University of Haripur.

Assignment #.

Machine Learning

Title: KMean Clustering.

To: Ms. Nadia

From: Hamza Sadaat.

Depart: Information Technology.

Program: Computer Science.

Roll No: F18-0501.

Semester: 7th 'B'.

Dated: 31/01/2022.

Building up KMean Clustering algorithm from scratch:

The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. There are many different types of clustering methods, but k-means is one of the oldest and most approachable. These traits make implementing k-means clustering in Python reasonably straightforward, even for novice programmers and data scientists.

Methodology:

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

Here, we are going to implement KMean Clustering from scratch in python with a given dataset.

Initializing Libraries and Creating a data frame and initializing random centroids

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 21 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [48]:
```
import pandas as pd
import numpy as np
import random
import math
import matplotlib.pyplot as plt
```

In [49]:
```
d = {'X1':[2,3,4,10,11,12,20,25,30]}
df = pd.DataFrame(data=d)
```

In [50]:
```
def calRandomCentroids(k, df):
    centroids = []
    for i in range(k):
        rand = random.randint(0,len(df)-1)
        randVal = tuple(df.loc[rand].values)
        while randVal in centroids:
            rand = random.randint(0,len(df)-1)
            randVal = tuple(df.loc[rand].values)
        else:
            centroids.append(randVal)
    return centroids
```

Calculating distance via Euclidian Distance Formula and making clusters after calculating clusters initializing new centroids

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 25 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [51]:
```
def calDist(a,b):
    return math.sqrt(sum((np.array(a)-np.array(b))**2))

def makeClusters(k, df, centroids):
    clusters = {}
    for tup in centroids:
        clusters[tup] = []
    for i in range(len(df)):
        pointDists = {}
        for tup in centroids:
            dist = calDist(tuple(df.loc[i].values),tup)
            pointDists[dist] = tup
        ncp = pointDists.get(min(pointDists))
        clusters[ncp].append(i) #or i
    return clusters
```

In [52]:
```
def calNewCentroids(clusters):
    newcentroids = []
    for k in clusters:
        sumc = 0
        for l in range(len(clusters[k])):
            sumc += df.loc[clusters[k][l]]
        cent = sumc/len(clusters[k])
        newcentroids.append(tuple(cent))
    return newcentroids
```

In [53]:
```
def checkConvergence(k,oldcentroids,newcentroids):
    result = []
    for i in range(k):
        rs = calDist(oldcentroids[i],newcentroids[i])
        result.append(rs)
    print("convergence result is {}".format(result))
    count = 0
    for i in range(len(result)):
        if result[i] <= 0.5:
            count = count+1
    return True if count == len(result) else False
```

Showing Output

jupyter Hamza Sadaat F18-0501 Last Checkpoint: 25 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [54]:
```
def kMeans(k, df):

    centroids = calRandomCentroids(k, df)
    print("random centroids are {}".format(centroids))
    oldcentroids = centroids

    clusters = makeClusters(k, df, oldcentroids)
    print("first iter clusters are {}".format(clusters))

    newcentroids = calNewCentroids(clusters)
    print("new centroids are {}".format(newcentroids))

    res = checkConvergence(k,oldcentroids,newcentroids)
    print(res)

    while res == False:
        oldcentroids = newcentroids
        clusters = makeClusters(k, df, oldcentroids)
        print("further iter clusters are {}".format(clusters))
        newcentroids = calNewCentroids(clusters)
        res = checkConvergence(k,oldcentroids,newcentroids)
        print(res)
    else:
        print("Final clusterings are {}".format(clusters))


kMeans(2, df)
```

```
random centroids are [(20,), (4,)]
first iter clusters are {(20,): [6, 7, 8], (4,): [0, 1, 2, 3, 4, 5]}
new centroids are [(25.0,), (7.0,)]
convergence result is [5.0, 3.0]
False
further iter clusters are {(25.0,): [6, 7, 8], (7.0,): [0, 1, 2, 3, 4, 5]}
convergence result is [0.0, 0.0]
True
Final clusterings are {(25.0,): [6, 7, 8], (7.0,): [0, 1, 2, 3, 4, 5]}
```