**PROJECT REPORT**

**Santander Customer Transaction Prediction**



**- Srijan Srivastava**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Contents**

1. Problem Statement……………………………………………3
2. Business Understanding………………………………………3
3. Data understanding……………………………………………3
4. Loading Libraries and data……………………………………3
5. Data Preparation (Data Pre-Processing)………………………5
6. Data Visualization………………………………………….....6
7. Modelling and Evaluation………………………………….....9
8. Deployment…………………………………………………..15
9. Summary……………………………………………………...15
10. Code Files:

* Python code ……………………………………...16
* R code…………………………………………….19

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. **Problem Statement:**

In this project, we need to identify which customers will make a specific transaction in

the future, irrespective of the amount of money transacted.

1. **Business Understanding:**

Santander group is the multinational commercial bank based in Madrid, Spain. It is the 16th largest banking institutions in the world. (source: Wiki).

In this project, we need to identify customers who will make transactions in future. This would help the bank to take measures accordingly.

1. **Data Understanding:**

We have been provided with two datasets:

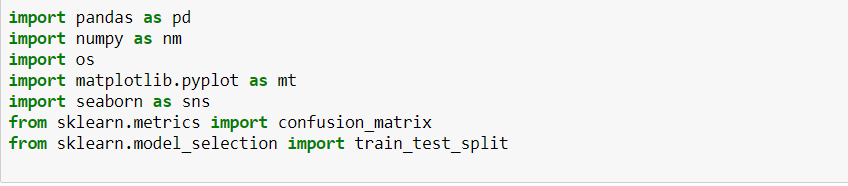
1) test.csv (200000 observations and 201 variables/features)

2) train.csv (200000 observations and 202 variables/features)

Our train data has 1 target variable by the name “target”, 1 categorical variable “ID\_code” and rest continuous variables “var\_0 to var\_199”.

1. **Loading Libraries and data:**

*Python:*



*Fig 4.1*

**Pandas**: used for creating data-frames and other data frame related things.

**Numpy**: used for basic numerical purposes.

**Os**: for basic operating system dependent functionalities like setting current working directories.

**Matplotlib.pyplot**: used for visualizations

**Seaborn**: based on matplotlib but provides attractive visualizations.

**Sklearn**: scikit learn library which allows machine learning algorithms to work on python.

**Math**: allows mathematical based functions.



*Fig 4.2*

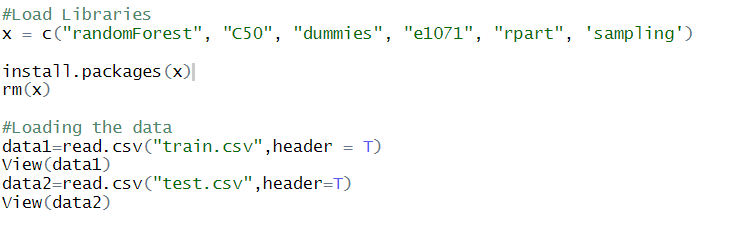
Train.csv dataset is stored in variable “data”.

Test.csv dataset is stored in variable “data\_test”

‘r’ is applied before the path in read\_csv stands for raw. Backslash with any character has a special meaning(for eg: \n means newline) . Applying ‘r’ before the path allows backslash to be treated as a normal character.

Sep=”,” is a delimeter which treats comma as the point of separation in the file.

*R:*



*Fig 4.3*

Rm(list=ls(all=T): for clearing up all the things available in Global Environment.

Setwd(): sets the path where the R notebook will be saved .

Getwd(): to see the working directory.

X=c(“…”): vector containing library names

Install.packages(x): installing all the packages.

Rm(x): removing the x variable from the global environment

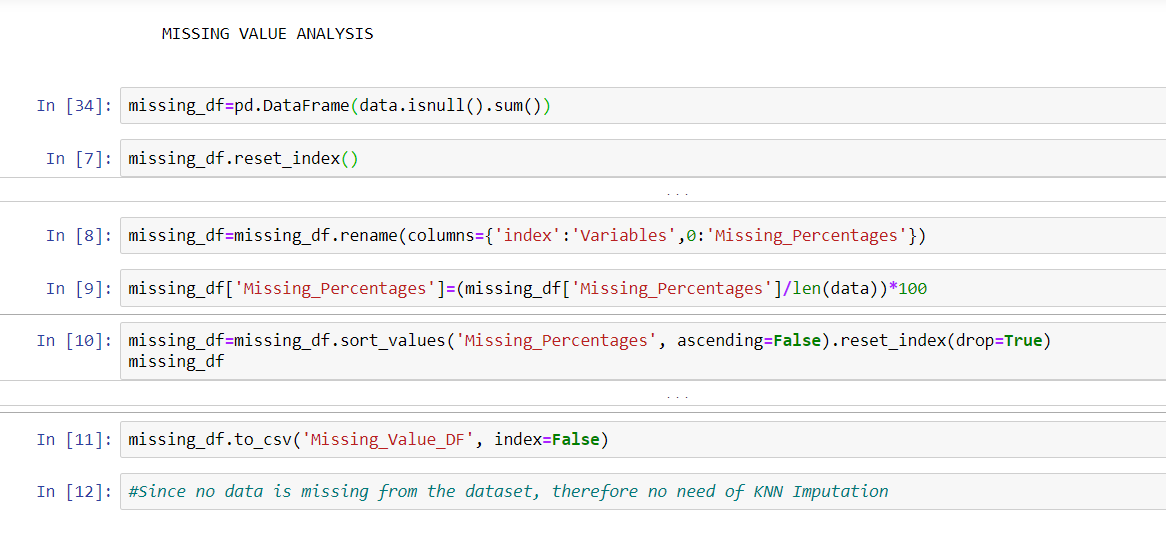
Datasets train and test are loaded in data1 and data2 variables respectively.

View(data1) and view(data2): for displaying the complete dataset.

1. **Data Preparation:**

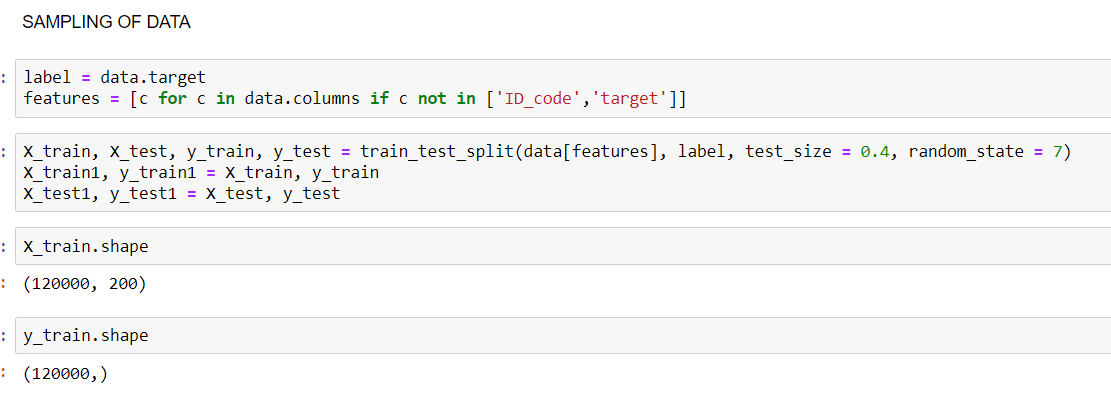
Missing values in dataset is a common phenomenon. Data may be missed while entering or due to non-availability of any particular detail or various other reasons. To handle this, we need to do Missing Value Analysis. We can either remove the observations with missing values or we can impute them using Central tendency methods(mean. median) or using distance based methods (KNN imputation), etc.

*Python:*



*Fig 5.1*

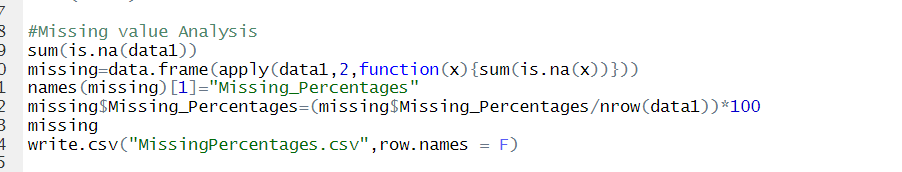
In the above code snippet, we are first finding the total number of null values. We have created a separate data frame “missing\_df”specifically for this purpose.After finding the total number of null values, we are finding what percentage of complete data is null value and store the data frame in the local storage using .to\_csv.



*Fig 5.2*

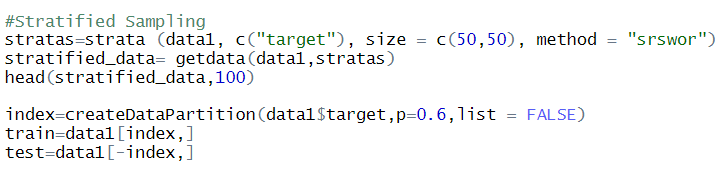
Since the data is huge with 200k observations and 200 variables, we need to divide data into train and test data. **Stratified data sampling** has been chosen since it works well with categorical data. Simple random sampling is not chosen since it is not fit to be used for categorical data.

*R:*



*Fig 5.3*

Similarly like what we did for Python Missing value analysis, we need to find the total number of null values in the complete dataset and saved in “missing” dataframe.



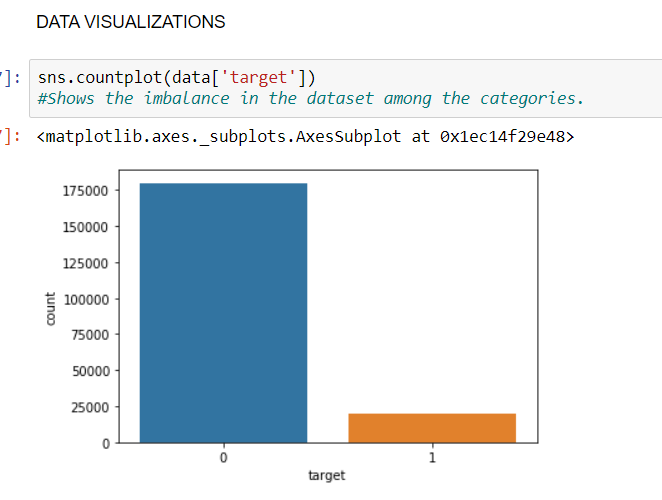
*Fig 5.4*

As in python, here also we have used stratified sampling. Stratas variable runs stratified sampling keeping “target” variable as the reference.

1. **Data Visualization:**
2. *Getting the distribution of categorical variable through bar graph:*

The purpose of this graph is to have an idea about which category in the target variable is present in majority and which in minority.

“sns” is for the seaborn library.

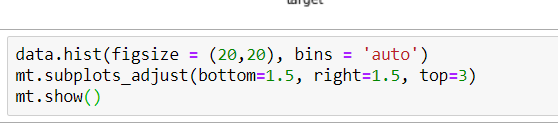


*Fig 6.1*

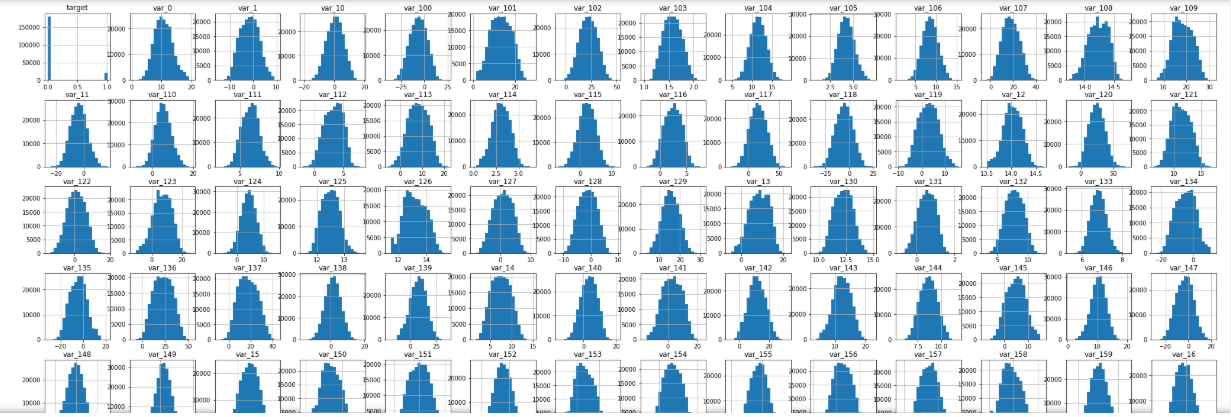
1. *Getting the overall variable distribution through histogram:*

The purpose of the below graph is to have a bird’s eye view of the distribution of all the variables present in the dataset.

