

**Final Project Report**

**Mobile Payments Fraud Detection using Machine Learning**

Table of Contents

[Introduction 4](#_Toc17155076)

[Related Work 4](#_Toc17155077)

[Data Collection and Sources 4](#_Toc17155078)

[Data Cleaning and Feature Engineering 5](#_Toc17155079)

[Visualisation 6](#_Toc17155080)

[Training, Testing and Validation Dataset 6](#_Toc17155081)

[Methods 6](#_Toc17155082)

[Data Approach 6](#_Toc17155083)

[Algorithm Approach 7](#_Toc17155084)

[Ensemble algorithms 7](#_Toc17155085)

[1. Random Forest Classifier 7](#_Toc17155086)

[Experiment 8](#_Toc17155087)

[Why we used Random Forest 9](#_Toc17155088)

[Result 9](#_Toc17155089)

[2. Gradient Boosting Classifier 9](#_Toc17155090)

[Experiment 9](#_Toc17155091)

[Why we use Gradient Boosting: 10](#_Toc17155092)

[Result: 10](#_Toc17155093)

[3. AdaBoosting Classifier 10](#_Toc17155094)

[Experiment 10](#_Toc17155095)

[Why we use AdaBoosting: 11](#_Toc17155096)

[Result: 11](#_Toc17155097)

[Non-Ensemble algorithms 11](#_Toc17155098)

[1. Logistic Regression 11](#_Toc17155099)

[Experiments 12](#_Toc17155100)

[Why we use Logistic Regression: 12](#_Toc17155101)

[Result: 13](#_Toc17155102)

[2. Support Vector Machine 13](#_Toc17155103)

[Experiments 14](#_Toc17155104)

[Why we use SVM classifier: 15](#_Toc17155105)

[Results: 15](#_Toc17155106)

[Implications 15](#_Toc17155107)

[Learning points from the project: 15](#_Toc17155108)

[References 15](#_Toc17155109)

# Introduction

We are implementing Artificial Intelligence and Machine learning algorithm based fraud detection tools to predict Mobile payment fraud in this project. For this, we are using the synthetic Mobile Payments Dataset found in the Kaggle web site

The input of the data to this algorithm is a mobile payment transaction and the output will be the transaction with the field isFraud set to ‘0’ (False) or ‘1’ (True) indicating whether the transaction is fraud or not fraud.

The data from this dataset is cleaned, transformed and perform SMOTE analysis / Random undersampling to handle imbalance. The upsampled or the undersampled data are used to train and validate the algorithms to predict if the transaction is fraud. Various ML technology Classification algorithms are used for the binary classification, to classify the Mobile payment transaction as fraud or non-fraud.

We are using the below mentioned classification algorithms for this project. we are comparing the performance of the algorithms by predicting the fraud transactions.

* Logistic regression
* Random Forest Classifier
* Gradient Boosting Classifier
* AdaBoost Classifier
* Support Vector Machines

The accuracy of the algorithms is evaluated using following metrics such as Cross validation, Classification Accuracy, Confusion Matrix, Area under Curve (AUC), F1 Score

# Related Work

We have referred the below links for our project

<https://www.kaggle.com/viznrvn/optimal-parameters-for-svc-using-gridsearch>

<https://www.kaggle.com/enespolat/grid-search-with-logistic-regression>

# Data Collection and Sources

The dataset can be found and downloaded from the link

<https://www.kaggle.com/ntnu-testimon/paysim1>

This dataset has around 6.5 Million records out of which around 8500 are marked as fraud.

The dataset has the following fields.

**step** – The step in the full transaction.

**type** – type of transaction. The dataset has 5 types of transactions.

**amount** – transaction amount

**nameOrig** – name of the person/organization who initiated the transaction. This is obfusticated.

**oldbalanceOrg** – Starting balance of the originator account.

**newbalanceOrig** – Ending balance of the originator after the transaction amount is posted.

**nameDest** – Name of the person/organization of the destination account.

**oldbalanceDest** – Starting balance of the destination account before the transaction amount is posted to the account

**newbalanceDest** – Ending balance of the destination account after posting the transaction amount.

**isFraud** –Boolean flag indicating if the transaction is fraud.

**isFlaggedFraud** – The description of this field is not clear. There are only 16 transactions out of 6 million that has a value of 1 in this field and rest are 0.

