Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

9ο Εξάμηνο
Φθινόπωρο 2021

Hierarchical Clustering και Density-based Clustering

1 Εισαγωγή

1.1 Εισαγωγή στο Hierarchical Clustering

Το hierarchical clustering είναι μια μέθοδος ομαδοποίησης που χρησιμοποιείται για να κατασκευάσει μια ιεραρχία από clusters. Στην bottom up (agglomerative) εκδοχή της μεθόδου, αρχικά κάθε σημείο έχει το δικό του cluster. Στη συνέχεια, τα clusters συνενώνονται με βάση την απόστασή τους. Υπάρχουν διαφορετικοί τρόποι ορισμού της απόστασης μεταξύ δύο clusters, όπως για παράδειγμα την απόσταση μεταξύ των δύο κοντινότερων σημείων των clusters, την απόσταση μεταξύ των δύο μακρινότερων σημείων των clusters, κ.α. Επίσης, είναι δυνατή η χρήση διαφορετικών μετρικών απόστασης (π.χ. ευκλείδεια, manhattan κ.α.). Εφόσον έχει κατασκευαστεί μια ιεραρχία με όλα τα clusters, επιλέγεται μια θέση αποκοπής όπου χωρίζονται τα clusters.

1.2 Εισαγωγή στο Density-based Clustering

Το density-based clustering αφορά το διαχωρισμό σε clusters με βάση την πυκνότητα των δεδομένων. Ο πιο γνωστός αλγόριθμος σε αυτήν την κατηγορία είναι ο DBSCAN (Density-based spatial clustering of applications with noise). Για ένα σύνολο σημείων ο DBSCAN ομαδοποιεί τα σημεία που είναι κοντά μεταξύ τους, δηλαδή που η απόσταση μεταξύ ενός σημείου του cluster με ένα τουλάχιστον άλλο σημείο είναι μεγαλύτερη ή ίση από την παράμετρο epsilon. Υπάρχει επίσης η παράμετρος minPoints που ορίζει τον ελάχιστο αριθμό σημείων που πρέπει να υπάρχουν σε κάθε cluster. Τα σημεία για τα οποία δεν ικανοποιούνται τα κριτήρια των παραμέτρων epsilon και minPoints θέτονται ως outliers.

1.3 Hierarchical Clustering στην Python

Για ομαδοποίηση με χρήση hierarchical clustering στην Python, χρησιμοποιούμε τη βιβλιοθήκη `AgglomerativeClustering`:

```
>>> from sklearn.cluster import AgglomerativeClustering

>>> clustering = AgglomerativeClustering(n_clusters=2,
linkage="single").fit(X)
```

Με την παράμετρο linkage επιλέγουμε τον τρόπο ορισμού της απόστασης μεταξύ δύο clusters. Μπορούμε να επιλέξουμε την απόσταση μεταξύ των δύο κοντινότερων σημείων των clusters (`single`), την απόσταση μεταξύ των δύο μακρινότερων σημείων των clusters (`complete`), τη μέση απόσταση μεταξύ όλων των σημείων των clusters (`average`), τη μείωση του τετραγωνικού σφάλματος που θα προκύψει αν συνενωθούν τα clusters (`ward.D2`) κ.α.

Μπορούμε να σχεδιάσουμε το δενδρόγραμμα με τις εντολές:

```
>>> from scipy.cluster.hierarchy import dendrogram

>>> clustering = AgglomerativeClustering(n_clusters=None,
linkage="single", distance_threshold=0).fit(hdata)

>>> linkage_matrix = np.column_stack([clustering.children_,
clustering.distances_, np.ones(4)]).astype(float)

>>> dendrogram(linkage_matrix, labels=labels)

>>> plt.show()
```

Μπορούμε επίσης να εμφανίσουμε την κατανομή των σημείων σε clusters:

```
>>> clusters = clustering.labels_
```

1.4 DBSCAN στην Python

Για να χρησιμοποιήσουμε τον αλγόριθμο DBSCAN χρειαζόμαστε τη βιβλιοθήκη DBSCAN:

```
>>> from sklearn.cluster import DBSCAN
```

Για να εφαρμόσουμε τον αλγόριθμο εκτελούμε την εντολή:

```
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
```

Όπου μπορούμε να θέσουμε τις τιμές των παραμέτρων epsilon και MinPoints με τα `eps` και `min_samples` αντίστοιχα.

Μπορούμε να εμφανίσουμε την κατανομή των σημείων σε clusters με την εντολή

```
>>> clusters = clustering.labels_
```

Στη μεταβλητή clusters υπάρχουν επίσης τιμές -1 για τα σημεία εκείνα που ο DBSCAN τα έθεσε ως θόρυβο (outliers).

2 Κατασκευή Hierarchical Clustering

Για την ομαδοποίηση δεδομένων με hierarchical clustering θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα του παρακάτω πίνακα.

	X	**Y**
x1	2	0
x2	8	4
x3	0	6
x4	7	2
x5	6	1

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα.

β) Εφαρμόστε στα δεδομένα hierarchical clustering με single linkage.

γ) Εφαρμόστε στα δεδομένα hierarchical clustering με complete linkage.

δ) Κατασκευάστε τα μοντέλα στην Python και επαναλάβετε τα ερωτήματα (β) και (γ). Σχεδιάστε το δενδρόγραμμα που προκύπτει σε κάθε περίπτωση.

