A Project Report

On

# Anonymized credit card transactions labeled as fraudulent or genuine

BY

Pedda Pavani Kalyani-17B01A1289

Pulla Meghana -17B01A1294

Bodu Leeladhar -17331A0422

Rajesh Talla -321710303041

Under the supervision of

**Dr.Aruna Malapati**

**Assistant Professor**

**(June 2020)**

# CONTENTS

# ACKNOWLEDGMENTS

We would like to express our immense gratitude and sincere thanks to Dr. Aruna Malapati madam, Asst. Professor for her guidance, valuable suggestions and encouragement in completing the internship project work within the stipulated time.

We would like to express the sincere thanks to GoalStreet ,Mr. Seshu Kandregula sir, director of Goal Street, Mr. Durga Naveen Kandregula sir, co-founder of Goalstreet for permitting us to do our Internship project work at GoalStreet Pvt ltd. Finally, we would also like to thank the people who have directly or indirectly helped us and other faculty, friends and goalstreet team for their cooperation in complete in the project work.

# ABSTRACT

In today's digital world, we are all travelling on the express-train to a destination of the cashless society. As per the World-Payments-Report (2016), the total transactions (non-cash) had augmented by 10.1 percent from the year 2015 for around a whopping 482.6 billion transactions! That is huge! Isn't it! Also, it is even predicted that in the years to come, there will be a steep growth of non-cash transactions. Though, this may sound very exciting, but on the flipside, credit card fraud transactions are rising too. So, what can be done to mitigate this risk?

**Need to Empower Our Financial Institutions with Credit Card Fraud Detection Systems Based on Machine Learning!**

While there may be many methods for limiting such losses and for preventing such kinds of credit card frauds, let us take you on a walkthrough of some widely used **Credit Card Fraud Detection processes** by the core banking sector:

Credit card fraud detection project using machine learning: In this, first the analysis on Credit Card Fraud Detection is done to then carry on with enlisting, writing, and implementation of the credit card fraud detection project source code.

Real-time credit card fraud detection using PHP source code.

A Feature Extraction Method for quick Credit Card Fraud Detection.

A Differentiate Analysis for quick Credit Card Fraud Detection using machine learning techniques.

A schematic method based on machine learning for detection of online transaction fraud based on individual user behavior.

A graph-based, semi-supervised, credit card fraud detection system

**Innumerable Advantages with Hosts of Benefits: Credit Card Fraud Detection**

Credit card fraud detection issues display a mods operand and act as a model of carrying out a fraud credit card transaction. By exactly pinpointing those fraudulent transactions, this particular model is then utilized for identifying if any new transaction being carried out is a fraud or not through AI enabled algorithms and applications. The aim is to detect all fraudulent transactions, and also to minimize the wrong fraud classification(s).

The significance of Machine Learning can't be overstated!

The application algorithms used for identifying such fraudulent transactions have different control parameters which must need to be tweaked so that they fit into the particular data set. Resultantly, the application, which has been developed, will improve, and it will become more efficient and empowered for solving the underlying issue of fraud detection and fraud classification.

Classification is basically a machine learning paradigm, which consists of deriving the function, which separates data into different categories/classes, and that are characterized by

a training set of data, which consists of observations (and/or instances) for which the category membership is already known. Such a function is then utilized for identifying the categories where the new observation belongs to.

In this project, we attempt to predict the transaction is a fraudulent transaction or not using various features given the data set. We attempt to implement classification algorithms and find the best algorithm along with fine-tuning the parameters of the algorithm.

**Proposed System:**We aim to build advanced classification models and also fine-tune the parameters of the model. These models would be trained on a data set which will be engineered carefully after performing the feature engineering.

# INTRODUCTION

In this crime-prime economy of today, if someone asks you for cash or credit, your first quick-thought-of answer would be credit as keeping cash or transacting cash with ATM's queues is always a hassle, let alone the fear of theft associated with the same. No wonder we are progressing today to become the cashless economy! But with merits, there are also some pitfalls with cashless transactions! You got into the habit of using a credit card; and Voila! What a terrific habit it is! But by now you must have got aware of the common demerits of using credit cards; one being the **credit card frauds** that have become the order of the day. Though tougher than cash to steal, fraudulent transactions are happening in these kinds of purchases, where the attackers steal the credit card information. If the credit cardholder has not realized the loss of his card, then it can lead to serious financial loss to the company to which the credit card belongs!

## Data Collection

This dataset is available on Kaggle.

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependent cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

# DATA VISUALIZATION

Data visualization is an important skill in applied statistics and machine learning.

Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding.

This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral to yourself and stakeholders than measures of association or significance.

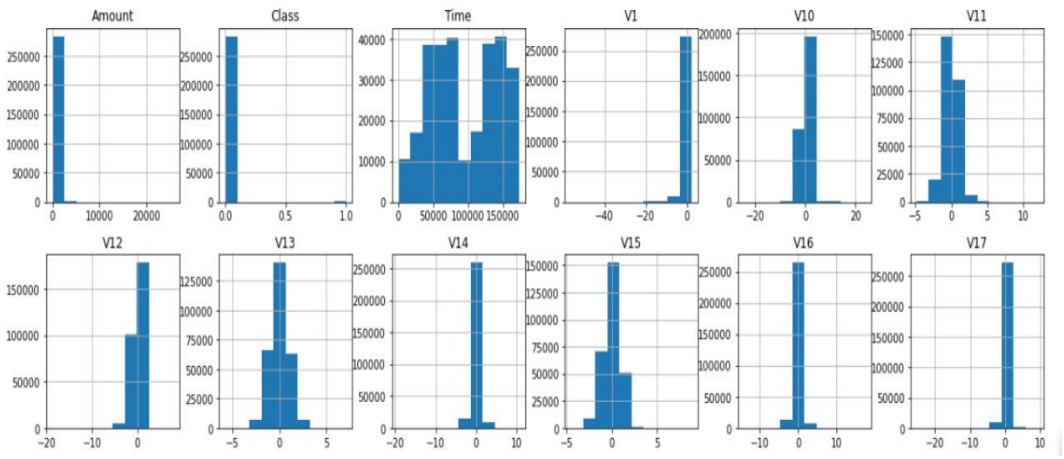
Data visualization and exploratory data analysis are whole fields themselves and I will recommend a deeper dive into some the books mentioned at the end.

**data.head():** It enables us to print the required records to get printed. In the code, we used an argument inside the head() method to display only the first 5 records from our data set. The data.head() display the given output below:

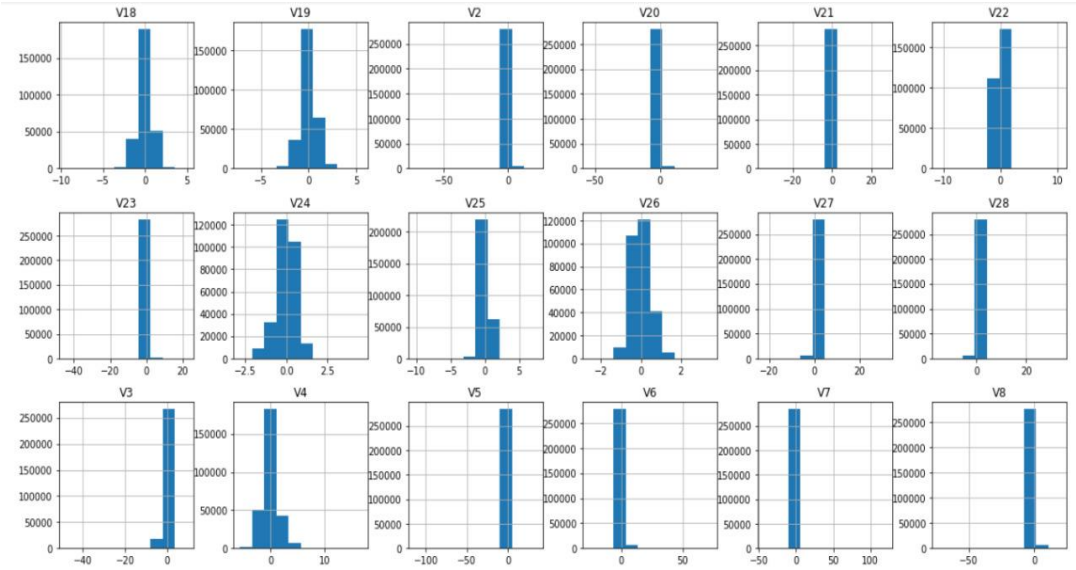
| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V14 | V15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 | -0.551600 | -0.617801 | -0.991390 | -0.311169 | 1.468177 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 | 1.612727 | 1.065235 | 0.489095 | -0.143772 | 0.635558 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 | 0.624501 | 0.066084 | 0.717293 | -0.165946 | 2.345865 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 | -0.226487 | 0.178228 | 0.507757 | -0.287924 | -0.631418 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 | -0.822843 | 0.538196 | 1.345852 | -1.119670 | 0.175121 |

| V16 | V17 | V18 | V19 | V20 | V21 | V22 | V23 | V24 | V25 | V26 | V27 | V28 | Amount | Class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -0.470401 | 0.207971 | 0.025791 | 0.403993 | 0.251412 | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0.128539 | -0.189115 | 0.133558 | -0.021053 | 149.62 | 0 |
| 0.463917 | -0.114805 | -0.183361 | -0.145783 | -0.069083 | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0.167170 | 0.125895 | -0.008983 | 0.014724 | 2.69 | 0 |
| -2.890083 | 1.109969 | -0.121359 | -2.261857 | 0.524980 | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0.327642 | -0.139097 | -0.055353 | -0.059752 | 378.66 | 0 |
| -1.059647 | -0.684093 | 1.965775 | -1.232622 | -0.208038 | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0.647376 | -0.221929 | 0.062723 | 0.061458 | 123.50 | 0 |
| -0.451449 | -0.237033 | -0.038195 | 0.803487 | 0.408542 | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0.206010 | 0.502292 | 0.219422 | 0.215153 | 69.99 | 0 |

**Histograms for various data columns:** Histograms group the data in bins and is the fastest way to get idea about the distribution of each attribute in dataset. It provides us a count of the number of observations in each bin created for visualization. From the shape of the bin, we can easily observe the distribution i.e. weather it is Gaussian, skewed or exponential. Histograms also help us to see possible outliers. We will be using *hist()* function on *Pandas* DataFrame to generate histograms and *matplotlib* for plotting them. The output is given as below:

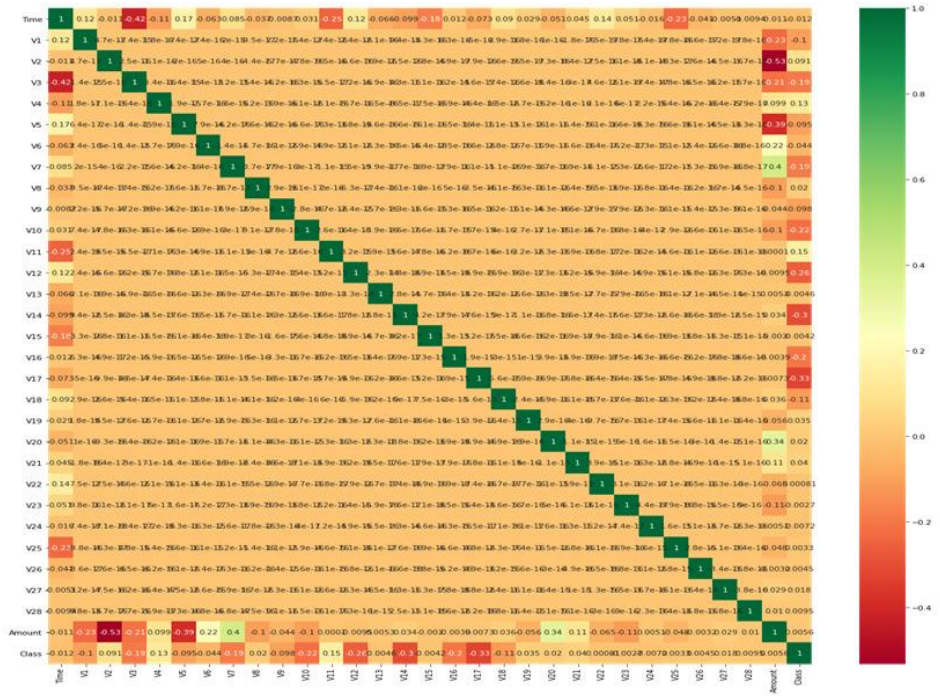



The above output shows that it created the histogram for each attribute in the dataset.

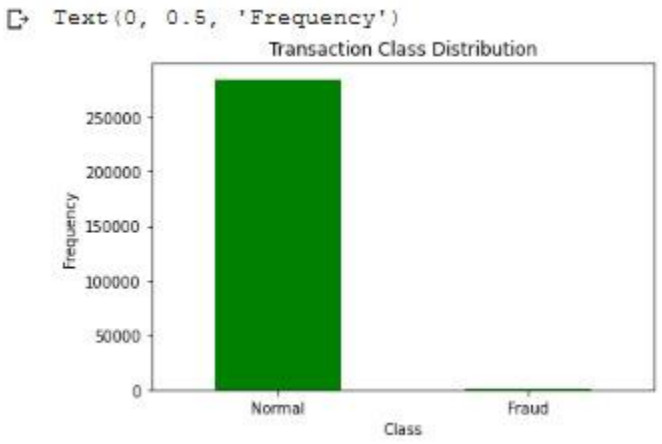



And from the above diagram we can observe that V11,V13, V15, V24 and V26 are nearer to the Gaussian curve model.

**Correlation Heat Map:** The most important argument in the function is to input the data since the end goal is to plot a correlation. A .corr() method will be added to the data and passed as the first argument. Interpreting the insights by just using the first argument is sufficient. For an even easier interpretation, an argument called annot=True should be passed as well, which helps display the correlation coefficient. cmap argument is used to specify the color map to differentiate the values in a easy way. Here we used "RdYlGn" for the color map.

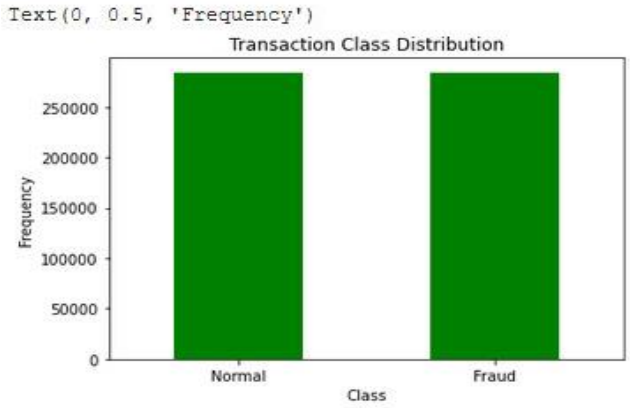


Imbalanced Data: This problem is predominant in scenarios where anomaly detection of credit card fraudulent. In this situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate. The data which is imbalanced is shown in the picture below:




Here in the above it shows that very less number of fraudulent transactions are present in given data i.e., there is only 0.17% percent of transactions that are fraudulent. Hence to balance the data oversampling is used. Here we used Sythetic Minority Over-sampling Technique. Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

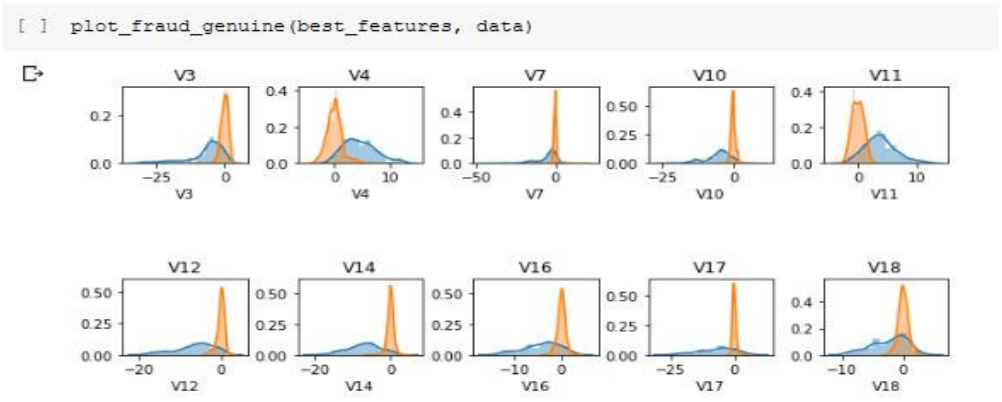
Whereas Synthetic Minority Over-Sampling Technique for imbalanced data technique is followed to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to the original dataset. The new dataset is used as a sample to train the classification models. The data is now balanced by using this technique which is shown below:




From the above picture we observe that number of fraudulent transactions are half of the total transactions. Hence the data is balanced.

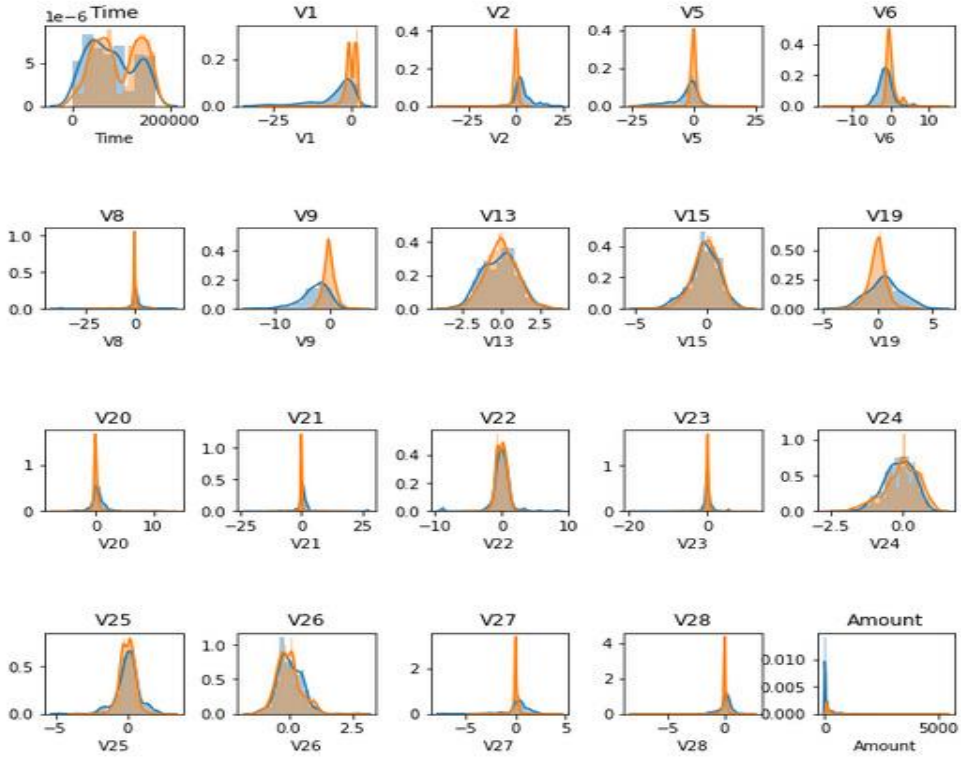
**Feature Selection:** Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features. By performing the feature selection process, we can reduce the over fitting, improve accuracy, and reduce the training time.

For this feature selection Univariate Selection is used. Univariate feature selection works by selecting the best features based on univariate statistical tests. We compare each feature to the target variable, to see whether there is any statistically significant relationship between them. It is also called analysis of variance (ANOVA). When we analyze the relationship between one feature and the target variable, we ignore the other features. That is why it is called 'univariate'. Each feature has its test score.



From the above diagram, we understand that V3, V4, V7, V10, V11, V12, V14, V16, V17, V18 are best features.

```
plot_fraud_genuine(bad_features, data)
```




And from the above diagram Time, V1, V2, V5, V6, V8, V9, V13, V15, V19, V20, V21, V22, V23, V24, V25, V26, V27, V28 and Amount are bad features.

# MODELS

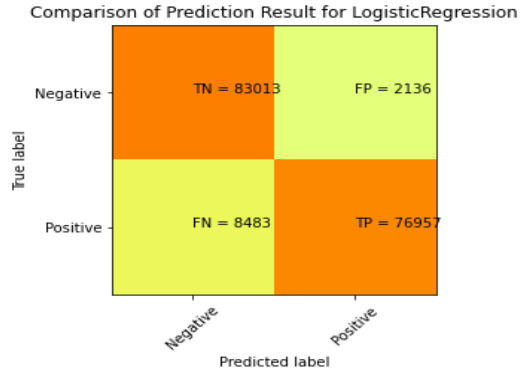
## Logistic Regression

A Logistic Regression algorithm is used to create a binary classifier that is optimized on our training dataset. The Logistic Regression function from sklearn library is used to create and train our classifier.

Then the model was tested for accuracy on the test dataset and a Confusion Matrix was plotted along with test Accuracy, Precision, Recall and F1Score was reported.

### Results:

The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.




```
8483
Accuracy of the Model LogisticRegression 93.8%
Logistic Regression:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     85149
           1       0.97      0.90      0.94     85440

    accuracy                           0.94    170589
   macro avg       0.94      0.94      0.94    170589
weighted avg       0.94      0.94      0.94    170589
```

## Support Vector Machine

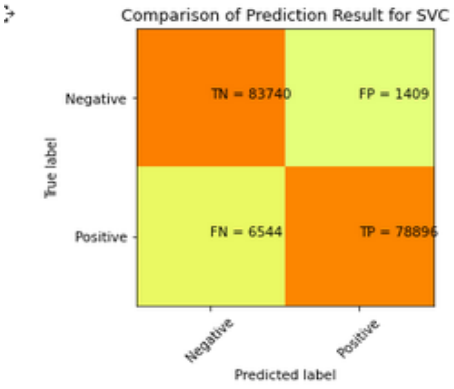
The SVM training algorithm builds a model that assigns a datapoint to one class or the other. The SVM model is the representation of sample points in n dimensional space, mapped so that examples of different class are divided by a clear boundary or gap that is as wide as possible. The test or unseen examples are then mapped to the same space belonging to one category or the other based on which side of the boundary they fall. This boundary solves both linear or nonlinear classification problems based on the kernel methods used for training. A Support Vector Classifier is created using the SVC function of sklearn library.

Then the model was tested for accuracy on the test dataset and a Confusion Matrix was plotted along with test Accuracy, Precision, Recall and F1Score was reported.

Support Vector Machine tries to find a hyperplane that separates two different classes such that the distance from the closest data point to that hyperplane is maximized.

**Results:**

The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.



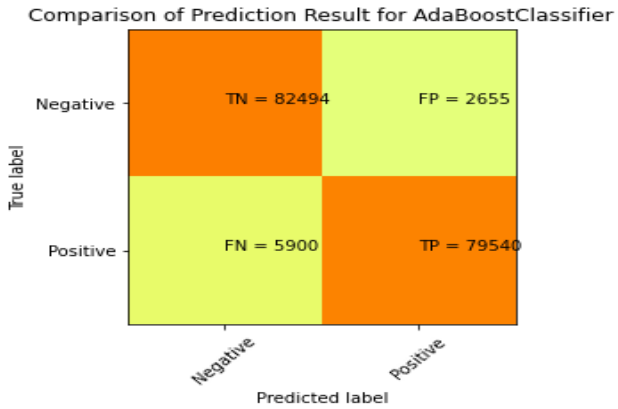

```
6544
Accuracy of the Model SVC 95.3%
SVC:
              precision    recall  f1-score   support

           0       0.93      0.98      0.95     85149
           1       0.98      0.92      0.95     85440

    accuracy                           0.95    170589
   macro avg       0.95      0.95      0.95    170589
weighted avg       0.96      0.95      0.95    170589
```

## AdaBoost Classifier

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights to each of the training samples. Initially, those weights are all set to $w_i=1/N$, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence.

**Results:**

The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.




```
5900
Accuracy of the Model AdaBoostClassifier 95.0%
AdaBoostClassifier:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     85149
           1       0.97      0.93      0.95     85440

    accuracy                           0.95    170589
   macro avg       0.95      0.95      0.95    170589
weighted avg       0.95      0.95      0.95    170589
```

## Decision Tree Classifier

**Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

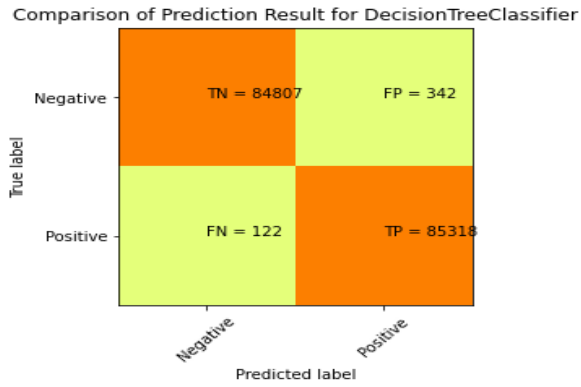
For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable.

**Results:**

The model performance in terms of the Accuracy, Precision, Recall and F1 Score are as follows.




```
122
Accuracy of the Model DecisionTreeClassifier 99.7%
Decision Tree:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85149
           1       1.00      1.00      1.00     85440

    accuracy                           1.00    170589
   macro avg       1.00      1.00      1.00    170589
weighted avg       1.00      1.00      1.00    170589
```

# Conclusion

- From the results obtained, it is observed that Decision Tree classifier is the best classification model which fits for the given data.
- Oversampling is a better option in this project as one class of data is underrepresented minority class in the data sample and this technique can be used to duplicate these results for a more balanced amount of positive results in training.
- The most important thing while making a model is the feature engineering and selection process.One should be able to extract maximum information from the features to make the model more robust and accurate. Feature selection and extraction comes with respect to time and experience. There could be several ways to deal with the information available in the data set.

# References


1. https://www.kaggle.com/mlg-ulb/creditcardfraud
2. http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html#:~:text=Decision%20Tree%20Classifier%20is%20a%20simple%20and%20widely,questions%20about%20the%20attributes%20of%20the%20test%20record.
3. https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/
4. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
5. https://scikit-learn.org/stable/modules/feature_selection.html