Εξόρυξη Δεδομένων

Εργασία θεωρίας

Αλγόριθμοι συσταδοποίησης – ποιότητα συσταδοποίσησης

Χρήστος Μπίτας

Μέρος 1ο: Διαμεριστική συσταδοποίηση με k-means

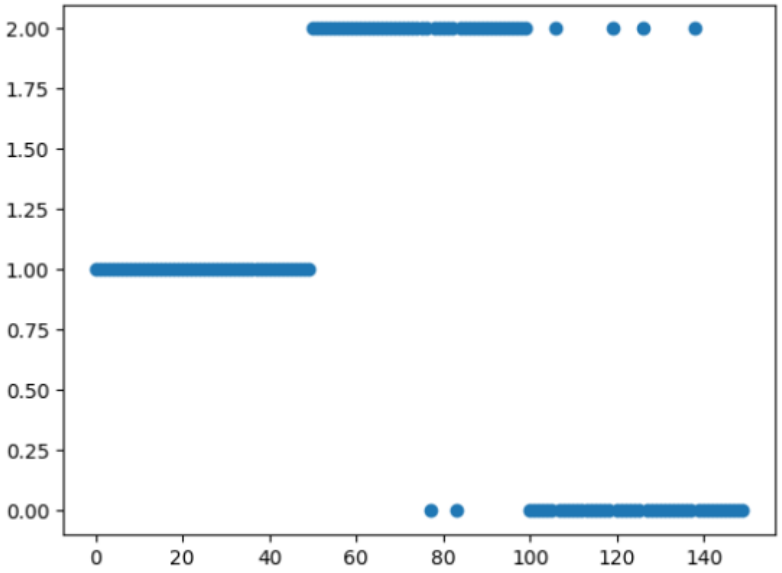
1.1. Εφαρμογή στο σύνολο δεδομένων iris

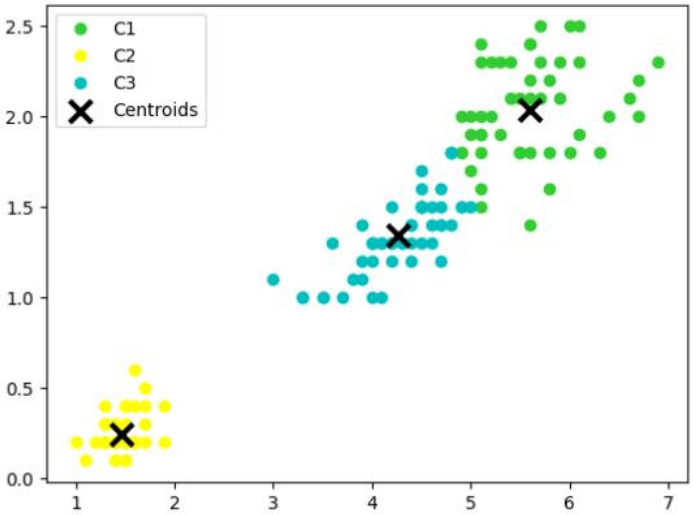
Κώδικας που χρησιμοποιήθηκε:

```
from sklearn.datasets import load_iris
meas = load_iris().data
from sklearn.cluster import KMeans
import math
from sklearn import metrics

# Χρησιμοποιούνται οι 2 τελευταίες διαστάσεις του πίνακα
X = meas[:, [2, 3]]
k = 3 # Ορίζεται ότι τα δεδομένα θα οργανωθούν σε 3 συστάδες
kmeans = KMeans(n_clusters=k).fit(X) # Εφαρμογή του k-means
IDX = kmeans.labels_
C = kmeans.cluster_centers_
import matplotlib.pyplot as plt
plt.figure(1)
# Παρουσιάζεται η κλάση που ανήκει η κάθε παρατήρηση
plt.plot(IDX[:],'o')
plt.show()
plt.plot(X[IDX==0][:,0], X[IDX==0][:,1], 'limegreen',
marker='o', linewidth=0, label='C1')
plt.plot(X[IDX==1][:,0], X[IDX==1][:,1], 'yellow',
marker='o', linewidth=0, label='C2')
plt.plot(X[IDX==2][:,0], X[IDX==2][:,1], 'c.', marker='o',
label='C3')
plt.scatter(C[:,0], C[:,1], marker='x', color='black', s=150
, linewidth=3, label="Centroids", zorder=10)
plt.legend()
plt.show()

# ΕΥΡΕΣΗ SSE

k_table = range(2, 10)
numberOfRows, numberOfColumns = X.shape

sse_table = []
silhouette_table = []
for k in k_table:  # Για κάθε k
    kmeans = KMeans(n_clusters=k)  # Αρχικοποίηση μοντέλου
    kmeans.fit(X)  # Εκπαίδευση μοντέλου
    IDX = kmeans.labels_  # Ετικέτες
    C = kmeans.cluster_centers_  # Κεντροειδή
    sse = 0.0  # Τετραγωνικό σφάλμα
    for i in range(k):  # Για κάθε συστάδα
        for j in range(numberOfRows):
            if IDX[j] == i:
                sse = sse + math.dist(X[j], C[i]) ** 2  #
Υπολόγισε την Ευκλείδεια απόσταση
    sse_table.append(sse)  # Πρόσθεσε το στον πίνακα
```

