DR. B.R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY, JALANDHAR (144011), PUNJAB

MACHINE LEARNING PROJECT

CREDIT CARD FRAUD DETECTION

Submitted To:

Dr. Jagdeep Kaur

Submitted By:

Noor Arora
(20103103)

Parambir Singh Kohli
(20103106)

Prisha Joshi
(20103110)

ABSTARCT

Retaining high profitable customers is the number one business goal for many banks. Banking frauds however possess a significant threat to this goal. The focus of this project is to apply machine learning techniques that can accurately predict whether a transaction made by a customer using a credit card is a valid transaction or a fraud.

This project analyses customer level data that has been collected and analysed during a research collaboration of WordLine and the machine learning group.

DATASET

The data set includes credit card transactions made by European card holders over a period of 2 days.

- Total number of transactions = **2,84,807**
- Fraud transactions = **492**
- Positive class (frauds) = **0.172%**

Some important features of the data such include :-

- **Time :** Seconds elapsed between the first transaction in the dataset and the subsequent transaction.
- **Amount :** Transaction amount
- **Other features** :V1, V2 … V28
- **Class :** Target function ;
    - It takes value 1 in case of fraud
    - and 0 in case of valid

DATA PREPROCESSING

As the data is it contained in our dataset is raw, hence unsuitable for machine learning model. It might contain noise, null values etc.

Under the task of pre-processing the data is refined and put in a formatted way, making it suitable for a machine learning model.

I. Importing Libraries :

Libraries used in this project are:

- Numpy
- Pandas
- Matplotlib
- Seaborn
- Sklearn
- Xgboost

II. Importing the Dataset :

Dataset used in this project is stored in CSV format. We can load it by using the command :

pd.read_csv()

this function is contained in pandas Library.

More about CSV files : CSV stands for "Comma separated values". It is a file format that allows users to save data in a tabular form.

CODE:

```
df = pd.read_csv('creditcard.csv')
df.head()
```

df -> variable that stores our dataset

III. Dependent and independent variable

Dependent variable in this case is class that is the target function.

It takes value 1 in case of fraud and zero in case of valid transaction.

The rest of the variables V1------V28, time and amount are independent variables.

- ***df.shape()***

This function is contained in the nampy library. It returns the shape of the data set that is number of rows and columns

Here it returns (2,84,807, 31)

Where 2,84,807 is number of rows 31 is number of columns.

- ***df.info( )***

This function is contained in pandas library. It prints the information about the data set. The information contains:

➔ Number of columns
➔ Colum labels (if any)
➔ Column data types
➔ Memory usage
➔ Range index
➔ Non-null values

- ***df.describe( )***

This function is contained in the pandas library. It returns the description of the data in the data set.

The description returned by this function is as follows

➔ Count. (The number of not empty values)
➔ Mean. (Average value)
➔ Std. (Standard deviation)

➔ Min. (Minimum value from the column)
➔ 25%. (25 % percentile)
➔ 50%. (50% percentile)
➔ 75%. (75% percentile)
➔ Max. (Maximum value)

- ***Df['class'].value-counts( )***

Returns the number of zeros and ones in the class column

No. of zeros = 2,84,315

No.of ones = 492

IV. Checking for Correlations in the dataset :

Correlation is defined as a measure of a mutual relationship b/w two variables.

In simple words it means if two variables are related in any way. Correlations can act as important factor in determining features while building machine learning models. It help to see which features are linerally related. If features are strongly correlated, we can remove them to prevent duplicating information.

Range of correlation. -1to 1

- ***corr= df.corr()***

This function is contained in pandas library. It helps us to see correlations for our numerical columns.

Return value : A new data frame showing each correlation.

From the output we observe that the diagonal values are 1 , because the correlation of a variable with itsself is 1.

We have also used heat map to find the correlation

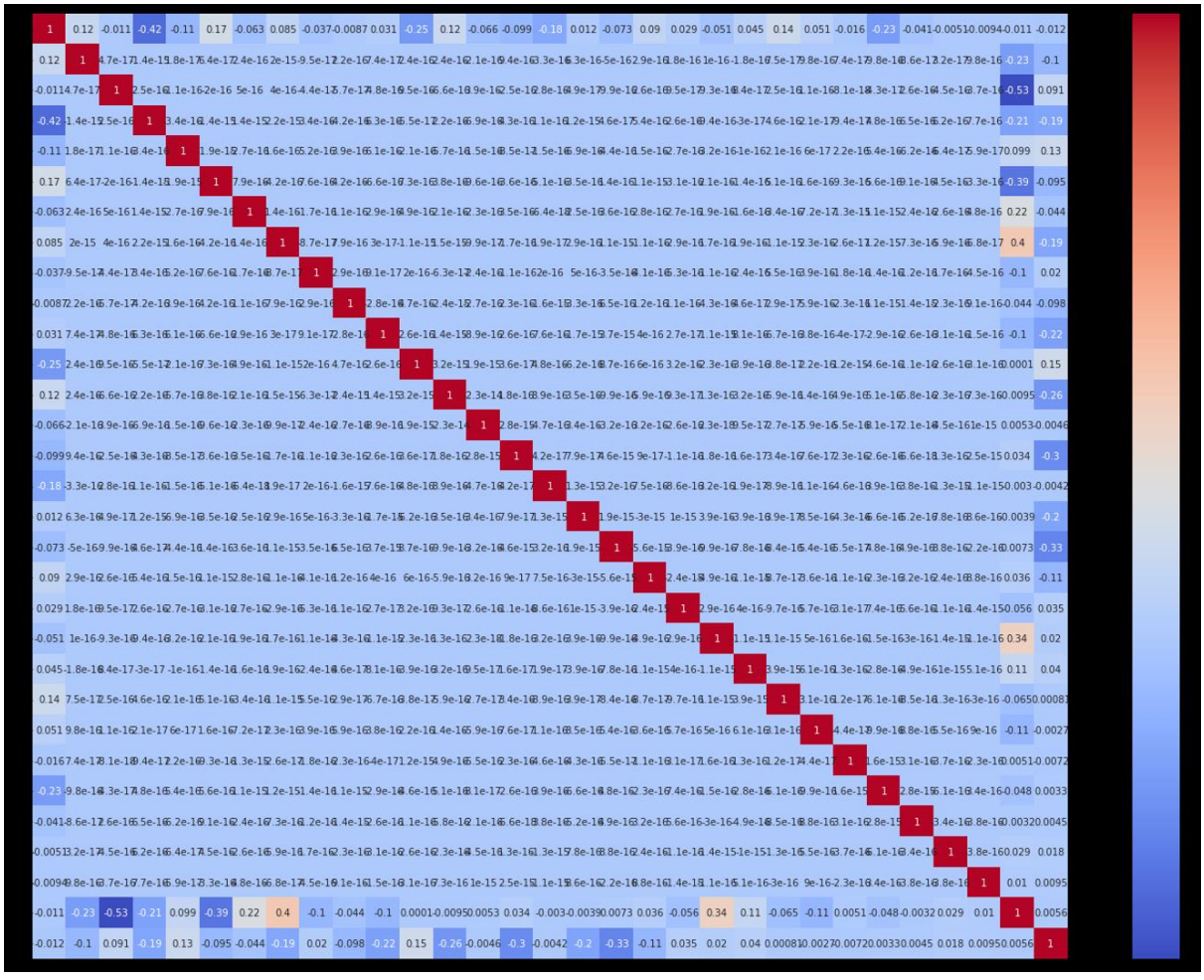
To use heatmap we need to import the seaborn library.

Heat maps are 2D representation of data. The colour of each cell of the heat map is proportional to the number of measurements that match the dimensional value.

Values closer to 1 in the heatmap indicates positive correlation from the output V23 and V4 are positively correlated as V23 increases, V4 also increases.