ε) Διαχωρίστε τα δεδομένα σε δύο clusters χρησιμοποιώντας τα μοντέλα του ερωτήματος (δ). Για κάθε μοντέλο, σχεδιάστε εκ νέου τα δεδομένα με διαφορετικά χρώματα για κάθε cluster.

2.1 Κατασκευή Δεδομένων

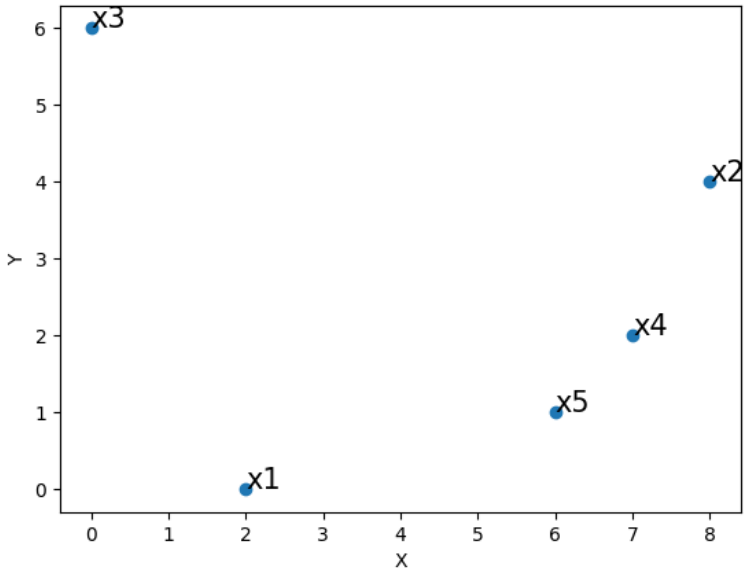
Αρχικά κατασκευάζουμε τα δεδομένα με τις παρακάτω εντολές:

```
>>> X = [2, 8, 0, 7, 6]

>>> Y = [0, 4, 6, 2, 1]

>>> labels = ["x1", "x2", "x3", "x4", "x5"]

>>> hdata = pd.DataFrame({"X": X, "Y": Y}, index=labels)
```

Στη συνέχεια μπορούμε να τα σχεδιάσουμε με τις εντολές:

```
>>> plt.scatter(hdata.X, hdata.Y)

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

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

>>> plt.show()
```

Τα δεδομένα φαίνονται στο σχήμα:



2.2 Υπολογισμός Single Linkage Hierarchical Clustering

Για το ερώτημα (β) εφαρμόζουμε Single Linkage Hierarchical Clustering.

1η επανάληψη

Ο πίνακας αποστάσεων είναι:

	x1	**x2**	**x3**	**x4**	**x5**
x1	0	7.21	6.32	5.39	4.12
x2	7.21	0	8.25	2.24	3.61
x3	6.32	8.25	0	8.06	7.81
x4	5.39	2.24	8.06	0	**1.41**
x5	4.12	3.61	7.81	**1.41**	0

Ενώνουμε τα σημεία x4 και x5.

2η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1	**x2**	**x3**	**x4-x5**
x1	0	7.21	6.32	4.12
x2	7.21	0	8.25	**2.24**
x3	6.32	8.25	0	7.81
x4-x5	4.12	**2.24**	7.81	0

Ενώνουμε τα σημεία x2 και x4-x5.

3η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1	**x2-x4-x5**	**x3**
x1	0	**4.12**	6.32
x2-x4-x5	**4.12**	0	7.81
x3	6.32	7.81	0

Ενώνουμε τα σημεία x1 και x2-x4-x5.

4η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1-x2-x4-x5	**x3**
x1-x2-x4-x5	0	**6.32**
x3	**6.32**	0

Ενώνουμε τα σημεία x3 και x1-x2-x4-x5.

2.3 Υπολογισμός Complete Linkage Hierarchical Clustering

Για το ερώτημα (γ) εφαρμόζουμε Complete Linkage Hierarchical Clustering.

1η επανάληψη

Ο πίνακας αποστάσεων είναι:

	x1	**x2**	**x3**	**x4**	**x5**
x1	0	7.21	6.32	5.39	4.12
x2	7.21	0	8.25	2.24	3.61
x3	6.32	8.25	0	8.06	7.81
x4	5.39	2.24	8.06	0	**1.41**
x5	4.12	3.61	7.81	**1.41**	0

Ενώνουμε τα σημεία x4 και x5.

2η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1	**x2**	**x3**	**x4-x5**
x1	0	7.21	6.32	5.39
x2	7.21	0	8.25	**3.61**
x3	6.32	8.25	0	8.06
x4-x5	5.39	**3.61**	8.06	1.41

Ενώνουμε τα σημεία x2 και x4-x5.

3η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1	**x2-x4-x5**	**x3**
x1	0	7.21	**6.32**
x2-x4-x5	7.21	0	8.25
x3	**6.32**	8.25	0

Ενώνουμε τα σημεία x1 και x3.

