Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No. : CSE 4108
Course Name : Artificial Intelligence Lab
Assignment No. : 02

Submitted To :

Md. Siam Ansary
Department of CSE, AUST

Tonmoy Hossain
Department of CSE, AUST

Submitted By:

Name : Sanjida Akter Ishita
ID No. : 170204089
Session : Fall 2020
Section : B
Lab group : B2

Date of Submission: July 24, 2021

QUESTION: Implement K-means Clustering and K-nearest Neighbor Classifier without using Scikit-learn.

ANSWER: Implementing K-means clustering using Scikit-learn is easy. But in this assignment we have to implement this without using any library function of Scikit-learn.

K-means Clustering

Code in python

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

data = pd.read_csv('worldcupplayers.csv')
data.head()

X = data[["age","caps"]]

plt.scatter(X["caps"],X["age"],c='black')
plt.xlabel('PalyerCap')
plt.ylabel('Player Age (In Years)')
#plt.show()

K=3

Centroids = (X.sample(n=K))
plt.scatter(X["caps"],X["age"],c='black')
plt.scatter(Centroids["caps"],Centroids["age"],c='red')
plt.xlabel('PalyerCap')
plt.ylabel('Player Age (In Years)')
#plt.show()

diff = 1
j=0

while(diff!=0):
    XD=X
    i=1
    for index1,row_c in Centroids.iterrows():
        ED=[]
        for index2,row_d in XD.iterrows():
            d1=(row_c["caps"]-row_d["caps"])**2
            d2=(row_c["age"]-row_d["age"])**2
            d=np.sqrt(d1+d2)
            ED.append(d)
        X[i]=ED
        i=i+1
    C=[]
```

```
    for index,row in X.iterrows():
        min_dist=row[1]
        pos=1
        for i in range(K):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                pos=i+1
        C.append(pos)
    X["Player"]=C
    Centroids_new = X.groupby(["Player"]).mean()[["age","caps"]]
    if j == 0:
        diff=1
        j=j+1
    else:
        diff = (Centroids_new['age'] - Centroids['age']).sum() + (Centroids_new['caps'] - Centroids['caps']).sum()
        print(diff.sum())
    Centroids = X.groupby(["Player"]).mean()[["age","caps"]]
color = ['blue', 'green', 'cyan']
for k in range(K):
    data = X[X["Player"] == k + 1]
    plt.scatter(data["caps"], data["age"],c=color[k])
plt.scatter(Centroids["caps"], Centroids["age"],c='red')
plt.xlabel('Cap of Player')
plt.ylabel('Player Age (In Years)')
plt.show()
```

Dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	team	player_number	position	birth_date	shirt_nam	club_name	height	weight	league	age	name	caps			
2	Argentina	1	GK	########	GUZMÃN	Tigres UANL	192	90	MEX	32.33973	Nahuel Gu	6			
3	Argentina	2	DF	########	MERCADO	Sevilla FC	181	81	ESP	31.2411	Gabriel Me	20			
4	Argentina	3	DF	########	TAGLIAFIC	AFC Ajax	169	65	NED	25.7863	NicolÃ¡s T	4			
5	Argentina	4	DF	########	ANSALDI	Torino FC	181	73	ITA	31.73151	Cristian An	5			
6	Argentina	5	MF	########	BIGLIA	AC Milan	175	73	ITA	32.36986	Lucas Bigli	57			
7	Argentina	6	DF	########	FAZIO	AS Roma	199	85	ITA	31.24384	Federico F	9			
8	Argentina	7	MF	########	BANEGA	Sevilla FC	175	73	ESP	29.9589	Ã‰ver Ba	62			
9	Argentina	8	DF	########	ACUÃ‘A	Sporting CP	172	77	POR	26.6274	Marcos Ac	10			
10	Argentina	9	FW	########	HIGUAÃN	Juventus FC	184	75	ITA	30.50959	Gonzalo H	71			
11	Argentina	10	FW	########	MESSI	FC Barcelona	170	72	ESP	30.9726	Lionel Mes	124			
12	Argentina	11	MF	########	DI MARÃA	Paris Saint-Germain F	178	75	FRA	30.32877	Ángel Di M	94			
13	Argentina	12	GK	########	ARMANI	CA River Plate	189	85	ARG	31.66027	Franco Arr	0			
14	Argentina	13	MF	########	MEZA	CA Independiente	180	76	ARG	25.49589	Maximiliar	2			
15	Argentina	14	DF	6/8/1984	MASCHER/	Hebei China Fortune F	174	73	CHN	34.01644	Javier Mas	143			
16	Argentina	15	MF	########	LANZINI	West Ham United FC	167	66	ENG	25.32603	Enzo PÃ©r	23			
17	Argentina	16	DF	########	ROJO	Manchester United F(	189	82	ENG	28.23562	Marcos Rc	56			
18	Argentina	17	DF	########	OTAMEND	Manchester City FC	181	81	ENG	30.33425	NicolÃ¡s O	54			
19	Argentina	18	DF	########	SALVIO	SL Benfica	167	69	POR	27.92055	Eduardo Sa	9			

worldcupplayers

Figure 1: A part of worldcupplayers.csv. From this dataset only two columns are taken, named age and caps.

Output:

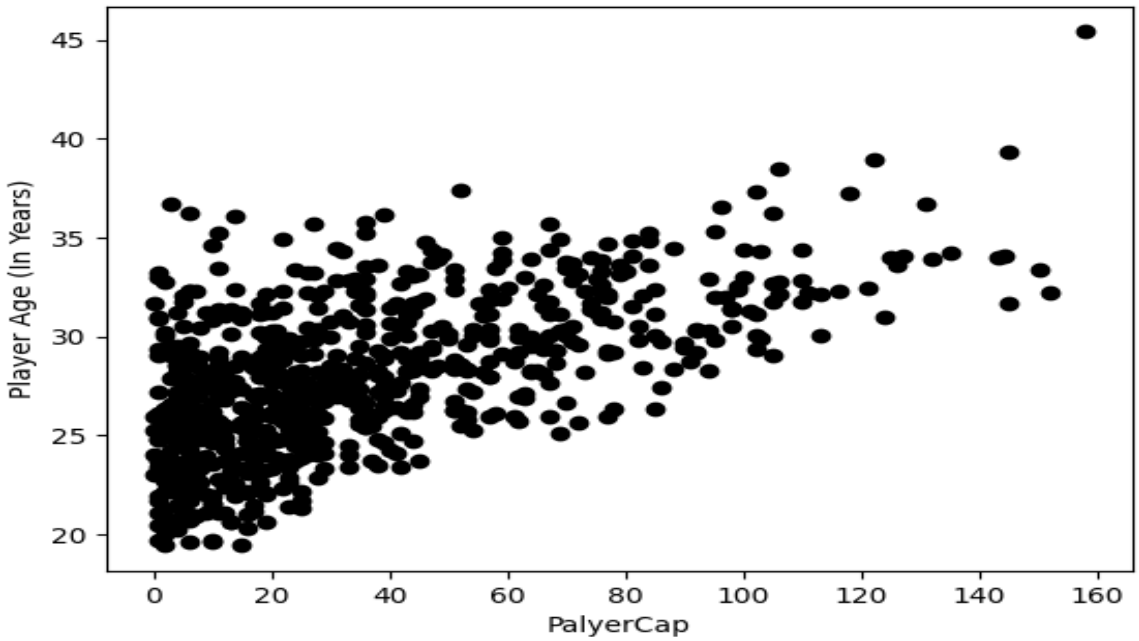

Figure 2 : Before Clustering

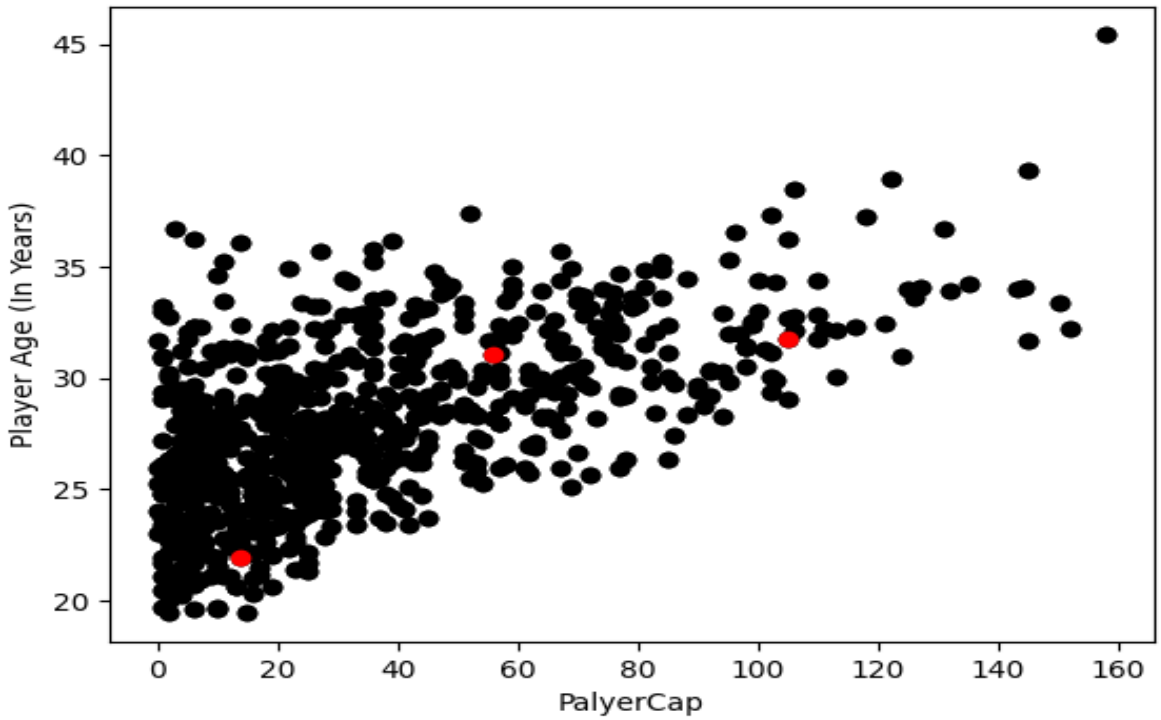

Figure 3 : Randomly finding 3 centroids

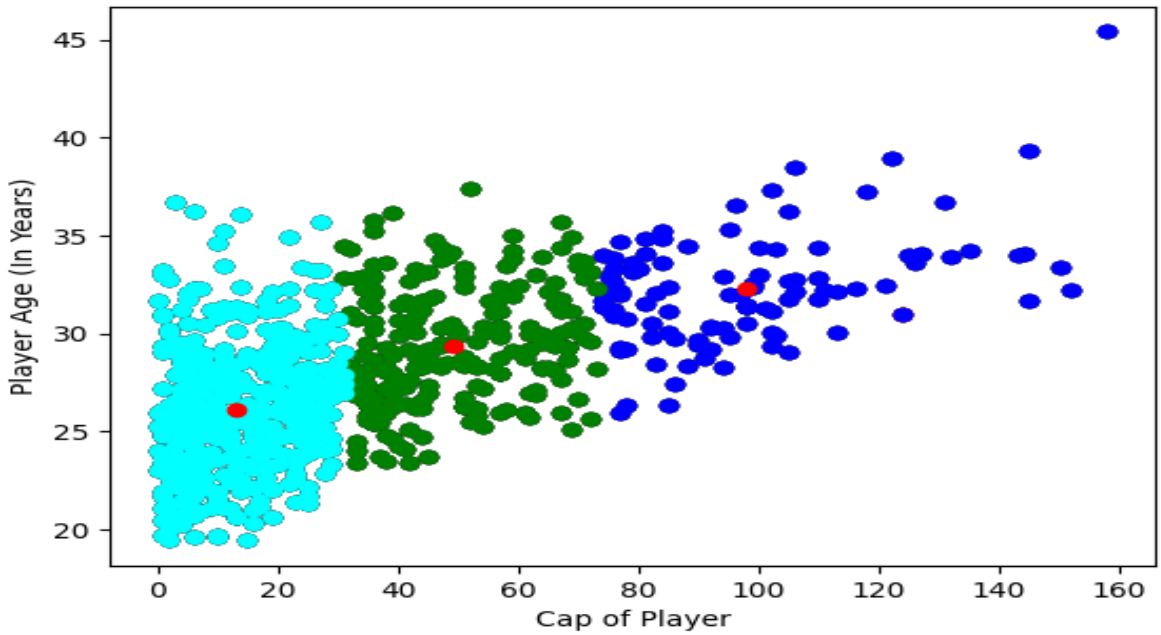

Figure 4 : After Clustering

K-nearest Neighbor

Here I have to implement K-nearest Neighbor without using library function of Scikit-learn. With library functions it would be easy. But I have to choose other way for implementing it. Here are the code and output

Python Code

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

# Use matplotlib in Jupyter Notebook Outputs
%matplotlib inline
```

```
# Input data - [Age, Caps]
X = [[32,6], [31,20], [25,4], [31,5], [32,57], [31,9], [29,62], [26,10], [30,71],
     [30,124], [30,94], [31,0], [25,2], [34,143], [25,23], [28,56], [30,54], [27,9],
     [30,85], [22,5], [24,12], [22,5], [36,3], [26,44], [24,18], [30,2], [38,106],
     [32,71], [28,4], [27,53], [25,36], [26,35], [29,64], [25,4], [36,6], [27,34],
     [24,6], [33,76], [27,23], [19,2], [33,1], [25,8], [26,35], [25,2], [25,19],
     [25,37], [26,58], [29,77], [32,66], [32,77], [31,102], [29,90], [26,62], [30,82]]

# Labels - Accepted or Rejected
Y = ['accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted',
     'accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted',
     'accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted','accepted',
     'rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected',
     'rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected',
     'rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected','rejected']
```

```
for i in range(len(X)):
    if Y[i] == 'accepted':
        plt.scatter(X[i][0], X[i][1], s=120, marker='P', linewidths=2, color='green')
    else:
        plt.scatter(X[i][0], X[i][1], s=120, marker='P', linewidths=2, color='red')

plt.plot()
```

```python
# Find which variable is the most in an array of variables
def most_found(array):
    list_of_words = []
    for i in range(len(array)):
        if array[i] not in list_of_words:
            list_of_words.append(array[i])

    most_counted = ''
    n_of_most_counted = None

    for i in range(len(list_of_words)):
        counted = array.count(list_of_words[i])
        if n_of_most_counted == None:
            most_counted = list_of_words[i]
            n_of_most_counted = counted
        elif n_of_most_counted < counted:
            most_counted = list_of_words[i]
            n_of_most_counted = counted
        elif n_of_most_counted == counted:
            most_counted = None

    return most_counted
```

```python
def find_neighbors(point, data, labels, k=3):
    # How many dimentions do the space have?
    n_of_dimensions = len(point)

    #find nearest neighbors
    neighbors = []
    neighbor_labels = []

    for i in range(0, k):
        # To find it in data later, I get its order
        nearest_neighbor_id = None
        smallest_distance = None

        for i in range(0, len(data)):
            eucledian_dist = 0
            for d in range(0, n_of_dimensions):
                dist = abs(point[d] - data[i][d])
                eucledian_dist += dist

            eucledian_dist = np.sqrt(eucledian_dist)

            if smallest_distance == None:
                smallest_distance = eucledian_dist
                nearest_neighbor_id = i
            elif smallest_distance > eucledian_dist:
                smallest_distance = eucledian_dist
                nearest_neighbor_id = i

        neighbors.append(data[nearest_neighbor_id])
        neighbor_labels.append(labels[nearest_neighbor_id])
```

```
            data.remove(data[nearest_neighbor_id])
            labels.remove(labels[nearest_neighbor_id])
        return neighbor_labels

    def k_nearest_neighbor(point, data, labels, k=3):

        # If two different labels are most found, continue to search for 1 more k
        while True:
            neighbor_labels = find_neighbors(point, data, labels, k=k)
            label = most_found(neighbor_labels)
            if label != None:
                break
            k += 1
            if k >= len(data):
                break

        return label
```

```
[ ]  point = [26,35]
     k_nearest_neighbor(point, X, Y, k=5)
```

Output

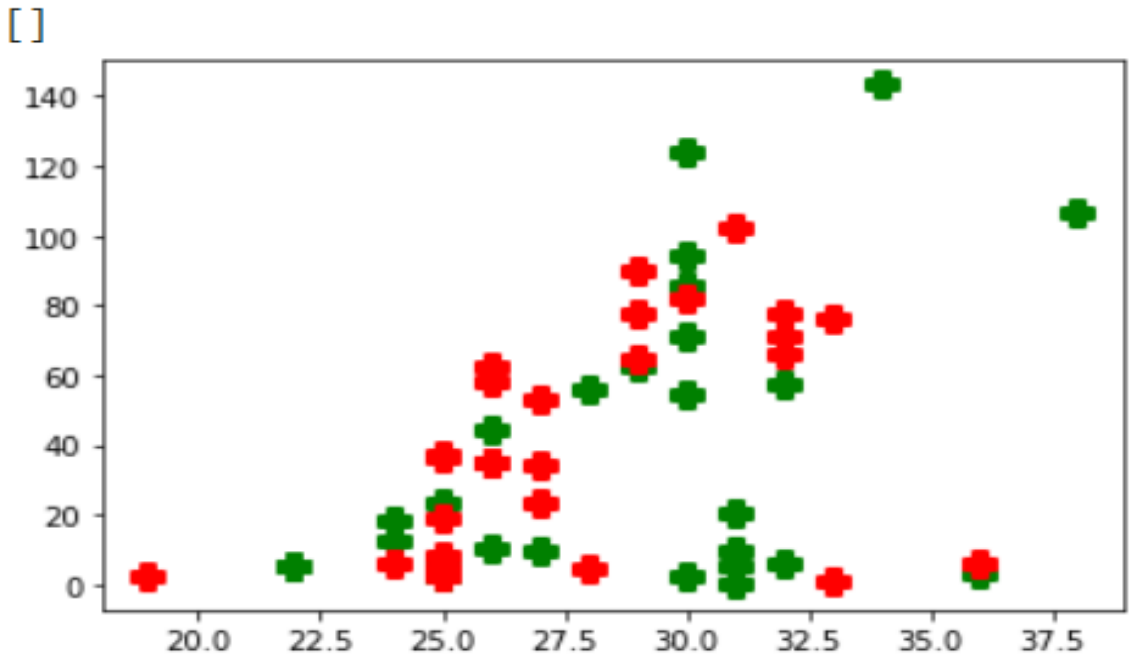

Figure 5: Plotting dataset in a graph after using matplotlib library

```
point = [26,35]
k_nearest_neighbor(point, X, Y, k=5)
```

```
'accepted'
```

Figure 6: Output `accepted' for input Age 26 and Caps 35