Santander Customer Transaction Prediction

SAI KARTHIK

22-09-2019

CONTENTS

PROBLEM STATEMENT

Background - 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?

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.

Data Set :

1) test.csv

2) train.csv

Number of attributes :

You 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.

Missing Values: Yes

EVALUATION BASIS

FOR CLASSIFICATION THE ACCURACY METRICS NEED TO BE CONSIDERED ARE AUC, PRECISION & RECALL. YOU ARE FREE TO USE OTHER METRICS IN ADDITION TO THE AFOREMENTIONED METRICS.

IMPORTING LIBRARIES

There are many libraries in python on Jupyter notebook which helps in contributing various functionalities. For instance **numpy** library is used for numerical caluclations, **seaborn** library used for visualisations, **pandas** library is used for dataframes ,high perfomance and analysis.

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- import os
- from fancyimpute import KNN
- from random import randrange, uniform
- from scipy.stats import chi2_contingency
- %matplotlib inline

UNDERSTANDING THE DATA

The training datset consists of **200000 observations and 202 variables**. Data is well-oragnized and the standardised data. It does not have any **Missing Values.**

We have one target variable - **'target'** and a Id-column **id-code** which contains all the unique id's of the variables **and 199 Independent variables** starting from **var-0,var-1 .....var-199.**

This is a **Classification problem** where the task is to build a **model** which predicts whether the **customer makes the transaction or not.**

```
In [6]: santander_data_train.describe()
```

Out[6]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.0000(
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.408949	16.545850	0.2841(
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.866607	3.418076	3.3326:
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.347300	5.349700	-10.5055(
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.767700	13.943800	-2.3178(
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.385100	16.456800	0.3937(
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.003000	19.102900	2.9379(
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.447700	27.691800	10.1513(

8 rows × 201 columns

The datset is well processed and there are no missing values present in the dataset and most of the data is already normalised.

DATA PRE-PROCESSING

Data preprocessing is used to explore various aspects of data so that data can be formed in presentable format. We can draw various insights from the data by applying different pre-processing techniques Data type of variable is are mismatched with there values datatype. So will change there data type. Sometimes extra variable are present which are not required so we will remove them. Some time variable columns contain some extra string which is not required so we will remove it.

After importing libraries we will upload train dataset and use head function which will gives top 5 observation for analysis after that we start analyzing the dataset .

MISSING VALUE ANALYSIS:

#Create dataframe with missing percentage for train dataset

```
missing_val_train=pd.DataFrame(santander_data_train.isnull().sum())
```

#Rename variable

```
missing_val = missing_val_train.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})
```

#Calculate percentage

missing_val['Missing_percentage']=(missing_val['Missing_percentage']/len(missing_val))*100

	Missing_percentage
ID_code	0.0
target	0.0
var_0	0.0
var_1	0.0
var_2	0.0

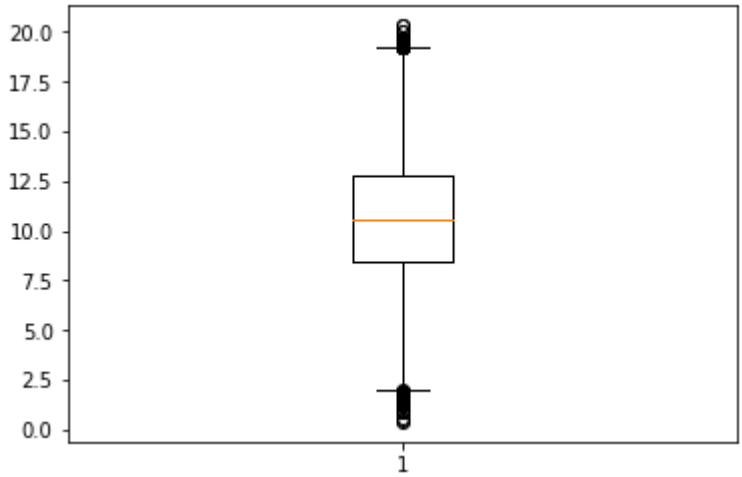
THERE ARE NO MISSING VALUES IN ANY VARIABLES.

OUTLIER ANALYSIS

Outliers are values which are inconsistent in nature which make data inappropriate . Which simply check that outlier using the box plot and the values locate at the extreme of the box plot are outlier we can also used the loop to get the outlier.