4η επανάληψη

Ο νέος πίνακας αποστάσεων είναι:

	x1-x3	**x2-x4-x5**
x1-x3	0	**8.25**
x2-x4-x5	**8.25**	0

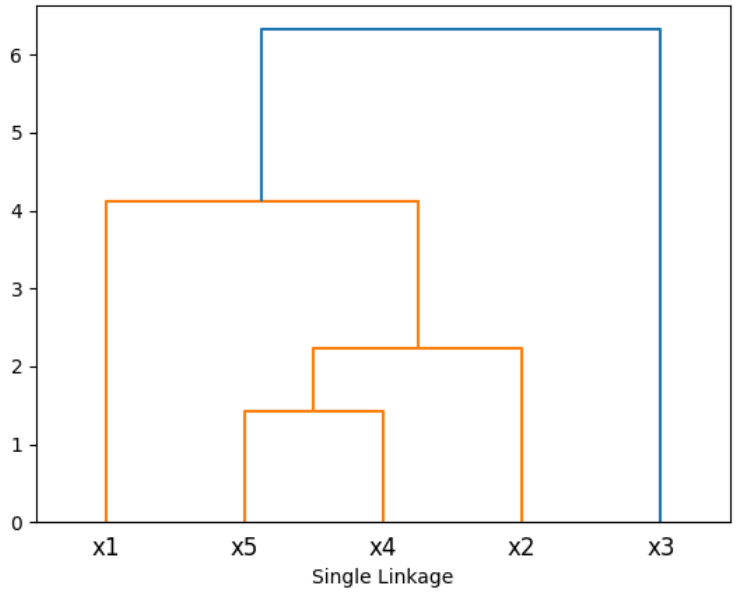
Ενώνουμε τα σημεία x1-x3 και x2-x4-x5.

2.4 Κατασκευή Μοντέλου με την Python

Για το ερώτημα (δ) σχεδιάζουμε αρχικά το δενδρόγραμμα:

```
>>> from scipy.cluster.hierarchy import dendrogram

>>> import numpy as np

>>> clustering = AgglomerativeClustering(n_clusters=None,
linkage="single", distance_threshold=0).fit(hdata)

>>> linkage_matrix = np.column_stack([clustering.children_,
clustering.distances_, np.ones(len(hdata.index)-1)]).astype(float)

>>> dendrogram(linkage_matrix, labels=labels)

>>> plt.show()
```

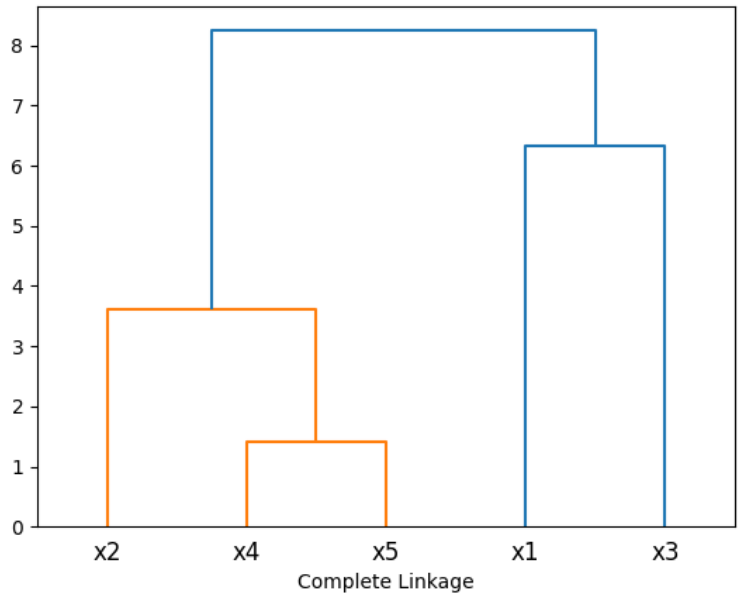
Οπότε προκύπτει το παρακάτω δενδρόγραμμα:



Εφαρμόζουμε complete linkage hierarchical clustering και σχεδιάζουμε το δενδρόγραμμα:

```
>>> clustering = AgglomerativeClustering(n_clusters=None,
linkage="complete", distance_threshold=0).fit(hdata)

>>> linkage_matrix = np.column_stack([clustering.children_,
clustering.distances_, np.ones(len(hdata.index)-1)]).astype(float)

>>> dendrogram(linkage_matrix, labels=labels)

>>> plt.show()
```

Οπότε προκύπτει το παρακάτω δενδρόγραμμα:



Τέλος, μπορούμε να διαχωρίσουμε τα δεδομένα σε δύο clusters με την εντολή:

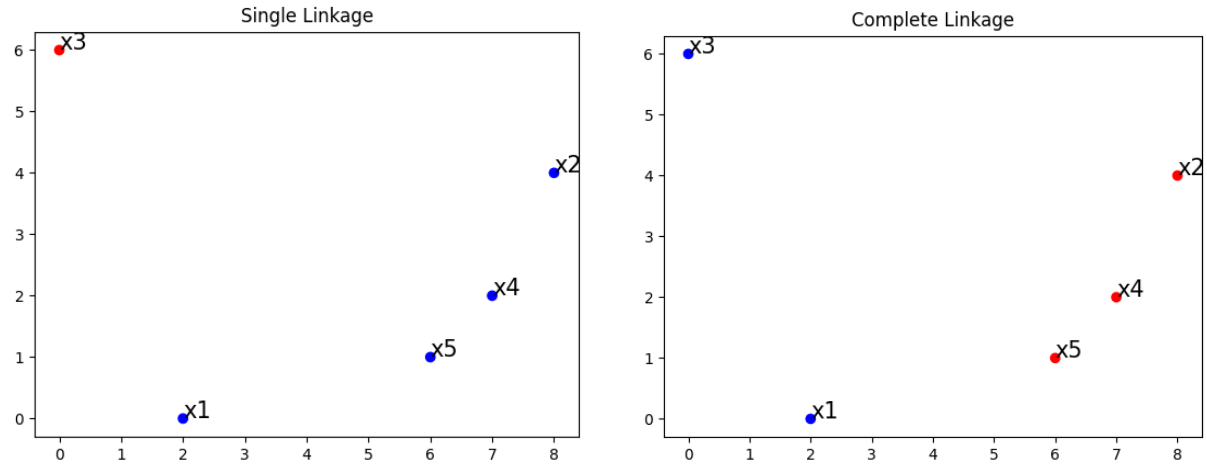
```
>>> clustering = AgglomerativeClustering(n_clusters=2,
linkage="single").fit(hdata)
```

(ή για complete linkage με την εντολή `clustering = AgglomerativeClustering(n_clusters=2, linkage="complete").fit(hdata)`)

Σχεδιάζουμε τα δεδομένα με διαφορετικά χρώματα για κάθε cluster:

```
>>> clustering = AgglomerativeClustering(n_clusters=2,
linkage="single").fit(hdata)

>>> plt.scatter(hdata.X, hdata.Y, c=clustering.labels_, cmap="bwr")

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

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

>>> plt.show()
```

(ή για complete linkage με όρισμα `linkage="complete"`)



3 Εφαρμογή με Hierarchical Clustering

Για την κατασκευή ενός μοντέλου Hierarchical Clustering θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα του αρχείου που δίνεται (europe.txt) για ένα πρόβλημα ομαδοποίησης. Ένα τμήμα των δεδομένων είναι το παρακάτω:

```
                 GDP Inflation Unemployment
Austria        41600       3.5          4.2
Belgium        37800       3.5          7.2
Bulgaria       13800       4.2          9.6
Croatia        18000       2.3         17.7
Czech Republic 27100       1.9          8.5
Denmark        37000       2.8          6.1
...              ...       ...          ...
```

Αρχικά, εισάγουμε τα δεδομένα:

```
>>> europe = pd.read_csv("./europe.txt")
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Εφαρμόστε hierarchical clustering με complete link ώστε να ομαδοποιήσετε τα δεδομένα. Σχεδιάστε το δενδρόγραμμα που προκύπτει.

β) Χρησιμοποιήστε το μέσο silhouette ώστε να επιλέξετε τον αριθμό των clusters.

γ) Ομαδοποιήστε τα δεδομένα σε 7 clusters. Απεικονίστε τα clusters σε δενδρόγραμμα.

δ) Σχεδιάστε τα δεδομένα σε τρεις διαστάσεις με διαφορετικά χρώματα για κάθε cluster.

ε) Υπολογίστε και σχεδιάστε το silhouette.

3.1 Κατασκευή Μοντέλου Hierarchical Clustering

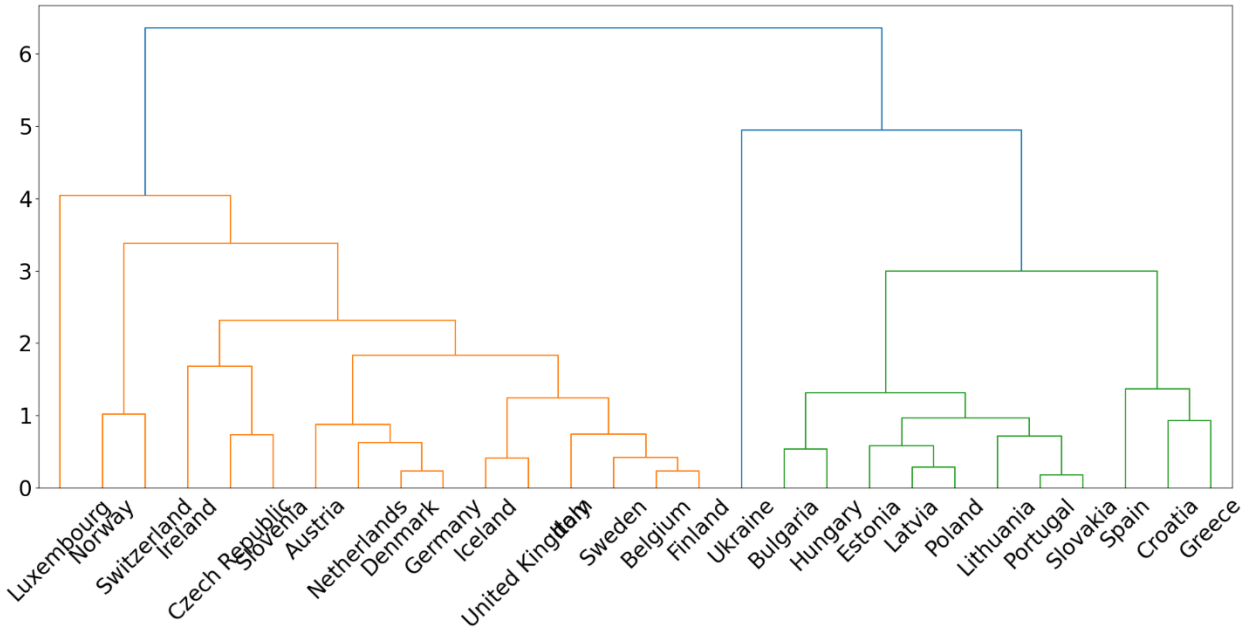
Αρχικά κανονικοποιούμε τα δεδομένα:

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler()
>>> scaler = scaler.fit(europe)
>>> europe = pd.DataFrame(scaler.transform(europe),
columns=europe.columns, index=europe.index)
```

