K-Means Clustering Implementation

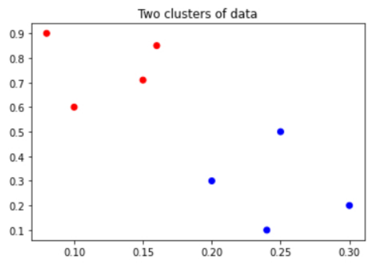
We have given a collection of 8 points. P1=[0.1,0.6] ,P2=[0.15,0.71], P3=[0.08,0.9] P4=[0.16, 0.85], P5=[0.2,0.3], P6=[0.25,0.5], P7=[0.24,0.1], P8=[0.3,0.2]. Perform the k-mean clustering with initial centroids as m1=P1 = Cluster#1=C1 and m2=P8=cluster#2=C2. Answer the following:
1] Which cluster does P6 belong to?
2] What is the population of cluster around m2?
3] What is updated value of m1 and m2?

In [4]:
```python
import matplotlib.pyplot as plt
import numpy as np
```

In [5]:
```python
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]
K=2

points=[P1,P2,P3,P4,P5,P6,P7,P8]
```

KMeans Clustering using sklearn

In [6]:
```python
from sklearn.cluster import KMeans

# Configuration options
num_samples_total = 8
cluster_centers = [(0.1,0.6), (0.3, 0.2)]
num_classes = len(cluster_centers)

X = np.array(points)

# Fit K-means with Scikit
kmeans = KMeans(init='k-means++', n_clusters=num_classes)
kmeans.fit(X)

# Predict the cluster for all the samples
P = kmeans.predict(X)

# Generate scatter plot for training data
colors = list(map(lambda x: 'blue' if x == 1 else 'red', P))
plt.scatter(X[:,0], X[:,1], c=colors, marker="o", picker=True)
plt.title('Two clusters of data')
plt.show()
```

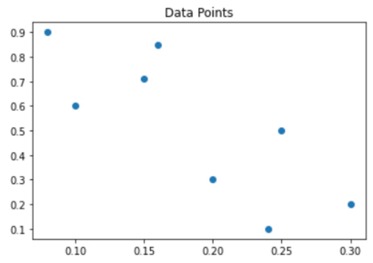


Kmeans clustering implementation

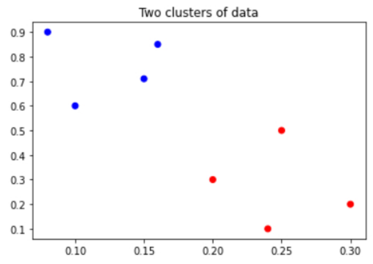
In [7]:
```python
import math
def dist(A,B):
    xd=(A[0]-B[0])**2
    yd=(A[1]-B[1])**2
    d=math.sqrt(xd+yd)
    return d
```

In [8]:
```python
def cluster(C1,C2):
    cluster1=list()
    cluster2=list()
    c1=C1
    c2=C2
    for p in points:
        d1=dist(p,C1)
        d2=dist(p,C2)
        if d1<d2:
            cluster1.append(p)
        else:
            cluster2.append(p)
    x1=0
    y1=0

    for i in cluster1:
        x1=x1+i[0]
        y1=y1+i[1]

    x1=x1/len(cluster1)
    y1=y1/len(cluster1)

    centroid1=[x1,y1]

    x2=0
    y2=0

    for i in cluster2:
        x2=x2+i[0]
        y2=y2+i[1]

    x2=x2/len(cluster2)
    y2=y2/len(cluster2)

    centroid2=[x2,y2]

    C1=centroid1
    C2=centroid2

    if centroid1[0]==c1[0] and centroid2[0]==c2[0] and centroid1[1]==c1[1] and centroid2[1]==c2[1]:
        print("Clusters are:")
        print(cluster1)
        print(cluster2)

        if [0.25,0.5] in cluster1:
            print("P6 belongs to cluster 1")
        elif [0.25,0.5] in cluster2:
            print("P6 belongs to cluster 2")

        print("Population of cluster around m2:",len(cluster2))
        print("Updated value of centroids:")
        print("C1:",C1)
        print("C2:",C2)
        P=list()

        for i in points:
            if i in cluster1:
                P.append(0)
            else:
                P.append(1)

        # Generate scatter plot for training data
        X = np.array(points)
        colors = list(map(lambda x: 'blue' if x == 1 else 'red', P))
        plt.scatter(X[:,0], X[:,1], c=colors, marker="o", picker=True)
        plt.title('Two clusters of data')
        plt.show()
    else:
        cluster(C1,C2)
```

In [9]:
```
plt.scatter(X[:,0], X[:,1], marker="o", picker=True)
plt.title('Data Points')
plt.show()

C1=P1
C2=P2

print("Points are:\n",points)
cluster(C1,C2)
```



```
Points are:
 [[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]]
Clusters are:
[[0.2, 0.3], [0.25, 0.5], [0.24, 0.1], [0.3, 0.2]]
[[0.1, 0.6], [0.15, 0.71], [0.08, 0.9], [0.16, 0.85]]
P6 belongs to cluster 1
Population of cluster around m2: 4
Updated value of centroids:
C1: [0.2475, 0.275]
C2: [0.1225, 0.765]
```



In [ ]: