**Date: 21/10/2019**

**Submitted By- Vishal Tyagi**

Santander Customer Prediction Project Report

**Contents**

1. **Introduction** 
   1. **Problem Statement**
   2. **Data**
   3. **Exploratory Data Analysis**

1. **Methodology** 
   1. **Pre-Processing** 
      1. **Missing Value Analysis**
      2. **Outlier Analysis**
      3. **Data Understanding**
      4. **Feature Selection**
      5. **Feature Scaling**

* 1. **Model Development** 
     1. **Choosing Model**
     2. **Logistic Regression**
     3. **Decision Tree**
     4. **Random Forest**

1. **Conclusion** 
   1. **Model Evaluation**

**4. Summarize**

**Appendix A – Python Script**

**Appendix B – R Script**

**References**

# Chapter 1

## 1. Introduction

### 1.1 Problem Statement

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

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

### 1.2 Dataset

Image 1 -Train Dataset

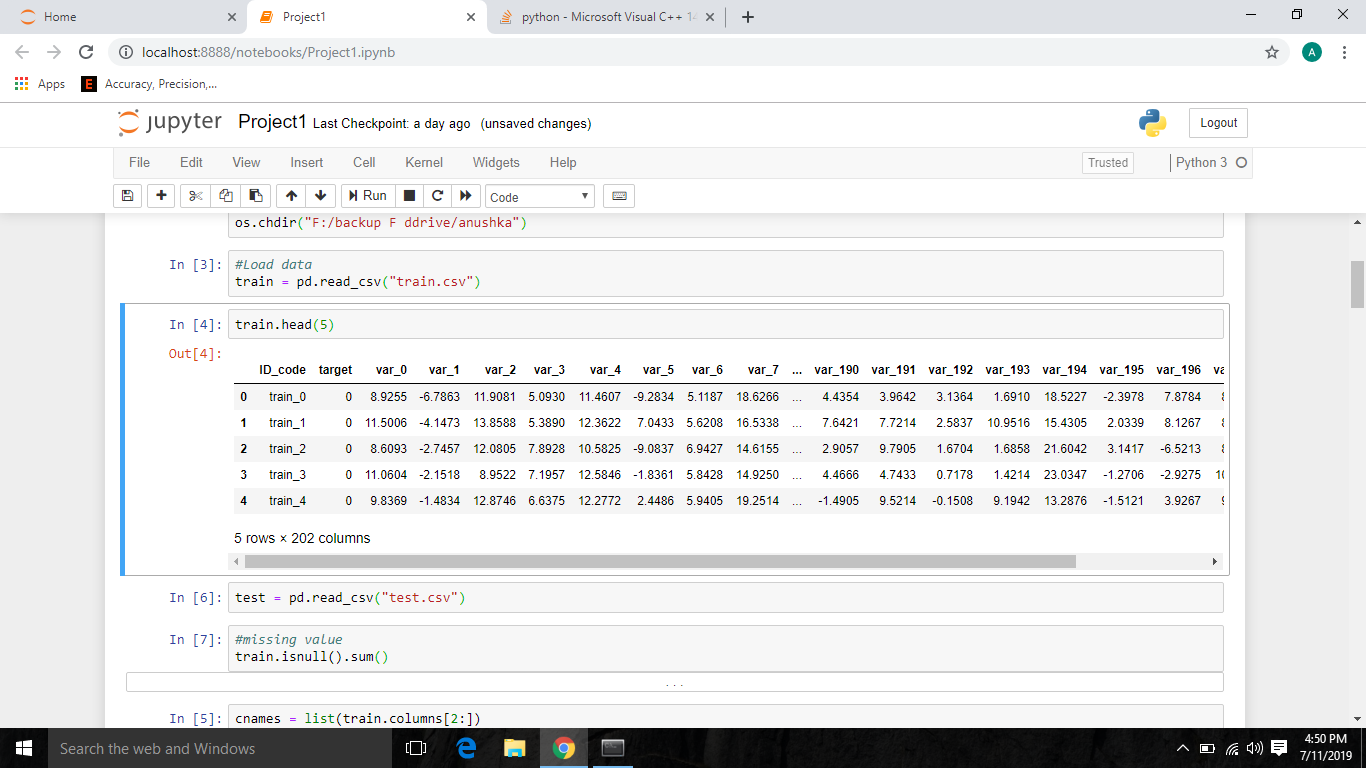
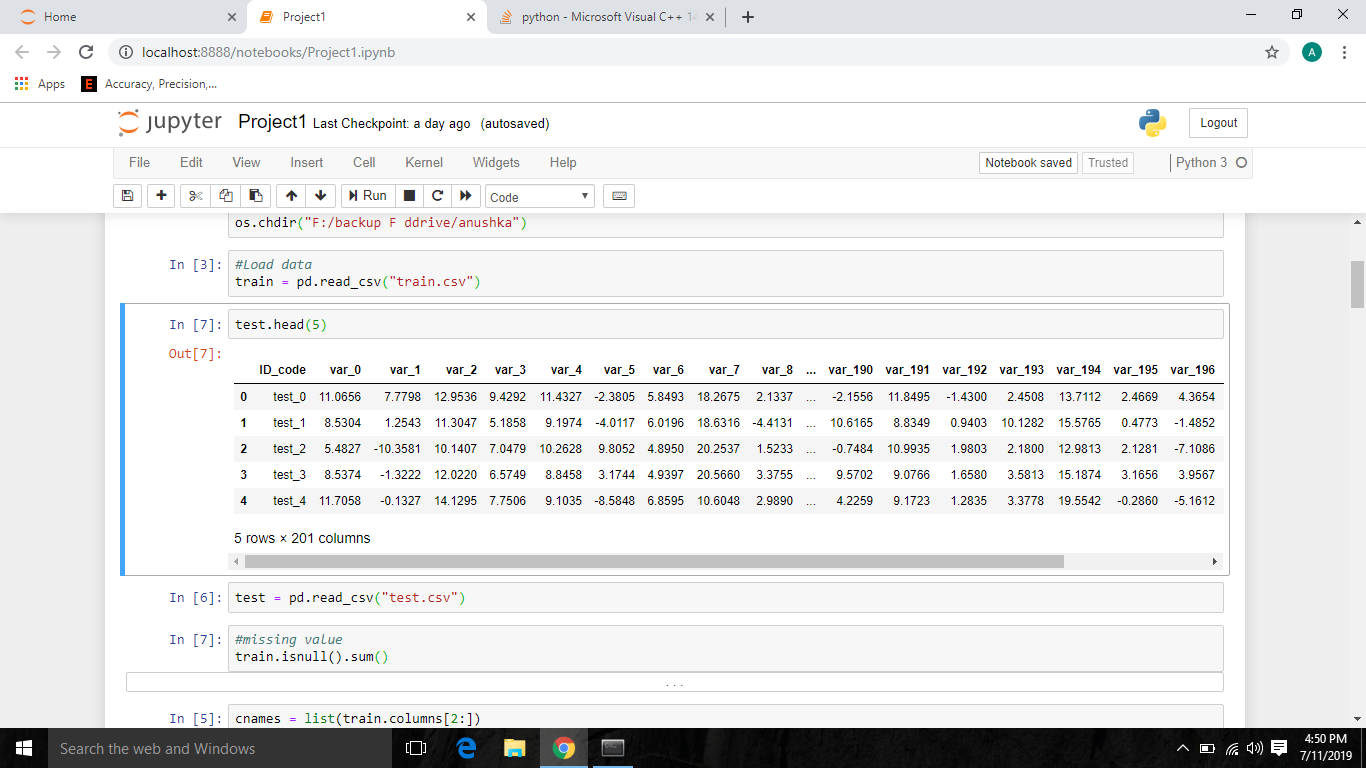


Image 2 -Test Dataset



In image 1, we can see our train data. It has 200000 observations and 202 features, including target which is our target variable.

In image 2, we can see test data, whose target variable i.e. “target” values to be predicted. It has 200000 observations and 201 features, where, “target” is not available, as we predict those values later in the journey.

As said above, our primary focus is to determine the values for our target variable “target” for future test cases using the above shown dataset in image 1.

### 1.3 Exploratory Data Analysis

In Exploratory Data Analysis, we go through different things, like:

* **Brainstorming** – Here, I have actually draw a rough sketch which talks about – the steps you are going to follow to achieve my given objective.

# Chapter 2

## 2. Methodology

Now, we have the dataset and also, we discussed about Exploratory Data Analysis, let’s talk about the **methodology** we are going to follow to achieve our goal.

We will be going through:

* Pre-processing which includes missing value analysis, outlier analysis, feature selection and feature scaling.
* Model development, where we will choose what machine learning algorithms to apply.

### 2.1 Pre-processing

In pre-processing, we actually apply few techniques like missing value analysis, outlier analysis, feature selection, feature scaling.

Why we do that? Well, actually we never get a structured data to work with. Always messy data is handed to us, and we need to clean that data.

The data may have many observations (rows in dataset), where values in few fields will be absent. We can also say, there may be some inconsistent values in a variable (column in dataset), when compared with other values.

When we go for model development, we should have a structured dataset. We can’t go forward for model development, if we don’t apply pre-processing techniques on data and convert it into structured format.

#### 2.1.1 Missing Value Analysis

Missing value analysis, as the name suggests, we face with situations, where we are given with dataset, and we have missing values in the observations.

