# Santander Customer Transaction Prediction

Nagarjuna Yegavinti

# 1. Introduction

At Santander, mission is to help people and businesses prosper. They are looking for ways to help their customers understand financial health and identify which products and services might help them achieve their monetary goals.

The data science team at Santander is continually challenging machine learning algorithms, working with the global data science community to make sure they 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?

# 2. Problem Statement

In this challenge, there is a need to identify which customer will make a specific transaction in the future. Based on the previous data provided whether the customer will do transaction in the future irrespective of the amount of money transacted. So, predicting the probability of customer transaction is a classification problem.

# 3. Data Set

Prediction of Santander customer transaction has been provided with historical data for which model has to be developed and developed model should be applied on test cases.

## 3.1. Number of attributes

The training dataset provided contains total of 201 variables starting from ID_code to var_199 of which var_1 to var_199 are numeric feature variables, a binary target column, and a string ID_code column. The test data set contains all the variables except the target variable. So, the task is to predict the value of target column in the test set.

```
train.info()
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 200000 entries, 0 to 199999
    Columns: 201 entries, target to var_199
    dtypes: float64(200), int64(1)
    memory usage: 306.7 M
test.info()
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 200000 entries, 0 to 199999
    Columns: 200 entries, var_0 to var_199
    dtypes: float64(200)
    memory usage: 305.2 MB
```

# 4. Data Pre-processing

Data pre-processing is to clean the data so that the machine learning model applied on the data can perform well and give good accuracy and minimum error rate. Cleaning of the data includes exploratory data analysis, missing value analysis, outlier analysis, feature selection and feature scaling.

## 4.1.Exploratory data analysis

Data exploration includes driving new attributes from the existing attributes so that they are more meaning full and making them operational by machine learning model.

## 4.2.Target variable distribution

In fig.1 the graph shows that there is imbalence in target classes as the target class belonging to the 1  are very less compared class zero.

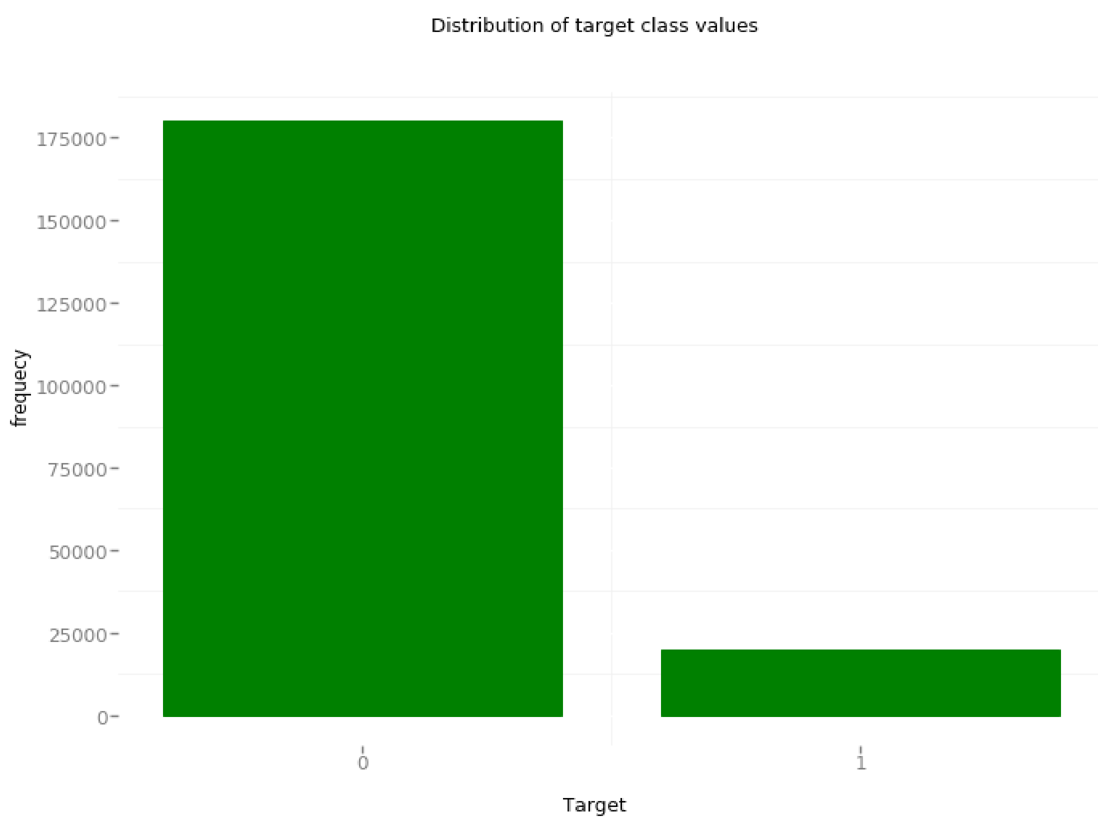


Figure.1. Bar plot for checking the distribution of target class values.

## 4.3.Missing value Analysis:

The data provided for santandar customer transaction prediction does not contain any missing values. With the below function in r it shows missing values are not present in both train and test data.

```
> sum(is.na(train))
[1] 0
> sum(is.na(test))
[1] 0
```

## 4.4. Outlier Analysis:

Outliers will be checked by distributing the values of a data between the 25$^{th}$ and 75$^{th}$ percentile. The values which comes beyond these ranges will be treated as an outlier. In this data we are not going for outlier treatment as the machine algorithm are preforming well even if there are outlier, and it is a time-consuming step to treat outliers on huge amount of data.

## 4.5. Feature selection:

Feature selection is done to remove the redundant information from the data as correlation analysis on numerical data, chi square test on categorical data and Annova between the categorical and continuous variables are performed to check the redundant information. Since the data provided for Santander customer transaction is numeric data I performed

### 4.5.1.Correlation Analysis

In fig. 2 the correlation plot clearly shows that there is no positive correlation between the variables instead some of the variables are highly negatively correlated. Even though the variables are highly correlated, I am not removing those variables as all variables are almost similarly negatively correlated.

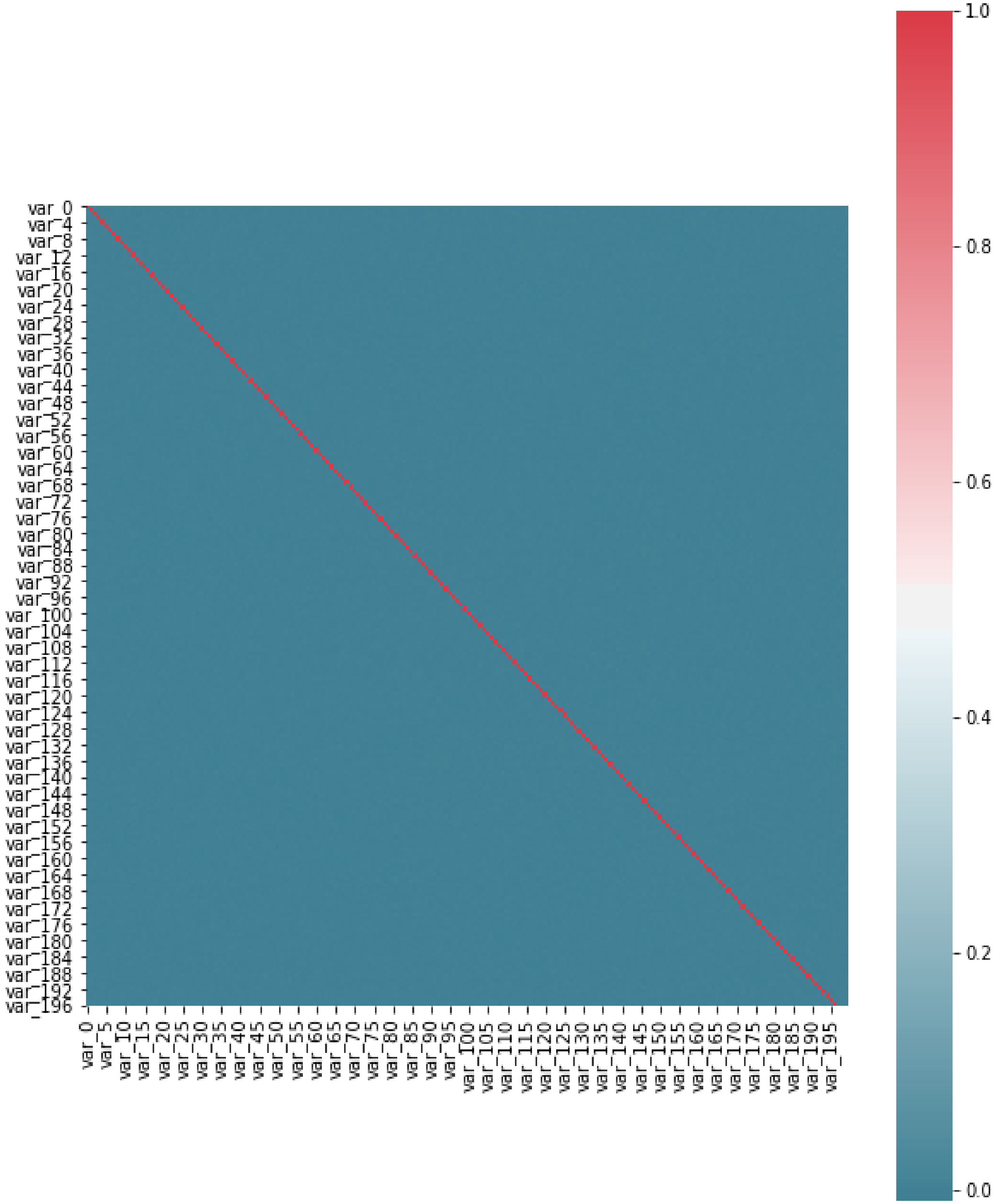


Figure:2. Correlation Analysis between continuous variables.

## 4.6. Feature Scaling:

As feature scaling is done to bring all the variables in common range and can be compared in common ground and in this data most of the variables are normally distributed I did standardisation. Even if I did the standardisation there is not much change in the prediction instead there is an increase in the false prediction so, I did not use standardisation.

# 5. Division of data:

For any machine learning problem, we should have historical data which has to be divided into data on which algorithm has to be trained and the data on which performance of the machine algorithm has to be evaluated. In python I used simple random sampling which did not result in good performance of any model, while in R I used stratified sampling for splitting the data with the function called createDataPartition. Due to target class imbalance in the data provided I tried up sampling and under sampling and SMOTE ( synthetic minority oversampling). In R I got almost same performance as without the sampling techniques with the all sampling techniques, while in python the performance of the model is worst compared the without the sampling techniques.

# 6. Modelling:

## 6.1. Model development

Since prediction of customer transaction is a classification problem the machine algorithms used for prediction of customer transaction on the data are

1. Logistic regression
2. Decision Tree Classification
3. Random Forest classification
4. Naïve Bayes Algorithm
5. KNN classification algorithm

### 6.1.1. Logistic Regression

Logistic regression uses the logistic function to estimate or calculate the predictions where the predictions are either binomial, ordinal or multinomial. Here, for this data I used glm function in r to train the model on training data while in case of python it is logit function. The summary of built model as follows.

Call:

```
glm(formula = target ~ ., family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.5926  -0.3996  -0.2322  -0.1227   3.8328  

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept)  5.846e+01  8.277e+00   7.063 1.63e-12 ***
var_0        5.733e-02  3.735e-03  15.351  < 2e-16 ***
var_1        4.116e-02  2.850e-03  14.441  < 2e-16 ***
var_2        6.301e-02  4.305e-03  14.635  < 2e-16 ***
var_3        1.891e-02  5.656e-03   3.344 0.000826 ***
var_4        2.566e-02  7.106e-03   3.610 0.000306 ***
var_5        1.457e-02  1.463e-03   9.955  < 2e-16 ***
var_6        2.713e-01  1.322e-02  20.530  < 2e-16 ***
var_7        4.188e-04  3.389e-03   0.124 0.901639    
var_8        2.192e-02  3.487e-03   6.286 3.25e-10 ***
var_9       -1.066e-01  9.306e-03 -11.459  < 2e-16 ***
var_10       8.543e-05  2.103e-03   0.041 0.967595    
var_11       1.191e-02  1.933e-03   6.165 7.05e-10 ***
var_12      -1.151e+00  5.996e-02 -19.196  < 2e-16 ***
var_13      -3.912e-02  2.468e-03 -15.854  < 2e-16 ***
var_14      -1.227e-02  5.164e-03  -2.376 0.017485 *  
var_15       1.272e-01  2.815e-02   4.518 6.25e-06 ***
var_16       5.868e-03  4.514e-03   1.300 0.193609    
var_17      -1.897e-04  1.724e-03  -0.110 0.912390    
var_18       1.826e-02  1.465e-03  12.463  < 2e-16 ***
var_19       3.490e-03  1.441e-03   2.423 0.015401 *  
var_20      -8.652e-03  1.967e-03  -4.398 1.09e-05 ***
var_21      -2.230e-02  1.413e-03 -15.785  < 2e-16 ***
var_22       7.012e-02  3.999e-03  17.533  < 2e-16 ***
var_23      -1.867e-01  2.197e-02  -8.497  < 2e-16 ***
var_24       2.812e-02  3.045e-03   9.233  < 2e-16 ***
var_25       1.805e-01  4.029e-02   4.482 7.41e-06 ***
var_26       3.384e-02  1.921e-03  17.620  < 2e-16 ***
var_27      -6.834e-03  7.585e-03  -0.901 0.367604    
var_28      -1.232e-01  1.477e-02  -8.343  < 2e-16 ***
var_29       9.478e-03  4.418e-03   2.145 0.031917 *  
var_30       3.534e-04  1.456e-03   0.243 0.808206    
var_31      -4.232e-02  5.373e-03  -7.877 3.36e-15 ***
var_32       3.844e-02  4.460e-03   8.617  < 2e-16 ***
var_33      -3.435e-02  2.686e-03 -12.786  < 2e-16 ***
var_34      -3.217e-01  2.133e-02 -15.082  < 2e-16 ***
var_35       2.443e-02  2.228e-03  10.963  < 2e-16 ***
var_36      -4.431e-02  3.713e-03 -11.936  < 2e-16 ***
var_37       1.375e-02  5.137e-03   2.676 0.007454 ** 
var_38       1.605e-03  2.711e-03   0.592 0.553976    
var_39       3.437e-04  2.847e-03   0.121 0.903898    
var_40       2.004e-02  1.387e-03  14.448  < 2e-16 ***
var_41      -5.719e-04  1.951e-03  -0.293 0.769369    
var_42      -4.177e-02  1.662e-02  -2.513 0.011988 *  
var_43      -3.088e-01  3.728e-02  -8.285  < 2e-16 ***
var_44      -2.803e-02  1.916e-03 -14.635  < 2e-16 ***
var_45      -2.678e-03  5.405e-04  -4.955 7.23e-07 ***
var_46       3.209e-03  4.037e-03   0.795 0.426739    
var_47       2.829e-03  1.096e-03   2.582 0.009829 ** 
var_48       8.486e-03  1.012e-03   8.386  < 2e-16 ***
var_49       1.361e-02  1.469e-03   9.261  < 2e-16 ***
```

| | | | | | |
|---|---|---|---|---|---|
| var_50 | -7.034e-02 | 1.669e-02 | -4.215 | 2.50e-05 | *** |
| var_51 | 9.383e-03 | 1.404e-03 | 6.683 | 2.34e-11 | *** |
| var_52 | 1.900e-02 | 2.318e-03 | 8.198 | 2.44e-16 | *** |
| var_53 | 2.855e-01 | 1.499e-02 | 19.047 | < 2e-16 | *** |
| var_54 | -7.997e-03 | 1.370e-03 | -5.835 | 5.38e-09 | *** |
| var_55 | 1.330e-02 | 2.031e-03 | 6.549 | 5.78e-11 | *** |
| var_56 | -3.195e-02 | 3.235e-03 | -9.878 | < 2e-16 | *** |
| var_57 | -7.255e-02 | 1.455e-02 | -4.985 | 6.19e-07 | *** |
| var_58 | -1.875e-02 | 2.688e-03 | -6.974 | 3.09e-12 | *** |
| var_59 | -4.931e-02 | 1.353e-02 | -3.645 | 0.000268 | *** |
| var_60 | 4.567e-03 | 2.734e-03 | 1.670 | 0.094826 | . |
| var_61 | 2.801e-03 | 9.969e-04 | 2.809 | 0.004964 | ** |
| var_62 | 2.409e-02 | 5.685e-03 | 4.238 | 2.26e-05 | *** |
| var_63 | -1.256e-02 | 3.708e-03 | -3.388 | 0.000703 | *** |
| var_64 | -3.604e-02 | 7.746e-03 | -4.653 | 3.27e-06 | *** |
| var_65 | 9.062e-03 | 3.078e-03 | 2.944 | 0.003243 | ** |
| var_66 | 5.757e-02 | 1.023e-02 | 5.625 | 1.86e-08 | *** |
| var_67 | 2.012e-02 | 1.571e-03 | 12.809 | < 2e-16 | *** |
| var_68 | -4.800e+00 | 1.607e+00 | -2.986 | 0.002827 | ** |
| var_69 | 3.894e-03 | 2.923e-03 | 1.332 | 0.182800 | |
| var_70 | 7.298e-03 | 9.702e-04 | 7.522 | 5.40e-14 | *** |
| var_71 | 3.931e-01 | 4.325e-02 | 9.088 | < 2e-16 | *** |
| var_72 | -9.956e-03 | 2.928e-03 | -3.400 | 0.000673 | *** |
| var_73 | -2.217e-03 | 1.552e-03 | -1.428 | 0.153172 | |
| var_74 | 4.298e-03 | 8.215e-04 | 5.232 | 1.68e-07 | *** |
| var_75 | -2.058e-02 | 1.901e-03 | -10.825 | < 2e-16 | *** |
| var_76 | -2.576e-02 | 1.443e-03 | -17.853 | < 2e-16 | *** |
| var_77 | -1.238e-02 | 3.039e-03 | -4.073 | 4.65e-05 | *** |
| var_78 | 7.632e-02 | 5.817e-03 | 13.119 | < 2e-16 | *** |
| var_79 | 2.440e-02 | 8.829e-03 | 2.763 | 0.005721 | ** |
| var_80 | -2.615e-02 | 1.524e-03 | -17.161 | < 2e-16 | *** |
| var_81 | -1.161e-01 | 4.915e-03 | -23.613 | < 2e-16 | *** |
| var_82 | 7.994e-03 | 1.361e-03 | 5.873 | 4.27e-09 | *** |
| var_83 | -6.412e-03 | 1.389e-03 | -4.618 | 3.88e-06 | *** |
| var_84 | 4.902e-03 | 1.857e-03 | 2.640 | 0.008292 | ** |
| var_85 | -1.790e-02 | 2.954e-03 | -6.060 | 1.36e-09 | *** |
| var_86 | -1.538e-02 | 1.477e-03 | -10.418 | < 2e-16 | *** |
| var_87 | -2.343e-02 | 2.051e-03 | -11.423 | < 2e-16 | *** |
| var_88 | -2.738e-02 | 4.639e-03 | -5.903 | 3.57e-09 | *** |
| var_89 | 3.775e-02 | 3.229e-03 | 11.689 | < 2e-16 | *** |
| var_90 | 6.443e-03 | 8.790e-04 | 7.330 | 2.30e-13 | *** |
| var_91 | 8.569e-01 | 7.557e-02 | 11.340 | < 2e-16 | *** |
| var_92 | -3.322e-02 | 2.756e-03 | -12.055 | < 2e-16 | *** |
| var_93 | -2.000e-01 | 2.095e-02 | -9.551 | < 2e-16 | *** |
| var_94 | 5.577e-02 | 4.151e-03 | 13.437 | < 2e-16 | *** |
| var_95 | 1.870e-01 | 1.849e-02 | 10.113 | < 2e-16 | *** |
| var_96 | 1.205e-03 | 1.357e-03 | 0.888 | 0.374604 | |
| var_97 | 3.360e-03 | 9.137e-04 | 3.678 | 0.000235 | *** |
| var_98 | -2.017e-02 | 1.619e-02 | -1.246 | 0.212831 | |
| var_99 | 1.129e-01 | 6.146e-03 | 18.364 | < 2e-16 | *** |
| var_100 | -7.953e-04 | 1.259e-03 | -0.632 | 0.527542 | |
| var_101 | -5.497e-03 | 2.333e-03 | -2.356 | 0.018471 | * |
| var_102 | -7.051e-03 | 1.341e-03 | -5.259 | 1.45e-07 | *** |
| var_103 | -8.386e-02 | 6.232e-02 | -1.346 | 0.178414 | |
| var_104 | -4.072e-02 | 5.897e-03 | -6.905 | 5.03e-12 | *** |
| var_105 | 1.011e-01 | 1.348e-02 | 7.501 | 6.36e-14 | *** |
| var_106 | 5.232e-02 | 6.083e-03 | 8.601 | < 2e-16 | *** |
| var_107 | -1.775e-02 | 1.528e-03 | -11.615 | < 2e-16 | *** |
| var_108 | -8.923e-01 | 6.687e-02 | -13.344 | < 2e-16 | *** |
| var_109 | -3.837e-02 | 2.642e-03 | -14.522 | < 2e-16 | *** |
| var_110 | 5.354e-02 | 2.986e-03 | 17.931 | < 2e-16 | *** |
| var_111 | 6.966e-02 | 1.059e-02 | 6.578 | 4.77e-11 | *** |
| var_112 | 5.171e-02 | 7.310e-03 | 7.074 | 1.51e-12 | *** |
| var_113 | -1.021e-02 | 2.583e-03 | -3.953 | 7.70e-05 | *** |

| | | | | | |
|---|---|---|---|---|---|
| var_114 | -8.347e-02 | 1.166e-02 | -7.157 | 8.23e-13 | *** |
| var_115 | -5.925e-02 | 4.384e-03 | -13.515 | < 2e-16 | *** |
| var_116 | -5.158e-02 | 7.008e-03 | -7.360 | 1.84e-13 | *** |
| var_117 | 5.320e-04 | 8.657e-04 | 0.615 | 0.538867 | |
| var_118 | 1.375e-02 | 1.317e-03 | 10.438 | < 2e-16 | *** |
| var_119 | 2.304e-02 | 2.741e-03 | 8.403 | < 2e-16 | *** |
| var_120 | -1.893e-03 | 9.512e-04 | -1.990 | 0.046603 | * |
| var_121 | -8.012e-02 | 6.771e-03 | -11.832 | < 2e-16 | *** |
| var_122 | -2.620e-02 | 2.225e-03 | -11.775 | < 2e-16 | *** |
| var_123 | -2.158e-02 | 1.857e-03 | -11.622 | < 2e-16 | *** |
| var_124 | 6.275e-03 | 4.219e-03 | 1.487 | 0.136922 | |
| var_125 | 2.583e-01 | 3.618e-02 | 7.140 | 9.32e-13 | *** |
| var_126 | 1.939e-02 | 1.486e-02 | 1.305 | 0.191896 | |
| var_127 | -4.098e-02 | 3.684e-03 | -11.122 | < 2e-16 | *** |
| var_128 | 2.834e-02 | 3.559e-03 | 7.964 | 1.67e-15 | *** |
| var_129 | -4.436e-03 | 2.799e-03 | -1.585 | 0.112960 | |
| var_130 | 1.396e-01 | 1.382e-02 | 10.105 | < 2e-16 | *** |
| var_131 | -2.163e-01 | 2.525e-02 | -8.564 | < 2e-16 | *** |
| var_132 | -5.290e-02 | 7.910e-03 | -6.688 | 2.26e-11 | *** |
| var_133 | 4.894e-01 | 3.033e-02 | 16.135 | < 2e-16 | *** |
| var_134 | 9.106e-03 | 1.868e-03 | 4.874 | 1.09e-06 | *** |
| var_135 | 1.139e-02 | 1.510e-03 | 7.544 | 4.57e-14 | *** |
| var_136 | -1.872e-03 | 1.115e-03 | -1.679 | 0.093080 | . |
| var_137 | 1.143e-02 | 1.300e-03 | 8.791 | < 2e-16 | *** |
| var_138 | 1.095e-02 | 2.547e-03 | 4.300 | 1.71e-05 | *** |
| var_139 | -3.004e-02 | 1.487e-03 | -20.196 | < 2e-16 | *** |
| var_140 | 1.196e-02 | 2.369e-03 | 5.048 | 4.46e-07 | *** |
| var_141 | -1.464e-02 | 1.717e-03 | -8.529 | < 2e-16 | *** |
| var_142 | -1.149e-02 | 2.028e-03 | -5.663 | 1.49e-08 | *** |
| var_143 | -9.348e-03 | 3.931e-03 | -2.378 | 0.017399 | * |
| var_144 | 6.398e-02 | 1.251e-02 | 5.114 | 3.15e-07 | *** |
| var_145 | 2.454e-02 | 2.966e-03 | 8.272 | < 2e-16 | *** |
| var_146 | -7.633e-02 | 4.523e-03 | -16.876 | < 2e-16 | *** |
| var_147 | 1.941e-02 | 1.555e-03 | 12.480 | < 2e-16 | *** |
| var_148 | -8.254e-01 | 5.761e-02 | -14.328 | < 2e-16 | *** |
| var_149 | -1.506e-02 | 1.109e-03 | -13.575 | < 2e-16 | *** |
| var_150 | -4.519e-02 | 4.699e-03 | -9.616 | < 2e-16 | *** |
| var_151 | 2.452e-02 | 2.891e-03 | 8.479 | < 2e-16 | *** |
| var_152 | -1.265e-02 | 3.821e-03 | -3.309 | 0.000935 | *** |
| var_153 | -1.020e-02 | 5.771e-03 | -1.768 | 0.077028 | . |
| var_154 | -2.890e-02 | 2.314e-03 | -12.492 | < 2e-16 | *** |
| var_155 | 2.044e-02 | 1.996e-03 | 10.241 | < 2e-16 | *** |
| var_156 | -7.395e-02 | 1.211e-02 | -6.108 | 1.01e-09 | *** |
| var_157 | 1.694e-02 | 2.057e-03 | 8.233 | < 2e-16 | *** |
| var_158 | -2.151e-03 | 1.473e-03 | -1.460 | 0.144253 | |
| var_159 | 1.331e-02 | 2.797e-03 | 4.759 | 1.95e-06 | *** |
| var_160 | -1.750e-03 | 1.062e-03 | -1.647 | 0.099537 | . |
| var_161 | 7.928e-02 | 5.314e-02 | 1.492 | 0.135751 | |
| var_162 | 7.198e-02 | 8.120e-03 | 8.865 | < 2e-16 | *** |
| var_163 | 1.559e-02 | 2.180e-03 | 7.151 | 8.58e-13 | *** |
| var_164 | 2.515e-02 | 2.124e-03 | 11.842 | < 2e-16 | *** |
| var_165 | -3.687e-02 | 2.290e-03 | -16.104 | < 2e-16 | *** |
| var_166 | -4.876e-01 | 3.104e-02 | -15.708 | < 2e-16 | *** |
| var_167 | 1.306e-02 | 1.478e-03 | 8.836 | < 2e-16 | *** |
| var_168 | 1.808e-02 | 3.685e-03 | 4.905 | 9.34e-07 | *** |
| var_169 | -4.508e-01 | 3.132e-02 | -14.392 | < 2e-16 | *** |
| var_170 | 3.926e-02 | 2.590e-03 | 15.161 | < 2e-16 | *** |
| var_171 | 9.901e-03 | 2.145e-03 | 4.617 | 3.90e-06 | *** |
| var_172 | -1.480e-02 | 1.329e-03 | -11.135 | < 2e-16 | *** |
| var_173 | 2.465e-02 | 1.946e-03 | 12.665 | < 2e-16 | *** |
| var_174 | -2.807e-02 | 1.602e-03 | -17.519 | < 2e-16 | *** |
| var_175 | 2.909e-02 | 3.968e-03 | 7.332 | 2.27e-13 | *** |
| var_176 | 3.797e-03 | 1.538e-03 | 2.469 | 0.013554 | * |
| var_177 | -6.051e-02 | 4.417e-03 | -13.699 | < 2e-16 | *** |

```
var_178     -6.951e-03  1.345e-03  -5.169 2.35e-07 ***
var_179      5.928e-02  4.047e-03  14.648  < 2e-16 ***
var_180      2.397e-02  2.187e-03  10.960  < 2e-16 ***
var_181      3.455e-02  8.388e-03   4.119 3.80e-05 ***
var_182     -4.971e-03  1.291e-03  -3.851 0.000117 ***
var_183     -2.719e-03  2.586e-03  -1.052 0.293023    
var_184      1.738e-02  1.233e-03  14.097  < 2e-16 ***
var_185     -2.154e-03  2.450e-03  -0.879 0.379270    
var_186     -3.330e-02  3.639e-03  -9.152  < 2e-16 ***
var_187      4.595e-03  1.004e-03   4.579 4.68e-06 ***
var_188     -2.655e-02  2.927e-03  -9.069  < 2e-16 ***
var_189      2.464e-02  1.185e-02   2.079 0.037572 *  
var_190      4.006e-02  2.527e-03  15.851  < 2e-16 ***
var_191      5.022e-02  3.760e-03  13.358  < 2e-16 ***
var_192     -9.570e-02  7.858e-03 -12.179  < 2e-16 ***
var_193     -1.400e-02  2.896e-03  -4.833 1.35e-06 ***
var_194     -1.989e-02  3.675e-03  -5.411 6.27e-08 ***
var_195      6.656e-02  8.031e-03   8.288  < 2e-16 ***
var_196      1.304e-02  2.115e-03   6.165 7.07e-10 ***
var_197     -1.157e-01  1.250e-02  -9.255  < 2e-16 ***
var_198     -5.503e-02  3.778e-03 -14.566  < 2e-16 ***
 [ reached getOption("max.print") -- omitted 1 row ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 73390  on 112501  degrees of freedom
Residual deviance: 51844  on 112301  degrees of freedom
AIC: 52246

Number of Fisher Scoring iterations: 6



Confusion Matrix and Statistics

   logit_Predictions
        0     1
  0 33204   463
  1  2707  1028
                                          
               Accuracy : 0.9152          
                 95% CI : (0.9124, 0.918) 
    No Information Rate : 0.9601          
    P-Value [Acc > NIR] : 1               
                                          
                  Kappa : 0.3568          
                                          
 Mcnemar's Test P-Value : <2e-16          
                                          
            Sensitivity : 0.9246          
            Specificity : 0.6895          
         Pos Pred Value : 0.9862          
         Neg Pred Value : 0.2752          
             Prevalence : 0.9601          
         Detection Rate : 0.8878          
   Detection Prevalence : 0.9001          
      Balanced Accuracy : 0.8070          
                                          
       'Positive' Class : 1               
```

From the above summary it is clear that var_7, var_10, var_16, var_17, var_27, var_30, var_38, var_39, var_41, var_46, var_60, var_69, var_73, var_96, var_98, var_100, var_107, var_113, var_127, var_129, var_136, var_153, var_160, var_161, var_183 and var_185 are not significant enough to prediction target class as their p > 0.05 and remaining all variable are significantly contributing in the prediction of the target class whose p values are >0.05. var_91 has highest estimate and var_71 follows next.

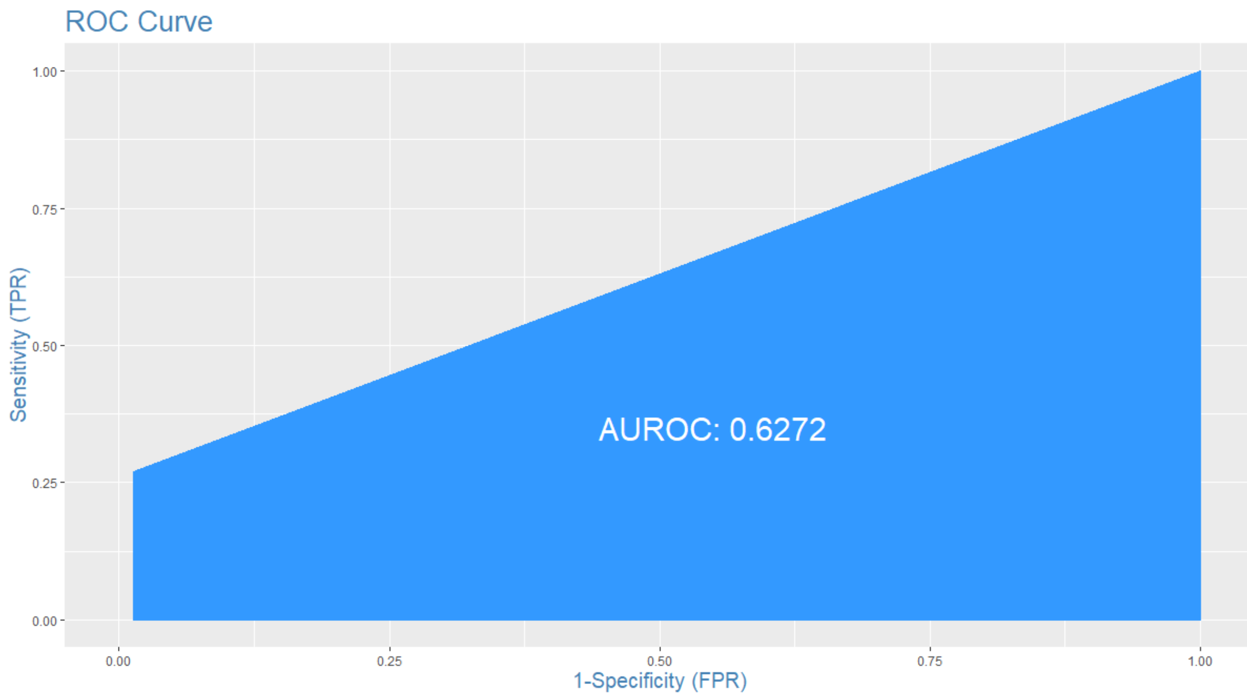


Figure.3. ROC for Logistic regression model performance

### 6.1.2. Decision Trees

In R c5.0 algorithm is trained on train data while in python tree.DecisionTreeClassifier function is used to train the model training data. There total of on an average more than 286 rules have been made for which the var_71 and var_175 are used most for building the decision trees, in proper I can say whose information gain is high. In python entropy criterion is used while in R the number of trails run are 10 which gives the lowest possible false negative rate.

```
#decision tree in python
clf=tree.DecisionTreeClassifier(criterion='entropy').fit(x_train, y_train)
DT_Predictions=clf.predict(x_test)

# Decision tree in R
Model_C50=C5.0(target~., data=train, trails=10, rules=TRUE)
```

```
#predicting the test target values with trained decision tree model
Predictions_c50=predict(Model_C50, test[,-1], type='class')
```

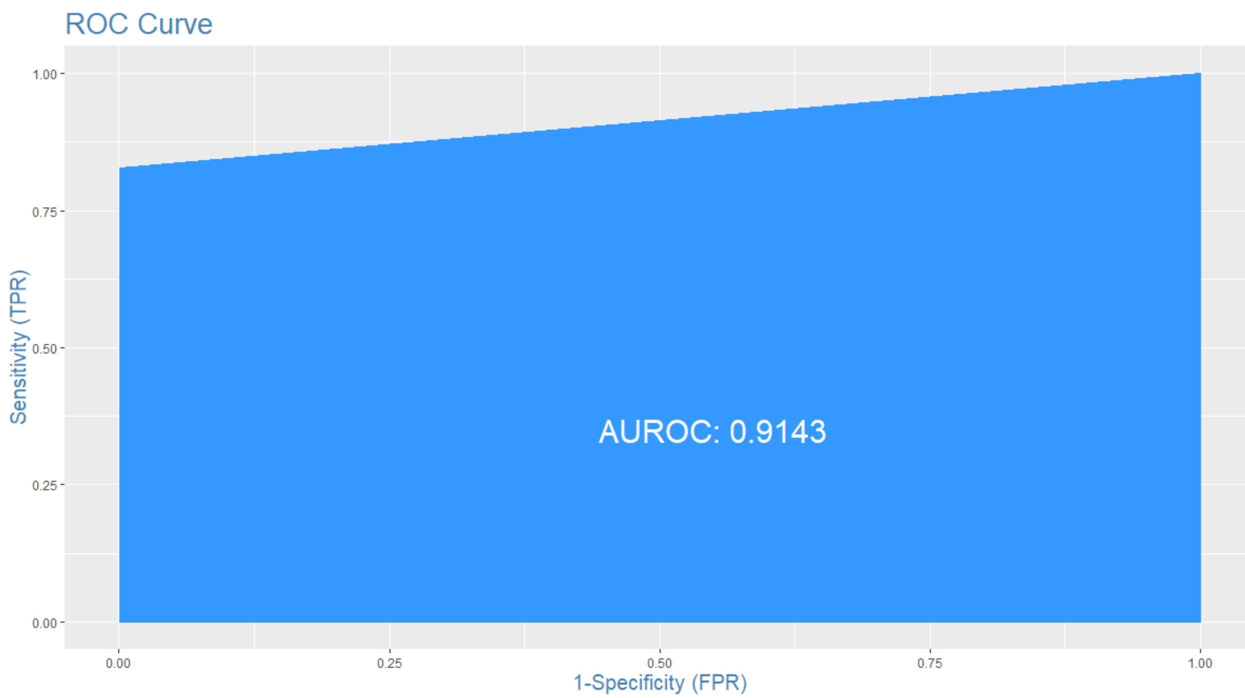


Figure.4. ROC for Decision tree model performance

### 6.1.3. Random Forest Model:

Random forest builds many decision trees and chooses a variable as parental node for each decision tree whose gini index is low or impurity is high. Below are the functions used to build the random forest model on train data both in R and python. The n_estimators are and ntrees in both python and R are same which produces lowest False negative and False positive rates.

In python

```
#Training on train  data
RF_model = RandomForestClassifier(n_estimators = 50).fit(x_train, y_train)
# Predicting test cases with trained model
RF_Predictions = RF_model.predict(x_test)
# Random forest model on traing data in R
Model_RF=randomForest(target~., train, importance=TRUE, ntree=5)
```

And the summary of the random forest models as follows

```
                Length Class  Mode
call                 5 -none- call
type                 1 -none- character
predicted       150001 factor numeric
err.rate           150 -none- numeric
confusion            6 -none- numeric
votes           300002 matrix numeric
oob.times       150001 -none- numeric
classes              2 -none- character
importance         800 -none- numeric
importanceSD       600 -none- numeric
localImportance      0 -none- NULL
proximity            0 -none- NULL
ntree                1 -none- numeric
mtry                 1 -none- numeric
forest              14 -none- list
y               150001 factor numeric
test                 0 -none- NULL
inbag                0 -none- NULL
terms                3 terms  call

3013 rules (length<=6) were extracted from the first 50 trees.

readableRules[1:2,]
[1] "var_0<=4.56045 & var_90<=7.2235 & var_108<=10.8551 & var_157<
=22.1357 & var_171<=12.5564 & var_191<=-0.4548"
[2] "var_0<=4.56045 & var_90<=7.2235 & var_108<=10.8551 & var_157>
22.1357 & var_171<=12.5564 & var_191<=-0.4548"
```

Below two readable rules of which random forest has made, we can see that if var_0 is less than or equal to 4.5 & var_90 less than or equal to 7.2 & var_108 less than or equal to 10.8 & var_157 less than or equal to 22.1 & var_171 less than or equal to 12.5 & var_191 less than equal to -0.45 then the target class will belong to 1 and Same way each rules will be made.

```
[1,] "X[,2]<=4.56045 & X[,92]<=7.2235 & X[,110]<=10.8551 & X[,159]<=22.1
357 & X[,173]<=12.5564 & X[,193]<=-0.4548"
[2,] "X[,2]<=4.56045 & X[,92]<=7.2235 & X[,110]<=10.8551 & X[,159]>22.13
57 & X[,173]<=12.5564 & X[,193]<=-0.4548"
     pred
[1,] "1"
[2,] "0"
```

Due to the best performance of the random forest model on training data I considered to go for building random forest on both random over sampled and random under sampled train data. In both the cases n_estimators or ntrees built were kept 50. In train data there are total of 20098 observation are belong to target class 1 whereas 179902 observations are belonging to target class 0. Target class 0 observations were brought down to the number of target class 1 observations in random under sampling technique while in random over sampling target class 1 observations were increased to target class 0 observations. Synthetic minority oversampling technique was also used in python that does not results increase in performance of the model instead it resulted very lower performance.

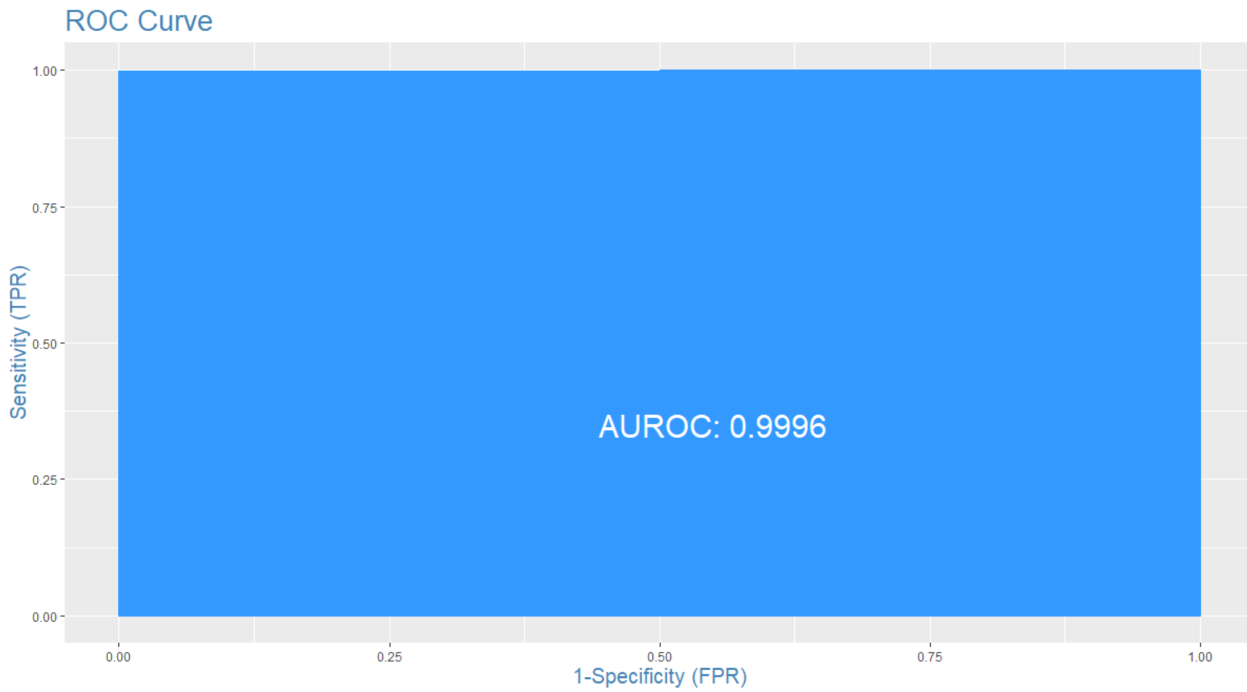


Figure.5. ROC for random Forest model performance

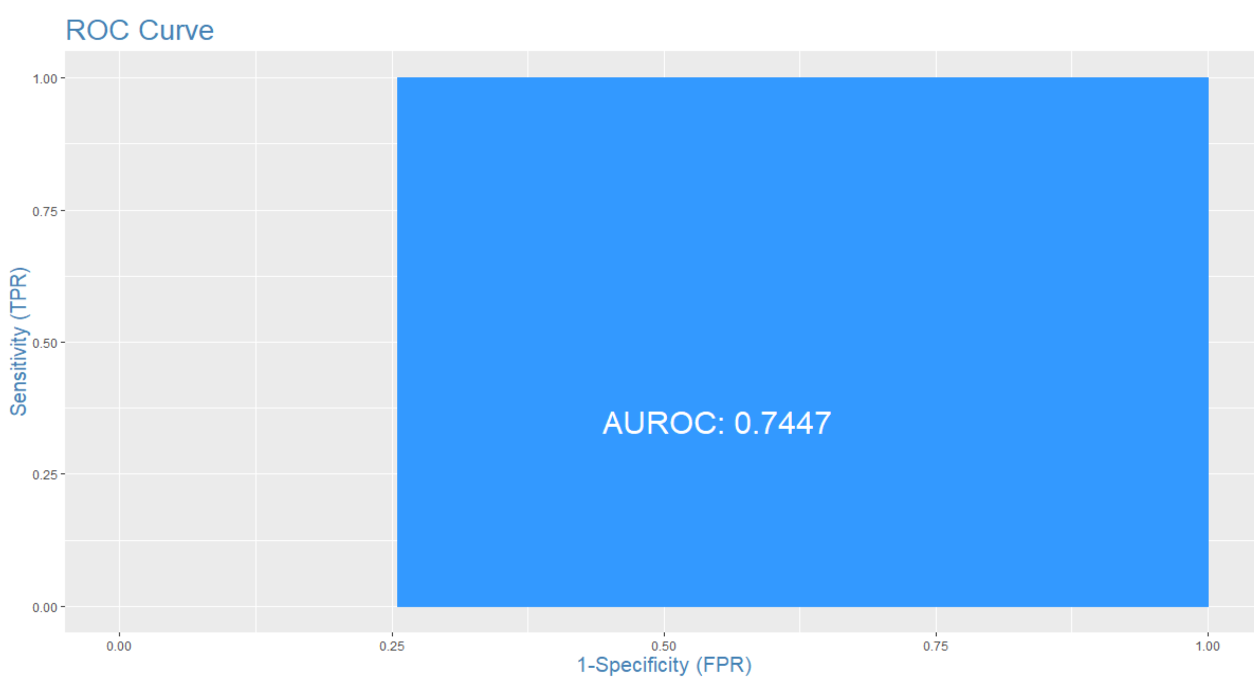


Figure.6. ROC of Random forest model developed over random under sampling method

It can be observed that in figure.6. the ROC curve shows that AUROC value 0.7 which is lower compared to the model performance in the data which randomly over sampled. And we can also observe that the performance of the model which was built on the original train data is almost equal to the performance of the model built on the data which was over sampled (figure.7).

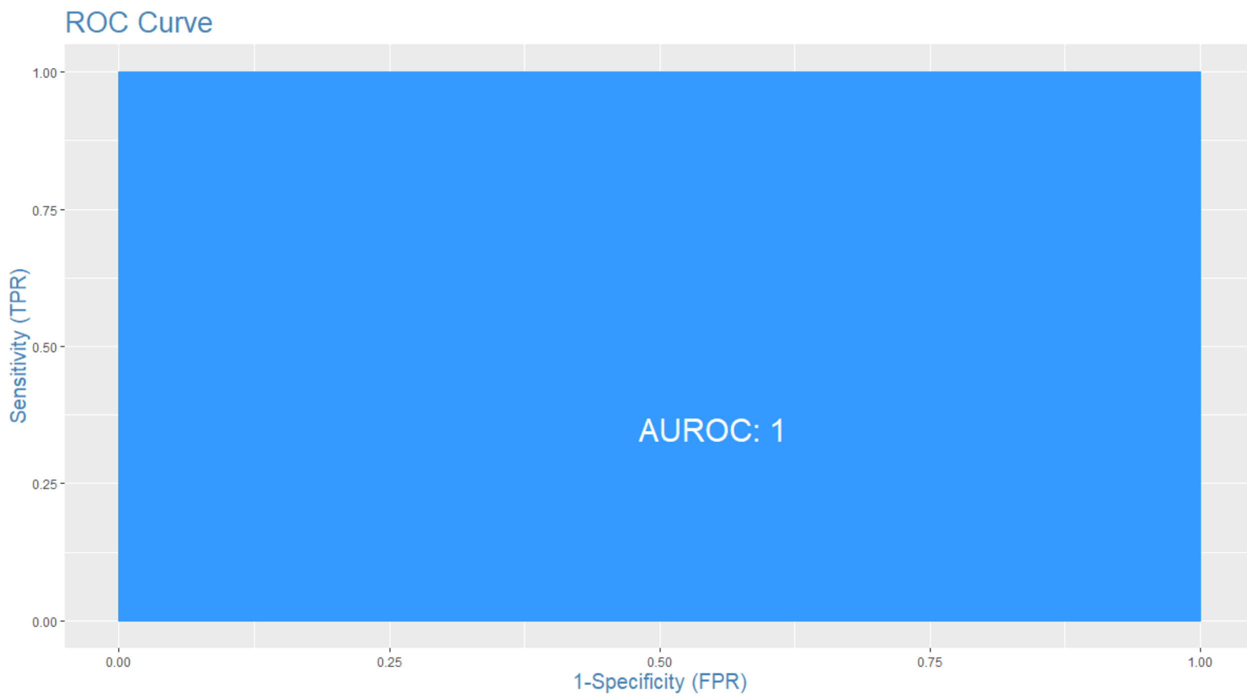


Figure.7. ROC of Random forest model developed over random up sampling method

## 6.1.4. Naïve Bayes Algorithm:

Naïve Bayes use each variable independently from other variables it means that even if we supply the variables which are having multicollinearity there will not be any bias in the model. It calculates probabilities of each independent variables and combines all to predict the class of target variable. Since the data provided for Santander customer transaction prediction are all numerical data except the target variable, it calculates the probability each variable for particular class and assigns a target class for test case whose probability is higher.

In python below code is used to build the model

#Naive Bayes implementation in python on train data

NB_model = GaussianNB().fit(x_train, y_train)

While in R following code is used

Model_NB=naiveBayes(target~., data=train)

```
summary(Model_NB)
           Length Class  Mode
apriori     2     table  numeric
tables    200     -none- list
levels      2     -none- character
isnumeric 200     -none- logical
call        4     -none- call
```

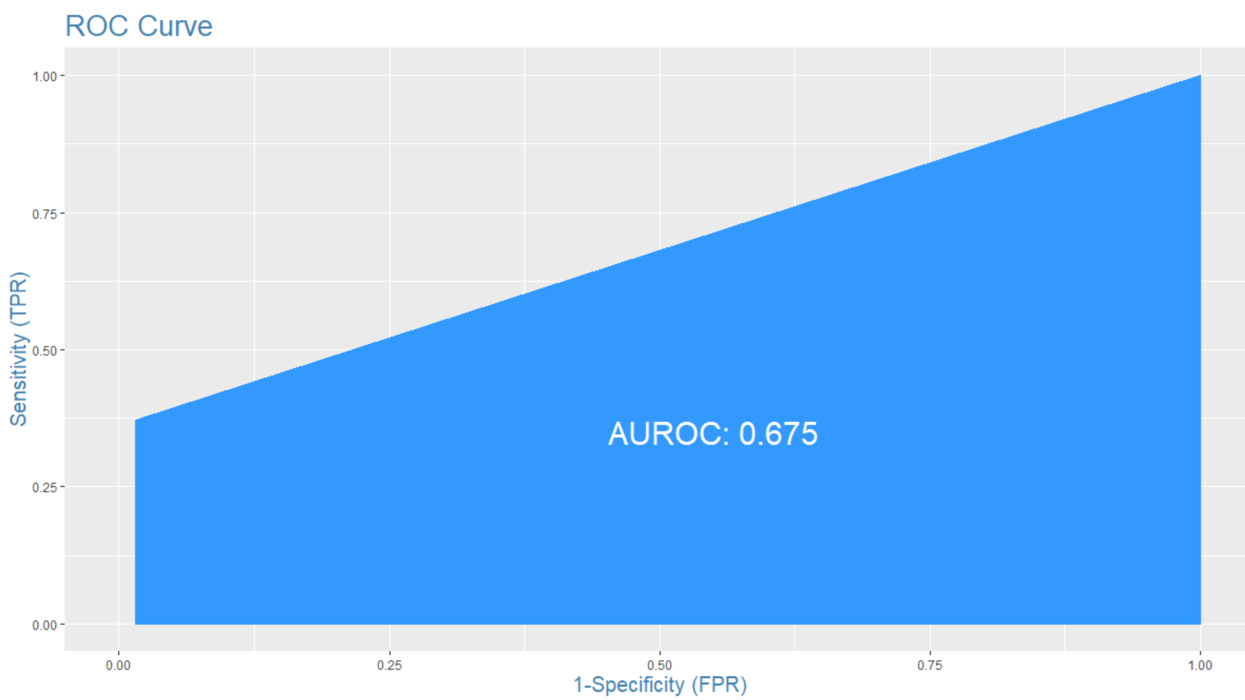


Figure.8. ROC curve for Naïve Bayes model performance

### 6.1.5. Knn Classifier

K nearest neighbour classification model stores all available cases and classifies new cases based on similarity measure. It calculates the distance between the test and train data assigns the target class whose distance score is vert less. If the input data is classification problem it passes majority and minority rule. Each time whenever we use knn algorithm we need to specify the K value, suppose if the k value is 3 then takes 3 nearest values of train data which are similar to test cases and calculates the distance assign the test cases class whose distance score is very less. Below mentioned code was used for building the classification model on train data.

```
###########KNN Classification model##########
KNN_Predictions=knn(train[,2:201], test[,2:201], train$target, k=3)
```

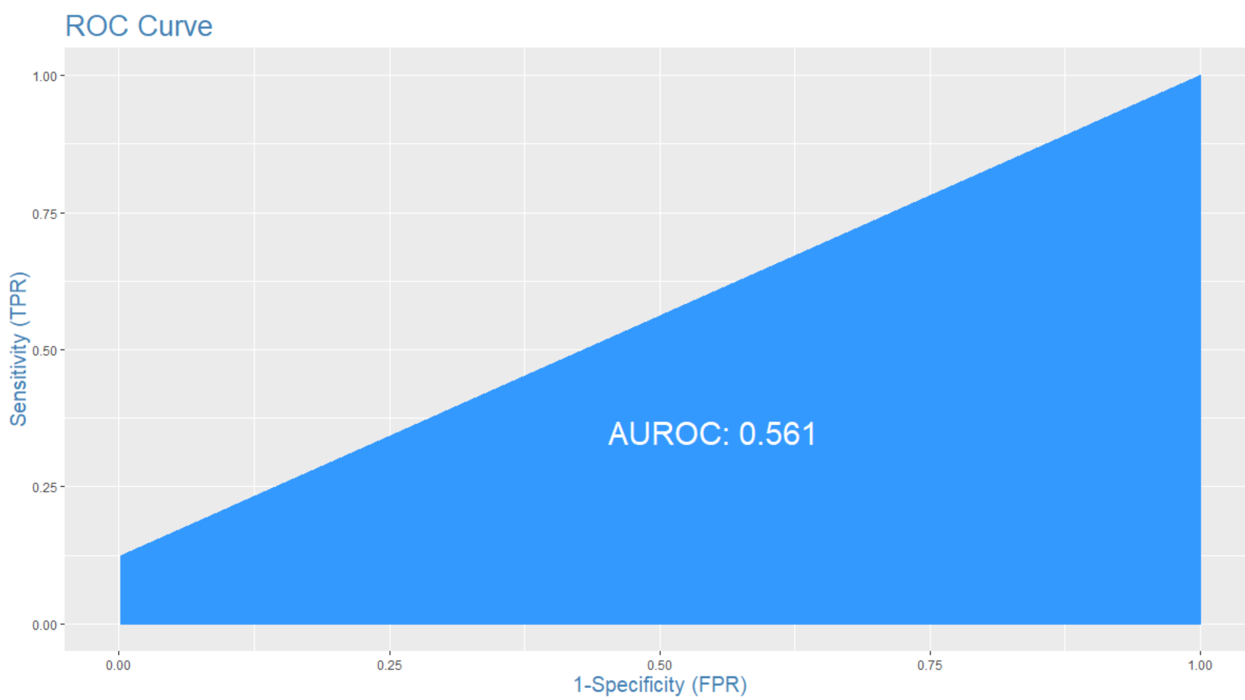


Figure.8. ROC curve for KNN classifier performance

## 6.2. Model evaluation

Now that some of classification models have been built on training data, there is need to choose which model to be selected to predict our test cases. For any model to be selected over other, there are different approaches are followed to compare the models. In which the power of predictiveness, interpretability and computational efficiency. Power of predictability and computational efficiency will be more suitable for classification problems for which accuracy, AUROC score and F1 score are best error metrics. The trained models tested for their performance on the test data which has been extracted from the train data. The trained models allowed to predict on test data and compared the actual values and predicted values. Different error metrics were chosen based on predictive performance of the model and metrics were tabulated below. Based on error metrics we can see that in table.1

Knn classifier has performed least while Random forest performed very well with maximum AUROC and F1 scores followed by Decision trees, Naïve Bayes and Logistic regression models.

| Error metrics | Logistic Regressor | Random Forest | Decision Tree | Naïve Bayes | KNN Model | Random Forest under sampling | Random Forest Up sampling |
|---|---|---|---|---|---|---|---|
| **Accuracy** | 0.9146293 | 0.999627 | 0.982574 | 0.92151 | 0.9095 | 0.770679 | 1 |
| **Error rate** | 0.0853707 | 0.000372 | 0.017425 | 0.07848 | 0.0904 | 0.229320 | 0 |
| **FNR** | 0.7207349 | 0.003674 | 0.171391 | 0.62414 | 0.8758 | 0 | 0 |
| **FPR** | 0.0137067 | 0 | 0.00006 | 0.01693 | 0.0018 | 0.255295 | 0 |
| **Precision/ PPV** | 0.6967911 | 1 | 0.999366 9 | 0.71457 0 | 0.8841 12 | 0.307326 5 | 1 |
| **Recall/ TPR / Sensitivity** | 0.2792651 | 0.996325 | 0.828608 | 0.37585 | 0.1241 47 | 1 | 1 |
| **Specificity /TNR** | 0.9862933 | 0.999585 | 0.999940 | 0.98306 | 0.9981 64 | 0.744704 9 | 1 |
| **AUROC** | 0.6272418 | 0.999641 | 0.914250 | 0.675 | 0.5610 | 0.744704 | 1 |
| **F1 score** | 0.3987259 | 0.998159 | 0.906012 | 0.49260 | 0.2177 | 0.470160 | 1 |

Table.1. Error metrics table for different models.

## 6.3. Model Selection

Due to the outperformance of the random forest model on random oversampled data with a maximum AUROC and F1 scores during model evaluation, it has been chosen for the prediction of test cases. The predicted test cases were saved into disc in csv format and the trained model saved for future predictions in the format of pkl format in python and RDS in case of R.

# 7. Summary

- Despite being taking longer time for training, Random forest outperforming over the other all other models
- Knn being the least performed model and taking most of the time for training and prediction.
- Random forest model built on the data which has not been sampled was performed as same as the random forest model trained on random oversampled train data.

- The number of test cases which are predicted target class as 1 are same with both the random forest models which are developed on un sampled and oversampled data.
- The Random forest model built on under sampled trained data has performed least compared to model built on data which has not been sampled.
- Decision tree model has performed as same as random forest model and well performed compared to logistic regression and Naïve Bayes model.
- Logistic regression model performance was reduced due to the outliers present in data.

# 8. Visuals

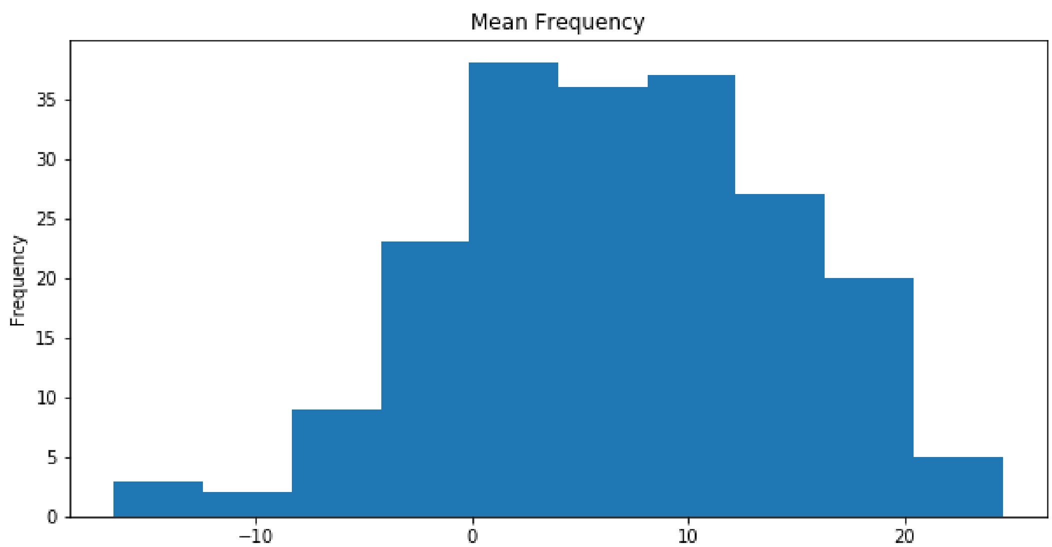


Figure.9. Distribution of mean frequency

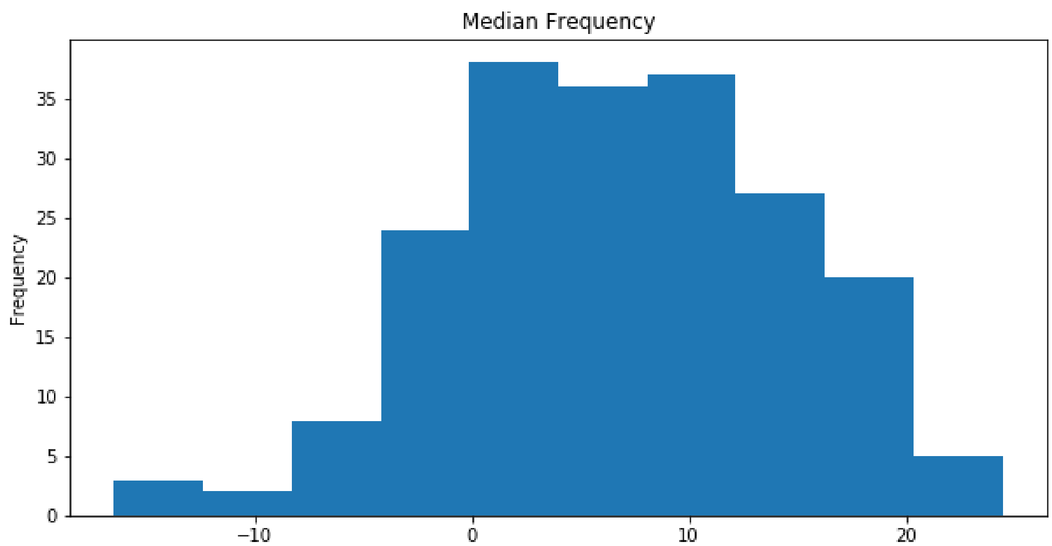


Figure.10. Distribution of median frequency

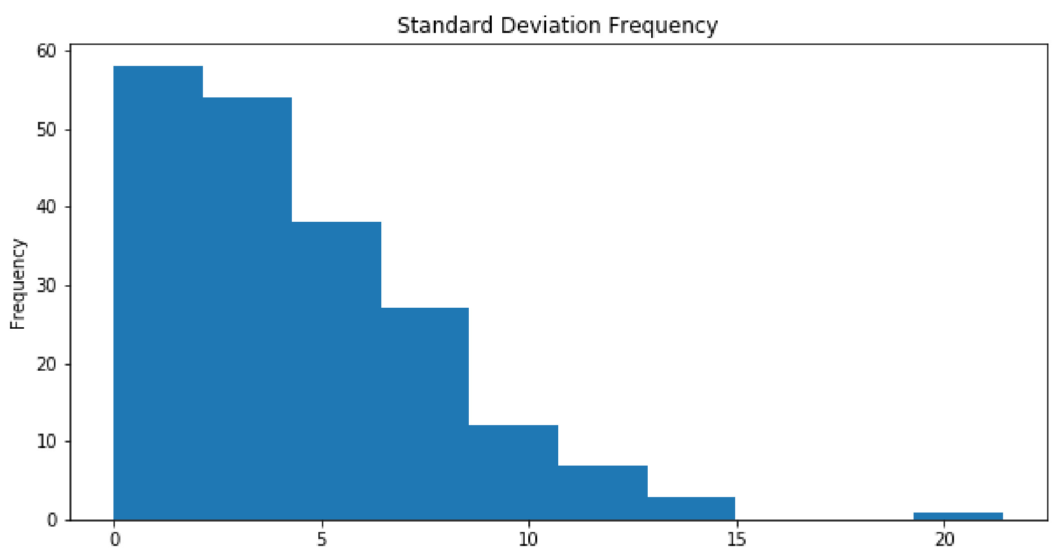


Figure.11. Distribution of standard deviation frequencies

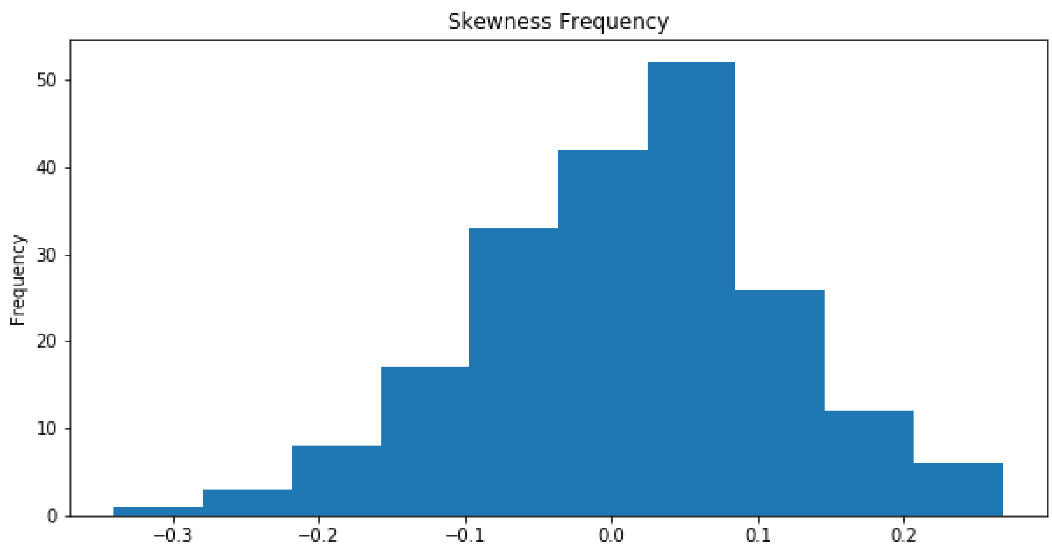


Figure.12. Distribution of Skewness frequency

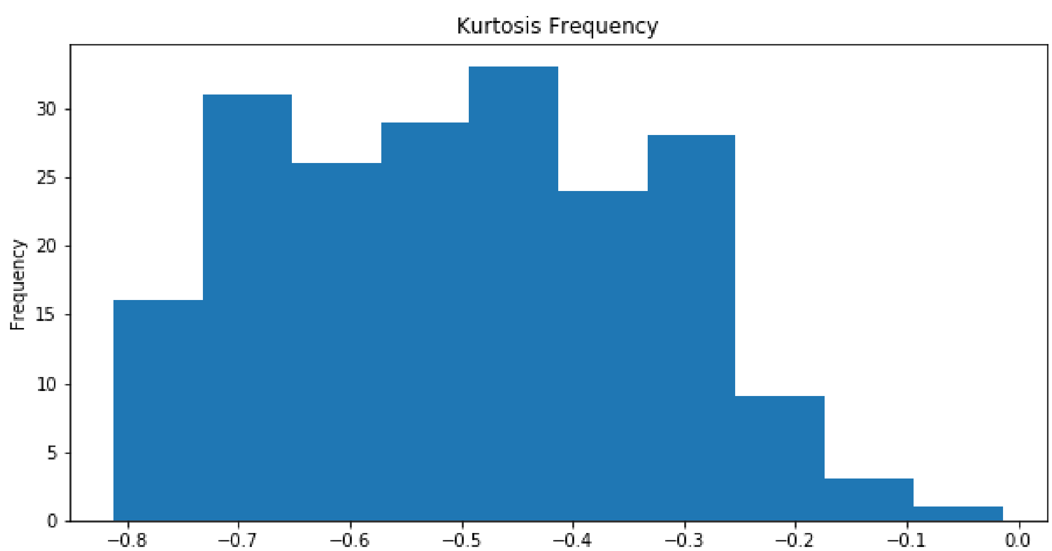


Figure.13. Distribution of Kurtosis frequency