So once we get outlier we can remove them from data or we can put null value on the place of it and the impute null value by the various method we have learnt in previous section.

BEFORE OUTLIER ANALYSIS :



There are outliers which are present in the dataset.

we need to remove the outiers which are below 25th percnetile and above 75th percentile.

```
for i in numeric_data_var:
    q75,q25 = np.percentile(santander_data_train.loc[:,i],[75,25])
    print(i)
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)
```

Removing values which are below 25th percentile

```
santander_data_train=santander_data_train.drop(santander_data_train[santander_data_train.loc[:,i] < min].index)
```

Removing values which are above 75th percentile

```
santander_data_train=santander_data_train.drop(santander_data_train[santander_data_train.loc[:,i] > max].index)
```

The above code will give Inter Quartile range iqr which is difference between 75th and 25th percentiles.

FEATURE SELECTION

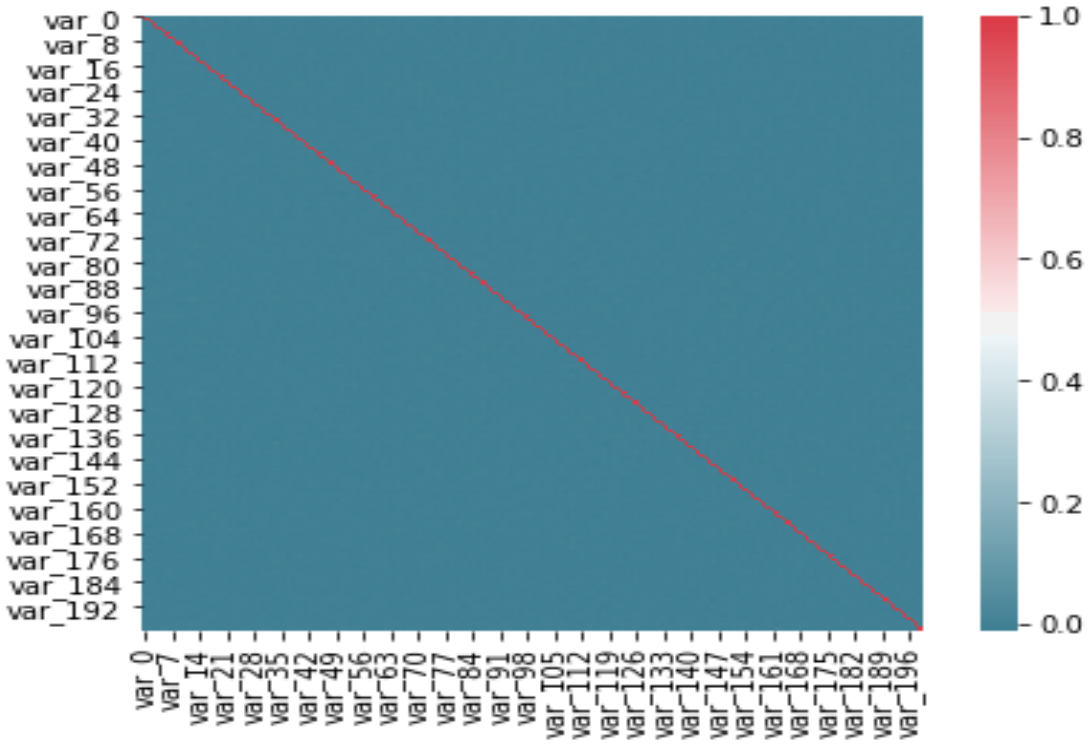
It is method in which we remove the continuous variable which are dependent on each other so that no two variables denoted same variation . this method used the concept of collinearity to find the dependency between the independent variable . most of the time we used heat maps to find there collinearity. If collinearity between two variables is equal to 1 or -1 we remove one variable from them.

CORRELATION ANALYSIS

Correlation analysis will gives us the variance among the variables.

- **There should be high correlation between Dependent and Independent Variable.**
- **There should be low correlation between any two Independent variables.**

HeatMap is the visualisation technique used for the correlation analysis.



From the above Heat Map we can infer that there is no correlation between the Indepent variables.i.e.. All the variables are independent of each other.

FEATURE SCALING

Feature scaling is the process of scaling the data into same measurement .There are two methods used for Feature scaling

1 . Normalisation

2 . Standardisation

As our Training set data is already standardised there is no need to apply Feature Scaling on the datset.

MODEL DEVELOPMENT

I applied different classification models to check which model performs well and selecting the model based on **AUC,PRECISION and RECALL**.

I used four different machine learning algorithms

1. **LOGISTIC REGRESSION MODEL**
2. **DECISION TREES CLASSIFICATION MODEL**
3. **RANDOM FORESTS**
4. **NAÏVE BAYES**

1. LOGISTIC REGRESSION MODEL

Logistic regression is a part of Generalised Linear Model. It uses log of odd to form a linear line . and a line which gives maximum log of likehood will be used to predict the data .

RESULTS OBTAINED FROM LOGISTIC MODEL

Recall = 26.34356068008599%

Precision = 69.05737704918033%

AUC =62.53%

Accuracy =92%

2. DECISION TREE CLASSIFICATION

It is a classification technique used C5.0 method to assigning a variable at every node from top to bottom. So the variable gives max information gain during the split will be prefer according to that every variable is decided . the variable gives maximum information gain will be present at the top of tree.

Information Gain = Entropy of system before split – Entropy of system after split

Entropy = uncertainity in the data / measure of impurity.

RESULT OF DECISION TREE

Recall = 19.53488372093023%

Precision = 19.174955160606554%

Auc =54.89%

Accuracy =84%

3 . RANDOM FORESTS

Random Forests uses many numbers of decision trees to form the forrest and that forrest uses to predict the value it uses essemble technique on that way error will be distributed.

It form many number number of trees by using the concept of bagging and boosting based on mode of result of that groups of trees we decide the result .

It uses Gini Impurity to split at node, lesser the gini impurity more chances are for variable preferability.

RESULT OF RANDOM FORREST

n_estimators=100

Recall = 0.019542700801250732

Precision = 100.0

Auc =50%

Accuracy =90%

4 . NAÏVE BAYES

It uses bayes laws of probability to predict the data . it assumes that all the variables are perfectly independent with each other , so we find probability

For each target variable on the bases of condition given. The target variable which gives max probability is considered as a result ,for example if probability of yes is more than no than no will be the result.

RESULT OF NAÏVE BAYES

Recall = 35.86085597029509%

Precision = 72.04554377699255%

Auc =67.08%

Accuracy =92%

SUMMARY OF THE MODEL

RESULTS OBTAINED FROM LOGISTIC MODEL

Recall = 26.34356068008599%

Precision = 69.05737704918033%

AUC =62.53%

Accuracy =92%

RESULT OBTAINED FROM DECISION TREE

Recall = 19.53488372093023%

Precision = 19.174955160606554%

Auc =54.89%

Accuracy =84%

RESULT OBTAINED FROM RANDOM FOREST

n_estimators=100

Recall = 0.019542700801250732

Precision = 100.0

Auc =50%

Accuracy =90%

RESULT OBTAINED FROM NAÏVE BAYES

Recall = 35.86085597029509%

Precision = 72.04554377699255%

Auc =67.08%

Accuracy =92%

CONCLUSION

PRECISION CRITERION:

* RF > NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE

RECALL CRITERION AS FOLLOW:

* NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE > RF

AUC CRITERION AS FOLLOW:

* NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE > RF

* For the most Accurate Model the value of AUC must be high.

* According to the question we have to predict the result based on recall , precision and AUC.

WE USE NAÏVE BAYES TO PREDICT OUR MODEL BECAUSE THE COMBINATION OF ALL THREE i.e recall ,precision and AUC MEASURING QUANTITY IS GOOD.

R CODE :

```
rm(list=ls(all=T))
#SETTING THE WORKING DIRECTORY
setwd("D:/Project_R/Inputfiles")
getwd()

#Loading Libraries in R
#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

install.packages(x)
lapply("unbalanced", require, character.only = TRUE)
rm(x)


## Read the data
santander_data_train = read.csv("train.csv", header = T, na.strings = c(" ", "", "NA"))

#Get the names of the Variables
names('santander_data_train')

#MISSING VALUE ANALYSIS
#-----------------------------------------------------------------------------------------------------------------------
#CREATE A DATAFRAME WITH MISSING PERCENTAGE
missing_values = data.frame(apply(santander_data_train,2,function(x){sum(is.na(x))}))

#CONVERT THE ROW INTO COLUMNS
```

```
missing_values$Columns = row.names(missing_values)

#RENAME THE COLUMN AS MISSING_PERCENTAGE

names(missing_valUes)[1] =  "Missing_percentage"

#CALUCLATING THE MISSING PERCENTAGES

missing_values$Missing_percentage = (missing_val$Missing_percentage/nrow(santander_data_train)) *
100

#WRITING THE MISSING VALUES INTO A FILE Missing_percentages.csv

write.csv(missing_values, "Missing_percentages.csv", row.names = F)

#Plot the missing values using bargraph

ggplot(data = missing_values[1:3,], aes(x=reorder(Columns, -Missing_percentage),y =
Missing_percentage))+

  geom_bar(stat = "identity",fill = "grey")+xlab("Parameter")+

  ggtitle("Missing data percentage (Train)") + theme_bw()

#we can observe that there are no missing values in the dataset



#----------------------------------------EXPLORATORY DATA ABALYSIS ----------------------------------------------------------
------------

#OUTLIER ANALYSIS

#CHECK FOR THE OUTLIERS USING BOX-PLOT METHOD

#WE CAN REMOVE OUTLIER VALUES



#sTORE THE NUMERIC VARIBALES IN ONE ARRAY

numeric_index = sapply(santander_data_train,is.numeric)

numeric_data = santander_data_train[,numeric_index]
```

```
cnames = colnames(numeric_data)


#plotting box plot for the variables

for (i in 1:length(cnames))

{

  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "target"), data =
subset(santander_data_train))+

        stat_boxplot(geom = "errorbar", width = 0.5) +

        geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

                     outlier.size=1, notch=FALSE) +

        theme(legend.position="bottom")+

        labs(y=cnames[i],x="target")+

        ggtitle(paste("Box plot of target for",cnames[i])))

}


## Plotting plots together

gridExtra::grid.arrange(gn1,gn5,gn2,ncol=3)

gridExtra::grid.arrange(gn6,gn7,ncol=2)

gridExtra::grid.arrange(gn8,gn9,ncol=2)


#Remove outliers using boxplot method

df = santander_data_train

#santander_data_train = df


val = santander_data_train$previous[santander_data_train$previous %in%
boxplot.stats(santander_data_train$previous)$out]

#

santander_data_train = santander_data_train[which(!santander_data_train$previous %in% val),]

#
```

```
#loop to remove from all variables
for(i in cnames){   print(i)
  val = trans[,i][santander_data_train[,i] %in% boxplot.stats(santander_data_train[,i])$out]
  #print(length(val))
  santander_data_train = santander_data_train[which(!santander_data_train[,i] %in% val),]
}
#AFTER REMOVING OUTLIER WE GET 175073 OBSERVATION FROM 200000 OBSERVATIONS SO 24927 OBSERVATION ARE EXIST AS AN OUTLIER IN TRANS DATASET¶
----------------------------------------------Feature Selection-----------------------------------------------
## Correlation Plot

#DATA SELECTION
#FOR DATA SELECTION WE HAVE SHOWN EARLIER THAT NO VARIABLE IS DEPENDENT WITH EACH OTHER CORRELATION VALUE IS COMES OUT TO BE 1
#WE HAVE FETCH OUT INITIAL TO 10 VARIABLES TO GET HEAT MAP TO GET FOW CORELATION FORM BUT IT SHOWS NO TWO ARE DEPENDENT ON EACH OTHER

corrgram(trans[,numeric_index], order = F,upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
#----------------------------------------------FeatureScaling-----------------------------------------------
#FeatureScaling will be done based on the model

#----------------------------------------------MODEL DEVELOPMENT-----------------------------------------------
#Divide data into train and test using stratified sampling method
set.seed(1234)
train.index = createDataPartition(santander_data_train$target, p = .80, list = FALSE)
train = santander_data_train[train.index,]
```

```
test  = santander_data_train[-train.index,]

########Logistic Regression############
logit_model = glm(target ~ ., data = train, family = "binomial")

#summary of the model
summary(logit_model)

#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = test, type = "target")

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)



##Evaluate the performance of classification model
ConfMatrix_RF = table(test$target, logit_Predictions)

#False Negative rate
FNR = FN/FN+TP

#recall
Recall=(TP*100)/(TP+FN)

#precision
precision= (TP*100)/(TP+FP))

#LOGISTIC REGRESSION RESULTS
recall = 26.34356068008599
precision = 69.05737704918033
AUC =62.53%
```

Accuracy =92%

```
#-------------------------------------------------------------------------------------------------------------
######DECISIONTREE CLASSIFICATION######
#Clean the environment
rmExcept("santander_data_train")

#Divide data into train and test using stratified sampling method
set.seed(1234)
train.index = createDataPartition(santander_data_train$target, p = .80, list = FALSE)
train = santander_data_train[train.index,]
test  = santander_data_train[-train.index,]

##Decision tree for classification
#Develop Model on training data
C50_model = C5.0(responded ~., train, trials = 100, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
C50_Predictions = predict(C50_model, test[,-202], type = "class")

##Evaluate the performance of classification model
ConfMatrix_C50 = table(test$responded, C50_Predictions)
confusionMatrix(ConfMatrix_C50)

#recall
```

Recall=(TP*100)/(TP+FN)

#precision

precision= (TP*100)/(TP+FP))

#DECISION TREE RESULTS

recall = 19.53488372093023%

precision = 19.174955160606554%

AUC =54.89%

accuracy =84%¶

#----------NAIVE BAYES MODEL----------

library(e1071)

#Develop model

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

#predict on test cases #raw

NB_Predictions = predict(NB_model, test[,2:202], type = 'class')

#Look at confusion matrix

Conf_matrix = table(observed = test[,202], predicted = NB_Predictions)

confusionMatrix(Conf_matrix)

#NAIVE BAYES RESULTS

recall = 35.86085597029509%

precision = 72.04554377699255%

AUC =67.08%

Accuracy =92%¶

#statical way

```
mean(NB_Predictions == test$target)



#------------------------------RANDOMFOREST PREDICTION---------------------------------------------------------------------
RF_model = randomForest(target ~ ., train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)

#Extract rules
exec = extractRules(treeList, train[,-202])  # R-executable conditions

#Visualize some rules
exec[1:2,]

#Make rules more readable:
readableRules = presentRules(exec, colnames(train))
readableRules[1:2,]

#Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-17], train$target)  # get rule metrics

#evaulate few rules
ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-17])

#Evaluate the performance of classification model
```

```
ConfMatrix_RF = table(test$target, RF_Predictions)
confusionMatrix(ConfMatrix_RF)

#recall
Recall=(TP*100)/(TP+FN)

#precision
precision= (TP*100)/(TP+FP))

#RANDOM FOREST RESULTS
recall = 0.019542700801250732
precision = 100.0
AUC =50%
accuracy =90%
#----------------------------------------------------SUMMARY-----------------------------------------------------------
LOGISTIC REGRESSION:Results
recall = 26.34356068008599
precision = 69.05737704918033
AUC =62.53%
accuracy =92%

DECISION TREES:Results
recall = 19.53488372093023%
precision = 19.174955160606554%
AUC =54.89%
accuracy =84%

NAIVE BAYES:Results
recall =  35.86085597029509%
precision =  72.04554377699255%
```

AUC =67.08%

accuracy =92%

RANDOM FORREST:Results

n_estimators=100

recall = 0.019542700801250732

precision = 100.0

AUC =50%

accuracy =90%

PRECISION CRITERION:

* RF > NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE

RECALL CRITERION AS FOLLOW:

* NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE > RF

AUC CRITERION AS FOLLOW:

* NAÏVE BAYES > LOGISTIC REGRESSION > DECISION TREE > RF

#* For the most Accurate Model the value of AUC must be high.

#* According to the question we have to predict the result based on recall , precision and AUC.

#* According to the results which were obtained using all four machine learning algorithm we can infer that "Naive Bayes" is giving all the three parameters equally good. In Random Forest precision is high but recal is very low.

#* So the accurate Model which can be selected from the results for this Santander problem is Naive Bayes.

PREDICTING OUR TEST DATA

After forming the machine learning algorithm and test it for test data we have fetch out from train dataset given. And the get most accurate method

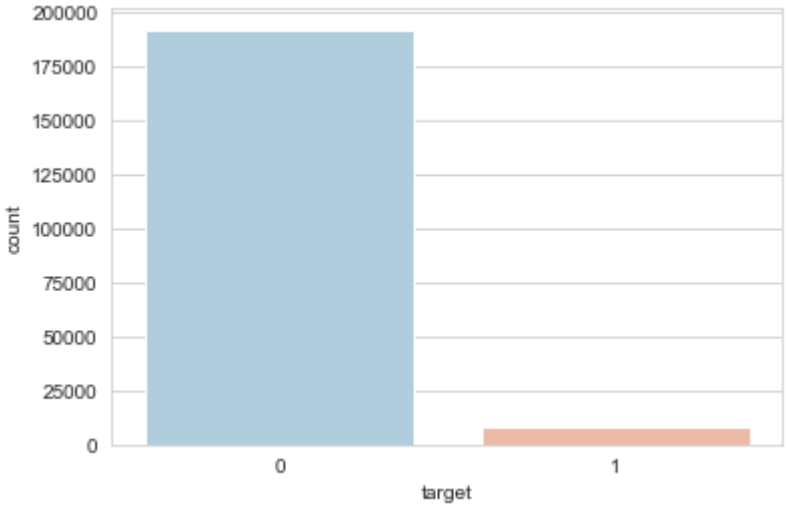
Now we are ready to predict any external data given to us.

Now in the external data named as test.csv is given to us contain all variables of train data except “target” variable. Our task is to predict the target value in form of 0 and 1 where using the most appropriate algorithm we have formed in previous section.

As we have developed in our previous section that NAÏVE BAYES gives most accurate model so we will use NAÏVE BAYES to predict the target value.

- So to do that first we upload test dataset into python
- Now get the head of the test dataset to check the values it contain
- Now form a vector contain the value of ID_code which later join to dataset.
- Now drop ID_code column from dataset as it has no use and our naïve bayes model can not read this variable
- Now predict the target variable using the this test dataset remains using the command predictions_test = NB_model.predict(test).
- Now new datset is form using vector prediction_test and id_code having the variable name target and ID_code respectively.
- Now this dataset is join with the test dataset formed to predict the model
- Now it upto us we can change 0 into “not make transaction” and 1 into “make transaction” string which is easy to understand for layman.
- Now after predicting anf forming the finel dataset we will upload it as .csv file in our hard disc named as "test_prediction result.csv"
- So the number of target values 0 and 1 formed as a prediction can be calculated using the function . count. The image formed is as follow:

the count present in train data is as follow:



The above bargraph indicates based on the prediction analysis most of the customers falls under the category of “not make transaction”.

SAVED FILES DURING PROCESS :

The file saved during the program is written with Italian font and blue in colour in this report at the place where it is formes. The test.csv and train.csv file is given in question and following files are formed during the course of project.

- Missing_percentage.csv : contain the number of missing value percentage in each column for a variable from maximum to minimum sort value. It show all the percentage are zero

- santander_final_predictions.csv: this file formed after predicting the value from test.csv and add the target column contain predicted value as a new column of test.csv dataset.

CODE FILES :

PYTHON - **Santander_Project_MainFile.ipynb**

R - **Santander_Project_MainFile_R.r**

Final predictions file - **santander_final_predictions.csv**

INSTRUCTIONS TO RUN CODE :

INSTRUCTION TO RUN AND DEPLOY THE **PYTHON AND R** CODE IS WRITTEN WITH THE CODE ITSELF.

ONLY THE FILE LOCATION SHOULD BE CHANGED IN **OS.CHDIR FUNCTION IN JUPYTER NOTEBOOK AND setwd("path of working directory") in R** ACCORDING TO THE USER FILE LOCATION.

ALL THE SAVED FILE DURING THE PROCESS ARE GIVEN WITH CODE.

USER CAN SIMPLY USE SHIFT + ENTER TO GET ALL COMMAND RUN IN PYTHON .

IN R WE CAN USE Ctrl + ENTER TO RUN THE SPECIFIC LINE.

REFERENCES :

> https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

> https://www.statisticssolutions.com/what-is-logistic-regression/

> https://www.kaggle.com/fl2ooo

> https://scikit-learn.org/stable/modules/generated/

> https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

> https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn

> https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

> https://www.edureka.co/blog/naive-bayes-tutorial/

> https://towardsdatascience.com/understanding-random-forest-58381e0602d2

> https://statinfer.com/104-3-5-box-plots-and-outlier-dectection-using-python/

THANK YOU