Values closer to -1 indicate negative correlation. Eg. amount and V2

The diagonals are darker as they indicate perfect correlation



NOTE: *As the data in our dataset has numeric values only, therefore, no need to convert the data. If there had been some alphanumeric values in the data, they need to be converted into numeric values using the function* ***get_dummies()*** *contained in the Pandas library. This is because ML models only work on numeric data.*

Also there are no NaN values in our data. Nan values can be handled by:

➔ *Deleting that particular row.*
➔ *Fill the missing value with the mean of the row/column. We can use* ***attrib.fillna(attrib.mean())*** *function for this.*

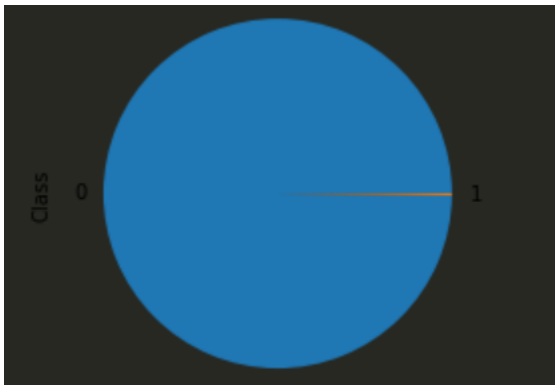
V. Data Visualization:

Matplotlib library is used for data visualization.

1. Piechart:

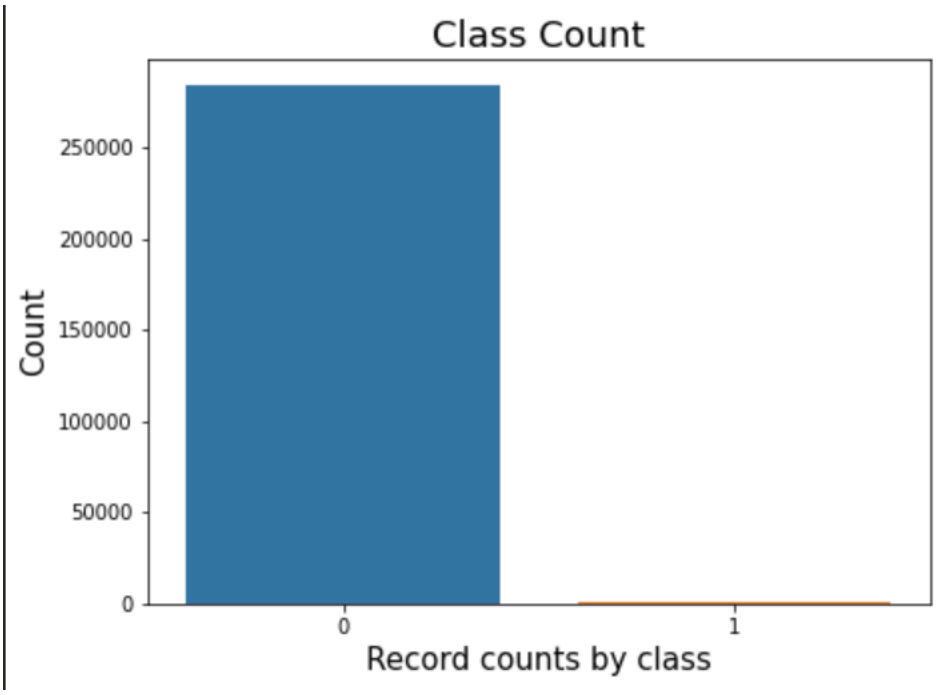
We calculate the percentage of 0 and 1 values in the target function and use them to plot a pie chart.

The piechart will be:



2. Bar plot:

A bar plot is plotted for fraud vs normal transactions.



VI. Splitting the data into train and test data:

Overfitting is a common issue when training a model. It occurs when model performs well on the data that was used to train the model but fails to work on any other data or unseen data points.

Underfitting occurs When model has poor performance even on the data that was used to train it.

One method to overcome these issues is creating different different data samples for training and testing the model.

The data set is divided into 2 subsets.

The 1st subset is used to fit the model and is called **training data set.**

The 2nd subset, Instead of training the model, The input element of the data set is provided to the model, then predictions are made and

compared to the expected values. This is subset is referred to as **test data set.**

In this project, **'y'** variable contain the target function 'class'

And **'X'** contains all other independent variables (Time, amount, V1..... V28).

Train's test split

It signifies the size of train and test sets. It is commonly expressed as percentage between 0 to 1. Commonly the data set is divided into 2/3 for training data set and 1/3 for test data set.

Although there is no optional split.

Other common splites can be

- Train: 80% test: 20%
- Train 67% test: 33%
- Train 50% test: 50%

We will use 80% and 20% split for our model.

Library used for this purpose is **sklearn**.

X_train, X_text, y_train, y_test = train_test_split(X, y, random_state=100, test_size=0.20)

➔ X_train (training part of the first sequence (X))
➔ X_text (test part of the first sequence (X))
➔ Y_train (training part of the second sequence (y))
➔ Y_test (test part of the second sequence (y))

train_test_split() function performs the split and returns the above mentioned 4 sequences in order.

Random state:

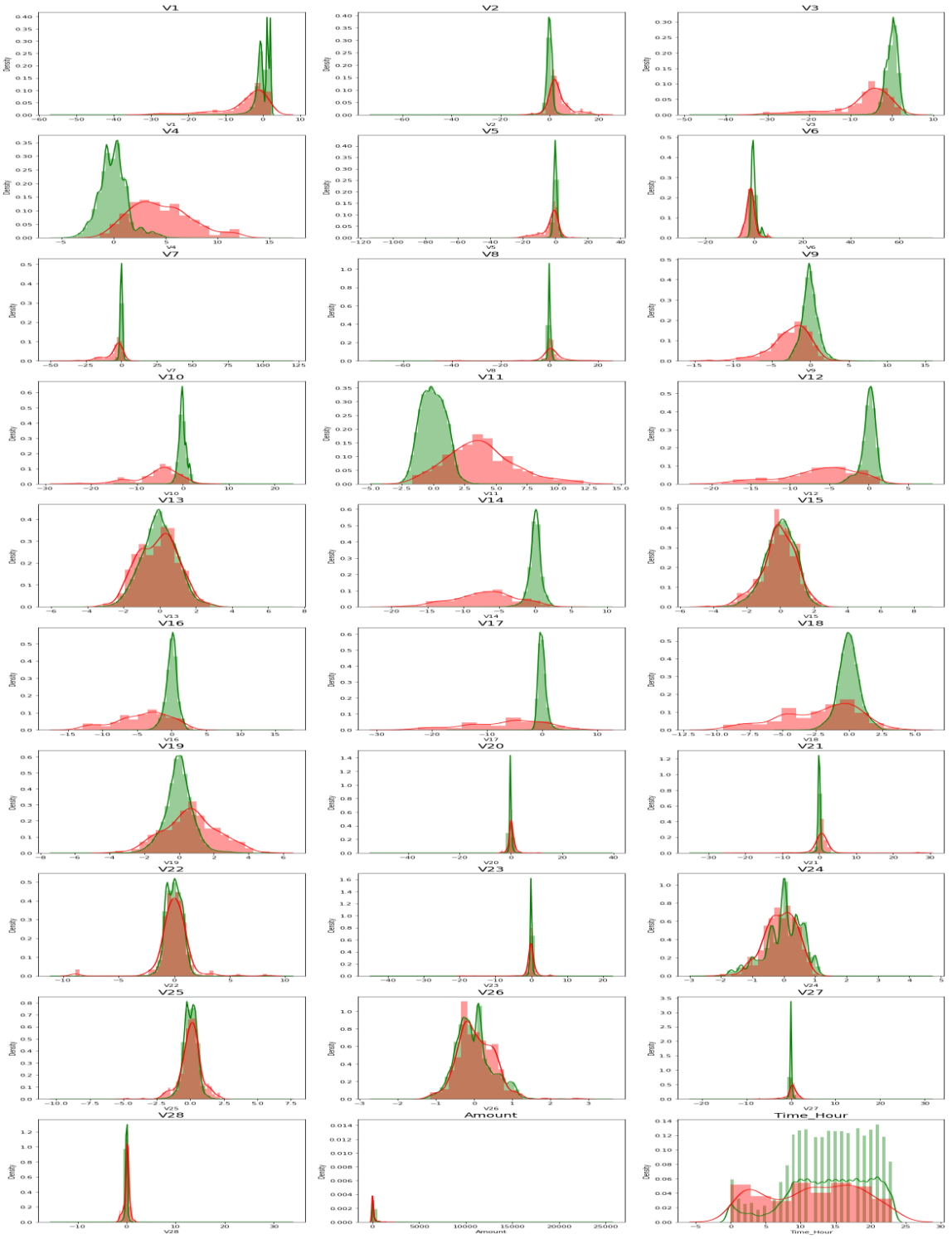
It can have any non-negative integer value. It is used for initializing the internal random number generator.

Histograms are plotted for the attributes of the dataset to see the skewness.

Here,

Green colour depicts **valid transactions** and

red colour depicts **fraud transactions**.

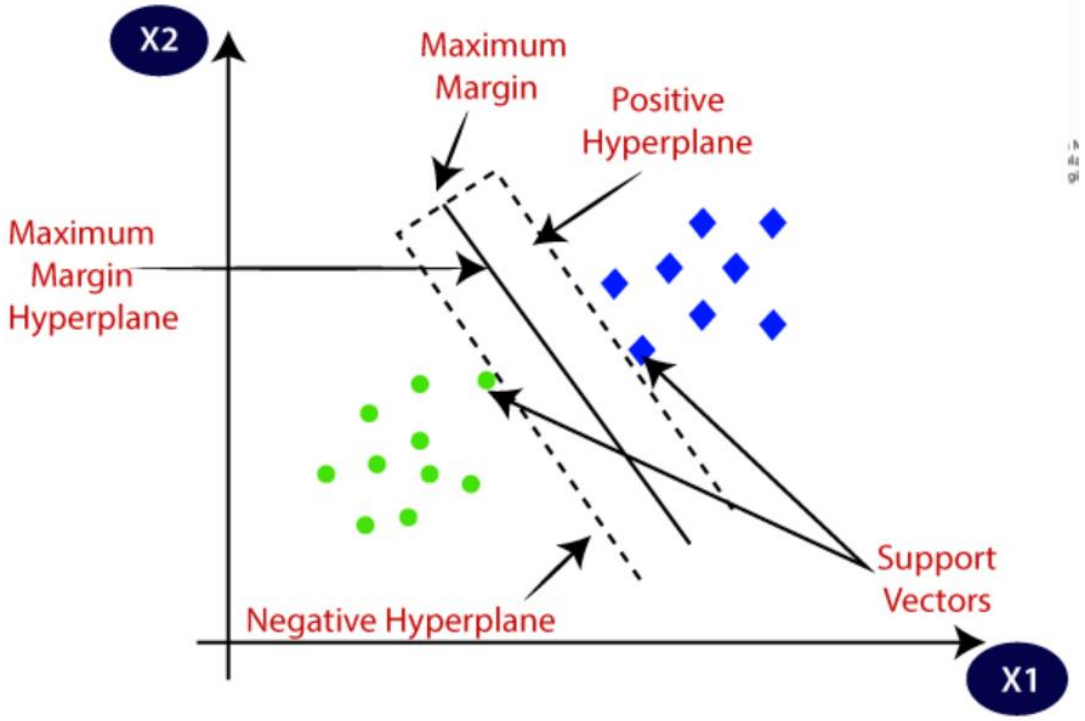


SVM

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

Advantages of SVM:

- Effective in high dimensional cases
- Its memory efficient as it uses a subset of training points in the decision function called support vectors
- Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

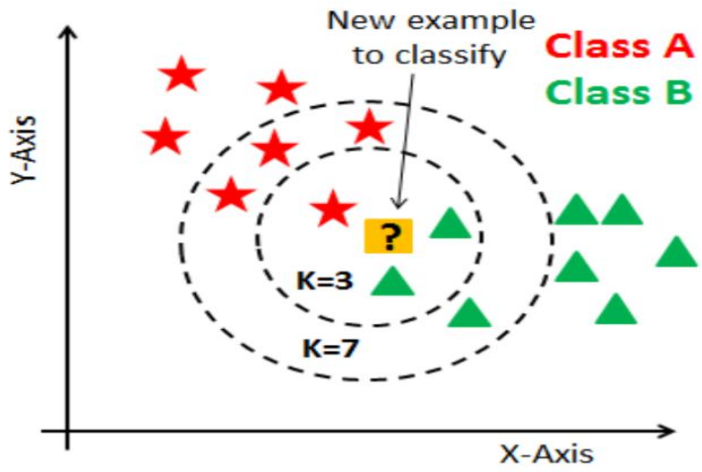


KNN

K-nearest neighbors algorithm (k-NN) is a non-parametric supervised learning method used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.



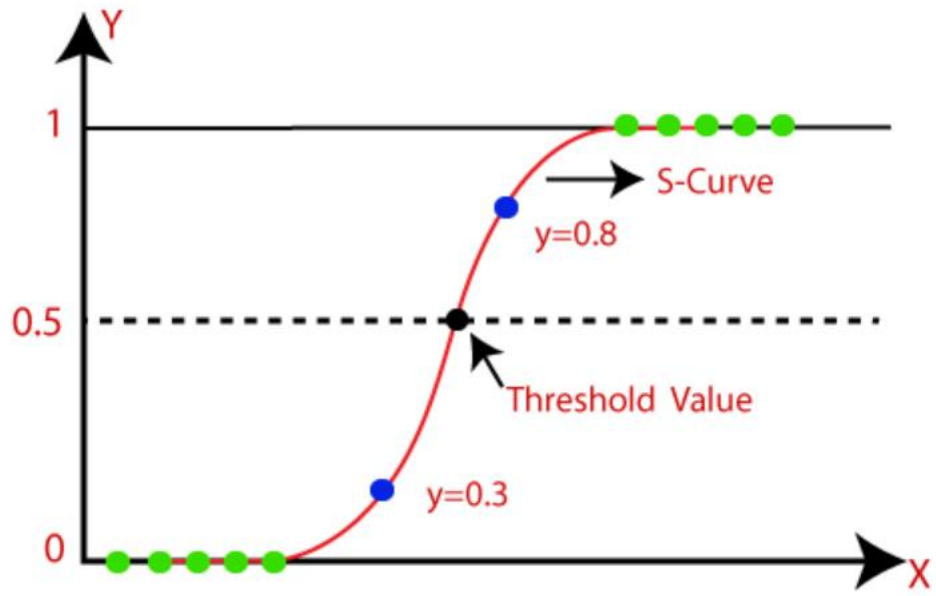
Logistic Regression

Regularization: Regularization is a technique used to prevent overfitting problem. It adds a regularization term to the equation-1(i.e. optimisation problem) in order to prevent overfitting of the model. The regression model which uses L1 regularization is called Lasso Regression and model which uses L2 is known as Ridge Regression.

- **Ridge Regression (L2 norm).**
  L2-norm loss function is also known as least squares error (LSE).
  $w^* = \text{minimization of } \sum_{i=1}^{n}[\log(1+\exp(-z_i))] + \lambda^* \sum (w_j)^2$
  $\sum (w_j)^2$ is a regularization term and $\sum [\log(1+\exp(-z_i))]$ is the Loss term. λ is a hyper parameter.
  We added the regularization term(i.e. squared magnitude) to the loss term to make sure that the model does not undergo overfit problem. Here we will minimize both the Loss term and the regularization term. If hyper parameter(λ) is 0 then there is no regularization term then it will overfit and if hyper parameter(λ) is very large then it will add too much weight which leads to underfit. We can find the best hyper parameter by using cross validation.

- **Lasso Regression (L1 norm)**
  L1-norm loss function is also known as least absolute deviations (LAD), least absolute errors (LAE). In L1 regularization we use L1 norm instead of L2 norm
  $w^* = \text{argmin} \sum[\log(1+\exp(-z_i))] + \boldsymbol{\lambda^* ||w||_1}$
  Here the L1 norm term will also avoid the model to undergo

overfit problem. The advantage of using L1 regularization is **Sparsity.**



XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data.

What Algorithm Does XGBoost Use?

The XGBoost library implements the gradient boosting decision tree algorithm.

Decision Tree Classification Algorithm

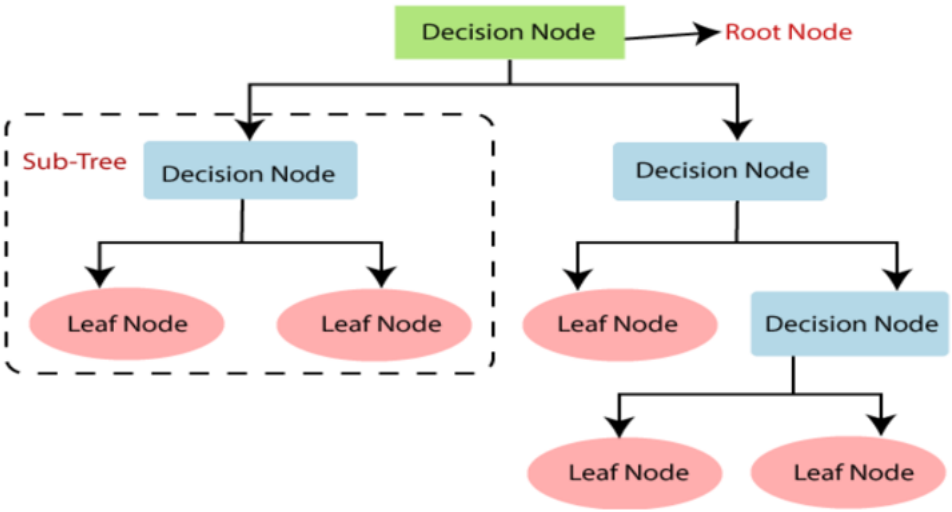
- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

- The decisions or the test are performed on the basis of features of the given dataset.

- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

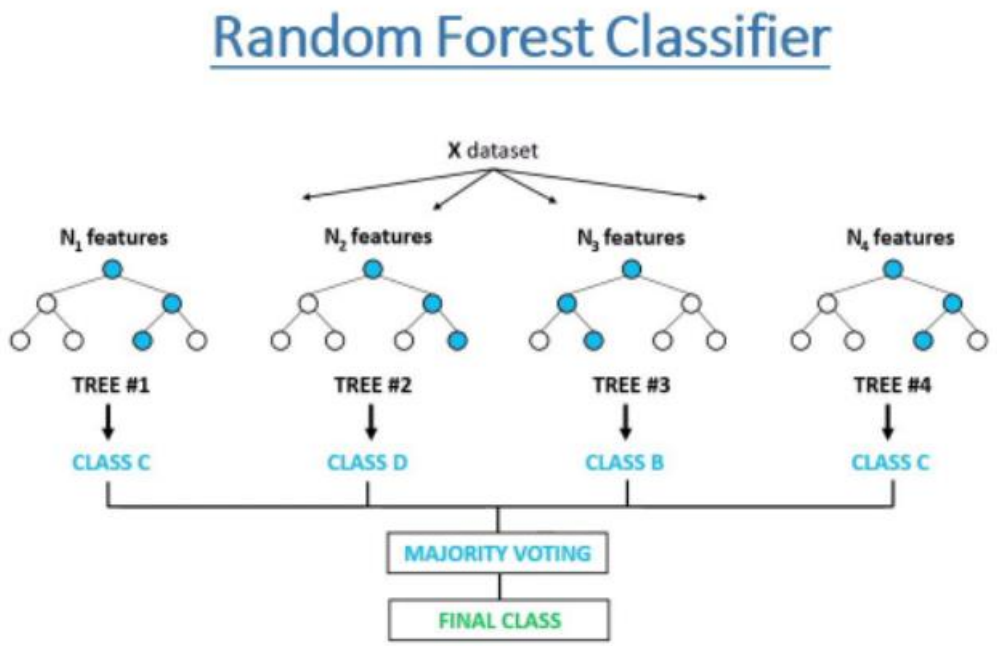
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.



Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree. Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned

Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.



TO INCREASE THE ACCURACY:

Repeated K FOLD

Repeated k-fold cross-validation provides a way to improve the estimated performance of a machine learning model. This involves simply repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs.

Stratified K FOLD

Stratified K-Folds cross-validator. Provides train/test indices to split data in train/test sets. This cross-validation object is a variation of KFold that returns stratified folds. The folds are made by preserving the percentage of samples for each class. It maintains the same class ratio throughout the K folds as the ratio in the original dataset.

Checking the Accuracy of Models:

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Conclusion:

Of all the models, the best accuracy was given by Logisitic Regression with L2 Regularisation for StratifiedKFold cross validation is 99.87%.