Στη συνέχεια εφαρμόζουμε hierarchical clustering με complete link (ερώτημα (α)) και εμφανίζουμε το δενδρόγραμμα:

```
>>> clustering = AgglomerativeClustering(n_clusters=None,
linkage="complete", distance_threshold=0).fit(europe)
>>> linkage_matrix = np.column_stack([clustering.children_,
clustering.distances_, np.ones(len(europe.index)-1)]).astype(float)
>>> dendrogram(linkage_matrix, labels=europe.index)
>>> plt.show()
```

Οπότε προκύπτει το παρακάτω δενδρόγραμμα:



3.2 Επιλογή Αριθμού Clusters με βάση το Silhouette

Για να βρούμε ένα πλήθος clusters που να περιγράφουν τα δεδομένα (ερώτημα (β)), υπολογίζουμε το silhouette για τους διαχωρισμούς σε 2, …, 20 clusters με τις παρακάτω εντολές:

```
>>> from sklearn.metrics import silhouette_score
>>> slc = []
>>> for i in range(2, 21):
>>>     clustering = AgglomerativeClustering(n_clusters=i, linkage="complete").fit(europe)
>>>     slc.append(silhouette_score(europe, clustering.labels_))
```

Στη συνέχεια μπορούμε να σχεδιάσουμε το silhouette coefficient με τις παρακάτω εντολές:

```
>>> plt.plot(range(2, 21), slc)
>>> plt.xticks(range(2, 21), range(2, 21))
>>> plt.show()
```

3.3 Ομαδοποίηση Δεδομένων σε Clusters

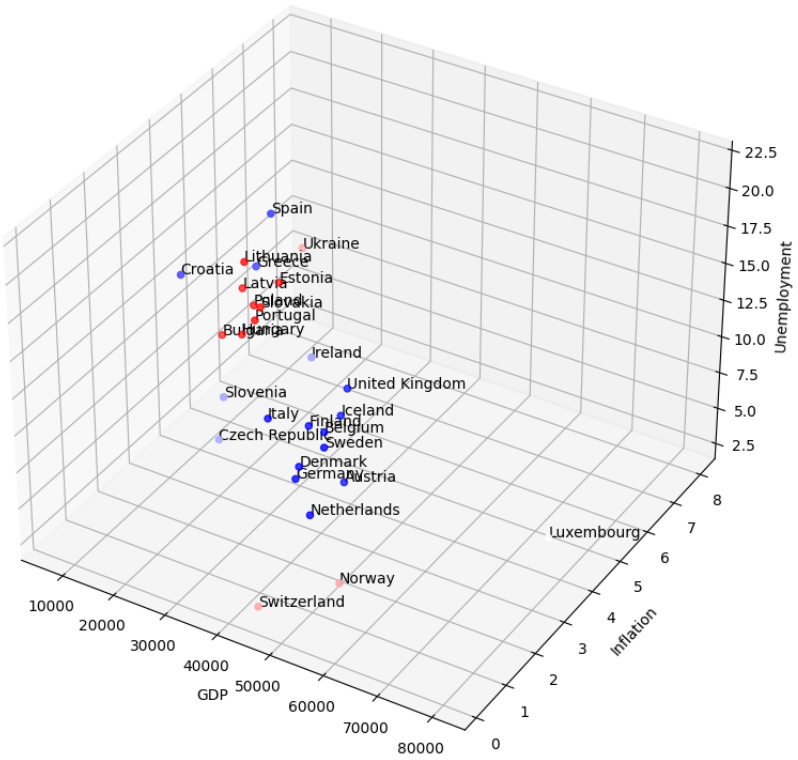
Για το ερώτημα (γ) ομαδοποιούμε τα δεδομένα σε 7 clusters:

```
>>> clustering = AgglomerativeClustering(n_clusters=7,
linkage="complete").fit(europe)
```

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

```
>>. fig = plt.figure()

>>> ax = fig.add_subplot(projection='3d')

>>> ax.scatter(europeData.GDP, europeData.Inflation,
europeData.Unemployment, c=clustering.labels_, cmap="bwr")

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

>>>     ax.text(europeData.loc[europeData.index[i], "GDP"],
europeData.loc[europeData.index[i], "Inflation"],
europeData.loc[europeData.index[i], "Unemployment"], '%s' %
(str(europeData.index[i])), size=20, zorder=1)

>>> ax.set_xlabel('GDP')

>>> ax.set_ylabel('Inflation')

>>> ax.set_zlabel('Unemployment')

>>> plt.show()
```

Οπότε προκύπτει η ομαδοποίηση που φαίνεται στο σχήμα:



3.4 Υπολογισμός Silhouette και Κατασκευή Silhouette Plot

Για τον υπολογισμό του silhouette (ερώτημα (ε)) εκτελούμε την εντολή:

```
>>> print(silhouette_score(europe, clustering.labels_))
```

4 Κατασκευή Density-based Clustering

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

	X	**Y**
x1	2	10
x2	2	5
x3	8	4
x4	5	8
x5	7	5
x6	6	4
x7	1	2
x8	4	9

Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα.

β) Εφαρμόστε στα δεδομένα dbscan με eps = 2 και minPts = 2.

γ) Επαναλάβετε το ερώτημα (β) για eps = 3.5.

δ) Κατασκευάστε τα μοντέλα στην Python και επαναλάβετε τα ερωτήματα (β) και (γ). Σχεδιάστε τα δεδομένα με διαφορετικό χρώμα για κάθε cluster και μαύρο χρώμα για τα σημεία που αποτελούν θόρυβο σε κάθε περίπτωση.

4.1 Κατασκευή Δεδομένων και Εισαγωγή Βιβλιοθηκών

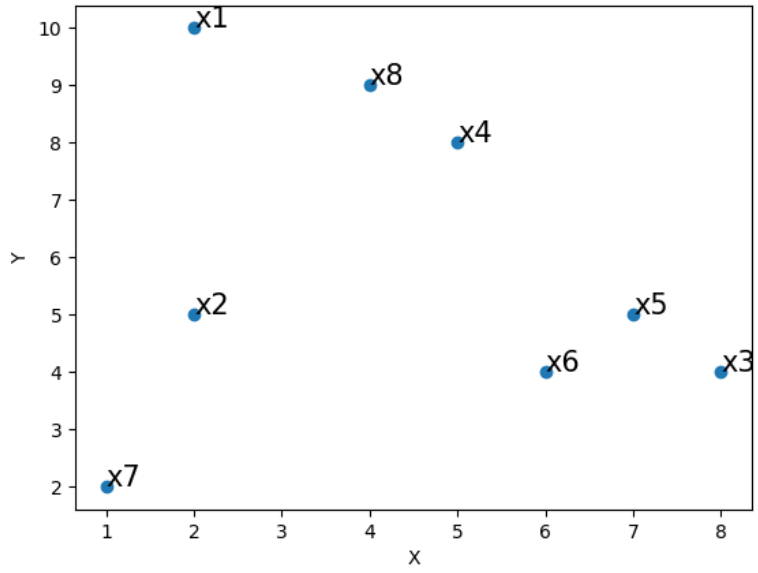
Αρχικά κατασκευάζουμε τα δεδομένα με τις παρακάτω εντολές:

```
>>> X = [2, 2, 8, 5, 7, 6, 1, 4]
>>> Y = [10, 5, 4, 8, 5, 4, 2, 9]
>>> labels = ["x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8"]
>>> ddata = pd.DataFrame({"X": X, "Y": Y}, index=labels)
```

Στη συνέχεια μπορούμε να τα σχεδιάσουμε (ερώτημα (α)) με τις εντολές:

```
>>> plt.scatter(ddata.X, ddata.Y)

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

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

>>> plt.xlabel("X")

>>> plt.ylabel("Y")

>>> plt.show()
```

Τα δεδομένα φαίνονται στο σχήμα:



4.2 Υπολογισμός DBSCAN

Για τα ερωτήματα (β) και (γ) υπολογίζουμε αρχικά τον πίνακα αποστάσεων για τα δεδομένα:

	x1	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**
x1	0.00							
x2	5.00	0.00						
x3	8.49	6.08	0.00					
x4	3.61	4.24	5.00	0.00				
x5	7.07	5.00	1.41	3.61	0.00			
x6	7.21	4.12	2.00	4.12	1.41	0.00		
x7	8.06	3.16	7.28	7.21	6.71	5.39	0.00	
x8	2.24	4.47	6.40	1.41	5.00	5.39	7.62	0.00

Για eps = 2 (ερώτημα (β)) είναι μικρότερες οι αποστάσεις που είναι έντονες στον πίνακα:

	x1	x2	x3	x4	x5	x6	x7	x8
x1	0.00							
x2	5.00	0.00						
x3	8.49	6.08	0.00					
x4	3.61	4.24	5.00	0.00				
x5	7.07	5.00	**1.41**	3.61	0.00			
x6	7.21	4.12	**2.00**	4.12	**1.41**	0.00		
x7	8.06	3.16	7.28	7.21	6.71	5.39	0.00	
x8	2.24	4.47	6.40	**1.41**	5.00	5.39	7.62	0.00

Εφαρμόζοντας τον αλγόριθμο (και με δεδομένο ότι κάθε ομάδα αποτελείται από τουλάχιστον 2 σημεία) σχηματίζονται τα clusters x3-x5-x6 και x4-x8

Για eps = 3 (ερώτημα (γ)) είναι μικρότερες οι αποστάσεις που είναι έντονες στον πίνακα:

	x1	x2	x3	x4	x5	x6	x7	x8
x1	0.00							
x2	5.00	0.00						
x3	8.49	6.08	0.00					
x4	3.61	4.24	5.00	0.00				
x5	7.07	5.00	**1.41**	3.61	0.00			
x6	7.21	4.12	**2.00**	4.12	**1.41**	0.00		
x7	8.06	**3.16**	7.28	7.21	6.71	5.39	0.00	
x8	**2.24**	4.47	6.40	**1.41**	5.00	5.39	7.62	0.00

Εφαρμόζοντας τον αλγόριθμο (και με δεδομένο ότι κάθε ομάδα αποτελείται από τουλάχιστον 2 σημεία) σχηματίζονται τα clusters x1-x4-x8, x2-x7 και x3-x5-x6

4.3 Κατασκευή Μοντέλου με την Python

Μπορούμε να εφαρμόσουμε τον DBSCAN στην Python για eps = 2 εκτελώντας τις εντολές:

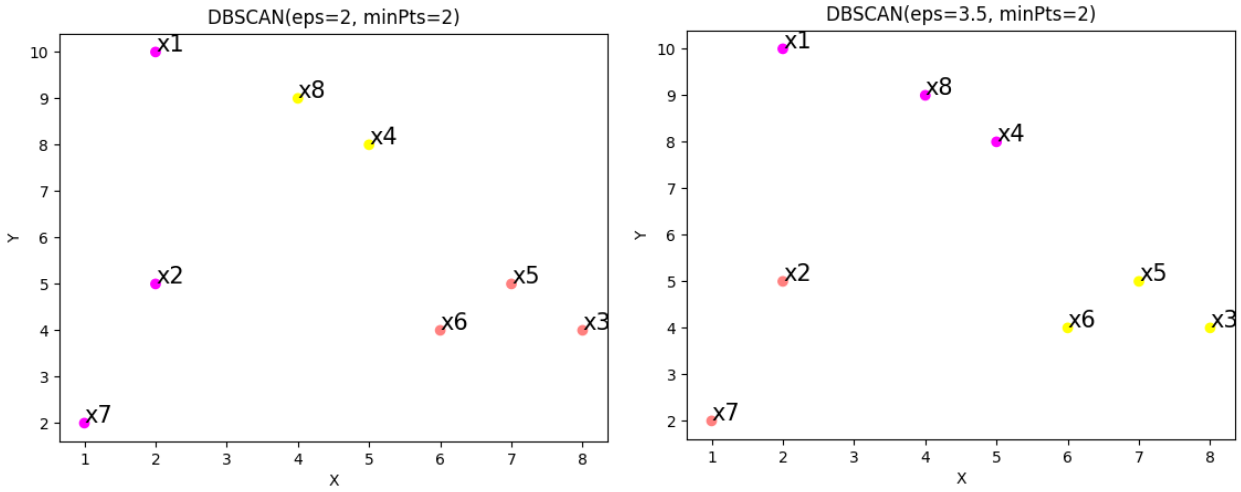
```
>>> from sklearn.cluster import DBSCAN
>>> clustering = DBSCAN(eps=2, min_samples=2).fit(ddata)
```

Για την κατανομή των σημείων σε clusters και τη σχεδίασή τους εκτελούμε τις εντολές:

```
>>> clusters = clustering.labels_
>>> plt.scatter(ddata.X, ddata.Y, c=clusters, cmap="spring")
>>> for i in range(len(ddata.index)):
```

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

>>> plt.xlabel("X")

>>> plt.ylabel("Y")

>>> plt.title("DBSCAN(eps=2, minPts=2)")

>>> plt.show()
```

Όμοια εκτελούμε τις εντολές με eps = 3.5. Οπότε προκύπτουν τα παρακάτω διαγράμματα:



5 Εφαρμογή με Density-based Clustering

Για την κατασκευή ενός μοντέλου Density-based Clustering θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα του αρχείου που δίνεται (mdata.txt) για ένα πρόβλημα ομαδοποίησης. Ένα summary των δεδομένων είναι το παρακάτω:

```
       X                Y
 Min.   : 5.09   Min.   : 7.537
 1st Qu.:10.45   1st Qu.: 9.775
 Median :13.30   Median :12.009
 Mean   :13.28   Mean   :12.027
 3rd Qu.:16.13   3rd Qu.:14.273
 Max.   :21.43   Max.   :16.480
```

Αρχικά εισάγουμε τα δεδομένα:

```
>>> mdata = pd.read_csv("./mdata.txt")
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Σχεδιάστε τα δεδομένα.

β) Εφαρμόστε τον k-Means στα δεδομένα ώστε να τα ομαδοποιήσετε σε 2 clusters. Στη συνέχεια, σχεδιάστε τα δεδομένα με διαφορετικό χρώμα για κάθε cluster. Περιγράφεται ικανοποιητικά το σύνολο δεδομένων;

γ) Υπολογίστε το kNN distance για τα δεδομένα με k ίσο με 10 ώστε να επιλέξετε την παράμετρο eps για τον DBSCAN.

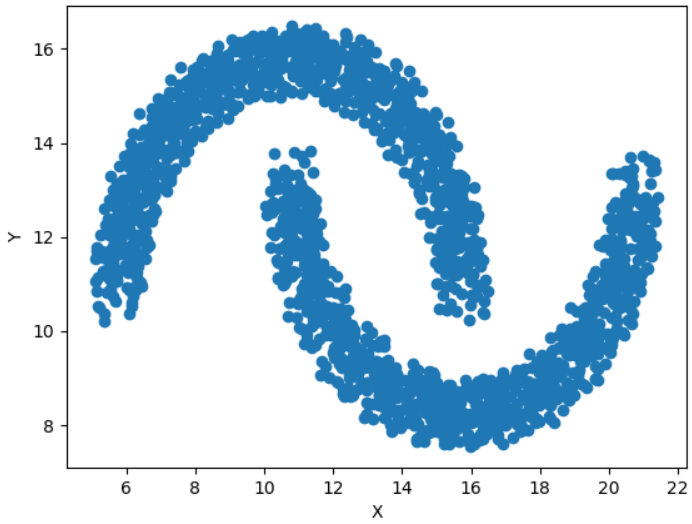
δ) Εφαρμόστε τον DBSCAN με eps = 0.4 και minPts = 10 για να ομαδοποιήσετε τα δεδομένα. Σχεδιάστε τα δεδομένα με διαφορετικό χρώμα για κάθε cluster και μαύρο χρώμα για τα σημεία που αποτελούν θόρυβο σε κάθε περίπτωση.

