Santander Customer Transaction Prediction Report

(edWisor.com learning project)

**Presented by- KAPIL NAHAR (November-2019)**

# Contents

###### Introduction

* 1. Background……………………………………………………………………………………………………….
  2. Problem Statement…………………………………………………………………………………………...

1.3 Data…………………………………………………………………………………………………………………..

###### Methodology

* 1. Pre Processing………………………………………………………………………………………..
     1. Missing Value Analysis……………………………………………………………………………
     2. Outlier Analysis……………………………………………………………………………………….
     3. Feature Importance…………………………………………………………………………………
     4. Feature Selection…………………………………………………………………………………….
     5. Handle imbalanced data…………………………………………………………………………..

###### Modeling

* + 1. Model Selection…………………………………………………………………………………………..
    2. Logistic Regression………………………………………………………………………………………..
    3. Decision Tree………………………………………………………………………………………………..

3.1.4 Random Forest………………………………………………………………………………………………

3.1.5 LightGBM………………………………………………………………………………………………………..

1. **Conclusion**
   1. Model Evaluation………………………………………………………………………………………………….
      1. AUC Score……………………………………………………………………………………………………………..
      2. Precision………………………………………………………………………………………………………………..
      3. Recall……………………………………………………………………………………………………………………….

**Model Selection**

### 5. Visualizations

* 1. Histogram of var\_81…………………………………………………………………………………..
  2. Histogram of var\_53……………………………………………………………………………………
  3. Histogram of var\_20…………………………………………………………………………………….
  4. Histogram of var\_14……………………………………………………………………………………

**6. Appendix A- R Code**

Importing Data…………………………………………………………………………………………………………

Missing value Analysis……………………………………………………………………………………………..

Important Features ………………………………………………………………………………………………….

Correlation between variables………………………………………………………………………………….

Logistic Regression……………………………………………………………………………………………………

Using ROSE Package in R and through SMOTE to balance the data……………………………

Random Forest………………………………………………………………………………………………………….

Using LightGBM…………………………………………………………………………………………………………

**7. Appendix B- Python Code**

Importing Data……………………………………………………………………………………………………………….

Missing value Analysis…………………………………………………………………………………………………….

Important Features (Extra Trees Classifier) ……………………………………………………………………..

Correlation between variables………………………………………………………………………………………..

Logistic Regression………………………………………………………………………………………………………….

Using SMOTE (Handle target class imbalance problem) ……………………………………………………

Decision Tree…………………………………………………………………………………………………………………….

Using LightGBM………………………………………………………………………………………………………………….

**1. INTRODUCTION**

**1.1 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?

**1.2 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.

## 1.3 Data

**Table 1.1: Sample Data (Columns: 1-7)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID\_code** | **target** | **var\_0** | **var\_1** | **var\_2** | **var\_3** | **var\_4** |
| train\_0 | 0 | 8.9255 | -6.7863 | 11.9081 | 5.093 | 11.4607 |
| train\_1 | 0 | 11.5006 | -4.1473 | 13.8588 | 5.389 | 12.3622 |
| train\_2 | 0 | 8.6093 | -2.7457 | 12.0805 | 7.8928 | 10.5825 |
| train\_3 | 0 | 11.0604 | -2.1518 | 8.9522 | 7.1957 | 12.5846 |
| train\_4 | 0 | 9.8369 | -1.4834 | 12.8746 | 6.6375 | 12.2772 |

**Table 1.2: Sample Data (Columns: 196-202)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **var\_193** | **var\_194** | **var\_195** | **var\_196** | **var\_197** | **var\_198** | **var\_199** |
| 1.691 | 18.5227 | -2.3978 | 7.8784 | 8.5635 | 12.7803 | -1.0914 |
| 10.9516 | 15.4305 | 2.0339 | 8.1267 | 8.7889 | 18.356 | 1.9518 |
| 1.6858 | 21.6042 | 3.1417 | -6.5213 | 8.2675 | 14.7222 | 0.3965 |
| 1.4214 | 23.0347 | -1.2706 | -2.9275 | 10.2922 | 17.9697 | -8.9996 |
| 9.1942 | 13.2876 | -1.5121 | 3.9267 | 9.5031 | 17.9974 | -8.8104 |



Look at the shape of the data. There are **202 variables** in the **train dataset** and **200000 observations.**

The details of dataset attributes are as follows -

* + - ID\_code – ID of the data starting from train\_0 to train\_199999.
    - target- Object variable indicating if a customer will make a transaction if the target value is zero then customer will not make transaction and if it is one then customer will make a transaction.
    - var\_0 to var\_199 – All these are float variable and Independent. These features will be used to predict the target variable of the test dataset.

Now let’s have a look at the data type of train attributes.

ID\_code object

target int64

var\_0 float64

var\_1 float64

var\_2 float64

...

var\_195 float64

var\_196 float64

var\_197 float64

var\_198 float64

var\_199 float64 Length: 202, dtype: object

Here, the datatype of target attribute is int which is actually object. Datatype of ID\_code is object. Similarly, datatype of variables from **var\_0** to **var\_199** is float.

