1. **Case study 1**

**Income prediction for the given data set using python?**

**Objective:**

* To predict individuals incomeusing Logistic Regression in python.

**Process:**

* Get the data set(population).
* Clean the data.(population).
* Find and fill the missing values(population).
* Check outliers(for dependent variables).
* After fill missing values take random sample from the population.
* Write the sample data into a CSV file(Easy to handle).
* Read the sample data from CSV file.
* Make it as a data frame.
* Check if there is any missing values.
* Calculate basic descriptive statistic for sample data.
* Check correlation of whole data frame.
* Take the variables which has the highly correlated with y(target) variable.
* Correlation range must lies in between -1 to 1.
* Take X variables and y variable.
* Split X and y into train and test data sat.
* Import logistic regression from sklearn library.
* Build the regression model.
* Fit the X\_train and y\_train data in to the model.
* Make predictions.
* Calculate the coefficients, intercept,confusion matrix by using sklearn.metrics library.
* Based on the confusion matrix we can calculate the accuracy,specificity and sensitivity also.

**Input:**

* Data sample(300 from population).

**Output:**

* Regression coefficients.
* Regression intercept.
* Confusion matrix.

**Source code:**

#import libraries

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

from sklearn.metrics import classification\_report

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#Take sample from population

#read the data sample

data=pd.read\_csv('/home/soft23/soft23/Sathish/Spyder workings/sample.csv')

df=pd.DataFrame(data)

print("Actual Data frame is:\n",df.head(10))

#checking missing values

print("Checking missing values in the sample")

print(df.isnull().sum())

print("\n")

print("Descriptive statistics")

print(df.describe())

print("\n")

print("Correlation is")

print(df.corr(method='pearson'))

#Depends upon the correlation choose X variable

X=df[['hoursperweek','relationship','EdType']]

#Fix the target variable

y=(df['SalStat'])

#Split the data into train and test data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=0)

print("Shape of train data of X\n",X\_train.shape)

print("Shape of train data of y\n",y\_train.shape)

#build the model

logmodel = LogisticRegression()

result=logmodel.fit(X\_train,y\_train)

#predictions

y\_pred = logmodel.predict(X\_test)

df1=pd.DataFrame({'Actual':y\_test, 'Predicted': y\_pred})

print(df1)

#regression c-efficients and intercept

print("Regression intercept is",logmodel.intercept\_)

print("Regression coefficient is",logmodel.coef\_)

#classification report

print(classification\_report(y\_test,y\_pred))

cm = metrics.confusion\_matrix(y\_test, y\_pred)

print("The confusion matrix is:\n",cm)

#finding score of the model

print("Model score")

score=result.score(X\_train,y\_train)

print(score)

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred))

print("Recall:",metrics.recall\_score(y\_test, y\_pred))

class\_names=[0,1]

fig, ax = plt.subplots()

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks,class\_names)

plt.yticks(tick\_marks,class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

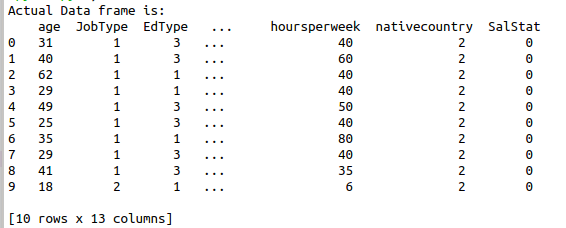
plt.tight\_layout()

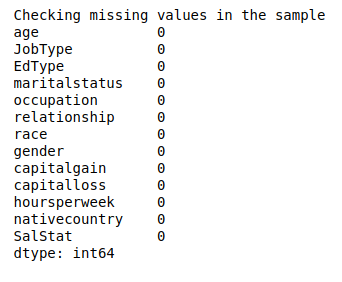
plt.title('Confusion matrix')

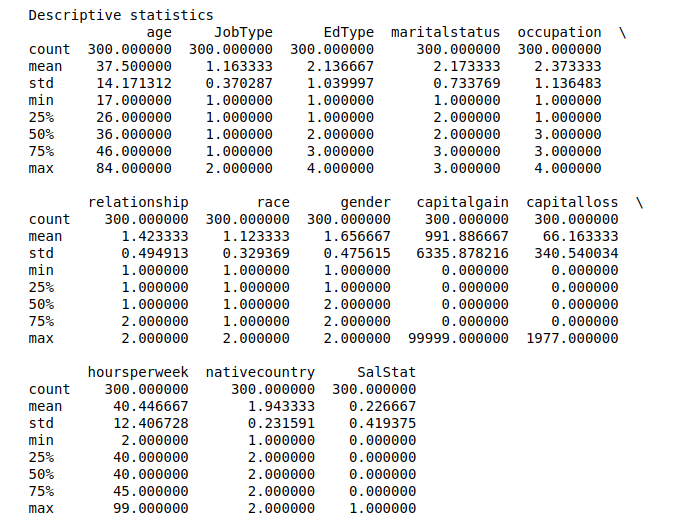
plt.ylabel('Actual label')

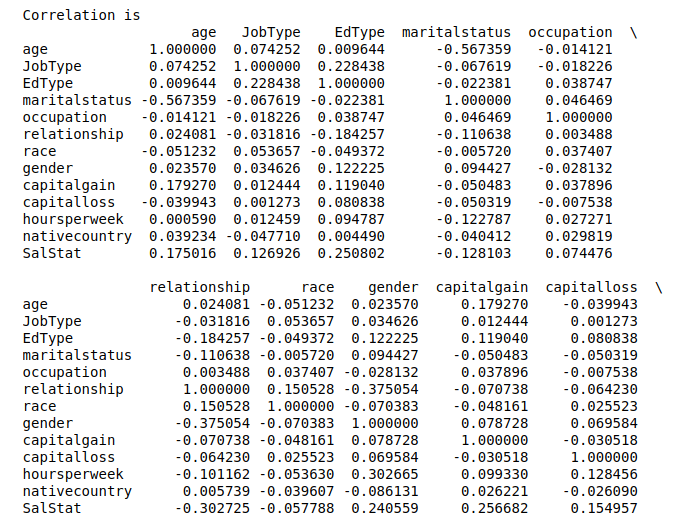
plt.xlabel('Predicted label')

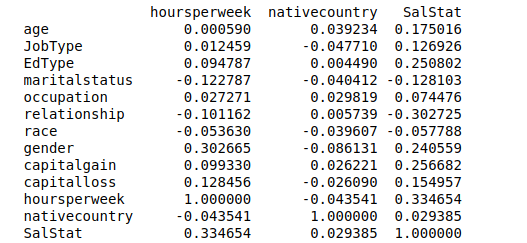
**Screen shots:**

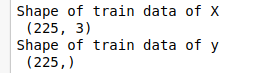
****

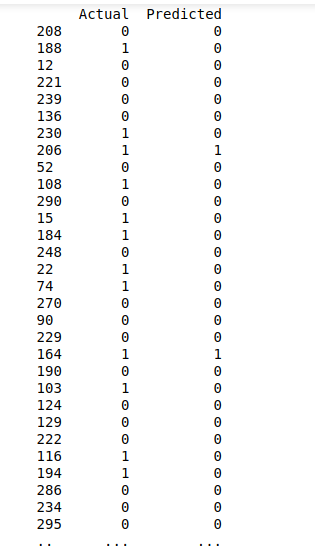
****

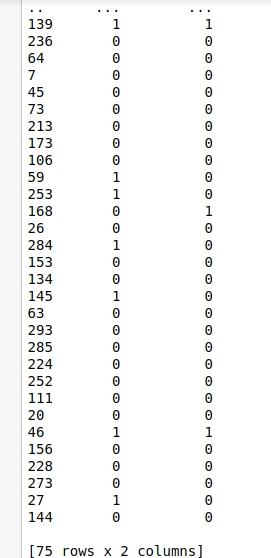
****

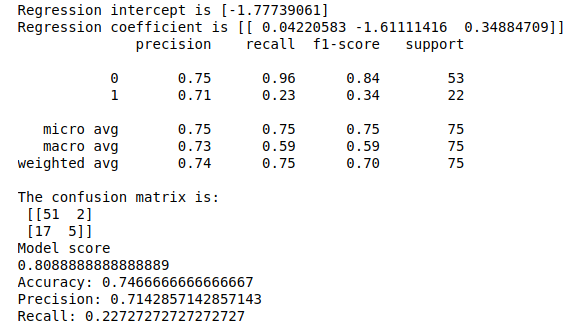
****

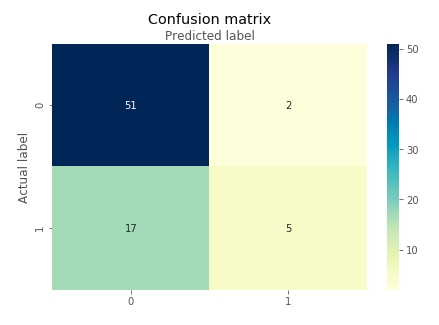
****

****

****

****

****

****