the majority class(non-fraud), but worse in capturing minority groups(fraud) in the dataset. Since fraud is a minority and our focus, resampling methods are used to mitigate data imbalance problems including Random Oversampling and Random Undersampling. These two methods will be used to modify only the training dataset used to fit the model [1]. Random oversampling is when fraudulent transactions are randomly duplicated to increase its data points and balance the distribution. Random undersampling is when non-fraudulent transactions are randomly resampled to make its data points equal to fraudulent data.

3. Analysis and Model Selection:

3.1. Data is split into a training and testing set with the proportion of 67% for the training set and 33% for the test set. Random undersampling and oversampling were used to modify the training dataset, which we will use to fit model.

3.2. **Logistic regression:** Logistic Regression is a supervised learning algorithm that is used when the target variable is categorical. The reason behind selecting logistic regression to build the classifier is related to its efficiency of detecting frauds based on its ability to isolate the data that belong to different binary classes.

3.3. **Naive Bayes algorithm:** Naive Bayes algorithm is a statistical classification technique based on Bayes Theorem that classifies data based on the greatest likelihood. GaussianNB from scikit-learn library was used and the benefit of using

this algorithm are that it is very fast since probability can be directly computed and is simple to implement [2].

3.4. **K-nearest Neighbour:** The K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. It makes use of a distance function like the Euclidean distance, and this makes the KNN algorithm a little more compute-intensive for fraud detection [3].

3.5. **Random Forest**: Random Forest is supervised machine learning that generates decision trees by choosing random data samples and selects the best solution among predictions of each tree by voting. It has methods which makes it suitable for unbalanced datasets by maintaining balancing [4].

3.6. **XGBoost:** XGBoost is an open-source Python library that provides a gradient-boosting framework. It helps in producing a highly efficient, flexible, and portable model. XGBoost is built on top of a gradient-boosting framework. It optimizes the model when making predictions [5].

4. Algorithm comparison and Conclusion

The confusion matrix and various evaluation metrics including accuracy, F1 score, recall, precision, and AUC are utilized to evaluate the algorithm and perform the comparison of the algorithms. As fraud data is an imbalance dataset, accuracy is not a suitable parameter to measure the effectiveness as it focuses more on the majority class (Non-Fraud cases). To mitigate imbalance problem, the resampling techniques were implemented to mitigate the imbalance data problem. After the imbalanced problem were mitigate, False negatives (FN) and false positives (FP) should be the main focus of evaluating algorithms because FN is wrong predictions and FP is false alarms.

As a result, the resampling method decreased the FN, but increased FP for every algorithm in our analysis as shown in table 2. One way to see the effect is to look at recall and precision. This directly affected both recall and precision metrics. When FN decreases, it causes the recall to increase, which means that the algorithm does a better job in detecting fraud as it is shown in table 3 to 5. However, this improvement also comes with the cost of increasing false alarm, which is increasing in FP. When FP increases, it will make precision decrease as shown in table 3 to 5. This means that most algorithms gave a significant amount of false alarms after resampling even though it gave less wrong prediction.

In addition, the resampling effect can also be seen in F1 scores. F1 score is also affected by the increase in FP and decrease in FN making its value decreasing for both oversampling and undersampling, except for random forest in the oversampling case. For random forest, oversampling does not cause FP to change significantly (from 47 to 78 in table 2), so F1 score remains the same.

Overall, one of the ways to evaluate overall effectiveness of algorithms is using AUC. AUC stands for area under the curve which tells us the measure of separability and how much a model can separate two classes. The higher the AUC, the higher the chance that the model would be able to segregate two points. AUC is increasing in both the cases of undersampling and oversampling compared to the original dataset. This means that resampling technique help improve overall effectiveness of algorithm. Among all algorithms, XGBoost has a high AUC value at 0.99 both after undersampling and oversampling.

