PROJECT REPORT ON SATAYNDER CUSTOMER TRANSACTION

Introduction

Problem Statement:

At Santander, mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals. Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Data Set :
1) test.csv
2) train.csv

Number of attributes:
We are provided with an anonymized dataset containing numeric feature variables, the binary target column, and a string ID_code column. The task is to predict the value of target column in the test set.

Attributes:

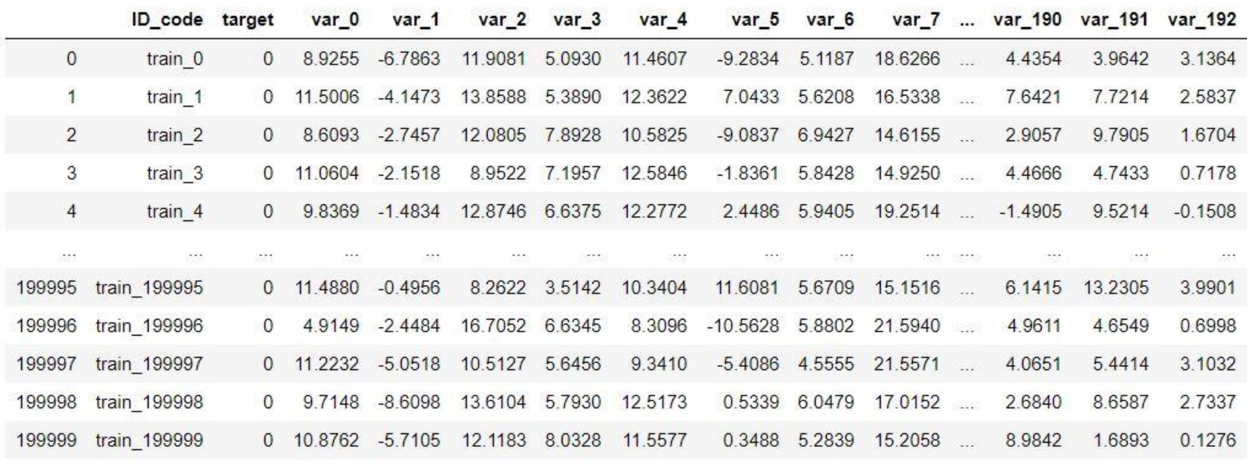
Test Data : ID_code , Var 0 , Var1 , Var2……….. Var 199
Train Data: ID_code ,target , Var 0 , Var1 , Var2……….. Var 199

Methodology

Pre-Processing :

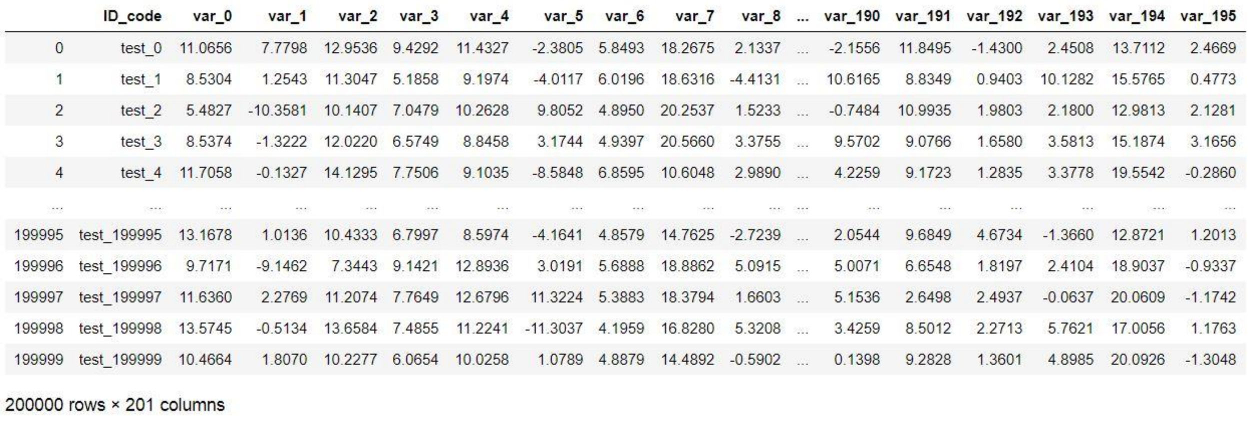
Data pre-processing is the first stage of any type of project. In this stage we get the feel of the data. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.

Train Dataset :



	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
199995	train_199995	0	11.4880	-0.4956	8.2622	3.5142	10.3404	11.6081	5.6709	15.1516	...	6.1415	13.2305	3.9901
199996	train_199996	0	4.9149	-2.4484	16.7052	6.6345	8.3096	-10.5628	5.8802	21.5940	...	4.9611	4.6549	0.6998
199997	train_199997	0	11.2232	-5.0518	10.5127	5.6456	9.3410	-5.4086	4.5555	21.5571	...	4.0651	5.4414	3.1032
199998	train_199998	0	9.7148	-8.6098	13.6104	5.7930	12.5173	0.5339	6.0479	17.0152	...	2.6840	8.6587	2.7337
199999	train_199999	0	10.8762	-5.7105	12.1183	8.0328	11.5577	0.3488	5.2839	15.2058	...	8.9842	1.6893	0.1276

Test Dataset :



	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_194	var_195
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	...	-2.1556	11.8495	-1.4300	2.4508	13.7112	2.4669
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	...	10.6165	8.8349	0.9403	10.1282	15.5765	0.4773
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	...	-0.7484	10.9935	1.9803	2.1800	12.9813	2.1281
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	...	9.5702	9.0766	1.6580	3.5813	15.1874	3.1656
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	...	4.2259	9.1723	1.2835	3.3778	19.5542	-0.2860
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
199995	test_199995	13.1678	1.0136	10.4333	6.7997	8.5974	-4.1641	4.8579	14.7625	-2.7239	...	2.0544	9.6849	4.6734	-1.3660	12.8721	1.2013
199996	test_199996	9.7171	-9.1462	7.3443	9.1421	12.8936	3.0191	5.6888	18.8862	5.0915	...	5.0071	6.6548	1.8197	2.4104	18.9037	-0.9337
199997	test_199997	11.6360	2.2769	11.2074	7.7649	12.6796	11.3224	5.3883	18.3794	1.6603	...	5.1536	2.6498	2.4937	-0.0637	20.0609	-1.1742
199998	test_199998	13.5745	-0.5134	13.6584	7.4855	11.2241	-11.3037	4.1959	16.8280	5.3208	...	3.4259	8.5012	2.2713	5.7621	17.0056	1.1763
199999	test_199999	10.4664	1.8070	10.2277	6.0654	10.0258	1.0789	4.8879	14.4892	-0.5902	...	0.1398	9.2828	1.3601	4.8985	20.0926	-1.3048

200000 rows × 201 columns

Missing Value Analysis:

The Missing Value Analysis procedure performs three primary functions:

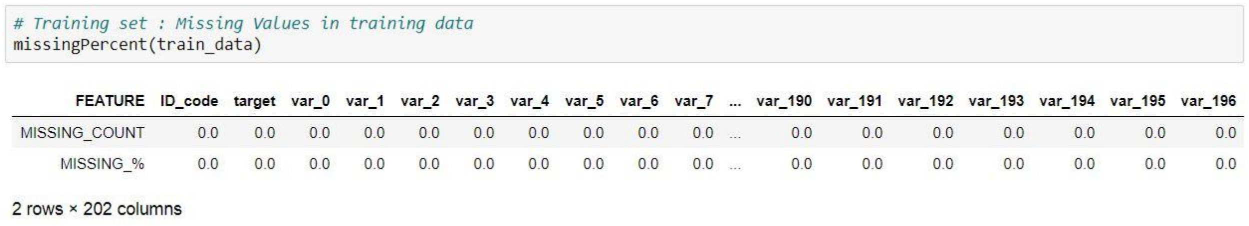
- Describes the pattern of missing data. Where are the missing values located? How extensive are they? Do pairs of variables tend to have values missing in multiple cases? Are data values extreme? Are values missing randomly?

- Estimates means, standard deviations, covariances, and correlations for different missing value methods: listwise, pairwise, regression, or EM (expectation-maximization). The pairwise method also displays counts of pairwise complete cases.

- Fills in (imputes) missing values with estimated values using regression or EM methods; however, multiple imputation is generally considered to provide more accurate results.

When we checked for the missing values in train and test dataset . Following are the observations :

For Training Data :

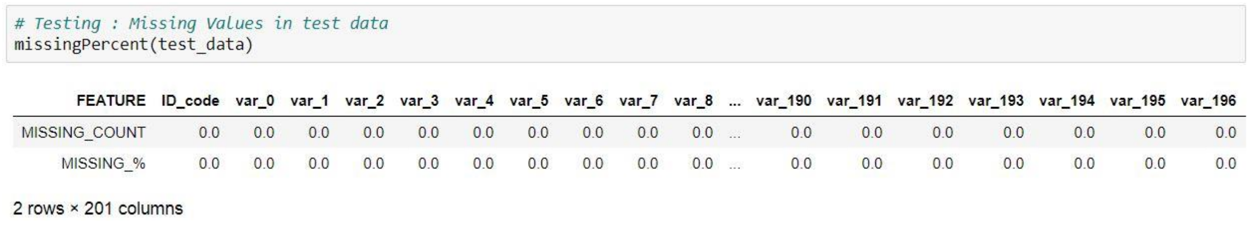


```
# Training set : Missing Values in training data
missingPercent(train_data)
```

FEATURE	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
MISSING_COUNT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MISSING_%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2 rows × 202 columns

For Testing Data :



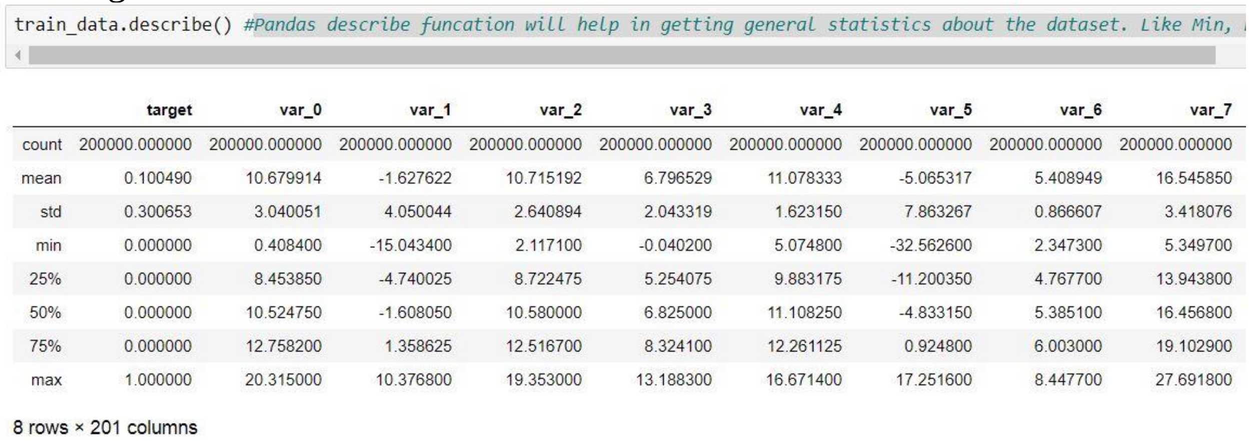
```
# Testing : Missing Values in test data
missingPercent(test_data)
```

FEATURE	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
MISSING_COUNT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MISSING_%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2 rows × 201 columns

Pandas describe function will help in getting general statistics about the dataset. Like Min, max STD and count etc.

Training Data :

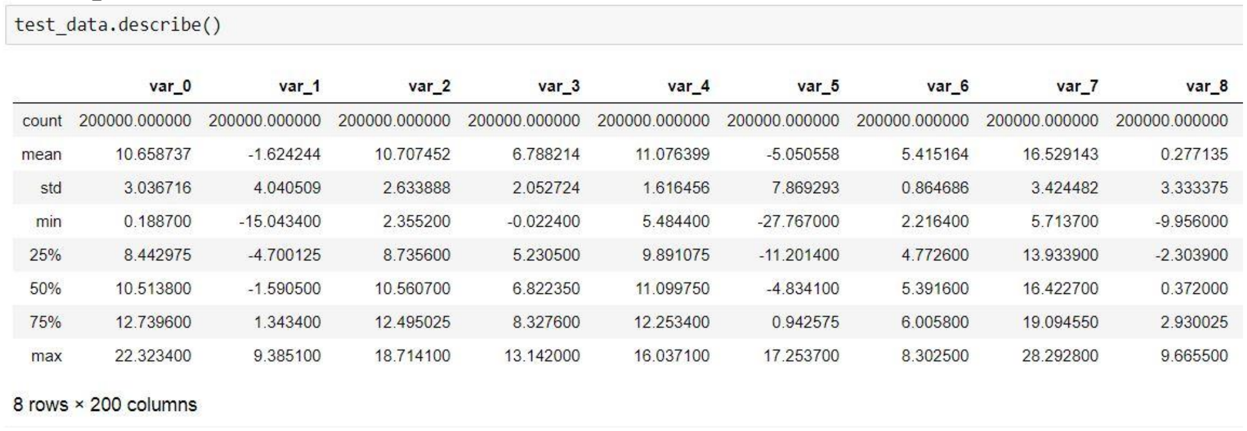


```
train_data.describe() #Pandas describe funcation will help in getting general statistics about the dataset. Like Min,
```

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949	16.545850
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607	3.418076
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300	5.349700
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700	13.943800
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100	16.456800
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000	19.102900
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700	27.691800

8 rows × 201 columns

Testing Data :



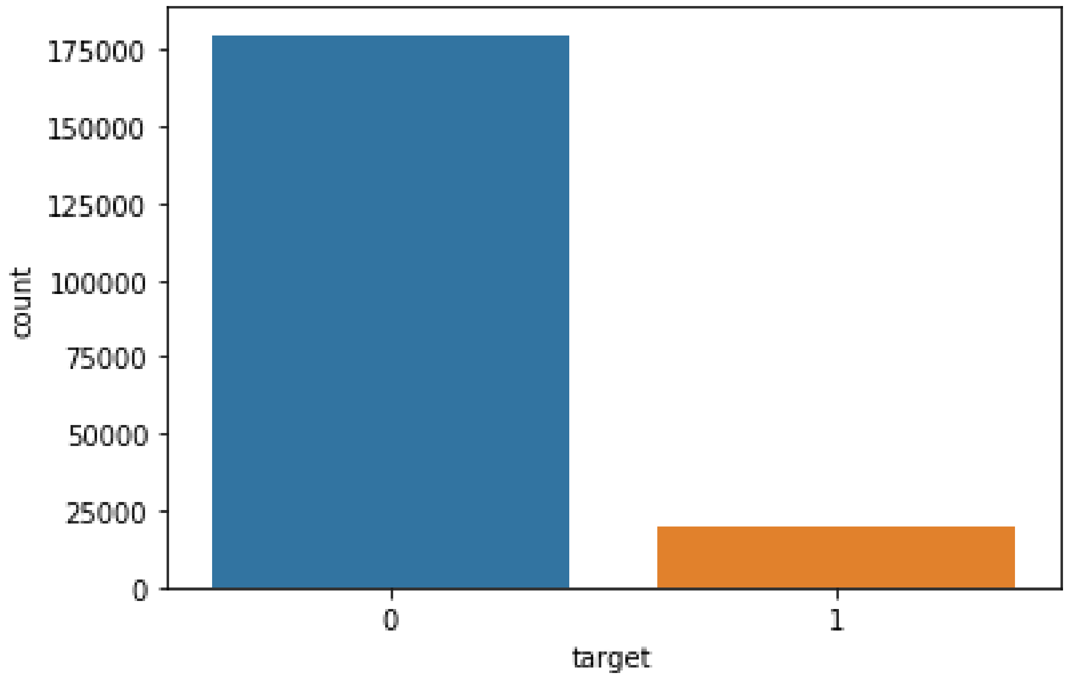
```
test_data.describe()
```

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000
mean	10.658737	-1.624244	10.707452	6.788214	11.076399	-5.050558	5.415164	16.529143	0.277135
std	3.036716	4.040509	2.633888	2.052724	1.616456	7.869293	0.864686	3.424482	3.333375
min	0.188700	-15.043400	2.355200	-0.022400	5.484400	-27.767000	2.216400	5.713700	-9.956000
25%	8.442975	-4.700125	8.735600	5.230500	9.891075	-11.201400	4.772600	13.933900	-2.303900
50%	10.513800	-1.590500	10.560700	6.822350	11.099750	-4.834100	5.391600	16.422700	0.372000
75%	12.739600	1.343400	12.495025	8.327600	12.253400	0.942575	6.005800	19.094550	2.930025
max	22.323400	9.385100	18.714100	13.142000	16.037100	17.253700	8.302500	28.292800	9.665500

8 rows × 200 columns

Data Visualization

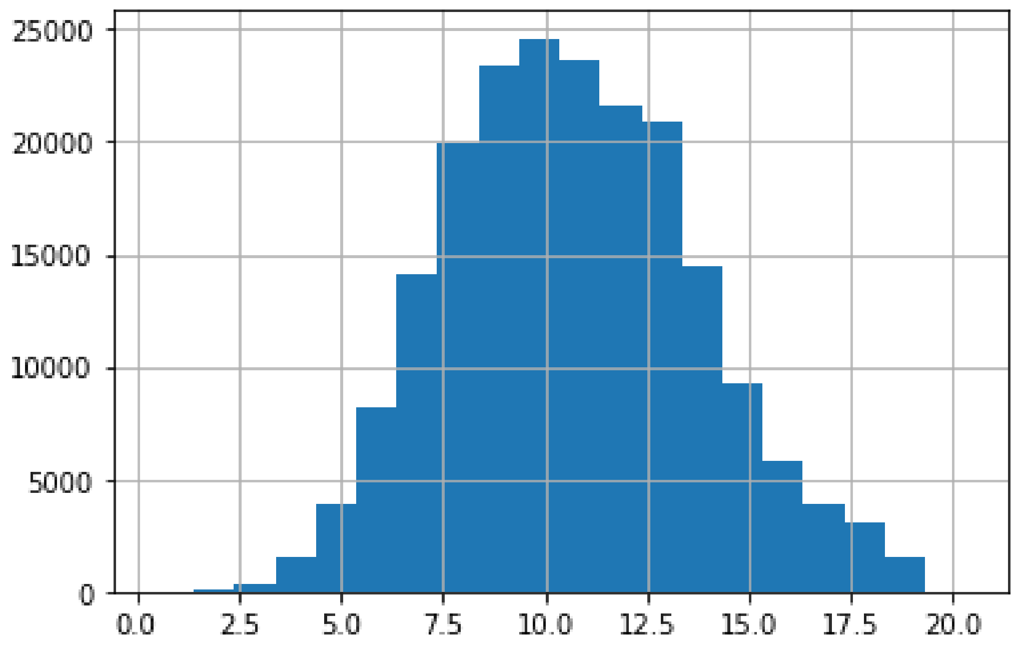
Visualization will help us to understand distribution. It's most important part of EDA. In the below given bar graph . we have shown the distribution of target variable of the dataset.



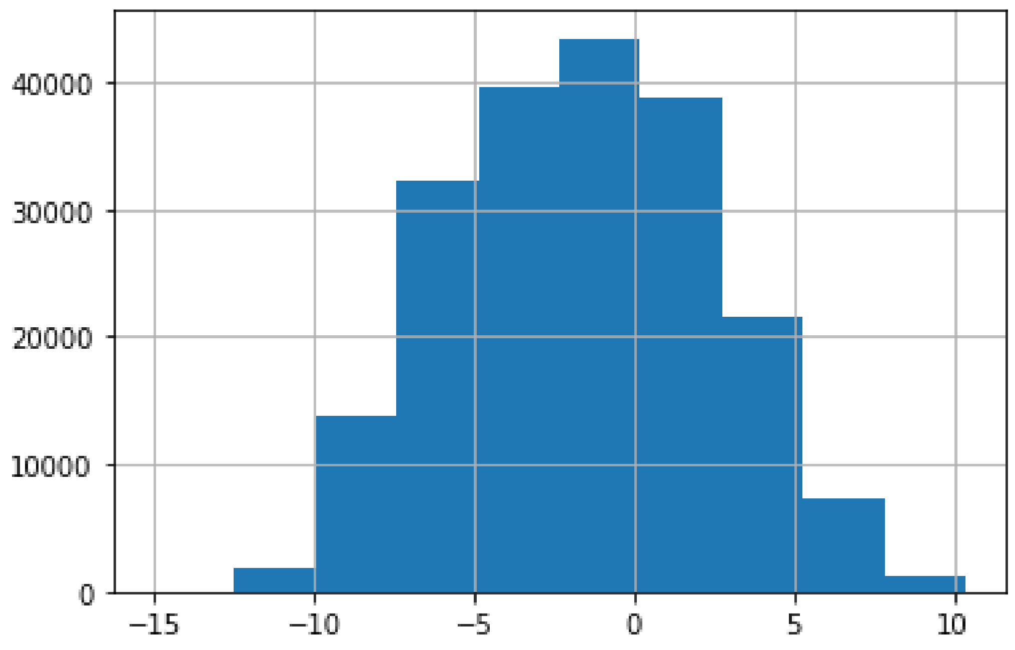
This is clearly showing that we are working or we have highly imbalanced dataset.

Histogram : A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable

Distribution of var0 values:



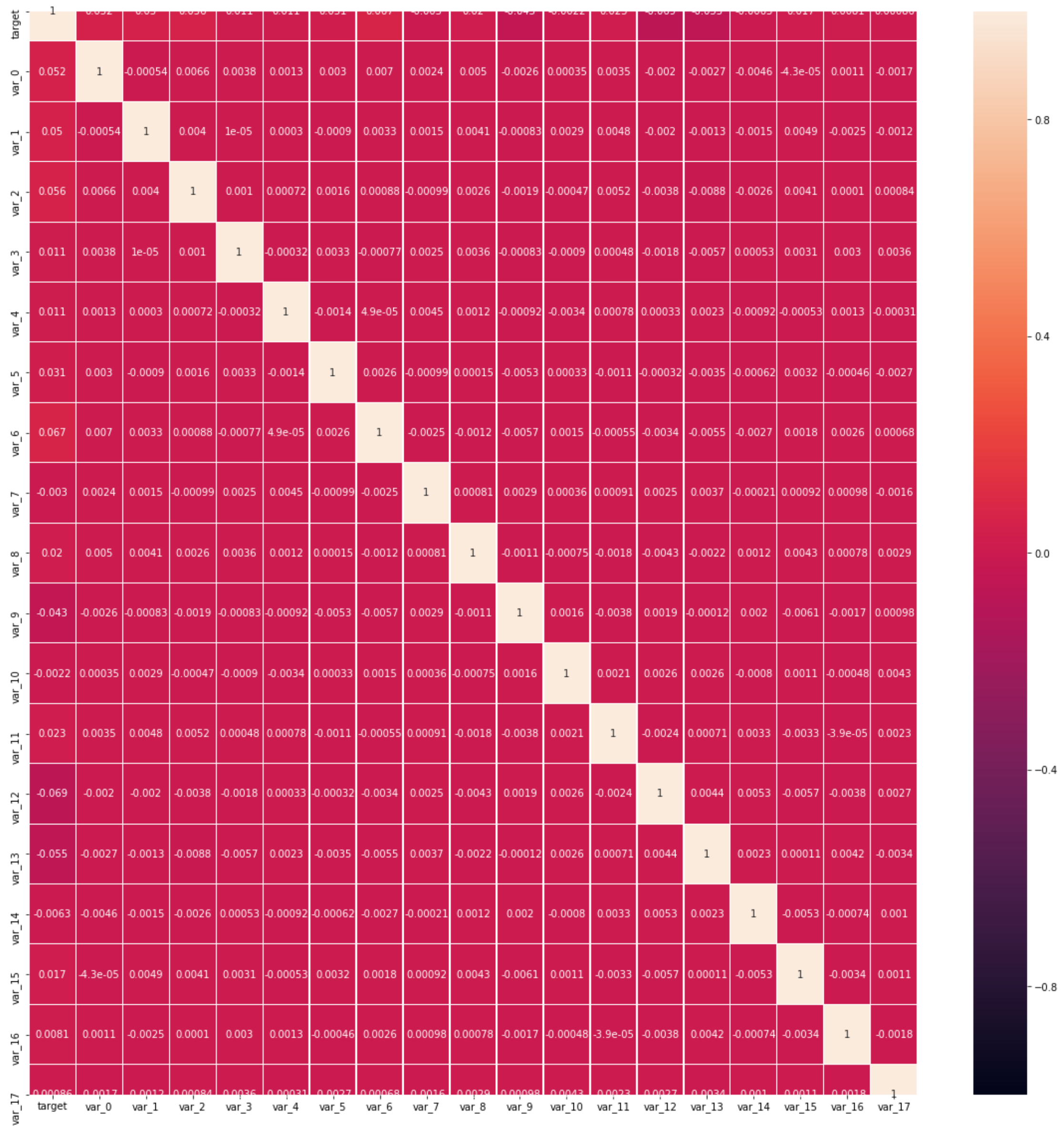
Distribution of var1 values:



Data Relationships

Visualization of data relationship, will help us to understand the degree of correlation between features and the dependencies. With the help of scatterplot and heatmap we can show the relationship of features.

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

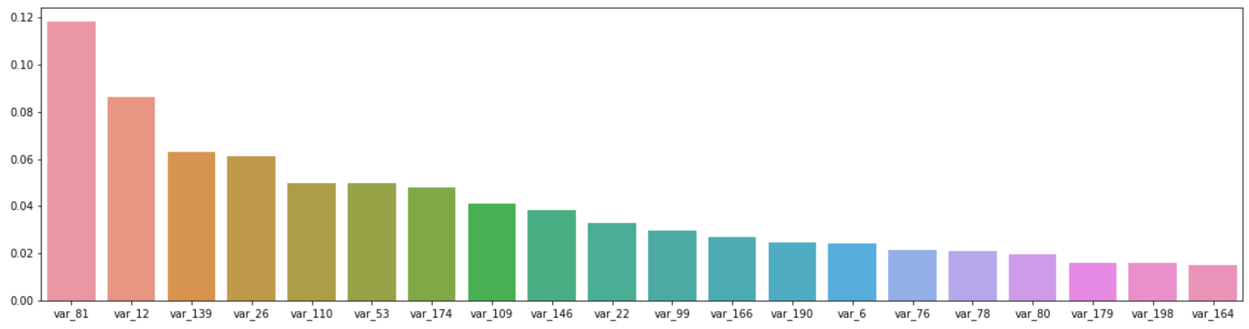


Heat Map Correlation Matrix of training data

<u>Feature Selection</u>

Feature selection is the process of selection the features, which are more relevant for our model and not carrying unnecessary noise to final model. Also it can help in giving idea about the new feature (Derived Features). I will use Random Forest for it, it will help us to detect the features importance.

Random Forests is used for feature selection for this dataset. The reason is because the tree-based strategies used by random forests naturally ranks by how well they improve the purity of the node. This mean decrease in impurity over all trees (called gini impurity). Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features.



In above given bar plot we have shown the top 20 features that we derived through Random Forest . Some of the features of the Train Data are var_81,var_12,var_139,var_53,var_174 and so on.

Model Selection

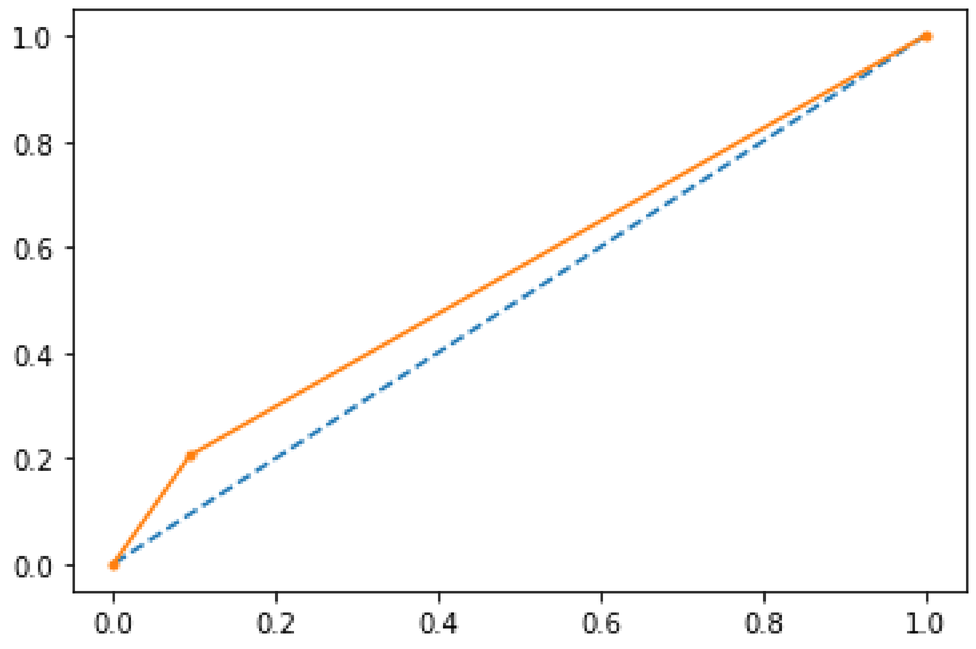
Our problem statement wants us to predict the target. This is a Classification problem. So, we are going to build classification models on training data and predict it on test data. In this project I have built models using 5 Classification Algorithms:

Decision Tree:

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

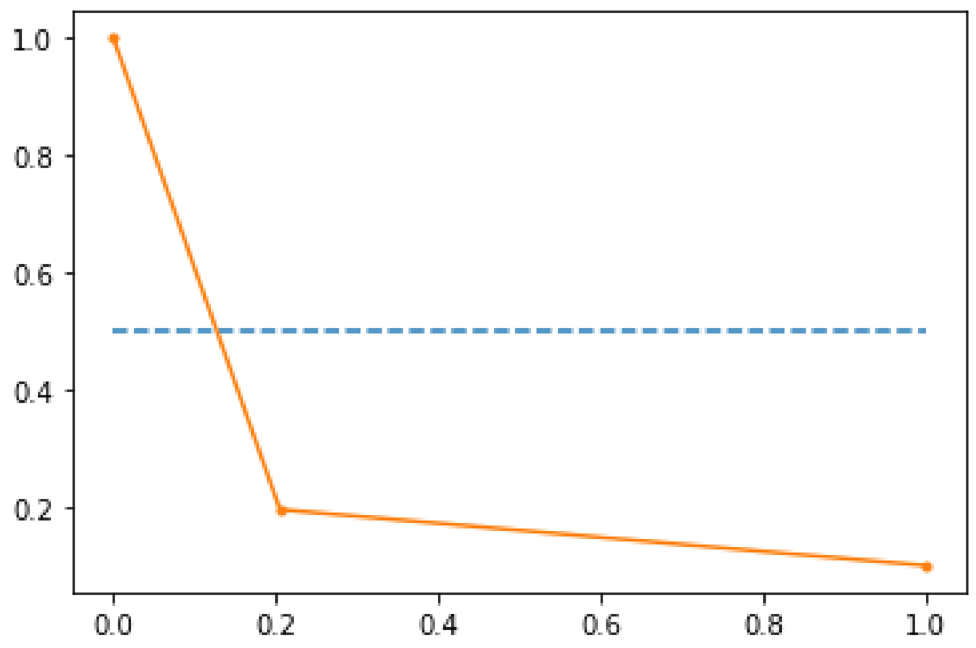
Classification Report :

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     44965
           1       0.20      0.21      0.20      5035

    accuracy                           0.84     50000
   macro avg       0.55      0.56      0.55     50000
weighted avg       0.84      0.84      0.84     50000
```



AUC Score : 0.556

Precision-Recall curve



Precision Recall Curve : 0.241

***************** Results of Decision Tree Model ***********************

Accuracy : 83.52%

False Negative Rate : 79.46%

Precision Recall AUC : 0.241

Random Forest

Random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

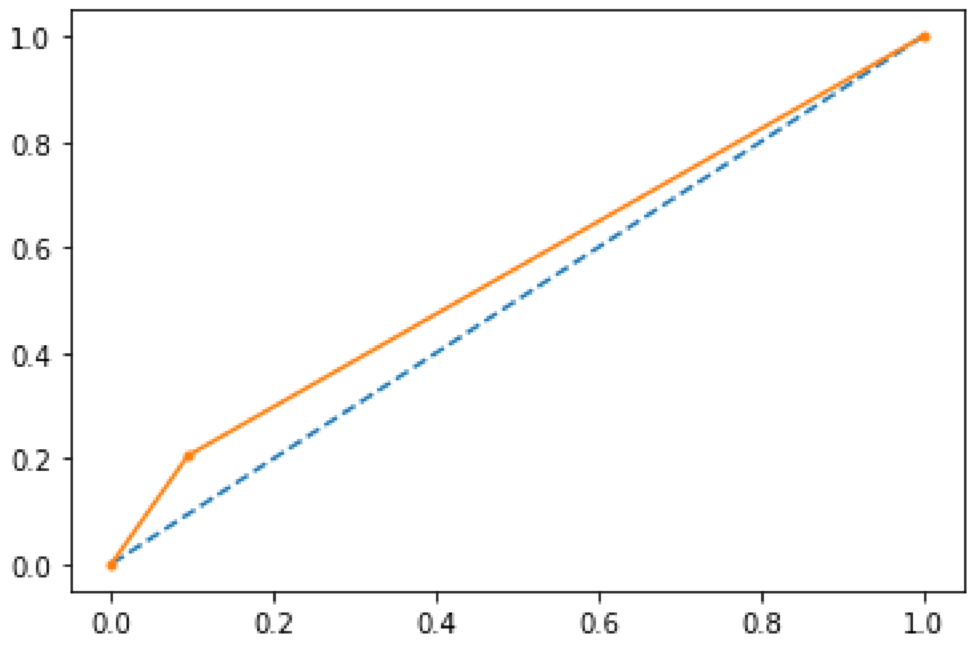
In the first model we have taken n_estimators = 100. For that we have following results :

```
Accuracy Random Forest Model 1 : 89.932000
```

Classification Report :

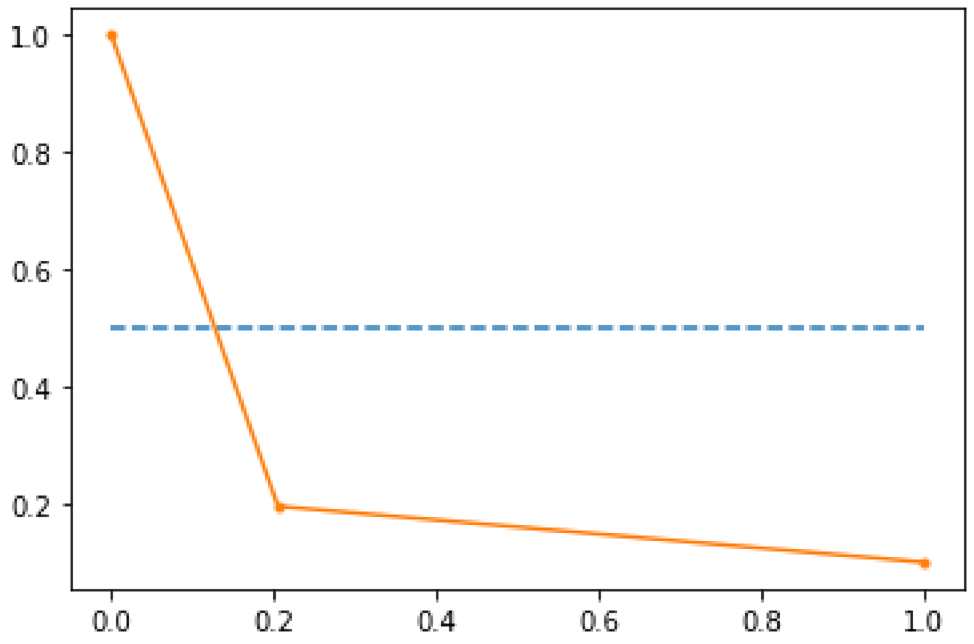
```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95     44965
           1       1.00      0.00      0.00      5035

    accuracy                           0.90     50000
   macro avg       0.95      0.50      0.47     50000
weighted avg       0.91      0.90      0.85     50000
```

ROC Curve :



```
AUC: 0.556
```

Precision Recall Curve :



```
AUC: 0.241
```

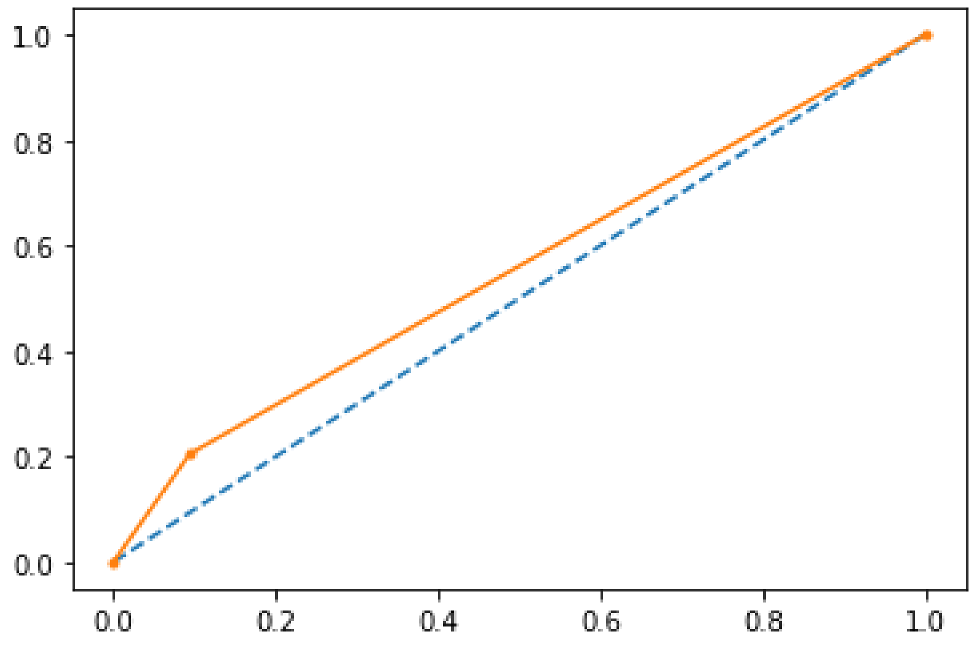
In the second model we have taken n_estimators = 500. For that we have following results :

```
Accuracy Random Forest Model 2 : 89.930000
```

Classification Report :

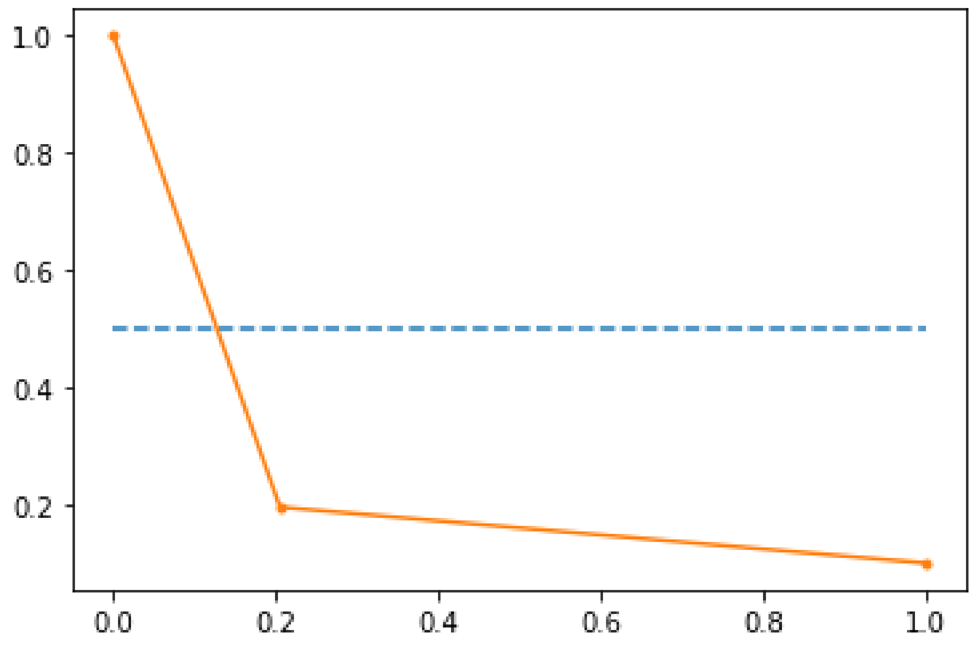
```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95     44965
           1       0.00      0.00      0.00      5035

    accuracy                           0.90     50000
   macro avg       0.45      0.50      0.47     50000
weighted avg       0.81      0.90      0.85     50000
```

ROC Curve :



```
AUC: 0.556
```

Precision Curve :



```
AUC: 0.241
```

So for the both the models Precision Score and recall is same no matter if we increased the number of trees or not

Logistic Regression Model

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

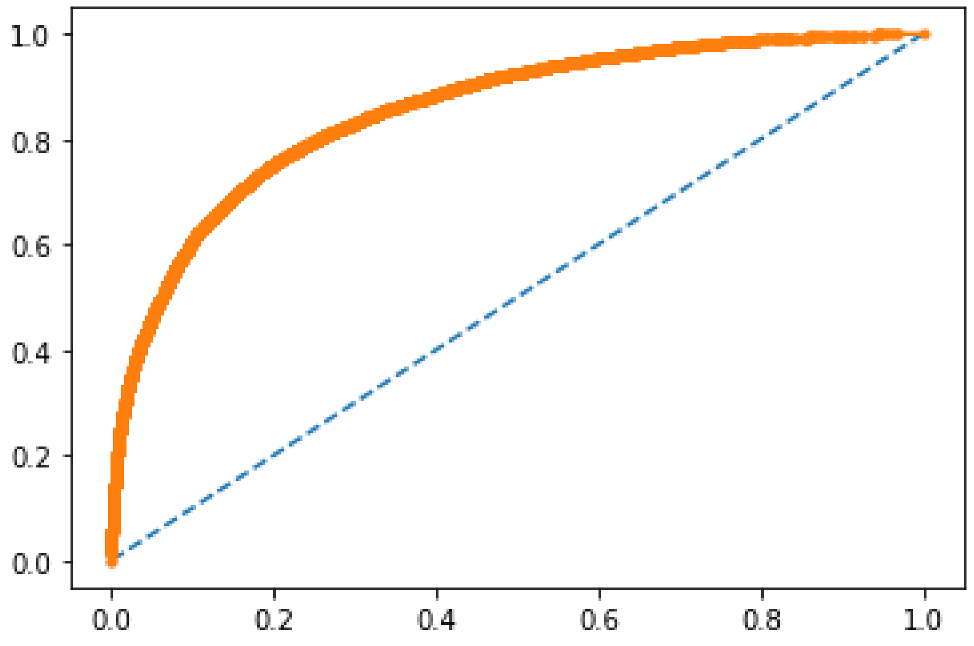
Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

```
Accuracy: 0.915
```

Classification Report :

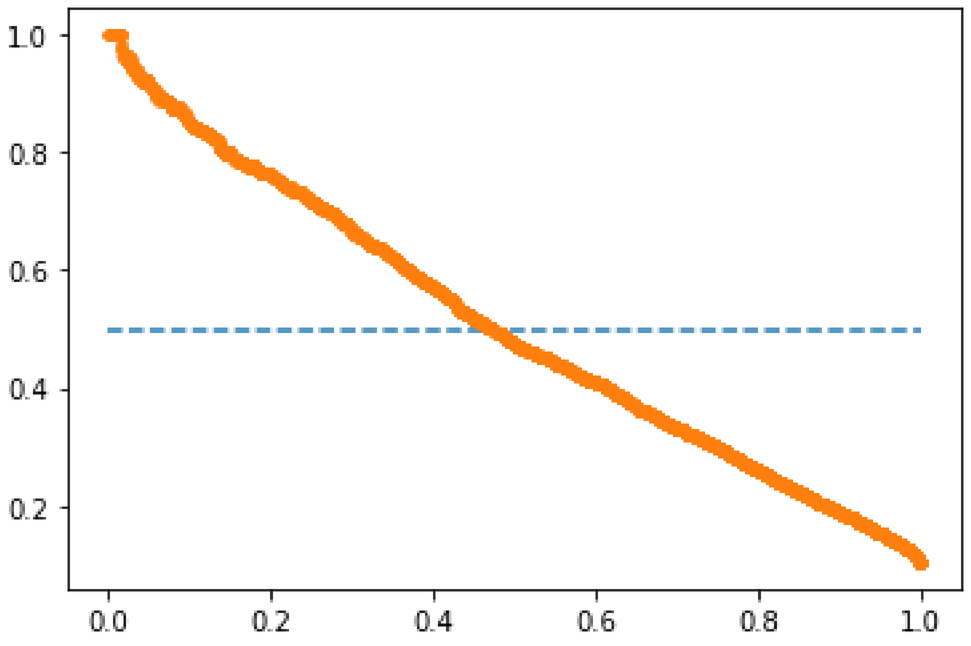
```
precision    recall  f1-score   support

           0       0.92      0.99      0.95     44965
           1       0.70      0.27      0.39      5035

    accuracy                           0.91     50000
   macro avg       0.81      0.63      0.67     50000
weighted avg       0.90      0.91      0.90     50000
```

ROC Curve :



```
AUC: 0.856
```

Precision-Recall curve :

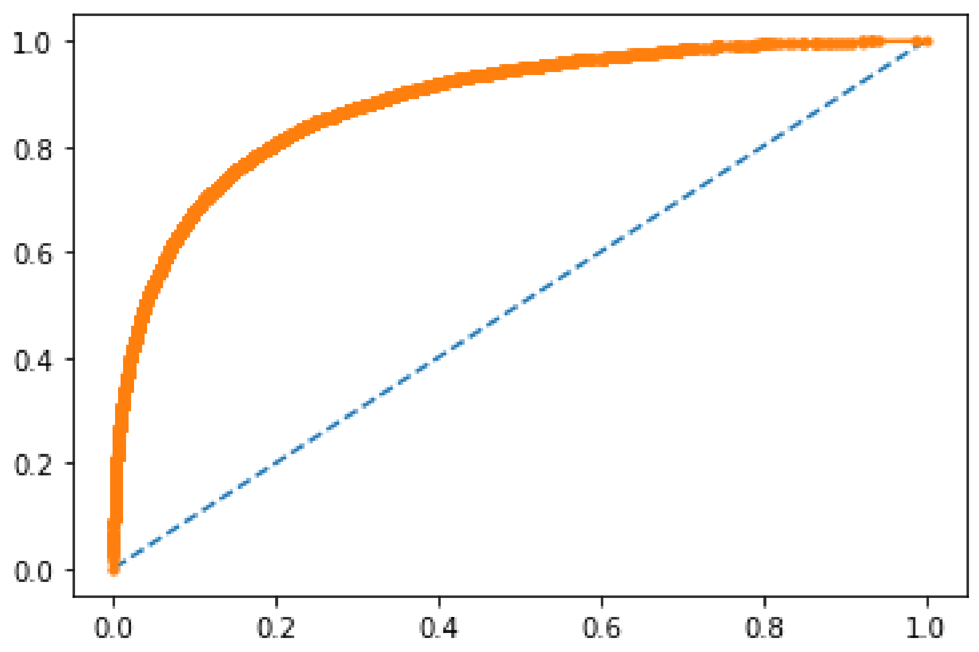


```
AUC: 0.507
```

<u>Naive Bayes Model:</u>

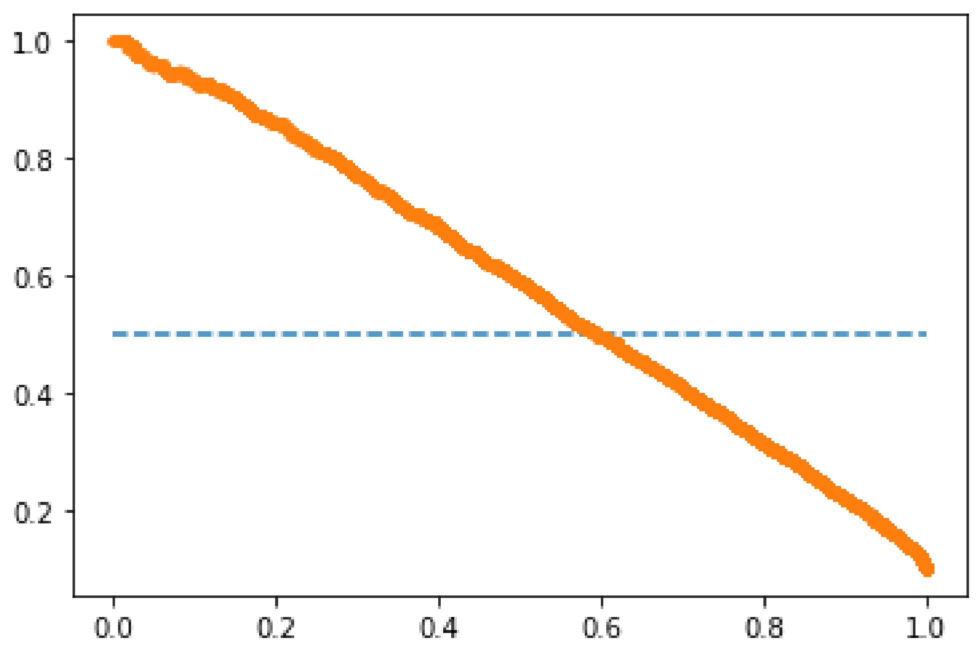
Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class. For example, a phone may be considered as smart if it is having touch screen, internet facility, good camera etc. Though all these features are dependent on each other, they contribute independently to the probability of that the phone is a smart phone.

```
Accuracy: 0.915
              precision    recall  f1-score   support

           0       0.93      0.98      0.96     44965
           1       0.71      0.36      0.48      5035

    accuracy                           0.92     50000
   macro avg       0.82      0.67      0.72     50000
weighted avg       0.91      0.92      0.91     50000
```



```
AUC: 0.885
```

<u>Precision-Recall curve :</u>



```
AUC: 0.584
```

Reason Behind choosing Precision Recall Curve over ROC Curve

Accuracy :

Accuracy simply measures the number of correct predicted samples over the total number of samples. For instance, if the classifier is 90% correct, it means that out of 100 instance it correctly predicts the class for 90 of them.

$$\text{accuracy} = \frac{\text{nr. correct predictions}}{\text{nr. total predictions}} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

This can be misleading if the number of samples per class in your problem is unbalanced. Having a dataset with two classes only, where the first class is 90% of the data, and the second completes the remaining 10%. If the classifier predicts every sample as belonging to the first class, the accuracy reported will be of 90% but this classifier is in practice useless.

With imbalanced classes, it’s easy to get a high accuracy without actually making useful predictions. So, accuracy as an evaluation metrics makes sense only if the class labels are uniformly distributed

Precision and Recall :

Precision and Recall are two metrics computed for each class. They can be easily explained through an example, imagine that we want to evaluate how well does a robot selects good apples from rotten apples There are “m” good apples and “n” rotten apples in a basket. A robot looks into the basket and picks out all the good apples, leaving the rotten apples behind, but is not perfect and could sometimes mistake a rotten apple for a good apple orange.
When the robot finishes, regarding the good apples, precision and recall means:

- Precision: number of good apples picked out of all the apples picked out;
- Recall: number of good apples picked out of all the apples in the basket;

Precision is about exactness, classifying only one instance correctly yields 100% precision, but a very low recall, it tells us how well the system identifies samples from a given class.
Recall is about completeness, classifying all instances as positive yields 100% recall, but a very low precision, it tells how well the system does and identify all the samples from a given class.

Typically these two metrics are combined together in a metric called F1F1 (i.e., harmonic mean of precision and recall), which eases comparison of different systems, and problems with many classes. They are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Receiver Operating Characteristic (ROC) Curves :

The curve is a plot of **false positive rate (x-axis)** versus **the true positive rate (y-axis)** for a number of different candidate threshold **values between 0.0 and 1.0.** An operator may plot the ROC curve and choose a threshold that gives a desirable balance between the false positives and false negatives.

- **x-axis**: the false positive rate is also referred to as the inverted specificity where specificity is the total number of true negatives divided by the sum of the number of true negatives and false positives.
- **y-axis**: the true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

Precision-Recall Curve :

A Precision-Recall curve is a plot of the Precision (y-axis) and the Recall (x-axis) for different thresholds, much like the ROC curve. Note that in computing precision and recall there is never a use of the true negatives, these measures only consider correct predictions

Summary

If you have an imbalanced dataset **accuracy** can give you false assumptions regarding the classifier's performance, it's better to rely on **precision** and **recall**, in the same way a Precision-Recall curve is better to calibrate the probability threshold in an imbalanced class scenario as a ROC curve.

- **ROC Curves**: summarise the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.
- **Precision-Recall curves**: summarise the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets. In both cases the area under the curve (AUC) can be used as a summary of the model performance.