# Logistic Regression

**Instructions:**

Please share your answers filled in-line in the word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: Anandakrishnan k v ;;;;;; Batch ID:**19042021

**Topic: Logistic Regression**

**Grading Guidelines:**

**1. An assignment submission is considered complete only when correct and executable code(s) are submitted along with the documentation explaining the method and results. Failing to submit either of those will be considered an invalid submission and will not be considered for evaluation.**

**2. Assignments submitted after the deadline will affect your grades.**

**Grading:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ans** | **Date** |  |  | **Ans** | **Date** |
| Correct | On time | A | 100 |  |  |
| 80% & above | On time | B | 85 | Correct | Late |
| 50% & above | On time | C | 75 | 80% & above | Late |
| 50% & below | On time | D | 65 | 50% & above | Late |
|  |  | E | 55 | 50% & below |  |
| Copied/No Submission |  | F | 45 |  |  |

* **Grade A: (>= 90):** When all assignments are submitted on or before the given deadline.
* **Grade B: (>= 80 and < 90):** 
  + When assignments are submitted on time but less than 80% of problems are completed.

(OR)

* + All assignments are submitted after the deadline.
* **Grade C: (>= 70 and < 80):** 
  + When assignments are submitted on time but less than 50% of the problems are completed.

(OR)

* + Less than 80% of problems in the assignments are submitted after the deadline.
* **Grade D: (>= 60 and < 70):**
  + Assignments submitted after the deadline and with 50% or less problems.
* **Grade E: (>= 50 and < 60):** 
  + Less than 30% of problems in the assignments are submitted after the deadline.

(OR)

* + Less than 30% of problems in the assignments are submitted before the deadline.
* **Grade F: (< 50):** No submission (or) malpractice.

**Hints:**

1. **Business Problem**
   1. **What is the business objective?**
   2. **Are there any constraints?**
2. **Work on each feature of the dataset to create a data dictionary as displayed in the below image:**



**2.1 Make a table as shown above and provide information about the features such as its data type and its relevance to the model building. And if not relevant, provide reasons and a description of the feature.**

**Using R and Python codes perform:**

1. **Data Pre-processing**

**3.1 Data Cleaning, Feature Engineering, etc.**

**3.2 Outlier Treatment.**

1. **Exploratory Data Analysis (EDA):**
   1. **Summary.**
   2. **Univariate analysis.**
   3. **Bivariate analysis.**
2. **Model Building**
   1. **Build the model on the scaled data (try multiple options).**
   2. **Build a Logistic Regression model.**
   3. **Train and test the model and compare accuracies by building a confusion matrix, plotting ROC and AUC curves.**
   4. **Briefly explain the model output in the documentation.**
3. **Write about the benefits/impact of the solution - in what way does the business (client) benefit from the solution provided?**

**Business Problem**

A psychological study has been conducted by a team of students at a university on married couples to determine the cause of having an extra marital affair. They have surveyed and collected a sample of data on which they would like to do further analysis. Apply Logistic Regression on the data to correctly classify whether a given person will have an affair or not given the set of attributes. Convert the naffairs column to discrete binary type before proceeding with the algorithm.

**What is the business objective?**

Determine the cause of having an extra marital affair on married couples by analysing the collected sample of data.

**Are there any constraints?**

**Maximize :** The accuracy of prediction

**Minimize :** the time lag

**Python Code:-**

#### affairs ####

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats.mstats import winsorize

import pandas as pd

import numpy as np

# import seaborn as sb

import matplotlib.pyplot as plt

import statsmodels.formula.api as sm

from sklearn.model\_selection import train\_test\_split # train and test

from sklearn import metrics

from sklearn.metrics import roc\_curve, auc

from sklearn.metrics import classification\_report

#Importing Data

df = pd.read\_csv("C:/Users/user/Downloads/logistic reg/Affairs.csv")

# droping index column

df = df.iloc[:,1:]

#removing CASENUM

c1 = df

c1.head(11)

c1.describe()

c1.isna().sum() # no null values

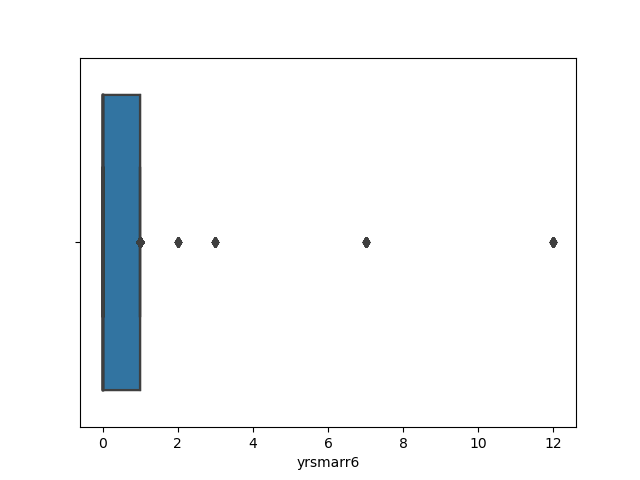
####### Outlier Treatment ########

# Boxplots

for i in c1.columns:

sns.boxplot(c1[i].dropna())

plt.show() ## no outliers ; combined boxplot not showig any outliers



# converting naffairs column to binary format

n= len(c1['naffairs'])

n

for i in range(0,n,1):

if(c1['naffairs'][i] ==0):

c1['naffairs'][i] = 0

else:

c1['naffairs'][i] = 1

# Sctter plot and histogram between variables

sns.pairplot(c1) # sp-hp, wt-vol multicolinearity issue

#################### Train & Test split #######################################

### Splitting the data into train and test data

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

train\_data, test\_data = train\_test\_split(c1, test\_size = 0.3) # 30% test data

# Model building

# import statsmodels.formula.api as smc1.columns

# Model building

import statsmodels.formula.api as sm

logit\_model = sm.logit('naffairs ~ kids + vryunhap + unhap + avgmarr + hapavg + vryhap + antirel + notrel + slghtrel + smerel + vryrel + yrsmarr1 + yrsmarr2 + yrsmarr3 + yrsmarr4 + yrsmarr5 + yrsmarr6', data = train\_data).fit()

#summary

logit\_model.summary2() # for AIC

logit\_model.summary() # nan values are appearing ; this should be consequence of the first error ==> " maximum likelihood failed to converge" so need to check the format and type of varables of data properly

pred = logit\_model.predict(train\_data.iloc[ :, 1:])

# from sklearn import metrics

fpr, tpr, thresholds = roc\_curve(train\_data.naffairs, pred)

optimal\_idx = np.argmax(tpr - fpr)

optimal\_threshold = thresholds[optimal\_idx]

optimal\_threshold # 0.2815 choose the threshold point giving max diff between tpr and fpr

import pylab as pl

i = np.arange(len(tpr))

roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})

roc.iloc[(roc.tf-0).abs().argsort()[:1]]

# fpr tpr 1-fpr tf thresholds

#0.326019 0.673267 0.673981 -0.000714 0.238503

#threshold = 0.238503 ; choose the threshold value where tpr and tnr are high and with minimum difference (sensitivity and specificity are high and with minimum diff)

# almost equal and near to optimum\_threshold getting from corresponding max tpr- fpr value

# Plot tpr vs 1-fpr

fig, ax = pl.subplots()

pl.plot(roc['tpr'], color = 'red')

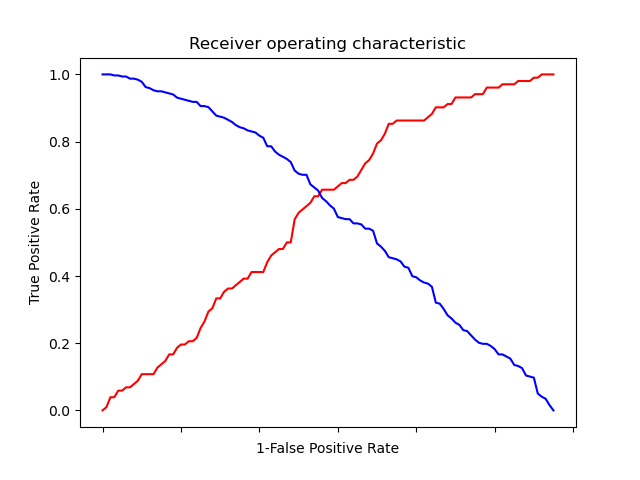
pl.plot(roc['1-fpr'], color = 'blue')

pl.xlabel('1-False Positive Rate')

pl.ylabel('True Positive Rate')

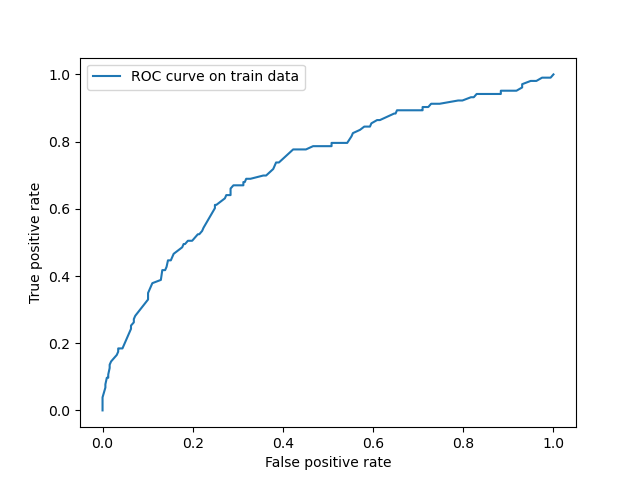
pl.title('Receiver operating characteristic')

ax.set\_xticklabels([]) # graph showing tpr value almost near to 0.67 for optimum threshold point



#ROC CURVE AND AUC

plt.plot(fpr, tpr);plt.xlabel("False positive rate (train data)");plt.ylabel("True positive rate (train data)")



roc\_auc = auc(fpr, tpr)

print("Area under the ROC curve : %f" % roc\_auc) # 0.73 ; means false in acceptable region

# prediction of output according to propbability and threshold value

# filling all the cells with zeroes

l\_train = len(train\_data["naffairs"])

l\_train

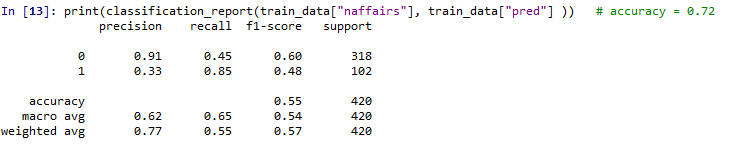
train\_data["pred"] = np.zeros(420)

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

train\_data.loc[pred > optimal\_threshold, "pred"] = 1

# classification report

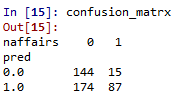
print(classification\_report(train\_data["naffairs"], train\_data["pred"] )) # accuracy = 0.72



# confusion matrix

confusion\_matrx = pd.crosstab(train\_data["pred"], train\_data['naffairs'])

confusion\_matrx



accuracy\_train = (243 + 60)/(420) # TP = 60, TN = 243 , FN = 41, FP = 76

print(accuracy\_train) # 0.72

#################### Prediction on Test data #######################################

# Prediction on Test data set

test\_pred = logit\_model.predict(test\_data)

# Creating new column for storing predicted class of Attorney

# filling all the cells with zeroes

l\_test = len(test\_data["naffairs"])

l\_test

test\_data["test\_pred"] = np.zeros(181)

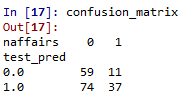
# taking threshold value as 'optimal\_threshold' and above the thresold prob value will be treated as 1

test\_data.loc[test\_pred > optimal\_threshold, "test\_pred"] = 1

# confusion matrix

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

confusion\_matrix

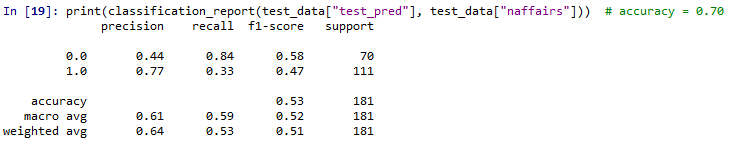


accuracy\_test = (101 + 25)/(181) # TP = 25, TN = 101 , FN = 24, FP = 31

accuracy\_test # 0.69

# classification report

print(classification\_report(test\_data["test\_pred"], test\_data["naffairs"])) # accuracy = 0.70

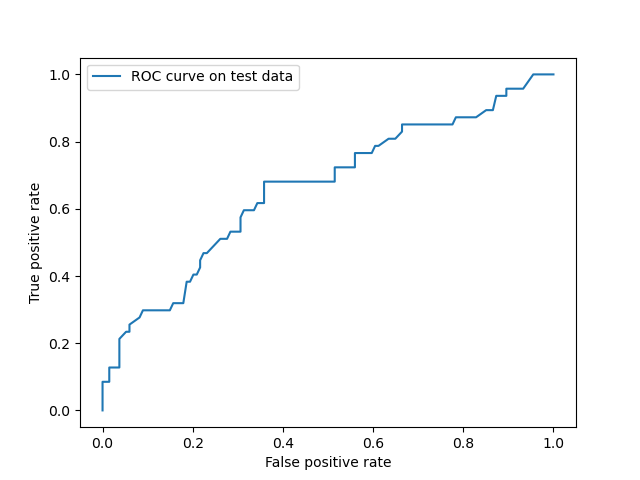


#ROC CURVE AND AUC

fpr, tpr, threshold = metrics.roc\_curve(test\_data["naffairs"], test\_pred)

#PLOT OF ROC

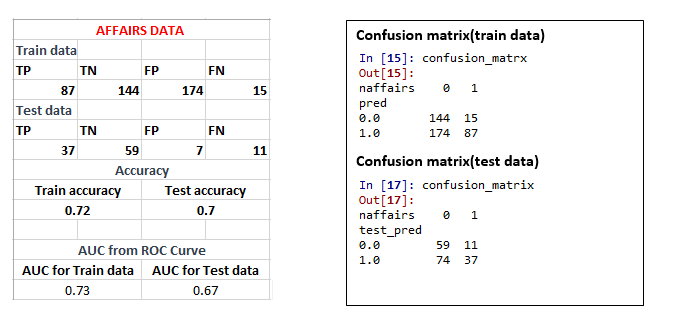
plt.plot(fpr, tpr);plt.xlabel("False positive rate");plt.ylabel("True positive rate")



roc\_auc\_test = metrics.auc(fpr, tpr)

roc\_auc\_test # area under the curve = 0.67 : fall under poor region

**Summary:-**



* Train & Test accuracies are almost equal and considerable ( no over fitting and under fitiing problems)
* AUC value are almost equal. But in ROC curve AUC value for Test data falls in poor region but that of Train data falls in acceptable region. So threshold value choosen by fitting the model on train data not much perform well on test data
* But in over all model performance is considerable

**Business benefit:-**

Model will helps to determine the cause of having an extra marital affair on married couples by analysing the collected sample of data.

**Business Problem:-**

In this time and age of widespread internet usage, effective and targeted marketing plays a vital role. A marketing company would like to develop a strategy by analyzing their customer data. For this, data like age, location, time of activity, etc. has been collected to determine whether a user will click on an ad or not. Perform Logistic Regression on the given data to predict whether a user will click on an ad or not.

**What is the business objective?**

Develop a strategy by analyzing the customer datas like age, location, time of activity, etc to determine whether a user will click on an ad or not

**Are there any constraints?**

**Maximize :** The accuracy of prediction

**Minimize :** The time lag

**Maximize :** customer satisfaction upon the service

**Python Code:-**

#### advertisement ####

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats.mstats import winsorize

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.formula.api as sm

from sklearn.model\_selection import train\_test\_split # train and test

from sklearn import metrics

from sklearn.metrics import roc\_curve, auc

from sklearn.metrics import classification\_report

#Importing Data

df = pd.read\_csv("C:/Users/user/Downloads/logistic reg/advertising.csv")

# create a copy

df1 = df.copy(deep= True)

# droping less informative columns

df1.columns

df1.drop('Ad\_Topic\_Line', axis=1, inplace = True)

df1.drop('Timestamp', axis=1, inplace = True)

#changing column names

df1.rename({'Daily\_Time\_ Spent \_on\_Site':'DTSS' ,'Daily Internet Usage':'DIU' }, axis=1, inplace =True)

#removing CASENUM

c1 = df1

c1.head(11)

c1.describe()

c1.info()

c1.isna().sum() # no null values

#converting into numerical

from sklearn.preprocessing import LabelEncoder

lb = LabelEncoder()

c1["City"] = lb.fit\_transform(c1["City"])

c1["Country"] = lb.fit\_transform(c1["Country"])

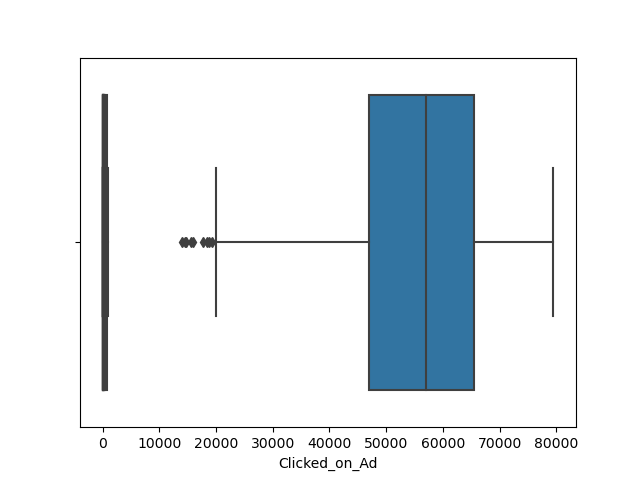
####### Outlier Treatment ########

# Boxplots

for i in c1.columns:

sns.boxplot(c1[i].dropna())

plt.show() ## no outliers ; combined boxplot not showig any outliers



### normalisation scaling ###

#Importing the Libraries

from sklearn.preprocessing import StandardScaler, MinMaxScaler,RobustScaler

# define standard scaler

scaler = MinMaxScaler() # Standard Scaler or Standardization

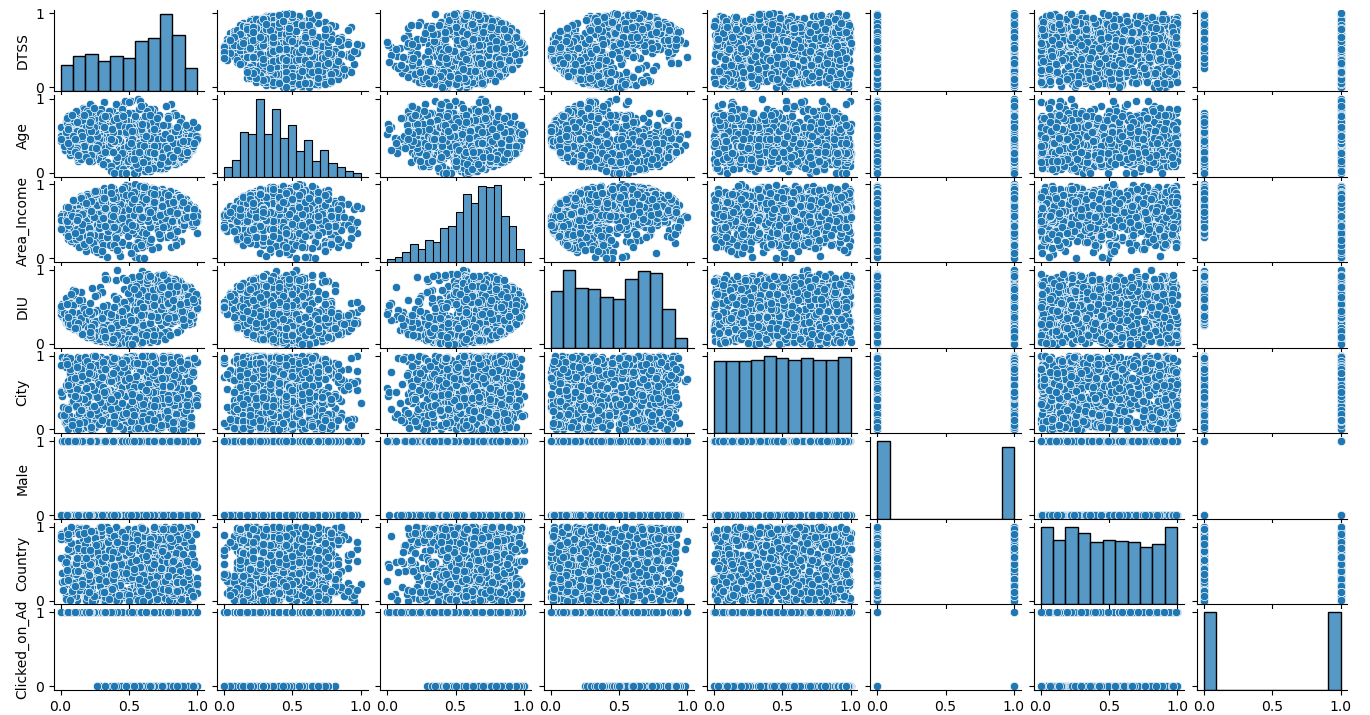
# scaling data except output column

c1.iloc[:,0:7] = scaler.fit\_transform(c1.iloc[:,0:7]) #Fit to data, then transform it.

c1.iloc[:,0:4].describe()

# Sctter plot and histogram between variables

sns.pairplot(c1) # sp-hp, wt-vol multicolinearity issue



#################### Train & Test split #######################################

### Splitting the data into train and test data

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

train\_data, test\_data = train\_test\_split(c1, test\_size = 0.3) # 30% test data

# Model building

# import statsmodels.formula.api as smc1.columns

# Model building

import statsmodels.formula.api as sm

logit\_model = sm.logit('Clicked\_on\_Ad ~ DTSS + Age + Area\_Income + DIU + City + Male + Country', data = train\_data).fit()

#summary

logit\_model.summary2() # for AIC

logit\_model.summary() # City, Male & Country column features showing P\_value > 0.05. so actually we have to rectify multicollinearity problem between input variables. But we skip that step and directly going logistic operation

pred = logit\_model.predict(train\_data.iloc[ :, 0:7])

# from sklearn import metrics

fpr, tpr, thresholds = roc\_curve(train\_data.Clicked\_on\_Ad, pred)

optimal\_idx = np.argmax(tpr - fpr)

optimal\_threshold = thresholds[optimal\_idx]

optimal\_threshold # 0.7066 choose the threshold point giving max diff between tpr and fpr

import pylab as pl

i = np.arange(len(tpr))

roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})

roc.iloc[(roc.tf-0).abs().argsort()[:1]]

# fpr tpr 1-fpr tf thresholds

#0.035088 0.963687 0.964912 -0.001225 0.437499

#threshold = 0.437499 ; choose the threshold value where tpr and tnr are high and with minimum difference (sensitivity and specificity are high and with minimum diff)

# almost equal and near to optimum\_threshold getting from corresponding max tpr- fpr value

# Plot tpr vs 1-fpr

fig, ax = pl.subplots()

pl.plot(roc['tpr'], color = 'red')

pl.plot(roc['1-fpr'], color = 'blue')

pl.xlabel('1-False Positive Rate')

pl.ylabel('True Positive Rate')

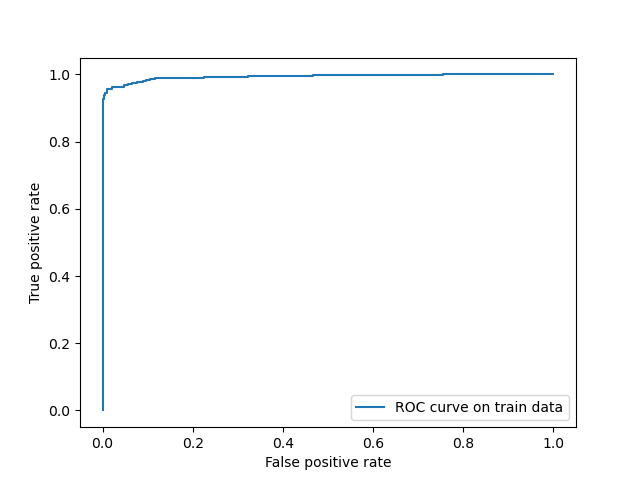
pl.title('Receiver operating characteristic')

ax.set\_xticklabels([]) # graph showing tpr value almost near to 0.96 for optimum threshold point



#ROC CURVE AND AUC

plt.plot(fpr, tpr, label = "ROC curve on train data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc = auc(fpr, tpr)

print("Area under the ROC curve : %f" % roc\_auc) # 0.99 ; means false in outstanding region

# prediction of output according to propbability and threshold value

# filling all the cells with zeroes

l\_train = len(train\_data["Clicked\_on\_Ad"])

l\_train

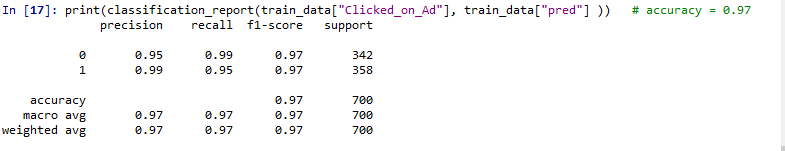
train\_data["pred"] = np.zeros(700)

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

train\_data.loc[pred > optimal\_threshold, "pred"] = 1

# classification report

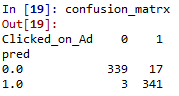
print(classification\_report(train\_data["Clicked\_on\_Ad"], train\_data["pred"] )) # accuracy = 0.97



# confusion matrix

confusion\_matrx = pd.crosstab(train\_data["pred"], train\_data['Clicked\_on\_Ad'])

confusion\_matrx



accuracy\_train = (340 + 339)/(700) # TP = 339, TN = 340 , FN = 19, FP = 2

print(accuracy\_train) # 0.97

#################### Predictions on Test data #######################################

# Prediction on Test data set

test\_pred = logit\_model.predict(test\_data)

# Creating new column for storing predicted class of Attorney

# filling all the cells with zeroes

l\_test = len(test\_data["Clicked\_on\_Ad"])

l\_test

test\_data["test\_pred"] = np.zeros(300)

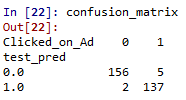
# taking threshold value as 'optimal\_threshold' and above the thresold prob value will be treated as 1

test\_data.loc[test\_pred > optimal\_threshold, "test\_pred"] = 1

# confusion matrix

confusion\_matrix = pd.crosstab(test\_data.test\_pred, test\_data['Clicked\_on\_Ad'])

confusion\_matrix

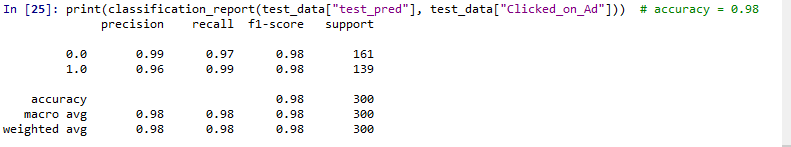


accuracy\_test = (157 + 136)/(300) # TP = 136, TN = 157 , FN = 6, FP = 1

accuracy\_test # 0.98

# classification report

print(classification\_report(test\_data["test\_pred"], test\_data["Clicked\_on\_Ad"])) # accuracy = 0.98

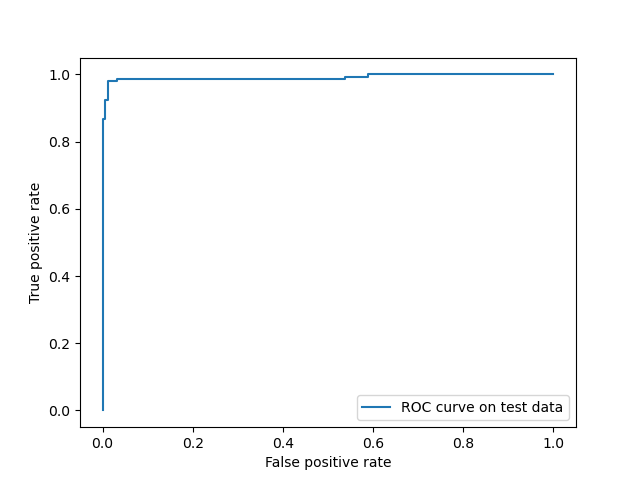


#ROC CURVE AND AUC

fpr, tpr, threshold = metrics.roc\_curve(test\_data["Clicked\_on\_Ad"], test\_pred)

#PLOT OF ROC

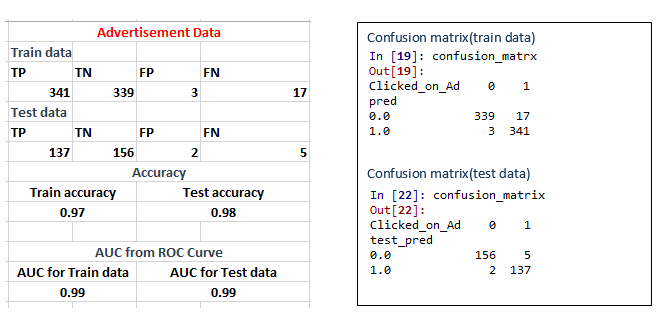
plt.plot(fpr, tpr, label = "ROC curve on test data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc\_test = metrics.auc(fpr, tpr)

roc\_auc\_test # area under the curve = 0.99 : fall in outstanding region

**Summary:-**



* Train & Test data accuracies are almost equal and good( no over fitting and under fitting problem)
* AUC value are almost equal and better ( falls in outstanding region) for test & train data. So threshold value choosen by fitting model on train data, gives better prediction on test data either. So model is good

**Business benefit:-**

It will helps by analyzing the customer datas like age, location, time of activity, etc to determine whether a user will click on an ad or not. Thereby understands the persons mentallity and do an action about recommending an add on window or not. Hence at the same time doesn’t gonna disturb persons they are not interested and recommending advertisement for those are interested. So satisfying customers satisfaction and improving the business on advertisement

**Business Problem:-**

Perform Logistic Regression on the dataset to predict whether a candidate will win or lose the election based on factors like amount of money spent and popularity rank.

**What is the business objective?**

Built a model to predict whether a candidate will win or lose the election based on factors like amount of money spent and popularity rank.

**Are there any constraints?**

**Maximize :** The accuracy of prediction

**Minimize :** The time lag

**Python Code:-**

#### advertisement ####

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats.mstats import winsorize

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.formula.api as sm

from sklearn.model\_selection import train\_test\_split # train and test

from sklearn import metrics

from sklearn.metrics import roc\_curve, auc

from sklearn.metrics import classification\_report

#Importing Data

df = pd.read\_csv("C:/Users/user/Downloads/logistic reg/election\_data.csv")

# create a copy

df1 = df.copy(deep= True)

# droping less informative columns

df1.columns

df1.drop('Election-id', axis=1, inplace = True)

df1.drop('Year', axis=1, inplace = True)

#changing column names

df1.rename({'Amount Spent':'m\_spent' ,'Popularity Rank':'p\_rank' }, axis=1, inplace =True)

#removing CASENUM

c1 = df1

c1.head(11)

c1.describe()

c1.info()

c1.isna().sum() # no null values

c1.dropna() # directly droping null value row without going for any other imputation strategy since it is less informative

c1.dropna(how = 'any',inplace = True)

c1.isna().sum()

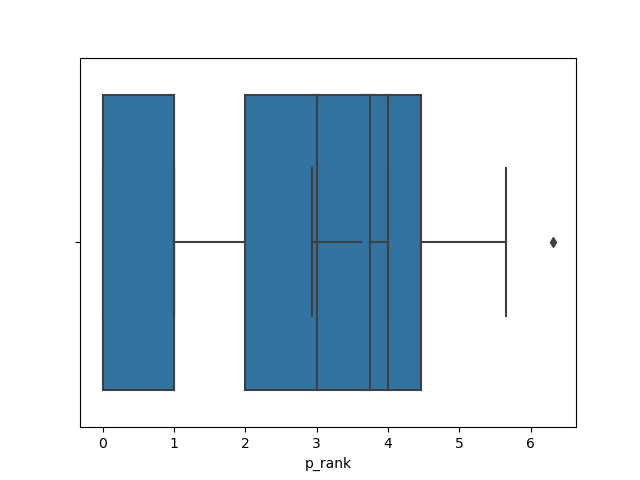
####### Outlier Treatment ########

# Boxplots

for i in c1.columns:

sns.boxplot(c1[i].dropna())

plt.show() ## no outliers ; combined boxplot not showig any outliers



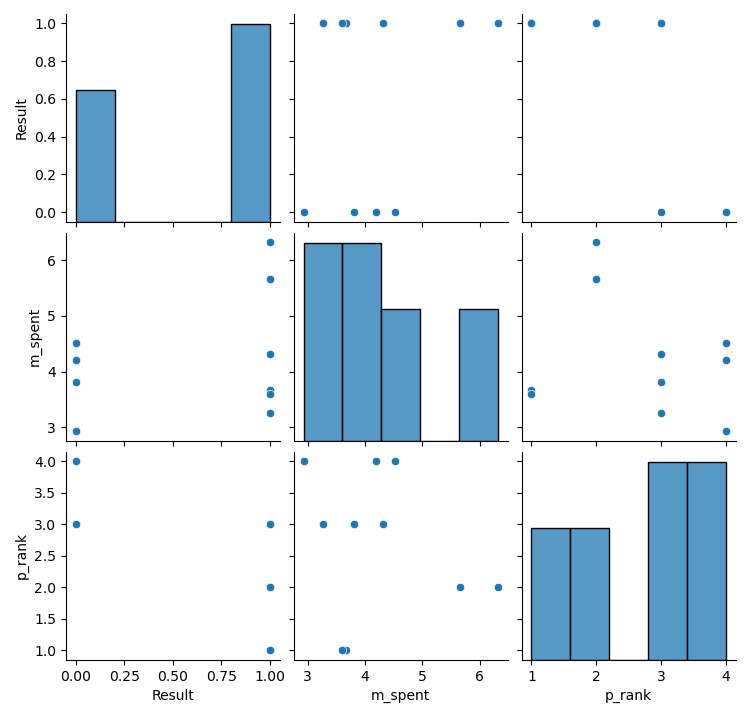
### normalisation scaling ###

# since input variables ranges are almost near we are directly going for analysis

# without do any scaling

# Sctter plot and histogram between variables

sns.pairplot(c1) # sp-hp, wt-vol multicolinearity issue



#################### Train & Test split #######################################

### Splitting the data into train and test data

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

train\_data, test\_data = train\_test\_split(c1, test\_size = 0.3) # 30% test data

# Model building

# import statsmodels.formula.api as smc1.columns

# Model building

import statsmodels.formula.api as sm

logit\_model = sm.logit('Result ~ m\_spent + p\_rank', data = train\_data).fit()

#summary

logit\_model.summary2() # for AIC

logit\_model.summary() # Both input features showing P\_value > 0.05. so actually we have to rectify multicollinearity problem between input variables. But we skip that step and directly going logistic operation

pred = logit\_model.predict(train\_data.iloc[ :, 1:])

# from sklearn import metrics

fpr, tpr, thresholds = roc\_curve(train\_data.Result, pred)

optimal\_idx = np.argmax(tpr - fpr)

optimal\_threshold = thresholds[optimal\_idx]

optimal\_threshold # 0.679 choose the threshold point giving max diff between tpr and fpr

import pylab as pl

i = np.arange(len(tpr))

roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})

roc.iloc[(roc.tf-0).abs().argsort()[:1]]

# fpr tpr 1-fpr tf thresholds

#0.0 0.8 1.0 -0.2 0.679359

#threshold = 0.679359 ; choose the threshold value where tpr and tnr are high and with minimum difference (sensitivity and specificity are high and with minimum diff)

# almost equal and near to optimum\_threshold getting from corresponding max tpr- fpr value

# Plot tpr vs 1-fpr

fig, ax = pl.subplots()

pl.plot(roc['tpr'], color = 'red')

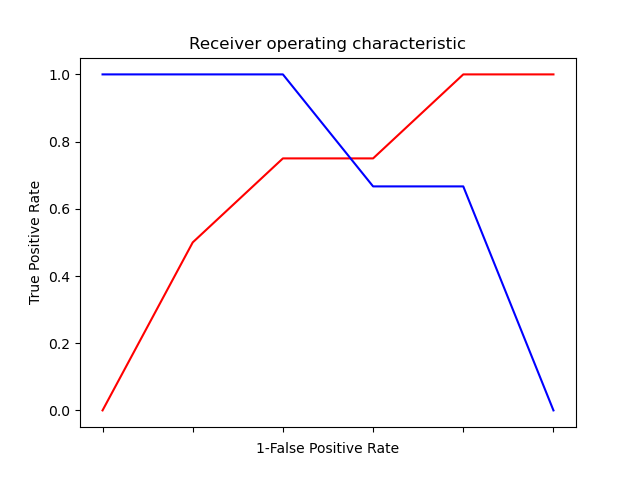
pl.plot(roc['1-fpr'], color = 'blue')

pl.xlabel('1-False Positive Rate')

pl.ylabel('True Positive Rate')

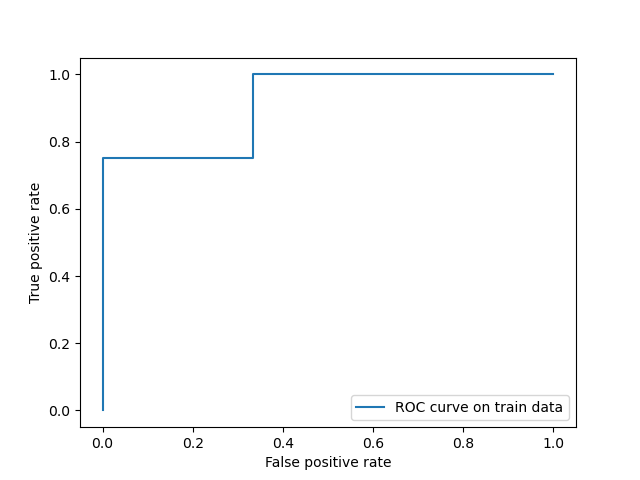
pl.title('Receiver operating characteristic')

ax.set\_xticklabels([]) # graph showing tpr value almost near to 0.8 for optimum threshold point



#ROC CURVE AND AUC

plt.plot(fpr, tpr, label = "ROC curve on train data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc = auc(fpr, tpr)

print("Area under the ROC curve : %f" % roc\_auc) # 0.9 ; means false in outstanding region

# prediction of output according to propbability and threshold value

# filling all the cells with zeroes

l\_train = len(train\_data["Result"])

l\_train

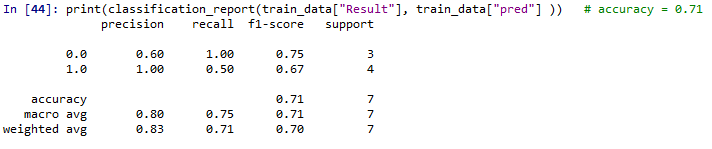
train\_data["pred"] = np.zeros(7)

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

train\_data.loc[pred > optimal\_threshold, "pred"] = 1

# classification report

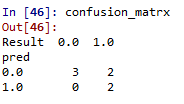
print(classification\_report(train\_data["Result"], train\_data["pred"] )) # accuracy = 0.71



# confusion matrix

confusion\_matrx = pd.crosstab(train\_data["pred"], train\_data['Result'])

confusion\_matrx



accuracy\_train = (2 + 3)/(7) # TP = 3, TN = 2 , FN = 2, FP = 0

print(accuracy\_train) # 0.71

#################### Analysis on Test data #######################################

# Prediction on Test data set

test\_pred = logit\_model.predict(test\_data)

# Creating new column for storing predicted class of Attorney

# filling all the cells with zeroes

l\_test = len(test\_data["Result"])

l\_test

test\_data["test\_pred"] = np.zeros(3)

# taking threshold value as 'optimal\_threshold' and above the thresold prob value will be treated as 1

test\_data.loc[test\_pred > optimal\_threshold, "test\_pred"] = 1

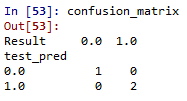
# confusion matrix

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

confusion\_matrix

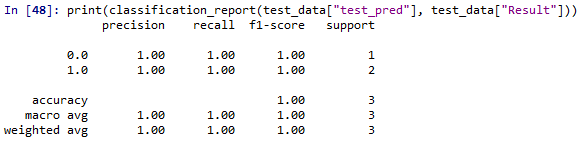
accuracy\_test = (2 + 1)/(3) # TP = 1, TN = 2 , FN = 0, FP = 0

accuracy\_test # 1.0



# classification report

print(classification\_report(test\_data["test\_pred"], test\_data["Result"]))

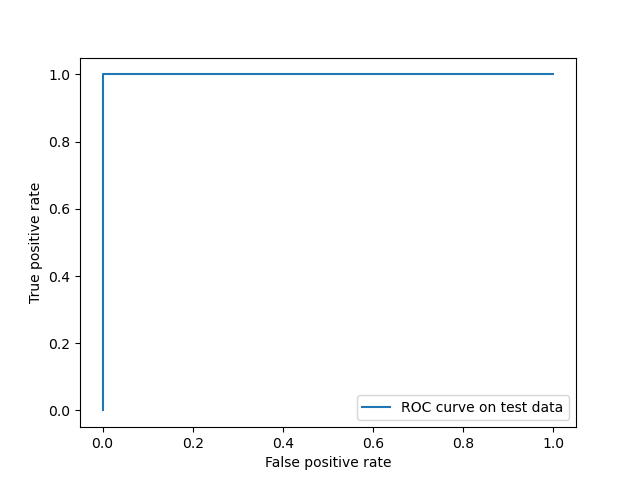


#ROC CURVE AND AUC

fpr, tpr, threshold = metrics.roc\_curve(test\_data["Result"], test\_pred)

#PLOT OF ROC

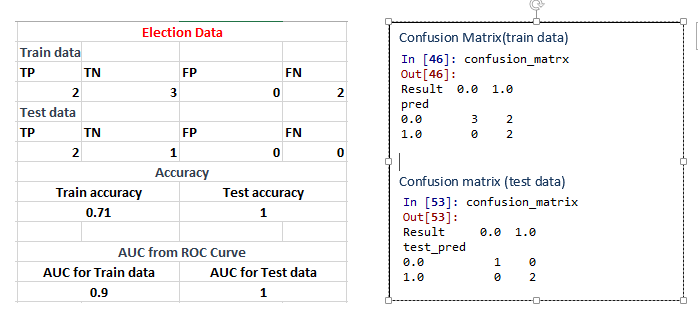
plt.plot(fpr, tpr, label = "ROC curve on test data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc\_test = metrics.auc(fpr, tpr)

roc\_auc\_test # area under the curve = 1.0 : fall in outstanding region

**Summary:-**

****

* Test accuracy is showing better value compare to that of train ( its because they have very less number of datas in test data set and all that value falls under true prediction)
* AUC value are almost equal (but better for test data) and better ( falls in outstanding region) for test & train data. So threshold value choosen by fitting model on train data, gives better prediction on test data either. So model is good

**Business benefit:-**

It will helps to predict whether a candidate will win or lose the election based on factors like amount of money spent and popularity rank.

**Business Problem:-**

It is vital for banks that customers put in long term fixed deposits as they use it to pay interest to customers and it is not viable to ask every customer if they will put in a long-term deposit or not. So, build a Logistic Regression model to predict whether a customer will put in a long-term fixed deposit or not based on the different variables given in the data. The output variable in the dataset is Y which is binary. Snapshot of the dataset is given below.

**What is the business objective?**

Build a Logistic Regression model to predict whether a customer will put in a long-term fixed deposit or not based on the different variables given in the data

**Are there any constraints?**

**Maximize :** The accuracy of prediction

**Minimize :** The time lag

**Python Code:-**

#### advertisement ####

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats.mstats import winsorize

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.formula.api as sm

from sklearn.model\_selection import train\_test\_split # train and test

from sklearn import metrics

from sklearn.metrics import roc\_curve, auc

from sklearn.metrics import classification\_report

#Importing Data

df = pd.read\_csv("C:/Users/user/Downloads/logistic reg/bank\_data.csv")

# create a copy

df1 = df.copy(deep= True)

df1 = df1[['y','age', 'default', 'balance', 'housing', 'loan', 'duration', 'campaign',

'pdays', 'previous', 'poutfailure', 'poutother', 'poutsuccess',

'poutunknown', 'con\_cellular', 'con\_telephone', 'con\_unknown',

'divorced', 'married', 'single', 'joadmin.', 'joblue.collar',

'joentrepreneur', 'johousemaid', 'jomanagement', 'joretired',

'joself.employed', 'joservices', 'jostudent', 'jotechnician',

'jounemployed', 'jounknown']]

#changing column names

df1.rename({'joadmin.':'j' ,'con\_cellular':'cc' ,'con\_telephone':'ct','con\_unknown':'cu','joblue.collar':'jc','joself.employed':'je' }, axis=1, inplace =True)

#removing CASENUM

c1 = df1

c1.head(11)

c1.describe()

c1.info()

c1.isna().sum() # no null values

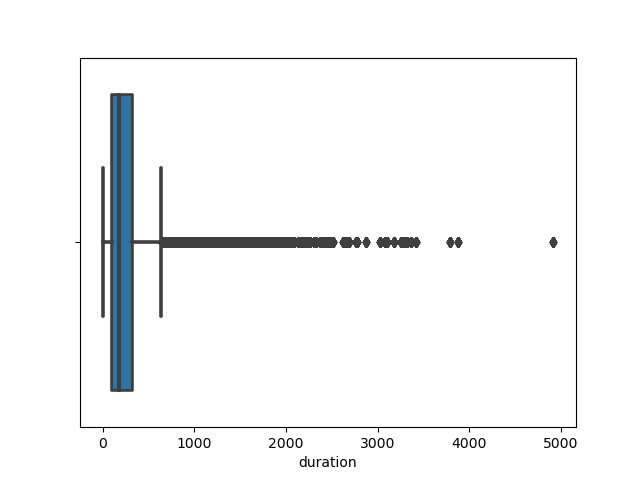
####### Outlier Treatment ########

# Boxplots

for i in c1.columns:

sns.boxplot(c1.iloc[:,6].dropna())

plt.show() ## have outliers ; combined boxplot showig outliers



#individual boxplot for numeric data columns(except binary)

sns.boxplot(c1['age']);plt.title('box plot for age') # outlier present

sns.boxplot(c1['balance']);plt.title('box plot for balance') # outlier present

sns.boxplot(c1['duration']);plt.title('box plot for duration') # outlier present

#outlier treatment for c1.age

q1 = c1['age'].quantile(0.25)

q1

q3 = c1['age'].quantile(0.75)

q3

IQR = q3 - q1

IQR

lower\_limit = q1-(1.5\*IQR)

lower\_limit

upper\_limit = q3+(1.5\*IQR)

upper\_limit

#finding outliers indexes

outliers = np.where(c1['age']<lower\_limit, True, np.where(c1['age']>upper\_limit, True, False))

outliers

#finding outlier values

outlier\_values = c1['age'][outliers]

outliers.sum()

outlier\_values

#Replacing by pulling outliers to lower and upper limit

c1['age'] = np.where(c1['age']<q1, lower\_limit, np.where(c1['age']>q3, upper\_limit, c1['age']))

sns.boxplot(c1['age']);plt.title('box plot for age after replacing')

#outlier treatment for c1.balance

q1 = c1['balance'].quantile(0.25)

q1

q3 = c1['balance'].quantile(0.75)

q3

IQR = q3 - q1

IQR

lower\_limit = q1-(1.5\*IQR)

lower\_limit

upper\_limit = q3+(1.5\*IQR)

upper\_limit

#finding outliers indexes

outliers = np.where(c1['balance']<lower\_limit, True, np.where(c1['balance']>upper\_limit, True, False))

outliers

#finding outlier values

outlier\_values = c1['balance'][outliers]

outliers.sum()

outlier\_values

#Replacing by pulling outliers to lower and upper limit

c1['balance'] = np.where(c1['balance']<q1, lower\_limit, np.where(c1['balance']>q3, upper\_limit, c1['balance']))

sns.boxplot(c1['balance']);plt.title('box plot for balance after replacing')

#outlier treatment for c1.duration

q1 = c1['duration'].quantile(0.25)

q1

q3 = c1['duration'].quantile(0.75)

q3

IQR = q3 - q1

IQR

lower\_limit = q1-(1.5\*IQR)

lower\_limit

upper\_limit = q3+(1.5\*IQR)

upper\_limit

#finding outliers indexes

outliers = np.where(c1['duration']<lower\_limit, True, np.where(c1['duration']>upper\_limit, True, False))

outliers

#finding outlier values

outlier\_values = c1['duration'][outliers]

outliers.sum()

outlier\_values

#Replacing by pulling outliers to lower and upper limit

c1['duration'] = np.where(c1['duration']<q1, lower\_limit, np.where(c1['duration']>q3, upper\_limit, c1['duration']))

sns.boxplot(c1['duration']);plt.title('box plot for duration after replacing')

###### zero variance operation ###

c1.shape

## importing ###

from sklearn.feature\_selection import VarianceThreshold

# Feature selector that removes all low-variance features that meets the variance threshold limit

var\_thres = VarianceThreshold(threshold=0.02) # Threshold is subjective.

var\_thres.fit(c1) ### fit the var\_thres to data set c11

# Generally we remove the columns with zero variance, but i took thresold value 0.02 (Near Zero Variance)

var\_thres.get\_support() ### it giving an array out, where zero variant column treat as False value. we already fit var\_thres to c1. so it gives corresponding information on c1

c1.columns[var\_thres.get\_support()] ## non-zero variant column names

constant\_columns = [column for column in c1.columns if column not in c1.columns[var\_thres.get\_support()]]

print(len(constant\_columns)) ### number of zero variant variables

for feature in constant\_columns:

print(feature) ### names of corresponding zero variant columns ; "default", "jounknown"

c1 = c1.drop(constant\_columns, axis = 1) ### data set with non-zero variant variables or features after droping "default", "jounknown" columns

### normalisation scaling ###

#Importing the Libraries

from sklearn.preprocessing import StandardScaler, MinMaxScaler,RobustScaler

# define standard scaler

scaler = MinMaxScaler() # Standard Scaler or Standardization

# scaling data except output column

c1.iloc[:,1:] = scaler.fit\_transform(c1.iloc[:,1:]) #Fit to data, then transform it.

c1.iloc[:,1:].describe()

# Sctter plot and histogram between variables

sns.pairplot(c1) # sp-hp, wt-vol multicolinearity issue

#################### Train & Test split #######################################

### Splitting the data into train and test data

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

train\_data, test\_data = train\_test\_split(c1, test\_size = 0.3) # 30% test data

# Model building

# import statsmodels.formula.api as smc1.columns

# Model building

import statsmodels.formula.api as sm

logit\_model = sm.logit('y ~ age + balance + housing + loan + duration + campaign + pdays + previous + poutfailure + poutother + poutsuccess + poutunknown + cc + ct + cu + divorced + married + single + j + jc + joentrepreneur + johousemaid + jomanagement + joretired + je + joservices + jostudent + jotechnician + jounemployed', data= train\_data).fit()

#summary

logit\_model.summary2() # for AIC

logit\_model.summary() # some of the feature column showing P\_value > 0.05 and nan values. so actually we have to rectify multicollinearity problem between input variables. But we skip that step and directly going logistic operation

#nan values are appearing ; this should be consequence of the first error ==> " maximum likelihood failed to converge" so need to check the format and type of varables of data properly

pred = logit\_model.predict(train\_data.iloc[ :, 1:])

# from sklearn import metrics

fpr, tpr, thresholds = roc\_curve(train\_data.y, pred)

optimal\_idx = np.argmax(tpr - fpr)

optimal\_threshold = thresholds[optimal\_idx]

optimal\_threshold # 0.086 choose the threshold point giving max diff between tpr and fpr

import pylab as pl

i = np.arange(len(tpr))

roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})

roc.iloc[(roc.tf-0).abs().argsort()[:1]]

# fpr tpr 1-fpr tf thresholds

#0.206087 0.794198 0.793913 0.000285 0.118624

#threshold = 0.118624 ; choose the threshold value where tpr and tnr are high and with minimum difference (sensitivity and specificity are high and with minimum diff)

# almost equal and near to optimum\_threshold getting from corresponding max tpr- fpr value

# Plot tpr vs 1-fpr

fig, ax = pl.subplots()

pl.plot(roc['tpr'], color = 'red')

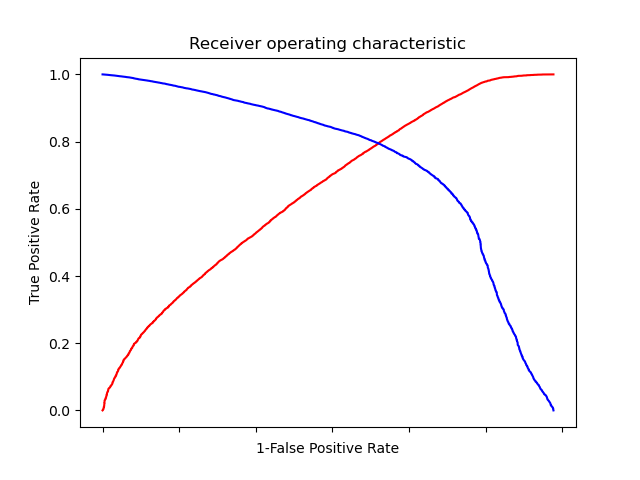
pl.plot(roc['1-fpr'], color = 'blue')

pl.xlabel('1-False Positive Rate')

pl.ylabel('True Positive Rate')

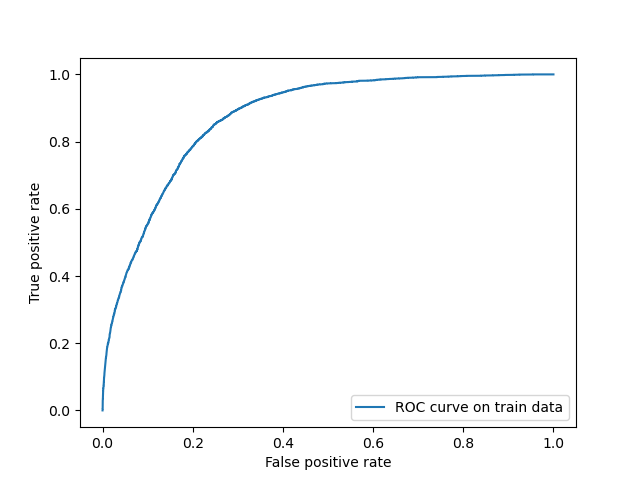
pl.title('Receiver operating characteristic')

ax.set\_xticklabels([]) # graph showing tpr value almost near to 0.81 for optimum threshold point



#ROC CURVE AND AUC

plt.plot(fpr, tpr, label = "ROC curve on train data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc = auc(fpr, tpr)

print("Area under the ROC curve : %f" % roc\_auc) # 0.87 ; means false in excellent region

# prediction of output according to propbability and threshold value

# filling all the cells with zeroes

l\_train = len(train\_data["y"])

l\_train

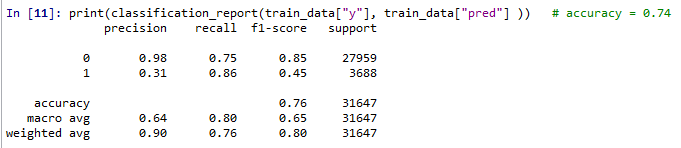
train\_data["pred"] = np.zeros(31647)

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

train\_data.loc[pred > optimal\_threshold, "pred"] = 1

# classification report

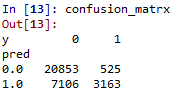
print(classification\_report(train\_data["y"], train\_data["pred"] )) # accuracy = 0.74



# confusion matrix

confusion\_matrx = pd.crosstab(train\_data["pred"], train\_data['y'])

confusion\_matrx



accuracy\_train = (20256 + 3215)/(31647) # TP = 3215, TN = 20256 , FN = 439, FP = 7737

print(accuracy\_train) # 0.74

#################### Analysis on Test data #######################################

# Prediction on Test data set

test\_pred = logit\_model.predict(test\_data)

# Creating new column for storing predicted class of Attorney

# filling all the cells with zeroes

l\_test = len(test\_data["y"])

l\_test

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

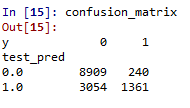
# taking threshold value as 'optimal\_threshold' and above the thresold prob value will be treated as 1

test\_data.loc[test\_pred > optimal\_threshold, "test\_pred"] = 1

# confusion matrix

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

confusion\_matrix

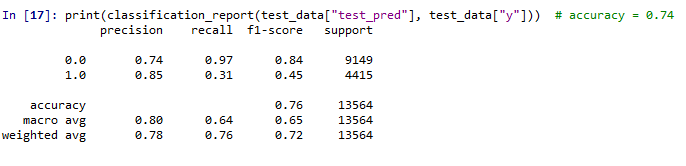


accuracy\_test = (8595 + 1409)/(13564) # TP = 1409, TN = 8595, FN = 226, FP = 3334

accuracy\_test # 0.74

# classification report

print(classification\_report(test\_data["test\_pred"], test\_data["y"])) # accuracy = 0.74