Finally, looking into the confusion matrices in table 2 and comparing all cases for every algorithm, the FN value of XGBoost is significantly low compared to other algorithms in all cases. In the undersampling case, FN is significantly decreasing from 345 to 12, but FP also extremely increased from 58 to 17,376. After performing oversampling, even though XGBoost is giving slightly higher FN (30) compared to the undersampling case (12), the algorithm produces much lower FP (2,598). This means that performing oversampling in the dataset and using XGBoost is the best for Fraud detection.

Source of code:
https://drive.google.com/drive/u/0/folders/1UdNAynZT3KwmFPYZ88jWOB0OsNb67AFg

Confusion matrix of algorithms

Table 2**:** Confusion matrix

		Original dataset		Undersampling		Oversampling	
Logistic		Isfraud=0	Isfraud=1	Isfraud=0	Isfraud=1	Isfraud=0	Isfraud=1
	Isfraud=0	2,095,012	1,987	1,916,661	180,328	1,918,324	178,665
	Isfraud=1	1,497	1,169	289	2382	290	2381
Naive Bayes	Isfraud=0	2,083,788	13,201	2,014,361	82,628	2,029,982	67,007
	Isfraud=1	2,215	456	1,685	986	1,631	1,040
Random Forest	Isfraud=0	2096942	47	2071732	25257	2096911	78
	Isfraud=1	572	2099	20	2651	532	2139
XGBoost	Isfraud=0	2,096,931	58	2079613	17376	2094391	2598
	Isfraud=1	345	2,326	12	2659	30	2641
KNN	Isfraud=0	2,096,816	173	2,034,690	62,299	2,096,223	766
	Isfraud=1	942	1729	165	2506	720	1951

Evaluation metrics

Table 3: evaluation metrics for original dataset

Algorithms	**accuracy**	**F1**	**recall**	**precision**	**AUC**
Logistic regression	0.99	0.40	0.44	0.37	0.7215
Naive Bayes	0.99	0.06	0.17	0.03	0.582
Random Forest	0.99	0.87	0.79	0.98	0.89
XGBoost	0.99	0.92	0.87	0.98	0.94
K-nearest neighbor	0.99	0.76	0.61	0.91	0.823

Table 4: evaluation metrics for undersampling dataset

Algorithms	accuracy	F1	recall	precision	AUC
Logistic regression	0.91	0.03	0.83	0.01	0.902
Naive Bayes	0.96	0.02	0.37	0.01	0.665
Random Forest	0.98	0.17	0.99	0.09	0.99
XGBoost	0.99	0.23	1.00	0.13	0.99
K-nearest neighbor	0.97	0.07	0.94	0.04	0.954

Table 5: evaluation metrics for oversampling dataset

Algorithms	accuracy	F1	recall	precision	AUC
Logistic regression	0.91	0.03	0.89	0.01	0.903
Naive Bayes	0.97	0.03	0.39	0.02	0.679
Random Forest	0.99	0.88	0.80	0.96	0.90
XGBoost	0.99	0.67	0.99	0.50	0.99
K-nearest neighbor	0.99	0.72	0.73	0.72	0.865

5. Reference

1. Baesens, B., Höppner, S., Ortner, I. et al. robROSE: A robust approach for dealing with imbalanced data in fraud detection. *SpringerLink.*

2. Banerjee, R., Bourla, G., Chen, S., Kashyap, M., Purohit, S., & Battipaglia, J. (2018). Comparative Analysis of Machine Learning Algorithms through Credit Card Fraud Detection. 2018 IEEE MIT Undergraduate Research Technology Conference (URTC).
3. Cao, L., Liu, X., Zhou, T., Zhongping , Z., & Liu, A. (n.d.). Based On The Flow Of K Nearest Neighbors Algorithm For Data Mining Outliers. 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT).
4. Goel, E., Abhilasha, & Agarwal, A. (n.d.). Fraud Detection Using Random Forest Algorithm. International Journal of Computer Science Engineering.
5. Abdulghani, A., Osman , N., & Khattab , M. (2021). Credit Card Fraud Detection Using XGBoost Algorithm. 2021 14th International Conference on Developments in eSystems Engineering (DeSE).