##################################################################################

############################# LOGISTIC ~ REGRESSION ##############################

##################################################################################

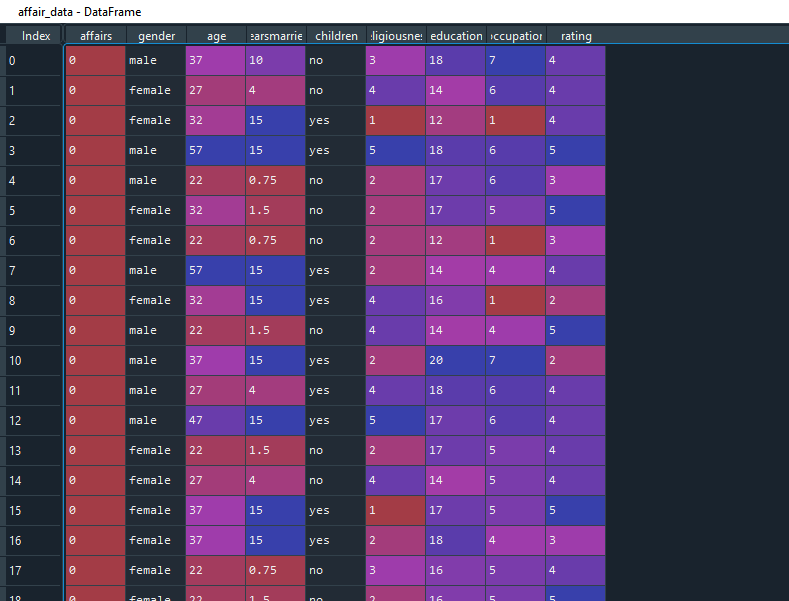
#### Importing packages and loading dataset ############

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

affair\_data= pd.read\_csv("C:\\Users\\home\\Desktop\\Data Science Assignments\\Python\_codes\Logistic\_Regression\\affairs.csv")



########### DATA CLEANING ########################

affair\_data.head(5)

affair\_data.isnull()#so there are no null or missing values in the dataset

affair\_data.duplicated(subset=None, keep='first')#there are no duplicate values

############### Exploratory data analysis(EDA) ################

affair\_data.describe()

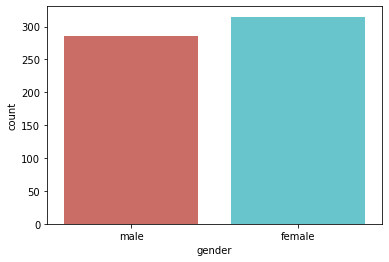


affair\_data.columns

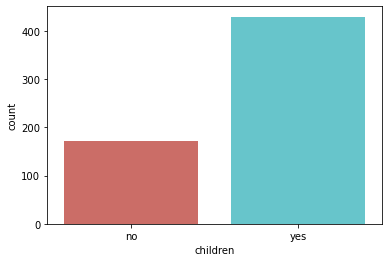
# Getting the barplot for the categorical columns

import seaborn as sb

sb.countplot(x="gender",data=affair\_data,palette="hls")



sb.countplot(x="children",data=affair\_data,palette="hls")



affair\_data["gender"].value\_counts()

#female 315

#male 286

affair\_data["age"].value\_counts()

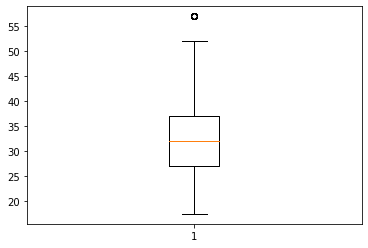
affair\_data["yearsmarried"].value\_counts()

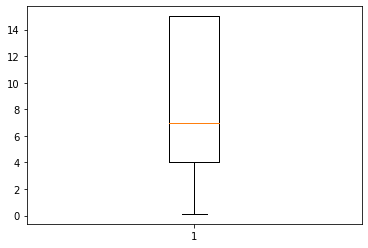
affair\_data["religiousness"].value\_counts()

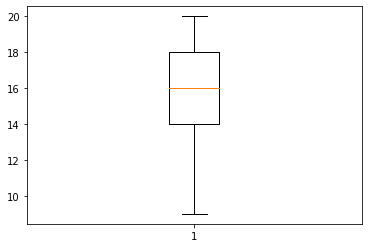
affair\_data["education"].value\_counts()

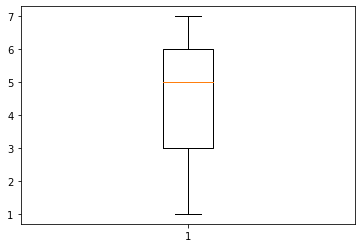
affair\_data["occupation"].value\_counts()

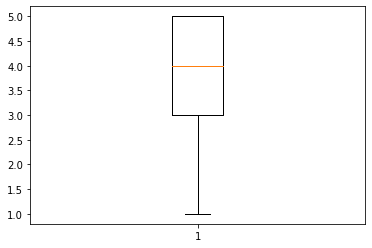
affair\_data["rating"].value\_counts()

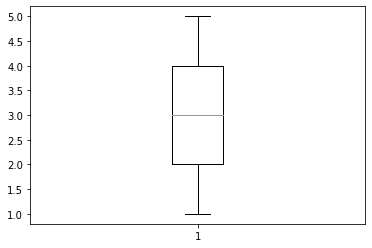
plt.boxplot(affair\_data["age"])

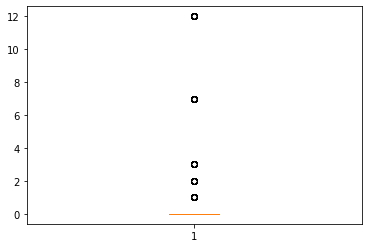
plt.boxplot(affair\_data["yearsmarried"])

plt.boxplot(affair\_data["education"])

plt.boxplot(affair\_data["occupation"])

plt.boxplot(affair\_data["rating"])

plt.boxplot(affair\_data["religiousness"])

plt.boxplot(affair\_data["affairs"])

#calculating the interquantile range

q25 = affair\_data.quantile(0.25)#lower 25%

q75 = affair\_data.quantile(0.75)#upper 25%

iqr= q75-q25 #50% of the data

lower\_bound = q25 - (1.5 \* iqr)

upper\_bound = q75 + (1.5\* iqr)

out\_25= (affair\_data < (q25 - (1.5 \* iqr)))

out\_75 = (affair\_data> (q75+(1.5 \* iqr)))

#we now have the IQR scores, it’s time to get hold on outliers.

#The above code will give an output with some true and false values.

#The data point where we have False that means these values are valid whereas True indicates presence of an outlier.

##Checking for outliers below lower\_bound.

out\_25["age"].value\_counts()

out\_25["affairs"].value\_counts()

out\_25["children"].value\_counts()

out\_25["education"].value\_counts()

out\_25["gender"].value\_counts()

out\_25["occupation"].value\_counts()

out\_25["rating"].value\_counts()

out\_25["religiousness"].value\_counts()

out\_25["yearsmarried"].value\_counts()

## There are no outliers below lower bound.

## Checking for outliers above upper bound

out\_75["age"].value\_counts() ## 22 are outliers

out\_75["affairs"].value\_counts() ## 150 are outliers

out\_75["children"].value\_counts()

out\_75["education"].value\_counts()

out\_75["gender"].value\_counts()

out\_75["occupation"].value\_counts()

out\_75["rating"].value\_counts()

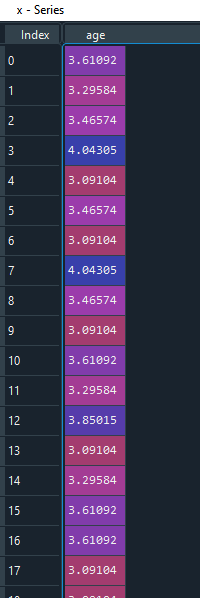
out\_75["religiousness"].value\_counts()

out\_75["yearsmarried"].value\_counts()

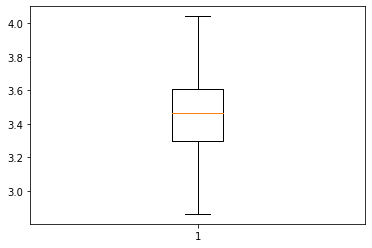
## There are no outliers above upper bound.

## log transformation for age, to convert the outliers

x= np.log(affair\_data["age"])



plt.boxplot(x)



q\_25 = np.log(affair\_data["age"]).quantile(0.25)

q\_75 = np.log(affair\_data["age"]).quantile(0.75)

iqr2 = q\_75-q\_25

lower\_bound\_one = q\_25 - (1.5 \* iqr2)

upper\_bound\_one = q\_75 + (1.5\* iqr2)

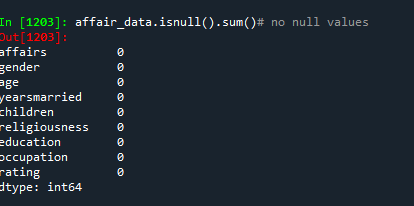
out\_25\_one= (np.log(affair\_data["age"]) < lower\_bound\_one)

out\_75\_one = (np.log(affair\_data["age"])> upper\_bound\_one)

out\_25\_one.value\_counts()## no outliers

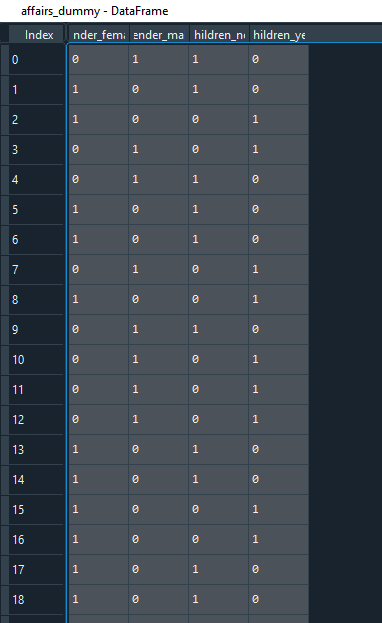
out\_75\_one.value\_counts()## no outliers

affair\_data.isnull().sum()# no null values



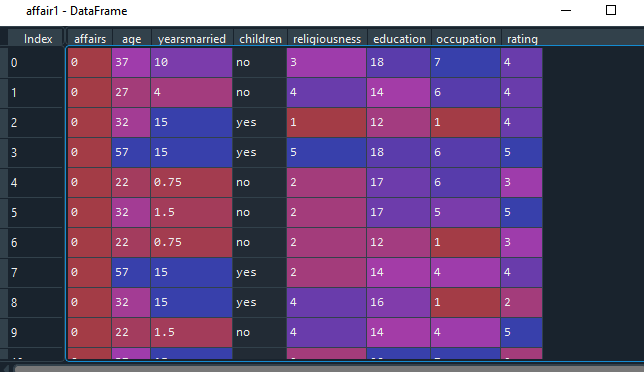
## Creating dummies for gender and children

affairs\_dummy = pd.get\_dummies(affair\_data[["gender","children"]])

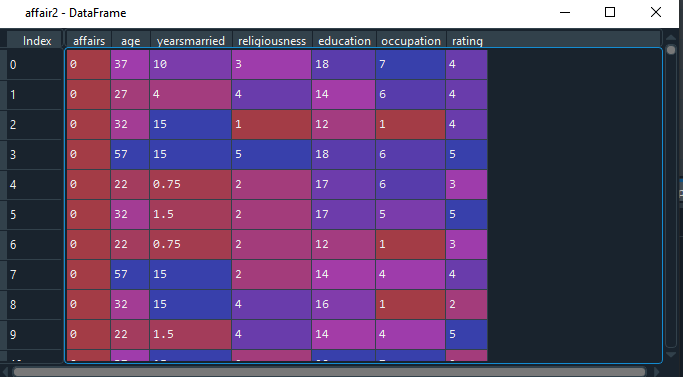


#affair2 =pd.get\_dummies(affair,drop\_first =True)

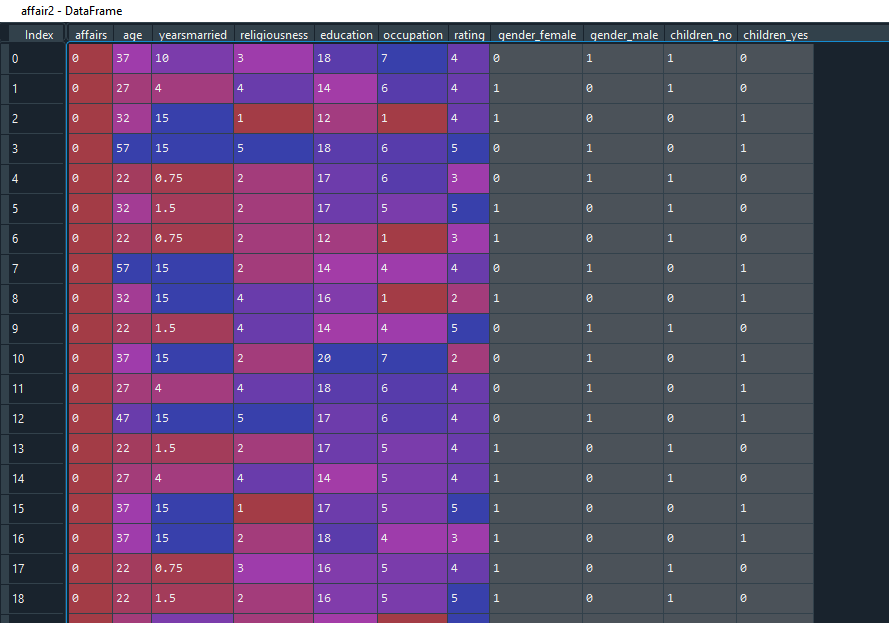
affair1= affair\_data.drop(["gender"], axis=1)



affair2 = affair1.drop(["children"],axis=1)

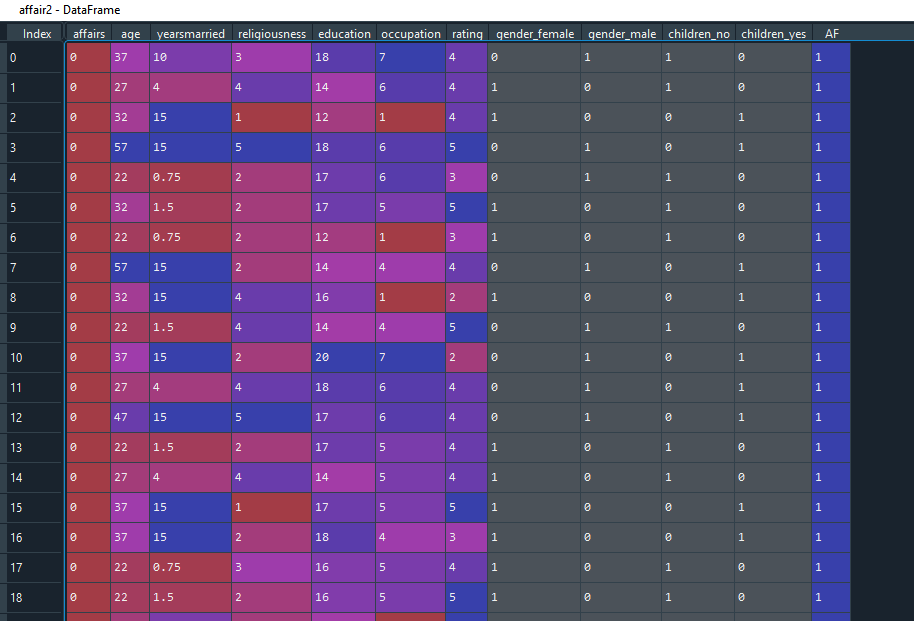


affair2= pd.concat([affair2,affairs\_dummy],axis=1)



affair2["AF"]=1

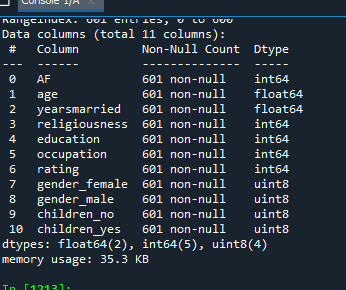
affair2.loc[affair2.affairs==0,"AF"]=0



affair2=affair2.drop(["affairs"],axis=1)

affair2=affair2.iloc[:,[10,0,1,2,3,4,5,6,7,8,9]]

affair2.info()



## coverting religiousness and rating into factors, as it is ordinal data

affair2["religiousness"]= pd.Categorical(affair2["religiousness"])

affair2["rating"]=pd.Categorical(affair2["rating"])

#affair2["occupation"]=pd.Categorical(affair2["occupation"])

#affair2["gender\_male"]=pd.Categorical(affair2["gender\_male"])

#affair2["children\_yes"]= pd.Categorical(affair2["children\_yes"])

#creating the model

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

train\_data,test\_data = train\_test\_split(affair2,test\_size = 0.2)

train\_data.to\_csv("training\_data.csv",encoding="utf-8")

test\_data.to\_csv("testing\_data.csv",encoding= "utf-8")

import statsmodels.formula.api as sm

## trailone

model\_one = sm.logit("AF~np.log(age)+yearsmarried+religiousness+rating+occupation+education+gender\_female+gender\_male+children\_no+children\_yes", data= train\_data).fit()

model\_one.summary()

model\_one.summary2()

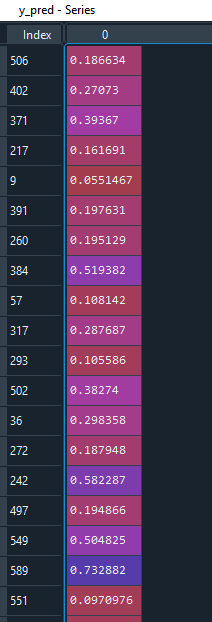
## AIC= 503.2311

from scipy import stats

import scipy.stats as st

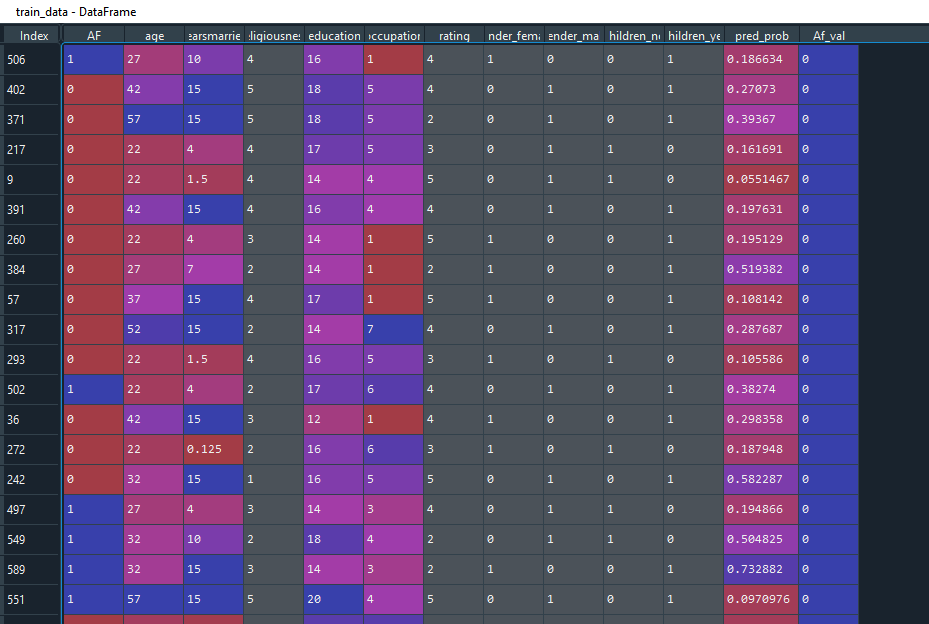
st.chisqprob = lambda chisq, df:stats.chi2.sf(chisq,df)

y\_pred = model\_one.predict(train\_data)



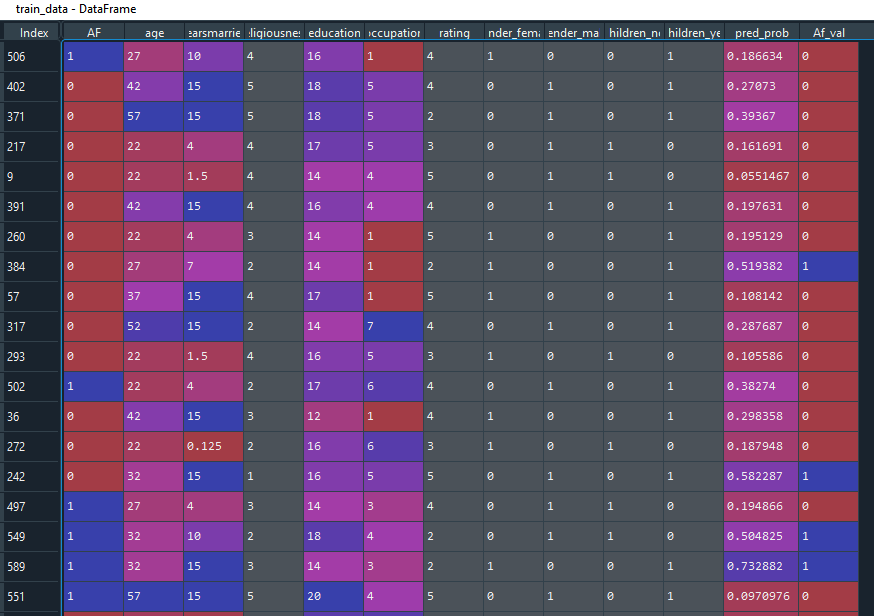
train\_data["pred\_prob"]=y\_pred

train\_data["Af\_val"]=np.zeros(480)



train\_data.loc[y\_pred>=0.50,"Af\_val"]=1

train\_data.Af\_val

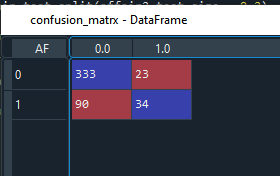


from sklearn.metrics import classification\_report

classification = classification\_report(train\_data["Af\_val"],train\_data["AF"])

#confusion matrix

confusion\_matrx = pd.crosstab(train\_data["AF"],train\_data["Af\_val"])