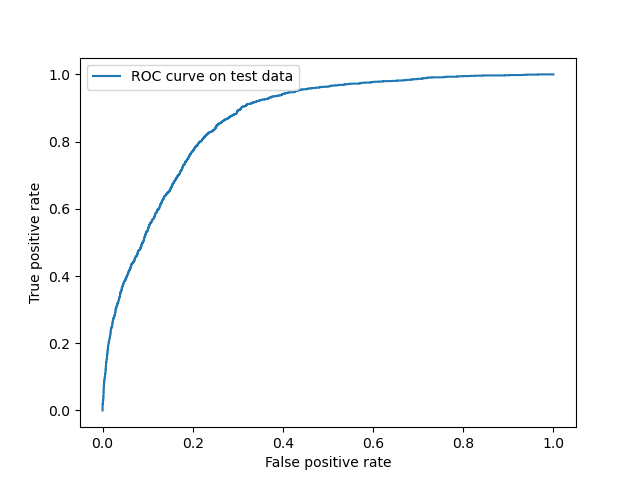


#ROC CURVE AND AUC

fpr, tpr, threshold = metrics.roc\_curve(test\_data["y"], test\_pred)

#PLOT OF ROC

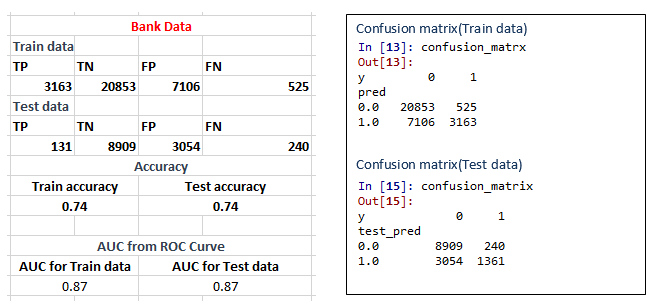
plt.plot(fpr, tpr, label = "ROC curve on test data");plt.xlabel("False positive rate");plt.ylabel("True positive rate");plt.legend()



roc\_auc\_test = metrics.auc(fpr, tpr)

roc\_auc\_test # area under the curve = 0.87 : fall in excellent region

**Summary:-**



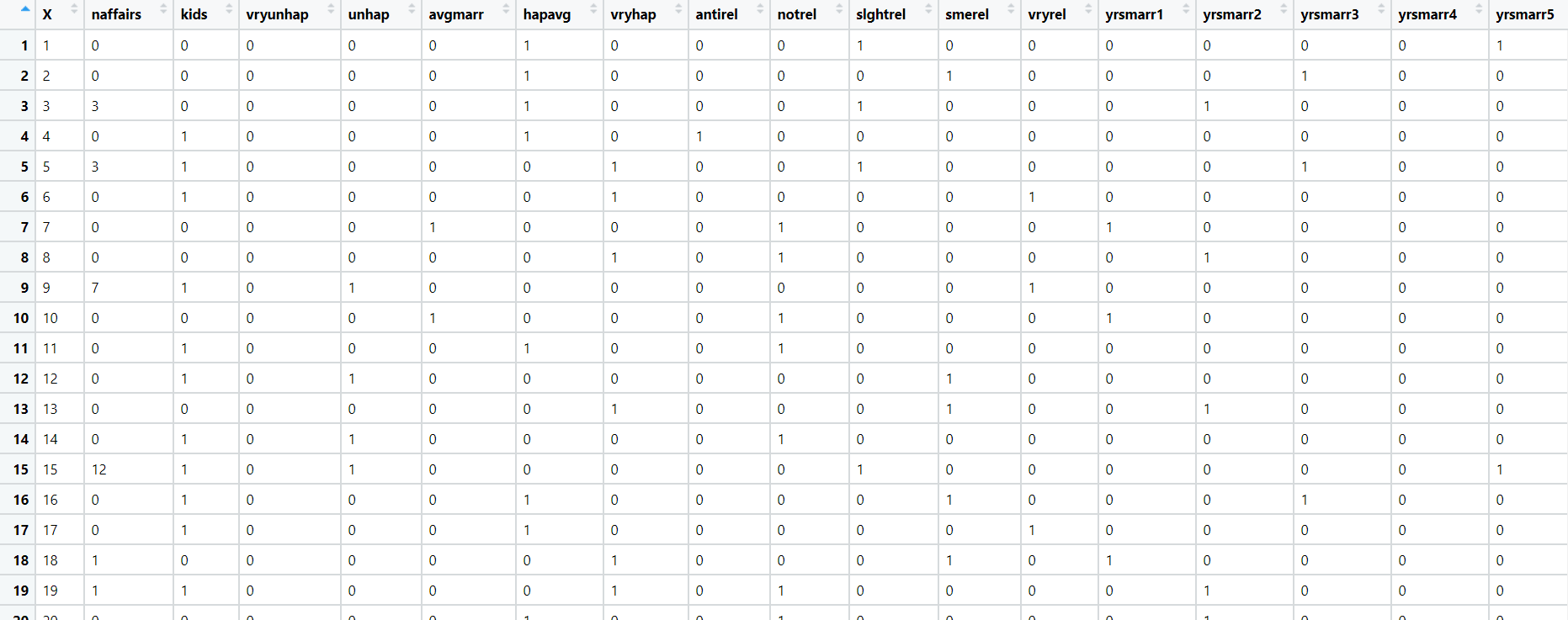
* Accuracy are almost equal and better for test & train data ( no over fitting & under fitting problem)
* AUC value are almost equal and better ( falls in excellent region) for test & train data. So threshold value choosen by fitting model on train data, gives better prediction on test data either. So model is good

**Business benefit:-**

Model will helps to predict whether a customer will put in a long-term fixed deposit or not based on the different variables given in the data. So without disturbing customers by asking question banks can use their deposited money for giving interest to other customers by knowing the duration they gonna deposit the money and when they gonna withdraw that. So bank doesn’t have to borrow mass money from any other financial agencies at high interest rate and at the same time ensure customers satisfaction by providing the money at the same time they wanna withdraw

Problem Statement: -

1. A psychological study has been conducted by a team of students at a university on married couples to determine the cause of having an extra marital affair. They have surveyed and collected a sample of data on which they would like to do further analysis. Apply Logistic Regression on the data to correctly classify whether a given person will have an affair or not given the set of attributes. Convert the naffairs column to discrete binary type before proceeding with the algorithm.

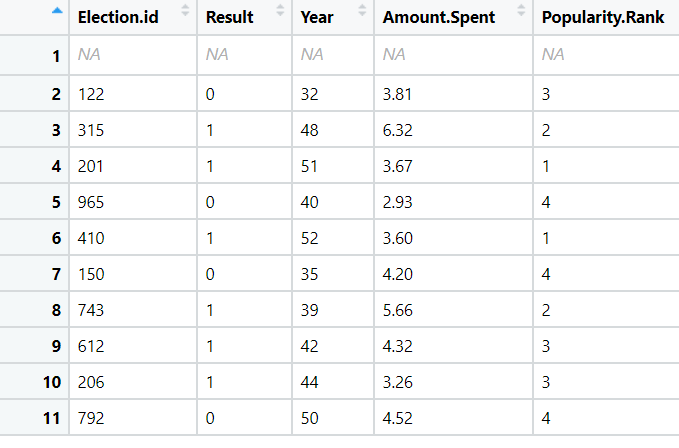


1. In this time and age of widespread internet usage, effective and targeted marketing plays a vital role. A marketing company would like to develop a strategy by analyzing their customer data. For this, data like age, location, time of activity, etc. has been collected to determine whether a user will click on an ad or not. Perform Logistic Regression on the given data to predict whether a user will click on an ad or not.

A screenshot of a cell phone

Description automatically generated

1. Perform Logistic Regression on the dataset to predict whether a candidate will win or lose the election based on factors like amount of money spent and popularity rank.



1. It is vital for banks that customers put in long term fixed deposits as they use it to pay interest to customers and it is not viable to ask every customer if they will put in a long-term deposit or not. So, build a Logistic Regression model to predict whether a customer will put in a long-term fixed deposit or not based on the different variables given in the data. The output variable in the dataset is Y which is binary. Snapshot of the dataset is given below.

**A picture containing large

Description automatically generated**