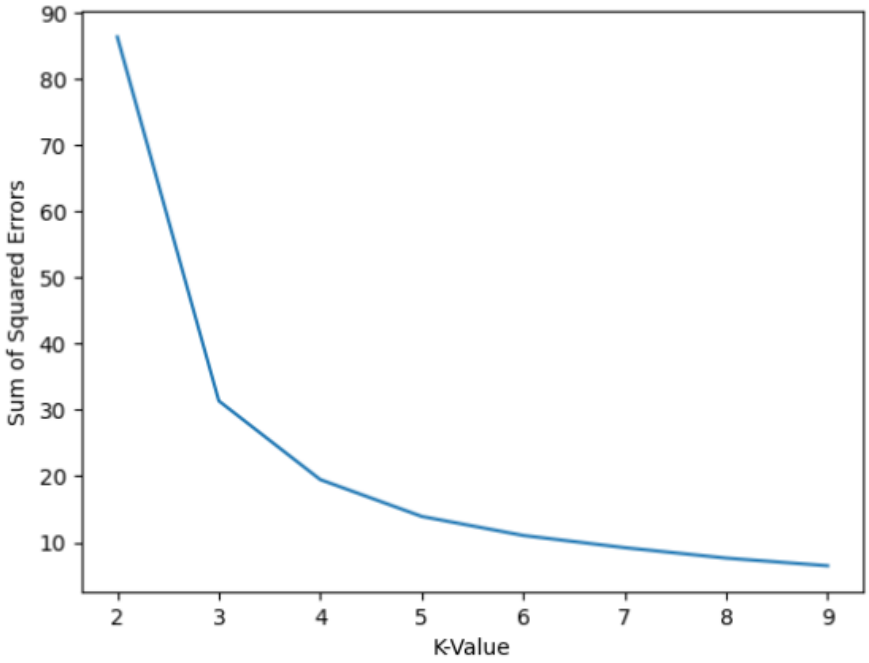
```
    silhouette_table.append(metrics.silhouette_score(X,
IDX))

for k, sil in zip(k_table, silhouette_table):
    print(f"k: {k} || Silhouette Coefficient: %0.3f" %sil)

plt.plot(k_table, sse_table)  # Παρουσίαση
plt.xlabel('K-Value')
plt.ylabel('Sum of Squared Errors')
plt.show()
```

Αποτελέσματα:

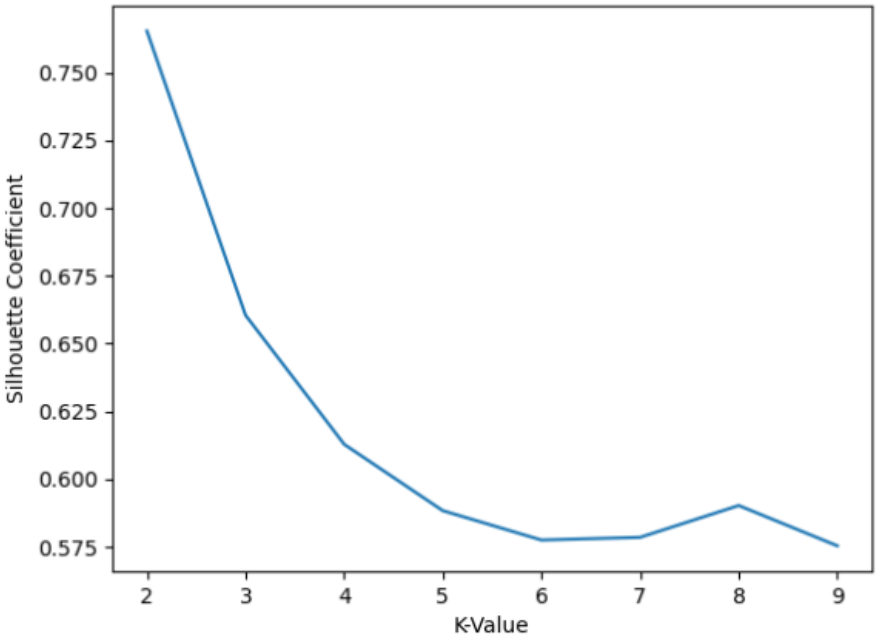




Αφού εφαρμόστηκε ο αλγόριθμος K-μέσων (K-means) χωρίστηκαν τα x δεδομένα (όπου x = 2 τελευταίες διαστάσεις του πίνακα) σε k συστάδες (όπου k = 3 για βήμα 3) προκειμένου τα γίνει παρουσίαση των δεδομένων σε γράφημα όπως φαίνεται στην δεύτερη εικόνα παραπάνω της 3ης σελίδας. Παρατηρούμε την δημιουργία 3 συστάδων όπου στην κάθε συστάδα έχει δημιουργηθεί και το αντίστοιχο κεντροειδές.



Παρατηρώντας το παραπάνω γράφημα όσο αυξάνεται το k παρατηρούμε ότι το sse μειώνεται και μετά το k=5 δεν παρατηρείται κάποια βελτίωση της συσταδοποίησης.



```
k: 2 || Silhouette Coefficient: 0.765
k: 3 || Silhouette Coefficient: 0.660
k: 4 || Silhouette Coefficient: 0.613
k: 5 || Silhouette Coefficient: 0.588
k: 6 || Silhouette Coefficient: 0.578
k: 7 || Silhouette Coefficient: 0.572
k: 8 || Silhouette Coefficient: 0.592
k: 9 || Silhouette Coefficient: 0.587
```

Παρατηρούμε ότι για 2 συστάδες ο αλγόριθμος δίνει silhouette coefficient κοντά στο 1 που σημαίνει οι συστάδες διαχωρίζονται καλά. Για κ=3, που γνωρίζουμε ότι είναι το πραγματικό πλήθος συστάδων το silhouette score αρχίζει να μειώνεται που σημαίνει οι συστάδες αρχίζουν να παρουσιάζουν επικαλύψεις. Δεν υπάρχουν αρνητικές τιμές για κανένα κ.

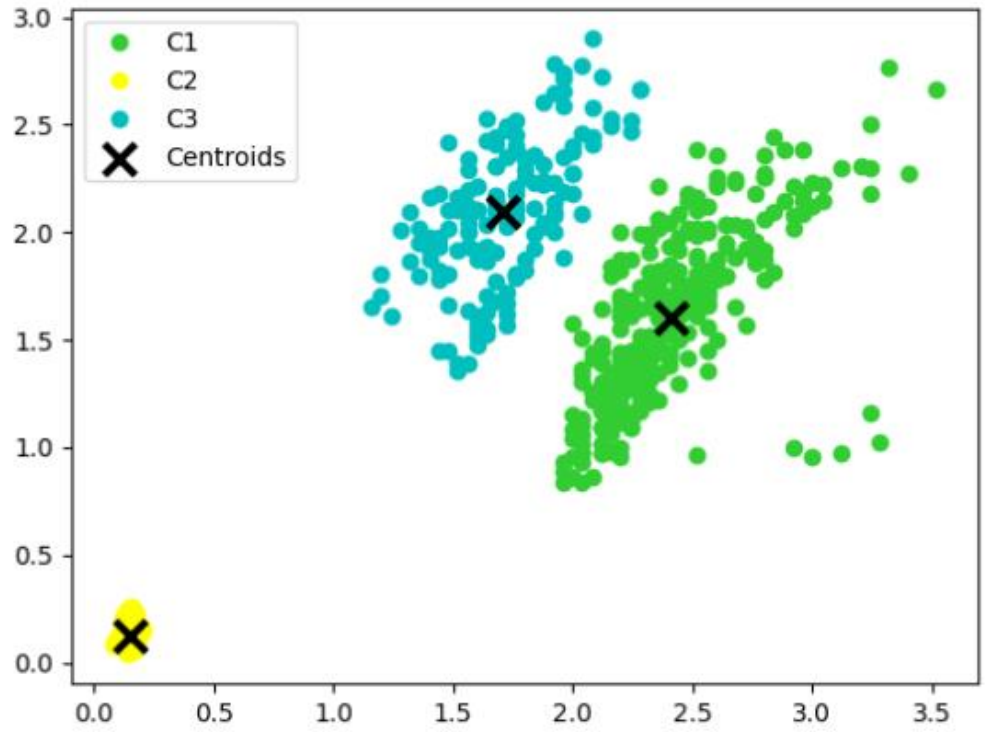
1.2. Εφαρμογή στο σύνολο δεδομένων xV.mat

Κώδικας που χρησιμοποιήθηκε:

```
import math
import scipy.io
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

mat_file = scipy.io.loadmat('xV.mat')
xV = np.array(mat_file['xV'])
def classification(idx_0, idx_1, p_flag=False): # Συνάρτηση για την
αποφυγή επανάληψης του κώδικα σε κάθε περίπτωση
    X = xV[:,[idx_0, idx_1]]
    k = 3
    kmeans = KMeans(n_clusters=k, n_init=10).fit(X)
    IDX = kmeans.labels_
    C = kmeans.cluster_centers_
    if p_flag:
        plt.plot(X[IDX==0][:,0], X[IDX==0][:,1], 'limegreen',
marker='o', linewidth=0, label='C1')
        plt.plot(X[IDX==1][:,0], X[IDX==1][:,1], 'yellow',
marker='o', linewidth=0, label='C2')
        plt.plot(X[IDX==2][:,0], X[IDX==2][:,1], 'c.', marker='o',
label='C3')
```

```
        plt.scatter(C[:,0], C[:,1], marker='x', color='black', s=150
, linewidth=3, label="Centroids", zorder=10)
        plt.legend()
        plt.show()

    numberOfRows, numberOfColumns = X.shape
    sse = 0.0  # Τετραγωνικό σφάλμα
    for i in range(k):  # Για κάθε συστάδα
        for j in range(numberOfRows):
            if IDX[j] == i:
                sse = sse + math.dist(X[j], C[i]) ** 2  # Υπολόγισε
την Ευκλείδεια απόσταση
    print(f"Sum of Squared Errors for columns {idx_0, idx_1}: {sse}")

# Το τρίτο όρισμα είναι μία σημαία για plot

classification(0, 1, True)  # Δύο πρώτες στήλες
classification(296, 305)
classification(467, 468) # Δύο τελευταίες στήλες
classification(205, 175)
```

Αποτελέσματα:




```
Sum of Squared Errors for columns (0, 1): 99.44939463705082
Sum of Squared Errors for columns (296, 305): 11.401819120120058
Sum of Squared Errors for columns (467, 468): 0.330020078971899
Sum of Squared Errors for columns (205, 175): 6.362905489884497
```

Το τετραγωνικό σφάλμα είναι μεγάλο για τις δυο πρώτες στήλες που σημαίνει ότι δεν είναι αντιπροσωπευτικές των κατηγοριών που ανήκουν τα δεδομένα. Για τις επόμενες δοκιμές το σφάλμα μειώνεται ενώ η χαμηλότερη τιμή είναι για τις δυο τελευταίες στήλες (βήμα 4).

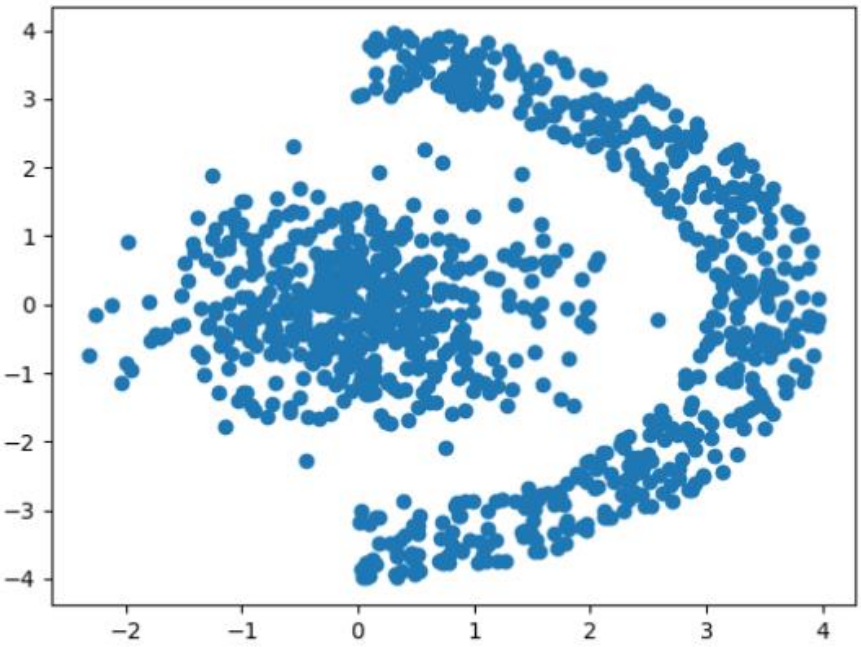
Μέρος 2ο: Συσταδοποίηση βάσει πυκνότητας με DBSCAN

2.1. Εφαρμογή στο σύνολο δεδομένων mydata

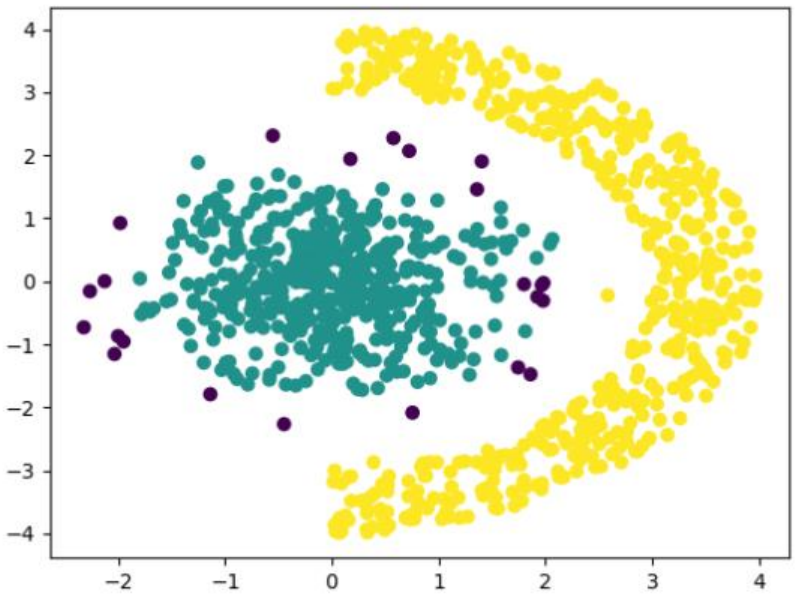
Κώδικας που χρησιμοποιήθηκε:

```
from sklearn import metrics

import scipy.io
import numpy as np
mat_file = scipy.io.loadmat( 'mydata.mat')
X = np.array(mat_file['X'])

from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=15).fit(X)
IDX = dbscan.labels_

import matplotlib.pyplot as plt
plt.figure(1)
plt.scatter(X[:,0],X[:,1])
plt.show()

plt.figure(1)
plt.scatter(X[:,0],X[:,1], c=IDX)
plt.show()

n_clusters_ = len(set(IDX)) - (1 if -1 in IDX else 0)
n_noise_ = list(IDX).count(-1)

print("Silhouette score: ",metrics.silhouette_score(X, IDX))
```

Αποτελέσματα:



Αρχικά έγινε παρουσίαση του γραφήματος διασποράς όπως φαίνεται παραπάνω μετά της εκτέλεση της μεθόδου DBSCAN για τις δύο πρώτες διαστάσεις του πίνακα X.



Στην συνέχεια έγινε παρουσίαση όπως απεικονίζεται παραπάνω το γράφημα διασποράς που προέκυψε από την μέθοδο της DBSCAN η οποία χώρισε τα δεδομένα σε δύο συστάδες και τον θόρυβο.

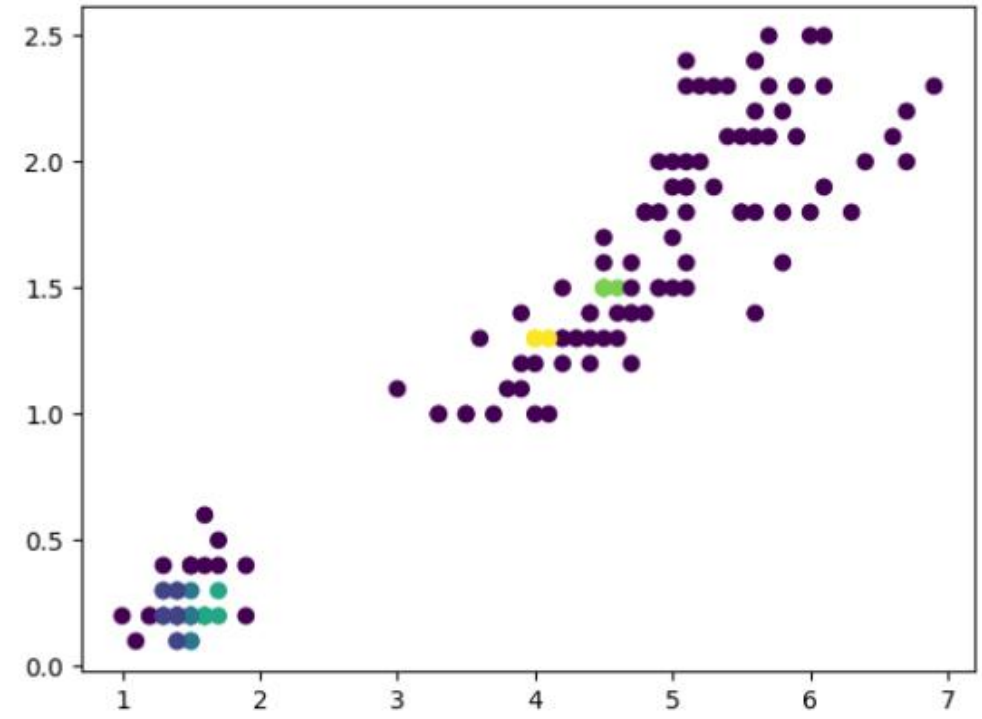
```
Silhouette score:  0.23254693824104214
```

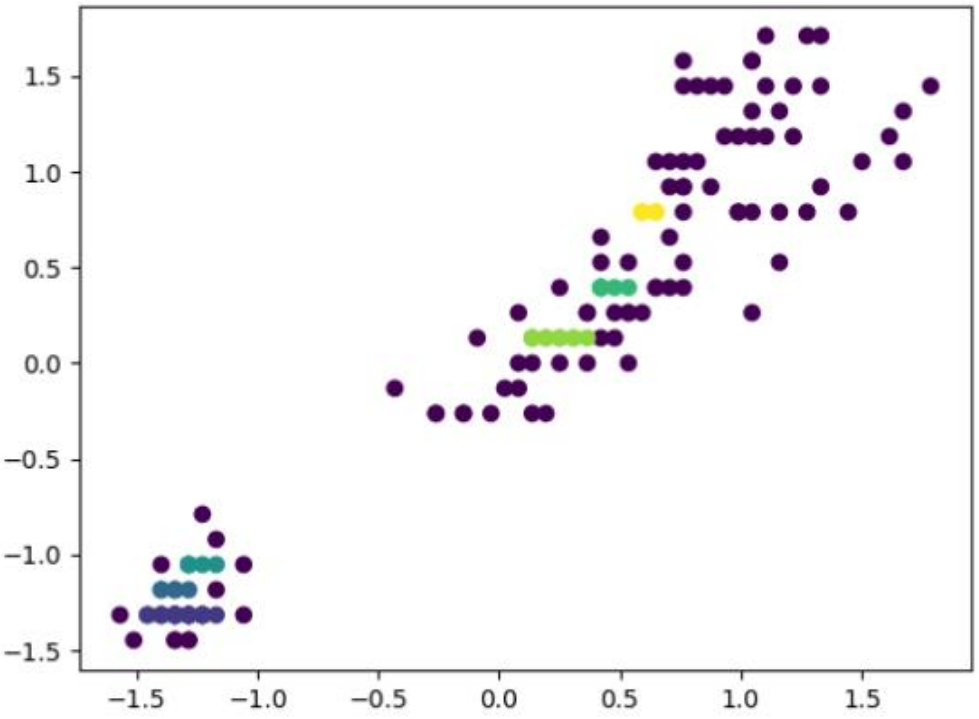
Παρατηρούμε με βάση το σχήμα ότι με κίτρινο χρώμα απεικονίζεται η μία κατηγορία και με πράσινο η άλλη κατηγορία ενώ με μωβ εντοπίζονται μερικά σημεία (θόρυβος) περιμετρικά της πράσινης κατηγορίας οι δυο κατηγορίες είναι ευδιάκριτες και χωρίζονται καλά μεταξύ τους.

2.2. Εφαρμογή στο σύνολο δεδομένων iris

```
import numpy as np
from sklearn import metrics
from sklearn.datasets import load_iris
meas = load_iris().data
X = meas[:, [2, 3]]

from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.1, min_samples=5).fit(X)
IDX = dbscan.labels_

import matplotlib.pyplot as plt

plt.figure(1)
plt.scatter(X[:,0],X[:,1], c=IDX)
plt.show()

print("Silhouette score(Pre normalization):
",metrics.silhouette_score(X, IDX))

from scipy.stats import zscore

xV1 = zscore(X[:,0])
xV2 = zscore(X[:,1])
X = np.array([xV1, xV2]).T  # [xV1, xV2] Έγινε αντικατάσταση της
συγκεκριμένης γραμμής για να
                            # πάρουμε το σωστο πληθος idx

dbscan = DBSCAN(eps=0.1, min_samples=5).fit(X)
IDX = dbscan.labels_
plt.figure(1)
plt.scatter(xV1, xV2, c=IDX)
plt.show()

print("Silhouette score(Post normalization):
",metrics.silhouette_score(X, IDX))
```

Αποτελέσματα:





```
Silhouette score(Pre normalization):  -0.18909852845949932
Silhouette score(Post normalization):  -0.1333635960655027
```

Γενικά καμία από τις δυο μεθόδους δεν αποδίδει καλά. Πριν την κανονικοποίηση και μετρά από αυτήν το silhouette score είναι αρνητικό που σημαίνει ότι υπάρχει έντονη επικάλυψη αναμεσά στις συστάδες, οι οποίες επίσης έχουν ανατεθεί λάθος. Η κανονικοποίηση αυξάνει ελάχιστα το silouehette score το οποίο παραμένει αρνητικό.