Santander Customer Transaction Prediction

by Shubh Gupta

Problem Background :-

At Santander, the 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?

Problem Statement :-

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

Dataset :-

Training Data

The training data have 200000 observations and 202 variables.

	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
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	1.6910	18.5227	-2.3978	7.8784
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	10.9516	15.4305	2.0339	8.1267
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	1.6858	21.6042	3.1417	-6.5213
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	1.4214	23.0347	-1.2706	-2.9275
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	9.1942	13.2876	-1.5121	3.9267

Test Data

The test data contains 200000 observations and 201 variables.

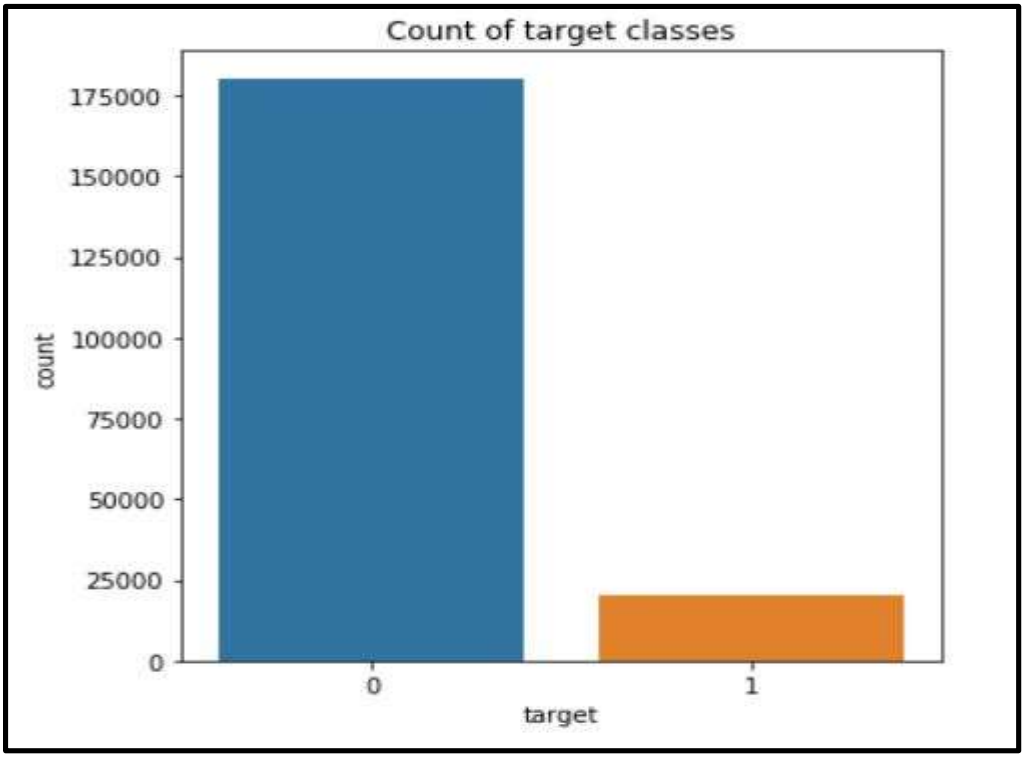
	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	...	-2.1556	11.8495	-1.4300	2.4508	13.7112	2.4669	4.3654
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	...	10.6165	8.8349	0.9403	10.1282	15.5765	0.4773	-1.4852
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	-7.1086
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	3.9567
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	-5.1612

Removing ID_code from both train and test data.

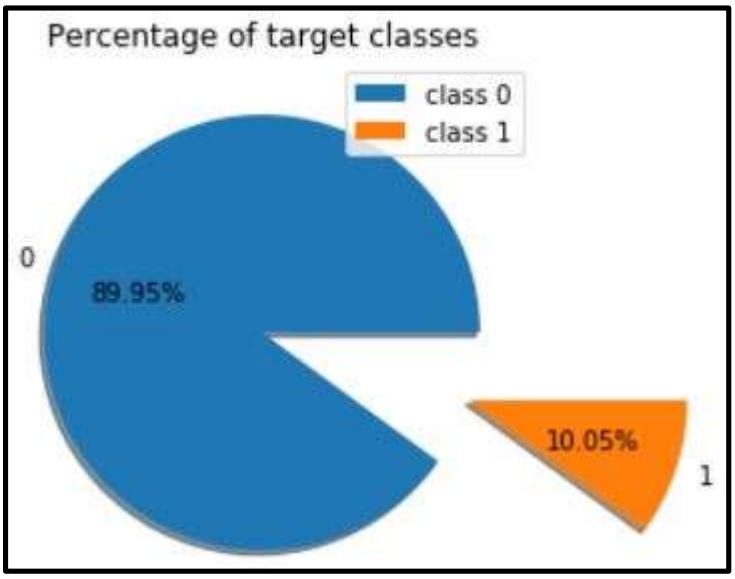
Distribution of Target Variable :-

We will check the distribution of the target variable by plotting a bar graph and pie chart.

Bar Graph



Pie Chart



Observations :-

1. The dataset is highly imbalanced.
2. The number of data points in class 0 is approximately 90%.
3. The number of data points in class 1 is approximately 10%.
4. The number of people that will make transactions are very less as compared to those who will not make.

Missing Value Analysis :-

Missing Values in training data

	count
target	0
var_0	0
var_1	0
var_2	0
var_3	0
var_4	0
var_5	0
var_6	0
var_7	0
var_8	0

Missing Values in test data

	count
var_0	0
var_1	0
var_2	0
var_3	0
var_4	0
var_5	0
var_6	0
var_7	0
var_8	0
var_9	0

Observation :-

No missing values in both the train and test dataset.

Splitting the Train Data :-

Splitting the train data into dependent and independent variables to check the multicollinearity between the independent variables.

Independent Variables

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196
0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.9200	5.7470	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784
1	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.1468	8.0851	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267
2	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193	5.9525	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213
3	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609	8.2450	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275
4	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654	7.6784	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267

Dependent Variable

0	0
1	0
2	0
3	0
4	0

Multicollinearity Analysis :-

We will calculate VIF(Variance Inflation Factor) for each independent variable to check the existence of multicollinearity.

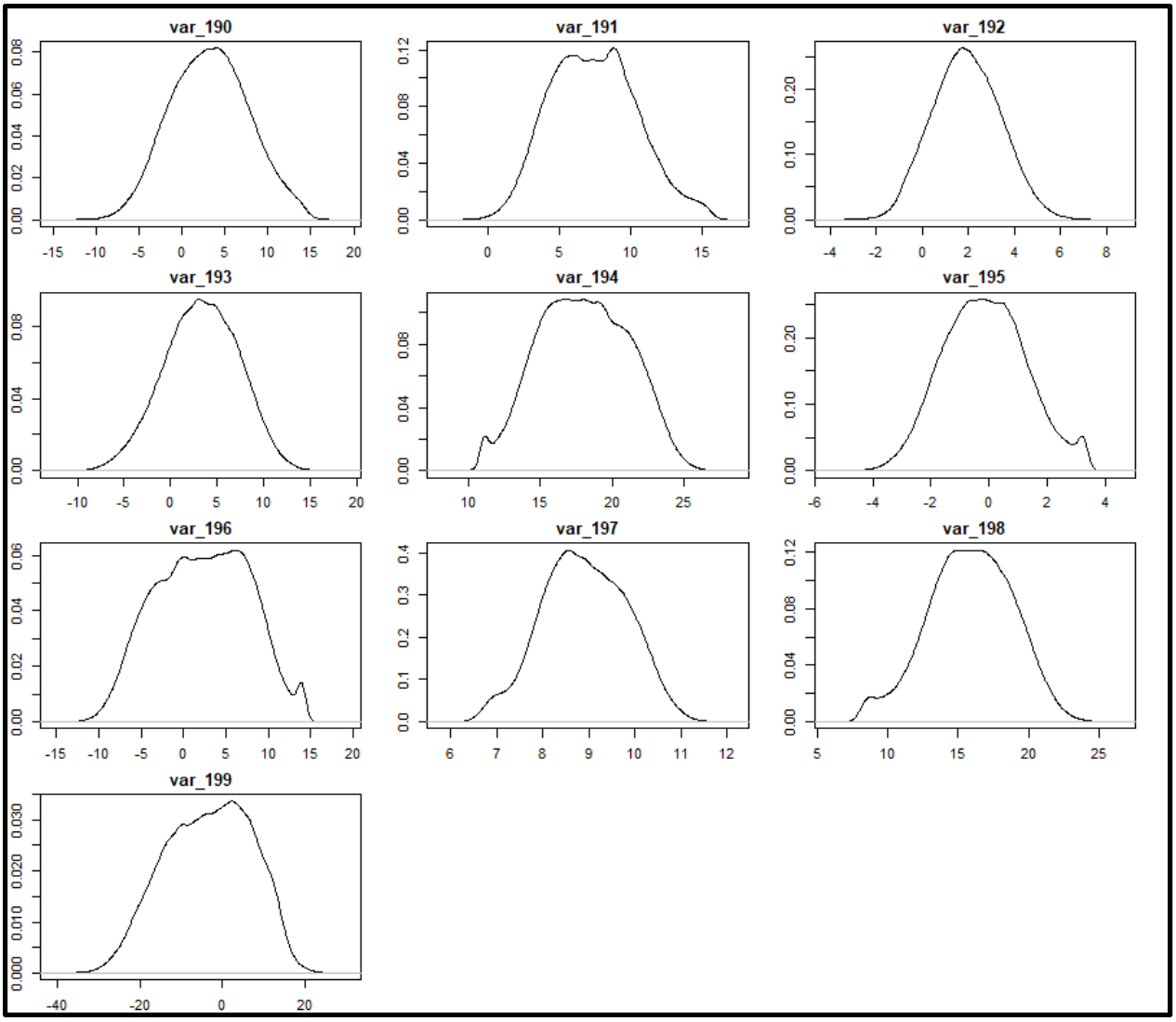
	var	vif
0	var_0	1.0
1	var_1	1.0
2	var_2	1.0
3	var_3	1.0
4	var_4	1.0
5	var_5	1.0
6	var_6	1.0
7	var_7	1.0
8	var_8	1.0
9	var_9	1.0

Observation :-

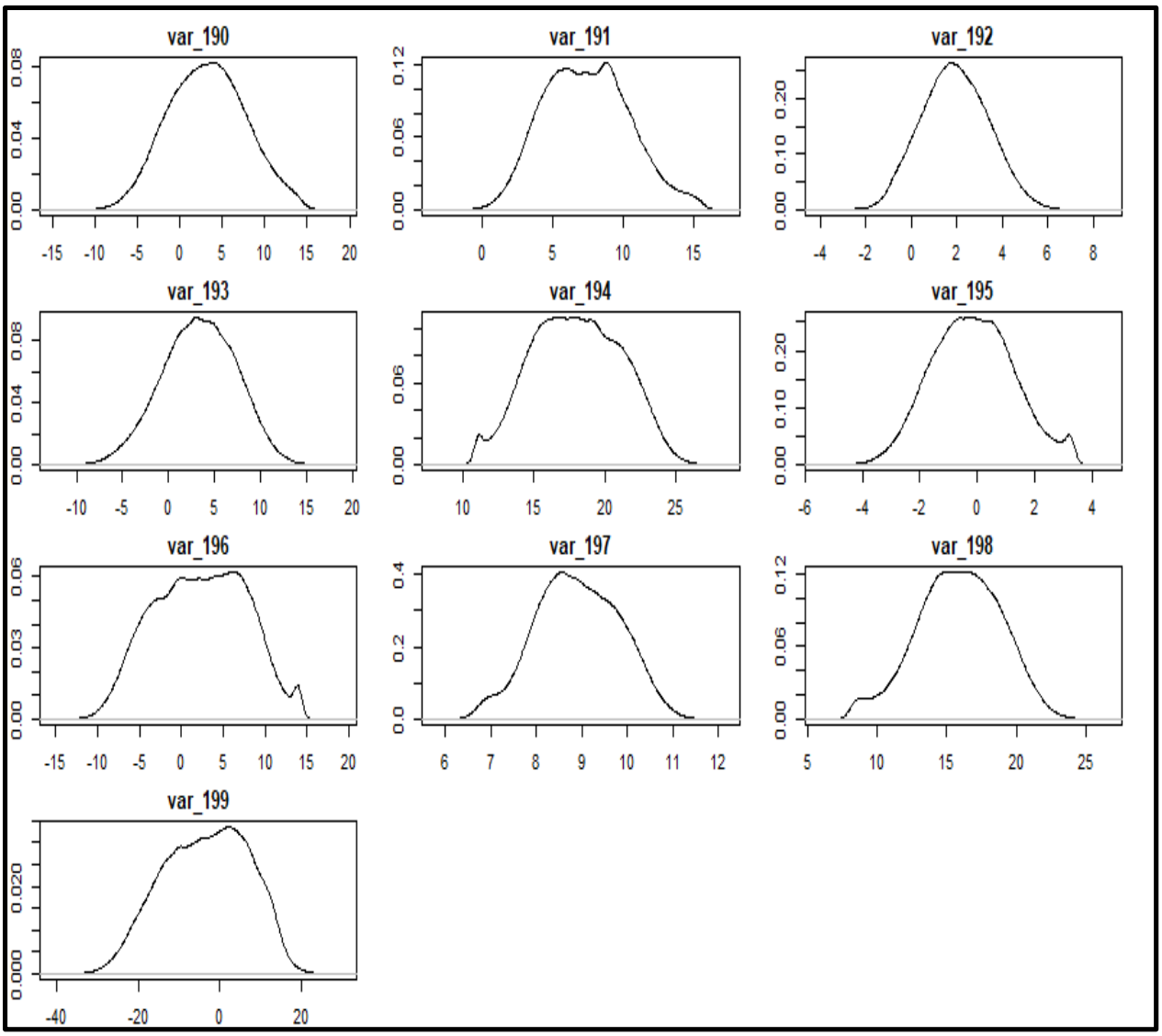
VIF of all the variables is 1 which is acceptable according to the rule of thumb.

Distribution of Independent Variables :-

Distribution for train data



Distribution for Test Data



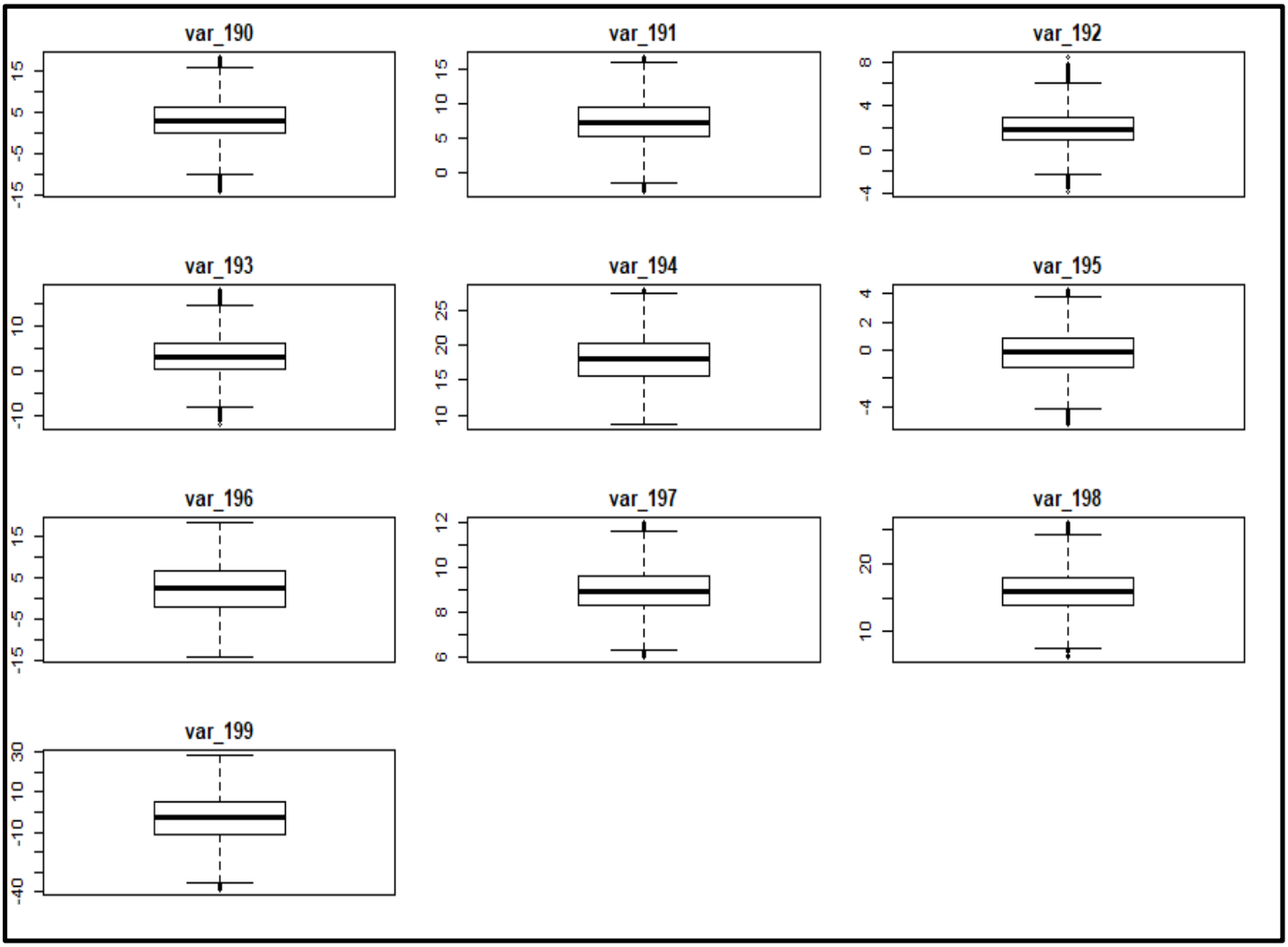
Observation :-

Both the train and test data are almost normally distributed.

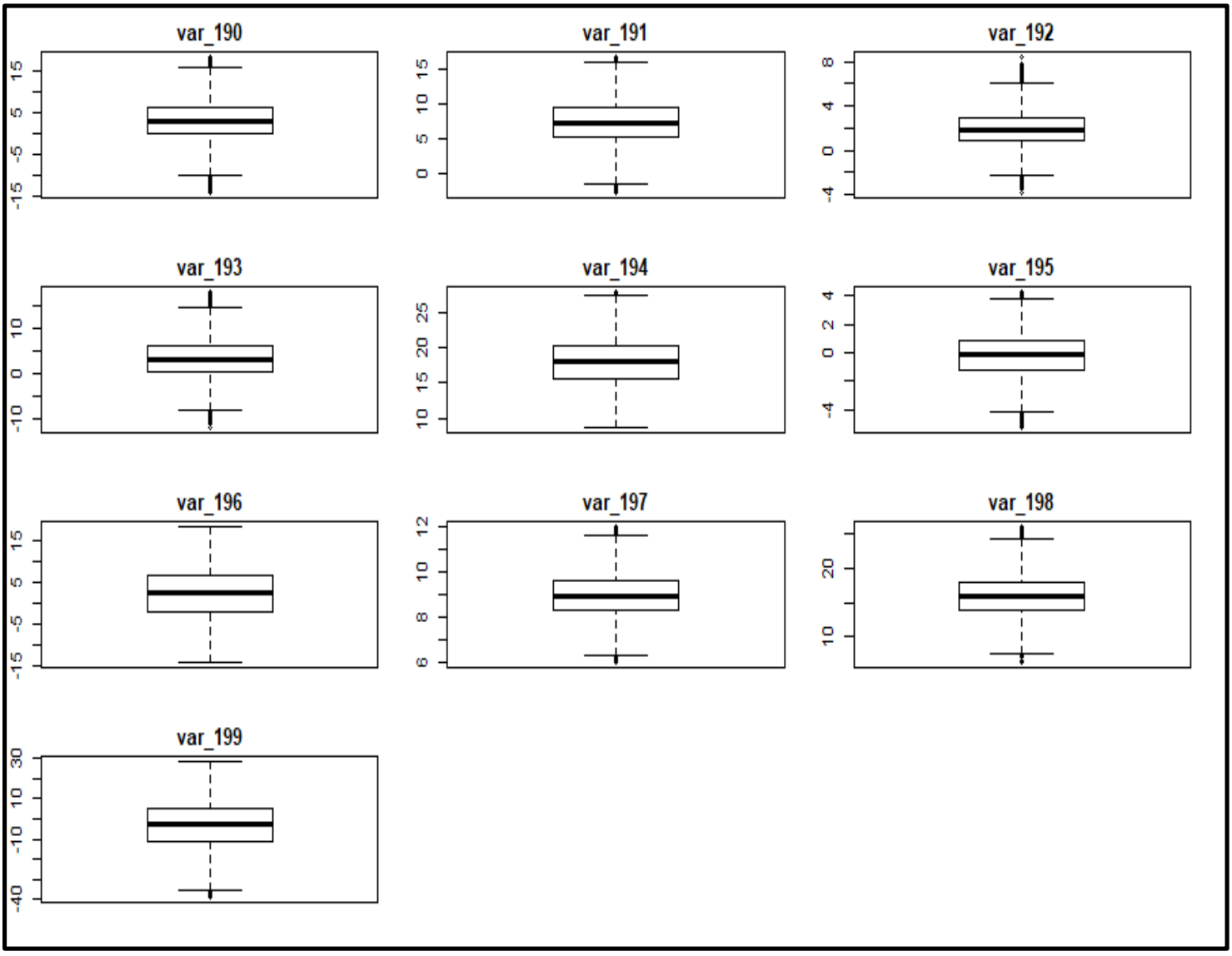
Outlier Analysis :-

We can check outliers in the dataset by plotting box plots.

Box Plots for the Train Data



Box Plots for the Test Data



Removing Outliers :-

We will calculate upper fence and lower fence based on IQR(Interquartile Range).
Data points which are falling below the lower fence and above the upper fence will be declared as outliers.

Remove Outliers from the Train Data

```
var_0 : iqr : 4.3043499999999995 : mimimum : 1.9973250000000018 : maximum : 19.214725
var_1 : iqr : 6.09865 : mimimum : -13.888000000000002 : maximum : 10.5066
var_2 : iqr : 3.794225000000001 : mimimum : 3.031137499999998 : maximum : 18.208037500000003
var_3 : iqr : 3.0700249999999993 : mimimum : 0.6490375000000013 : maximum : 12.9291375
var_4 : iqr : 2.377950000000002 : mimimum : 6.316249999999997 : maximum : 15.828050000000005
var_5 : iqr : 12.12515 : mimimum : -29.388075 : maximum : 19.112525
var_6 : iqr : 1.2352999999999996 : mimimum : 2.91475 : maximum : 7.855949999999998
var_7 : iqr : 5.159100000000002 : mimimum : 6.205149999999996 : maximum : 26.841550000000005
var_8 : iqr : 5.2557 : mimimum : -10.20135 : maximum : 10.821449999999999
var_9 : iqr : 1.9656249999999993 : mimimum : 3.6703625000000013 : maximum : 11.532862499999998
var_10 : iqr : 7.977875000000001 : mimimum : -15.561762500000002 : maximum : 16.349737500000003
var_11 : iqr : 8.363425 : mimimum : -20.0557375 : maximum : 13.397962499999998
var_12 : iqr : 0.2701999999999991 : mimimum : 13.488700000000001 : maximum : 14.569499999999998
var_13 : iqr : 7.201975 : mimimum : -5.7301625 : maximum : 23.077737499999998
var_14 : iqr : 3.488549999999999 : mimimum : 0.549050000000002 : maximum : 14.503249999999998
var_15 : iqr : 0.611699999999999 : mimimum : 13.345250000000002 : maximum : 15.792049999999998
var_16 : iqr : 3.603624999999999 : mimimum : 2.0468375000000014 : maximum : 16.4613375
var_17 : iqr : 9.665450000000002 : mimimum : -24.974400000000003 : maximum : 13.687400000000004
var_18 : iqr : 11.835375000000003 : mimimum : -8.575112500000007 : maximum : 38.76638750000001
```

Remove Outliers from the Test Data

```
var_0 : iqr : 4.296624999999999 : mimimum : 1.9980375000000024 : maximum : 19.184537499999998
var_1 : iqr : 6.043525 : mimimum : -13.7654125 : maximum : 10.4086875
var_2 : iqr : 3.7594250000000002 : mimimum : 3.0964624999999995 : maximum : 18.134162500000002
var_3 : iqr : 3.0971 : mimimum : 0.5848500000000003 : maximum : 12.97325
var_4 : iqr : 2.3623250000000002 : mimimum : 6.347587499999999 : maximum : 15.7968875
var_5 : iqr : 12.143975 : mimimum : -29.4173625 : maximum : 19.1585375
var_6 : iqr : 1.2332 : mimimum : 2.9227999999999996 : maximum : 7.8556
var_7 : iqr : 5.160650000000002 : mimimum : 6.192924999999996 : maximum : 26.835525000000004
var_8 : iqr : 5.233925 : mimimum : -10.154787500000001 : maximum : 10.780912500000001
var_9 : iqr : 1.9610249999999985 : mimimum : 3.6822625000000024 : maximum : 11.526362499999996
var_10 : iqr : 7.9884 : mimimum : -15.608600000000003 : maximum : 16.345000000000002
var_11 : iqr : 8.354525 : mimimum : -20.0537875 : maximum : 13.3643125
var_12 : iqr : 0.27190000000000225 : mimimum : 13.483149999999995 : maximum : 14.570750000000004
var_13 : iqr : 7.197524999999999 : mimimum : -5.722912499999998 : maximum : 23.067187499999996
var_14 : iqr : 3.5016250000000015 : mimimum : 0.5170624999999971 : maximum : 14.523562500000004
var_15 : iqr : 0.6132000000000009 : mimimum : 13.342599999999997 : maximum : 15.7954
var_16 : iqr : 3.581100000000001 : mimimum : 2.082749999999998 : maximum : 16.40715
var_17 : iqr : 9.6863 : mimimum : -25.02735 : maximum : 13.717849999999999
var_18 : iqr : 11.776800000000001 : mimimum : -8.427500000000002 : maximum : 38.679700000000004
```

Observation :-

There are 26536 outliers in the train data and 27091 outliers in the test data.

Missing Values Imputation :-

First we will replace all the outliers with the Null values and then replace all the null values with the mean of that particular variable.

Imputing Mean Values in the Train data

```
var_0      0.0520
var_1      0.0030
var_2      0.0245
var_3      0.0110
var_4      0.0380
             ...
var_195    0.0660
var_196    0.0000
var_197    0.0255
var_198    0.0470
var_199    0.0100
```

Imputing Mean Values in the Test Data

```
var_0      0.0610
var_1      0.0020
var_2      0.0280
var_3      0.0065
var_4      0.0445
             ...
var_195    0.0520
var_196    0.0000
var_197    0.0255
var_198    0.0575
var_199    0.0070
```

Standardization :-

Since both the train and test data are normally distributed, we will use Standardization for feature scaling in order to scale all the variables of both train and test data to have a mean of 0 and variance or standard deviation of 1.

Model Training :-

We will use following three models for training the dataset in both R and Python and then compare the performance of these models to select the best one:-

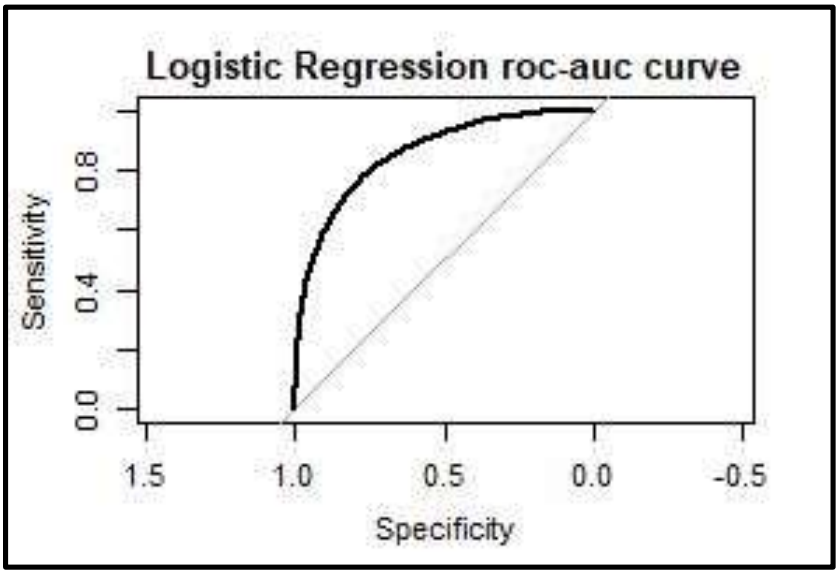
1. *Logistic Regression*
2. *Decision Tree*
3. *XGBoost*

Performance Metrics :-

We will check the performance of each model with the help of following eight performance metrics :-

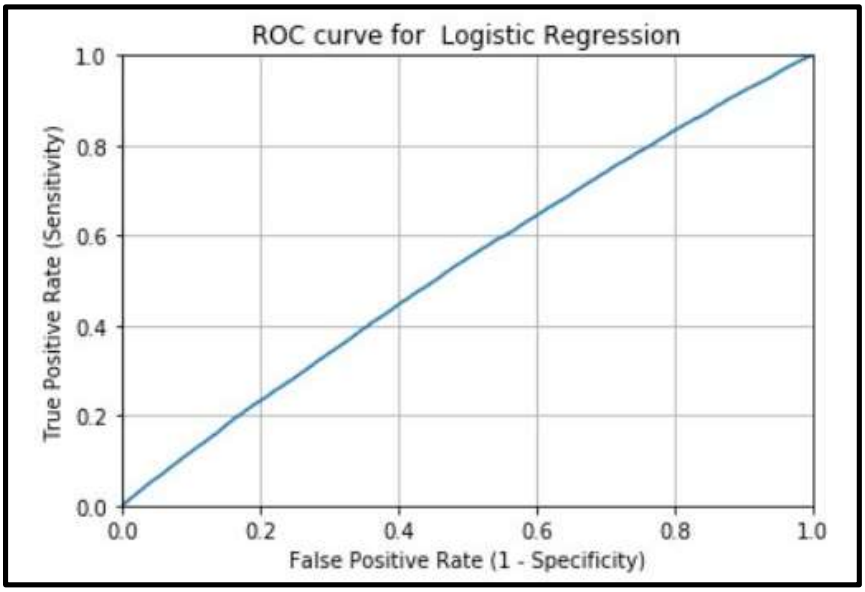
1. *Accuracy*
2. *Recall*
3. *Precision*
4. *Specificity*
5. *False Positive Rate*
6. *False Negative Rate*
7. *F1 Score*
8. *ROC-AUC curve*

Logistic Regression in R :-



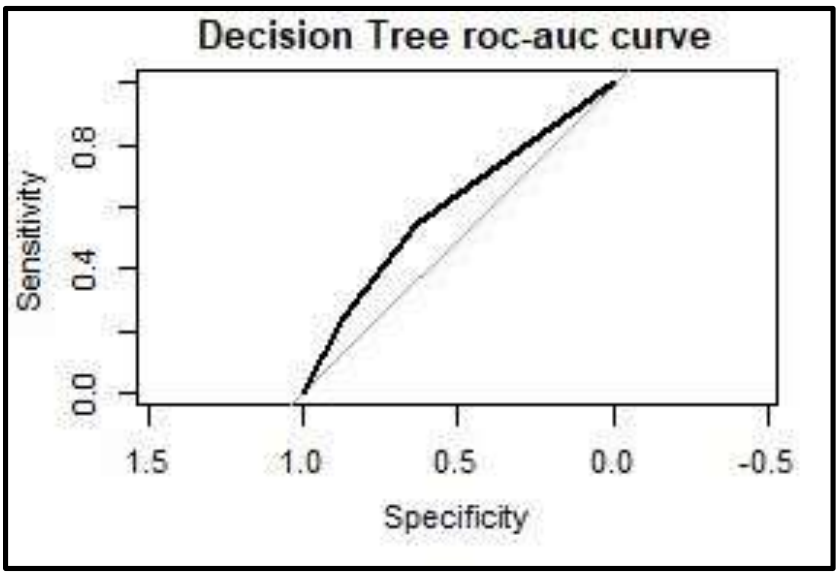
Evaluation Metrics	***Percentage Values***
Accuracy	78%
Recall	78%
Precision	97%
Specificity	77%
FPR	23%
FNR	22%
F1 Score	87%
AUC	85%

Logistic Regression in Python :-



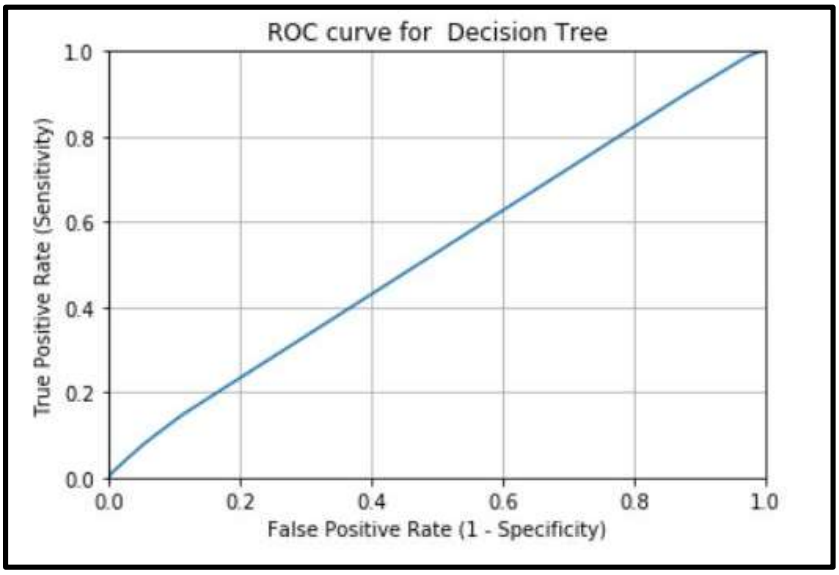
Evaluation Metrics	***Percentage Values***
Accuracy	52%
Recall	53%
Precision	11%
Specificity	52%
FPR	48%
FNR	47%
F1 Score	18%
AUC	53%

Decision Tree in R :-



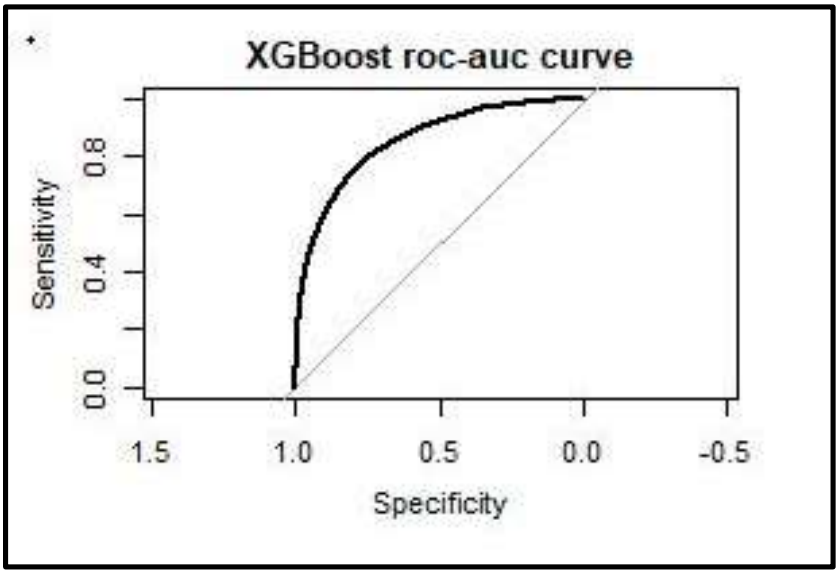
Evaluation Metrics	***Percentage Values***
Accuracy	62%
Recall	63%
Precision	93%
Specificity	54%
FPR	46%
FNR	37%
F1 Score	75%
AUC	59%

Decision Tree in Python :-



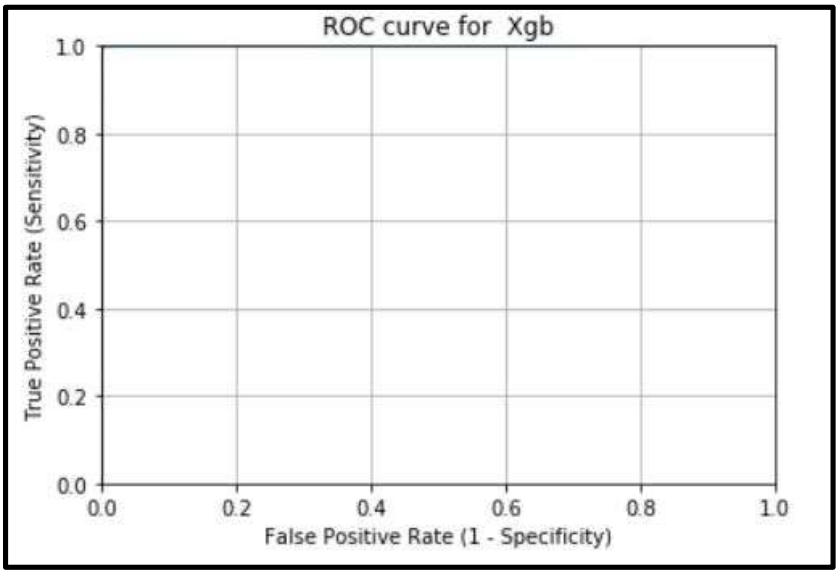
Evaluation Metrics	***Percentage Values***
Accuracy	81%
Recall	15%
Precision	13%
Specificity	88%
FPR	12%
FNR	85%
F1 Score	14%
AUC	53%

XGBoost in R :-



Evaluation Metrics	***Percentage Values***
Accuracy	78%
Recall	78%
Precision	97%
Specificity	77%
FPR	23%
FNR	22%
F1 Score	87%
AUC	85%

XGBoost in Python :-



Evaluation Metrics	***Percentage Values***
Accuracy	99%
Recall	97%
Precision	99%
Specificity	99%
FPR	0.005%
FNR	2%
F1 Score	98%
AUC	100%

Models Comparison in R :-

	Accuracy	*Recall*	*Precision*	*Specificity*	*FPR*	*FNR*	*F1 Score*	*AUC*
Logistic Regression	0.78	0.78	0.97	0.77	0.23	0.22	0.87	0.85
Decision Tree	0.62	0.63	0.93	0.54	0.46	0.37	0.75	0.59
XGBoost	0.78	0.78	0.97	0.77	0.23	0.22	0.87	0.85

Models Comparison in Python :-

	Accuracy	*Recall*	*Precision*	*Specificity*	*FPR*	*FNR*	*F1 Score*	*AUC*
Logistic Regression	0.52	0.53	0.11	0.52	0.48	0.47	0.18	0.53
Decision Tree	0.81	0.15	0.13	0.88	0.12	0.85	0.14	0.53
XGBoost	0.99	0.97	0.99	0.99	0.00005	0.02	0.98	1.00

Model Selection Parameters :-

The model should be selected based on the following parameters :-

1. *High Accuracy*
2. *High F1 Score*
3. *High AUC Score*
4. *Low FPR*
5. *Low FNR*

Freezed Model :-

Since XGBoost performs much better than other models based on the above parameters in both R and Python, we will freeze XGBoost as our final model for predicting the target class of test data.

Conclusion :-

The above freezed model can be used for ***Santander Customer Transaction Prediction*** *problem to identify* ***which customers will make a specific transaction in the future, irrespective of the amount of money transacted.***