# Methodology

## Pre Processing

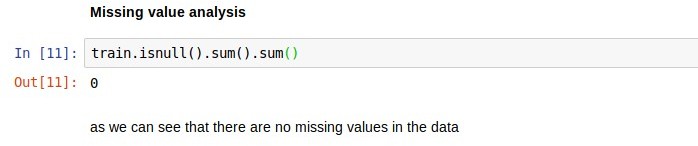
Before developing any model, we first need to look into the data. By, saying look into the data, I mean to explore the data. Look at the datatypes of attributes, find the minimum and maximum values of variables compare it with its mean value. Convert the required datatypes. Finding and imputing missing values using various methods like mean, median, etc. This is nothing but data preprocessing where we analyze the data, clean the data and transform the data. Data preprocessing is the probably most or one of the most important things in model development. So, it needs to be taken care of.

###### 2.1.1 Missing Value Analysis

Missing value analysis is a method or technique to find out if there are missing values in the attributes of the dataset. When we applied missing value analysis on our dataset we found out that there were no missing values in our dataset.

* + - * is.null( ).sum() in python
      * function(x){sum(is.na(x))} in R

We got the following result after applying missing value analysis on our dataset



Now, as we can see that there were no missing values in the dataset.

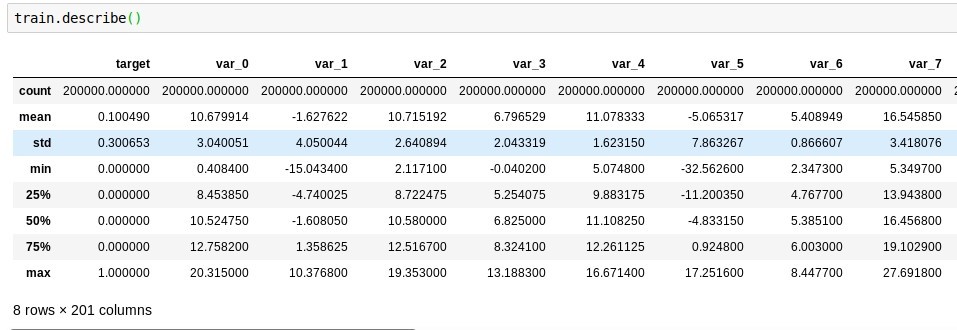
* + 1. **Outlier Analysis**

In statistics, an outlier is defined as a data point that differs significantly from other observations. Outlier analysis is a technique to find these points. Outlier analysis can only be done on a numerical variable.

Causes of Outliers

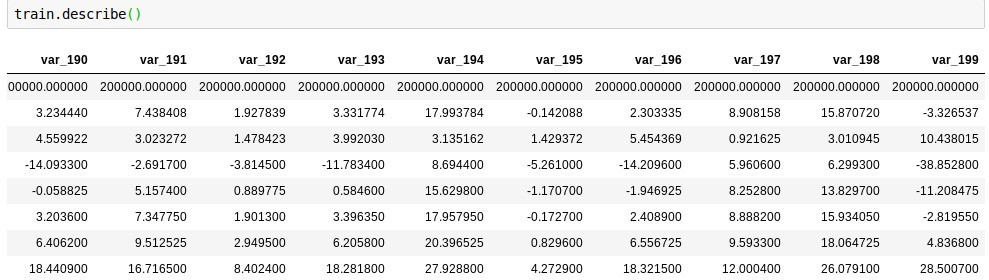
* Poor data quality/contamination
* Low-quality measurements, malfunctioning equipment, manual error
* Correct but exceptional data

Let’s look at the summary of the data and try to find out if the outlier exists in our dataset.



This is a summary of the first 9 variables. Look at the summary of var\_0, you can see that it has mean value as 3.04 and max value is 20.31. That means max value is not differing with mean value by very large value.

So, by these details we can say that there are no outliers in var\_0 variable. Similarly, you can look at the other variables and find out that there are no outliers.



Similarly, you can see last 10 variables also do not contain outliers. For example, look at the var\_199. It has mean value as 10.438015 and the max value is 28.500 that means they are not differing by large margin hence it does not contain outliers.

### Feature Importance

Feature Importance is used to find out what features are contributing the most in order to predict the result. To find out what features are most important we used “**ExtraTreesClassifier”.**

We used the below code to find out important features.

X = train.iloc[:,2:202] #independent columns

y = train.iloc[:,1] #target column i.e price range model = ExtraTreesClassifier(random\_state=42) model.fit(X,y) print(model.feature\_importances\_)

#plot graph of feature importances for better visualization plt.subplots(figsize=(40,45))

feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns) feat\_importances.nlargest(200).plot(kind='barh')

plt.show()

Here, X is the variable that contains all the independent variables and y is also a variable that contains dependent variable i.e target.

Let’s look at the top 20 features that we got from the above code.



You can see that var\_81, var\_53, var\_26, and var\_139 are some of the most important features.

## Feature Selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

In machine learning and statistics feature selection is also known as variable selection, attribute selection or variable subset selection.

In our dataset ID\_code which is an object, this variable is of no use hence we dropped this column from our train and test dataset. And selected the rest of the features.

## 2.1.5 Handle Imbalanced data

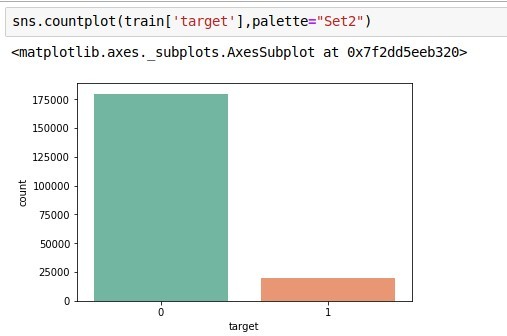
If you will look at the count of the target variable, then you will find out that number of zero’s is much more than the number of ones. This is nothing but target class imbalance problem. Have a look at the count plot of the target variable.

Count

0 179902

1 20098

Name: target, dtype: int64



As you can see that there are 20098 occurrences of positive class i.e 1 and 179902 occurrences of negative class i.e 0. This clearly shows imbalance of data. The problem with imbalanced data is that model might be biased towards majority class which is zero in this case.

So, to handle imbalanced data there are several methods like under-sampling, over-sampling and synthetic data generation.

## Under-sampling: -

This method works with the majority class. It reduces observations from majority class in order to make dataset balanced.

But the problem with this model is that while reducing the observations from majority class we may lose the important data. Which is also called as under-fitting.

## Over-sampling: -

This method works with minority class. It replicates the observation of minority class to make balance between minority and majority class. But the problem with this method is that as we are replicating the data there will be many similar observations and will cause overfitting.

## 

## Synthetic Data Generation: -

In simple words instead of replicating the same data, this method creates new artificial data.

SMOTE (synthetic minority oversampling technique) is a powerful and widely used method to create synthetic data.

So, we have used SMOTE to handle imbalance data. Library of SMOTE

* + in R- ROSE
  + in python-SMOTE

sm = SMOTE(random\_state=42) X\_train\_smote,y\_train\_smote=sm.fit\_sample(X\_train,y\_train) X\_test\_smote,y\_test\_smote=sm.fit\_sample(X\_test,y\_test) print(X\_train\_smote.shape)

print(X\_test\_smote.shape)

## 3. Modeling

### 3.1.1 Model Selection

As we know that we need to predict whether a customer will make a transaction or not we can understand that this is a classification problem. So, we knew that we have to use classification models to predict the target variable. Hence we used the following regression models to predict the result.

1. In python

* Logistic Regression
* Decision Tree
* LightGBM

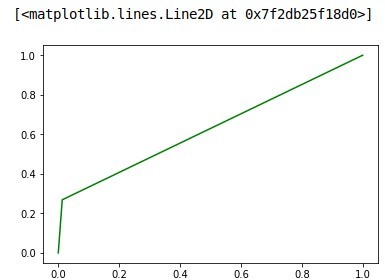
1. In R

* Logistic Regression
* Random Forest
* LightGBM

### 3.1.2 Logistic Regression on imbalanced data

#Logistic Regression logit=LogisticRegression(random\_state=42) logit\_model=logit.fit(X\_train,y\_train) logit\_pred=logit\_model.predict(X\_test) FPR,TPR,thresholds=roc\_curve(y\_test,logit\_pred) plt.plot(FPR,TPR,'g')

Roc Curve



Classification Report

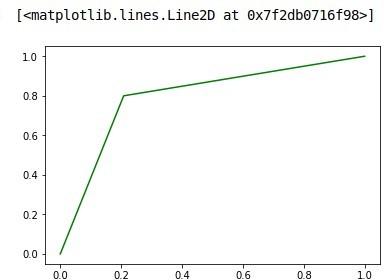
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  |  |
|  |  | **precision** | **recall** | **f1-score** | **support** |
|  |  |  |  |  |  |  |
|  | **0** | **0.92** | **0.99** | **0.95** | **35980** |  |
|  | **1** | **0.70** | **0.27** | **0.39** | **4020** |  |
|  |  |  |  |  |  |  |
|  | **accuracy** |  |  | **0.91** | **40**  **000** |  |
|  | **macro avg** | **0.81** | **0.63** | **0.67** | **40000** |  |
|  | **weighted avg** | **0.90** | **0.91** | **0.90** | **40000** |  |
|  |  |  |  |  |  |  |

Look at the ROC curve and classification report. Even though the model is giving accuracy of 91% but has very low recall rate for positive class. That means number of times positive class was predicted right is very less.