5.1 Κατασκευή Μοντέλου k-Means

Αρχικά σχεδιάζουμε τα δεδομένα με την εντολή (ερώτημα (α)):

```
>>> plt.scatter(mdata.X, mdata.Y, marker="o")
>>> plt.show()
```

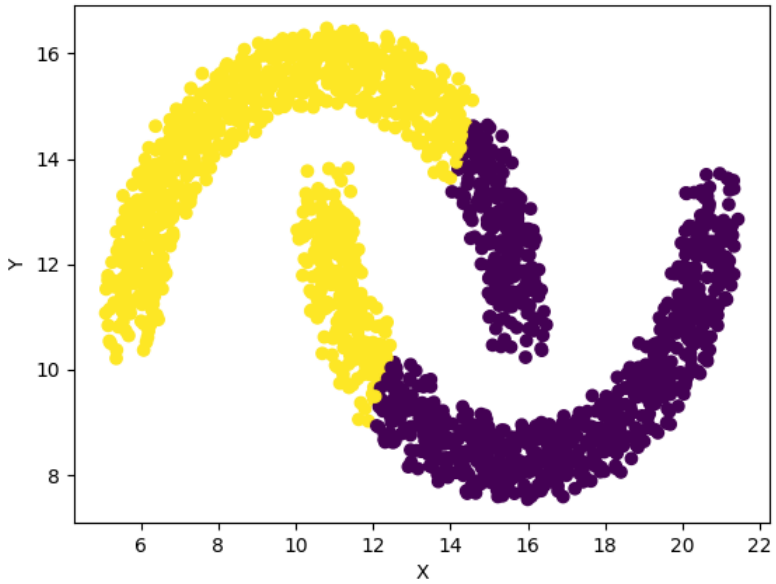
Οπότε προκύπτει το παρακάτω διάγραμμα:



Εφαρμόζουμε τον k-Means για 2 clusters και σχεδιάζουμε τα δεδομένα (ερώτημα (β)):

```
>>> from sklearn.cluster import KMeans
>>> kmeans = KMeans(n_clusters=2).fit(mdata)
>>> plt.scatter(mdata.X, mdata.Y, c=kmeans.labels_, cmap="bwr")
>>> plt.show()
```

Οπότε προκύπτει το παρακάτω διάγραμμα:



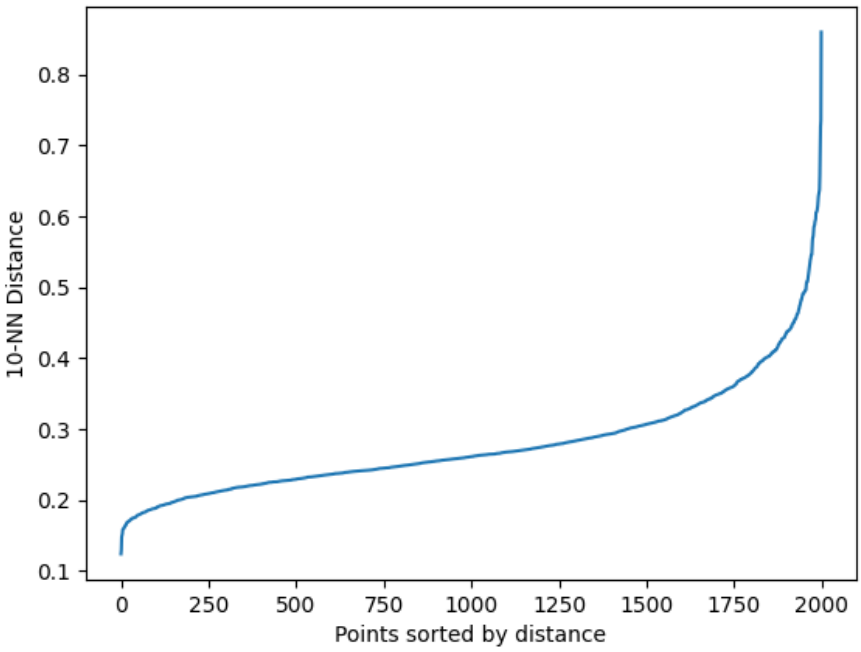
5.2 Επιλογή Παραμέτρου epsilon με βάση το kNN-distance plot

Για να βρούμε μια τιμή για την παράμετρο epsilon (ερώτημα (γ)) θα υπολογίσουμε αρχικά την απόσταση κάθε σημείου από τους 10 κοντινότερους γείτονές του. Εκτελούμε τις εντολές:

```
>>> from sklearn.neighbors import NearestNeighbors
>>> nbrs = NearestNeighbors(n_neighbors=10).fit(mdata)
>>> distances, indices = nbrs.kneighbors(mdata)
```

Στη συνέχεια θα σχεδιάσουμε την απόσταση κάθε σημείου από τον μακρινότερο από αυτούς τους γείτονες, αφού ταξινομήσουμε τα σημεία, με τις εντολές:

```
>>> distanceDec = sorted(distances[:, 9])
>>> plt.plot(distanceDec)
>>> plt.ylabel("10-NN Distance")
>>> plt.xlabel("Points sorted by distance")
>>> plt.show()
```



5.3 Κατασκευή Μοντέλου DBSCAN

Εφαρμόζουμε τον DBSCAN με eps = 2 και minPts = 10 (ερώτημα (δ)):

```
>>> clustering = DBSCAN(eps=0.4, min_samples=10).fit(mdata)
>>> plt.scatter(mdata.X, mdata.Y, c=clustering.labels_)
>>> plt.show()
```

Οπότε προκύπτει το παρακάτω διάγραμμα: