## horizontal line



Santander Customer Transaction Prediction

13.12.2019

**Sushant Thakur**

# 

# Table of contents

1. Introduction
2. Problem Statement
3. Data Collection
4. Data Analysis and visualisation
5. Feature Engineering
6. Model Building
7. Model Selection
8. Instructions to run project
9. Code

.

# Introduction

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 transaction in

the future, irrespective of the amount of money transacted.

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.

As from the problem statement it is confirmed that the problem that we are going to solve is binary classification problem.

In this problem we have to predict target variable which is 0 or 1?

# Data Collection

We have two dataset file which are train.csv and test.csv.

We will split our train dataset in two parts. One part will be used to train our model and one part will be used for validating the model.

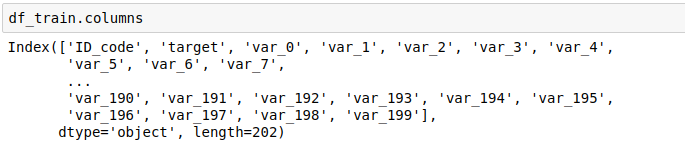
We will be using test dataset for prediction of our target variables.

# Data Analysis and Visualisation

We will start by getting shape of the train dataset.

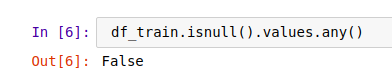
Our train dataset have 200000 rows and 202 columns attributes.

Dataset contains ID\_code as index variable , target as output variable and all other variables from var\_0 to var\_199 as input variables also called predictors



Missing value analysis

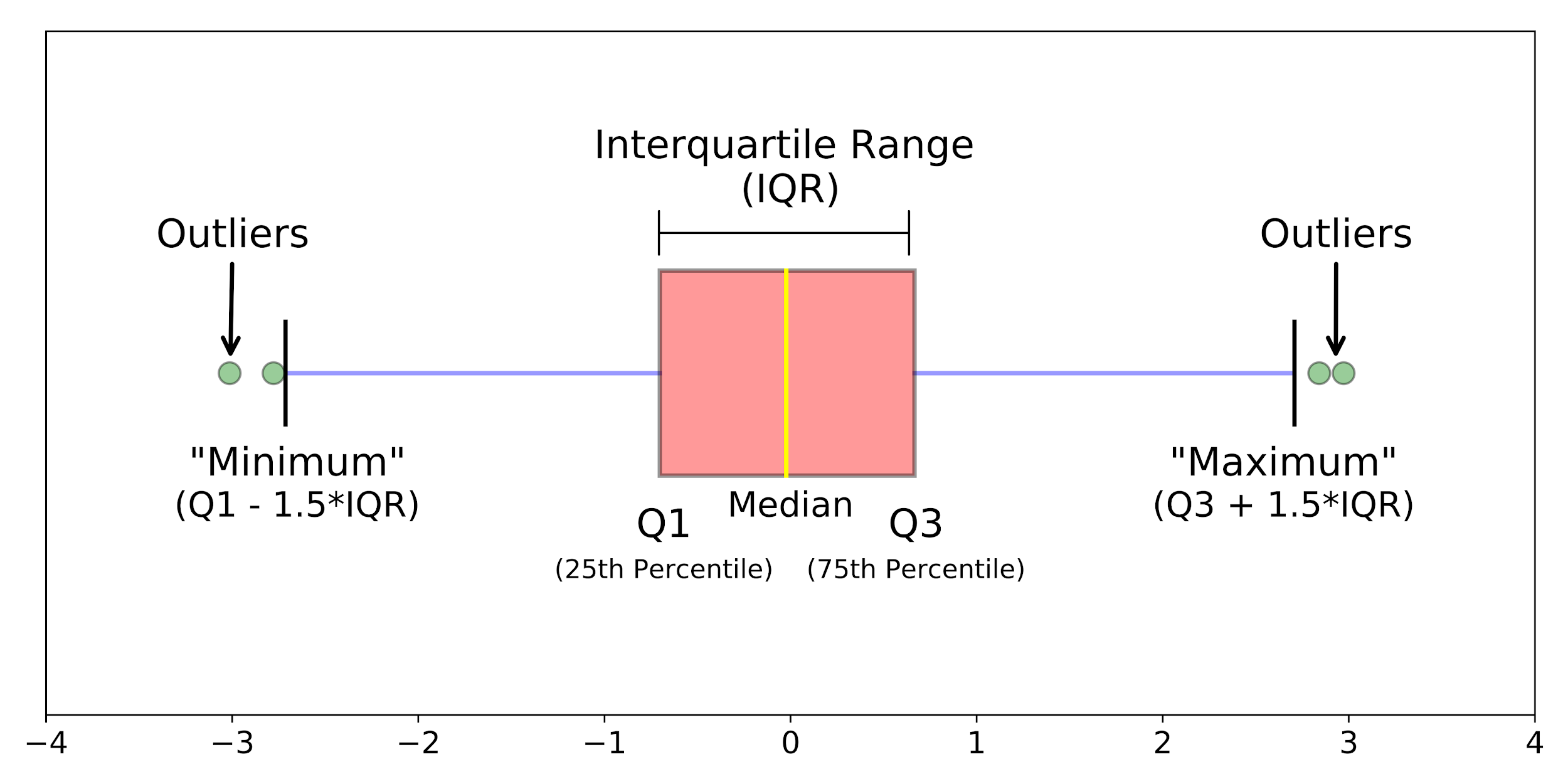
Now we will check for missing values in our dataset.



As you can see in output below we didn’t have any missing value in any column so we can go on to the next step which is outlier detection and removal.

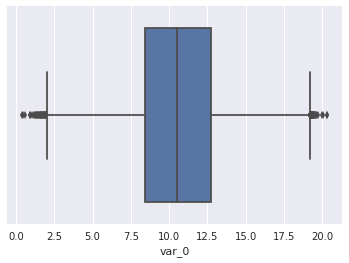
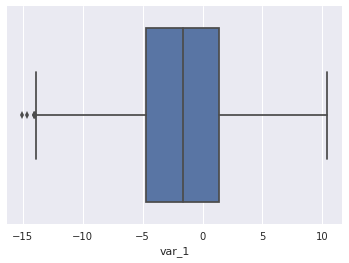
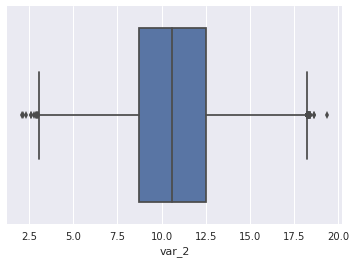
Outlier detection

We can detect outliers with the help of boxplot method



Above you can see the image of the boxplot values which are beyond the upper whiskers(maximum) and lower whiskers(minimum) are called outliers.

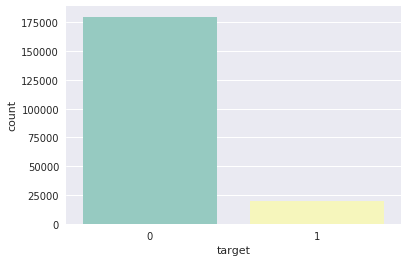
Here we plot outliers for var\_0,var\_1 and var\_2

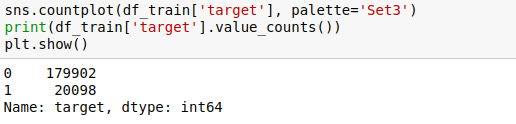


From the above boxplot we can see that there are many points which are beyond upper and lower whiskers but we can also see that the point in all the column lie between -50 and 50.Even though outliers exists in above boxplot we can say these outliers doesn’t seem to be strange so we can take either decision to remove or keep the outliers.

Class Analysis

As this problem is classification problem it is important to see no.,counts and contribution of these classes in dataset.



From the Above barplot we can see we have two classes 0 and 1. In addition to that we can clearly see the dataset is highly imbalanced dataset

We have very less number of class 1 variable as compared to class 0 variable.

Solution for Imbalanced dataset

Oversampling

Undersampling

Model hyperparameter.

We cannot afford go for oversampling as it will shoot no. of rows more than 400000 with 200 column which will increase our complexity and also we cannot go for undersampling as there are less no. of custom to make transaction and balancing 0 and 1 will surely loose some pattern as we will loose lots of data.

So we will choose model hyperparameters class\_weight=’balanced’ provided by scikitlearn which will fulfill our needs.

Performance metrics

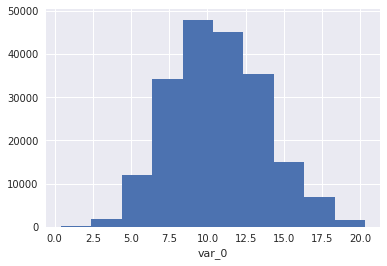
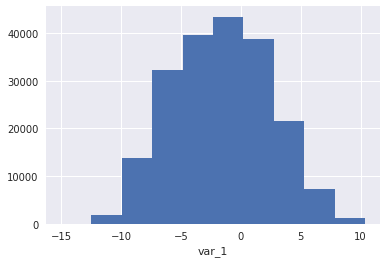
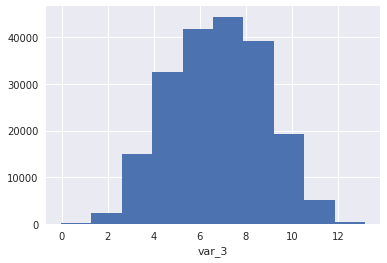
Selection of performance metrics

we cannot use accuracy because a random or a dumb model which returns 1 almost all the times can also have 90% accuracy if all the queries which are made includes 90% of class 1 input varibles

**As this is classification problem with imbalanced dataset we will be using confusion metrics,precision,recall,Roc curve and auc score.**

Univariate analysis

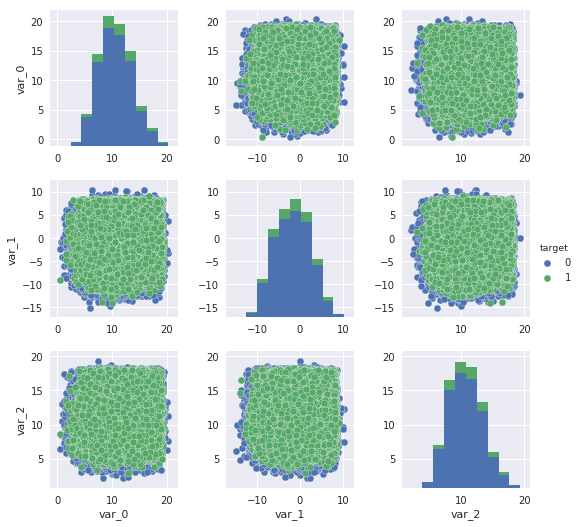
We will be using histogram from univariate analysis.



As in above histograms for var\_0,var\_1 ,var\_3 and all others have similar distributions they all are almost same there is not much difference in their distributions.

Bi-variate analysis

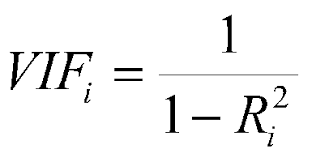
We will be using pairplot for performing bi-variate analysis.



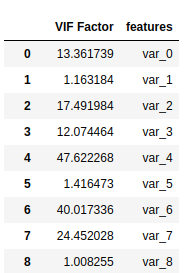
In the above pair plot we can see there is classes are somehow separable for these variables and hence these variables can be used as classifier.

Multivariate analysis

We don’t want our model to have multicollinearity so we would like to use variable inflation factor for multicollinearity detection.



Above is the formulae for VIF detection. We will use above formulae for detecting VIF for every variable with respect to other variables

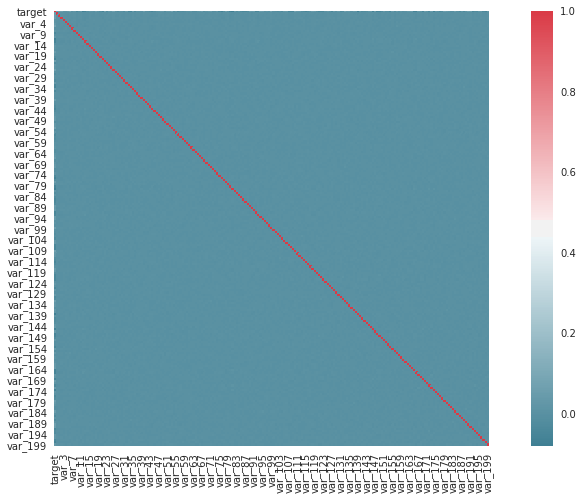


In the above sample image we can see different VIF factors for a variable.

And we can choose a threshold for the VIF that the variable we want to keep or not.

Correlation Matrix

Now we will see correlation between pairs to identify if two variables are carrying same values or not.

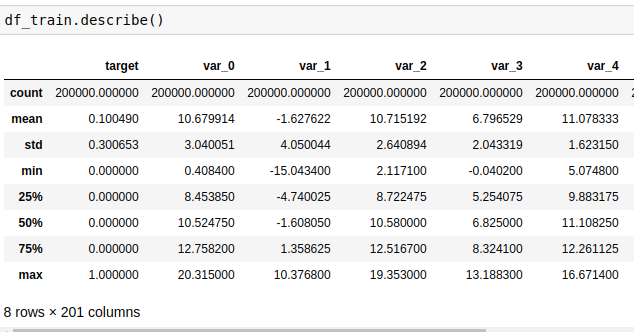


From the above correlation matrix diagram we can see that there is no correlation between variables at all, no two variables are carrying same information.

So we can proceed with our next steps.

Statistical Analysis

Getting some of the stats will help to build our model with great insights.

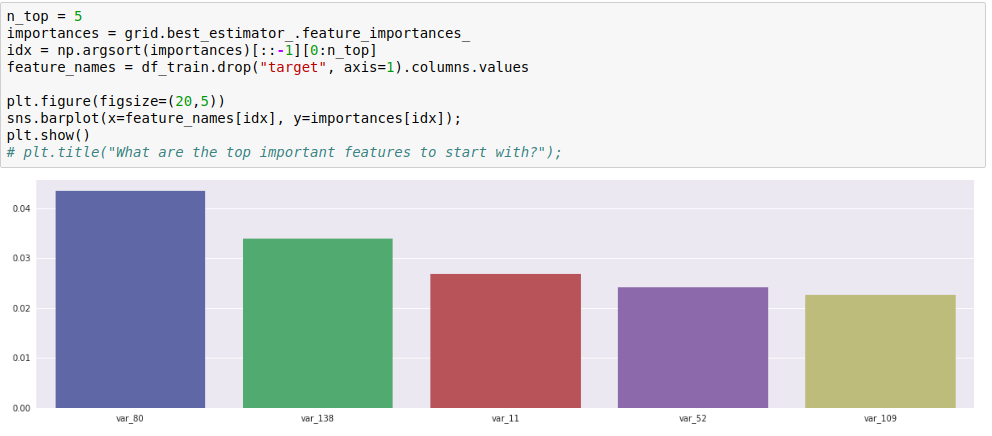


Here in the above table we can see mostly all the values in variable for particular cell lies between -50 to 50 so it’s not compulsion to do feature scaling for our variables.

Feature Importance

From some important variables which will help us to build our model with most important features and great insights

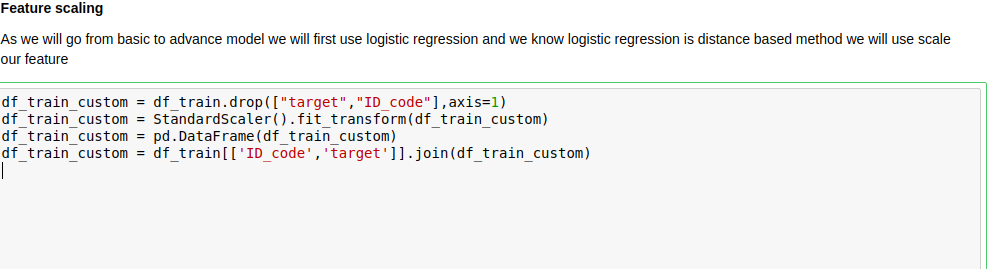
We will use random forest model to get feature importance and our first roc score on model without much hyperparameter tuning.



From the above model the **roc\_auc score that we have is 50** and there some important variables are var\_80,var\_138,var\_11,var\_52,var\_109.

Feature Scaling

Before building our model we can scale our features so that our feature are in -1 to 1 so that if we will use logistic regression this will help model to build more accurate results and model feature that are having slightly greater or smaller value will not dominate



Splitting dataset

Before model building we would like to split our dataset so that we can compare the test score of all the models this will help us to choose best models from the models that we will built.

We will be going to split our dataset in 70:30 ratio. 70% of data we will be using for training and 30 % of data we will be using for testing.

Model building

In model building approach we will be going from easiest linear model logistic regression to advance model like random forest.

**1.Logistic regression.**

Logistic regression is a linear model which is used for classification problems.

And we will be going to feed our model with scaled dataset instead of regular one as we know Logistic regression is distance based approach.

**Hyperparameters tunning**

**1.penalty:**we can use l1 or l2 regularisation which will help our model so that it does not overfit .

**2.C:**Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

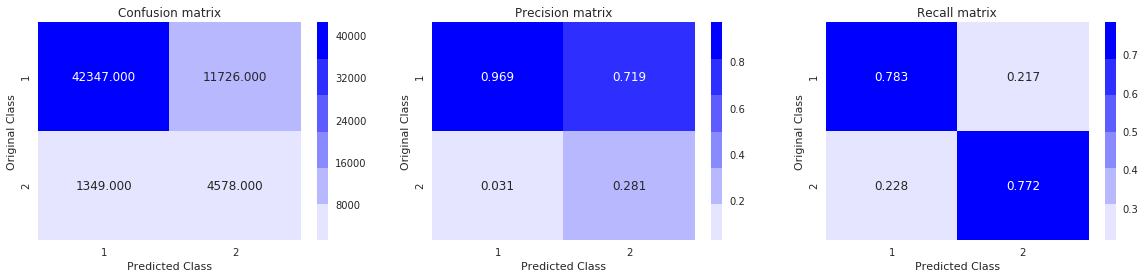
**3.class\_weight:**The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y)).

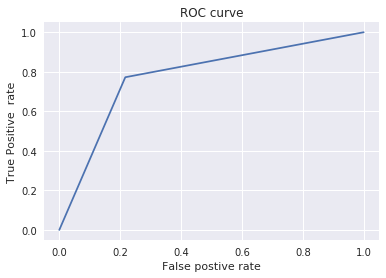
We will run our model on different C scores in for loop and then select optimal C for our model.

After running model on different optimal C scores we will have auc\_score,confusion matrix,precision matrix and recall matrix.

**Train auc\_score:0.78**

**Test auc\_score:0.77**





**2.Decision Trees.**

Decision Tree Classifier is a simple and widely used classification technique. It applies a straight forward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record.

**Hyperparameters tunning**

**As there are lots of hyperparameters and hence we will use GridSearchCV for selecting best parameters.**

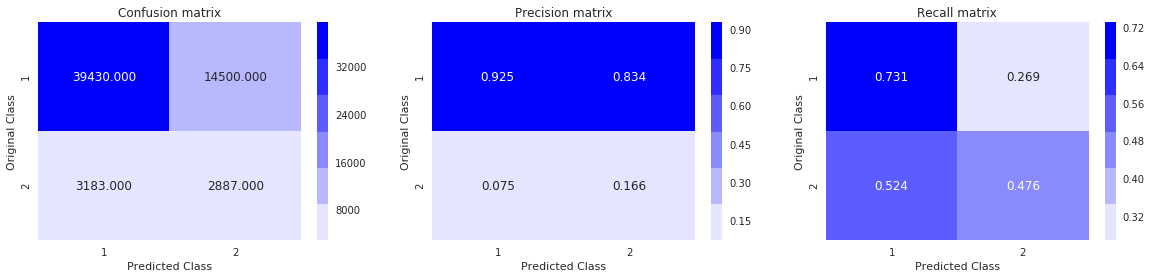
**1.max\_depth:**The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

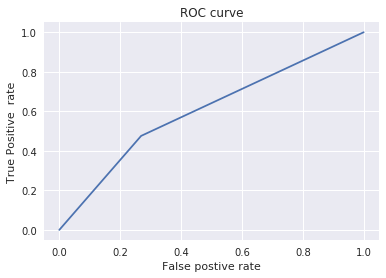
**2.criterion:**The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

**3.class\_weight:**The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y)).

**Train auc\_score:0.61**

**Test auc\_score:0.60**





**2.Random Forest.**

Random Forest Classifier is ensemble algorithm. In next one or two posts we shall explore such algorithms. Ensembled algorithms are those which combines more than one algorithms of same or different kind for classifying objects.

**Hyperparameters tunning**

**As there are lots of hyperparameters and hence we will use GridSearchCV for selecting best parameters.**

**1.n\_estimators:**The number of trees in the forest..

**2.min\_samples\_leaf:**The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min\_samples\_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

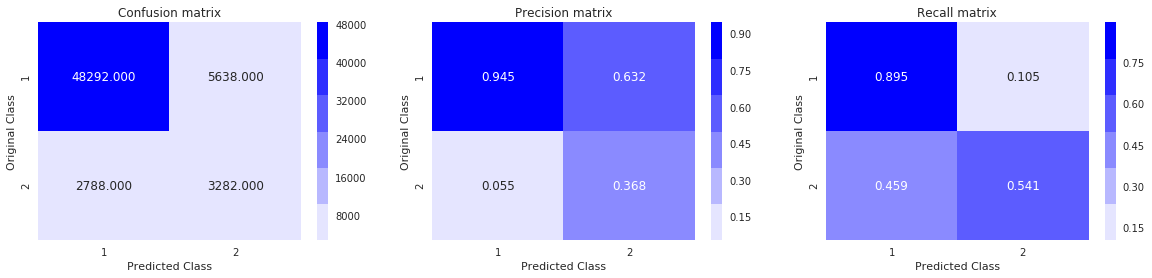
**3.max\_depth:**The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

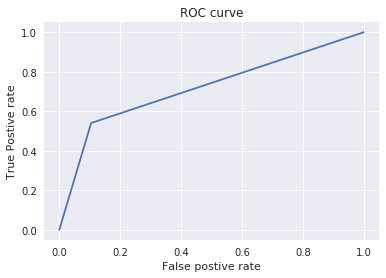
**3.bootstrap:**Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

**3.class\_weight:**The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y)).

**Train auc\_score:0.88**

**Test auc\_score:0.71**





Model Selection

In model selection approach we will select the model on the basis of speed,auc\_score,precision matrix and recall matrix.

**On above 3 matrix and test auc\_score we will choose Logistic regression for our prediction**

**As Logistic regression have**

**Test auc\_score:0.77.**

Instructions to run project

**1.Python**

1.Make sure all the files are in same directory like py or ipynb,train.csv,test.csv else you will have to change the directory in code .

2.The file can be best viewed in jupyter notebook.I have made the project step by step including some great visualisations,missing value analysis, outlier analysis and model building.

3.If you have anaconda install you can just download the py from jupyter notebook and can run from the terminal only but it will be not as interactive as jupyter notebook.

4.use python3 <file\_name> to run the project.

**2.R**

1.First you will have to install R and R studio.The file can be best viewed in R studio,you can just uncomment all the install packages so that all the packages which are used in project is installed and select all the statement and run the project will run successfully.

2.Make sure to change the directory as per file(where train and test file is located else you will get error).

3.The code can be more effectively run on the r studio because it has some visualisations,graphs and histograms.

4.For running Rscript from terminal you can type Rscript <file\_name>

Business Understanding and project application

From this project we can identify which customers are going to do transactions according to this data we can have several benefits like.

1.We can send the campaigns and emailers about specific transaction to specific customer,we don’t have to send offers to all the customers. So we can be specific on our marketing strategies.

2.We can also have some forecast about how many customers are going to turn on.

Code

# coding: utf-8

# <h2>Santander Customer Transaction Prediction</h2>

# <br>

# <h4>Background -</h4>

# <p>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?

# </p>

# <h4>Problem Statement -</h4>

# <p>In this challenge, we need to identify which customers will make a transaction in

# the future, irrespective of the amount of money transacted.<p>

# <p>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.<p>

# <h4> As from the problem statement it is confirmed that the problem that we are going to solve is binary classification problem</h4>

# <p>In this problem we have to predict target variable which is 0 or 1?</p>

# In[1]:

#importing all the libraries

import pandas as pd #for dataframe manipulation

import seaborn as sns #written on top of matplotlib for data visualization

import numpy as np #for Numerical computing

import random

from math import radians, cos, sin, asin, sqrt

from sklearn.model\_selection import train\_test\_split,GridSearchCV #fo gridsearch and train-test split

import warnings

warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt #for visualization

sns.set()

random.seed(113)

from tqdm import tqdm

from sklearn.preprocessing import StandardScaler

#slearn for machine learning algorithms

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import make\_scorer

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from imblearn.under\_sampling import NearMiss #for performing under-sampling based on NearMiss methods.

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_curve

from sklearn.tree import DecisionTreeClassifier

from scipy.stats import randint

from sklearn.ensemble import RandomForestClassifier

# <h3>Data Collection</h3>

# In[2]:

df\_train = pd.read\_csv("train.csv");

# In[3]:

df\_train.head()

# <h3>Data Exploration and visualisation</h3>

# In[4]:

print(df\_train.shape) #Let us check the shape of dataset

# In[5]:

"""

We have 200000 rows and 202 columns in dataset

"""

#Let us have some information about the dataframe like null values and types

df\_train.info()

# In[6]:

df\_train.count()

# In[7]:

df\_train.columns

# <h4>Now we will check for missing values </h4>

# In[8]:

df\_train.isnull().values.any()

#

# As we can see we don't have any null values

# In[9]:

df\_train\_custom = df\_train.drop(["target","ID\_code"],axis=1)

# In[10]:

for i in df\_train\_custom.columns:

ax = sns.boxplot(x=df\_train\_custom[i])

plt.xlabel(i)

plt.show()

# In[11]:

# # columns = df\_train.columns

# for i in tqdm(reversed(range(202))):

# # print( df\_train.columns[i])

# if(df\_train.columns[i] == "target" or df\_train.columns[i] == "ID\_code" ):

# print(i)

# else:

# print("operating on {}".format(df\_train.columns[i]))

# q75,q25 = np.percentile(df\_train.loc[:,df\_train.columns[i]],[75,25])

# iqr = q75-q25

# print(i)

# min = q25 - (iqr\*1.5)

# max = q75 + (iqr\*1.5)

# print(df\_train.shape)

# df\_train = df\_train[~(df\_train[df\_train.columns[i]] < min)]

# df\_train = df\_train[~(df\_train[df\_train.columns[i]] > max)]

# df =df.drop(df[df.loc[:,i] < min].index)

# df = df.drop(df[df.loc[:,i] > max].index)

# In[12]:

df\_train.shape

# In[13]:

# df\_train.to\_csv(r'df\_train\_removed\_outlier.csv',index=False)

# In[14]:

# df\_train = pd.read\_csv('df\_train\_removed\_outlier.csv')

# In[15]:

sns.countplot(df\_train['target'], palette='Set3')

print(df\_train['target'].value\_counts())

plt.show()

# <h4>As from the diagram we can see that the data is highly imbalanced </h4>

# <p>

# we have several ways to tackle the imbalanced dataset like

# </p>

# <li>Oversampling</li>

# <li>Undersampling</li>

# <li>Or we can use hyperparameter tunning to tune the model</li>

#

# <p>We have huge amount of data so we cannot o for oversampling hence we can undersample the data to make our model training faster

# </p>

#

# <br>

# <br>

# <h3>Selection of performance metrics</h3>

# <p>we cannot use accuracy because a random or a dumb model which returns 1 almost all the times can also have 90% accuracy if all the queries which are made includes 90% of class 1 input varibles

# <p>As this is classification problem with imbalanced dataset we will be using confusion metrics,precision,recall,Roc and auc curve

# </p>

# <h4>Univariate Analysis</h4>

# In[16]:

#Now we will check the distributions of different variables

# ax = sns.distplot(df['temp'])

ax = plt.hist(df\_train['var\_0'])

plt.xlabel('var\_0')

plt.show()

# In[17]:

# ax = sns.distplot(df['temp'])

ax = plt.hist(df\_train['var\_1'])

plt.xlabel('var\_1')

plt.show()

# In[18]:

# ax = sns.distplot(df['temp'])

ax = plt.hist(df\_train['var\_3'])

plt.xlabel('var\_3')

plt.show()

# In[19]:

g = sns.pairplot(df\_train,hue="target", vars=["var\_0", "var\_1","var\_2"])

plt.show()

# In[20]:

df\_train.describe()

# <h4> Feature Correlation</h4>

#

# In[21]:

"""

Also displaying correlation plot to detect the collinearity in data

"""

f, ax = plt.subplots(figsize=(15, 8))

corr =df\_train.corr()

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()

# <h4>Multicollinearity detection</h4>

# As from this correlation plot we can say that collinarity beteen variables are very small and hence correlation is not our problem.Let's check for multicollinearity

# <p>For multicollinearity we can check for VIF (variable inflation factor)<p>

# In[22]:

# vif = pd.DataFrame()

# df\_train\_custom = df\_train.drop(["target","ID\_code"],axis=1)

# df\_train\_custom.head()

# vif["VIF Factor"] = [variance\_inflation\_factor(df\_train\_custom.values, i) for i in range(df\_train\_custom.shape[1])]

# vif["features"] = df\_train\_custom.columns

# In[23]:

# vif.head(200)

# As from the above data we can see we don't have huge multicollinearirt between variables that we have randomly selected

# <h4>Feature scaling</h4>

# <p>As we will go from basic to advance model we will first use logistic regression and we know logistic regression

# is distance based method we will use scale our feature </p>

# In[24]:

df\_train\_custom = df\_train.drop(["target","ID\_code"],axis=1)

df\_train\_custom = StandardScaler().fit\_transform(df\_train\_custom)

df\_train\_custom = pd.DataFrame(df\_train\_custom)

df\_train\_custom = df\_train[['ID\_code','target']].join(df\_train\_custom)

# In[25]:

df\_train\_custom.head()

# <h4>Feature Importance</h4>

# In[26]:

parameters = {'min\_samples\_leaf': [20, 25]}

forest = RandomForestClassifier(max\_depth=15, n\_estimators=15)

grid = GridSearchCV(forest, parameters, cv=3, n\_jobs=-1, verbose=2, scoring=make\_scorer(roc\_auc\_score))

# In[27]:

grid.fit(df\_train.drop(["target","ID\_code"], axis=1).values, df\_train.target.values)

# In[28]:

grid.best\_score\_

# In[29]:

n\_top = 5

importances = grid.best\_estimator\_.feature\_importances\_

idx = np.argsort(importances)[::-1][0:n\_top]

feature\_names = df\_train.drop("target", axis=1).columns.values

plt.figure(figsize=(20,5))

sns.barplot(x=feature\_names[idx], y=importances[idx]);

plt.show()

# plt.title("What are the top important features to start with?");

# In[30]:

# fig, ax = plt.subplots(n\_top,2,figsize=(20,5\*n\_top))

# for n in range(n\_top):

# sns.distplot(train.loc[train.target==0, feature\_names[idx][n]], ax=ax[n,0], color="Orange", norm\_hist=True)

# sns.distplot(train.loc[train.target==1, feature\_names[idx][n]], ax=ax[n,0], color="Red", norm\_hist=True)

# sns.distplot(test.loc[:, feature\_names[idx][n]], ax=ax[n,1], color="Mediumseagreen", norm\_hist=True)

# ax[n,0].set\_title("Train {}".format(feature\_names[idx][n]))

# ax[n,1].set\_title("Test {}".format(feature\_names[idx][n]))

# ax[n,0].set\_xlabel("")

# ax[n,1].set\_xlabel("")

# In[31]:

train, test = train\_test\_split(df\_train\_custom, test\_size=0.3)

# In[32]:

train.shape

# In[33]:

test.shape

# In[34]:

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

x\_test = test.drop(["target","ID\_code"],axis=1)

y\_train = train[['target']]

y\_test = test[['target']]

# In[35]:

# This function plots the confusion matrices given y\_i, y\_i\_hat.

def plot\_confusion\_matrix(test\_y, predict\_y):

C = confusion\_matrix(test\_y, predict\_y)

# C = 9,9 matrix, each cell (i,j) represents number of points of class i are predicted class j

A =(((C.T)/(C.sum(axis=1))).T)

#divid each element of the confusion matrix with the sum of elements in that column

# C = [[1, 2],

# [3, 4]]

# C.T = [[1, 3],

# [2, 4]]

# C.sum(axis = 1) axis=0 corresonds to columns and axis=1 corresponds to rows in two diamensional array

# C.sum(axix =1) = [[3, 7]]

# ((C.T)/(C.sum(axis=1))) = [[1/3, 3/7]

# [2/3, 4/7]]

# ((C.T)/(C.sum(axis=1))).T = [[1/3, 2/3]

# [3/7, 4/7]]

# sum of row elements = 1

B =(C/C.sum(axis=0))

#divid each element of the confusion matrix with the sum of elements in that row

# C = [[1, 2],

# [3, 4]]

# C.sum(axis = 0) axis=0 corresonds to columns and axis=1 corresponds to rows in two diamensional array

# C.sum(axix =0) = [[4, 6]]

# (C/C.sum(axis=0)) = [[1/4, 2/6],

# [3/4, 4/6]]

plt.figure(figsize=(20,4))

labels = [1,2]

# representing A in heatmap format

cmap=sns.light\_palette("blue")

plt.subplot(1, 3, 1)

sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted Class')

plt.ylabel('Original Class')

plt.title("Confusion matrix")

plt.subplot(1, 3, 2)

sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted Class')

plt.ylabel('Original Class')

plt.title("Precision matrix")

plt.subplot(1, 3, 3)

# representing B in heatmap format

sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)

plt.xlabel('Predicted Class')

plt.ylabel('Original Class')

plt.title("Recall matrix")

plt.show()

# In[36]:

alpha = [10 \*\* x for x in range(-5, 2)]

tr\_scores = []

cv\_scores = []

# y\_train\_predict

# y\_test\_predict

for i in alpha:

lr = LogisticRegression(penalty='l2',C=i,class\_weight='balanced')

lr.fit(x\_train,y\_train)

y\_train\_predict = lr.predict\_proba(x\_train)

y\_test\_predict = lr.predict\_proba(x\_test)

print(y\_train\_predict.shape)

print(y\_train.shape)

tr\_scores.append(roc\_auc\_score(y\_train,np.argmax(y\_train\_predict,axis=1)))

cv\_scores.append(roc\_auc\_score(y\_test,np.argmax(y\_test\_predict,axis=1)))

# In[37]:

optimal = alpha[cv\_scores.index(np.max(cv\_scores))]

# In[38]:

lr = LogisticRegression(penalty='l2',C=optimal,class\_weight='balanced')

lr.fit(x\_train,y\_train)

y\_train\_predict = lr.predict\_proba(x\_train)

y\_test\_predict = lr.predict\_proba(x\_test)

# print(y\_train\_predict.shape)

# print(y\_train.shape)

print(roc\_auc\_score(y\_train,np.argmax(y\_train\_predict,axis=1)))

print(roc\_auc\_score(y\_test,np.argmax(y\_test\_predict,axis=1)))

plot\_confusion\_matrix(y\_test, np.argmax(y\_test\_predict,axis=1))

fpr, tpr, thresholds = roc\_curve(y\_test, np.argmax(y\_test\_predict,axis=1))

# In[39]:

plt.plot(fpr, tpr, label='ROC curve')

plt.xlabel("False postive rate")

plt.ylabel("True Positive rate")

plt.title("ROC curve")

plt.show()

# In[40]:

train,test = train\_test\_split(df\_train, test\_size=0.3)

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

y\_train = train[['target']]

x\_test = test.drop(["target","ID\_code"],axis=1)

y\_test = test[['target']]

# In[41]:

x\_train.shape

# In[42]:

x\_test.shape

# In[43]:

# Creating the hyperparameter grid

param\_dist = {"max\_depth": [3,4,5],

"criterion": ["entropy"]}

# Instantiating Decision Tree classifier

tree = DecisionTreeClassifier(class\_weight='balanced')

tree\_cv = GridSearchCV(tree, param\_dist)

tree\_cv.fit(x\_train, y\_train)

y\_train\_predict = tree\_cv.predict\_proba(x\_train)

y\_test\_predict = tree\_cv.predict\_proba(x\_test)

print(roc\_auc\_score(y\_train,np.argmax(y\_train\_predict,axis=1)))

print(roc\_auc\_score(y\_test,np.argmax(y\_test\_predict,axis=1)))

plot\_confusion\_matrix(y\_test, np.argmax(y\_test\_predict,axis=1))

fpr, tpr, thresholds = roc\_curve(y\_test, np.argmax(y\_test\_predict,axis=1))

# In[44]:

plt.plot(fpr, tpr, label='ROC curve')

plt.xlabel("False postive rate")

plt.ylabel("True Positive rate")

plt.title("ROC curve")

plt.show()

# In[45]:

parameters = {'min\_samples\_leaf': [100,200],'max\_depth': [100], 'bootstrap': [True]}

forest = RandomForestClassifier(max\_depth=15, n\_estimators=100,class\_weight="balanced")

grid = GridSearchCV(forest, parameters, n\_jobs=-1)

grid.fit(x\_train,y\_train)

y\_train\_predict = grid.predict\_proba(x\_train)

y\_test\_predict = grid.predict\_proba(x\_test)

print(roc\_auc\_score(y\_train,np.argmax(y\_train\_predict,axis=1)))

print(roc\_auc\_score(y\_test,np.argmax(y\_test\_predict,axis=1)))

plot\_confusion\_matrix(y\_test, np.argmax(y\_test\_predict,axis=1))

fpr, tpr, thresholds = roc\_curve(y\_test, np.argmax(y\_test\_predict,axis=1))

# In[46]:

plt.plot(fpr, tpr, label='ROC curve')

plt.xlabel("False postive rate")

plt.ylabel("True Postive rate")

plt.title("ROC curve")

plt.show()