## Data Cleaning and Feature Engineering

However, on further data exploration the following scenarios are identified which are the drivers for data clean up and preparation.

1. There are no null values in the dataset.
2. The field isFlaggedFraud doesn’t seem to have any relation to the fraud transactions. There are only 16 transactions out of 6 million transactions which has a value ‘1’ in this field and there is no pattern that dictates this value. This column can be dropped.
3. The features nameOrig and nameDest doesn’t contribute to the fraud transactions and these can be dropped.
4. Among all the types of transactions, only “TRANSFER” and “CASH\_OUT” types have transactions that are tagged as fraud. Rest of the transaction types can be dropped. This brings down the number of transactions to around 2.5 Million.
5. Upon closer observation, most of the Fraud transactions in fact 48% for them have mismatch in the newbalanceDest and the oldbalanceDest after the transaction amount is posted. This error is contributing to majority of the transactions. We need add two new features to store the difference in expected newbalanceDest and actual newbalanceDest and also expected newbalanceOrig and actual newbalanceOrig.
6. We also need to perform one-hot encoding on the transaction type.
7. After the data is cleaned and the new features are added, we split the data into training, visualisation and testing set, then we performed either SMOTE analysis and Upsampled or undersampled the dataset to create balance between the fraud transactions and non-fraud transactions.

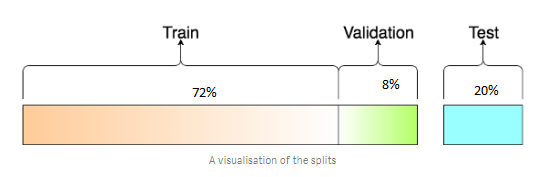
## Visualisation

The step feature is an important unit to represent time. We deployed a strip plot to capture the genuine and fraudulent transactions over time versus the differences in original and destination account. Leading to an understanding that the destination account behaviour is a better representation of fraud.

For the purpose of visualisation, the type is binary coded, the Type TRANSFER represented as “0” and type CASH\_OUT represented as “1”.

## Training, Testing and Validation Dataset

* We split dataset into training and testing datasets in a ratio of 80:20. Since the data is heavily imbalanced, we have used stratified splitting in order to get a similar ratio of the fraud transactions in both the training and testing datasets.
* Then we created a validation set from the training set. We set aside 10% of the training set for validation. We have used stratified splitting for creating the validation set.
* The data split for training, validation and testing dataset are 72:8:20



# Methods

There are the approaches for handling an imbalanced dataset. There can be the Data approach and Algorithm approach. In this project, we adopted both approaches.

## Data Approach

Typically, we went through feature extraction and engineering, then we executed an imputation, and followed by resampling approach. The objective is to balance the classes by increasing minority or decreasing majority. There are the random under-sampling and random oversampling methods.

While random over-sampling may lead to bias and random over-sampling leads to overfitting due to copying same information.

We adopted SMOTE (Synthetic Minority Oversampling Technique), and complete with cross validation. For skewed data set, it is better to over-sample on the minority class, which are the fraud data.

SMOTE typically creates new synthetic observations. The process typically starts with:

* identify the feature vector and its nearest neighbour
* take the difference between the two
* multiply the difference with a random number
* identify a new point on the line segment by adding the random number to the feature vector
* repeat the process for identified feature vectors

## Algorithm Approach

The goal of the algorithm is to minimise the cost of miscalculation. This is a binary classification problem. For this project we have selected the following algorithms for evaluation and identify the best model and parameters out of these algorithms. Majority of the algorithms that we selected are ensemble algorithms which we believe give better performance than non-ensemble algorithms. To prove our hypothesis we have also selected few non-ensemble algorithms to compare their performance with ensemble algorithms.

|  |  |
| --- | --- |
| Ensemble algorithms | Non-Ensemble algorithms |
| * Random Forest Classifier * Gradient Boosting Classifier * AdaBoost Classifier | * Support Vector Machine * Logistic regression |

The performance of these algorithms is measured using the following metrics

* Cross validation
* Classification Accuracy
* Confusions Matrix
* Area under Curve (AUC)
* F1 Score

## Ensemble algorithms

Ensemble combine several decision trees classifiers to produce better predictive performance compared to a single algorithm

