بنام خدا

جلسه نهم داده کاوی

یادآوری مجدد k-means با مثال عددی

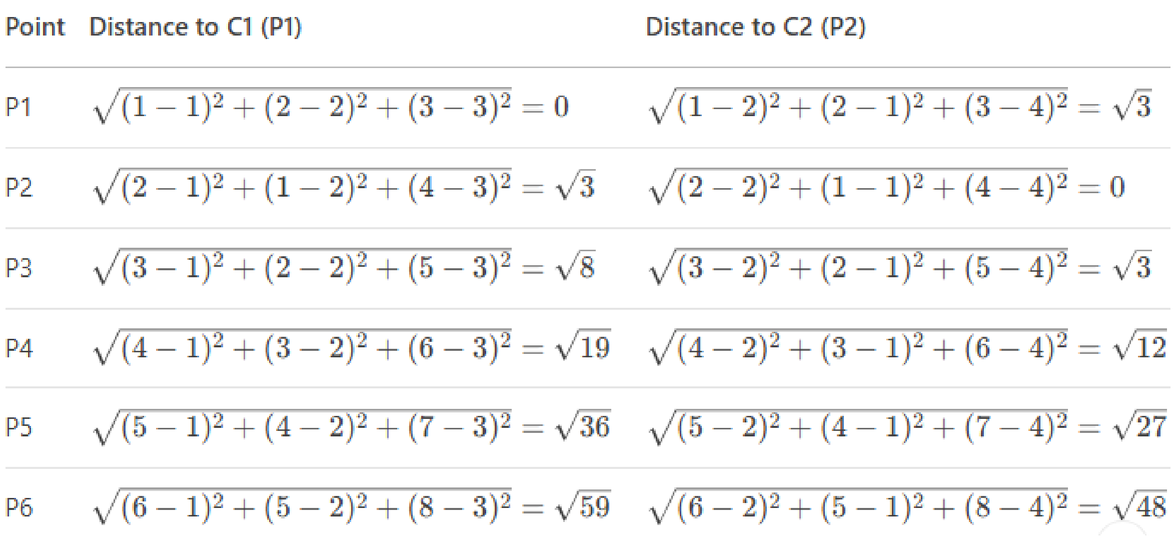
Point	X1	X2	X3	Y
P1	1	2	3	1.5
P2	2	1	4	2.5
P3	3	2	5	7.3
P4	4	3	6	4.2
P5	5	4	7	9.11
P6	6	5	8	1.2

انتخاب مراکز (در شروع رندم انتخاب می شود)

Centroid C1 = P1 = (1, 2, 3)

Centroid C2 = P2 = (2, 1, 4)

انتصاب:



Point	Distance to C1 (P1)	Distance to C2 (P2)
P1	$\sqrt{(1-1)^2+(2-2)^2+(3-3)^2}=0$	$\sqrt{(1-2)^2+(2-1)^2+(3-4)^2}=\sqrt{3}$
P2	$\sqrt{(2-1)^2+(1-2)^2+(4-3)^2}=\sqrt{3}$	$\sqrt{(2-2)^2+(1-1)^2+(4-4)^2}=0$
P3	$\sqrt{(3-1)^2+(2-2)^2+(5-3)^2}=\sqrt{8}$	$\sqrt{(3-2)^2+(2-1)^2+(5-4)^2}=\sqrt{3}$
P4	$\sqrt{(4-1)^2+(3-2)^2+(6-3)^2}=\sqrt{19}$	$\sqrt{(4-2)^2+(3-1)^2+(6-4)^2}=\sqrt{12}$
P5	$\sqrt{(5-1)^2+(4-2)^2+(7-3)^2}=\sqrt{36}$	$\sqrt{(5-2)^2+(4-1)^2+(7-4)^2}=\sqrt{27}$
P6	$\sqrt{(6-1)^2+(5-2)^2+(8-3)^2}=\sqrt{59}$	$\sqrt{(6-2)^2+(5-1)^2+(8-4)^2}=\sqrt{48}$

تعیین انتصاب

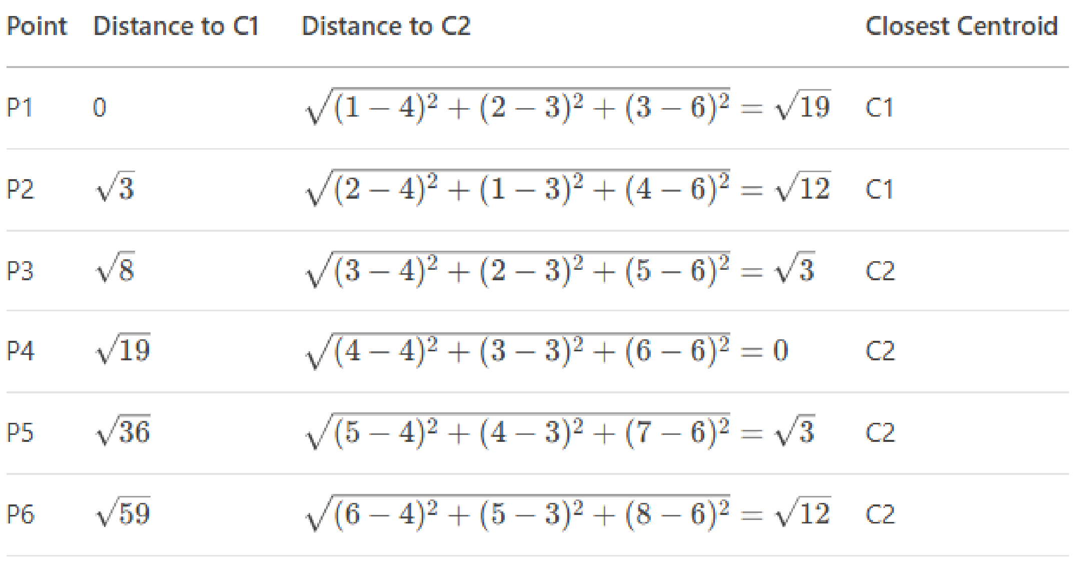
Cluster 1 (C1): {P1}

Cluster 2 (C2): {P2, P3, P4, P5, P6}

بروز رسانی مراکز

برای c1 که فقط p1 را داریم که همان به عنوان c1 در مرحله بعد خواهدبود. برای c2 :

X1=(2+3+4+5+6)/5=4,   *X2*=(1+2+3+4+5)/5=3,   *X3*=(4+5+6+7+8)/5=6



Point	Distance to C1	Distance to C2	Closest Centroid
P1	0	$\sqrt{(1-4)^2+(2-3)^2+(3-6)^2}=\sqrt{19}$	C1
P2	$\sqrt{3}$	$\sqrt{(2-4)^2+(1-3)^2+(4-6)^2}=\sqrt{12}$	C1
P3	$\sqrt{8}$	$\sqrt{(3-4)^2+(2-3)^2+(5-6)^2}=\sqrt{3}$	C2
P4	$\sqrt{19}$	$\sqrt{(4-4)^2+(3-3)^2+(6-6)^2}=0$	C2
P5	$\sqrt{36}$	$\sqrt{(5-4)^2+(4-3)^2+(7-6)^2}=\sqrt{3}$	C2
P6	$\sqrt{59}$	$\sqrt{(6-4)^2+(5-3)^2+(8-6)^2}=\sqrt{12}$	C2

در یک فولدر مانند DM-9-TXTs تعدادی فایل متنی داریم.
اگر هر کدام از این فایل ها به مانند یک مستند باشد هدف آن
است که بردار tf-idf مربوط به هر کدام را بدست آوریم:

```
import os
from sklearn.feature_extraction.text import
TfidfVectorizer

# Folder containing your text files
folder_path = "DM-9-TXTs"

# Read all files and store their contents
documents = []
file_names = []

for file_name in os.listdir(folder_path):
    if file_name.endswith(".txt"):
        file_names.append(file_name)
        with open(os.path.join(folder_path,
file_name), 'r', encoding='utf-8') as file:
            documents.append(file.read())
# Compute TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix =
vectorizer.fit_transform(documents)

feature_names =
vectorizer.get_feature_names_out()
```

```
# Print TF-IDF vectors
for i, file_name in enumerate(file_names):
    print(f"TF-IDF for {file_name}:")
    vector = tfidf_matrix[i]
    for index, value in zip(vector.indices, vector.data):
        print(f"  {feature_names[index]}: {value}")
    print()

from sklearn.cluster import KMeans

num_clusters = 3  # We want three clusters
kmeans = KMeans(n_clusters=num_clusters,
random_state=42)
kmeans.fit(tfidf_matrix)

# Interpret the Results
clusters = kmeans.labels_

# Display the files in each cluster
for cluster_num in range(num_clusters):
    print(f"Cluster {cluster_num + 1}:")
    for i, label in enumerate(clusters):
        if label == cluster_num:
            print(f"  {file_names[i]}")
    print()
```