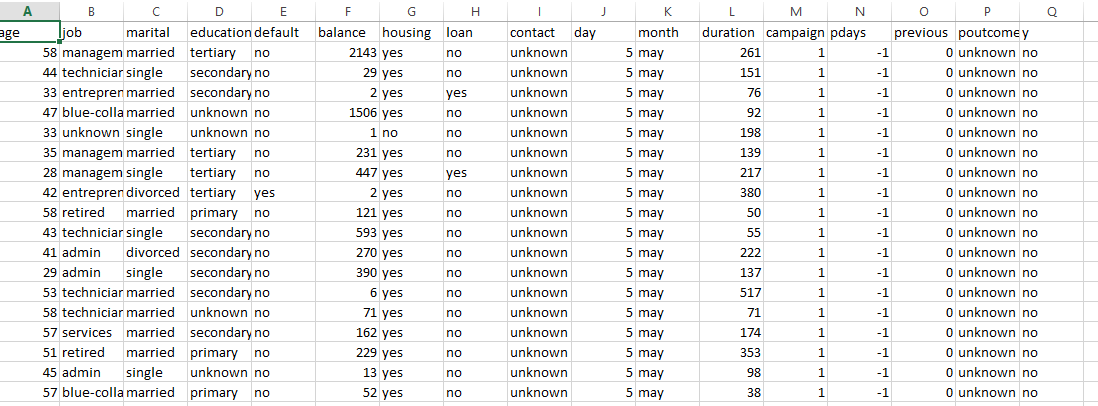
LOGISTIC REGRESSION

Datasets: bank.csv

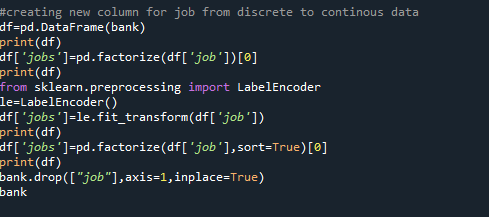
Initally dataset are improper separated by semi colon then changed into columns.



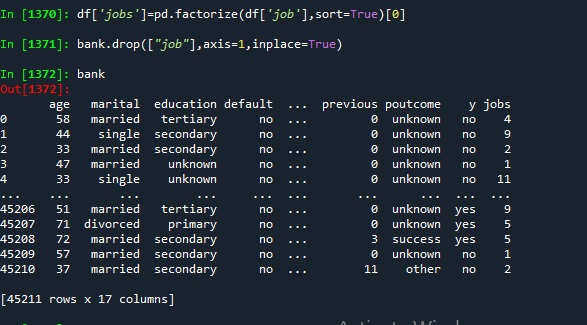
Here, We can see that all data is Discrete We should convert into numerical data (Continuous data)

Using Label Encoding created a data for all columns:

JOB:

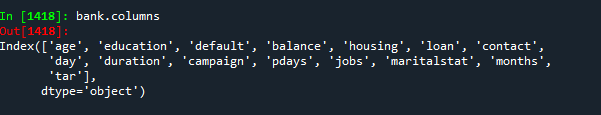


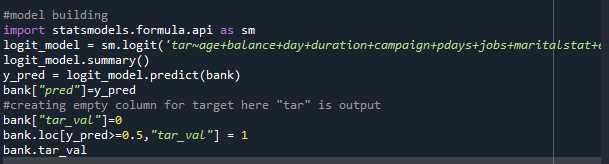
Output:



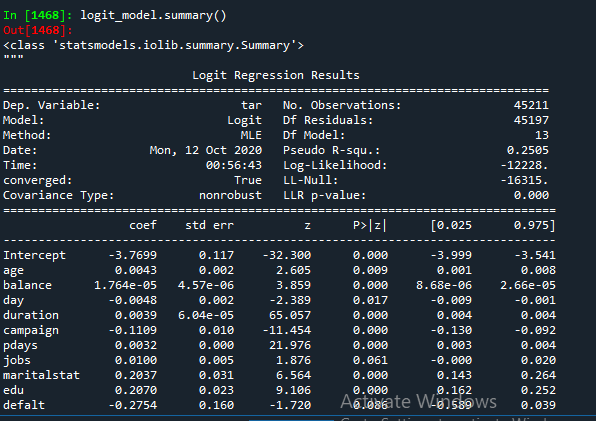
Certainly implementing this technique for all columns:

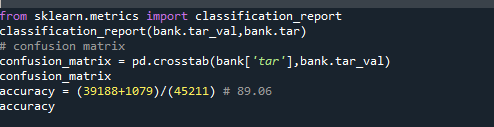
Recreating all column with new variable:



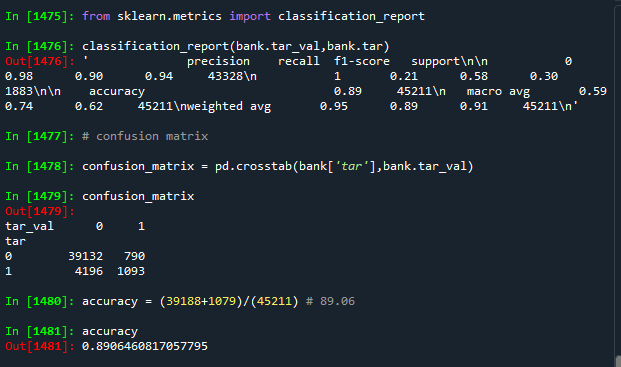
MODEL BUILDING:  


Summary:

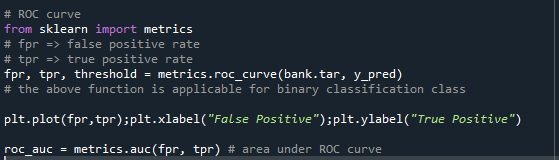


CREATING CONFUSION MATRIX:  


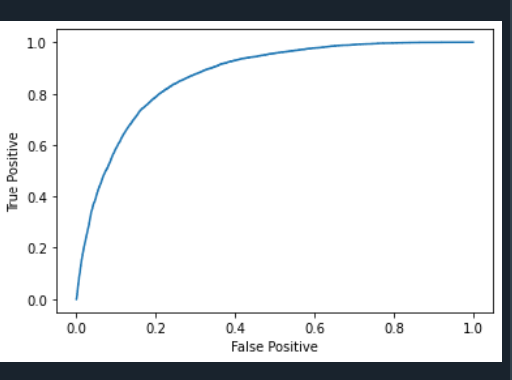
By creating confusion matrix we got the accuracy as 89.06



ROC Curve:

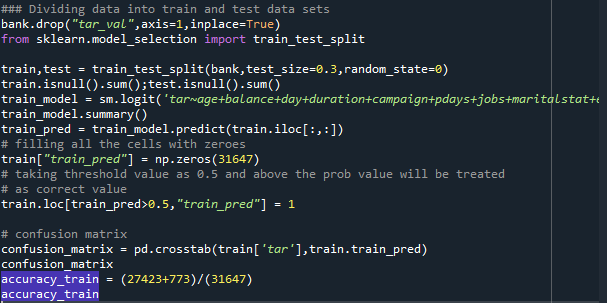


PLOT:

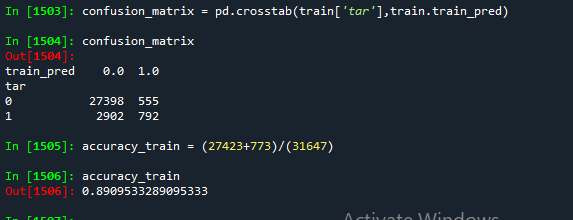


Creating training and testing data

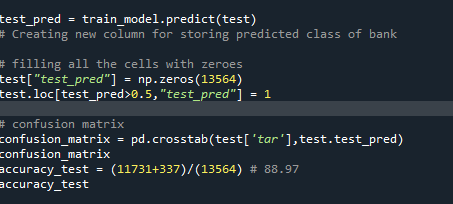
Predicting the accuracy for training data with another model:



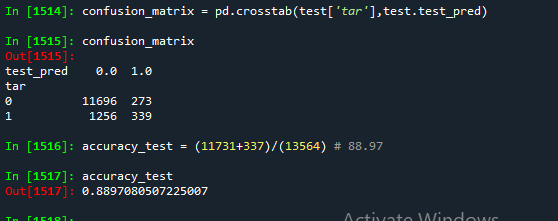
We get the accuracy as 89.09



For Testing Data:



We got the accuracy for testing data as 88.97



FULL CODE:  
# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 11 19:37:02 2020

@author: sunil

"""

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Importing Data

bank=pd.read\_csv("Downloads\\bank-full.csv")

bank.columns

bank.head(10)

#model building

#creating new column for job from discrete to continous data

df=pd.DataFrame(bank)

print(df)

df['jobs']=pd.factorize(df['job'])[0]

print(df)

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df['jobs']=le.fit\_transform(df['job'])

print(df)

df['jobs']=pd.factorize(df['job'],sort=True)[0]

print(df)

bank.drop(["job"],axis=1,inplace=True)

bank

#creating new column for martial for continous data

df1=pd.DataFrame(bank)

df1['maritalstat']=pd.factorize(df1['marital'])[0]

print(df)

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df1['maritalstat']=le.fit\_transform(df['marital'])

print(df1)

df1['maritalstat']=pd.factorize(df1['marital'],sort=True)[0]

print(df1)

bank.drop(['marital'],axis=1,inplace=True)

bank

#creating new column for education to numerical

df2=pd.DataFrame(bank)

df2['edu']=pd.factorize(df2['education'])[0]

print(df2)

df2['edu']=le.fit\_transform(df2['education'])

print(df2)

df2['edu']=pd.factorize(df2['education'],sort=True)[0]

print(df2)

bank.drop(['education'],axis=1,inplace=True)

bank

#creating new column for default to numerical

df3=pd.DataFrame(bank)

df3['defalt']=pd.factorize(df3['default'])[0]

print(df3)

df3['defalt']=le.fit\_transform(df3['default'])

print(df3)

df3['defalt']=pd.factorize(df3['default'],sort=True)[0]

print(df3)

bank.drop(['default'],axis=1,inplace=True)

bank

#creating new column for housing to numerical

df4=pd.DataFrame(bank)

df4['house']=pd.factorize(df4['housing'])[0]

print(df4)

df4['house']=le.fit\_transform(df4['housing'])

print(df4)

df4['house']=pd.factorize(df4['housing'],sort=True)[0]

print(df4)

bank.drop(['housing'],axis=1,inplace=True)

bank

#creating new column for loan ti numerical

df5=pd.DataFrame(bank)

df5['loann']=pd.factorize(df5['loan'])[0]

print(df5)

df5['loann']=le.fit\_transform(df5['loan'])

print(df5)

df5['loann']=pd.factorize(df5['loan'],sort=True)[0]

print(df5)

bank.drop(['loan'],axis=1,inplace=True)

bank

#removing contact it is unknown data

bank.drop(['contact'],axis=1,inplace=True)

#converting month into numerical data

df6=pd.DataFrame(bank)

df6['months']=pd.factorize(df6['month'])[0]

print(df6)

df6['months']=le.fit\_transform(df6['month'])

print(df6)

bank.drop(['month'],axis=1,inplace=True)

bank

#removing previous and poutcome becoz of unknown values

bank.drop(['previous'],axis=1,inplace=True)

bank.drop(['poutcome'],axis=1,inplace=True)

#converting target y into numerical data

df7=pd.DataFrame(bank)

df7['tar']=pd.factorize(df7['y'])[0]

print(df7)

df7['tar']=le.fit\_transform(df7['y'])

print(df7)

df7['tar']=pd.factorize(df7['y'],sort=True)[0]

print(df7)

bank.drop(['y'],axis=1,inplace=True)

print(bank)

bank.columns

#model building

import statsmodels.formula.api as sm

logit\_model = sm.logit('tar~age+balance+day+duration+campaign+pdays+jobs+maritalstat+edu+defalt+house+loann+months',data = bank).fit()

logit\_model.summary()

y\_pred = logit\_model.predict(bank)

bank["pred"]=y\_pred

#creating empty column for target here "tar" is output

bank["tar\_val"]=0

bank.loc[y\_pred>=0.5,"tar\_val"] = 1

bank.tar\_val

from sklearn.metrics import classification\_report

classification\_report(bank.tar\_val,bank.tar)

# confusion matrix

confusion\_matrix = pd.crosstab(bank['tar'],bank.tar\_val)

confusion\_matrix

accuracy = (39188+1079)/(45211) # 89.06

accuracy

# ROC curve

from sklearn import metrics

# fpr => false positive rate

# tpr => true positive rate

fpr, tpr, threshold = metrics.roc\_curve(bank.tar, y\_pred)

# the above function is applicable for binary classification class

plt.plot(fpr,tpr);plt.xlabel("False Positive");plt.ylabel("True Positive")

roc\_auc = metrics.auc(fpr, tpr) # area under ROC curve

### Dividing data into train and test data sets

bank.drop("tar\_val",axis=1,inplace=True)

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

train,test = train\_test\_split(bank,test\_size=0.3,random\_state=0)

train.isnull().sum();test.isnull().sum()

train\_model = sm.logit('tar~age+balance+day+duration+campaign+pdays+jobs+maritalstat+edu+defalt+house+loann+months',data =train).fit()

train\_model.summary()

train\_pred = train\_model.predict(train.iloc[:,:])

# filling all the cells with zeroes

train["train\_pred"] = np.zeros(31647)

# taking threshold value as 0.5 and above the prob value will be treated

# as correct value

train.loc[train\_pred>0.5,"train\_pred"] = 1

# confusion matrix

confusion\_matrix = pd.crosstab(train['tar'],train.train\_pred)

confusion\_matrix

accuracy\_train = (27423+773)/(31647)

accuracy\_train

# Prediction on Test data set

test\_pred = train\_model.predict(test)

# Creating new column for storing predicted class of bank

# filling all the cells with zeroes

test["test\_pred"] = np.zeros(13564)

test.loc[test\_pred>0.5,"test\_pred"] = 1

# confusion matrix

confusion\_matrix = pd.crosstab(test['tar'],test.test\_pred)

confusion\_matrix

accuracy\_test = (11731+337)/(13564) # 88.97

accuracy\_test