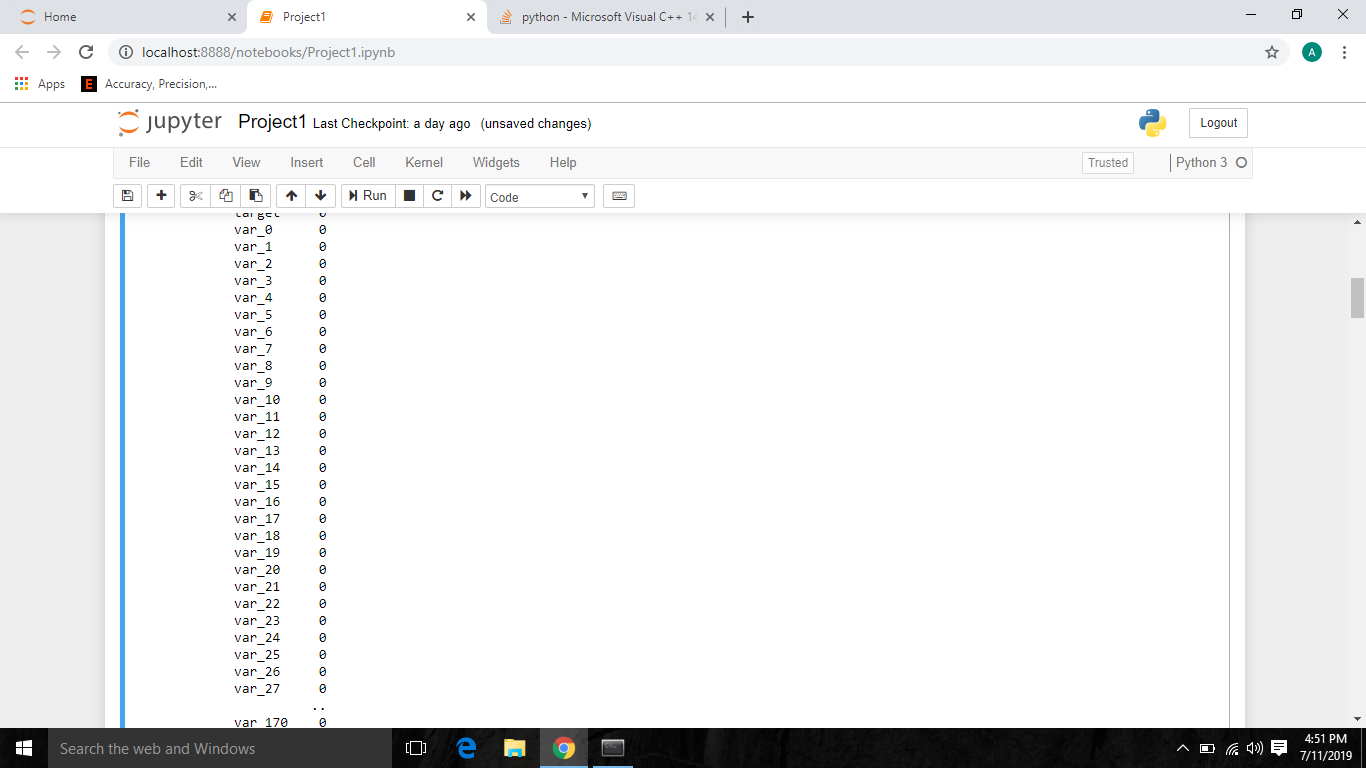
The reason why we have missing values may be plenty, like human error, the user didn’t want to share his complete information, the one who was supposed to fill the data may didn’t work properly.

But, as discussed earlier, we have to give structured data to the model. In order to achieve that, we perform missing value analysis on top of the data to clean the data, to transform the data from unstructured format to structured format.

We give a line of code and it gives us the total number of missing values in each column. Later, we impute those missing values using mean, median or KNN method for numerical variables or using mode for categorical variables. In some cases, we may delete the observations with missing values, only when we have a case where we got few observations with missing values.

After imputing the missing values, we proceed further with outlier analysis. In our project, we had 0 missing values.

Image 3



Another important thing is, we apply missing value techniques only on numerical variables.

#### 2.1.2 Outlier Analysis

Outliers may be defined as the inconsistent values in a variable. For example, a = 1,2,3,4, 20. In object a, 20 is inconsistent, in terms of mean. Another important thing is, we apply outlier analysis only on numerical variables.

Outliers are used for fraud detection. Let’s say, in one bank account, consistently, we observe an amount which ranges from 50,000 to 1,00,000, but in one case 10 lakhs gets deposited. In that case, simply using outlier technique, we can get the inconsistent values.

In our project, we found outliers and we deleted the outliers in these variables as the number of outliers were minimum. We used to describe function to know minimum and maximum.

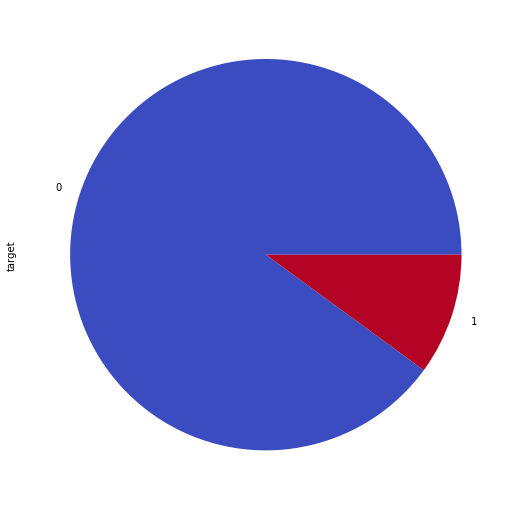
#### 2.1.3 Data Understanding

Understanding visually is easier sometimes and for that purpose we have few libraries in python which allows us to plot awesome visualizations.

We get data, and in order to talk about how few attributes relationship with each other, we can use these visualizations.

You can give two variables, to know how they are related with each other. You can also give three variables to understand the relationship between three variables. You got infinite possibilities to plot. Let’s have a view on few of the plots we used in our project.

Image 4



Since our problem is a case of classification, it is only fair we check how our target variable is distributed. There are only 2 values for the target, i.e., 0 and 1. The outcome we are interested is in 1, because it denotes a positive scenario where the customer made a specific transaction. As can be seen from the figure the output is highly uneven, with only about 20% of the data contributing to the desired outcome. From the pie chart we can clearly tell that there are a lot of instances of the negative case rather than the desired one.

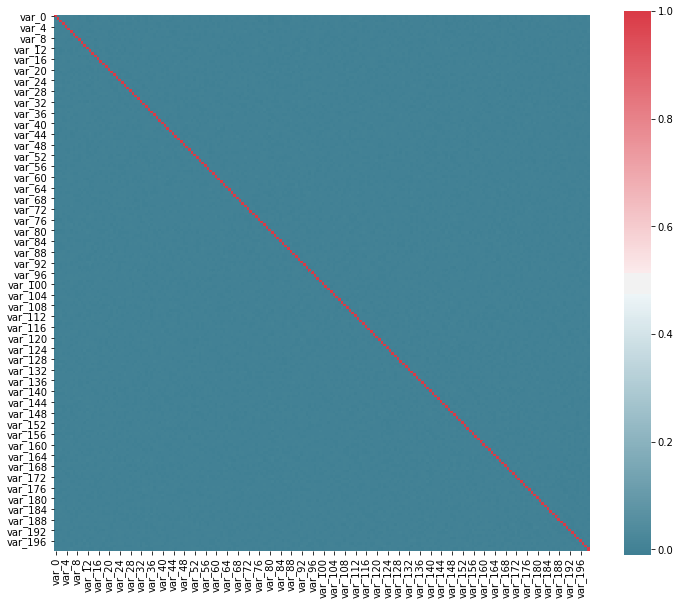
#### 2.1.4 Feature Selection

In Feature Selection, we use few techniques to know which variables are important to us, it’s all about selecting subset of variables from all variables. Actually, when we are given with a dataset, we perform exploratory data analysis, later data pre-processing to clean the data and transform it from unstructured to structured data to feed into model.

However, after this, we may face a situation where we may have variables which have same information with them about the target variable. Let’s talk about an example, in a situation where we are sending five people on a mission. Later, you came to know that, two individuals have the same exact information with them about the mission. Definitely you would drop one, in order to reduce infrastructure and complexity.

The same way, we also drop few variables if they have same information. We always aim that, there should be no independent variable which talks the same as other independent variables but, we appreciate those variables which talks more about the target variable.

In our project, we did correlation analysis among Indpendent variable. Let’s check it:



Now, we can see, data is not correlated.

#### 2.1.5 Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or

features of data. In data processing, it is also known as data normalization and is

generally performed during the data preprocessing step. Since the range of values of

raw data varies widely, in some machine learning algorithms, objective functions will

not work properly without normalization. Widely used feature scaling methods are Normalization and Standardization.

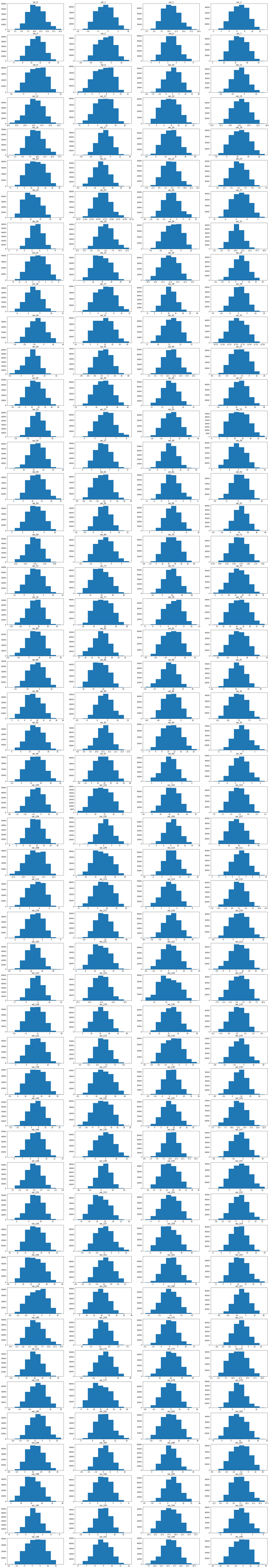
• **Normalization**: Normalization refer to the dividing of a vector by its length. normalization normalizes the data in the range of 0 to 1. It is generally used when we are planning to use distance method for our model development purpose such as KNN. Normalizing the data improves convergence of such algorithms. Normalisation of data scales the data to a very small interval, where outliers can be loosed.

• **Standardization**: Standardization refers to the subtraction of mean from individual point and then dividing by its SD. Z is negative when the raw score is below the mean and Z is positive when above mean. When the data is distributed normally you should go for standardization.

.

As we can see from the figure below most of the data is normally distributed. It is very important for the data to be normally distributed because then 68% of the data would be explained by 1 standard deviation, while 95% of data can be explained by 2 standard deviations. So, we would be running our models to predict for a majority of the population. If there were skewness it would suggest the presence of outliers which would have to be removed. However, since the data is normally distributed there are no outliers.

Fig - Distribution of independent variables using histogram



### 2.2 Model Development

Model Development, is the phase which comes after we are done with applying the exploratory data analysis, data pre-processing techniques, on the top of data.

The data, will be in structured format, which was our goal, is now ready to develop model.

After we defined our objective and received the data, we transformed it into our required form, we enter into model development, but before that, let’s discuss about model selection.

#### 2.2.1 Model Selection

The dependent variable, i.e., ‘target’ is a categorical variable, with values as 0 or 1. Therefore, the models we use must tend to classification problems. I chose Logistic Regression, Decision Tree and Naïve Bayes algorithms to build a model to fit the data and come to accurate conclusions.

The error metrics used was Confusion matrix and AUC (Area under the curve) or ROC (Receiver Operating Characteristics).

Confusion matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

Fig -Confusion Matrix



AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.

Fig- AUC – ROC curve



#### 2.2.2 Logestic Regression

Logistic regression is used for classification problems with binary outputs. Since our output also has two possible outcomes, logistic regression would fit the data properly. Logistic regression is based on the sigmoid function. If all the input variables result in a value higher than the threshold value the output will be considered 1, while if all the input variables result in a value lower than the threshold value the result will be considered as 0.

Our model also works on the same principle.

Fig- Logistic Regression sigmoid function



Confusion Matrix: Based on the figures from the confusion matrix in the figure below, various model predictors were calculated as follows.

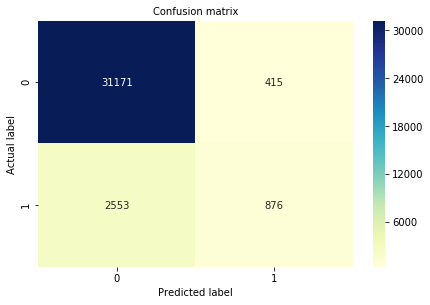
Accuracy: 0.9170069970012852

Precision: 0.6906419180201083

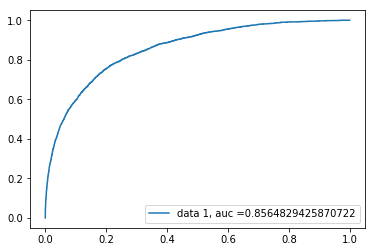
Recall: 0.2627243306854957

The accuracy of the model is 91.7% which is very good, however the precision is 69% meaning that the model predicted correct for 69% of the predictions it made.

Fig - Confusion Matrix for logistic regression



ROC -AUC curve: As can be seen from the figure below the AUC is 85.64, which is a pretty good number. We can say that our model is performing well



#### 2.2.3 Decision Tree

Decision tree as the name suggests is a tree – structured algorithm based on the decision made at every node. The leaf node is reached when no further branching is possible. We will use all of the 200 predictor variables to come at the decision of whether the customer will perform the transaction or not.

Confusion Matrix: Based on the figures from the confusion matrix in the figure below, various model predictors were calculated as follows.

Accuracy: 0.8422104812223333

Precision: 0.19920769666100735

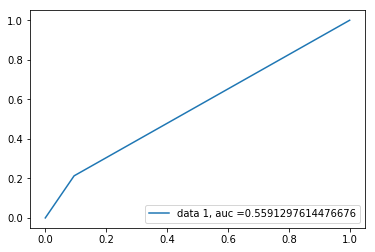
Recall: 0.20711974110032363

The accuracy is pretty good at 84.2%. However, if we have a look at the precision and recall values, both of them are extremely low indicating that the model is not up to the mark. The model has probably over fitted the data. This can be expected due to the large number of predictor variables. It is advisable not to use decision tree for such problems.

#### Fig - Confusion Matrix for decision tree model

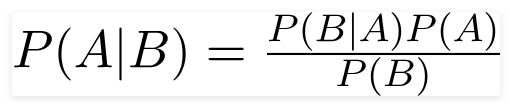
#### 

ROC – AUC curve: As can be seen from the figure below the AUC is 56, which is not a good number. We can say that our model is under performing and decision tree is not fit for this type of classification problem.



#### 2.2.4 Naive\_Bayes

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem.



Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Confusion Matrix: Based on the figures from the confusion matrix in the figure below, various model predictors were calculated as follows.

Accuracy: 0.9229187491075254

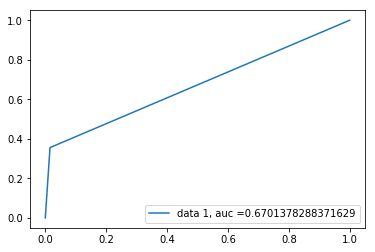
Precision: 0.7134502923976608

Recall: 0.3557888597258676

The accuracy is very good at 92%, so is the precision at 71%. Recall value is lower but however, we desire a higher precision value rather than recall. So based on that the model is performing pretty well.

ROC – AUC curve: As can be seen from the figure below the AUC is 67, which is good compared to Decision tree model. We can say that our model is performing fairly.

Fig - ROC – AUC curve for Naïve Bayes model



# Chapter 3

## Conclusion

### 3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of predicting Santander Customer transaction, Computation Efficiency and predictive performance hold much higher significance than interpretability. Therefore, we will use Predictive performance and Computational efficiency as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure. For our case we have used Confusion matrix and ROC as an error metric measure.

For our model it is highly important that it be precise, because correct prediction will help the customer grow their business. Considering all the attributes from confusion matrix and ROC curve, I choose Naïve Bayes to be the final model.

Naïve Bayes has a very high precision and accuracy. Though the AUC value is lower compared to Logistic regression model, for our model precision holds a higher weightage.

# Chapter 4

## **Summarize**

This project can help the business in achieving the strategic goals by predicting the percentage of customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Will help the client to predict about his customers and increase his business.

## Appendix A – Python Script

# Import libraries

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

from collections import Counter

from scipy.stats import chi2\_contingency

from sklearn.model\_selection import train\_test\_split

from sklearn import tree

import graphviz

from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

# Set Current directory

os.chdir("C:\\Users\\Vishal Tyagi\\Desktop\\R\\projects\\Santander")

os.getcwd()

#import data

train = pd.read\_csv('train.csv')

test = pd.read\_csv('test.csv')

# explore data

train.describe()

test.decribe()

# dimension of data

train.shape

test.shape

# name of columns

list(train)

list(test)

# data detail

train.info()

test.info()

################################# Missing value analysis ###################

train.isnull().sum().sum()

test.isnull().sum().sum()

############################## outliers analysis ########################

cnames1 = train.columns[2:]

cnames2 = test.columns[1:]

#Detect and delete outliers from train data

for i in cnames1:

q75, q25 = np.percentile(train.loc[:,i], [75 ,25])

iqr = q75 - q25

minimum = q25 - (iqr\*1.5)

maximum = q75 + (iqr\*1.5)

train = train.drop(train[train.loc[:,i] < minimum].index)

train = train.drop(train[train.loc[:,i] > maximum].index)

#Detect and delete outliers from test data

for i in cnames2:

q75, q25 = np.percentile(test.loc[:,i], [75 ,25])

iqr = q75 - q25

minimum = q25 - (iqr\*1.5)

maximum = q75 + (iqr\*1.5)

test = test.drop(test[test.loc[:,i] < minimum].index)

test = test.drop(test[test.loc[:,i] > maximum].index)

################################## Feature Selection ###########################

# Calculation of correlation between numerical variables

cnames1 = train.columns[2:]

df\_num = train.loc[:,cnames1]

corr = df\_num.corr()

print(corr)

# plotiing the heatmap

f, ax = plt.subplots(figsize=(12, 10))

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),square=True, ax=ax)

plt.show()

#################################### Feature Scaling ############################

train['target'].value\_counts()

train['target'].value\_counts().plot(kind="pie", figsize=(12,9), colormap="coolwarm")

print('Distribution plot for predictor variables')

plt.figure(figsize=(30, 185))

for i, col in enumerate(cnames1):

plt.subplot(50, 4, i + 1)

plt.hist(train[col])

plt.title(col)

print('Distribution of predictor variables with respect to target variable')

plt.figure(figsize=(30, 185))

for i, col in enumerate(cnames1):

plt.subplot(50, 4, i+1)

plt.hist(train[train['target'] == 0][col], alpha = 0.5, label = '0', color = 'b')

plt.hist(train[train['target'] == 1][col], alpha = 0.5, label = '1', color = 'r')

plt.title(col)

# Data is noramlly distributed,no need to use normalization and standarization techniques

################################## Split into train and test data ##################

#Seperate target and predictor variables

y = train['target']

x = train.drop(['target', 'ID\_code'], axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2)

#Model building

########################## Logistic Regression ################################

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X\_train,y\_train)

y\_pred = logreg.predict(X\_test)

#Model evaluation using confusion matrix

from sklearn import metrics

conf\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

conf\_matrix

from matplotlib import pyplot as plt

class\_names=[0,1] # name of classes

fig, ax = plt.subplots(figsize = (10,10))

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(conf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix', y=1, fontsize = 10)

plt.ylabel('Actual label', fontsize =10)

plt.xlabel('Predicted label', fontsize = 10)

#Model Evaluation

print('Accuracy:', metrics.accuracy\_score(y\_test,y\_pred)) #0.9170069970012852

print('Precision:', metrics.precision\_score(y\_test,y\_pred)) #0.6906419180201083

print('Recall:', metrics.recall\_score(y\_test,y\_pred)) #0.2627243306854957

# F-1 score

Accuracy = metrics.accuracy\_score(y\_test,y\_pred)

Precision = metrics.precision\_score(y\_test,y\_pred)

Recall = metrics.recall\_score(y\_test,y\_pred)

f1\_score = 2\*((Recall\*Precision)/(Recall+Precision))

print(f1\_score)

#Area under Receiver Operating Curve(AUC)

y\_pred\_prob = logreg.predict\_proba(X\_test)[::, 1]

fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_prob)

auc = metrics.roc\_auc\_score(y\_test, y\_pred\_prob) #0.8564829425870722

plt.plot(fpr, tpr, label = 'data 1, auc ='+str(auc))

plt.legend(loc = 4)

plt.show()

############################ Decision Tree ###############################

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf = clf.fit(X\_train,y\_train)

y\_pred\_clf = clf.predict(X\_test)

conf\_matrix\_clf = metrics.confusion\_matrix(y\_test, y\_pred\_clf)

conf\_matrix\_clf

class\_names=[0,1] # name of classes

fig, ax = plt.subplots(figsize = (8,8))

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(conf\_matrix\_clf),

annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix', y=1, fontsize = 10)

plt.ylabel('Actual label', fontsize =10)

plt.xlabel('Predicted label', fontsize = 10)

#Model Evaluation

print('Accuracy:', metrics.accuracy\_score(y\_test,y\_pred\_clf)) #0.8422104812223333

print('Precision:', metrics.precision\_score(y\_test,y\_pred\_clf)) #0.19920769666100735

print('Recall:', metrics.recall\_score(y\_test,y\_pred\_clf)) #0.20711974110032363

# F-1 score

Accuracy = metrics.accuracy\_score(y\_test,y\_pred\_clf)

Precision = metrics.precision\_score(y\_test,y\_pred\_clf)

Recall = metrics.recall\_score(y\_test,y\_pred\_clf)

f1\_score = 2\*((Recall\*Precision)/(Recall+Precision))

print(f1\_score) #0.20308668685994521

from sklearn.metrics import auc

fpr, tpr, thresholds = metrics.roc\_curve(y\_test, y\_pred\_clf)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color='blue', lw=2, label='SVM ROC area = %0.2f)' % roc\_auc)

plt.legend(loc="lower right")

plt.show()

#Area under Receiver Operating Curve(AUC)

fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_clf)

auc = metrics.roc\_auc\_score(y\_test, y\_pred\_clf)

plt.plot(fpr, tpr, label = 'data 1, auc ='+str(auc)) #0.5588

plt.legend(loc = 4)

plt.show()

######################################## Naive Bayes Model #######################

from sklearn.naive\_bayes import GaussianNB

NB\_model = GaussianNB().fit(X\_train,y\_train)

NB\_predictions = NB\_model.predict(X\_test)

conf\_matrix\_NB = metrics.confusion\_matrix(y\_test, NB\_predictions)

conf\_matrix\_NB

class\_names=[0,1] # name of classes

fig, ax = plt.subplots(figsize = (8,8))

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(conf\_matrix\_NB),annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix', y=1, fontsize = 10)

plt.ylabel('Actual label', fontsize =10)

plt.xlabel('Predicted label', fontsize = 10)

#Model Evaluation

print('Accuracy:', metrics.accuracy\_score(y\_test,NB\_predictions)) #0.9229187491075254

print('Precision:', metrics.precision\_score(y\_test,NB\_predictions)) #0.7134502923976608

print('Recall:', metrics.recall\_score(y\_test,NB\_predictions)) #0.3557888597258676

# F-1 score

Accuracy = metrics.accuracy\_score(y\_test,NB\_predictions)

Precision = metrics.precision\_score(y\_test,NB\_predictions)

Recall = metrics.recall\_score(y\_test,NB\_predictions)

f1\_score = 2\*((Recall\*Precision)/(Recall+Precision))

print(f1\_score) #0.4613861386138614

#Area under Receiver Operating Curve(AUC)

fpr, tpr, \_ = metrics.roc\_curve(y\_test, NB\_predictions)

auc = metrics.roc\_auc\_score(y\_test, NB\_predictions)

plt.plot(fpr, tpr, label = 'data 1, auc ='+str(auc))

plt.legend(loc = 4)

plt.show()

# ### As we got best Accuracy And preccision with Naive Bayes Model we will use this Model to predict Fare

# test data

test.describe()

test.shape

# prediction on test data using Naive Bayes model;

predicted\_fare=NB\_model.predict(test)

# Saving predicted target in test data

test['predicted\_target']=predicted\_target

test.head(10)

# saving test data in our memory

test.to\_csv("test\_predicted.csv",index=False)

## Appendix B – R Script

rm(list=ls())

setwd('C:\\Users\\Vishal Tyagi\\Desktop\\R\\projects\\Santander')

getwd()

## Read the data

train = read.csv("train.csv", header = T)

test = read.csv("test.csv", header = T)

####################################Explore the data################################

str(train)

str(test)

########################Missing Values Analysis#######################################

sum(is.na(train))

sum(is.na(test))

################### Outlier Analysis #####################################

# ## BoxPlots - Distribution and Outlier Check for train data

cnames = colnames(train[,3:4])

# # #Remove outliers using boxplot method

df = train

#train = df

#0 # #loop to remove from all variables

cnames = colnames(train[,3:202])

for(i in cnames){

print(i)

val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]

print(length(val))

train = train[which(!train[,i] %in% val),]

}

#outlier analysis in test data

cnames1 = colnames(test[,2:201])

# # #Remove outliers using boxplot method in test data

df1 = test

#test = df1

# # #loop to remove from all variables

cnames1 = colnames(test[,2:201])

for(i in cnames1){

print(i)

val = test[,i][test[,i] %in% boxplot.stats(test[,i])$out]

print(length(val))

test = test[which(!test[,i] %in% val),]

}

###############################Feature Selection ###################################

## Correlation Plot

#install.packages("corrgram")

library(corrgram)

corrgram(train[1:30,3:202], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

###########################Feature Scaling ########################################

#Normality check

qqnorm(train$var\_1)

cnames = colnames(train[,2:202])

for(i in cnames){

print(i)

train[,i] = (train[,i] - min(train[,i]))/

(max(train[,i] - min(train[,i])))

}

######################### Model Development ####################################

rmExcept("train")

#Divide data into train and test using stratified sampling method

set.seed(12234)

#install.packages('caret')

library(caret)

train.index = createDataPartition(train$target, p = .80, list = FALSE)

train = train[ train.index,]

test = train[-train.index,]

##Decision tree for classification

#install.packages('C50')

library(C50)

#Develop Model on training data

str(train$target)

train$target = as.factor(train$target)

C50\_model = C5.0(target ~., train, trials = 1, 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[,-2], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_C50 = table(test$target, C50\_Predictions)

confusionMatrix(ConfMatrix\_C50)

#False Negative rate

FNR = FN/FN+TP

#Accuracy: 90.89%

#FNR: 63.09%

#Logistic Regression

logit\_model = glm(target ~ ., data = trn, family = "binomial")

#summary of the model

summary(logit\_model)

#predict using logistic regression

logit\_Predictions = predict(logit\_model, newdata = test, type = "response")

#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

#Accuracy: 90.89

#FNR: 67.85

############################ naive Bayes ##############################

library(e1071)

#Develop model

NB\_model = naiveBayes(target ~ ., data = trn)

#predict on test cases #raw

NB\_Predictions = predict(NB\_model, test[,3:202], type = 'class')

#Look at confusion matrix

Conf\_matrix = table(observed = test[,2], predicted = NB\_Predictions)

confusionMatrix(Conf\_matrix)

#Accuracy: 92.16

#Recall: 0.93

#precision: 0.63

#specivity:0.70

2052/(2052+1169)

**References**

* Edwisor.com
* Edwisor Community.
* Google