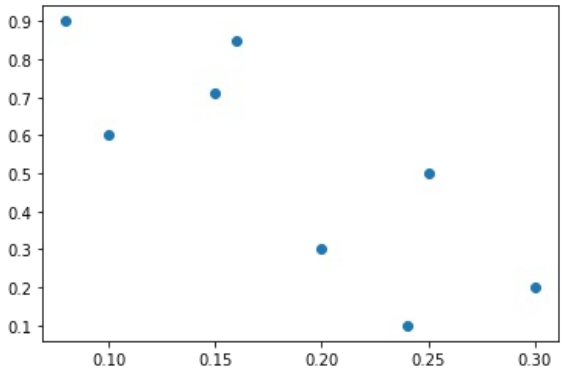
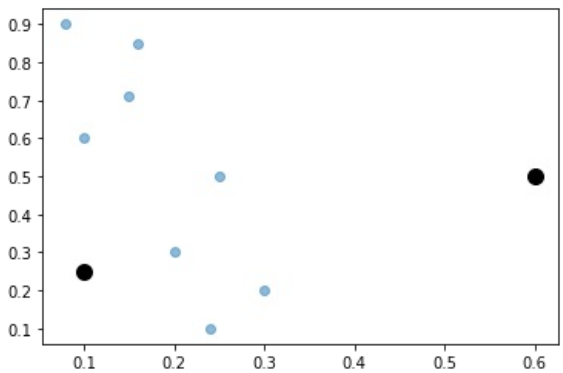
Question 1

Apply K-Means Clustering technique of machine learning to solve the given problem. We have given a collection of 8 points. P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16, 0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2]. Perform the kmean clustering with initial centroids as m1=P1 =Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following 1] Which cluster does P6 belongs to? 2] What is the population of cluster around m2? 3] What is updated value of m1 and m2? 4] What is the best value of K for the given problem

In [2]:

```
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)
df

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


model = KMeans(n_clusters=2, init=centroids, n_init=1, random_state=0)
model.fit(df)


model.labels_


print("Point P6 belongs to cluster", model.labels_[5])
print("Population of cluster 2:", sum(model.labels_==1))


print("Initial value of cluster centroids m1 & m2:")
print("m1 = ", centroids[0])
print("m2 = ", centroids[1])
print("Updated value of cluster centroids m1 & m2:")
print("m1 = ", model.cluster_centers_[0])
print("m2 = ", model.cluster_centers_[1])
```





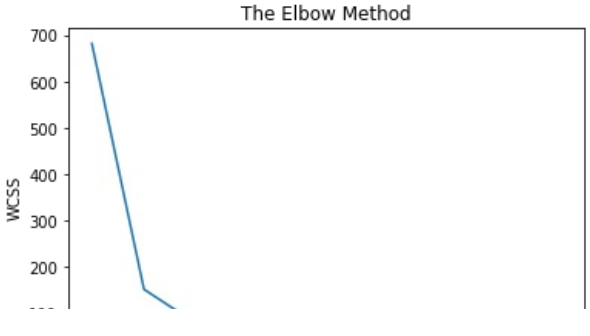
```
Point P6 belongs to cluster 1
Population of cluster 2: 4
Initial value of cluster centroids m1 & m2:
m1 =  [0.1 0.6]
m2 =  [0.25 0.5 ]
Updated value of cluster centroids m1 & m2:
m1 =  [0.1225 0.765 ]
m2 =  [0.2475 0.275 ]
```

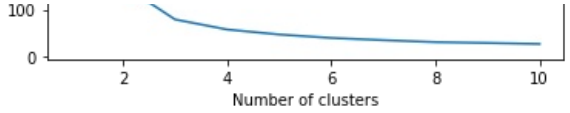
Question 2

Apply K-Means Clustering technique of machine learning to analyze the Iris dataset. Use Elbow method to find best value of K.

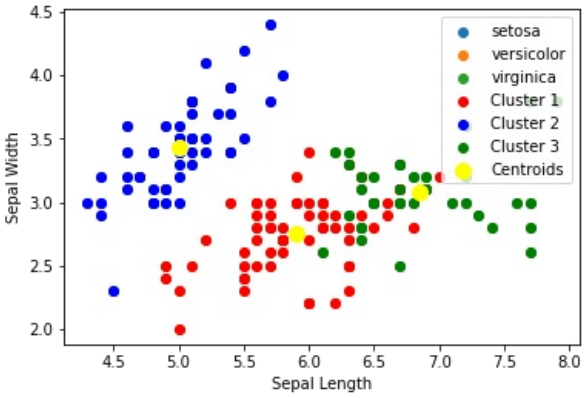
In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris


dataset = load_iris()
X = dataset.data
y = dataset.target

"""WCSS is the sum of squared distance between each point and the centroid in a cluster.
When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases,
the WCSS value will start to decrease. WCSS value is largest when K = 1."""

# Finding the optimum number of clusters
wcss = []
for i in range(1,11):
 kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
 kmeans.fit(X)
 wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

kemans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_kmenas = kemans.fit_predict(X)

dataset.feature_names

# Before Prediction

n_classes=len(dataset.target_names)
for i in range(n_classes):
    index = np.where(y == i)
    plt.scatter(X[index, 0], X[index, 1],
    label=dataset.target_names[i])
plt.legend()
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")


# After Prediction

plt.scatter(X[y_kmenas == 0, 0], X[y_kmenas == 0, 1], c='red', label='Cluster 1')
plt.scatter(X[y_kmenas == 1, 0], X[y_kmenas == 1, 1], c='blue', label='Cluster 2')
plt.scatter(X[y_kmenas == 2, 0], X[y_kmenas == 2, 1], c='green', label='Cluster 3')

# Plotting the centroids of the clusters
plt.scatter(kemans.cluster_centers_[:, 0], kemans.cluster_centers_[:, 1], s=100, c='yellow', label='Centroids')
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```





Out[ ]: <matplotlib.legend.Legend at 0x7f4f12284810>



Question 3

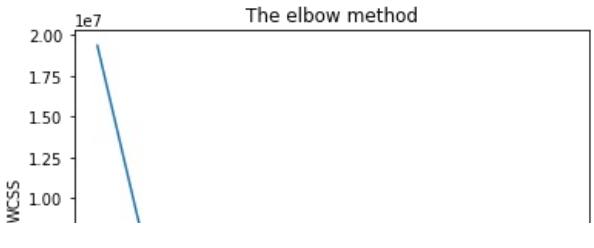
- Apply K-Means Clustering technique of machine learning to analyze the Bostan dataset. Use Elbow method to find best value of K.

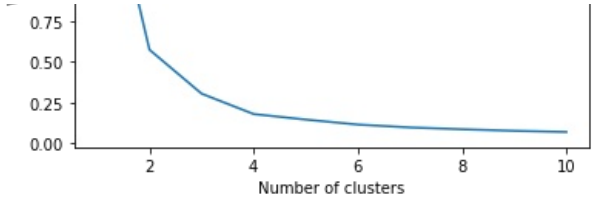
In [1]:

```
from sklearn.cluster import KMeans
from sklearn.datasets import load_boston
import pandas as pd
import matplotlib.pyplot as plt

boston = load_boston()

df = pd.DataFrame(boston.data,columns=boston.feature_names)
df.head()

print(df.shape)

wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(boston.data)
    wcss.append(kmeans.inertia_)
    kmeans.labels_
    print(len(kmeans.labels_))

plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
(506, 13)
C:\Users\user\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

```
506
506
506
506
506
506
506
506
506
506
```





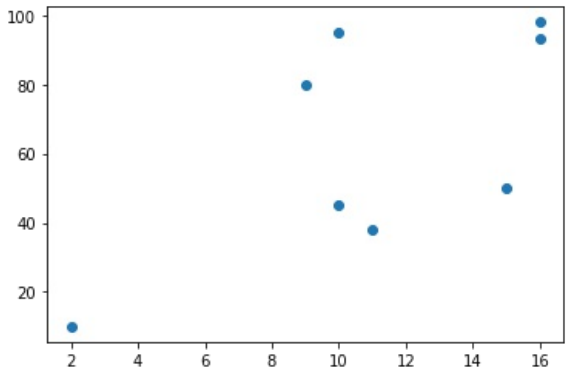
Question 4

Apply Linear Regression technique to solve the given problem. The following table shows the results of a recently conducted study on the correlation of the number of hours spent driving with the risk of developing acute backache. Find the equation of the best fit line for this data.

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = [[10,95],[9,80],[2,10],[15,50],[10,45],[16,98],[11,38],[16,93]]
data=pd.DataFrame(data)

X=data.iloc[:,:-1]
Y=data.iloc[:,-1]

plt.scatter(X,Y)

from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(X,Y)

print(lr.score(X,Y))

print(lr.coef_)

print(lr.intercept_)
```

```
0.43709481451010035
[4.58789861]
12.584627964022907
```



Question 5

Apply Linear Regression technique of machine learning to analyze the Diabetes dataset. Display accuracy of the model. Find the equation of the best fit line for this data.

In [ ]:

```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
X=diabetes.data
Y=diabetes.target

print(X.shape)
print(Y.shape)

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lr=LinearRegression()
lr.fit(X,Y)

lr_scores = cross_val_score(lr,X,y,cv=5)
```

```
print(lr_scores.mean())
```

```
(442, 10)
(442,)
0.48231812211149394
```

Question 6

Apply Linear, Ridge, Lasso Regression technique of machine learning to analyze and build the model of the Diabetes dataset. Display and compare the accuracy (Cross-Validation, R2 Score) of all the models.

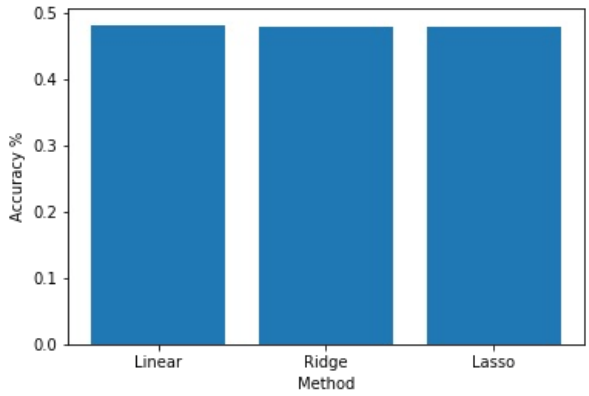
In [ ]:

```
# Linear Regression on Diabetes Dataset

from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
X=diabetes.data
Y=diabetes.target

print(X.shape)
print(Y.shape)

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lr=LinearRegression()
lr.fit(X,Y)

lr_scores = cross_val_score(lr,X,Y,cv=5)

lr_scores.mean()

#Ridge Regression on Diabetes Dataset

from sklearn.linear_model import Ridge
rg = Ridge(alpha=0.05)
rg.fit(X,Y)

ridge_scores = cross_val_score(rg,X,Y,cv=5)
ridge_scores.mean()

from sklearn.linear_model import RidgeCV
rg2 = RidgeCV(alphas=[0.01,0.02,0.04,0.05,0.1,0.2,0.5])
rg2.fit(X,Y)

rg2.alpha_

ridge_scores = cross_val_score(rg2,X,Y,cv=5)
ridge_scores.mean()

# Lasso Regression on Diabetes Dataset

from sklearn.linear_model import Lasso
lg = Lasso(alpha=0.05)
lg.fit(X,Y)


from sklearn.linear_model import LassoCV
lg2 = LassoCV(alphas=[0.01,0.02,0.04,0.05,0.1,0.2,0.5])
lg2.fit(X,Y)

lg2.alpha_

lasso_scores = cross_val_score(lg2,X,Y,cv=5)
lasso_scores.mean()

import matplotlib.pyplot as plt
x = ['Linear', 'Ridge','Lasso']
y = [lr_scores.mean(),ridge_scores.mean(),lasso_scores.mean()]
plt.xlabel("Method")
plt.ylabel("Accuracy %")
plt.bar(x,y)


"""
from sklearn.metrics import r2_score

### Assume y is the actual value and f is the predicted values
y =[10, 20, 30]
f =[20, 20, 20]
r2 = r2_score(y, f)
print('r2 score for a model which predicts mean value always is', r2)

"""
```

```
(442, 10)
```

```
(442,)
```

