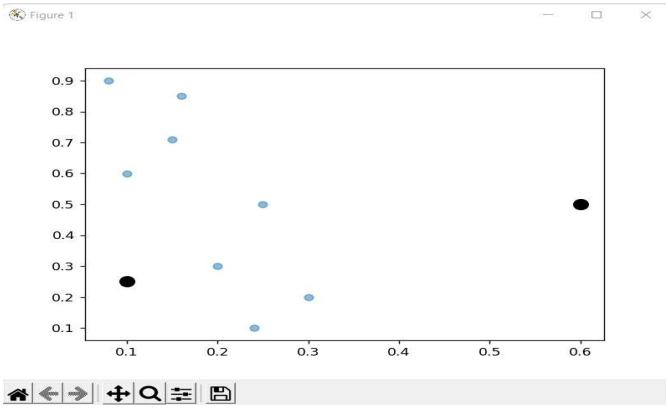
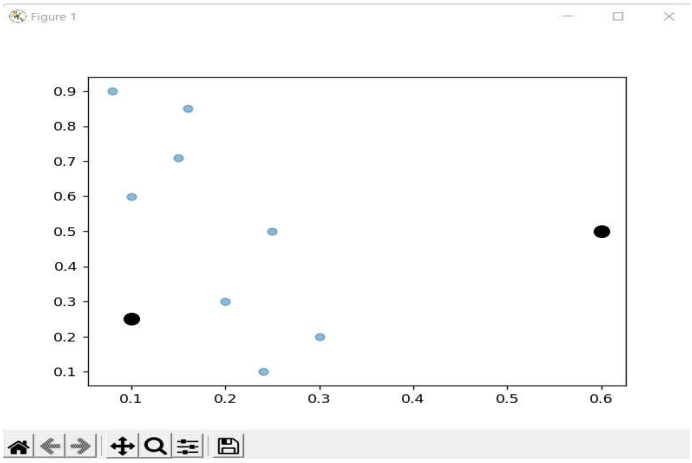
Assignment 1

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()
```

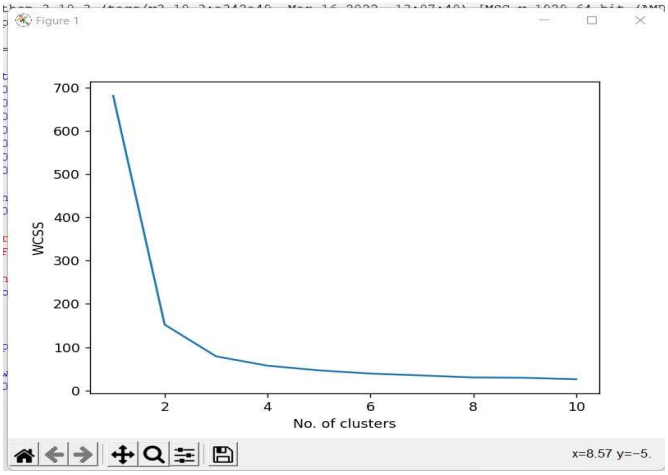



```
kmeans=KMeans(n_clusters=2,init=centroids)
kmeans.fit(X)
print("Labels after trainig:",kmeans.labels_)

#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_)
```

```
Python 3.10.3 (tags/v3.10.3:a342a49, Mar 16 2022, 13:07:40) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/Yash/BE Lab Assignments/ML/Assign 1/Kmeans.py ==========

Dataset:- [[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 ]]

Centeroids:- [[0.1  0.6 ]
 [0.25 0.5 ]]

Warning (from warnings module):
  File "C:\Python310\lib\site-packages\sklearn\cluster\_kmeans.py", line 1146
    self._check_params(X)
RuntimeWarning: Explicit initial center position passed: performing only one init in KMeans instead of n_init=10.
Labels after trainig: [0 0 0 0 1 1 1 1]

P6 belongs to  1 cluster

Population around centroid 2(P6) is  4

New values of centroids are: [[0.1225 0.765 ]
 [0.2475 0.275 ]]
>>>
```

#IRIS Dataset

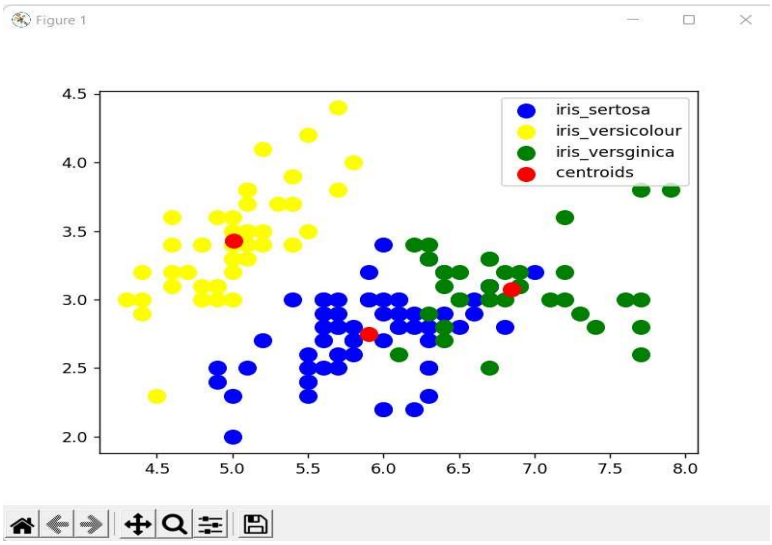
```
from sklearn import datasets
iris=datasets.load_iris()
iris_x=iris.data


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
```

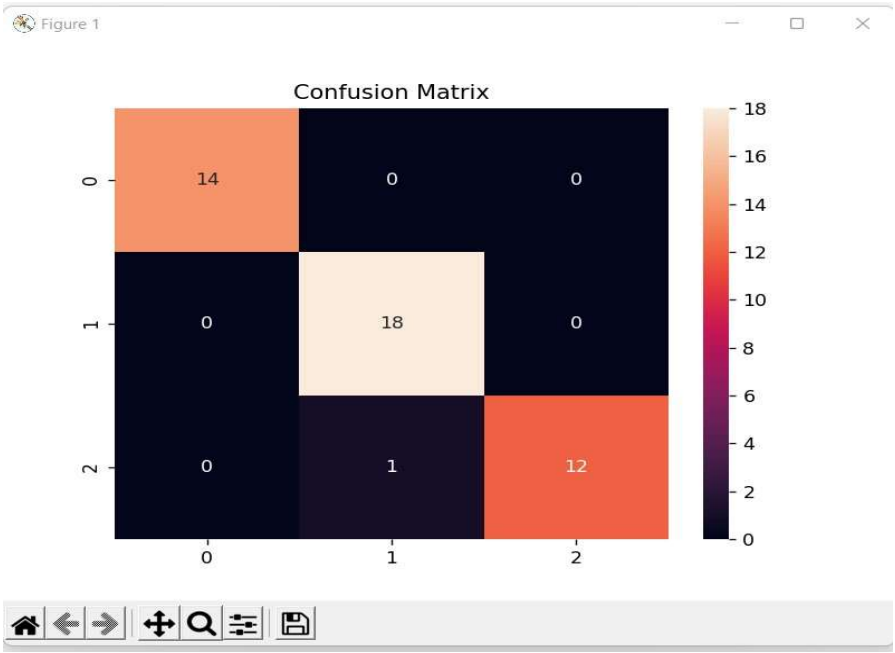
```
classifier = KNeighborsClassifier(n_neighbors=3, weights='distance')
classifier.fit(X,y)

# 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, ')')
```

```
============ RESTART: D:/Yash/BE Lab Assignments/ML/Assign 2/KNN.py ============
X:  [[2 4]
 [4 6]
 [4 4]
 [4 2]
 [6 4]
 [6 2]]
Y:  ['negative' 'negative' 'positive' 'negative' 'negative' 'positive']

General KNN :  negative ( orange )

Distance Weighted KNN :  negative ( orange )
```

```
# 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)
```

```
Iris Dataset:-
     sepal length (cm)  sepal width (cm)  ...  petal width (cm)    Species
0                  5.1               3.5  ...               0.2     setosa
1                  4.9               3.0  ...               0.2     setosa
2                  4.7               3.2  ...               0.2     setosa
3                  4.6               3.1  ...               0.2     setosa
4                  5.0               3.6  ...               0.2     setosa
..                 ...               ...  ...               ...        ...
145                6.7               3.0  ...               2.3  virginica
146                6.3               2.5  ...               1.9  virginica
147                6.5               3.0  ...               2.0  virginica
148                6.2               3.4  ...               2.3  virginica
149                5.9               3.0  ...               1.8  virginica

[150 rows x 5 columns]
Accuracy:-  0.9777777777777777

Confusion Matrix:-
 [[14  0  0]
 [ 0 18  0]
 [ 0  1 12]]
```

```
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])
```

```
Python 3.10.3 (tags/v3.10.3:a342a49, Mar 16 2022, 13:07:40) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======= RESTART: D:\Yash\BE Lab Assignments\ML\Assign 4\Decsion Tree.py =======
Dataset:
       Age  Income  Gender MaritialStatus Buys
0     <21    High    Male         Single   No
1     <21    High    Male        Married   No
2   21-35    High    Male         Single  Yes
3     >35  Medium    Male         Single  Yes
4     >35     Low  Female         Single  Yes
5     >35     Low  Female        Married   No
6   21-35     Low  Female        Married  Yes
7     <21  Medium    Male         Single   No
8     <21     Low  Female        Married  Yes
9     >35  Medium  Female         Single  Yes
10    <21  Medium  Female        Married  Yes
11  21-35  Medium    Male        Married  Yes
12  21-35    High  Female         Single  Yes
Age: [('<21', 1), ('<21', 1), ('21-35', 0), ('>35', 2), ('>35', 2), ('>35', 2), ('21-35', 0), ('<21', 1), ('<21', 1), ('>35', 2), ('<21', 1), ('21-35', 0), ('21-35', 0)]

Income: [('High', 0), ('High', 0), ('High', 0), ('Medium', 2), ('Low', 1), ('Low', 1), ('Low', 1), ('Medium', 2), ('Low', 1), ('Medium', 2), ('Medium', 2), ('Medium', 2), ('High', 0)]

Gender: [('Male', 1), ('Male', 1), ('Male', 1), ('Male', 1), ('Female', 0), ('Female', 0), ('Female', 0), ('Male', 1), ('Female', 0), ('Female', 0), ('Female', 0), ('Male', 1), ('Femal
e', 0)]

maritialStatus: [('Single', 1), ('Married', 0), ('Single', 1), ('Single', 1), ('Single', 1), ('Married', 0), ('Married', 0), ('Single', 1), ('Married', 0), ('Single', 1), ('Married', 0
), ('Married', 0), ('Single', 1)]

X:
    Age  Income  Gender  MaritialStatus
0     1       0       1               1
1     1       0       1               0
2     0       0       1               1
3     2       2       1               1
4     2       1       0               1
5     2       1       0               0
6     0       1       0               0
7     1       2       1               1
8     1       1       0               0
9     2       2       0               1
10    1       2       0               0
11    0       2       1               0
12    0       0       0               1
```

```
Y:
 0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
Name: Buys, dtype: object

Warning (from warnings module):
  File "C:\Python310\lib\site-packages\sklearn\base.py", line 450
    warnings.warn(
UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names

Predicted result for given conditions:-  Yes
```

#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()
```

```
Accuracy: 0.7661016949152543
```

```
array([[154,  20],
       [ 49,  72]], dtype=int64)
```