מטלת למידת מכונה מתקדם- סיווג- קובץ קוד

**קוד עבור silhouette :**

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_samples, silhouette\_score

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import numpy as np

import pandas as pd

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Generating the sample data from make\_blobs

# This particular setting has one distinct cluster and 3 clusters placed close

# together.

X, y = make\_blobs(n\_samples=3430,

n\_features=1545,

centers=4,

cluster\_std=1,

center\_box=(-10.0, 10.0),

shuffle=True,

random\_state=1) # For reproducibility

range\_n\_clusters = [2, 3, 7, 8]

for n\_clusters in range\_n\_clusters:

# Create a subplot with 1 row and 2 columns

fig, (ax1, ax2) = plt.subplots(1, 2)

fig.set\_size\_inches(18, 7)

# The 1st subplot is the silhouette plot

# The silhouette coefficient can range from -1, 1 but in this example all

# lie within [-0.1, 1]

ax1.set\_xlim([-0.1, 1])

# The (n\_clusters+1)\*10 is for inserting blank space between silhouette

# plots of individual clusters, to demarcate them clearly.

ax1.set\_ylim([0, len(X) + (n\_clusters + 1) \* 10])

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

# The silhouette\_score gives the average value for all the samples.

# This gives a perspective into the density and separation of the formed

# clusters

silhouette\_avg = silhouette\_score(X, cluster\_labels)

print("For n\_clusters =", n\_clusters,

"The average silhouette\_score is :", silhouette\_avg)

# Compute the silhouette scores for each sample

sample\_silhouette\_values = silhouette\_samples(X, cluster\_labels)

y\_lower = 10

for i in range(n\_clusters):

# Aggregate the silhouette scores for samples belonging to

# cluster i, and sort them

ith\_cluster\_silhouette\_values = \

sample\_silhouette\_values[cluster\_labels == i]

ith\_cluster\_silhouette\_values.sort()

size\_cluster\_i = ith\_cluster\_silhouette\_values.shape[0]

y\_upper = y\_lower + size\_cluster\_i

color = cm.nipy\_spectral(float(i) / n\_clusters)

ax1.fill\_betweenx(np.arange(y\_lower, y\_upper),

0, ith\_cluster\_silhouette\_values,

facecolor=color, edgecolor=color, alpha=0.7)

# Label the silhouette plots with their cluster numbers at the middle

ax1.text(-0.05, y\_lower + 0.5 \* size\_cluster\_i, str(i))

# Compute the new y\_lower for next plot

y\_lower = y\_upper + 10 # 10 for the 0 samples

ax1.set\_title("The silhouette plot for the various clusters.")

ax1.set\_xlabel("The silhouette coefficient values")

ax1.set\_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values

ax1.axvline(x=silhouette\_avg, color="red", linestyle="--")

ax1.set\_yticks([]) # Clear the yaxis labels / ticks

ax1.set\_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 2nd Plot showing the actual clusters formed

colors = cm.nipy\_spectral(cluster\_labels.astype(float) / n\_clusters)

ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,

c=colors, edgecolor='k')

# Labeling the clusters

centers = clusterer.cluster\_centers\_

# Draw white circles at cluster centers

ax2.scatter(centers[:, 0], centers[:, 1], marker='o',

c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):

ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,

s=50, edgecolor='k')

ax2.set\_title("The visualization of the clustered data.")

ax2.set\_xlabel("Feature space for the 1st feature")

ax2.set\_ylabel("Feature space for the 2nd feature")

plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "

"with n\_clusters = %d" % n\_clusters),

fontsize=14, fontweight='bold')

plt.show()

**קוד עבור SSE :**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Assuming 'df' is your DataFrame and you want to cluster based on its values

X = df.values

range\_n\_clusters = [2, 3, 7, 8]

sse\_values = []

for n\_clusters in range\_n\_clusters:

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

cluster\_centers = clusterer.cluster\_centers\_

sse = 0

for i in range(len(X)):

cluster\_center = cluster\_centers[cluster\_labels[i]]

sse += np.linalg.norm(X[i] - cluster\_center) \*\* 2

sse\_values.append(sse)

print("For n\_clusters =", n\_clusters, "The SSE is:", sse)

# Plotting SSE values

plt.plot(range\_n\_clusters, sse\_values, marker='o')

plt.title('Elbow Method For Optimal k')

plt.xlabel('Number of clusters')

plt.ylabel('SSE')

plt.show()

**קוד עבור בדיקת לK=4 ב silhouette:**

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_samples, silhouette\_score

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import numpy as np

import pandas as pd

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Generating the sample data from make\_blobs

# This particular setting has one distinct cluster and 3 clusters placed close

# together.

X, y = make\_blobs(n\_samples=3430,

n\_features=1545,

centers=4,

cluster\_std=1,

center\_box=(-10.0, 10.0),

shuffle=True,

random\_state=1) # For reproducibility

range\_n\_clusters = [4]

for n\_clusters in range\_n\_clusters:

# Create a subplot with 1 row and 2 columns

fig, (ax1, ax2) = plt.subplots(1, 2)

fig.set\_size\_inches(18, 7)

# The 1st subplot is the silhouette plot

# The silhouette coefficient can range from -1, 1 but in this example all

# lie within [-0.1, 1]

ax1.set\_xlim([-0.1, 1])

# The (n\_clusters+1)\*10 is for inserting blank space between silhouette

# plots of individual clusters, to demarcate them clearly.

ax1.set\_ylim([0, len(X) + (n\_clusters + 1) \* 10])

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

# The silhouette\_score gives the average value for all the samples.

# This gives a perspective into the density and separation of the formed

# clusters

silhouette\_avg = silhouette\_score(X, cluster\_labels)

print("For n\_clusters =", n\_clusters,

"The average silhouette\_score is :", silhouette\_avg)

# Compute the silhouette scores for each sample

sample\_silhouette\_values = silhouette\_samples(X, cluster\_labels)

y\_lower = 10

for i in range(n\_clusters):

# Aggregate the silhouette scores for samples belonging to

# cluster i, and sort them

ith\_cluster\_silhouette\_values = \

sample\_silhouette\_values[cluster\_labels == i]

ith\_cluster\_silhouette\_values.sort()

size\_cluster\_i = ith\_cluster\_silhouette\_values.shape[0]

y\_upper = y\_lower + size\_cluster\_i

color = cm.nipy\_spectral(float(i) / n\_clusters)

ax1.fill\_betweenx(np.arange(y\_lower, y\_upper),

0, ith\_cluster\_silhouette\_values,

facecolor=color, edgecolor=color, alpha=0.7)

# Label the silhouette plots with their cluster numbers at the middle

ax1.text(-0.05, y\_lower + 0.5 \* size\_cluster\_i, str(i))

# Compute the new y\_lower for next plot

y\_lower = y\_upper + 10 # 10 for the 0 samples

ax1.set\_title("The silhouette plot for the various clusters.")

ax1.set\_xlabel("The silhouette coefficient values")

ax1.set\_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values

ax1.axvline(x=silhouette\_avg, color="red", linestyle="--")

ax1.set\_yticks([]) # Clear the yaxis labels / ticks

ax1.set\_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 2nd Plot showing the actual clusters formed

colors = cm.nipy\_spectral(cluster\_labels.astype(float) / n\_clusters)

ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,

c=colors, edgecolor='k')

# Labeling the clusters

centers = clusterer.cluster\_centers\_

# Draw white circles at cluster centers

ax2.scatter(centers[:, 0], centers[:, 1], marker='o',

c="white", alpha=1, s=200, edgecolor='k')

for i, c in enumerate(centers):

ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,

s=50, edgecolor='k')

ax2.set\_title("The visualization of the clustered data.")

ax2.set\_xlabel("Feature space for the 1st feature")

ax2.set\_ylabel("Feature space for the 2nd feature")

plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "

"with n\_clusters = %d" % n\_clusters),

fontsize=14, fontweight='bold')

plt.show()

**קוד עבור בדיקת לK=4 ב SSE:**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Assuming 'df' is your DataFrame and you want to cluster based on its values

X = df.values

range\_n\_clusters = [2, 3, 4, 5, 6, 7, 8]

sse\_values = []

for n\_clusters in range\_n\_clusters:

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

cluster\_centers = clusterer.cluster\_centers\_

sse = 0

for i in range(len(X)):

cluster\_center = cluster\_centers[cluster\_labels[i]]

sse += np.linalg.norm(X[i] - cluster\_center) \*\* 2

sse\_values.append(sse)

print("For n\_clusters =", n\_clusters, "The SSE is:", sse)

# Plotting SSE values

plt.plot(range\_n\_clusters, sse\_values, marker='o')

plt.title('Elbow Method For Optimal k')

plt.xlabel('Number of clusters')

plt.ylabel('SSE')

plt.show()

**קוד עבור בדיקת איזה אשכול עם מספק דגימות מינימלי ואיזה עם מקסימלי:**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import numpy as np

import pandas as pd

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Assuming 'df' is your DataFrame and you want to cluster based on its values

X = df.values

range\_n\_clusters = [4]

for n\_clusters in range\_n\_clusters:

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

cluster\_centers = clusterer.cluster\_centers\_

# Initialize variables to keep track of cluster counts

min\_examples\_cluster = -1

min\_examples\_count = float('inf')

max\_examples\_cluster = -1

max\_examples\_count = -1

for i in range(n\_clusters):

cluster\_size = np.sum(cluster\_labels == i)

if cluster\_size < min\_examples\_count:

min\_examples\_count = cluster\_size

min\_examples\_cluster = i

if cluster\_size > max\_examples\_count:

max\_examples\_count = cluster\_size

max\_examples\_cluster = i

print("For n\_clusters =", n\_clusters)

print("Cluster", min\_examples\_cluster, "has the minimum number of examples:", min\_examples\_count)

print("Cluster", max\_examples\_cluster, "has the maximum number of examples:", max\_examples\_count)

**קוד עבור צרו קובץ CSV המכיל את הנתונים השייכים לכל אשכול בנפרד (ביקשתי בקוד עבור תיקייה ספציפית שנמצאת אצלי):**

from sklearn.cluster import KMeans

import numpy as np

import pandas as pd

import os

# Read data from the file

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

X = df.values

# Number of clusters to check

range\_n\_clusters = [4]

# Create a new directory if it doesn't exist

output\_path = 'C:/Users/ronyh/Desktop/pythonProject4'

if not os.path.exists(output\_path):

os.makedirs(output\_path)

for n\_clusters in range\_n\_clusters:

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)

cluster\_labels = clusterer.fit\_predict(X)

# Create a DataFrame for each cluster

cluster\_data = pd.DataFrame(X, columns=df.columns)

cluster\_data['Cluster'] = cluster\_labels

# Iterate over each cluster

for cluster\_num in range(n\_clusters):

# Filter data points belonging to the current cluster

cluster\_subset = cluster\_data[cluster\_data['Cluster'] == cluster\_num]

# Drop the 'Cluster' column before saving to CSV

cluster\_subset.drop(columns=['Cluster'], inplace=True)

# Save the data points belonging to the current cluster to a CSV file

cluster\_subset.to\_csv(f'{output\_path}/cluster\_{cluster\_num}\_data.csv', index=False)

print("Data for each cluster saved to CSV files.")

**קוד עבור סיווג על פי ארבעת ערכי הlinkage בספרייה:**

import pandas as pd

from scipy.cluster.hierarchy import dendrogram, linkage

import matplotlib.pyplot as plt

# Read the data from the file

df = pd.read\_excel('C://Users//ronyh//Downloads//dailykos.xlsx')

# Convert the data to a numerical array

X = df.values

# Linkage methods to test

linkage\_methods = ['single', 'complete', 'average', 'ward']

# Plot dendrogram for each linkage method

plt.figure(figsize=(12, 6))

for i, method in enumerate(linkage\_methods, 1):

plt.subplot(2, 2, i)

Z = linkage(X, method=method)

dendrogram(Z)

plt.title(f'Dendrogram - {method.capitalize()} linkage')

plt.xlabel('Samples')

plt.ylabel('Distance')

plt.tight\_layout()

plt.show()