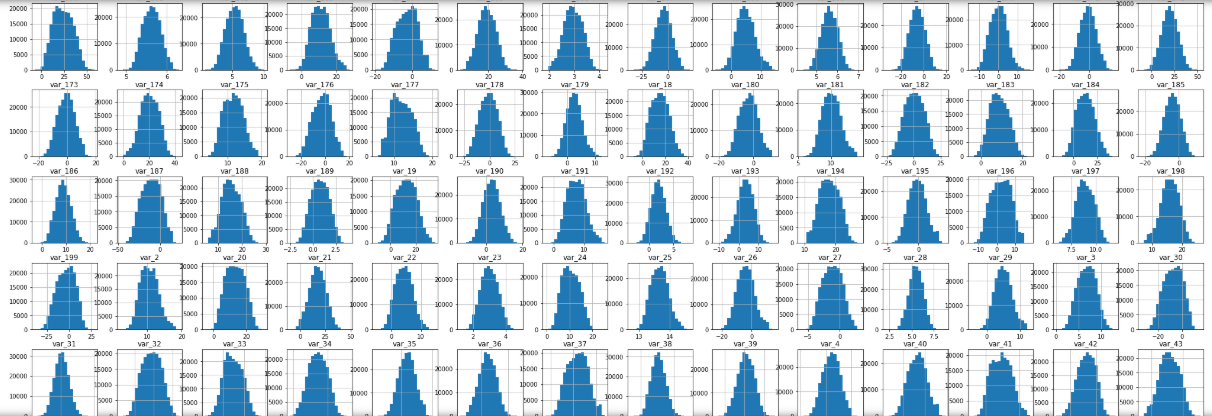
x-axis represents the value and the y-axis represents the count.



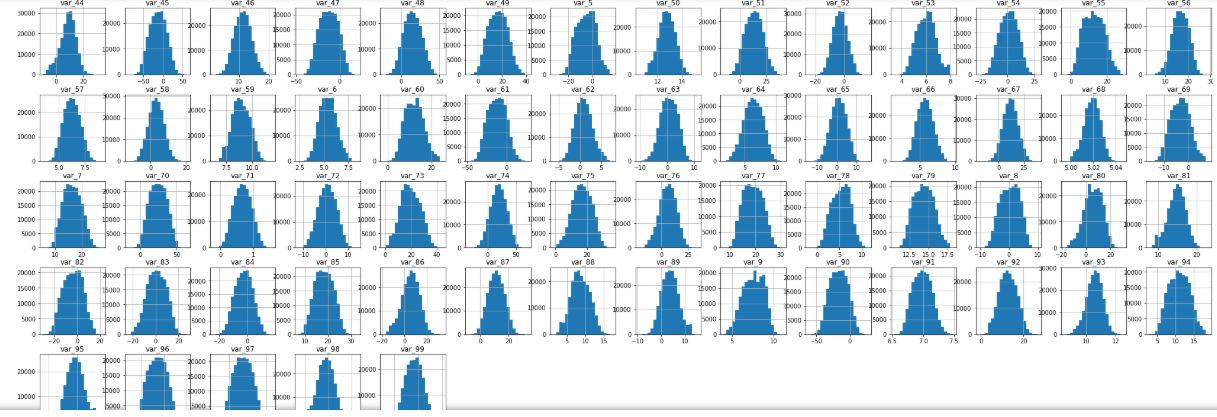
*Fig 6.2*



*Fig 6.3*



*Fig 6.4*



*Fig 6.5*

1. **Modelling and Evaluation:**

There are various modelling techniques:

*Supervised Machine Learning techniques:*

Decision tree

Random forest

KNN

Naïve Bayes

*Supervised Statistical Modelling:*

Linear regression

Logistic regression

*Unsupervised machine Learning techniques:*

Clustering

Since the dataset we are provided contains 1 target variable that is “***categorical”*** and 200 independent variables. Therefore we need to use **Supervised Machine Learning techniques**. Unsupervised ML Algorithms like clustering don’t require target variable and they work on patterns.

Also in the dataset, the target variable needs to be predicted and consists of only 0s and 1s. Due to target variable being **categorical** in nature, we need to use **Logistic Regression** technique. We won’t be using Linear regression since it woks when target variable is numerical (continuous) in nature.

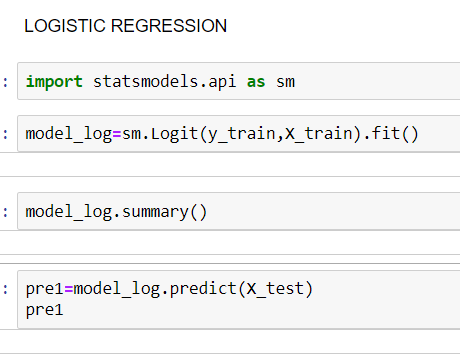
Apart from **Logistic regression, Decision tree and random forest techniques** have been used since they work well with categorical data.

KNN algorithm has not been used since it is the “*lazy algorithm*” and was taking too much time to predict the data.

For Evaluation of models, confusion metrics has been used over RMSE,MAPE and MSE because former is used for evaluating classification models and the latter is used for Regression models. Since we had classification model (target variable is categorical), therefore confusion error metrics is preferred.

1. **ALGORITHM 1:** Logistic Regression :

*Python:*



*Fig 7.1*

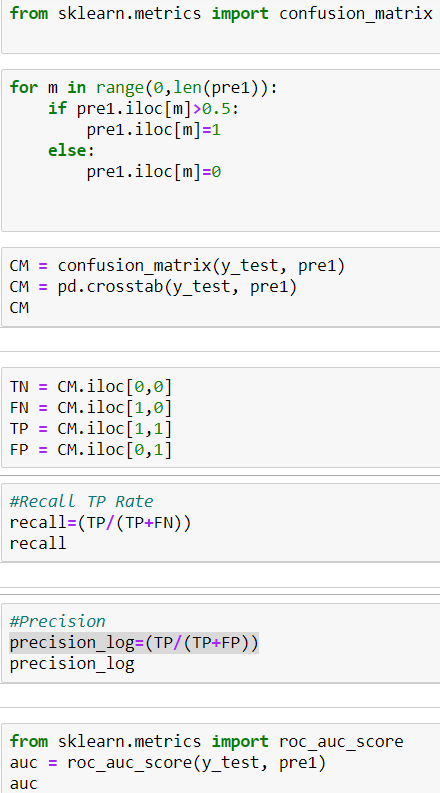
In the above code snippet, “model\_log” variable has stored the logistic model prepared by passing the target and independent variables of train dataset in the form of y\_train and X\_train repectively.

Model\_log.summary() explains the details of the model.

“Pre1” variable stores the predicted value obtained by running predict() function over the test data X\_test.

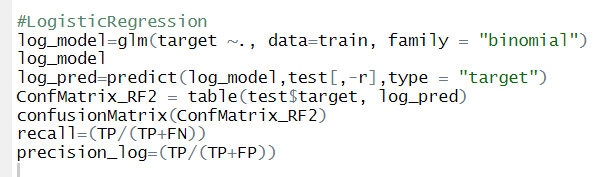
For evaluation, **Confusion Error Matrix** has been created since it works well with categorical data. MSE,RMSE and MAPE can’t be used since they are for numerical data.

From the contingency table, the values of TN, FN,TP and FP are calculated. On top of it, the values of recall, precision and auc(Area Under Curve) are calculated. These values are used to check the accuracy of the model.



*Fig 7.2*

*R:*



*Fig 7.3*

In the above code snippet, Logistic Regression has been applied over the dataset.

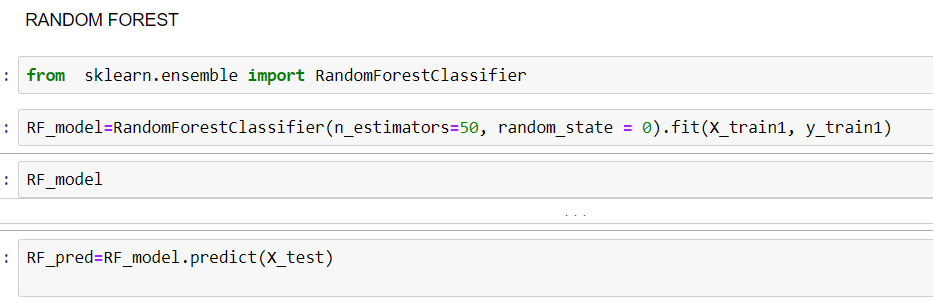
Glm() function with family=”binomial” is used for creating the logistic regression model

For evaluation, confusion error metrics has been used.

1. **ALGORITHM 2:** Random Forest

Random Forest is an ensemble of multiple Decision Trees. Therefore, it has better accuracy and removes weak learner. RF uses CART algorithm (Classification and Regression) which works on Gini index.

*Python:*

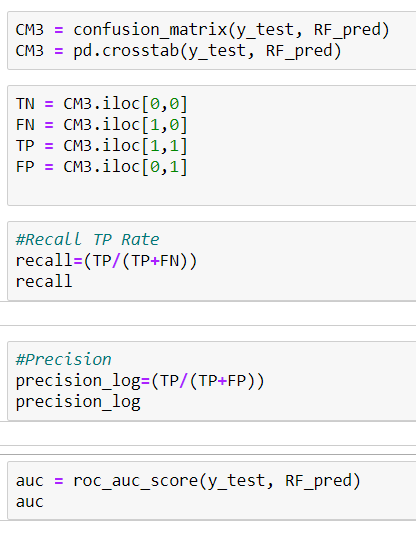


*Fig 7.4*

We have imported RandomForestClassifier function from the sklearn.ensemble package.

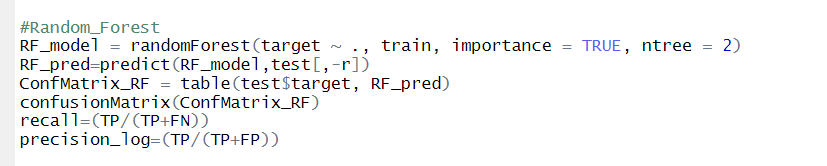
And stored the model in RF\_model variable. N\_estimators is the total number of trees to be produced. Random\_state is the number that states id of model. When we re-run the Classifier, the data won’t change if same random\_state is used. RF\_pred stores the predicted values of the target variable.

Below code snippet shows the Confusion Error Metrics for checking the accuracy of the prediction done by Random Forest. CM3 stores the confusion matrix produced from the target variable of the training dataset and the predicted target values.



*Fig 7.5*

*R:*



*Fig 7.6*

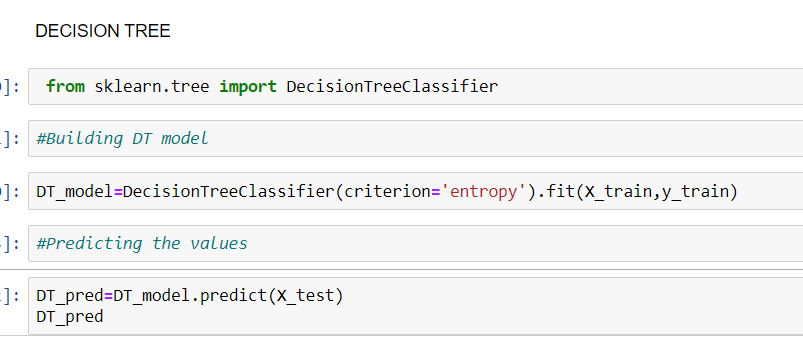
The above code depicts the Random Forest in R. randomForest() functions is used to create the model. Importance = TRUE will show the important variables. nTree count is the number of trees to be created.

For checking the accuracy, Confusion Error metrics is used.

1. **ALGORITHM 3:** Decision Tree

Decision Tree is a very useful ML algorithm for classification model. In other words, it is a flowchart of questions leading to a prediction. Here we have used C5.0 algorithm that works on information gain. Information Gain is the purity measure or certainty measure. On the other hand, Entropy is the impurity measure or the measure of uncertainty.

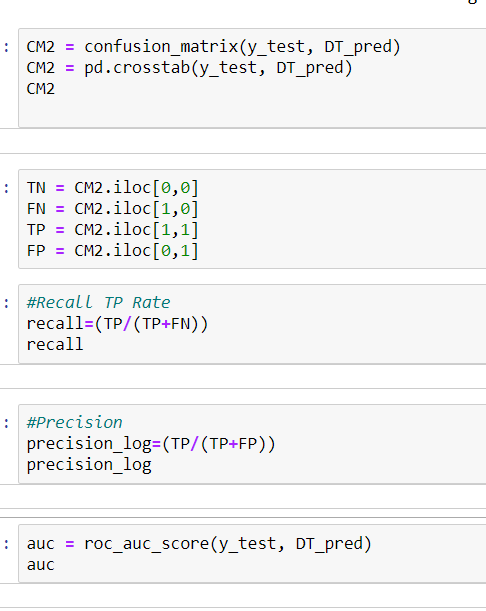
*Python:*



*Fig 7.7*

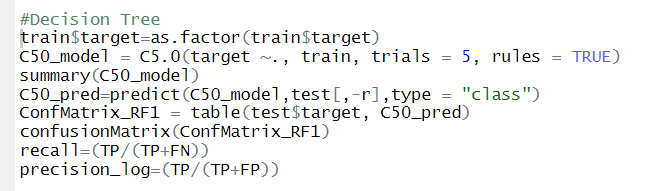
Here, from the sklearn.tree DecisionTreeClassifier function has been imported and target and independent variables are fitted over it to create the model which is stored in DT\_model.

DT\_pred stores the predicted values

 *Fig 7.8*

In the above code snippet, Confusion error metrics have been applied. Thus finding the values of recall, precision and auc.

*R:*

**

*Fig 7.9*

Since DT works on C5.0 model, C5.0( ) function has been applied to create Decision Tree. Trials count gives the number of trees to be made. Rules=True means to extract the business rules from the decision tree.C50\_pred stores the predicted values of the model.

Confusion Error metrics is saved in ConfMatrix\_RF1 variable. Recall and precision values are calculated on top of it.

1. **Deployment and Instructions to run:**

* For online deployment, we can use Orange, RevoDeployR or any other cloud software.
* We can use Rshiny library for deploying over Web framework.

*Few instructions for run:*

* Save the datasets locally and modify the current working directory commands in both R and python according to the path of the dataset .
* For running R code, clear up the space beforehand by running the following command: *rm(list=ls(all=T)).*

1. **Summary:**

As the project name suggests, Santander Customer Transaction Prediction, this project predicts the customer transaction in Santander Bank.

* The primary purpose of the project is to predict if a given customer,
* with his few details, *will make the transaction or not*.
* If the bank can predict which customer will make transaction and which will not, they can *focus more* on the former category of customers.
* Having known the transaction prediction capability for a customer, bank can take measures to retain the customers with the transaction capabilities, with some extra benefits and perks.
* Likewise, for customers who won’t make transactions, bank can take several steps to *manipulate and woo* the customers to make transactions.
* Using this project, banks can find which details of the customers are useful in finding the predictions and can omit rest of the details.
* If any detail has *strong covariance* with the prediction values, bank can monitor that detail specifically and boost the chances of transactions.

1. **Code Files:**

***Python Code File:***

 #Loading Libraries

*import pandas as pd*

*import numpy as nm*

*import os*

*import matplotlib.pyplot as mt*

*import seaborn as sns*

*from sklearn.metrics import confusion\_matrix*

*from sklearn.metrics import mean\_squared\_error*

*from sklearn import tree*

*from math import sqrt*

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

#Loading datasets

*data=pd.read\_csv(r"C:\Users\.hp\Desktop\EdWisor\Project\_1\train.csv",sep=",",encoding="ISO-8859-1")*

*data\_test=pd.read\_csv(r"C:\Users\.hp\Desktop\EdWisor\Project\_1\test.csv",sep=",",encoding="ISO-8859-1")*

data.head()

data\_test.head()

#MISSING VALUE ANALYSIS

missing\_df=pd.DataFrame(data.isnull().sum())

missing\_df.reset\_index()

missing\_df=missing\_df.rename(columns={'index':'Variables',0:'Missing\_Percentages'})

missing\_df['Missing\_Percentages']=(missing\_df['Missing\_Percentages']/len(data))\*100

missing\_df=missing\_df.sort\_values('Missing\_Percentages', ascending=False).reset\_index(drop=True)

missing\_df

missing\_df.to\_csv('Missing\_Value\_DF', index=False)

#Since no data is missing from the dataset, therefore no need of KNN Imputation

#DATA VISUALIZATIONS

sns.countplot(data['target'])

#Shows the imbalance in the dataset among the categories.

data.hist(figsize = (20,20), bins = 20)

mt.subplots\_adjust(bottom=1.5, right=1.5, top=3)

mt.show()

#SAMPLING OF DATA

label = data.target

features = [c for c in data.columns if c not in ['ID\_code','target']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data[features], label, test\_size = 0.4, random\_state = 7)

X\_train.shape

y\_train.shape

#LOGISTIC REGRESSION

import statsmodels.api as sm

model\_log=sm.Logit(y\_train,X\_train).fit()

model\_log.summary()

pre1=model\_log.predict(X\_test)

pre1

a=0

b=0

len(pre1)

for x in range(0,len(pre1)):

if pre1.iloc[x] > 0.5 :

a=a+1

else:

b=b+1

print ("The value of a:", a ,"The value of b: ", b)

from sklearn.metrics import confusion\_matrix

for m in range(0,len(pre1)):

if pre1.iloc[m]>0.5:

pre1.iloc[m]=1

else:

pre1.iloc[m]=0

CM = confusion\_matrix(y\_test, pre1)

CM = pd.crosstab(y\_test, pre1)