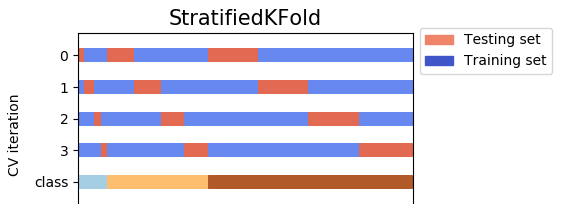
### Random Forest Classifier

Random forest selects random samples from a given dataset. For each sample, it constructs a decision tree and get a prediction result from each decision tree. Then performs a vote for each predicted result and selects the result with the most votes as the final prediction.



#### Experiment

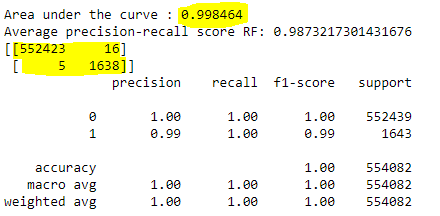
1. Using StratifiedKFold, the data of the training set is shuffled once in the beginning then divided into( 6 splits) it will randomly use a part of each split as test set and training set, to make sure the data from each class is randomly selected



1. Training set generated using StratifiedKFold, is upsampled to create balance between the fraud and non-fraud transactions and no changes are made to the test set.
2. Train the RandomForestClassifier model (w default parameters) on the Training set. Evaluate the model on the testing set of StratifiedKFold (by calculating performance metrics such as accuracy, recall). The accuracy and recall generated for the StraifiedKFold test set are as below.



1. Then we evaluate the model over the Testing dataset predict the values and compare the expected result and the predicted result. The results are as below along with confusion matrix. *The accuracy is around 100%*.



Why we used Random Forest**:**- Produces highly accurate and robust result as number of decision trees participate in the process.  
- Takes average of all the predictions, cancels out biases and reduces over-fitting  
- Handles missing values very efficiently

Result**:**

Random forest classifier to able to predict the result with the accuracy of around 99.8%.

### Gradient Boosting Classifier

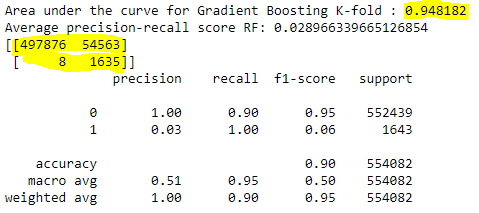
Gradient Boosting works by combining a number of weak learners to form one strong learner. Regression trees are used as a base learner for Gradient Boosting. Each subsequent tree in series is built on the errors calculated by the previous regression tree. It is one of the leading Ensemble methods.

#### Experiment

1. Similar to Randomforest classifier we use StratifiedKFold for Gradient Boosting as well. We Train the Gradient Boosting model (w default parameters) on the Training set of StratifiedKFold. Evaluate the model on the testing set of StratifiedKfold (by calculating performance metrics such as accuracy, recall).



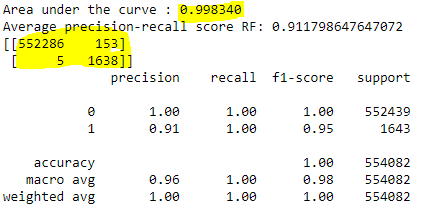
1. Then, we evaluate the model over the Testing dataset, predict the values and compare the expected result with the predicted result. The accuracy is 94%.



1. Using GridSearchCV from sklearn, We find the optimal parameter to improve our accuracy. The best parameters we found are

{'learning\_rate': 0.005, 'max\_depth': 3, 'n\_estimators': 2}

1. After finding best parameters from tuning, we Fit the model with best parameters over training set and evaluate with Testing set and compare the results generated at step ii. *We found significant improvement after tuning and the accuracy is around 99.8%*



Why we use Gradient Boosting:- Often provides predictive accuracy that cannot be beaten

- Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible  
- No data pre-processing required - works great with categorical and numerical values as is

#### Result:

After parameter tuning Gradient Boosting is able to predict with the accuracy of around 99.8%.

### AdaBoosting Classifier

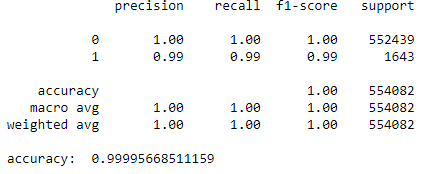
AdaBoost trains base classifiers using random subsets of samples drawn from the training set. AdaBoost uses adaptive probability distribution. Each instance in the training set is given a weight. The weight determines the probability of being drawn from the training set. AdaBoost adaptively changes the weights at each boosting round

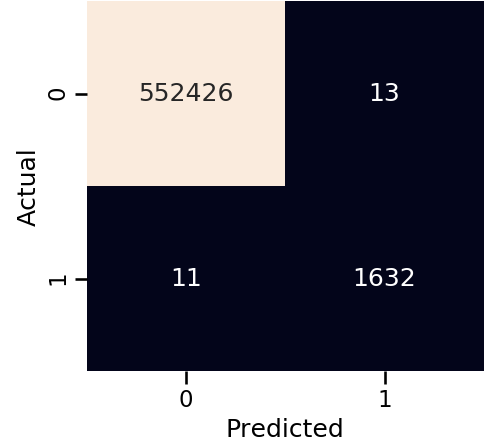
#### Experiment

1. Using GridSearchCV from sklearn, We find the optimal parameter to improve our accuracy of AdaBoost. The best parameters we found are

{'learning\_rate': 0.1, 'n\_estimators': 100}

1. After finding best parameters from tuning, we Fit the AdaBoost model with best parameters over training set and evaluate with Testing set. We got around 99.9% accuracy





Why we use AdaBoosting:- Very simple to implement

- Fairly good generalization

#### Result:

AdaBoosting is able to predict with the accuracy of around 99.9% with best parameters.

## Non-Ensemble algorithms

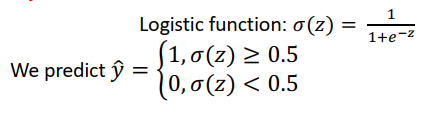
### Logistic Regression

Logistic regression takes the input data and outputs a probability within 0 and 1 (fraud and not fraud). Given the input dataset, we calculate the logistic function 𝜎(𝑧) = 0.8, the chances that it is a fraud is 80%

P(X) = P(Y=1|X)

P(X) -> probability of fraud, P(Y)-> probability of not fraud

We make the prediction based on the calculated probability of the occurrence by fitting the below logistic function

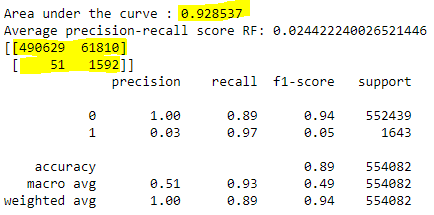


#### **Experiments**

1. Similar to Randomforest classifier we use StratifiedKFold for Linear Regression as well. we Train the Linear Regression model (w default parameters) on the Training set of StratifiedKFold. Evaluate the model on the testing set of StratifiedKfold (by calculating performance metrics such as accuracy, recall).



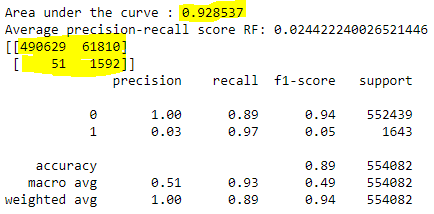
1. Then, we evaluate the model over the Testing dataset predict the values and compare the expected result with the predicted result. The accuracy is 92%.



1. Using GridSearchCV from sklearn,We find the optimal parameter to improve our accuracy. We use validation set to find the optimal parameters, as using GridSearchCV on training set is expensive.The best parameters we found are

{{'C': 0.001, 'penalty': 'none'}

1. After finding best parameters from tuning, we Fit the model with best parameters over training set and evaluate with Testing set and compare the results generated at step ii. *We found no difference in the score before and after tuning*



Why we use Logistic Regression:- Widely used and efficient and doesn’t require too many computational resources

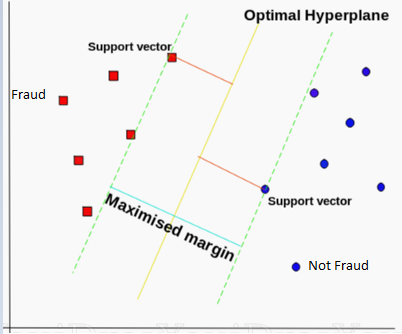
- Doesn’t require to scale to input features  
- Interpretable and easy to regularize and outputs well calibrated probabilities

#### Result:

Instead of showing improvement after parameter tuning it is returning the results with same accuracy, overall the prediction accuracy is low.

