k-Means και Clustering Evaluation

1 Εισαγωγή

1.1 Εισαγωγή στον k-Means

Ο k-Means είναι ένας αλγόριθμος διαχωρισμού που χρησιμοποιείται για να ομαδοποιήσει τα δείγματα ενός σετ x1, x2, … σε ομάδες C1, C2, … με κέντρα m1, m2, … Ο σκοπός του αλγορίθμου είναι να βρει την ομαδοποίηση που ελαχιστοποιεί το άθροισμα των τετραγώνων των αποστάσεων κάθε σημείου από το κέντρο της ομάδας που ανήκει (Within cluster Sum of Squares – WSS ή SSE):

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - m_i\|^2$$

όπου K είναι ο αριθμός των ομάδων (clusters) και C_i είναι η i-οστή ομάδα που έχει κέντρο (centroid) το m_i. Ο αλγόριθμος επαναλαμβάνει διαδοχικά δύο βήματα: την ανάθεση σε κέντρα των ομάδων και την ενημέρωση των κέντρων. Με δεδομένα κάποια αρχικά κέντρα, αρχικά ο αλγόριθμος τοποθετεί κάθε σημείο στην ομάδα που βρίσκεται το κοντινότερο κέντρο. Στη συνέχεια, αφού έχουν τοποθετηθεί όλα τα σημεία σε ομάδες, για κάθε ομάδα υπολογίζεται το νέο κέντρο ως ο μέσος όρος όλων των σημείων της ομάδας.

Η πιο γνωστή παραλλαγή του αλγορίθμου είναι ο αλγόριθμος k-Medoids, ο οποίος χρησιμοποιεί σημεία από τα δεδομένα ως medoids των clusters (αντί για μέσους όρους σημείων όπως ο k-Means). Έτσι, επιτρέπει τη χρήση κατηγορικών δεδομένων καθώς και διαφόρων συναρτήσεων απόστασης.

1.2 k-Means στην Python

Για ομαδοποίηση με χρήση του k-Means στην Python, χρησιμοποιούμε τη βιβλιοθήκη KMeans:

```
>>> from sklearn.cluster import KMeans

>>> kmeans = KMeans(n_clusters=3, random_state=0)
```

Η εντολή μπορεί να πάρει ως όρισμα είτε τα αρχικά κέντρα είτε τον αριθμό των κέντρων που θα θέλαμε να βρει ο αλγόριθμος οπότε τα κέντρα αρχικοποιούνται τυχαία.

Μπορούμε επίσης να εμφανίσουμε τα centroids και την κατανομή των σημείων σε clusters:

```
>>> print(kmeans.cluster_centers_)
```

```
>>> print(kmeans.labels_)
```

1.3 k-Medoids στην Python

Για να χρησιμοποιήσουμε τον αλγόριθμο k-Medoids χρειαζόμαστε τη βιβλιοθήκη sklearn_extra.cluster:

```
>>> from sklearn_extra.cluster import KMedoids
```

Για να εφαρμόσουμε τον αλγόριθμο για 3 clusters εκτελούμε την εντολή:

```
>>> kmedoids = KMedoids(n_clusters=3, method="pam").fit(conferences)
```

Μπορούμε να εμφανίσουμε τα medoids με την εντολή `kmedoids.cluster_centers_` και την κατανομή των σημείων σε clusters με την εντολή `kmedoids.labels_`.

1.4 Clustering Evaluation στην Python

Για να αξιολογήσουμε ένα μοντέλο ομαδοποίησης, έχοντας το αποτέλεσμα του k-means (`kmeans`) μπορούμε αρχικά να το λάβουμε το cohesion (within-cluster sum of squares), ενώ το separation (between-cluster sum of squares) πρέπει να το υπολογίσουμε:

```
>>> cohesion = kmeans.inertia_

>>> separation = 0

>>> distance = lambda x1, x2: math.sqrt(((x1.X - x2.X) ** 2) + ((x1.Y
- x2.Y) ** 2))

>>> m = data.mean()

>>> for i in list(set(kmeans.labels_)):

>>>     mi = data.loc[kmeans.labels_ == i, :].mean()

>>>     Ci = len(data.loc[kmeans.labels_ == i, :].index)

>>>     separation += Ci * (distance(m, mi) ** 2)
```

Για τον υπολογισμό του silhouette εκτελούμε την εντολή:

```
>>> silhouette_score(data, kmeans.labels_)
```

όπου `data` είναι τα αρχικά δεδομένα. Στη συνέχεια, μπορούμε να σχεδιάσουμε το silhouette plot με την βιβλιοθήκη `yellowbrick.cluster` ή να εμφανίσουμε το μέσο silhouette για ένα cluster με την εντολή:

```
>>> mean(silhouette_samples(data, kmeans.labels_)[kmeans.labels_ ==
0]).
```

2 Κατασκευή Μοντέλου k-Means

Για την κατασκευή ενός μοντέλου k-Means θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα ομαδοποίησης.

	X	**Y**
x1	7	1
x2	3	4
x3	1	5
x4	5	8
x5	1	3
x6	7	8
x7	8	2
x8	5	9

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα.

β) Εφαρμόστε τον αλγόριθμο K-means ώστε να ομαδοποιηθούν τα σημεία σε 3 ομάδες. Θεωρήστε πως τα αρχικά κέντρα είναι τα x1, x2 και x3.

γ) Υπολογίστε το cohesion και το separation της τελικής ομαδοποίησης.

δ) Κατασκευάστε το μοντέλο στην Python και επαναλάβετε τα ερωτήματα (β) και (γ).

ε) Σχεδιάστε εκ νέου τα δεδομένα με διαφορετικά χρώματα για κάθε cluster και στο ίδιο διάγραμμα σχεδιάστε επίσης τα centroids.

2.1 Κατασκευή Δεδομένων και Εισαγωγή Βιβλιοθηκών

Αρχικά κατασκευάζουμε τα δεδομένα με τις παρακάτω εντολές:

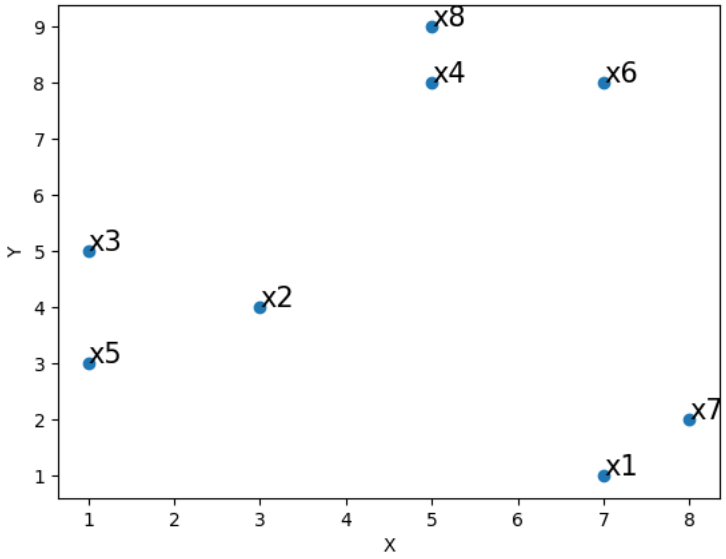
```
>>> import pandas as pd
>>> X = [7, 3, 1, 5, 1, 7, 8, 5]
>>> Y = [1, 4, 5, 8, 3, 8, 2, 9]
>>> labels = ["x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8"]
>>> kdata = pd.DataFrame({"X": X, "Y": Y}, index=labels)
```

Στη συνέχεια μπορούμε να τα σχεδιάσουμε με τις εντολές:

```
>>> import matplotlib.pyplot as plt
>>> plt.scatter(kdata.X, kdata.Y)
```

```
>>> for i in range(len(kdata.index)):

>>>     plt.text(kdata.loc[labels[i], "X"], kdata.loc[labels[i],
"Y"], '%s' % (str(labels[i])), size=15, zorder=1)

>>> plt.show()
```

Τα δεδομένα φαίνονται στο σχήμα:



2.2 Υπολογισμός k-Means

Για το ερώτημα (β) εφαρμόζουμε τον k-Means με αρχικά κέντρα τα A1, A2, A3:

$z_1 = \begin{bmatrix} 7 \\ 1 \end{bmatrix}$ $z_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ $z_3 = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$

1η επανάληψη

Υπολογίζουμε τις αποστάσεις όλων των σημείων από τα κέντρα:

	z1	**z2**	**z3**
x1	**0.00**	5.00	7.21
x2	5.00	**0.00**	2.24
x3	7.21	2.24	**0.00**
x4	7.28	**4.47**	5.00
x5	6.32	2.24	**2.00**
x6	7.00	**5.66**	6.71
x7	**1.41**	5.39	7.62
x8	8.25	**5.39**	5.66

Τα νέα κέντρα είναι:

$$z_1' = \frac{x1 + x7}{2} = \begin{bmatrix} 7.5 \\ 1.5 \end{bmatrix} \qquad z_2' = \frac{x2 + x4 + x6 + x8}{4} = \begin{bmatrix} 5 \\ 7.25 \end{bmatrix} \qquad z_3' = \frac{x3 + x5}{2} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

Τα κέντρα έχουν αλλάξει οπότε συνεχίζουμε με την επόμενη επανάληψη του αλγορίθμου.

2η επανάληψη

Υπολογίζουμε τις αποστάσεις όλων των σημείων από τα κέντρα:

	z1΄	**z2΄**	**z3΄**
x1	**0.71**	6.56	6.71
x2	5.15	3.82	**2.00**
x3	7.38	4.59	**1.00**
x4	6.96	**0.75**	5.66
x5	6.67	5.84	**1.00**
x6	6.52	**2.14**	7.21
x7	**0.71**	6.05	7.28
x8	7.91	**1.75**	6.40

Τα νέα κέντρα είναι:

$$z_1'' = \frac{x1 + x7}{2} = \begin{bmatrix} 7.5 \\ 1.5 \end{bmatrix} \qquad z_2'' = \frac{x4 + x6 + x8}{3} = \begin{bmatrix} 5.67 \\ 8.33 \end{bmatrix} \qquad z_3'' = \frac{x2 + x3 + x5}{3} = \begin{bmatrix} 1.67 \\ 4 \end{bmatrix}$$

Τα κέντρα έχουν αλλάξει οπότε συνεχίζουμε με την επόμενη επανάληψη του αλγορίθμου.

3η επανάληψη

Υπολογίζουμε τις αποστάσεις όλων των σημείων από τα κέντρα:

	z1΄	**z2΄**	**z3΄**
x1	**0.71**	7.45	6.12
x2	5.15	5.09	**1.33**
x3	7.38	5.73	**1.20**
x4	6.96	**0.75**	5.21
x5	6.67	7.09	**1.20**
x6	6.52	**1.37**	6.67
x7	**0.71**	6.75	6.64
x8	7.91	**0.94**	6.01

Τα νέα κέντρα είναι:

$$z_1''' = \frac{x1 + x7}{2} = \begin{bmatrix} 7.5 \\ 1.5 \end{bmatrix} \qquad z_2''' = \frac{x4 + x6 + x8}{3} = \begin{bmatrix} 5.67 \\ 8.33 \end{bmatrix} \qquad z_3''' = \frac{x2 + x3 + x5}{3} = \begin{bmatrix} 1.67 \\ 4 \end{bmatrix}$$

Τα κέντρα έχουν σταθεροποιηθεί οπότε ο αλγόριθμος έχει συγκλίνει στις 3 επαναλήψεις.

2.3 Υπολογισμός Cohesion και Separation

Θεωρούμε τα clusters C_1, C_2, C_3 με τα centroids m_1, m_2, m_3 όπως υπολογίστηκαν παραπάνω.

Για το (γ), υπολογίζουμε το Cohesion (Within cluster Sum of Squares) για κάθε cluster:

$$WSS(C_1) = \sum_{x \in C_1} \|x - m_1\|^2 = \|x_1 - m_1\|^2 + \|x_7 - m_1\|^2$$

$$= \sqrt{(7 - 7.5)^2 + (1 - 1.5)^2}^2 + \sqrt{(8 - 7.5)^2 + (2 - 1.5)^2}^2$$

$$= 0.25 + 0.25 + 0.25 + 0.25 = 1$$

$$WSS(C_2) = \sum_{x \in C_2} \|x - m_2\|^2 = \|x_4 - m_2\|^2 + \|x_6 - m_2\|^2 + \|x_8 - m_2\|^2$$

$$= \sqrt{(5 - 5.67)^2 + (8 - 8.33)^2}^2 + \sqrt{(7 - 5.67)^2 + (8 - 8.33)^2}^2 + \sqrt{(5 - 5.67)^2 + (9 - 8.33)^2}^2$$

$$= 0.4489 + 0.1089 + 1.7689 + 0.1089 + 0.4489 + 0.4489 = 3.3334$$

$$WSS(C_3) = \sum_{x \in C_3} \|x - m_3\|^2 = \|x_2 - m_2\|^2 + \|x_3 - m_2\|^2 + \|x_5 - m_2\|^2$$

$$= \sqrt{(3 - 1.67)^2 + (4 - 4)^2}^2 + \sqrt{(1 - 1.67)^2 + (5 - 4)^2}^2 + \sqrt{(1 - 1.67)^2 + (3 - 4)^2}^2$$

$$= 1.7689 + 0 + 0.4489 + 1 + 0.4489 + 1 = 4.6667$$

Το συνολικό Cohesion για όλα τα clusters θα είναι:

$$WSS_{Total} = \sum_i WSS(C_i) = WSS(C_1) + WSS(C_2) + WSS(C_3) = 1 + 3.3334 + 4.6667 = 9$$

Για το Separation θα υπολογίσουμε αρχικά το μέσο όρο των δεδομένων:

$$m = \frac{x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8}{8} = \begin{bmatrix} 4.625 \\ 5 \end{bmatrix}$$

Το Separation (Between cluster Sum of Squares) θα είναι:

$$BSS_{Total} = \sum_i |C_i| \cdot \|m - m_1\|^2$$

$$= 2 \cdot \sqrt{(4.625 - 7.5)^2 + (5 - 1.5)^2}^2 + 3 \cdot \sqrt{(4.625 - 5.67)^2 + (5 - 8.33)^2}^2$$

$$+ 3 \cdot \sqrt{(4.625 - 1.67)^2 + (5 - 4)^2}^2$$

$$= 41.031 + 36.589 + 29.255 = 106.875$$

2.3 Κατασκευή Μοντέλου με την Python

Για το ερώτημα (δ) εφαρμόζουμε τον k-Means για το dataset:

```
>>> from sklearn.cluster import KMeans

>>> kmeans = KMeans(n_clusters=3, init=kdata.loc[["x1", "x2", "x3"],
:]).fit(kdata)
```

Εμφανίζουμε αν θέλουμε τα centroids και την κατανομή των σημείων σε clusters:

```
>>> print(kmeans.cluster_centers_)

>>> print(kmeans.labels_)
```

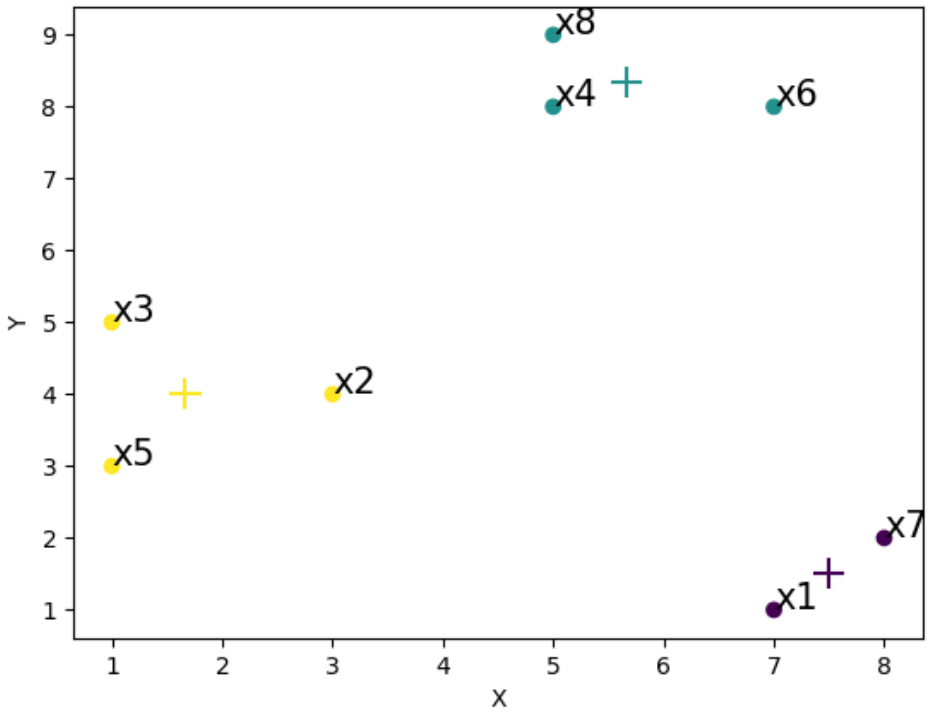
Υπολογίζουμε εκ νέου το ερώτημα (γ):

```
>>> print(kmeans.inertia_)

>>> separation = 0

>>> distance = lambda x1, x2: math.sqrt(((x1.X - x2.X) ** 2) + ((x1.Y
- x2.Y) ** 2))

>>> m = kdata.mean()

>>> for i in list(set(kmeans.labels_)):

>>>     mi = kdata.loc[kmeans.labels_ == i, :].mean()

>>>     Ci = len(kdata.loc[kmeans.labels_ == i, :].index)

>>>     separation += Ci * (distance(m, mi) ** 2)

>>> print(separation)
```

Μπορούμε τέλος να σχεδιάσουμε τα δεδομένα με τα clusters (ερώτημα (ε)) με τις εντολές:

```
>>> plt.scatter(kdata.X, kdata.Y, c=kmeans.labels_)

>>> for i in range(len(kdata.index)):

>>>     plt.text(kdata.loc[labels[i], "X"], kdata.loc[labels[i],
"Y"], '%s' % (str(labels[i])), size=15, zorder=1)

>>> plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], marker="+", s=169, c=range(3))

>>> plt.xlabel("X")

>>> plt.ylabel("Y")

>>> plt.show()
```

Οπότε προκύπτει η ομαδοποίηση που φαίνεται στο σχήμα:



3 Εφαρμογή με k-Means και Μετρικές Αξιολόγησης

Για την κατασκευή ενός μοντέλου k-Means θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του αρχείου που δίνεται (cdata.txt) για ένα πρόβλημα ομαδοποίησης.

Ένα summary των δεδομένων είναι το παρακάτω:

```
       X1                X2                Y
 Min.   :-4.720   Min.   :-3.974   Min.   :1
 1st Qu.:-1.237   1st Qu.:-1.059   1st Qu.:1
 Median : 1.757   Median : 2.096   Median :2
 Mean   : 1.322   Mean   : 1.532   Mean   :2
 3rd Qu.: 3.592   3rd Qu.: 3.675   3rd Qu.:3
 Max.   : 6.077   Max.   : 6.689   Max.   :3
```

Αρχικά, κάνουμε import τη βιβλιοθήκη cluster, εισάγουμε τα δεδομένα και τα διαχωρίζουμε:

```
>>> cdata = pd.read_csv("./cdata.txt")
>>> target = cdata.loc[:, "Y"]
>>> cdata = cdata.loc[:, ["X1", "X2"]]
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τις τιμές του συνόλου δεδομένων με διαφορετικά χρώματα για κάθε κατηγορία.

β) Εφαρμόζοντας τον αλγόριθμο k-Means επιλέξτε τον ελάχιστο αριθμό των clusters έτσι ώστε να περιγράφουν ικανοποιητικά τα δεδομένα με βάση το SSE.

γ) Εφαρμόστε τον k-Means για 3 clusters.

δ) Υπολογίστε το cohesion και το separation για την ομαδοποίηση που προέκυψε.

ε) Σχεδιάστε εκ νέου τα δεδομένα με διαφορετικά χρώματα για κάθε cluster και στο ίδιο διάγραμμα σχεδιάστε επίσης τα centroids.

στ) Υπολογίστε το silhouette και σχεδιάστε το silhouette plot για την ομαδοποίηση που προέκυψε.

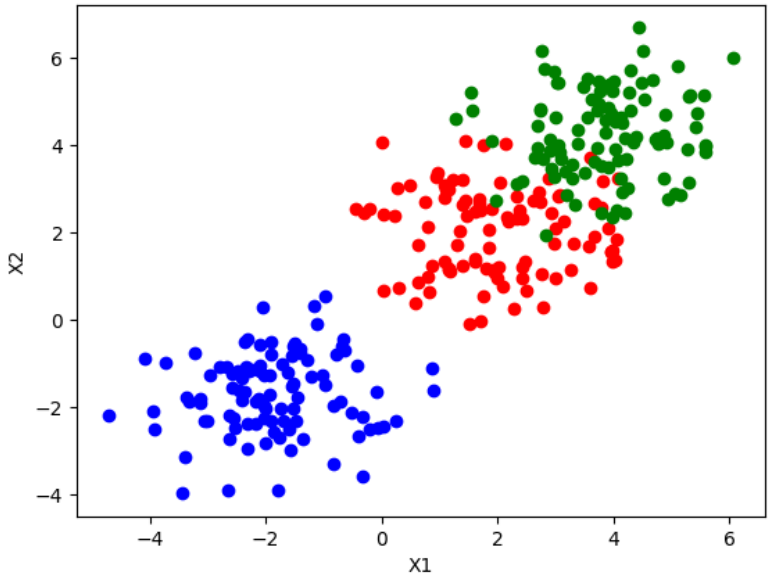
ζ) Σχεδιάστε το heatmap για την ομαδοποίηση που προέκυψε.

3.1 Επιλογή Αριθμού Clusters με τον k-Means

Αρχικά, μπορούμε να σχεδιάσουμε τα δεδομένα με την παρακάτω εντολή (ερώτημα (α)):

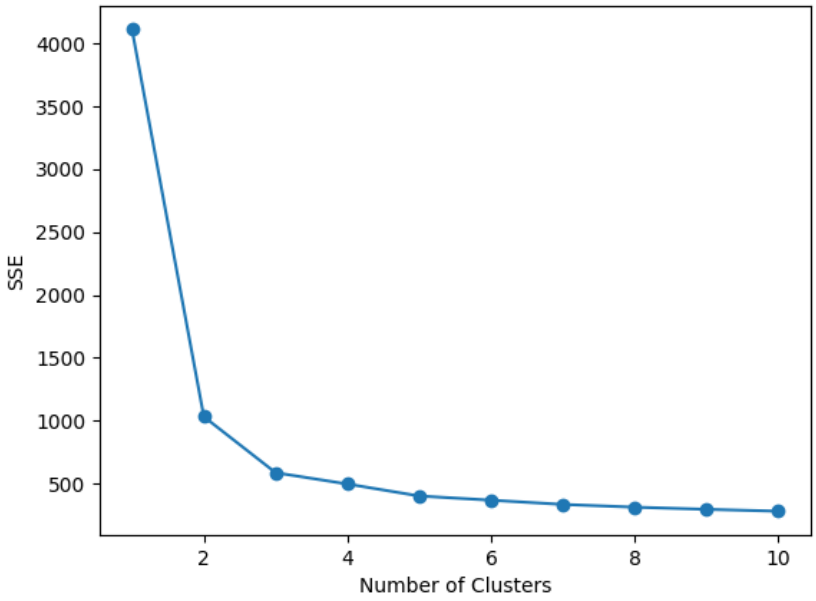
```
>>> plt.scatter(cdata[(target == 1)].X1, cdata[(target == 1)].X2,
c="red", marker="o")
```

```
>>> plt.scatter(cdata[(target == 2)].X1, cdata[(target == 2)].X2,
c="blue", marker="o")

>>> plt.scatter(cdata[(target == 3)].X1, cdata[(target == 3)].X2,
c="green", marker="o")

>>> plt.show()
```



Για να βρούμε ένα πλήθος clusters που να περιγράφουν τα δεδομένα (ερώτημα (β)), υπολογίζουμε το SSE για τους διαχωρισμούς σε 1, 2, …, 10 clusters με τις παρακάτω εντολές:

```
>>> sse = []

>>> for i in range(1, 11):

>>>     sse.append(KMeans(n_clusters=i, init=cdata.loc[0:i-1,
:]).fit(cdata).inertia_)

>>> plt.plot(range(1, 11), sse)

>>> plt.scatter(range(1, 11), sse, marker="o")

>>> plt.show()
```



3.2 Κατασκευή Μοντέλου k-Means

Για το ερώτημα (γ) εφαρμόζουμε τον k-Means για το dataset:

```
>>> kmeans = KMeans(n_clusters=3, init=cdata.loc[0:2, :]).fit(cdata)
```

Εμφανίζουμε αν θέλουμε τα centroids και την κατανομή των σημείων σε clusters:

```
>>> print(kmeans.cluster_centers_)
>>> print(kmeans.labels_)
```

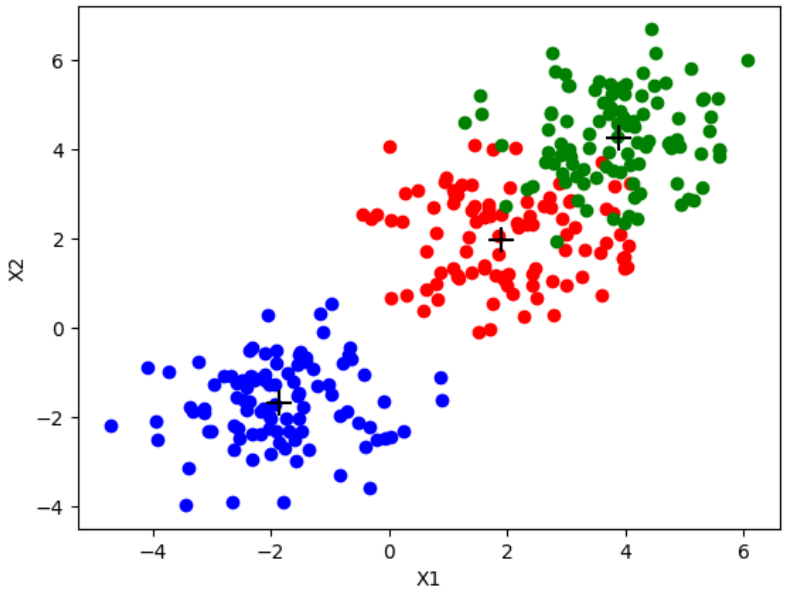
Υπολογίζουμε το cohesion και το separation (ερώτημα (δ)):

```
>>> print(kmeans.inertia_)
>>> separation = 0
>>> distance = lambda x1, x2: math.sqrt(((x1.X1 - x2.X1) ** 2) +
((x1.X2 - x2.X2) ** 2))
>>> m = cdata.mean()
>>> for i in list(set(kmeans.labels_)):
>>>     mi = cdata.loc[kmeans.labels_ == i, :].mean()
>>>     Ci = len(cdata.loc[kmeans.labels_ == i, :].index)
>>>     separation += Ci * (distance(m, mi) ** 2)
>>> print(separation)
```

Μπορούμε τέλος να σχεδιάσουμε τα δεδομένα με τα clusters (ερώτημα (ε)) με τις εντολές:

```
>>> plt.scatter(cdata[(target == 1)].X1, cdata[(target == 1)].X2,
c="red", marker="o")

>>> plt.scatter(cdata[(target == 2)].X1, cdata[(target == 2)].X2,
c="blue", marker="o")

>>> plt.scatter(cdata[(target == 3)].X1, cdata[(target == 3)].X2,
c="green", marker="o")

>>> plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], marker="+", s=169, color="black")

>>> plt.show()
```

Οπότε προκύπτει η ομαδοποίηση που φαίνεται στο σχήμα:



3.3 Υπολογισμός Silhouette και Κατασκευή Silhouette Plot

Για τον υπολογισμό του silhouette (ερώτημα (στ)) εισάγουμε τις συναρτήσεις:

```
>>> from sklearn.metrics import silhouette_samples, silhouette_score
```

Στη συνέχεια μπορούμε να υπολογίσουμε το μέσο silhouette για την ομαδοποίηση:

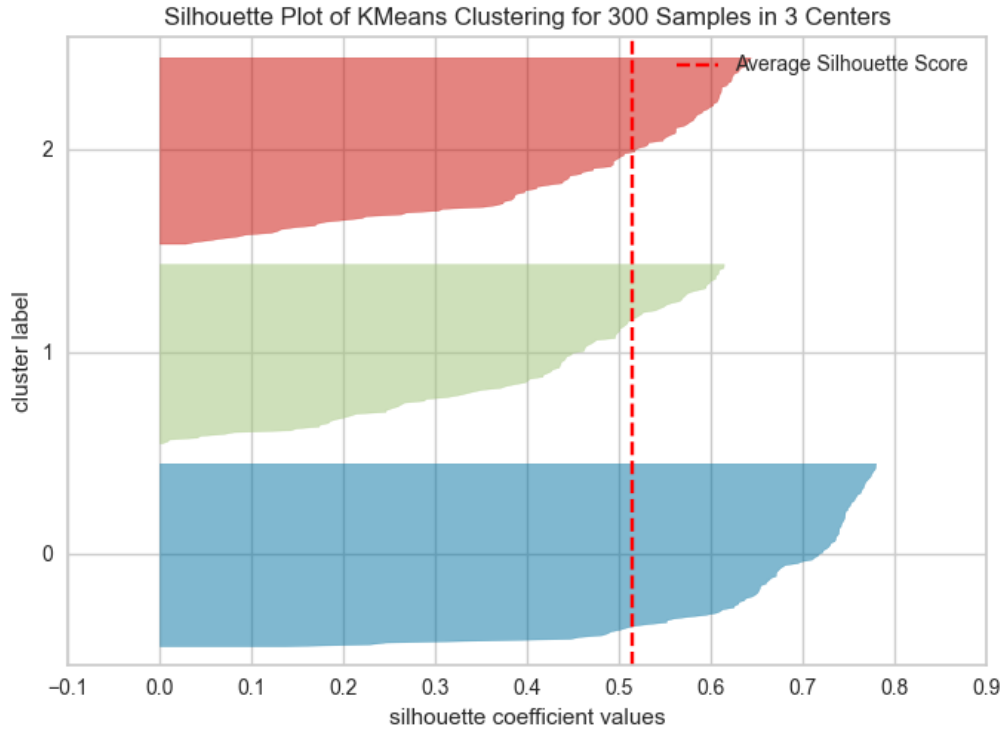
```
>>> print(silhouette_score(cdata, kmeans.labels_))
```

Επίσης, μπορούμε να υπολογίσουμε το μέσο silhouette για κάθε cluster ξεχωριστά:

```
>>> print(mean(silhouette_samples(cdata,
kmeans.labels_)[kmeans.labels_ == 0]))

>>> print(mean(silhouette_samples(cdata,
kmeans.labels_)[kmeans.labels_ == 1]))

>>> print(mean(silhouette_samples(cdata,
kmeans.labels_)[kmeans.labels_ == 2]))
```

Τέλος, μπορούμε να οπτικοποιήσουμε τον συντελεστή silhouette για κάθε σημείο με τις παρακάτω εντολές:

```
>>> from yellowbrick.cluster import SilhouetteVisualizer

>>> visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')

>>> visualizer.fit(cdata)

>>> visualizer.show()
```



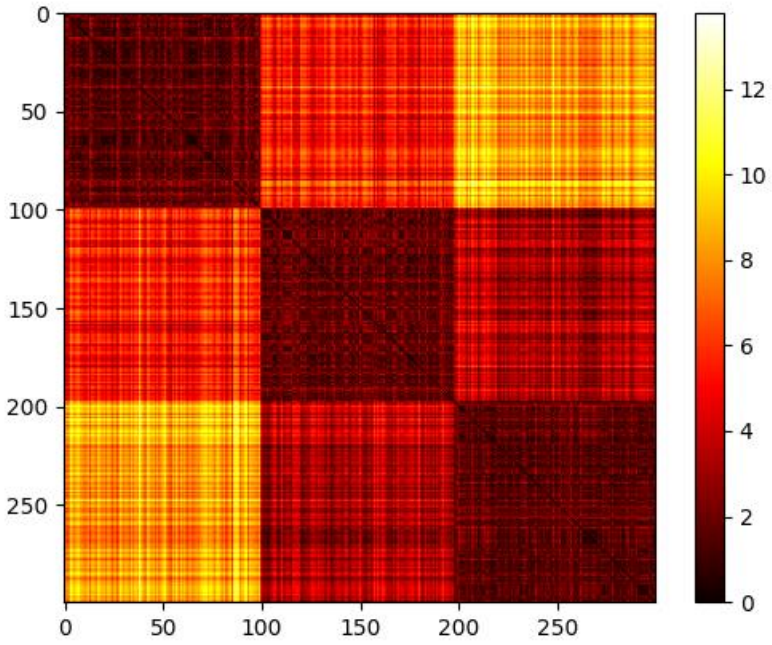
3.4 Κατασκευή Heatmap

Για να κατασκευάσουμε το heatmap (ερώτημα (ζ)) αρχικά ταξινομούμε τα δεδομένα με βάση το cluster στο οποίο ανήκουν:

```
>>> cdata["cluster"] = kmeans.labels_
>>> cdata = cdata.sort_values("cluster").drop("cluster", axis=1)
```

Στη συνέχεια, σχεδιάζουμε το heatmap με την παρακάτω εντολή:

```
>>> from scipy.spatial import distance_matrix
>>> dist = distance_matrix(cdata, cdata)
>>> plt.imshow(dist, cmap='hot')
>>> plt.colorbar()
>>> plt.show()
```



4 Εφαρμογή με k-Medoids

Για την κατασκευή ενός μοντέλου k-Medoids θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα ομαδοποίησης.

Rank	Topic
High	SE
Low	SE
High	ML
Low	DM
Low	ML
High	SE

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Εφαρμόστε τον αλγόριθμο k-Medoids ώστε να ομαδοποιηθούν τα δεδομένα σε 3 ομάδες. Εμφανίστε τα κέντρα των 3 ομάδων.

4.1 Κατασκευή Δεδομένων και Εισαγωγή Βιβλιοθηκών

Εισάγουμε τη βιβλιοθήκη cluster και κατασκευάζουμε τα δεδομένα με τις εντολές:

```
>>> Rank = ["High", "Low", "High", "Low", "Low", "High"]
>>> Topic = ["SE", "SE", "ML", "DM", "ML", "SE"]
>>> conferences = pd.DataFrame({"Rank": Rank, "Topic": Topic})
```

4.2 Κατασκευή Μοντέλου k-Medoids

Για το ερώτημα (α) εφαρμόζουμε τον k-Medoids για το dataset (μπορείτε να εγκαταστήσετε τη βιβλιοθήκη sklearn_extra με την εντολή `pip install scikit-learn-extra`):

```
>>> from sklearn_extra.cluster import KMedoids
>>> from sklearn.preprocessing import OneHotEncoder
>>> encoder = OneHotEncoder(handle_unknown="ignore", sparse=False)
>>> encoder = encoder.fit(conferences)
>>> conferences = encoder.transform(conferences)
>>> kmedoids = KMedoids(n_clusters=3, method="pam").fit(conferences)
```

Εμφανίζουμε τα medoids και την κατανομή των σημείων σε clusters:

```
>>> print(kmedoids.cluster_centers_)
```

```
>>> print(kmedoids.labels_)
```

Μπορούμε επίσης να εμφανίσουμε τα κέντρα των ομάδων με τις μεταβλητές τους:

```
>>> print(conferences.loc[kmedoids.medoid_indices_, :])
```