### Logistic Regression on synthetic data

smote=LogisticRegression(random\_state=42) smote\_model=smote.fit(X\_train\_smote,y\_train\_smote) smote\_pred=smote\_model.predict(X\_test\_smote) FPR,TPR,thresholds=roc\_curve(y\_test\_smote,smote\_pred) plt.plot(FPR,TPR,'g')

ROC curve



Classification Report

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | | | |
| **precision** | **recall** | **f1-score** | **support** |
|  |  |  |  |  |  |
|  | **0** | **0.80** | **0.79** | **0.79** | **35980** |
|  | **1** | **0.79** | **0.80** | **0.80** | **35980** |
|  |  |  |  |  |  |
|  | **accuracy** |  |  | **0.80** | **71960** |
|  | **macro avg** | **0.80** | **0.80** | **0.80** | **71960** |
|  | **weighted avg** | **0.80** | **0.80** | **0.80** | **71960** |
|  |  |  |  |  |  |

Now, look at the ROC curve and classification report. You can see that roc curve has improved a lot and recall value has also improved that means Logistic Regression is performing well on synthetic data.

### 3.1.3 Decision Tree

We have divided train data into 80% train and 20% test datasets for the decision tree model. Let’s look at the decision tree model development code in python.

C50\_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X\_train, y\_train) #predict new test cases

C50\_Predictions = C50\_model.predict(X\_test)

Here, X\_train is subset data from the train dataset for training and has all independent variables. Similarly, y\_train is a training dataset with only the target variable.

X\_test is test data that is a subset of the train dataset and has all the independent variables.

### 3.1.4 Random Forest

For Random Forest also we have divided train data into 80% train and 20% test datasets. Then we used ROSE in R to create new synthetic data as train\_rose and test\_rose. Let’s look at the random forest model development code in R.

RF\_model = randomForest(target ~ ., train\_rose, importance = TRUE, ntree = 100,seed=2) RF\_Predictions = predict(RF\_model, test\_rose[,-1])

Here, train\_rose is the new data that we created by using ROSE for training. Similarly, test\_rose new test dataset is a training dataset.

Now let’s look at how this model performed in R

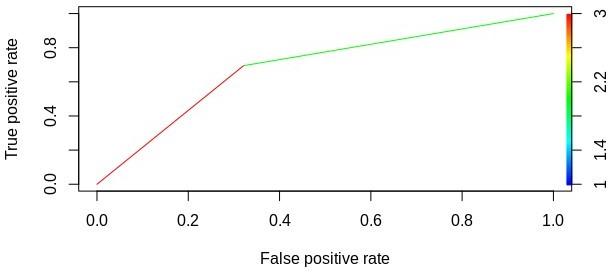
Examples are labelled as positive when predicted is greater than 0.5

precision: 0.507

recall: 1.000

F: 0.337

Roc curve:



So, if you will look at the roc curve and precision, F-1 score you can say that random forest is not performing very good.

### 3.1.5 LightGBM

In the lightGBM model, first of all, we need to set parameters in order to tune our model.

p aram = {'objective' : "binary", 'boost':"gbdt",

'metric':"auc",

'boost\_from\_average':"false",

'num\_threads':8,

'learning\_rate' : 0.01,

'num\_leaves' : 13,

'max\_depth':-1,

'tree\_learner' : "serial",

'feature\_fraction' : 0.05,

'bagging\_freq' : 5,

'bagging\_fraction' : 0.4,

'min\_data\_in\_leaf' : 80,

'min\_sum\_hessian\_in\_leaf' : 10.0,

'verbosity' : 1}

So, these are the parameters we used to tune our lightGBM model. Let’s look at the lightGBM code.

num\_rounds=20000 lgbm=

lgb.train(param,lgb\_train,num\_rounds,valid\_sets=[lgb\_train,lgb\_test],verbose\_eval=400,early\_stopping\_rounds = 3000)

lgbm

Here, num\_rounds are nothing but the number of iterations we want lightGBM to perform in order to get optimal results. lgb\_train is train dataset and lgb\_test is test dataset.

verbose\_eval will basically print the AUC score of train and test dataset for every 400th iteration. early\_stopping round will stop lightGBM model if it doesn’t see any improvement in AUC score of the test dataset consistently for 3000 rounds.

# 4. Conclusion

##### 4.1 Model Evaluation

Now that we have three models for predicting whether a customer will make a transaction or not, we need to decide which one to choose. There are several criteria that are used for evaluating and comparing models. We can compare our models using the following criteria:

1. AUC score
2. Precision
3. Recall

###### AUC score

AUC stands for the area under roc curve. That is AUC measures entire area under the roc curve. The value of AUC ranges from 0 to 1. A model whose predictions are 100 percent correct has an AUC score of 1 and model whose predictions are 100 percent wrong has AUC score of 0.

###### Precision

Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actually positive. Precision is a good measure to determine when the costs of False Positive is high. As in our case if we predicted that certain customer will make a transaction but in reality he is not going to make any transaction and we will lose all our expenditure on that customer. Precision is given by.

Precision= True positive/(True positive+False Positive)

= True positive/Total predicted positive

* + 1. **Recall**

Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

For example, if we predict that a customer will not make any transaction and if he may make a transaction then we might lose that customer. The recall is given by

#### Recall= True positive/(True positive+False Negative)

= True positive/Total actual positive

Performance Measures of the models.

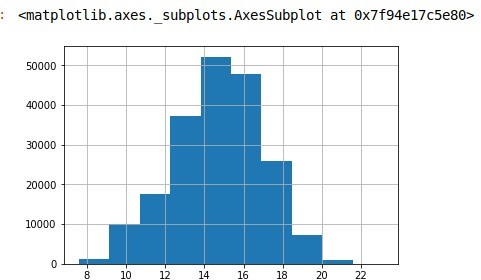
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Auc Score** | **Report** | | | |
|  |  |  | **precision** | **recall** | **f1-score** |
|  |  | **0** | **0.80** | **0.79** | **0.79** |
|  |  | **1** | **0.79** | **0.80** | **0.80** |
| **Logistic Regression** | **79.20** | **accuracy** |  |  | **0.80** |
|  |  | **macro avg** | **0.80** | **0.80** | **0.80** |
|  |  | **weighted avg** | **0.80** | **0.80** | **0.80** |
|  |  |  | **precision** | **recall** | **f1-score** |
|  |  | **0** | **0.62** | **0.75** | **0.68** |
|  |  | **1** | **0.68** | **0.53** | **0.60** |
| **Decision Tree** | **64.58** | **accuracy** |  |  | **0.64** |
|  |  | **macro avg** | **0.65** | **0.64** | **0.64** |
|  |  | **weighted avg** | **0.65** | **0.64** | **0.64** |
|  |  |  | **precision** | **recall** | **f1-score** |
|  |  | **0** | **0.86** | **0.93** | **0.89** |
|  |  | **1** | **0.92** | **0.85** | **0.89** |
| **LightGBM** | **89** | **accuracy** |  |  | **0.89** |
|  |  | **macro avg** | **0.89** | **0.89** | **0.89** |
|  |  | **weighted avg** | **0.89** | **0.89** | **0.89** |

##### 4.2 Model Selection

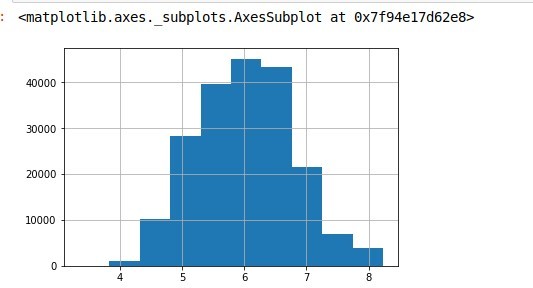
As we can see from the above table that LightGBM model is giving us the best AUC score.

Similarly, LightGBM is also giving best Precision and Recall score. So based on these result we can see that LightGBM outperforms other models and we selected this model for predicting target variables.

1. **Visualizations**
   1. **Histogram of the var\_81**

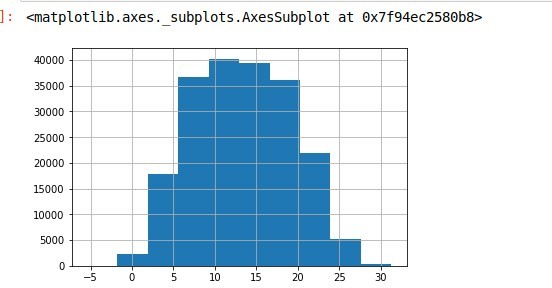


* 1. **Histogram of the var\_53**

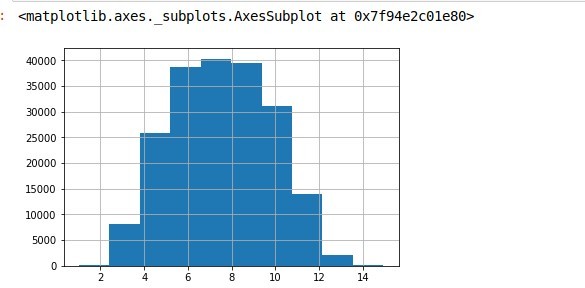


As you can see that histogram of var\_81 and var\_53 is similar which are the most important variables. This indicates that distribution of data of important variables is similar.

### Histogram of the var\_20



* 1. **Histogram of the var\_14**



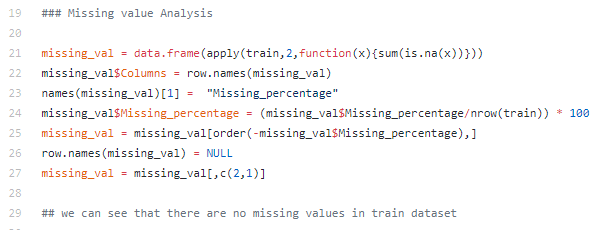
Similarly look at the histogram of var\_20 and var\_14 which are least important variables. You can see that the distribution of data of these variables is similar

1. **Appendix A- R Code**

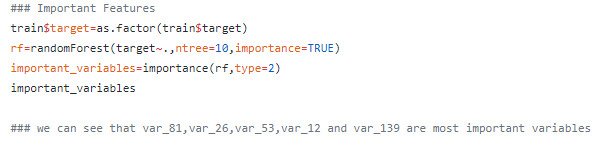
Importing Data



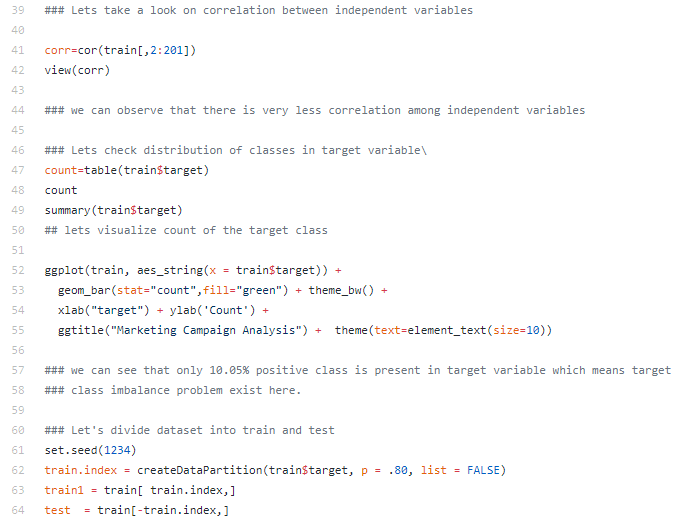
Missing Value Analysis



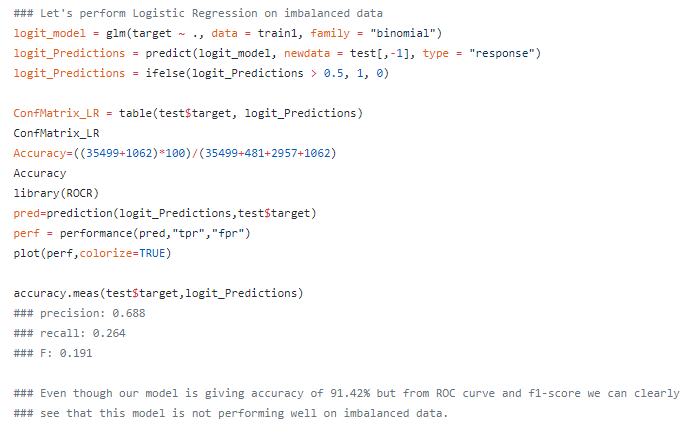
Important Features



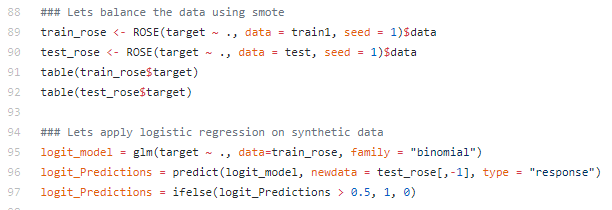
Correlation between variables

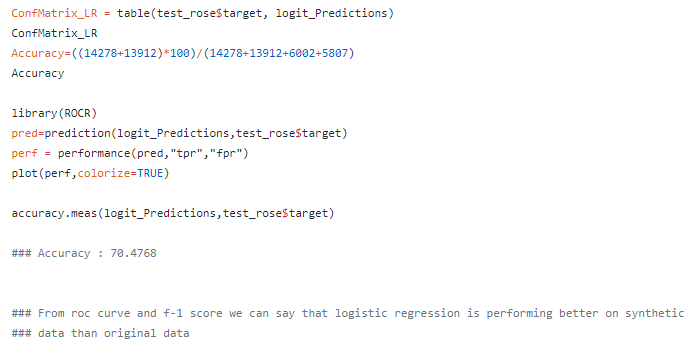


Logistic Regression

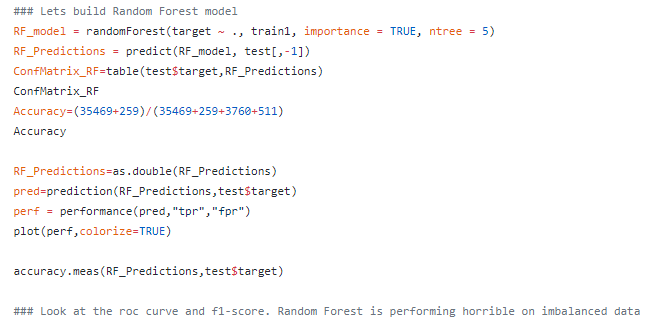


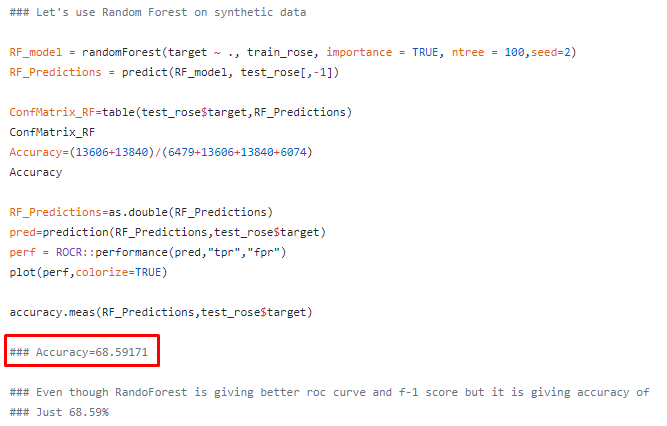
Using ROSE Package in R and through SMOTE to balance the data



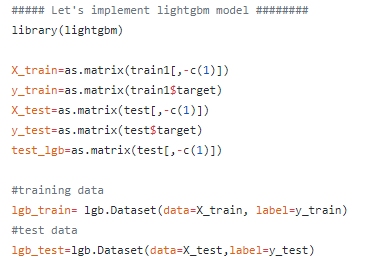


Random Forest

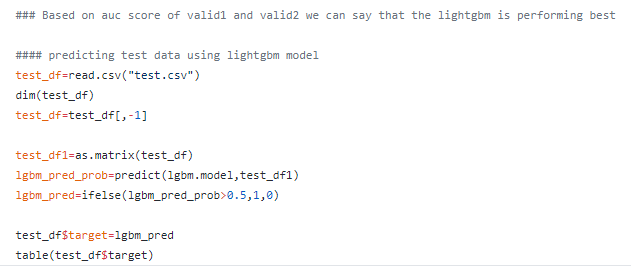




LightGBM Model



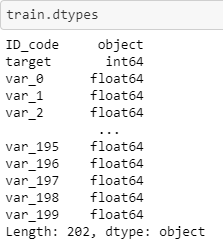




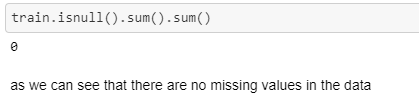
**Appendix B- Python Code**

Importing Data

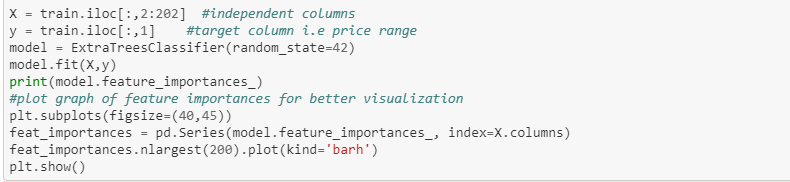


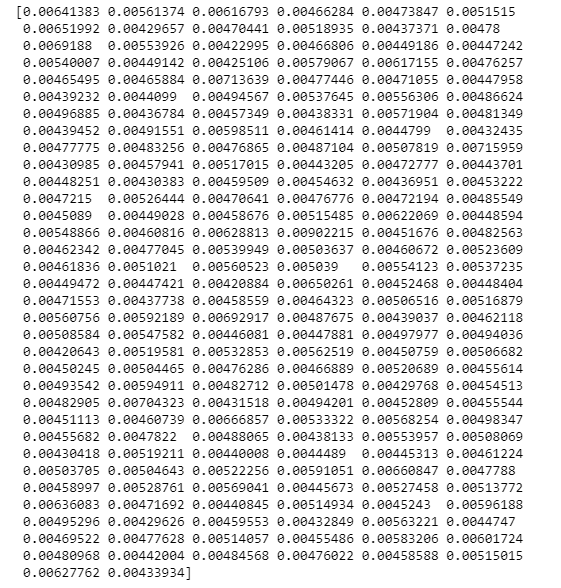


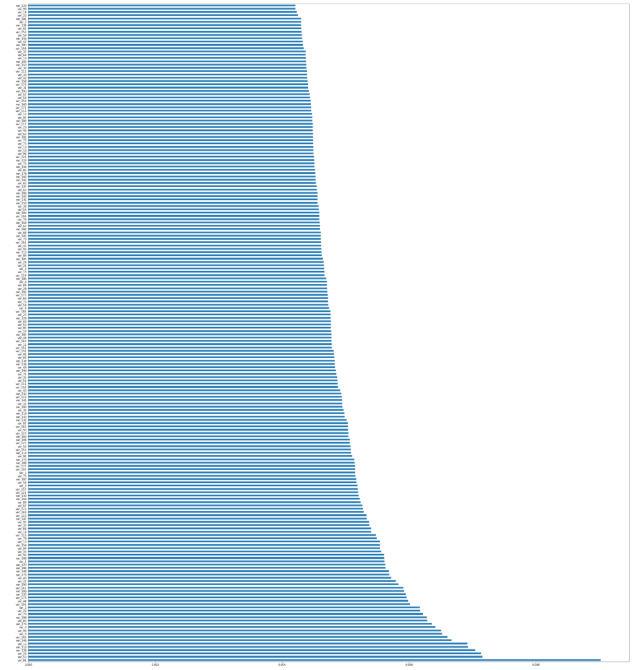
Missing Value Analysis



Important Features (Extra Trees Classifier)

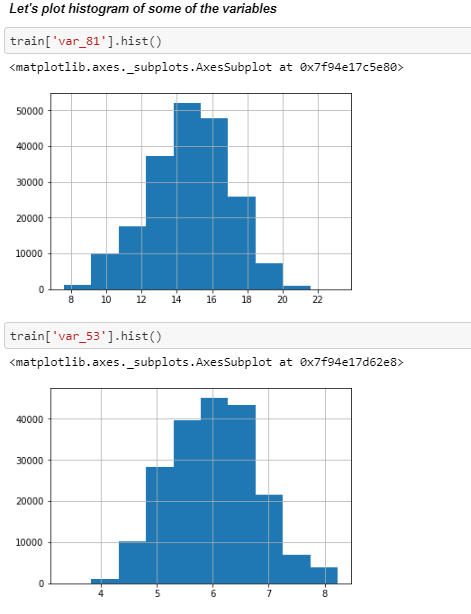


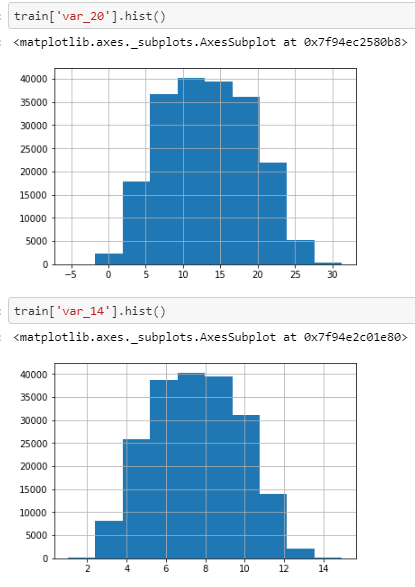




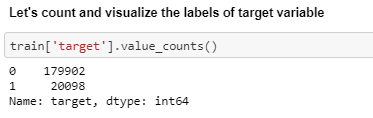
**(Note)** As we can see the above image some, of the most important variables are var\_81, var\_53, var\_26 and var\_139 and some of the least important variables in 200 variables are var\_20, var\_14, var\_98 and var\_120

Due to large number of variables size, we have minimized the image (kindly understand)





If we will see all the histograms, we can find out histogram of two of most important features are similar and histogram of least 2 important feature are similar



# 

# 

Correlation between Variables

# 

# 

# (Note) As we can see that there is very less correlation among independent variables. We have taken this above image for reference, and due to large number of variables we have shown only Var\_9 only for the reference.

# Logistic Regression Model

# 

# 

# (Note) Even though we are getting high accuracy but sometimes accuracy is not the best measure to evaluate a model specially if we have imbalanced data. So we will be using other measures like ROC, AUC and f1-scores

# 

# 

# (Note) As we can see that f1-score is very low for positive class. so by looking at the roc\_curve and f1-score we can say that this model is not performing well on imbalanced data and we will try other models.

# Using SMOTE (Handle target class imbalance problem)

# 

# 

# 

# 

# 

**(Note)** we can see from roc\_curve and f1-score that logistic regression is performing well on synthetic data.

# Decision Tree

# 

# 

# 

**(Note)** So, looking at the roc\_curve and f1-score we can say that decision tree also not performing well on imbalanced target class data.

# 

# 

# 

# (Note) As we can see accuracy is very low when we applied decision tree model on synthetic data that means decision tree is not performing good on synthetic data as well.

# Using LightGBM

# 

# 

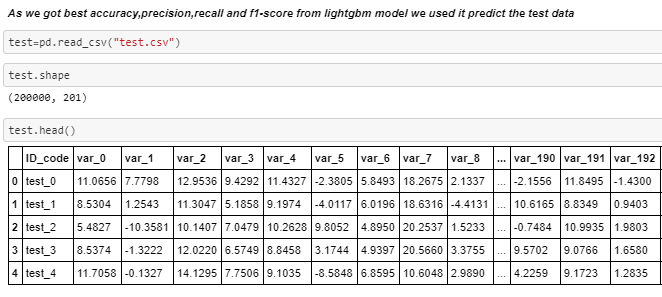
# 

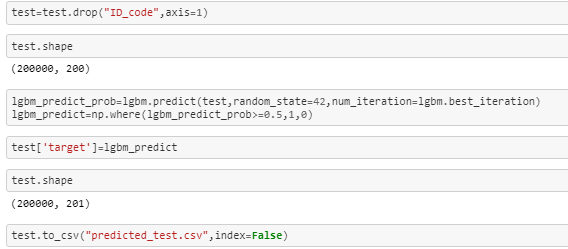
# 

# 

# 

**(Note)** from the roc\_curve and f1-score we can conclude that LightGBM model is performing well.





**(Important Note-)**

I would like to inform you that, I made this project on behalf of my knowledge and experience which I learned during my assessment from edwisor platform. However, I have attached Both R and Python code files for the reference. Kindly have a look at them.

This is my humble request you to evaluate my project report thoroughly. This is the last chance for me to complete the Project stage. Thereafter, I will move forward to Hiring stage.

Previously, I was failed in Bike Renting project due to not submitted the R and Python code files, and I was very disappointed at that moment.