**Experiment 7**

**Aim: Implementation of clustering algorithms (K-means)**

**Theory:**

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

Determines the best value for K center points or centroids by an iterative process.

Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

Algorithm:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

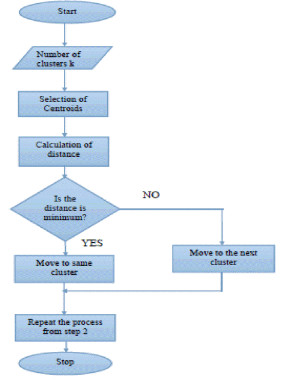
Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Flowchart:



**Fig 7.1 Flowchart of K-Means**

Example:

Cluster the following eight points (with (x, y) representing locations) into three clusters:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).

The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

Ρ(a, b) = |x2 – x1| + |y2 – y1|

Use K-Means Algorithm to find the three cluster centers after the second iteration.

Iteration 1:

Ρ(A1, C1) = |x2 – x1| + |y2 – y1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Given Points** | **Distance from center (2, 10) of Cluster-01** | **Distance from center (5, 8) of Cluster-02** | **Distance from center (1, 2) of Cluster-03** | **Point belongs to Cluster** |
| A1(2, 10) | 0 | 5 | 9 | C1 |
| A2(2, 5) | 5 | 6 | 4 | C3 |
| A3(8, 4) | 12 | 7 | 9 | C2 |
| A4(5, 8) | 5 | 0 | 10 | C2 |
| A5(7, 5) | 10 | 5 | 9 | C2 |
| A6(6, 4) | 10 | 5 | 7 | C2 |
| A7(1, 2) | 9 | 10 | 0 | C3 |
| A8(4, 9) | 3 | 2 | 10 | C2 |

Iteration 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Given Points** | **Distance from center (2, 10) of Cluster-01** | **Distance from center (6, 6) of Cluster-02** | **Distance from center (1.5, 3.5) of Cluster-03** | **Point belongs to Cluster** |
| A1(2, 10) | 0 | 8 | 7 | C1 |
| A2(2, 5) | 5 | 5 | 2 | C3 |
| A3(8, 4) | 12 | 4 | 7 | C2 |
| A4(5, 8) | 5 | 3 | 8 | C2 |
| A5(7, 5) | 10 | 2 | 7 | C2 |
| A6(6, 4) | 10 | 2 | 5 | C2 |
| A7(1, 2) | 9 | 9 | 2 | C3 |
| A8(4, 9) | 3 | 5 | 8 | C1 |

Therefore, new clusters:

Cluster-01:

A1(2, 10)

A8(4, 9)

Cluster-02:

A3(8, 4)

A4(5, 8)

A5(7, 5)

A6(6, 4)

Cluster-03:

A2(2, 5)

A7(1, 2)

Center of Cluster-01

= ((2 + 4)/2, (10 + 9)/2) = (3, 9.5)

Center of Cluster-02

= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4) = (6.5, 5.25)

Center of Cluster-03

= ((2 + 1)/2, (5 + 2)/2) = (1.5, 3.5)

Ans:

* C1(3, 9.5)
* C2(6.5, 5.25)
* C3(1.5, 3.5)

**Code:**

import csv

import matplotlib.pyplot as plt

def initializeCentroids(k, data):

centroids = []

for i in range(k):

centroids.append(data[i])

return centroids

def plot(clusters):

colors = ['red', 'green', 'blue', 'yellow', 'black', 'purple', 'pink', 'cyan']

i = 0

for cluster in clusters:

x = [point[0] for point in cluster]

y = [point[1] for point in cluster]

plt.scatter(x, y, s=10, c=colors[i])

i += 1

def distance(xc, yc, px, py):

x = (px - xc) \*\* 2

y = (py - yc) \*\* 2

distance = (x + y) \*\* 0.5

return round(distance, 2)

def mean(cluster):

sumx = 0

sumy = 0

for coord in cluster:

sumx += coord[0]

sumy += coord[1]

return (round(sumx / len(cluster), 2), round(sumy / len(cluster), 2))

data = []

k = int(input('Enter the number of clusters: '))

with open('ufc\_master\_data.csv', mode='r') as csv\_file:

csv\_reader = csv.DictReader(csv\_file)

for row in csv\_reader:

data.append((float(row['ufc\_wins']), float(row['ufc\_loses'])))

centroids = initializeCentroids(k, data)

cluster\_coords = [[] for \_ in range(k)] # Initialize cluster\_coords with empty lists for each cluster

while True:

for coords in data:

point\_distance = [distance(centroid[0], centroid[1], coords[0], coords[1]) for centroid in centroids]

cluster\_coords[point\_distance.index(min(point\_distance))].append(coords)

prev\_cents = centroids

centroids = [mean(cluster) for cluster in cluster\_coords]

if prev\_cents == centroids:

break

i = 1

for centroid in centroids:

print('Centroid ', i, ': ', centroid)

i += 1

plot(cluster\_coords)

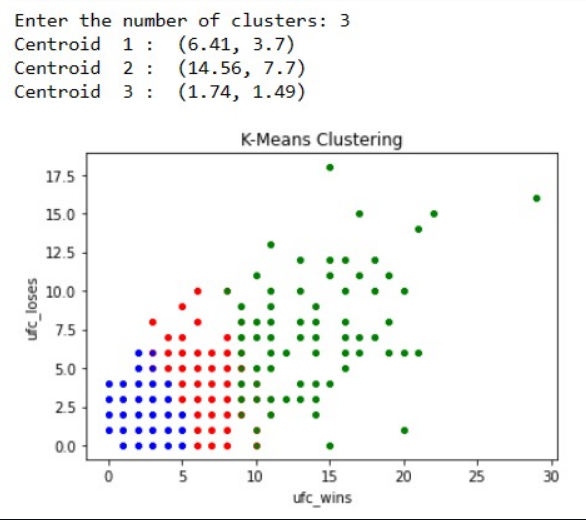
plt.title('K-Means Clustering')

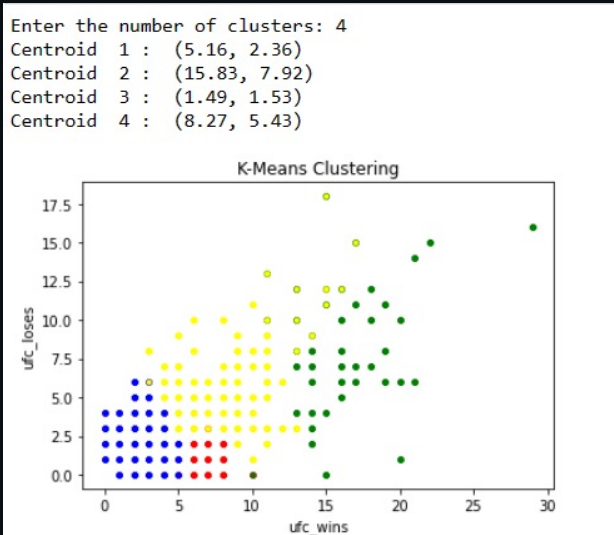
plt.xlabel('ufc\_wins')

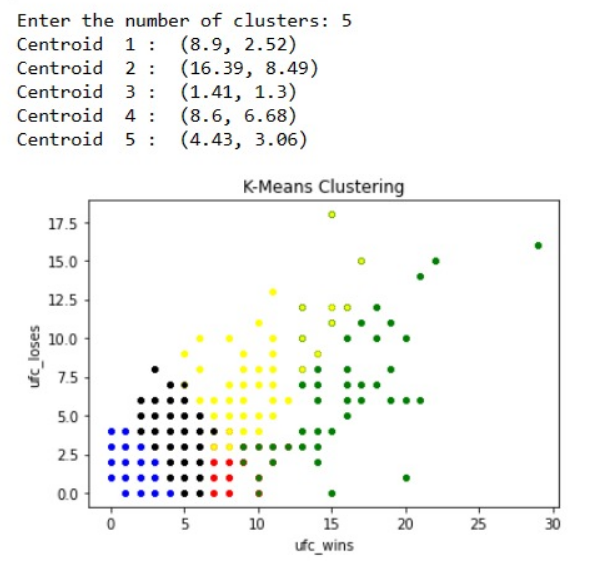
plt.ylabel('ufc\_loses')

plt.show()

**Output:**

****

****

****