Out[ ]:
```
"\nfrom sklearn.metrics import r2_score\n\n### Assume y is the actual value and f is the predicted values\ny =[10
, 20, 30]\nf =[20, 20, 20]\nr2 = r2_score(y, f)\nprint('r2 score for a model which predicts mean value always is'
, r2)\n\n"
```



Question 7

Apply Decision Tree Classification technique to solve given problem: A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in table below. Use this dataset to build a decision tree, with Buys as the target variable, to help in buying lip-sticks in the future. Find the root node of decision tree. According to the decision tree you have made from previous training data set, what is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

In [ ]:
```
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")
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(x)

y=data.iloc[:,-1]
print(y)

dt=DecisionTreeClassifier()
dt.fit(x,y)

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

export_graphviz(dt,out_file="data1.dot",feature_names=x.columns,class_names=["No","Yes"])
!dot -Tpng data1.dot -o tree1.png
Image("tree1.png")
```

Question 8

Apply k-NN Classification technique to solve given problem: In the following diagram let blue circles indicate positive examples and orange squares indicate negative examples. We want to use k-NN algorithm for classifying the points. If k=3, find the class of the point (6,6).

In [ ]:
```
dataset = [[2,4,'negative'],[4,6,'negative'],[4,4,'positive'],[4,2,'negative'],[6,4,'negative'],[6,2,'positive']]
dataset = pd.DataFrame(dataset)
X = dataset.iloc[:,:-1]
Y = dataset.iloc[:,-1]

print(X)
print(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('General 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('General KNN : ', y_pred[0],'(', ans, ')')
```

```
   0  1
0  2  4
1  4  6
2  4  4
3  4  2
4  6  4
5  6  2
0    negative
1    negative
2    positive
3    negative
4    negative
5    positive
Name: 2, dtype: object
General KNN :  negative ( orange )
General KNN :  negative ( orange )
```

Question 9 :

Consider the following training data set. Write a program to construct a decision tree using ID3 algorithm. Display Accuracy measures for the same and predict a class of suitable query.

https://www.youtube.com/watch?v=2YQHPfwVuF8

- https://medium.datadriveninvestor.com/decision-tree-algorithm-with-hands-on-example-e6c2afb40d38

In [ ]:
```
# weather dataset
```

Question 10 :

Consider tissue paper factory application. Apply KNN algorithm to find class of new tissue paper (X1= 3, X2=7). Assume K=3 X1 = Acid Durability (seconds) X2 = Strength (kg/square meter) Y = Classification 7 7 Bad 7 4 Bad 3 4 Good 1 4 Good

In [ ]:
```
import pandas as pd
import numpy as np

data = [[7,7,'B'],[7,4,'B'],[3,4,'G'],[1,4,'G']]

data = pd.DataFrame(data)

test = [[3,7]]

x = data.iloc[:,:-1].values
y = data.iloc[:, -1].values
```

```
print(x)

print(y)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

print(y)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(x,y)

print(classifier.predict(test))
```

```
[[7 7]
 [7 4]
 [3 4]
 [1 4]]
['B' 'B' 'G' 'G']
[0 0 1 1]
[1]
```

ICS

Implementation of Playfair Cipher Matrix
Implementation of Hill Cipher.
Implementation of columnar Transposition cipher.
Implementation of Diffie-Hellman key exchange Implementation of RSA algorithm.

In [ ]: