A PROJECT REPORT ON

"CREDIT CARD FRAUD DETECTION"

BY

"DIPESH HEMCHANDRA CHAUDHARI"
"NIKHIL BHAGAWAN PATIL"
"PRITAJ VASUDEV BOROLE"

FOR THE

POST GRADUATION DIPLOMA IN
BIG DATA ANALYTICS

SUBMITTED TO,

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, DELHI

MARCH-2022

CDAC, DELHI

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, DELHI

CERTIFICATE

This is to certify that **Dipesh Hemchandra Chaudhari**, **Nikhil Bhagawan Patil** and **Pritaj Vasudev Borole** students of **“CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING, DELHI”** has completed the full time PG Diploma Course with Project titled **“CREDIT CARD FRAUD DETECTION”** at **“CDAC-Delhi”**.

We are submitted satisfactory report in partial fulfilment of the requirement for the award of the Post-Graduation Diploma In Big Data Analytics in Batch March-2022.

(**Dr. Zeeshan Ahmad Khan**)	(**Mr. Ankit Khurana**)
Project Guide	Course Co-ordinator

Date :-

ACKNOWLEDGEMENT

I would like thank to all those who are involved in this endeavour for their kind co-operation for its successful completion. At the outset, I wish to express my sincere gratitude to all those people who have helped me to complete this project in an efficient manner.

I offer my special thanks to my Project guide Dr. Zeeshan Ahmad Khan Sir, Project Co-ordinator, without whose help and support throughout this project would not have been this success.

I am thankful to **Mr. Abhinav Dixit**, Director of CDAC, Delhi for his kind support in all respect during my study. I would like to thank **Mr. Ankit Khurana**, Course Co-ordinator CDAC, who gave opportunity to do this project at an extreme organization. Most of all and more than ever, I would like to thanks my family members for their warmness, support, encouragement, kindness and patience. I am really thankful to all my friends who always advised and motivated me throughout the course.

Nikhil Bhagawan Patil, {PRN No. 220310125010}

Dipesh Hemchandra Chaudhari, {PRN No. 220310125012}

Pritaj Vasudev Borole, {PRN No. 220310125023}

ABSTRACT

Credit card fraud is a severe issue in financial services area. Every year billions of dollars are lost due to credit card fraud. Credit card has been one of the most flourishing financial services by banks over the past years.

It is very essential for credit card companies to identify the fraudulent credit card transactions to ensure that their customers are not being charged for the items that they dint purchase. Data mining plays a major role in detecting fraudulent online credit card transactions and it becomes very challenging due to two major reasons such as the profiles of fraudulent and normal behaviours changes constantly and the credit can't fraud data sets are highly skewed. Such problems can be addressed with machine learning and data science. And this technique is applied on pre-processed and raw data. This mechanism is implemented in python and the performance is evaluated based on the sensitivity, accuracy, Correlation coefficient. It is easy to distinguish between fraudulent and genuine transition. In this process the main focus is on pre-processing and analysing the data sets and algorithms such as forest isolation algorithm and local outlier factor.

In this research work, various machine learning classification techniques and methods are used to analysed and predict the accuracy of credit card fraud detection. Dataset of credit card transactions is sourced from European cardholders containing 2,84,807 transactions. Thus, Logistic Regression, KNN, SVC and Random Forest are used to test the variable in predicting credit fraud detection.

INDEX

1. INTRODUCTION OF CREDIT CARD FRAUD DETECTION

1.1 Description :-

Financial fraud is a growing concern with far reaching consequences in the government, corporate organizations, finance industry. In today's world, high dependency on internet technology has enjoyed increased credit card transactions, but, credit card fraud has also accelerated as online and offline transaction. As credit card transactions become a widespread mode of payment, focus has been given to recent computational methodologies to handle the credit card fraud problem. There are many fraud detection solutions and software which prevent frauds in businesses such as credit card, retail, e-commerce, insurance, and industries.

Data mining technique is one notable and popular methods used in solving credit fraud detection problem. It is impossible to be sheer certain about the true intention and rightfulness behind an application or transaction. In reality, to seek out possible evidences of fraud from the available data, using mathematical algorithms is the best effective option. Fraud detection in credit card is truly the process of identifying those transactions that are fraudulent into two classes of legit class and fraud class transactions. Several techniques are designed and implemented to solve credit card fraud detection such as Genetic Algorithm, Artificial Neural Network, Frequent Item set Mining, Migrating Birds optimization algorithm, comparative analysis of Logistic Regression, SVC, KNN and Random Forest is carried out.

Credit card transaction datasets are rarely available, highly imbalanced and skewed. Optimal feature (variables) selection for the models, suitable metric is most important part of data mining to evaluate performance of techniques on skewed credit card fraud data. A number of challenges are associated with credit card detection, namely fraudulent behaviour profile is dynamic, that is fraudulent transactions tend to look like legitimate ones. Credit card fraud detection performance is greatly affected by type of sampling approach used, selection of variables and detection technique used. In the end, conclusions about results of classifier evaluative testing are made and collated.

From the experiments, the result that has been concluded is that Logistic regression has an accuracy of 94.01% while SVC shows accuracy of 93.73%, KNN with 93.44% and Random forest shows accuracy of 94.58% Random Forest is the best Model.

1.2 Problem Formulation :-

The problem formulation consists of just one sentence and should make it clear to everyone what research problem, you aim to address and to whom and where it is relevant. Problem Formulation for our project: Are the existing techniques used for detecting credit card frauds correctly and accurately providing efficient results or can it be improved using Machine Learning? Thus, our project tries to predict credit-card frauds using Machine Learning as it is believed to provide better results as compared to the existing techniques used to detect these frauds.

1.3 Proposed System :-

These are the proposed techniques used in this project for detecting the frauds in credit card system. The comparison is made for different machine learning algorithms such as Logistic Regression, SVC, Random Forest, to determine which algorithm suits best and can be adapted by credit card merchants for identifying fraud transactions. The Figure shows the architectural diagram for representing the overall system framework.

2. OBJECTIVES OF CREDIT CARD FRAUD DETECTION

2.1 Objectives :-

1. To implement machine learning algorithms to detect credit card fraud detection with respect to time and amount of transaction.
2. Finding hidden and implicit correlations in data.
3. The Reduced number of Verification measures.
4. Real Time Processing.

2.2 Benefits :-

Machine learning (ML) models are much better than conventional fraud detection models. They can recognise thousands of patterns from large datasets. ML offers an insight into how users behave by understanding their app usage, payments, and transaction methods. Some of the benefits of fraud detection using ML are as follows:

Faster detection:-

A machine learning model can quickly identify any drifts from regular transactions and user behaviours in real time. By recognising anomalies, such as a sudden increase in transactional amount or location change, ML algorithms can minimise the risk of fraud and ensure more secure transactions.

Higher accuracy:-

Conventional fraud detection techniques cause errors at the payment gateways that sometimes result in genuine customers being blocked. With sufficient training data and insights, ML models can achieve higher accuracy and precision, reducing these errors along with the time required to be spent on performing manual analysis.

Improved efficiency with larger data:-

Once an algorithm picks up different transactional patterns and behaviours, it can efficiently work with large datasets to separate authentic payments from fraudulent ones. The models can analyse huge amounts of data in seconds while offering real-time insights for improved decision-making capabilities.

3. SCOPE OF CREDIT CARD FRAUD DETECTION

Credit card fraud detection is a very popular but also a difficult problem to solve. Firstly, due to issue of having only a limited amount of data, credit card makes it challenging to match a pattern for dataset. Secondly, there can be many entries in dataset with truncations of fraudsters which also will fit a pattern of legitimate behaviour. Also, the problem has many constraints. Firstly, data sets are not easily accessible for public and the results of researches are often hidden and censored, making the results inaccessible and due to this it is challenging to benchmarking for the models built. Secondly, the improvement of methods is more difficult by the fact that the security concern imposes a limitation to exchange of ideas and methods in fraud detection, and especially in credit card fraud detection. Lastly, the data sets are continuously evolving and changing making the profiles of normal and fraudulent behaviours always different, that is, a legit transaction in the past may be a fraud in present or vice versa. We evaluate four advanced data mining approaches, Logistic Regression, K-Nearest Neighbours, Support Vector Classifiers and Random Forests and then a collative comparison is made to evaluate which model performed best.

4. SOFTWARE REQUIREMENT OF CREDIT CARD FRAUD DETECTION

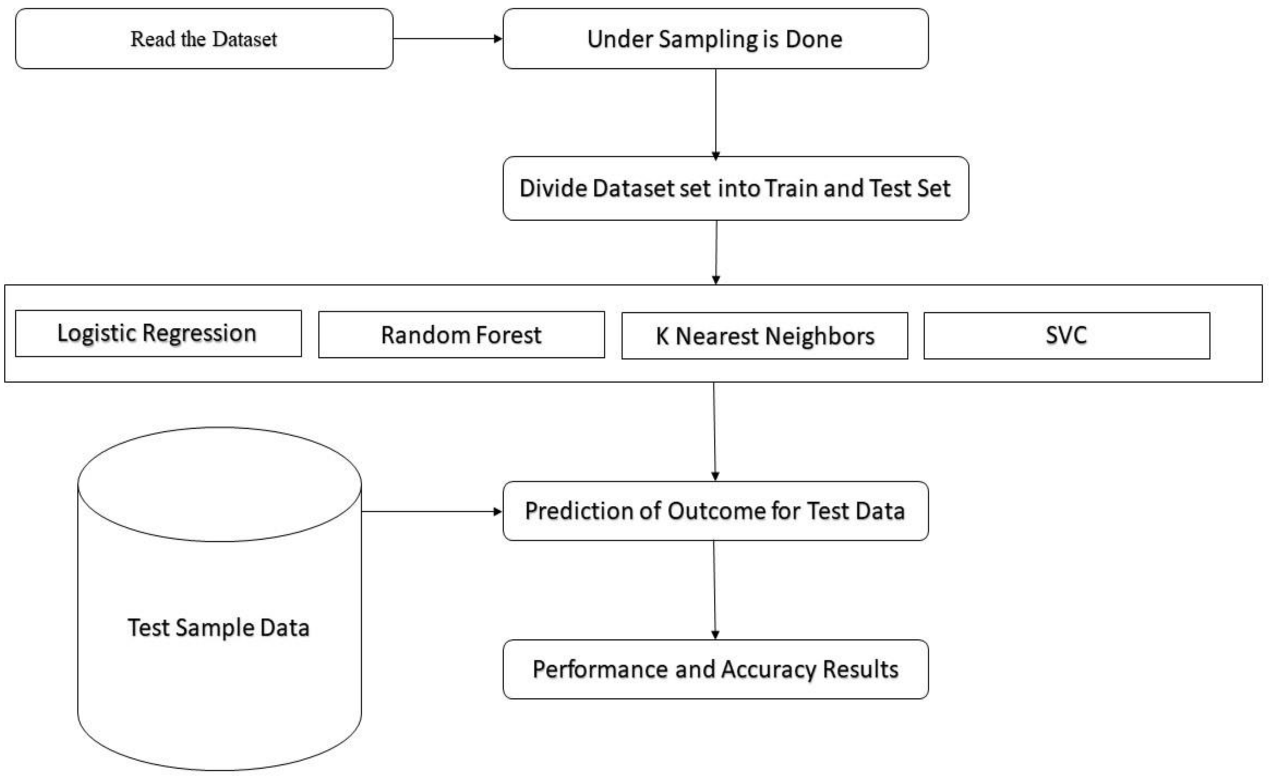
1.	Languages	:- Python-3
2.	Operating System	:- Windows, Linux, etc.
3.	Back End Software	: - Anaconda, Jupyter Notebook.

5. HARDWARE REQUIREMENT OF CREDIT CARD FRAUD DETECTION

1.	CPU	: - Intel Pentium IV 600MHz
2.	Hard Disk Space	: - 20 GB or More
3.	Memory	: - 4GB RAM

6. TESTING STRATEGY OF CREDIT CARD FRAUD DETECTION

Algorithm Steps :-	
Step – 1:	Read The Dataset.
Step – 2:	Random Sampling is done on the data set to make it Balanced**.**
Step – 3:	Divide the Dataset into two parts i.e.,Train dataset and test dataset.
Step – 4:	Accuracy and Performance Metrics has been calculated to know the Efficiency for Different Algorithms.
Step – 5:	The retrieve the best algorithm based on Efficiency for the dataset.


Read the Dataset
Under Sampling is Done
Divide Dataset set into Train and Test Set
Logistic Regression
Random Forest
K Nearest Neighbors
SVC
Test Sample Data
Prediction of Outcome for Test Data
Performance and Accuracy Results

--Testing Strategy Architecture--

7. ALL MODULES AND DESCRIPTION OF CREDIT CARD FRAUD DETECTION

7.1 Algorithms :-

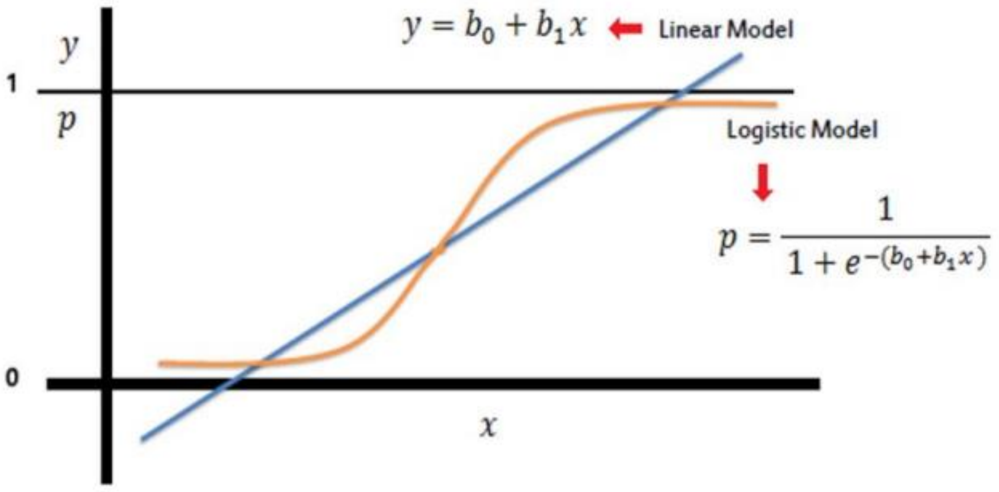
Ability of system to automatically learn and improve from experience without being explicitly programmed is called machine learning and it focuses on the development of computer programs that can access data and use it to learn by themselves. And classifier can be stated as an algorithm that is used to implement classification especially in concrete implementation, it also refers to a mathematical function implemented by algorithm that will map input data into category. It is an instance of supervised learning i.e. where training set of correctly identified observations is available.

A. Logistic Regression: -

Logistic Regression is a supervised classification method that returns the probability of binary dependent variable that is predicted from the independent variable of dataset i.e. logistic regression predicts the probability of an outcome which has two values, either zero or one, no or yes and false or true. Logistic regression has similarities to linear regression, but, in linear regression a straight line is obtained, logistic regression shows a curve. The use of one or several predictors or independent variable is on what prediction is based, logistic regression produces logistic curves which plots the values between zero and one. Logistic Regression is a regression model where the dependent variable is categorical and analyses the relationship between multiple independent variables. There are many types of logistic regression model such as binary logistic model, multiple logistic model, binomial logistic models. Binary Logistic Regression model is used to estimate the probability of a binary response based on one or more predictors.

$$p = \frac{e^{\alpha+\beta_n X}}{1 + e^{\alpha+\beta_n X}}$$

Above equation represents the logistic regression in mathematical form.

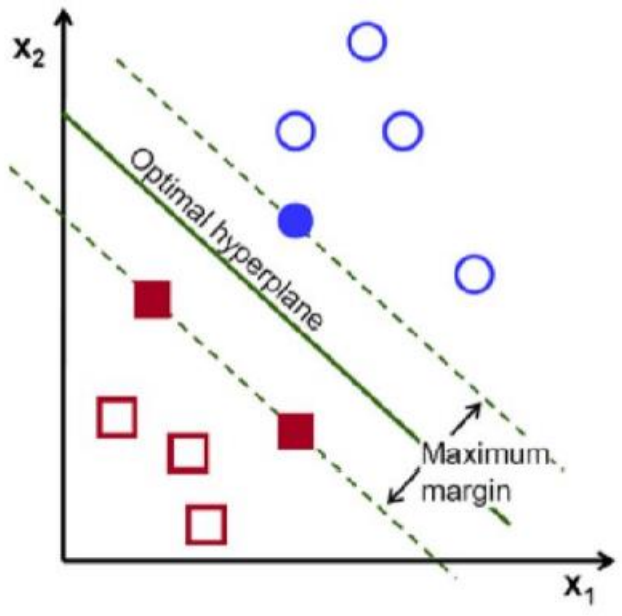


--Logistic Curve--

This graph shows the difference between linear regression and logistic regression where logistic regression shows a curve but linear regression represents a straight line.

B. <u>SVM Model (Support Vector Machine)</u> :-

SVM is a one of the popular machine learning algorithm for regression, classification. It is a supervised learning algorithm that analyses data used for classification and regression. SVM modeling involves two steps, firstly to train a data set and to obtain a model & then, to use this model to predict information of a testing data set. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane where SVM model represents the training data points as points in space and then mapping is done so that the points which are of different classes are divided by a gap that is as wide as possible. Mapping is done in to the same space for new data points and then predicted on which side of the gap they fall.

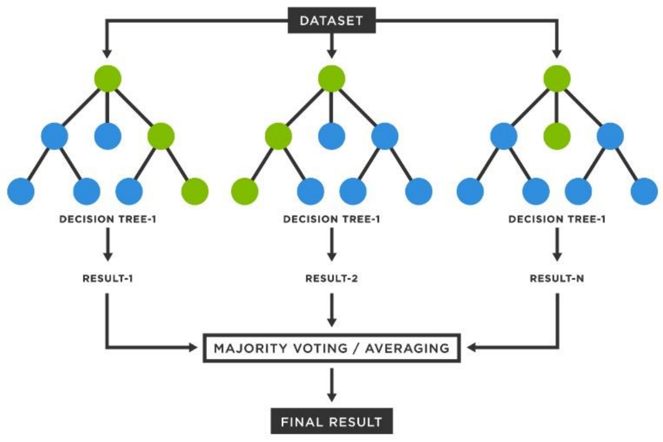


--SVM Model Graph--

In SVM algorithm, plotting is done as each data item is taken as a point in n-dimensional space where n is number of features, with the value of each feature being the value of a particular coordinate. Then, classification is performed by locating the hyperplane that separates the two classes very well.

C. <u>Random Forest</u> :-

Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of overfitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provides a good estimate of the generalization error and to be resistant to overfitting. Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.



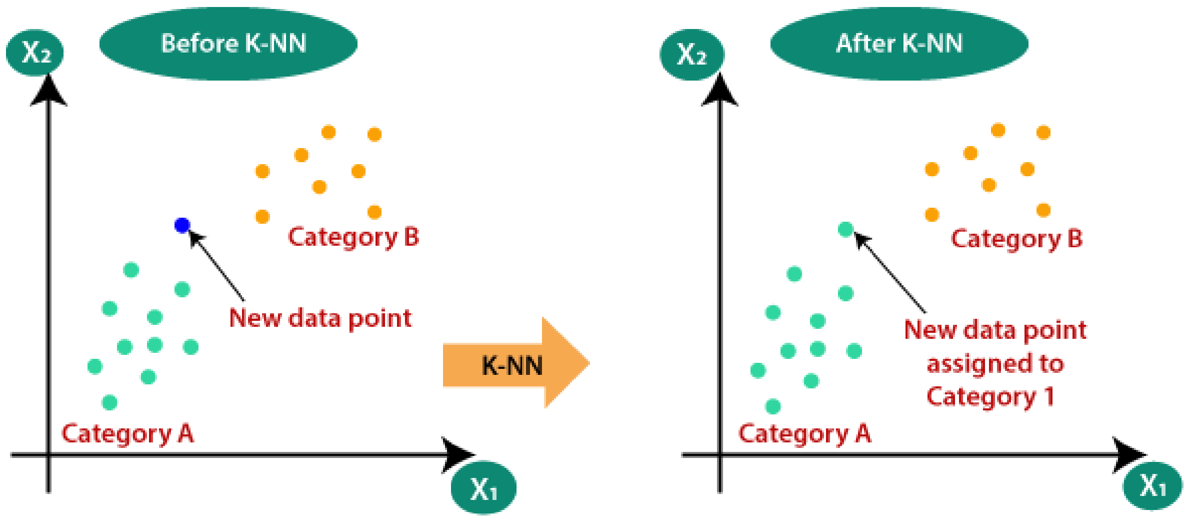
--Random Forest Model Design--

D. <u>K-Nearest Neighbour Classifier</u> :-

The k-nearest neighbour is an instance based learning which carries out its classification based on a similarity measure, like Euclidean, Manhattan or Minkowski distance functions. The first two distance measures work well with continuous variables while the third suits categorical variables. The Euclidean distance measure is used in this study for the kNN classifier. The Euclidean distance (Dij) between two input vectors (Xi, Xj) is given by:

$$D_{ij} = \sqrt{\sum_{k=1}^{n} \left(X_{ik} - X_{jk}\right)^2} \qquad k=1,2,\ldots,n$$

For every data point in the dataset, the Euclidean distance between an input data point and current point is calculated. These distances are sorted in increasing order and k items with lowest distances to the input data point are selected. The majority class among these items is found and the classifier returns the majority class as the classification for the input point. Parameter tuning for k is carried out for k = 1, 3, 5, 7, 9, 11 and k = 3 showed optimal performance. Thus, value of k = 3 is used in the classifier.



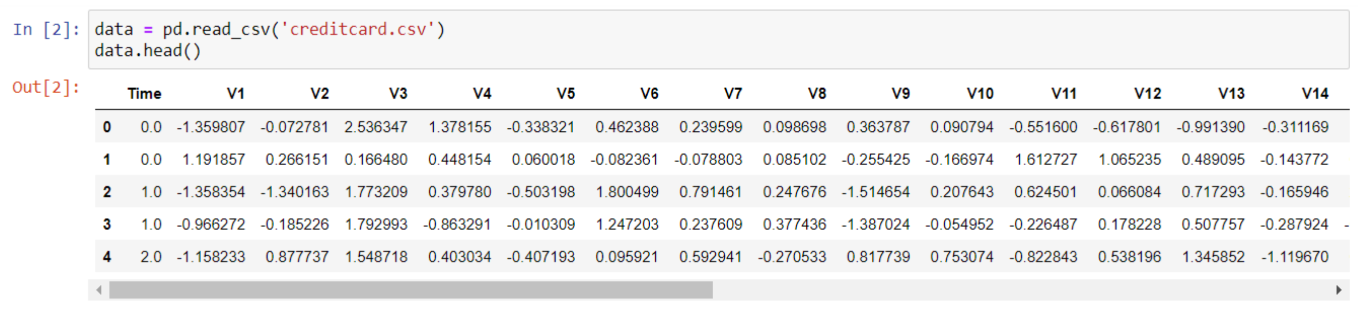
--KNN Model Graph--

7.2 Working Of The Project :-

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

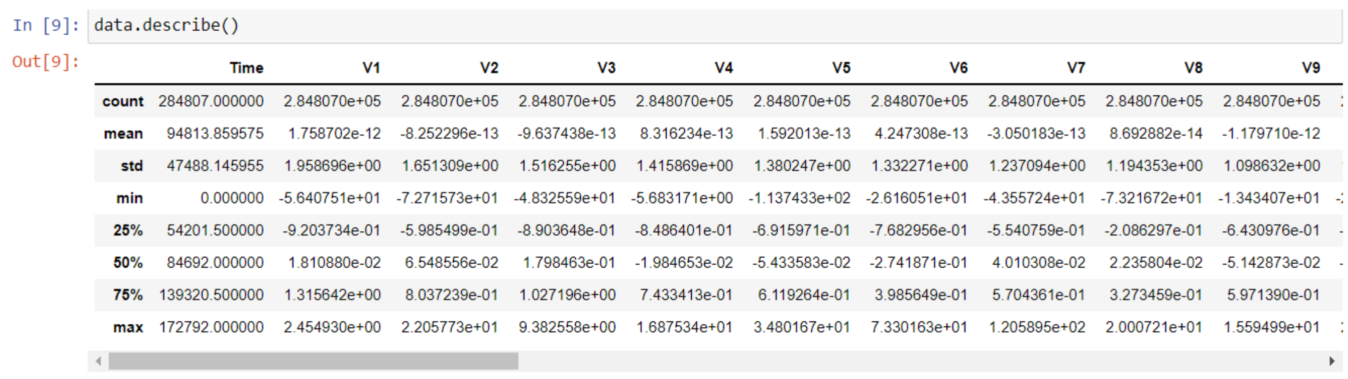
Dataset

The datasets contain 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 numeric 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 12 first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. There are no "Null" values, so we don't have to work on ways to replace values.



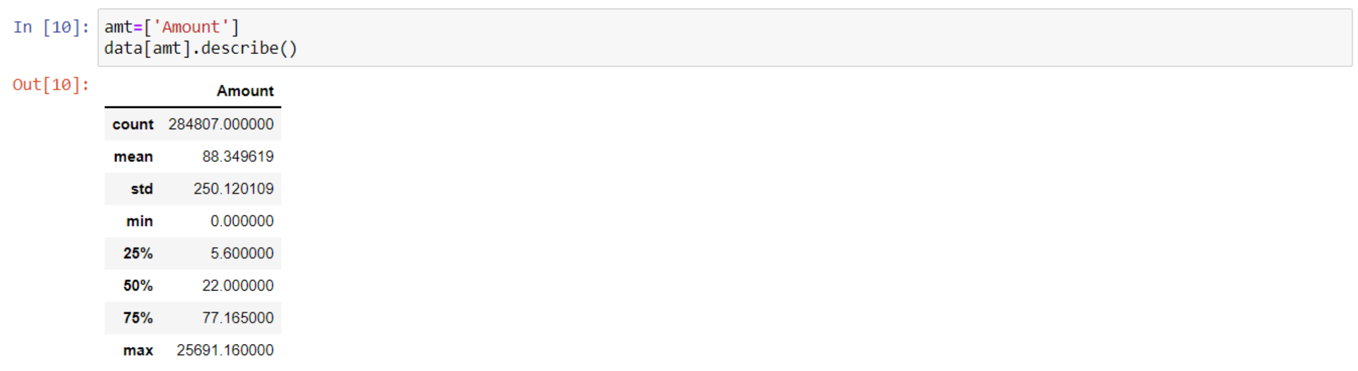
```
In [2]: data = pd.read_csv('creditcard.csv')
        data.head()
Out[2]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
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	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
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
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
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



```
In [9]: data.describe()
Out[9]:
```

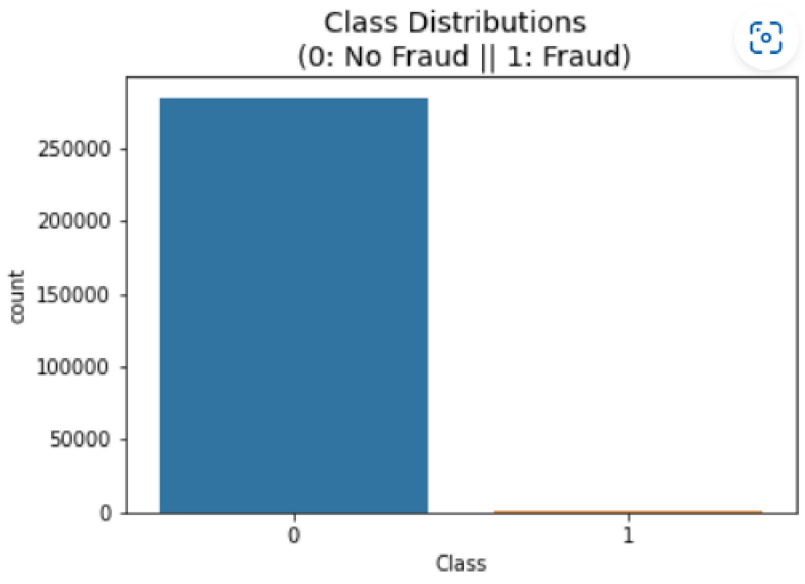
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.758702e-12	-8.252296e-13	-9.637438e-13	8.316234e-13	1.592013e-13	4.247308e-13	-3.050183e-13	8.692882e-14	-1.179710e-12
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01



```
In [10]: amt=['Amount']
         data[amt].describe()
Out[10]:
```

	Amount
count	284807.000000
mean	88.349619
std	250.120109
min	0.000000
25%	5.600000
50%	22.000000
75%	77.165000
max	25691.160000

Why directly running prediction on the dataset is not a good idea?



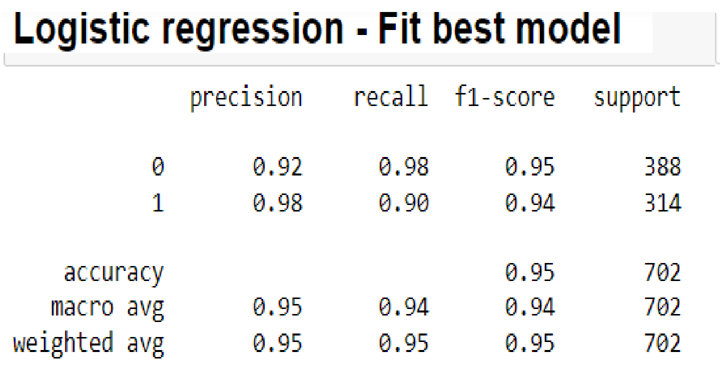
No frauds **99.83%** of dataset

Frauds **0.17%** of dataset

Notice how imbalanced is our original dataset is! Most of the transactions are non-fraud. If we use this data frame as the base for our predictive models and analysis, we might get a lot of errors and our algorithms will probably overfit since it will "assume" that most transactions are not fraud. But we don't want our model to assume, we want our model to detect patterns that give signs of fraud!

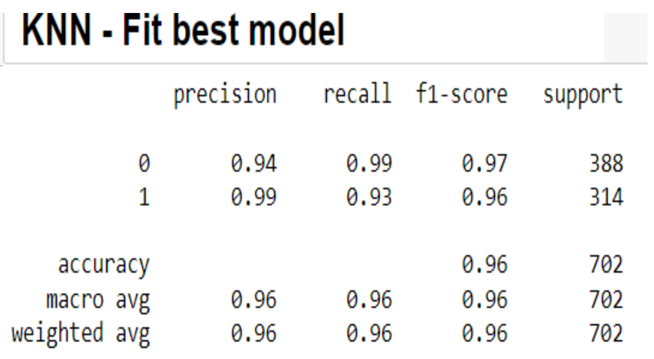
Credit card dataset is highly imbalanced dataset because it carries more legitimate transactions as compared to the fraudulent one. That means prediction will get very high accuracy score without detecting a fraud transaction. When we apply any classification algorithm directly on the dataset we get the following results.

Directly applying the classification:



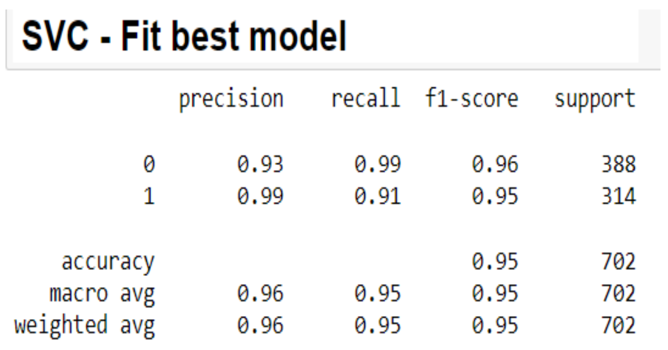
Logistic regression - Fit best model

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       388
           1       0.98      0.90      0.94       314

    accuracy                           0.95       702
   macro avg       0.95      0.94      0.94       702
weighted avg       0.95      0.95      0.95       702
```



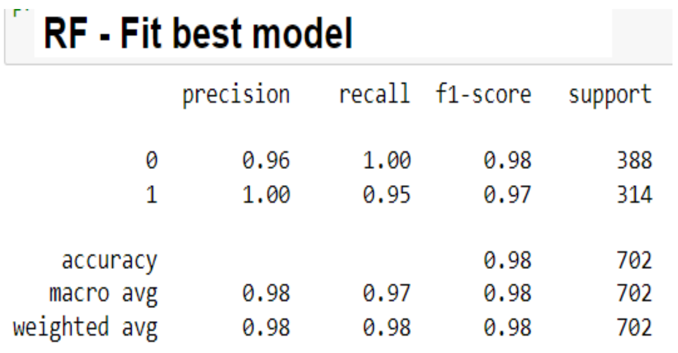
KNN - Fit best model

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       388
           1       0.99      0.93      0.96       314

    accuracy                           0.96       702
   macro avg       0.96      0.96      0.96       702
weighted avg       0.96      0.96      0.96       702
```



SVC - Fit best model

```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96       388
           1       0.99      0.91      0.95       314

    accuracy                           0.95       702
   macro avg       0.96      0.95      0.95       702
weighted avg       0.96      0.95      0.95       702
```

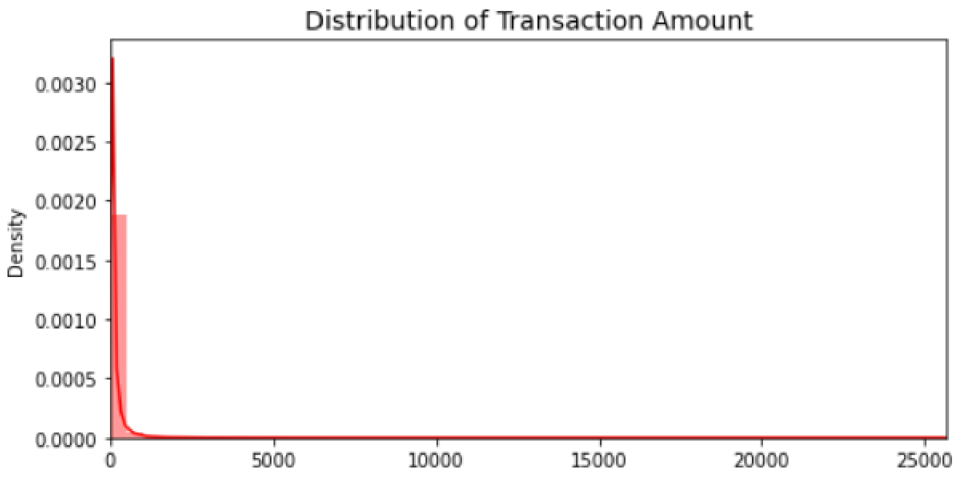


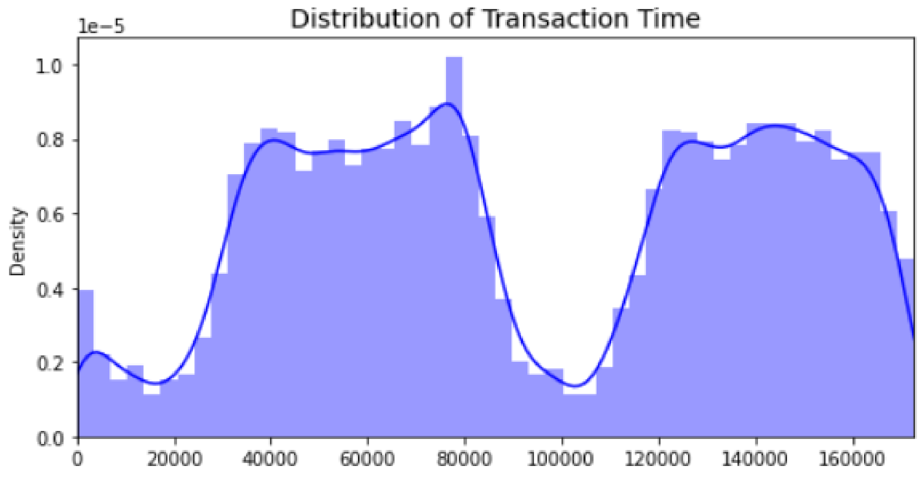
RF - Fit best model

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       388
           1       1.00      0.95      0.97       314

    accuracy                           0.98       702
   macro avg       0.98      0.97      0.98       702
weighted avg       0.98      0.98      0.98       702
```

The results deal with a problem called accuracy paradox. The ACCURACY PARADOX is the paradoxical finding that accuracy is not a good metric for predictive models when classifying in predictive analytics. This is because a simple model may have a high level of accuracy but be too crude to be useful. For example, if the incidence of category A is dominant, being found in 99% of cases, then predicting that every case is category A will have an accuracy of 99%. Precision and recall are better measures in such cases. The underlying issue is that there is a class imbalance between the positive class and the negative class. Prior probabilities for these classes need to be accounted for in error analysis. Precision and recall help, but precision too can be biased by very unbalanced class priors in the test sets.

Also, by seeing the distributions, we can have an idea how skewed are these features. There are techniques that can help the distributions be less skewed which will be implemented in this report in the future.





--Distribution of Transaction Amount and Time--

To handle this kind of problem one better way is to class distribution, i.e., sampling minority classes. In sampling minority, class training example can be increased in proportion to the majority class to raise the chance of correct prediction by the algorithm.

<u>Scaling and Distributing</u> :-

In this phase, we will first scale the columns comprise of Time and Amount. Time and amount should be scaled as the other columns. On the other hand, we need to also create a subsample of the data frame in order to have an equal amount of Fraud and Non-Fraud cases, helping our algorithms better understand patterns that determines whether a transaction is a fraud or not.

In this scenario, our subsample will be a data frame with a 50/50 ratio of fraud and non-fraud transactions. Meaning our sub-sample will have the same amount of fraud and non-fraud transactions.

We need to create a sub sample because we saw that the original data frame was heavily imbalanced! Using the original data frame will cause the following issues:

1. Overfitting: Our classification models will assume that in most cases there are no frauds! What we want for our model is to be certain when a fraud occurs.

2. Wrong Correlations: Although we don't know what the "V" features stand for, it will be useful to understand how each of this feature influence the result (Fraud or No Fraud) by having an imbalance data frame we are not able to see the true correlations between the class and features.

We use RobustScalar to scale time and amount, the reason being that they scale features using statistics that are rob ust to outliers.

This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). The IQR is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile).

Cantering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Median and interquartile range are then stored to be used on later data using the transform method. Standardization of a dataset is a common requirement for many machine learning estimators. Typically, this is done by removing the mean and scaling to unit variance.

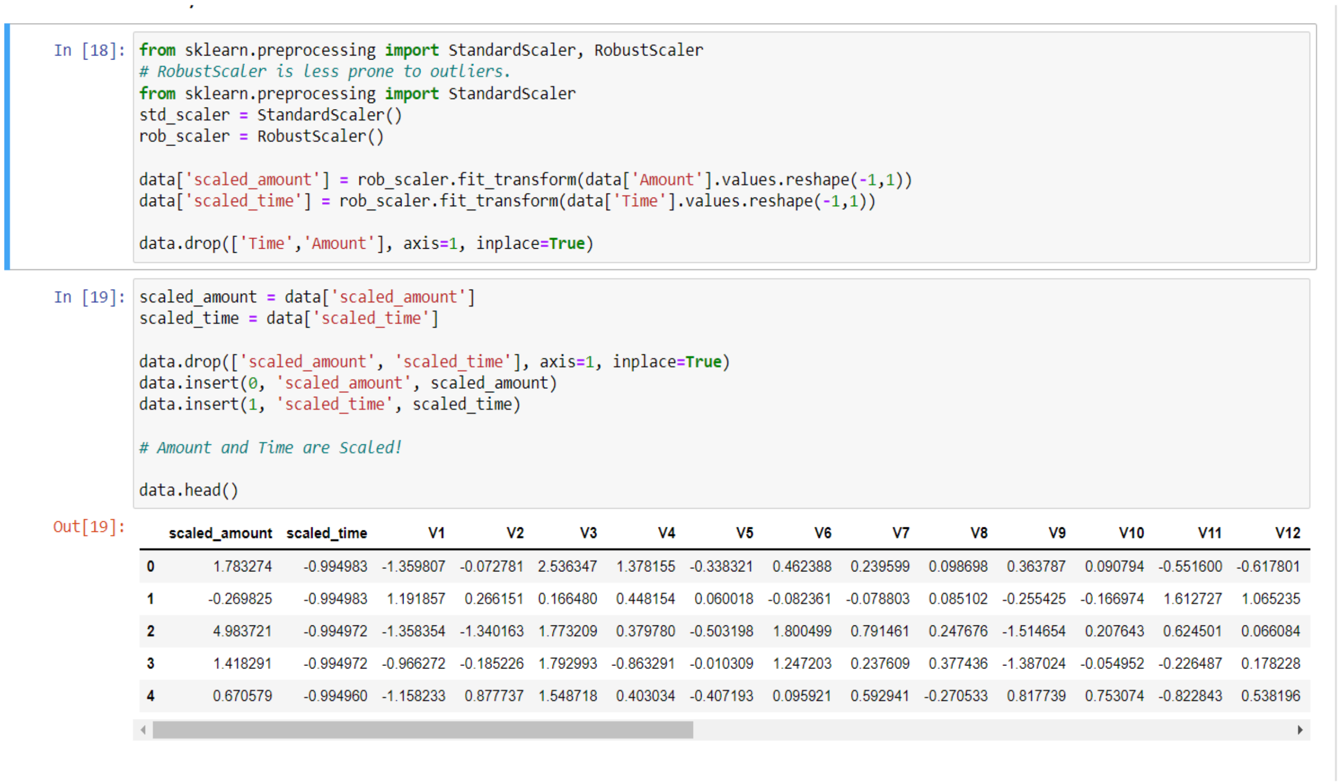
However, outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results.

As RobustScaler, QuantileTransformer is robust to outliers in the sense that adding or removing outliers in the training set will yield approximately the same transformation on held out data. But contrary to RobustScaler, QuantileTransformer will also automatically collapse any outlier by setting them to the a priori defined range boundaries (0 and 1). Splitting the Data (Original DataFrame).

Summary:

- There are 492 cases of fraud in our dataset so we can randomly get 492 cases of non-fraud to create our new sub data frame.
- We concatenate the 492 cases of fraud and non-fraud, creating a new sub-sample.

Scaled amount and scaled time are the columns with scaled values:



```
In [18]: from sklearn.preprocessing import StandardScaler, RobustScaler
         # RobustScaler is less prone to outliers.
         from sklearn.preprocessing import StandardScaler
         std_scaler = StandardScaler()
         rob_scaler = RobustScaler()

         data['scaled_amount'] = rob_scaler.fit_transform(data['Amount'].values.reshape(-1,1))
         data['scaled_time'] = rob_scaler.fit_transform(data['Time'].values.reshape(-1,1))

         data.drop(['Time','Amount'], axis=1, inplace=True)
```

```
In [19]: scaled_amount = data['scaled_amount']
         scaled_time = data['scaled_time']

         data.drop(['scaled_amount', 'scaled_time'], axis=1, inplace=True)
         data.insert(0, 'scaled_amount', scaled_amount)
         data.insert(1, 'scaled_time', scaled_time)

         # Amount and Time are Scaled!

         data.head()
```

Out[19]:

	scaled_amount	scaled_time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
0	1.783274	-0.994983	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801
1	-0.269825	-0.994983	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235
2	4.983721	-0.994972	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084
3	1.418291	-0.994972	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228
4	0.670579	-0.994960	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196

Under Sampling :-

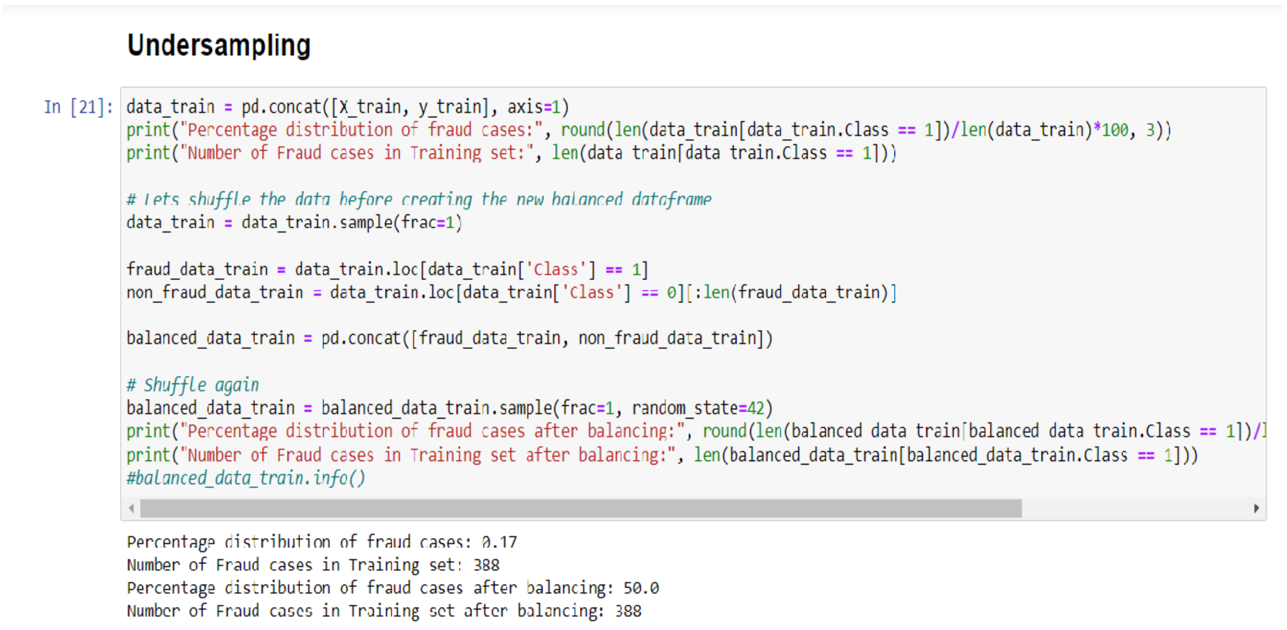
In this phase of the project we will implement "Under Sampling" which basically consists of removing data in order to have a more balanced dataset and thus avoiding our models to overfitting.

● The first thing we have to do is determine how imbalanced is our class (use "value_counts()" on the class column to determine the amount for each label)

● Once we determine how many instances are considered fraud transactions (Fraud = "1") , we should bring the non-fraud transactions to the same amount as fraud transactions (assuming we want a 50/50 ratio), this will be equivalent to 492 cases of fraud and 492 cases of non-fraud transactions.

● After implementing this technique, we have a sub-sample of our data frame with a 50/50 ratio with regards to our classes. Then the next step we will implement is to shuffle the data to see if our models can maintain a certain accuracy every time, we run this script.

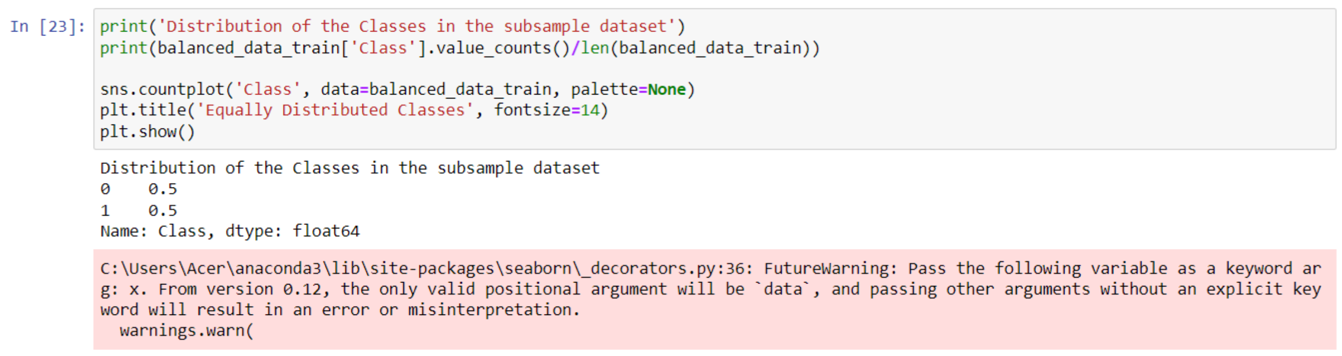
Note: The main issue with "Random Under-Sampling" is that we run the risk that our classification models will not perform as accurate as we would like to since there is a great deal of information loss (bringing 492 non-fraud transaction from 284,315 non-fraud transaction)



Undersampling

```
In [21]: data_train = pd.concat([X_train, y_train], axis=1)
         print("Percentage distribution of fraud cases:", round(len(data_train[data_train.Class == 1])/len(data_train)*100, 3))
         print("Number of Fraud cases in Training set:", len(data train[data train.Class == 1]))

         # Lets shuffle the data before creating the new balanced dataframe
         data_train = data_train.sample(frac=1)

         fraud_data_train = data_train.loc[data_train['Class'] == 1]
         non_fraud_data_train = data_train.loc[data_train['Class'] == 0][:len(fraud_data_train)]

         balanced_data_train = pd.concat([fraud_data_train, non_fraud_data_train])

         # Shuffle again
         balanced_data_train = balanced_data_train.sample(frac=1, random_state=42)
         print("Percentage distribution of fraud cases after balancing:", round(len(balanced data train[balanced data train.Class == 1])/l
         print("Number of Fraud cases in Training set after balancing:", len(balanced_data_train[balanced_data_train.Class == 1]))
         #balanced_data_train.info()
```

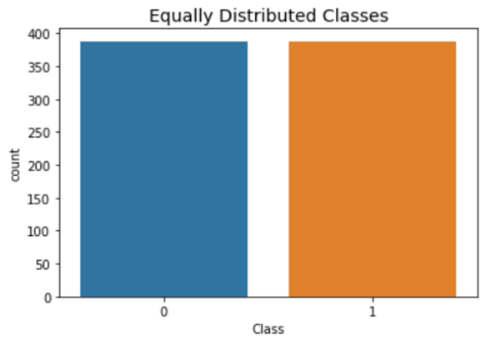
```
Percentage distribution of fraud cases: 0.17
Number of Fraud cases in Training set: 388
Percentage distribution of fraud cases after balancing: 50.0
Number of Fraud cases in Training set after balancing: 388
```



```
In [23]: print('Distribution of the Classes in the subsample dataset')
         print(balanced_data_train['Class'].value_counts()/len(balanced_data_train))

         sns.countplot('Class', data=balanced_data_train, palette=None)
         plt.title('Equally Distributed Classes', fontsize=14)
         plt.show()
```

```
Distribution of the Classes in the subsample dataset
0    0.5
1    0.5
Name: Class, dtype: float64
```

```
C:\Users\Acer\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

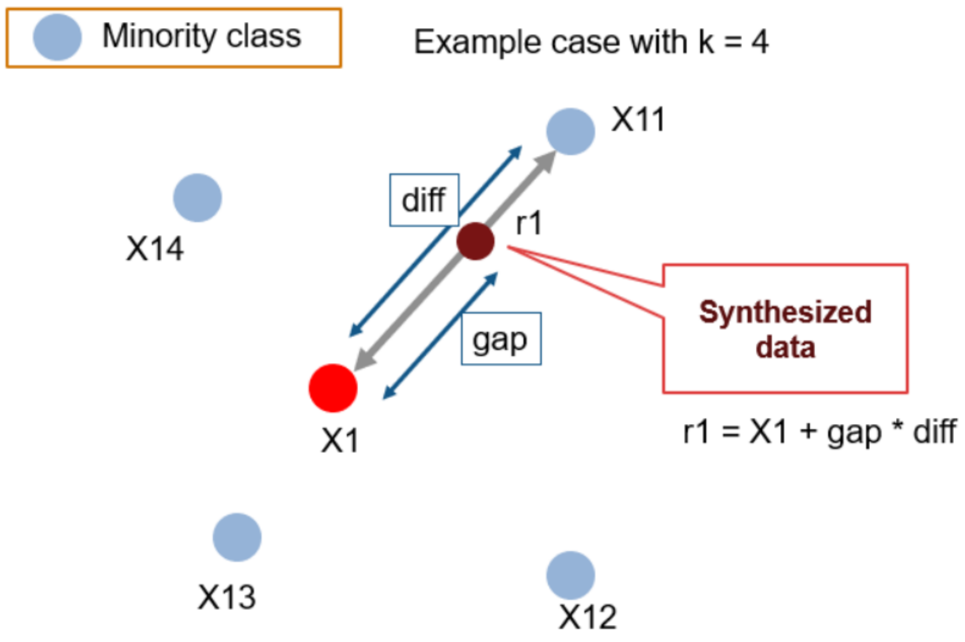


Now that we have our dataframe correctly balanced, we can go further with our analysis and data pre-processing.

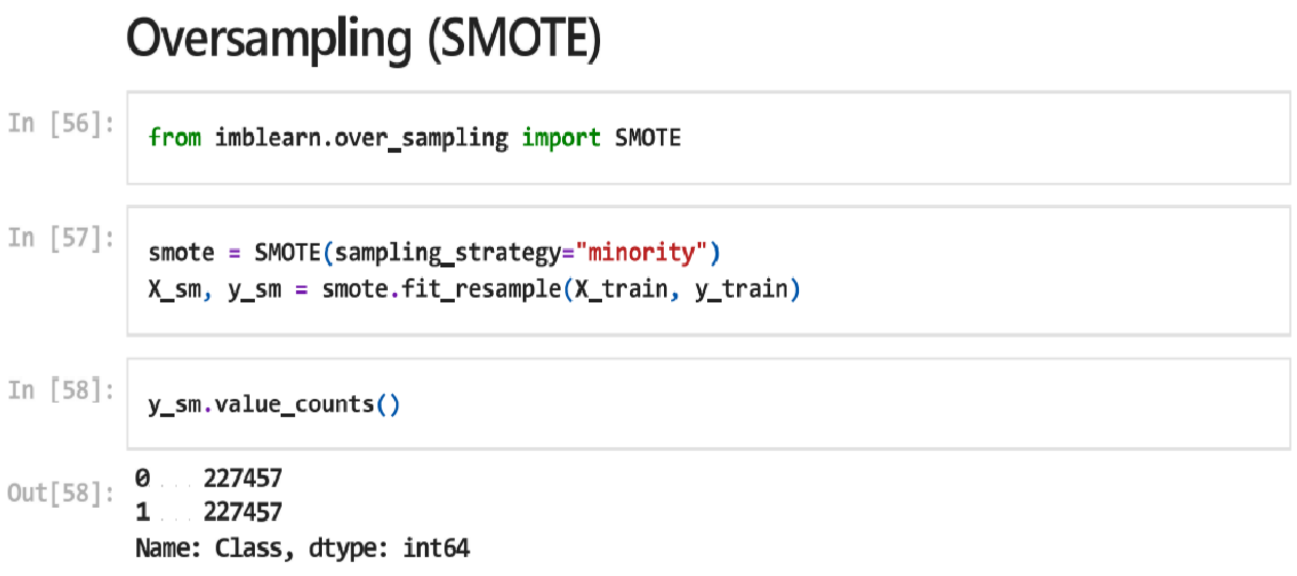
SMOTE (Synthetic Minority Oversampling Technique):-

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

At first the total no. of oversampling observations, N is set up. Generally, it is selected such that the binary class distribution is 1:1. But that could be tuned down based on need. Then the iteration starts by first selecting a positive class instance at random. Next, the KNN's (by default 5) for that instance is obtained. At last, N of these K instances is chosen to interpolate new synthetic instances. To do that, using any distance metric the difference in distance between the feature vector and its neighbors is calculated. Now, this difference is multiplied by any random value in (0,1] and is added to the previous feature vector. This is pictorially represented below:



--SMOTE--



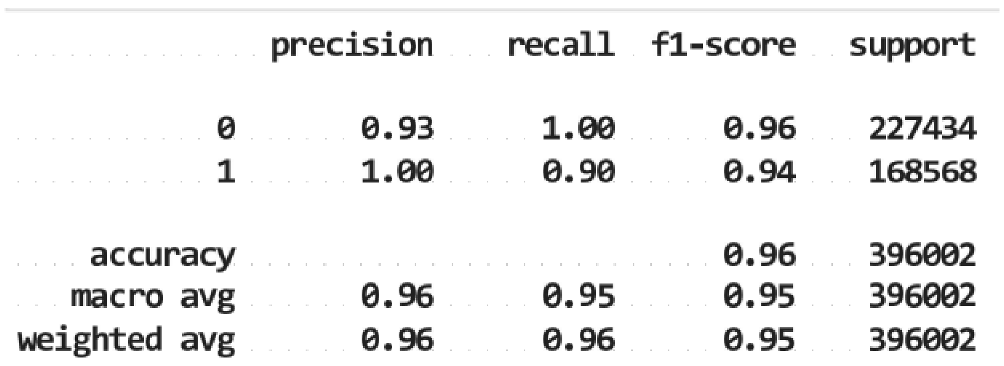
Oversampling (SMOTE)

```
In [56]: from imblearn.over_sampling import SMOTE
```

```
In [57]: smote = SMOTE(sampling_strategy="minority")
         X_sm, y_sm = smote.fit_resample(X_train, y_train)
```

```
In [58]: y_sm.value_counts()
```

```
Out[58]: 0    227457
         1    227457
         Name: Class, dtype: int64
```

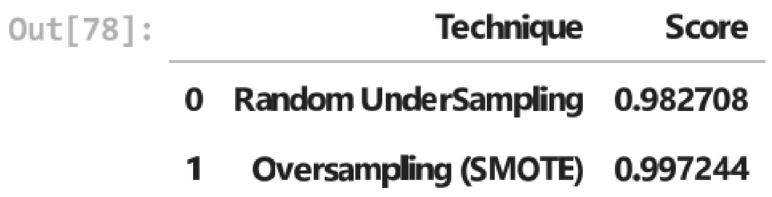


```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96    227434
           1       1.00      0.90      0.94    168568

    accuracy                           0.96    396002
   macro avg       0.96      0.95      0.95    396002
weighted avg       0.96      0.96      0.95    396002
```

--Random Forest Classification Report--

8. RESULT OF CREDIT CARD FRAUD DETECTION

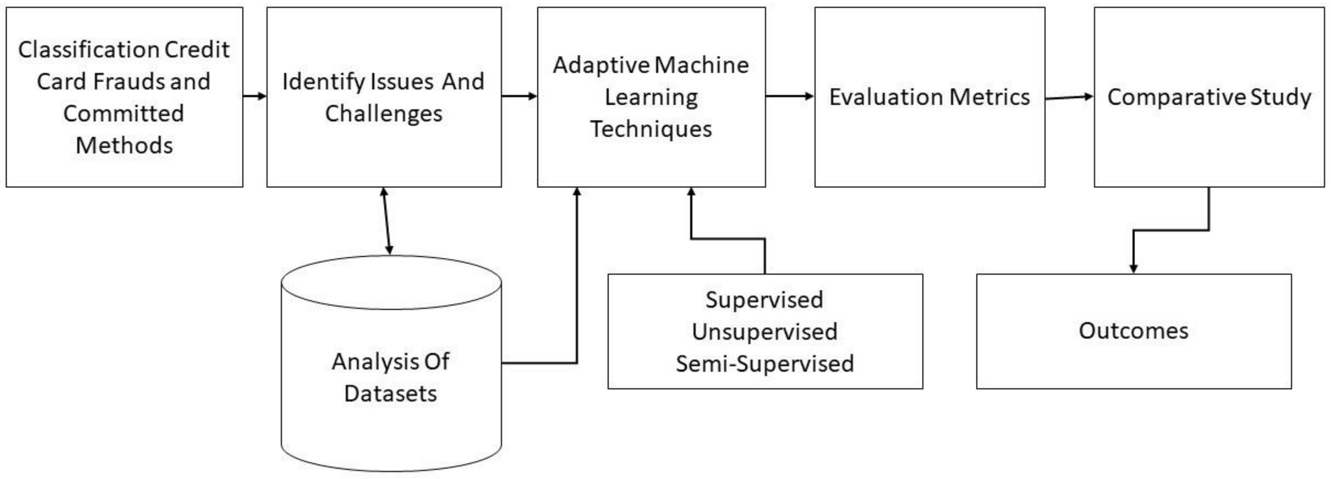
Implementing SMOTE on our imbalanced dataset helped us with the imbalance of our labels (more no fraud than fraud transactions). Nevertheless, I still have to state that sometimes the neural network on the oversampled dataset predicts less correct fraud transactions than our model using the undersample dataset. However, remember that the removal of outliers was implemented only on the random undersample dataset and not on the oversampled one. Also, in our undersample data our model is unable to detect for a large number of cases non fraud transactions correctly and instead, misclassifies those non fraud transactions as fraud cases. Imagine that people that were making regular purchases got their card blocked due to the reason that our model classified that transaction as a fraud transaction, this will be a huge disadvantage for the financial institution. The number of customer complaints and customer disatisfaction will increase.



```
Out[78]:
                   Technique     Score
0     Random UnderSampling  0.982708
1      Oversampling (SMOTE)  0.997244
```

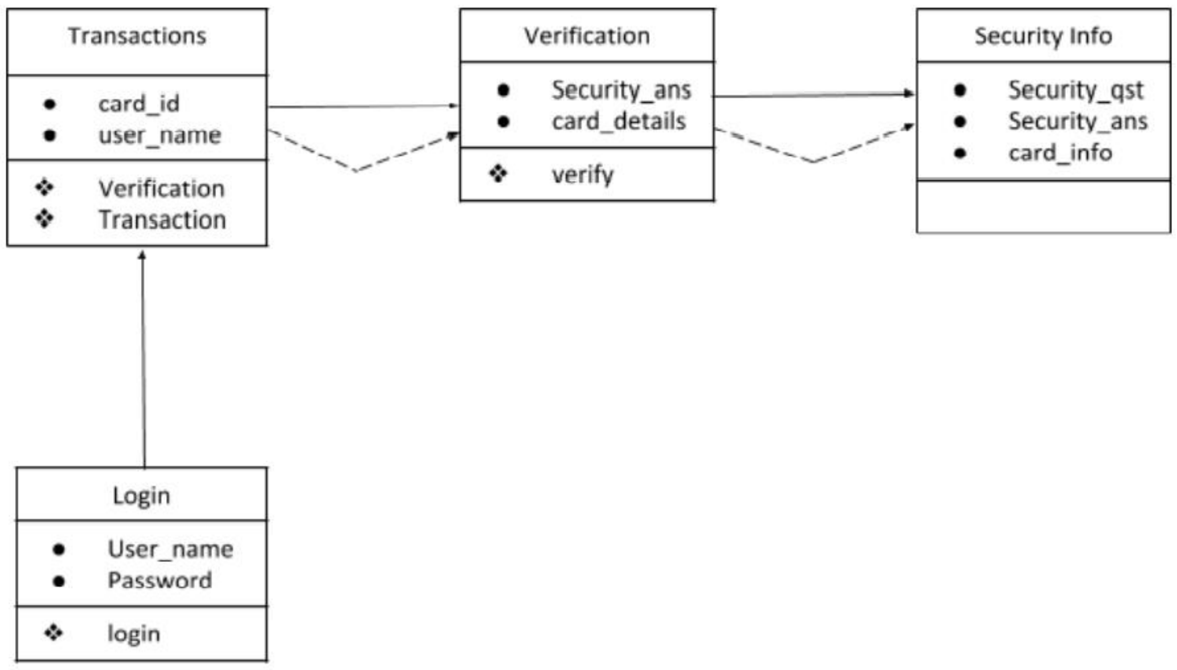
--Result--

9. DATAFLOW DIAGRAM OF CREDIT CARD FRAUD DETECTION


Classification Credit Card Frauds and Committed Methods
Identify Issues And Challenges
Adaptive Machine Learning Techniques
Evaluation Metrics
Comparative Study
Analysis Of Datasets
Supervised
Unsupervised
Semi-Supervised
Outcomes

--Dataflow Diagram--

10. ENTITY RELATIONSHIP (ER) DIAGRAM OF CREDIT CARD FRAUD DETECTION



--Entity Relationship Diagram--

11. CONCLUSION OF CREDIT CARD FRAUD DETECTION

Machine learning technique like Logistic regression, Random forest, K-Nearest Neighbours, SVC classifiers were used to detect the fraud in credit card system. Sensitivity, Specificity, accuracy and error rate are used to evaluate the performance for the proposed system. From the experiments, the result that has been concluded is that Logistic regression has an accuracy of 94.01% while SVC shows accuracy of 93.73%, KNN with 93.44% and Random forest shows accuracy of 94.58% but the best results are obtained by Random Forest with a precise accuracy of 94.58%.

12. FUTURE SCOPE OF CREDIT CARD FRAUD DETECTION

With increasing number of bank fraudulency and cyber-crime cases, need of a secure testing system is on rise. And this is a direct solution to this problem. It can be extended to a duplex verification of not only a customer(debit-ant) but also of the seller(credit-ant). It can be taken and used on a regular basis just like OTP. It can be used to even assess past transaction in database to find whether certain transactions were fraudulent or not and also would be able to produce evidence in such cases. Also optimization techniques on the proposed models and testing on new models can be done.

13. REFRENCES

1. Y. Sahin, S. Bulkan, and E. Duman, ‘‘A cost-sensitive decision tree approach for fraud detection,’’ Expert Syst. Appl., vol. 40, no. 15, pp. 5916–5923, 2013.

2. Sahil Dhankhad, Emad A. Mohammed and Behrouz Far, “Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study ”

3. Navanshu Khare and Saad Yunus Sait, “Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models “

4. Dal Pozzolo, Andrea, et al. "Calibrating Probability with Undersampling for Unbalanced Classification." Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, 2015

5. Lakshmi S V S S1,Selvani Deepthi Kavila2 “Machine Learning For Credit Card Fraud Detection System”

6. S. J. K. T. J. C. W. Siddhatha Bhattacharya, “Data Mining for credit card fraud: A comparative study,” Elsevire, vol. 50, no. 3, pp. 602- 613, 2011.