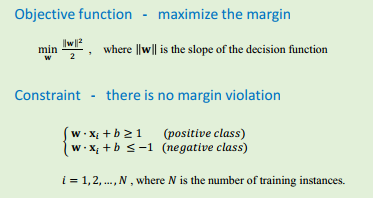
### Support Vector Machine

SVM algorithm takes the input data and outputs a line (decision boundary) that separates classes if possible (fraud and not fraud). SVM algorithm finds the best decision boundary which separates two classes to the maximum. It uses support vectors and margin for this.



In SVM algorithm, we find the data points that are closest to the line from both the classes. These points are called support vectors. Then, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The line for which the margin is maximum is the optimal hyperplane.

The training task in SVM can be formalized as the following constrained optimization problems

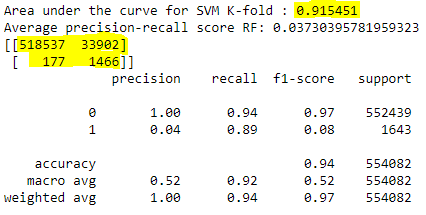


#### **Experiments**

1. Training set generated using StratifiedKFold with 10 splits, is undersampled to create balance between the fraud and non-fraud transactions and no changes are made to the test set.
2. Train the SVM classifier model (w default parameters) on the Training set. Evaluate the model on the testing set of StratifiedKfold (by calculating performance metrics such as accuracy, recall). The accuracy and recall generated for the StratifiedKFold test set are as below.



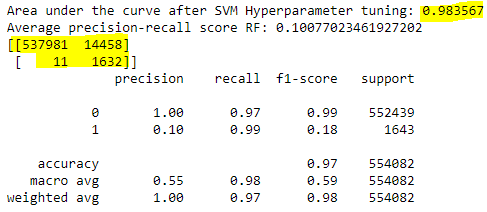
1. Evaluate the model over the Testing dataset predict the values and compare the expected result with the predicted result. The accuracy is 91%.



1. Using GridSearchCV from sklearn, We find the optimal parameters for C, gamma and kernel to improve our accuracy. We use validation set to find the optimal parameters, as using GridSearchCV on training set is expensive. The best parameters we found are

{'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}

1. After finding best parameters from tuning, we Fit the model with best parameters over training set and evaluate with Testing set and compare the results generated at step iii. *We found significant improvement in our accuracy with 98%.*



Why we use SVM classifier:- Works well for data with clear margin of separation but very slow to handle large dataset

#### Results:

After parameter tuning, the accuracy improved from 91% to 98%

# Implications

The result of these project can be put to good use. Based on the feature importance that we derive, we can already reduce some data, and on an operational level for mobile payment companies, they can reduce false alerts.

From the precision scores and other results, can be compiled into a dashboard for the fraud analyst, based on the likelihood of a transactions being fraudulent.

In future, we can possibly explore other algorithms such as Tomek Link removal and Condensed nearest neighbour popular with KNN techniques.

# Learning points from the project:

Along the way, we learnt to always split into test and train sets before trying oversampling techniques. Oversampling before splitting the data can allow the exact same observations to be present in both the test and train sets. There is leaking of data that cause our model to simply memorize specific data points and lead to overfitting and poor generalization to the test data

# References

Synthetic financial datasets for fraud detection, synthetic datasets generated by the PaySim mobile money simulator. <https://www.kaggle.com/ntnu-testimon/paysim1>

Boyle Tara, Dealing with Imbalanced Data – a guide to effectively handling of imbalanced datasets in Python, Feb 2019

Imbalanced-learn.org

<https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989#targetText=Support%20Vector%20Machines(SVM)%20%E2%80%94%20An%20Overview&targetText=Machine%20learning%20involves%20predicting%20and,algorithms%20according%20to%20the%20dataset.&targetText=The%20idea%20of%20SVM%20is,separates%20the%20data%20into%20classes.>

Machine Learning Algorithms Written Assignment and course Notes