CM

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

#Recall TP Rate

recall=(TP/(TP+FN))

recall

#Precision

precision\_log=(TP/(TP+FP))

precision\_log

from sklearn.metrics import roc\_auc\_score

auc = roc\_auc\_score(y\_test, pre1)

auc

#DECISION TREE

from sklearn.tree import DecisionTreeClassifier

#Building DT model

DT\_model=DecisionTreeClassifier(criterion='entropy').fit(X\_train,y\_train)

#Predicting the values

DT\_pred=DT\_model.predict(X\_test)

DT\_pred

#Verifying using Error Metrics

c=0

d=0

for x in range(0,len(DT\_pred)):

if DT\_pred[x]>0.5:

DT\_pred[x]=1

c=c+1

else:

DT\_pred[x]=0

d=d+1

print("the value of c:",c,"the value of d:", d, "length of pre:", len(DT\_pred))

CM2 = confusion\_matrix(y\_test, DT\_pred)

CM2 = pd.crosstab(y\_test, DT\_pred)

CM2

TN = CM2.iloc[0,0]

FN = CM2.iloc[1,0]

TP = CM2.iloc[1,1]

FP = CM2.iloc[0,1]

#Recall TP Rate

recall=(TP/(TP+FN))

recall

#Precision

precision\_log=(TP/(TP+FP))

precision\_log

auc = roc\_auc\_score(y\_test, DT\_pred)

auc

#RANDOM FOREST

from sklearn.ensemble import RandomForestClassifier

RF\_model=RandomForestClassifier(n\_estimators=50, random\_state = 0).fit(X\_train1, y\_train1)

RF\_model

RF\_pred=RF\_model.predict(X\_test)

CM3 = confusion\_matrix(y\_test, RF\_pred)

CM3 = pd.crosstab(y\_test, RF\_pred)

TN = CM3.iloc[0,0]

FN = CM3.iloc[1,0]

TP = CM3.iloc[1,1]

FP = CM3.iloc[0,1]

#Recall TP Rate

recall=(TP/(TP+FN))

recall

#Precision

precision\_log=(TP/(TP+FP))

precision\_log

auc = roc\_auc\_score(y\_test, RF\_pred)

auc

***R Code File:***

rm(list=ls(all=T))

setwd("C:/Users/.hp/Desktop/EdWisor/Project\_1")

getwd()

#Load Libraries

x = c("ggplot2", "corrgram", "randomForest", "C50", "MASS", "rpart", "gbm", 'sampling', 'inTrees')

install.packages(x)

lapply(x, require, character.only = TRUE)

rm(x)

#Loading the data

data1=read.csv("train.csv",header = T)

View(data1)

data2=read.csv("test.csv",header=T)

View(data2)

#Missing value Analysis

sum(is.na(data1))

missing=data.frame(apply(data1,2,function(x){sum(is.na(x))}))

names(missing)[1]="Missing\_Percentages"

missing$Missing\_Percentages=(missing$Missing\_Percentages/nrow(data1))\*100

missing

write.csv("MissingPercentages.csv",row.names = F)

#Stratified Sampling

stratas=strata (data1, c("target"), size = c(50,50), method = "srswor")

stratified\_data= getdata(data1,stratas)

head(stratified\_data,100)

index=createDataPartition(data1$target,p=0.6,list = FALSE)

train=data1[index,]

test=data1[-index,]

r=c("ID\_code","target")

#LogisticRegression

log\_model=glm(target ~., data=train, family = "binomial")

log\_model

log\_pred=predict(log\_model,test[,-r],type = "target")

log\_pred=ifelse(log\_pred>0.5,1,0)

ConfMatrix\_RF2 = table(test$target, log\_pred)

confusionMatrix(ConfMatrix\_RF2)

recall=(TP/(TP+FN))

precision\_log=(TP/(TP+FP))

#Decision Tree

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

C50\_model = C5.0(target ~., train, trials = 5, rules = TRUE)

summary(C50\_model)

C50\_pred=predict(C50\_model,test[,-r],type = "class")

ConfMatrix\_RF1 = table(test$target, C50\_pred)

confusionMatrix(ConfMatrix\_RF1)

recall=(TP/(TP+FN))

precision\_log=(TP/(TP+FP))

#Random\_Forest

RF\_model = randomForest(target ~ ., train, importance = TRUE, ntree = 2)

RF\_pred=predict(RF\_model,test[,-r])

ConfMatrix\_RF = table(test$target, RF\_pred)

confusionMatrix(ConfMatrix\_RF)

recall=(TP/(TP+FN))

precision\_log=(TP/(TP+FP))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*