**Assignment 1**

**Code and Output:-**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import math

from sklearn.cluster import KMeans

## Create Dataset

X = [[0.1,0.6],[0.15,0.71],[0.08,0.9],[0.16,0.85],[0.2,0.3],[0.25,0.5],[0.24,0.1],[0.3,0.2]]

df = np.array(X)

print('\nDataset:-',df)

centroids = np.array([[0.1,0.6],[0.25,0.5]])

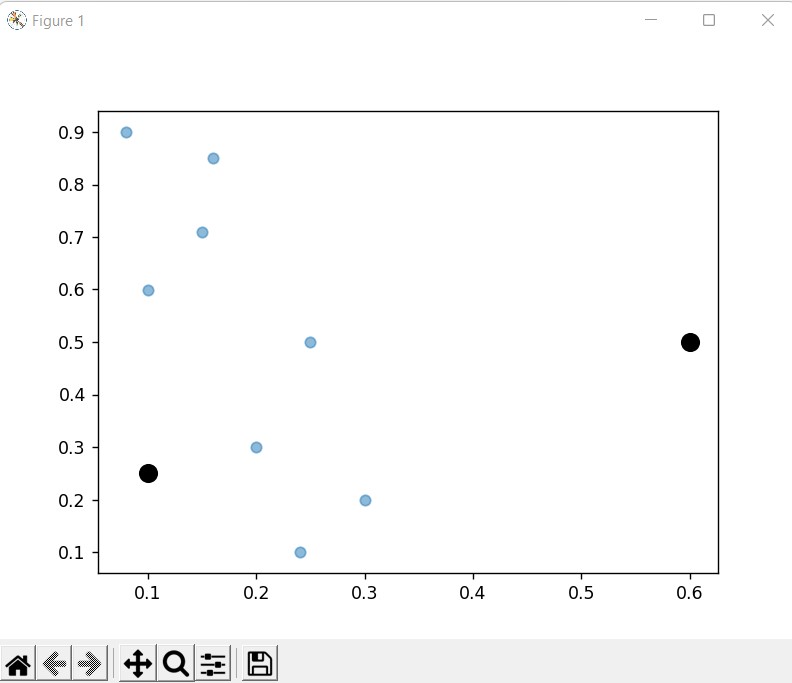
print('\nCenteroids:-',centroids)

# Data Points

plt.figure()

plt.scatter(df[:,0],df[:,1])

plt.show()



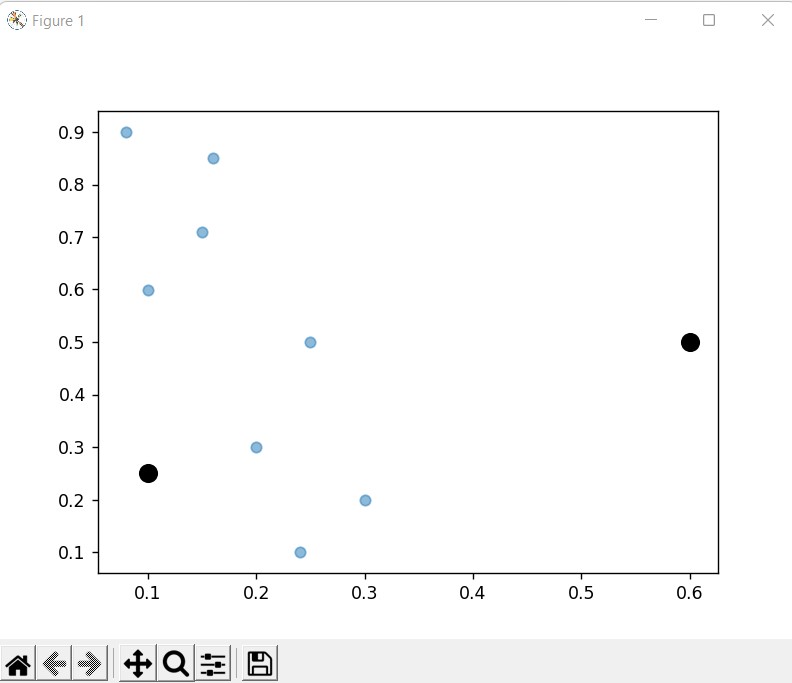
# Data Points with two clusters centroids

plt.figure()

plt.scatter(df[:,0],df[:,1],alpha = 0.5)

plt.scatter(centroids[0],centroids[1],color = 'black', marker='o', s=100)

plt.show()



kmeans=KMeans(n\_clusters=2,init=centroids)

kmeans.fit(X)

print("Labels after trainig:",kmeans.labels\_)

#Q1

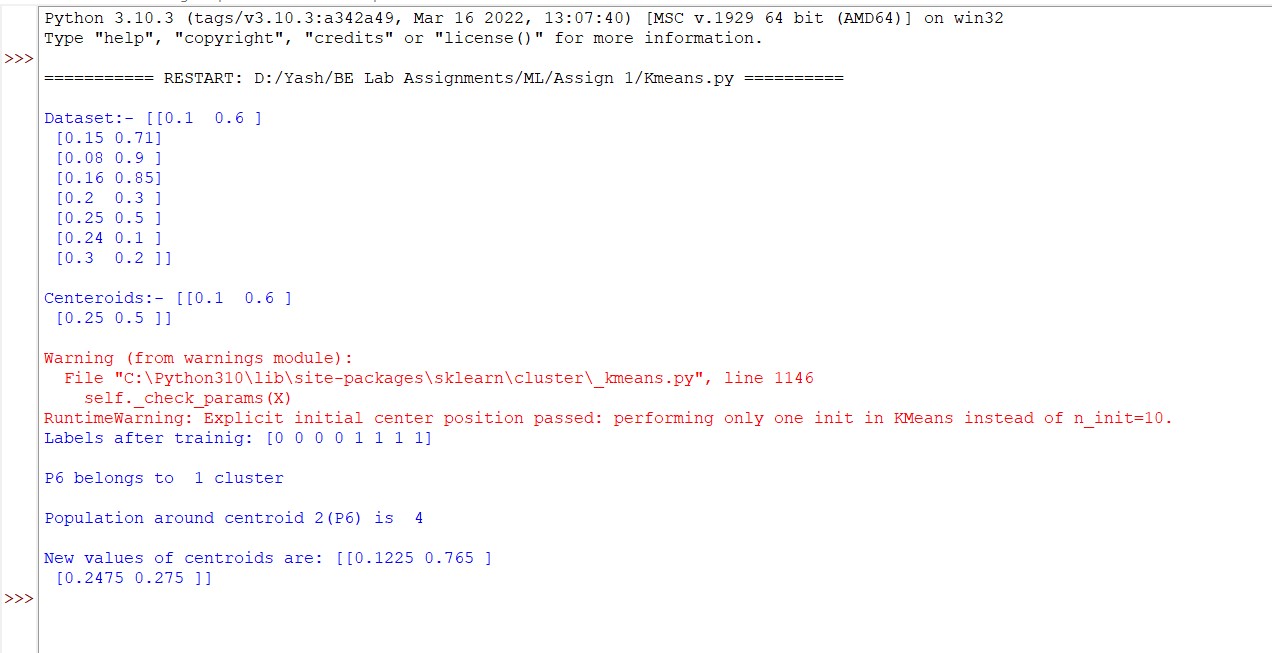
print("\nP6 belongs to ",kmeans.labels\_[5],"cluster")

#Q2

print("\nPopulation around centroid 2(P6) is ",np.count\_nonzero(kmeans.labels\_==1))

#Q3

print("\nNew values of centroids are:",kmeans.cluster\_centers\_)



**#IRIS Dataset**

from sklearn import datasets

iris=datasets.load\_iris()

iris\_x=iris.data

wcss=list()

for i in range(1,11):

kmeans=KMeans(n\_clusters=i,init='k-means++')

kmeans.fit(iris\_x)

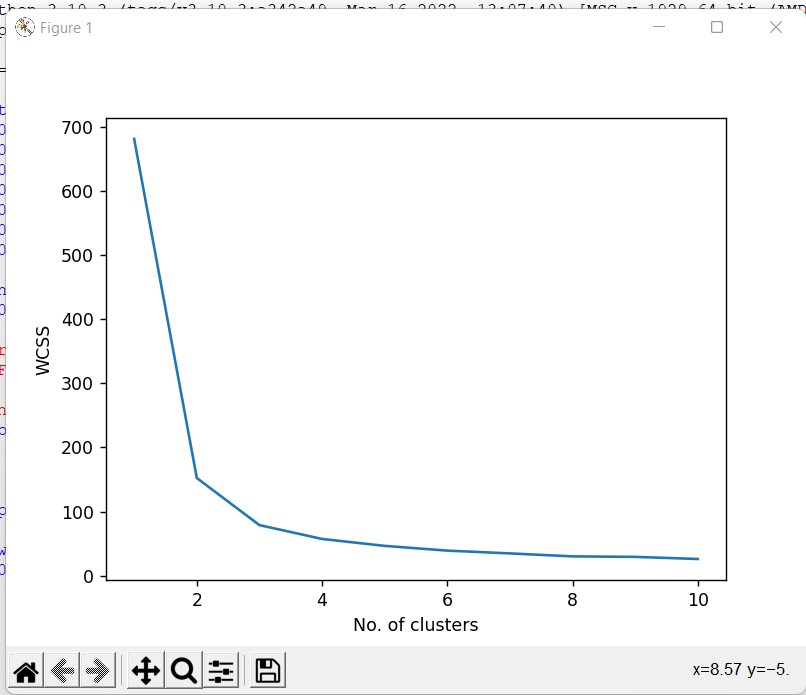
wcss.append(kmeans.inertia\_)

plt.plot(range(1,11),wcss)

plt.xlabel('No. of clusters')

plt.ylabel('WCSS')

plt.show()



kmeans=KMeans(n\_clusters=3,init='k-means++')

kmeans.fit(iris\_x)

y\_means=kmeans.predict(iris\_x)

plt.scatter(iris\_x[y\_means==0,0],iris\_x[y\_means==0,1],c='blue',s=100,label='iris\_sertosa')

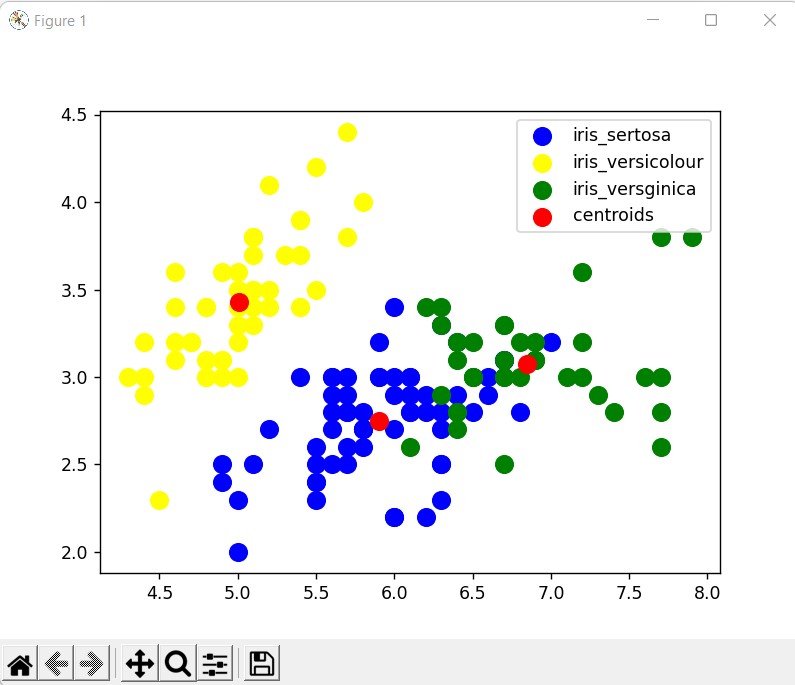
plt.scatter(iris\_x[y\_means==1,0],iris\_x[y\_means==1,1],c='yellow',s=100,label='iris\_versicolour')

plt.scatter(iris\_x[y\_means==2,0],iris\_x[y\_means==2,1],c='green',s=100,label='iris\_versginica')

plt.scatter(kmeans.cluster\_centers\_[:,0],kmeans.cluster\_centers\_[:,1],c='red',s=100,label='centroids')

plt.legend()

plt.show()

****

**Assignment 2**

**Code and Output:-**

import pandas as pd

import numpy as np

dataset = pd.read\_csv("dataset.csv")

X = dataset.iloc[:,:-1].values

y = dataset.iloc[:,2].values

print("X: ",X)

print("Y: ",y)

# Import KNN

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=3)

classifier.fit(X,y)

# Predict the class of point (6,6)

x\_test = np.array([6,6])

y\_pred = classifier.predict([x\_test])

ans = ""

if y\_pred[0] =='negative':

ans = "orange"

else:

ans = "blue"

print('\nGeneral KNN : ', y\_pred[0],'(', ans, ')')

# Distance Weighted KNN

classifier = KNeighborsClassifier(n\_neighbors=3, weights='distance')

classifier.fit(X,y)

# Predict the class of point (6,6)

x\_test = np.array([6,6])

y\_pred = classifier.predict([x\_test])

ans = ""

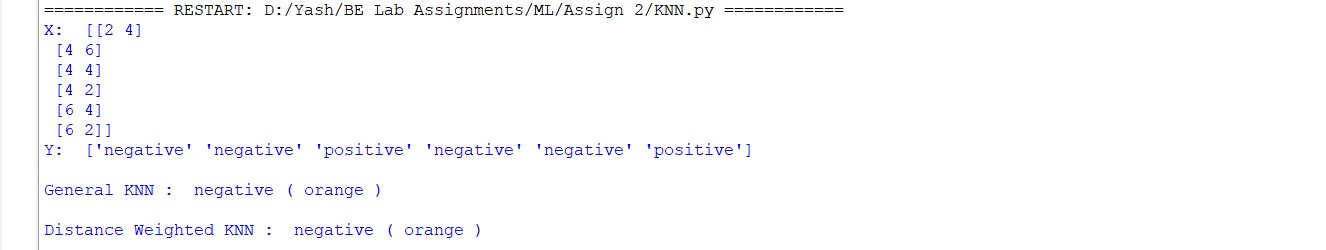
if y\_pred[0] =='negative':

ans = "orange"

else:

ans = "blue"

print('\nDistance Weighted KNN : ', y\_pred[0],'(', ans, ')')



# Using Iris Dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import preprocessing

from sklearn.metrics import confusion\_matrix, accuracy\_score

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.datasets import load\_iris

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['Species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

print("\nIris Dataset:-\n",df)

df['Species'].value\_counts()

df.isnull().sum()

X = df.iloc[:,:4]

X = preprocessing.StandardScaler().fit\_transform(X)

y = df['Species']

X\_train,X\_test, y\_train,y\_test = train\_test\_split(X,y,test\_size=0.3, random\_state=1)

knn = KNeighborsClassifier(n\_neighbors=3)

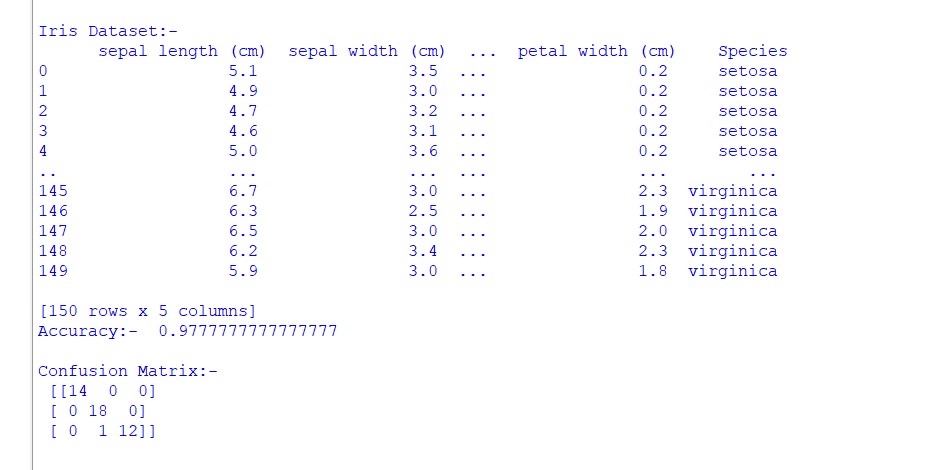
knn.fit(X\_train,y\_train)

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

acc = accuracy\_score(y\_test,y\_pred)

print("Accuracy:- ", acc)

conf\_matrix = confusion\_matrix(y\_test,y\_pred)



sns.heatmap(conf\_matrix,annot=True,fmt="d")

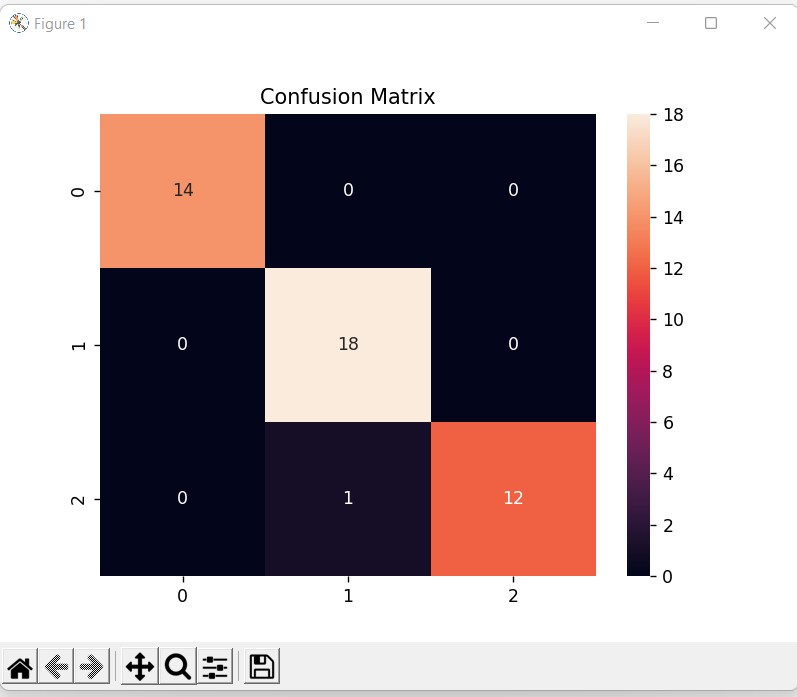
plt.ylabel = "Actual"

plt.xlabel = "Predicted"

plt.title("Confusion Matrix")

print("\nConfusion Matrix:- \n",conf\_matrix)

plt.show()



**Assignment 4**

**Code and Output:-**

import numpy as np

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import export\_graphviz

from IPython.display import Image

data = pd.read\_csv("data.csv")

print('Dataset: \n',data)

le=LabelEncoder();

x=data.iloc[:,:-1]

x=x.apply(le.fit\_transform)

print("Age:",list( zip(data.iloc[:,0], x.iloc[:,0])))

print("\nIncome:",list( zip(data.iloc[:,1], x.iloc[:,1])))

print("\nGender:",list( zip(data.iloc[:,2], x.iloc[:,2])))

print("\nmaritialStatus:",list( zip(data.iloc[:,3], x.iloc[:,3])))

print("\nX: \n",x)

y=data.iloc[:,-1]

print("Y: \n",y)

dt=DecisionTreeClassifier()

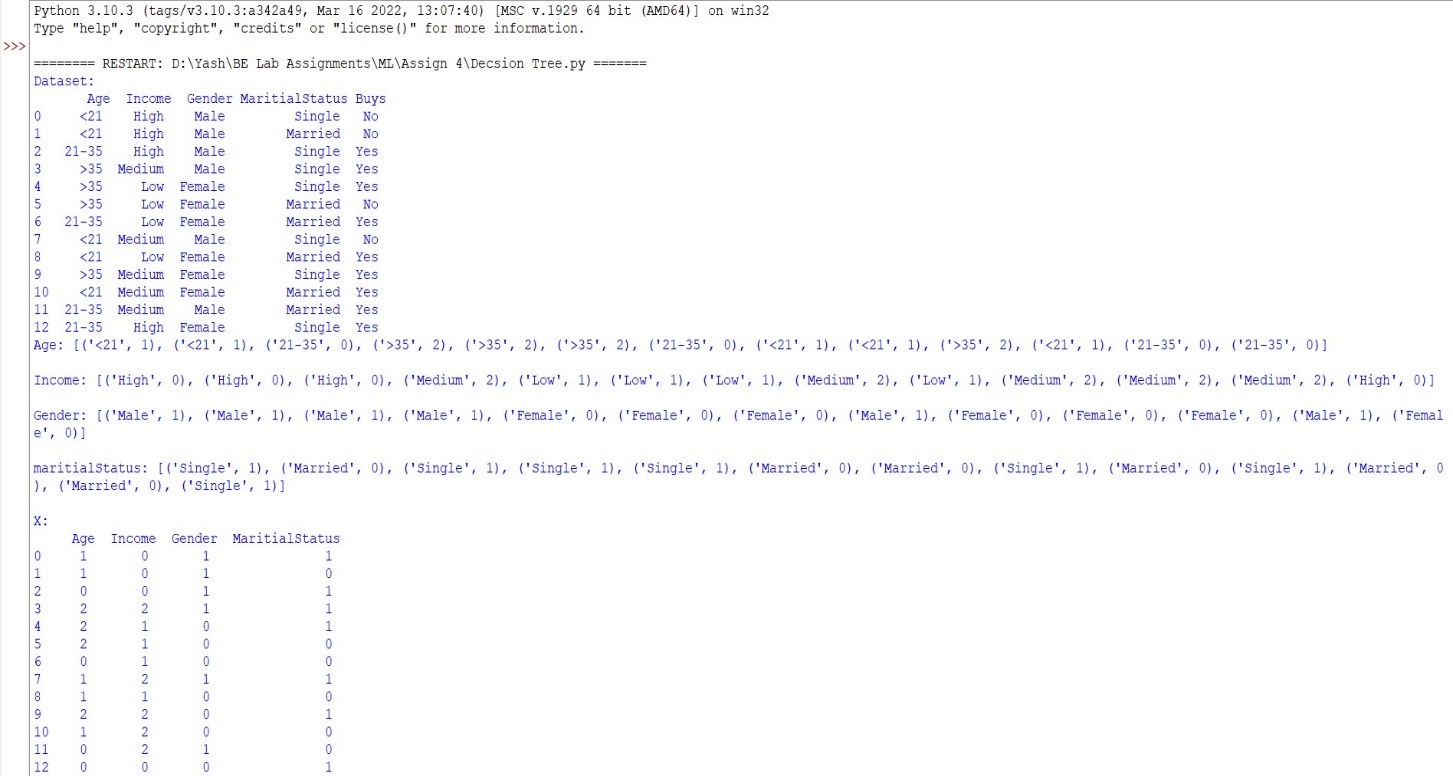
dt.fit(x,y)

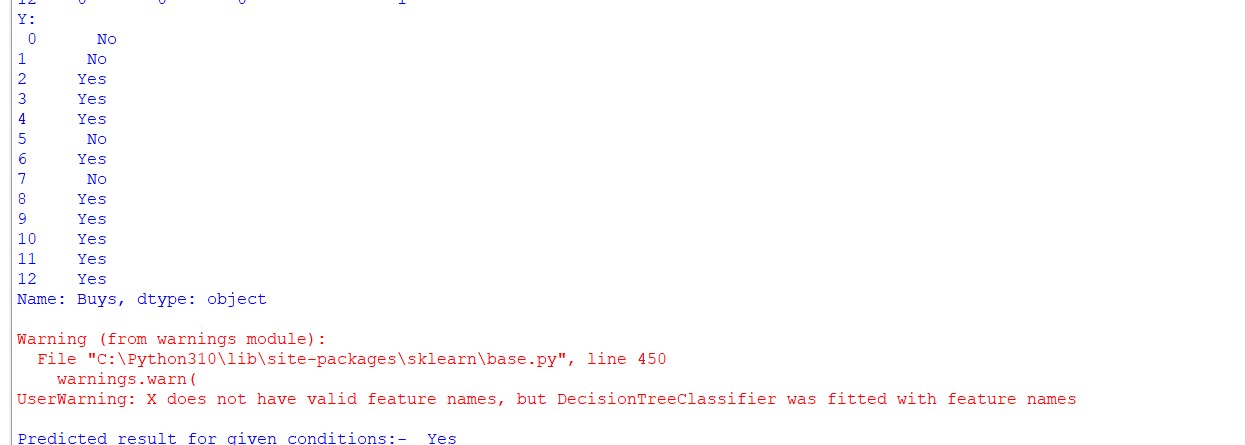
#[Age < 21, Income = Low,Gender = Female, Marital Status = Married]

query=np.array([1,1,0,0])

pred=dt.predict([query])

print("\nPredicted result for given conditions:- ",pred[0])





#Titanic Dataset

import numpy as np

import pandas as pd

from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import export\_graphviz

from IPython.display import Image

from sklearn.compose import make\_column\_transformer

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read\_csv("titanic.csv")

print('Dataset: \n',df)

print('Dataset Description:-',df.describe())

#filling age and embarked null values

cols = ['Pclass', 'Sex']

age\_class\_sex = df.groupby(cols)['Age'].mean().reset\_index()

df['Age'] = df['Age'].fillna(df[cols].reset\_index().merge(age\_class\_sex, how='left', on=cols).set\_index('index')['Age'])

df['Embarked'] = df['Embarked'].fillna('S')

#converting data attributes into categorial numerical form

df['Cabin'] = df["Cabin"].apply(lambda x: 0 if type(x) == float else 1)

df['Embarked'] = df['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

df['Sex'] = df['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

df.loc[ df['Fare'] <= 7.91, 'Fare'] = 0

df.loc[(df['Fare'] > 7.91) & (df['Fare'] <= 14.454), 'Fare'] = 1

df.loc[(df['Fare'] > 14.454) & (df['Fare'] <= 31), 'Fare'] = 2

df.loc[ df['Fare'] > 31, 'Fare'] = 3

df['Fare'] = df['Fare'].astype(int)

df.loc[ df['Age'] <= 16, 'Age'] = 0

df.loc[(df['Age'] > 16) & (df['Age'] <= 32), 'Age'] = 1

df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'Age'] = 2

df.loc[(df['Age'] > 48) & (df['Age'] <= 64), 'Age'] = 3

df.loc[ df['Age'] > 64, 'Age'] = 4;

df['Age'] = df['Age'].astype(int)

y = df['Survived']

x = df.drop(['Survived'], axis=1).values

x\_features = df.iloc[:,1:]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.33, random\_state=1)

dt=DecisionTreeClassifier()

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

y\_pred = dt.predict(x\_test)

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

res = pd.DataFrame(list(zip(y\_test, y\_pred)), columns =['Actual', 'Predicted'])

res.head(100)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

fig, ax = plt.subplots(figsize=(8,6))

sns.heatmap(conf\_matrix,annot=True,cbar=True)

plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.title('Confusion Matrix')

print('\nConfusion Matrix: \n',conf\_matrix)

plt.show()