##accuracy

accuracy = (333+34)/(333+34+23+90) ##0.7645833333333333

76.5%

##ROC curve

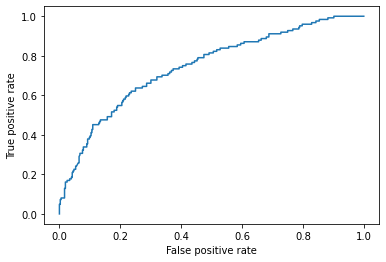
from sklearn import metrics

#fpr=> false positive rate

#tpr=> true positive rate

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

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



roc\_auc = metrics.auc(fpr,tpr) ## 0.7436004893077202

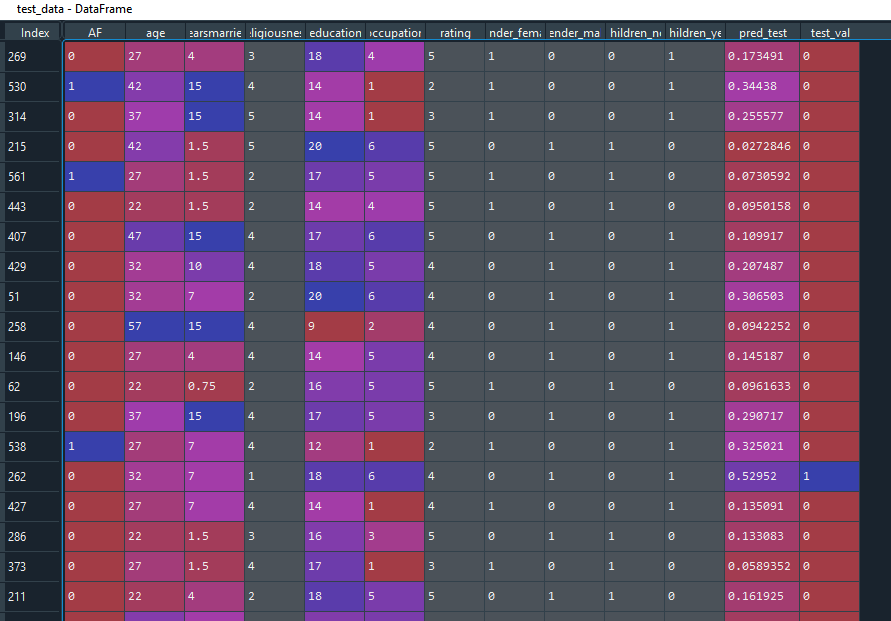
## test data

test\_pred = model\_one.predict(test\_data)

test\_data["pred\_test"] = test\_pred

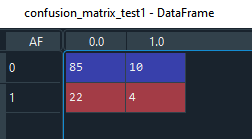
test\_data["test\_val"]= np.zeros(121)

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



## confusion matrix

confusion\_matrix\_test1 = pd.crosstab(test\_data["AF"],test\_data["test\_val"])



##accuracy

accuracy\_test= (85+4)/(85+4+22+10) ## 0.7355371900826446

############ trails for perfect cut-off value 0.58

y\_pred1 = model\_one.predict(train\_data)

train\_data["pred\_prob1"]=y\_pred1

train\_data["Af\_val1"]=np.zeros(480)

train\_data.loc[y\_pred1>=0.58,"Af\_val1"]=1

train\_data.Af\_val1

train\_data["Af\_val1"].value\_counts()

#0.0 447

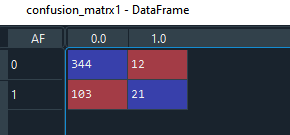
#1.0 33

from sklearn.metrics import classification\_report

classification = classification\_report(train\_data["Af\_val1"],train\_data["AF"])

#confusion matrix

confusion\_matrx1= pd.crosstab(train\_data["AF"],train\_data["Af\_val1"])

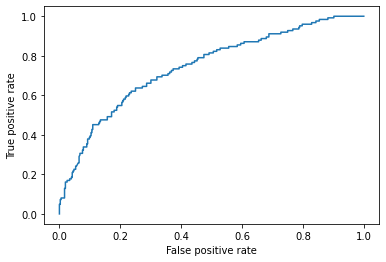


##accuracy

accuracy1 = (344+21)/(344+21+12+103)## 0.7604166666666666

fpr2,tpr2,threshold2 = metrics.roc\_curve(train\_data["AF"], y\_pred1)

plt.plot(fpr2,tpr2);plt.xlabel("False positive rate");plt.ylabel("True positive rate")



roc\_auc1 = metrics.auc(fpr, tpr) ##0.7436004893077202

## test data

test\_pred1 = model\_one.predict(test\_data)

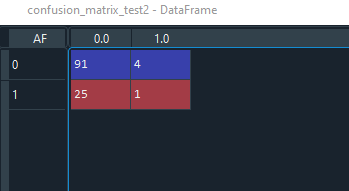
test\_data["pred\_test1"] = test\_pred1

test\_data["test\_val1"]= np.zeros(121)

test\_data.loc[test\_pred>=0.58,"test\_val1"]=1

## confusion matrix

confusion\_matrix\_test2 = pd.crosstab(test\_data["AF"],test\_data["test\_val1"])



##accuracy

accuracy\_test1= (91+1)/(25+4+91+1) ## 0.7603305785123967

########################################################trail two##############################################

### as all the variables are insignificant i would remove education, occupation, gender and children to build the model.

model\_two = sm.logit("AF~np.log(age)+yearsmarried+religiousness+rating", data = train\_data).fit()

model\_two.summary()

model\_two.summary2()

#AIC: 500.3136

y\_pred2 = model\_two.predict(train\_data)

train\_data["pred\_prob2"]=y\_pred2

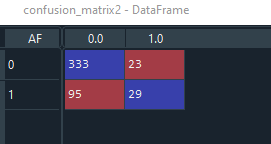
train\_data["Af\_val2"]=np.zeros(480)

train\_data.loc[y\_pred2>=0.5,"Af\_val2"]=1

classification1 = classification\_report(train\_data["Af\_val2"],train\_data["AF"])

## confusion matrix

confusion\_matrix2 = pd.crosstab(train\_data["AF"], train\_data["Af\_val2"])



## accuracy

accuracy2 = (333+29)/(333+29+23+95)

##0.7541666666666667

fpr3,tpr3, threshold3 = metrics.roc\_curve(train\_data["AF"],y\_pred2)

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

roc\_auc2 = metrics.auc(fpr,tpr)## 0.7436004893077202

## for test data

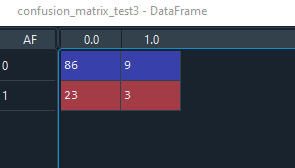
test\_pred2= model\_two.predict(test\_data)

test\_data["pred\_test2"]=test\_pred2

test\_data["test\_val2"]= np.zeros(121)

test\_data.loc[test\_pred2>=0.5,"test\_val2"]=1

confusion\_matrix\_test3 = pd.crosstab(test\_data["AF"],test\_data["test\_val2"])



accuracy2 = (86+3)/(86+3+23+9) ## 0.7355371900826446

## Checking for different cut off values 0.58

y\_pred3 = model\_two.predict(train\_data)

train\_data["pred\_prob3"]=y\_pred3

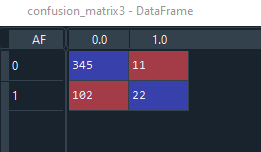
train\_data["Af\_val3"]= np.zeros(480)

train\_data.loc[y\_pred3>=0.54,"Af\_val3"]=1

classification2 = classification\_report(train\_data["Af\_val3"],train\_data["AF"])

# confusion matrix

confusion\_matrix3 = pd.crosstab(train\_data["AF"],train\_data["Af\_val3"])



accuracy3= (345+22)/(345+22+11+102)

## 0.7645833333333333

fpr4,tpr4,threshold4 = metrics.roc\_curve(train\_data["AF"],y\_pred3)

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

roc\_auc3 = metrics.auc(fpr,tpr) ## 0.7436004893077202

## test data

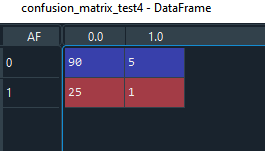
test\_pred3 = model\_two.predict(test\_data)

test\_data["pred\_test3"]=test\_pred3

test\_data["test\_val3"] = np.zeros(121)

test\_data.loc[test\_pred3>=0.54,"test\_val3"]=1

confusion\_matrix\_test4= pd.crosstab(test\_data["AF"],test\_data["test\_val3"])



accuracy\_test3 = (90+1)/(90+1+25+5) ## 0.7520661157024794

###model\_two is selected and with the cut off value as 0.54. As it has least fpr and high tpr.