private note @115 2 views

HW5_2017310936_Md_Shirajum_Munir

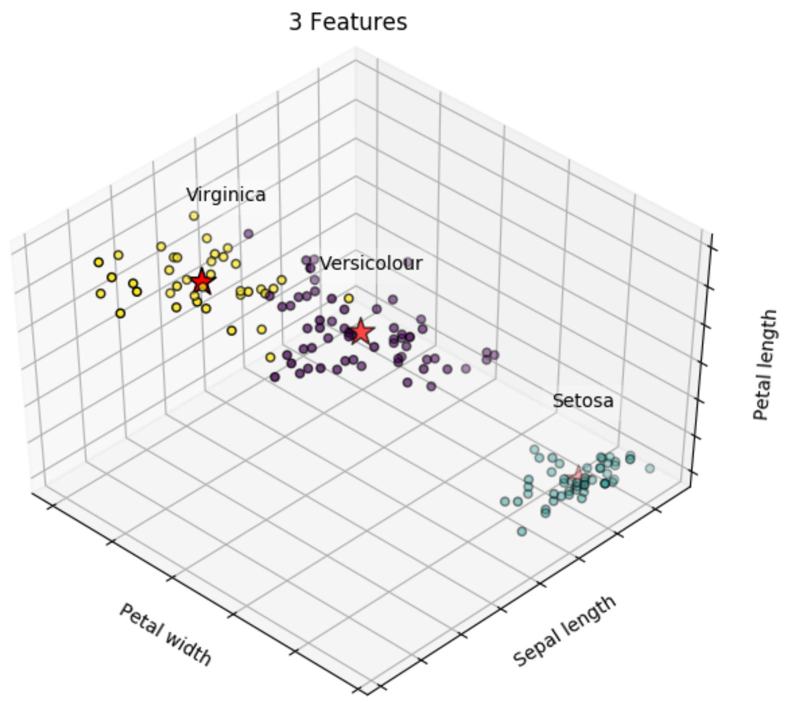
I) Use K-means with following parameters:

n_clusters=3,
init='k-means++',
n_init=10,
max_iter=300,
tol=1e-04,
random_state=0,

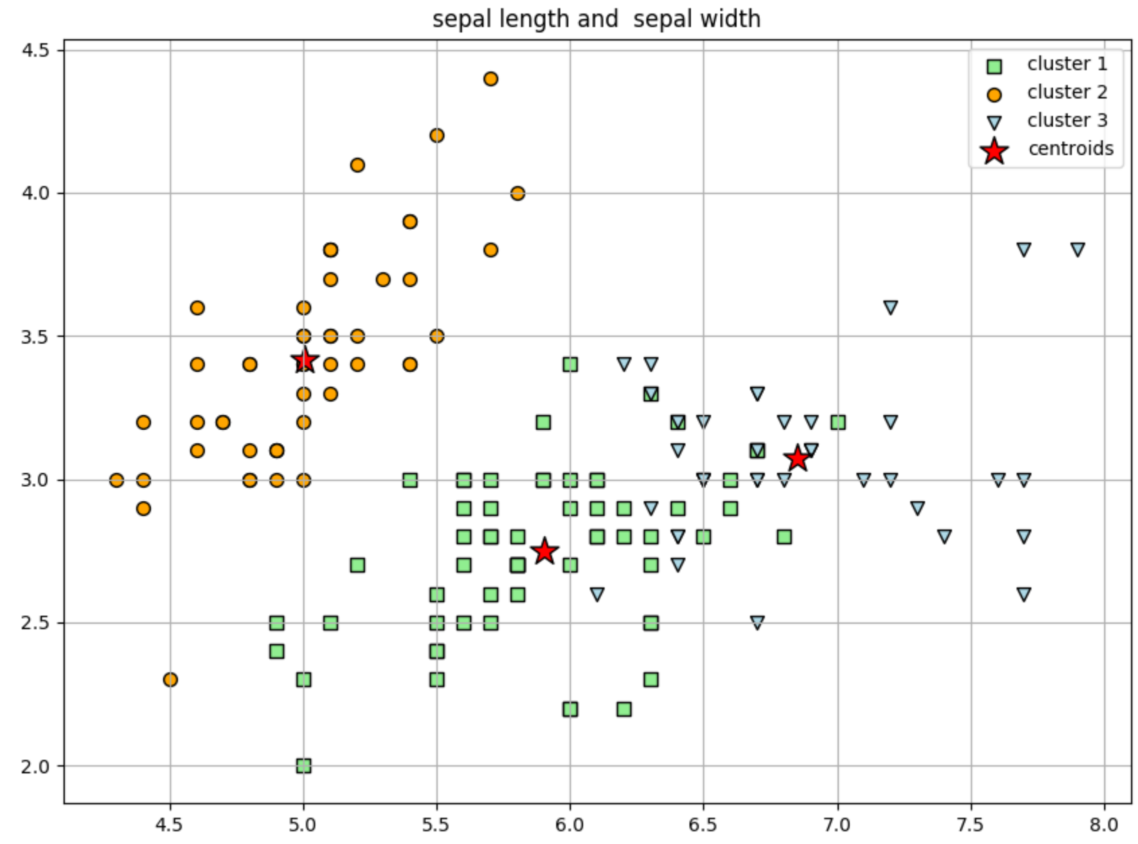
a) Plot the clusters in 3-D plot with 3 features and 2-D plot with 2 features (similar to input [6] of this notebook, you can choose any combination that shows clear clusters)

All features:

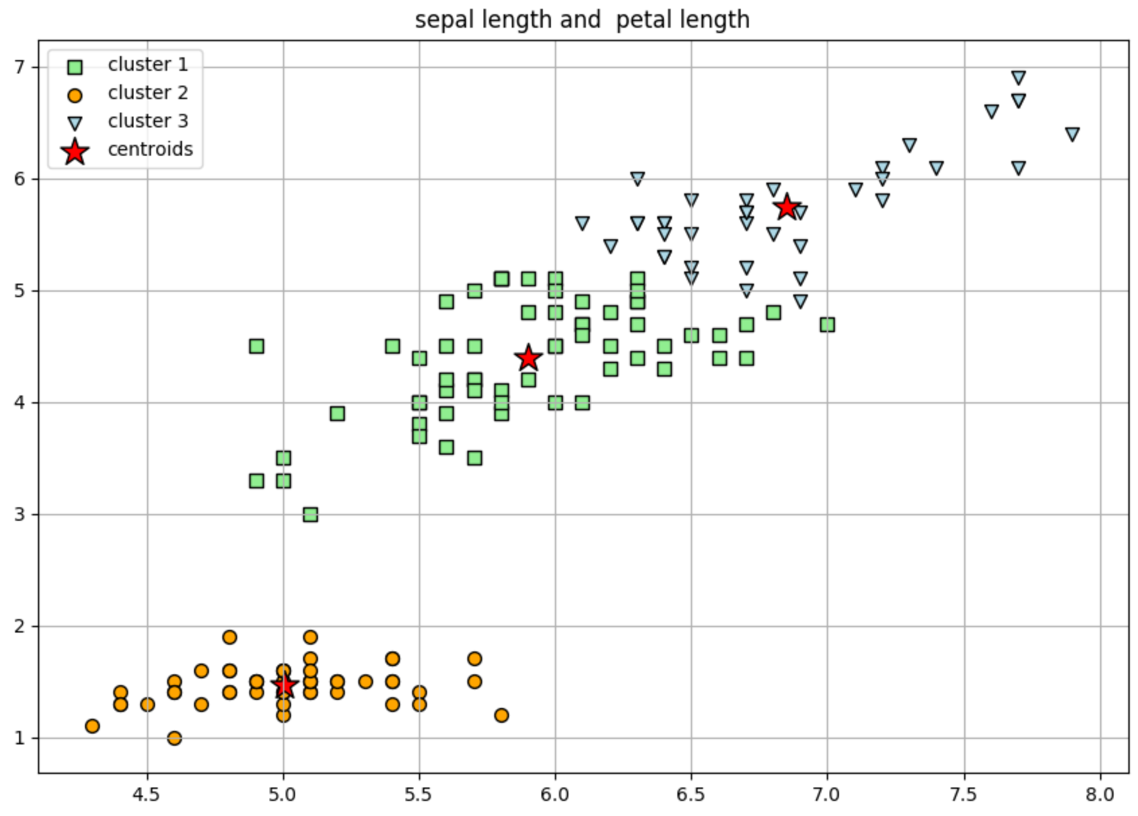
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm



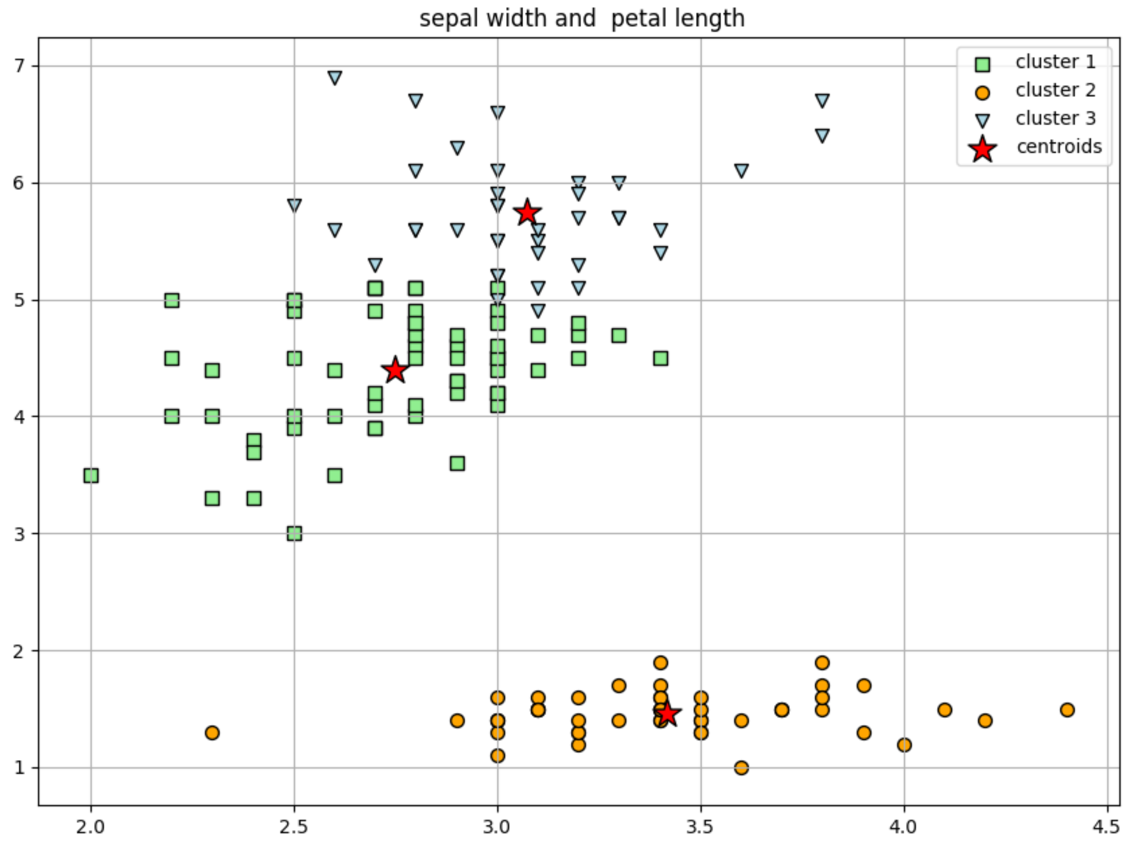
Features : sepal length and sepal width



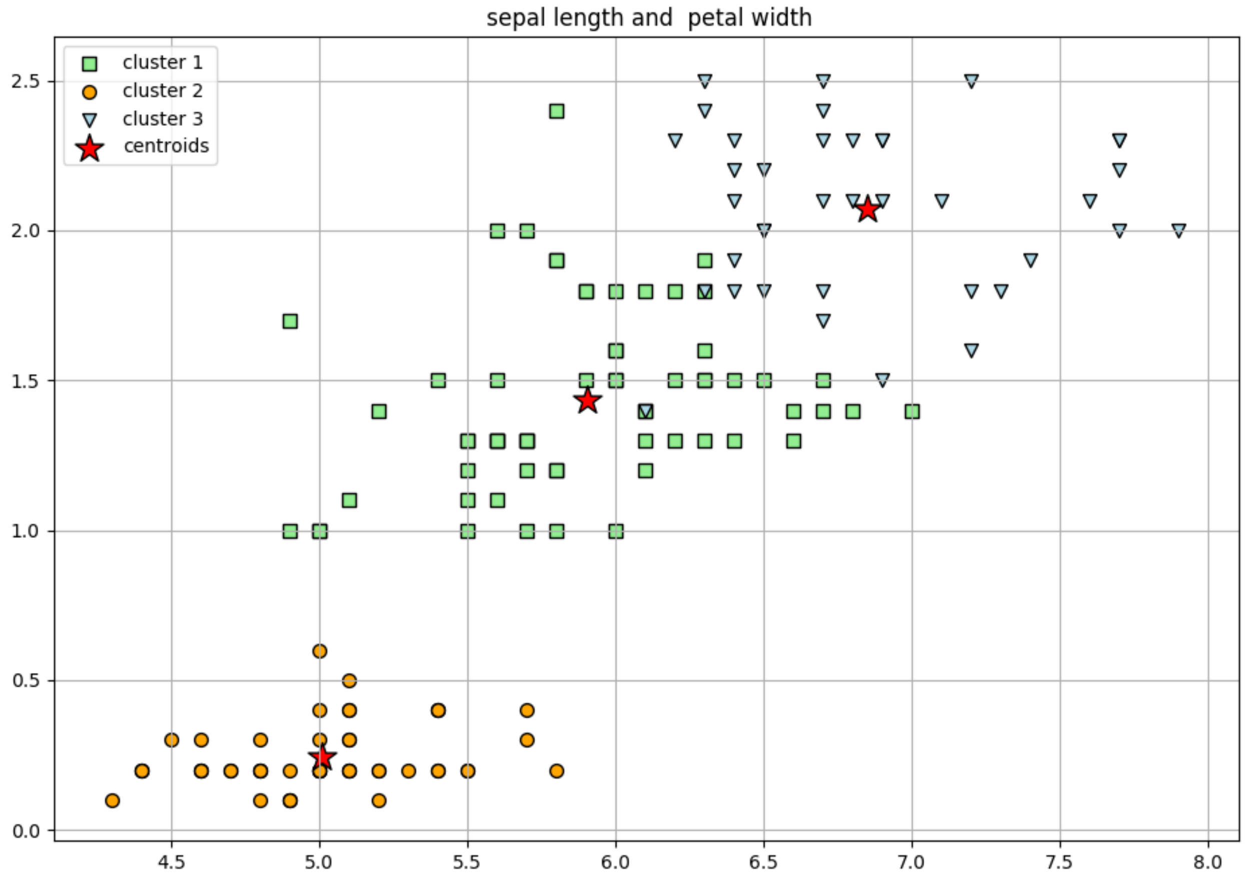
Features : sepal length and petal length



Features : sepal width and petal length



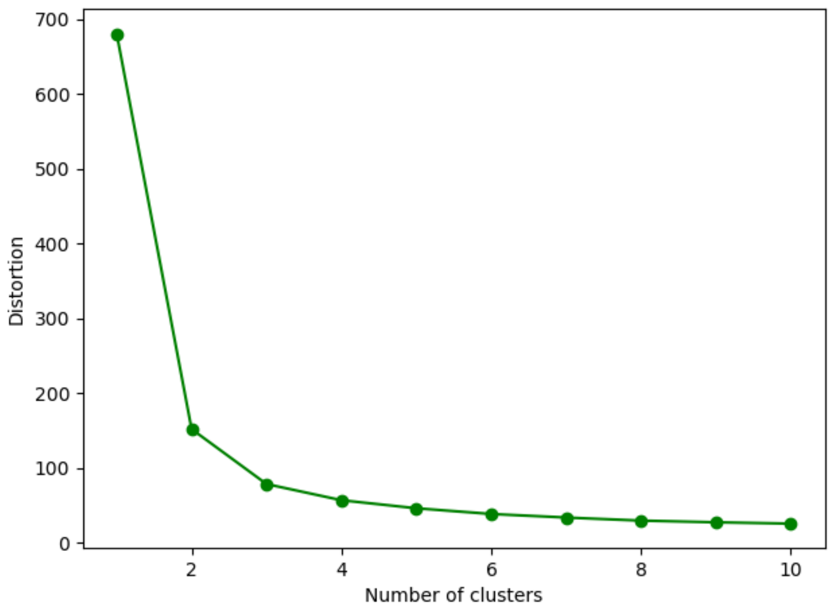
Features : sepal length and petal width



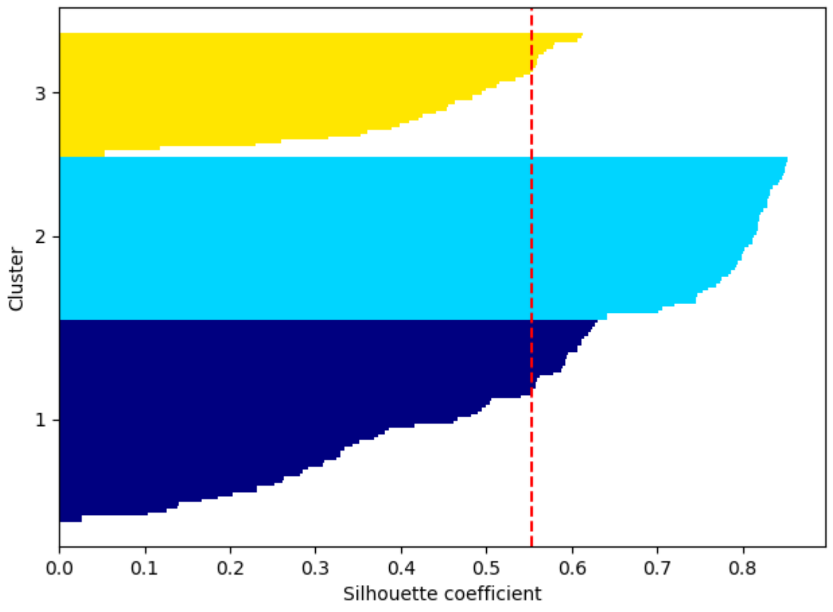
```
import numpy as np
import matplotlib.pyplot as plt
```

```python
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn import datasets

np.random.seed(5)

iris = datasets.load_iris()
X = iris.data
y = iris.target
print(len(X))

km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)

y_km = km.fit_predict(X)

titles = ['3 clusters']
fig = plt.figure( figsize=(8, 6))
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
labels = km.labels_

for name, label in [('Setosa', 0),
                    ('Versicolour', 1),
                    ('Virginica', 2)]:
    ax.text3D(X[y == label, 3].mean(),
              X[y == label, 0].mean(),
              X[y == label, 2].mean() + 2, name,
              horizontalalignment='center',
              bbox=dict(alpha=.2, edgecolor='w', facecolor='w'))

ax.scatter(X[:, 3], X[:, 0], X[:, 2],
           c=labels.astype(np.float), edgecolor='k')
ax.scatter(km.cluster_centers_[:, 3],
            km.cluster_centers_[:, 0],
            km.cluster_centers_[:, 2],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Petal width')
ax.set_ylabel('Sepal length')
ax.set_zlabel('Petal length')
ax.set_title('3 Features')
ax.dist = 12
plt.show()

plt.scatter(X[y_km == 0, 0],
            X[y_km == 0, 1],
            s=50, c='lightgreen',
            marker='s', edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_km == 1, 0],
            X[y_km == 1, 1],
            s=50, c='orange',
            marker='o', edgecolor='black',
            label='cluster 2')
plt.scatter(X[y_km == 2, 0],
            X[y_km == 2, 1],
            s=50, c='lightblue',
            marker='v', edgecolor='black',
            label='cluster 3')
plt.scatter(km.cluster_centers_[:, 0],
            km.cluster_centers_[:, 1],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
plt.title('sepal length and  sepal width')
#plt.savefig('images/11_02.png', dpi=300)
plt.show()

plt.scatter(X[y_km == 0, 0],
            X[y_km == 0, 2],
            s=50, c='lightgreen',
            marker='s', edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_km == 1, 0],
            X[y_km == 1, 2],
            s=50, c='orange',
            marker='o', edgecolor='black',
            label='cluster 2')
plt.scatter(X[y_km == 2, 0],
            X[y_km == 2, 2],
            s=50, c='lightblue',
            marker='v', edgecolor='black',
            label='cluster 3')
plt.scatter(km.cluster_centers_[:, 0],
            km.cluster_centers_[:, 2],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('images/11_02.png', dpi=300)
plt.title('sepal length and  petal length')
plt.show()


plt.scatter(X[y_km == 0, 1],
            X[y_km == 0, 2],
            s=50, c='lightgreen',
            marker='s', edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_km == 1, 1],
            X[y_km == 1, 2],
            s=50, c='orange',
            marker='o', edgecolor='black',
            label='cluster 2')
plt.scatter(X[y_km == 2, 1],
            X[y_km == 2, 2],
            s=50, c='lightblue',
            marker='v', edgecolor='black',
            label='cluster 3')
plt.scatter(km.cluster_centers_[:, 1],
            km.cluster_centers_[:, 2],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('images/11_02.png', dpi=300)
plt.title(' sepal width and  petal length')
plt.show()


plt.scatter(X[y_km == 0, 0],
            X[y_km == 0, 3],
            s=50, c='lightgreen',
            marker='s', edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_km == 1, 0],
            X[y_km == 1, 3],
            s=50, c='orange',
            marker='o', edgecolor='black',
            label='cluster 2')
plt.scatter(X[y_km == 2, 0],
            X[y_km == 2, 3],
            s=50, c='lightblue',
            marker='v', edgecolor='black',
            label='cluster 3')
plt.scatter(km.cluster_centers_[:, 0],
            km.cluster_centers_[:, 3],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')
plt.legend(scatterpoints=1)
plt.grid()
plt.tight_layout()
#plt.savefig('images/11_02.png', dpi=300)
plt.title(' sepal length and  petal width')
plt.show()
```

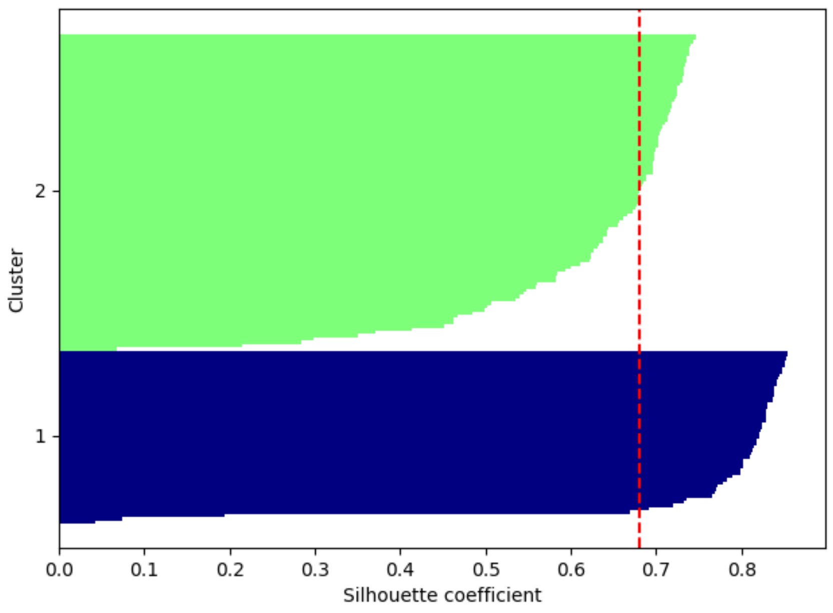
b) Use elbow method to plot and choose the best "k", similar to inputs [7] and [8] of this notebook.



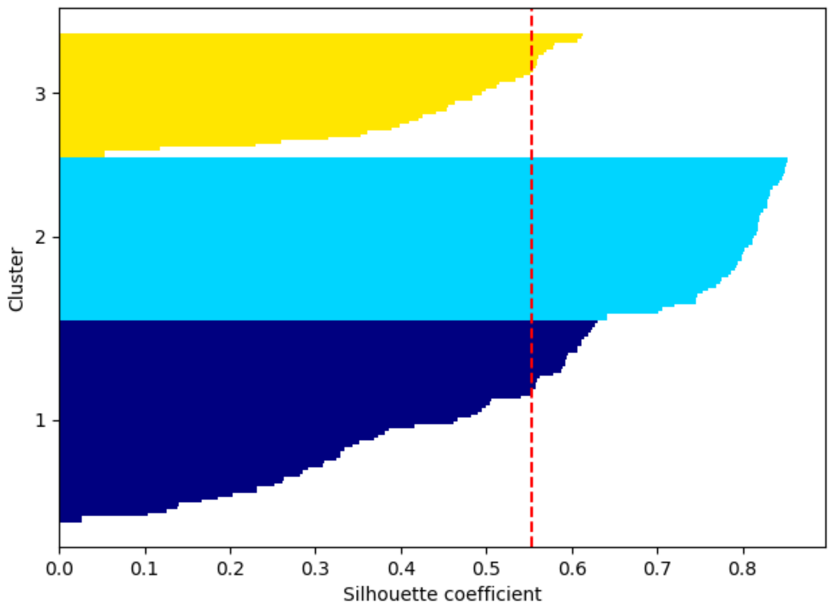
c) Show the silhouette plots of these clusters similar to input [9] of this notebook.



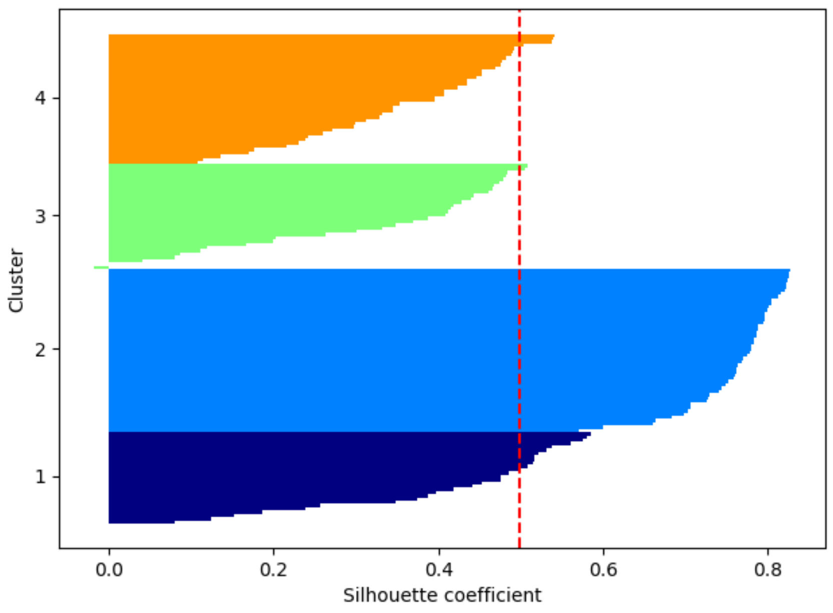
d) Show the silhouette plots of K-means with *n_clusters=2, 3, and 4.*
n_clusters=2



n_clusters=3



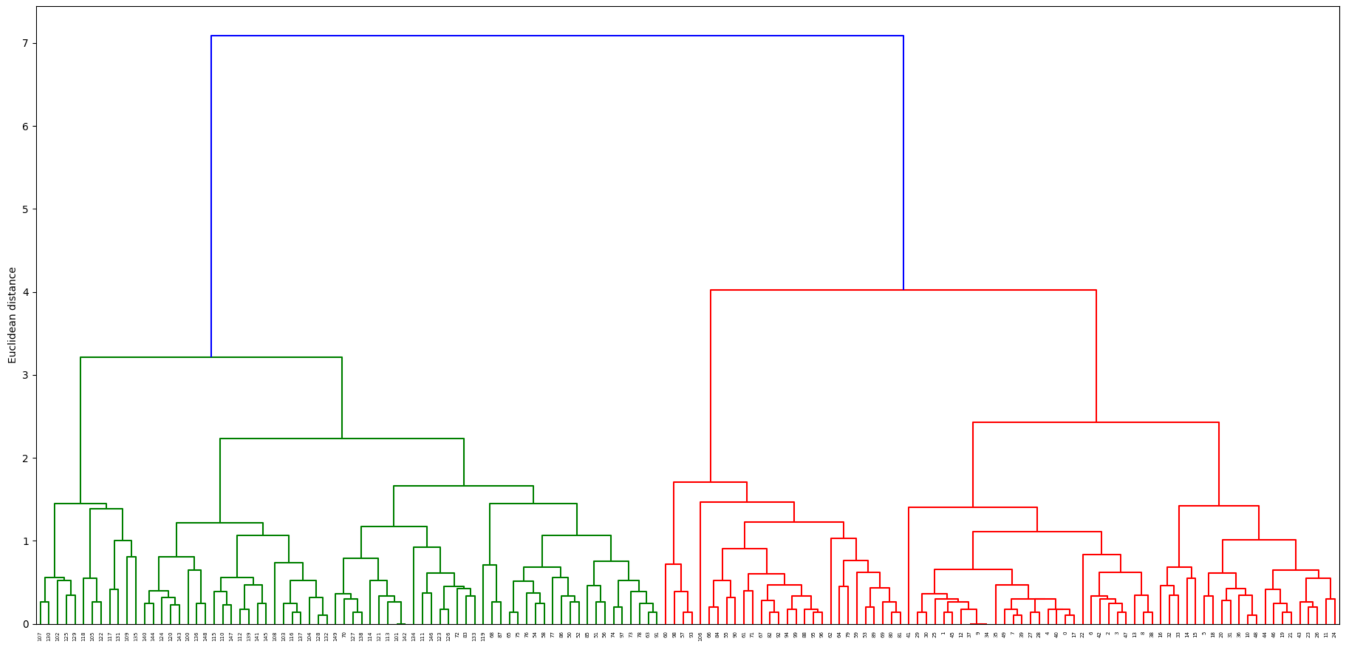
n_clusters=4



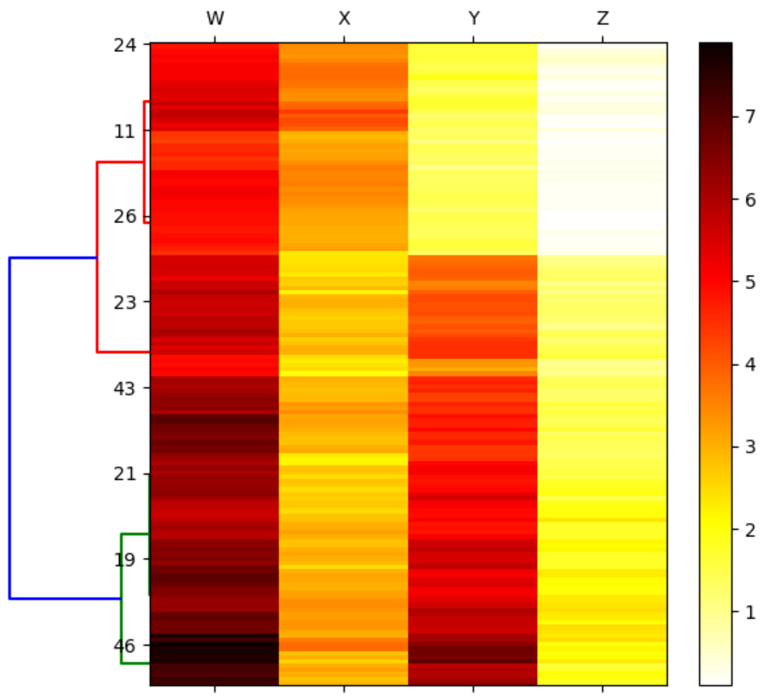
```
print('Distortion: %.2f' % km.inertia_)
distortions = []
for i in range(1, 11):
    km = KMeans(n_clusters=i,
                init='k-means++',
                n_init=10,
                max_iter=300,
                random_state=0)
    km.fit(X)
    distortions.append(km.inertia_)
plt.plot(range(1, 11), distortions, marker='o', c='g')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.tight_layout()
plt.show()


# c) Show the silhouette plots of these clusters similar to input  [9] of this  notebook.

from sklearn.metrics import silhouette_samples
from matplotlib import cm

km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(X)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')
plt.tight_layout()
plt.show()


# d) Show the silhouette plots of K-means with n_clusters=2, 3, and 4.
```

```
from sklearn.metrics import silhouette_samples
from matplotlib import cm

km = KMeans(n_clusters=2,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(X)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')
plt.tight_layout()
plt.show()


from sklearn.metrics import silhouette_samples
from matplotlib import cm

km = KMeans(n_clusters=3,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(X)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")

plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')
plt.tight_layout()
plt.show()


from sklearn.metrics import silhouette_samples
from matplotlib import cm

km = KMeans(n_clusters=4,
            init='k-means++',
            n_init=10,
            max_iter=300,
            tol=1e-04,
            random_state=0)
y_km = km.fit_predict(X)

cluster_labels = np.unique(y_km)
n_clusters = cluster_labels.shape[0]
silhouette_vals = silhouette_samples(X, y_km, metric='euclidean')
y_ax_lower, y_ax_upper = 0, 0
yticks = []
for i, c in enumerate(cluster_labels):
    c_silhouette_vals = silhouette_vals[y_km == c]
    c_silhouette_vals.sort()
    y_ax_upper += len(c_silhouette_vals)
    color = cm.jet(float(i) / n_clusters)
    plt.barh(range(y_ax_lower, y_ax_upper), c_silhouette_vals, height=1.0,
             edgecolor='none', color=color)

    yticks.append((y_ax_lower + y_ax_upper) / 2.)
    y_ax_lower += len(c_silhouette_vals)

silhouette_avg = np.mean(silhouette_vals)
plt.axvline(silhouette_avg, color="red", linestyle="--")
plt.yticks(yticks, cluster_labels + 1)
plt.ylabel('Cluster')
plt.xlabel('Silhouette coefficient')
plt.tight_layout()
plt.show()
```

II) Performing the hierarchical clustering on a distance matrix, similar to inputs [14] to [19] of this notebook.



Attaching dendrograms to a heat map



```
Output:
150
       W    X    Y    Z
0    5.1  3.5  1.4  0.2
1    4.9  3.0  1.4  0.2
2    4.7  3.2  1.3  0.2
3    4.6  3.1  1.5  0.2
4    5.0  3.6  1.4  0.2
5    5.4  3.9  1.7  0.4
6    4.6  3.4  1.4  0.3
7    5.0  3.4  1.5  0.2
8    4.4  2.9  1.4  0.2
9    4.9  3.1  1.5  0.1
10   5.4  3.7  1.5  0.2
11   4.8  3.4  1.6  0.2
12   4.8  3.0  1.4  0.1
13   4.3  3.0  1.1  0.1
14   5.8  4.0  1.2  0.2
15   5.7  4.4  1.5  0.4
16   5.4  3.9  1.3  0.4
17   5.1  3.5  1.4  0.3
18   5.7  3.8  1.7  0.3
19   5.1  3.8  1.5  0.3
20   5.4  3.4  1.7  0.2
21   5.1  3.7  1.5  0.4
22   4.6  3.6  1.0  0.2
23   5.1  3.3  1.7  0.5
24   4.8  3.4  1.9  0.2
25   5.0  3.0  1.6  0.2
26   5.0  3.4  1.6  0.4
27   5.2  3.5  1.5  0.2
28   5.2  3.4  1.4  0.2
29   4.7  3.2  1.6  0.2
..   ...  ...  ...  ...
120  6.9  3.2  5.7  2.3
121  5.6  2.8  4.9  2.0
122  7.7  2.8  6.7  2.0
123  6.3  2.7  4.9  1.8
124  6.7  3.3  5.7  2.1
125  7.2  3.2  6.0  1.8
126  6.2  2.8  4.8  1.8
127  6.1  3.0  4.9  1.8
128  6.4  2.8  5.6  2.1
129  7.2  3.0  5.8  1.6
130  7.4  2.8  6.1  1.9
131  7.9  3.8  6.4  2.0
132  6.4  2.8  5.6  2.2
133  6.3  2.8  5.1  1.5
134  6.1  2.6  5.6  1.4
135  7.7  3.0  6.1  2.3
136  6.3  3.4  5.6  2.4
137  6.4  3.1  5.5  1.8
138  6.0  3.0  4.8  1.8
139  6.9  3.1  5.4  2.1
```

```
140  6.7  3.1  5.6  2.4
141  6.9  3.1  5.1  2.3
142  5.8  2.7  5.1  1.9
143  6.8  3.2  5.9  2.3
144  6.7  3.3  5.7  2.5
145  6.7  3.0  5.2  2.3
146  6.3  2.5  5.0  1.9
147  6.5  3.0  5.2  2.0
148  6.2  3.4  5.4  2.3
149  5.9  3.0  5.1  1.8
[150 rows x 4 columns]
          0         1         2         3         4         5         6    \
0    0.000000  0.538516  0.509902  0.648074  0.141421  0.616441  0.519615
1    0.538516  0.000000  0.300000  0.331662  0.608276  1.090871  0.509902
2    0.509902  0.300000  0.000000  0.244949  0.509902  1.086278  0.264575
3    0.648074  0.331662  0.244949  0.000000  0.648074  1.166190  0.331662
4    0.141421  0.608276  0.509902  0.648074  0.000000  0.616441  0.458258
5    0.616441  1.090871  1.086278  1.166190  0.616441  0.000000  0.994987
6    0.519615  0.509902  0.264575  0.331662  0.458258  0.994987  0.000000
7    0.173205  0.424264  0.412311  0.500000  0.223607  0.700000  0.424264
8    0.921954  0.509902  0.435890  0.300000  0.921954  1.459452  0.547723
9    0.469042  0.173205  0.316228  0.316228  0.529150  1.009950  0.479583
10   0.374166  0.866025  0.883176  1.000000  0.424264  0.346410  0.866025
11   0.374166  0.458258  0.374166  0.374166  0.346410  0.812404  0.300000
12   0.591608  0.141421  0.264575  0.264575  0.640312  1.161895  0.489898
13   0.994987  0.678233  0.500000  0.519615  0.974679  1.571623  0.616441
14   0.883176  1.360147  1.363818  1.529706  0.916515  0.678233  1.360147
15   1.104536  1.627882  1.587451  1.714643  1.086278  0.616441  1.493318
16   0.547723  1.053565  1.009950  1.166190  0.547723  0.400000  0.953939
17   0.100000  0.547723  0.519615  0.655744  0.173205  0.591608  0.509902
18   0.741620  1.174734  1.236932  1.322876  0.793725  0.331662  1.208305
19   0.331662  0.836660  0.754983  0.866025  0.264575  0.387298  0.648074
20   0.435890  0.707107  0.830662  0.877496  0.538516  0.538516  0.860233
21   0.300000  0.761577  0.700000  0.806226  0.264575  0.412311  0.600000
22   0.648074  0.781025  0.509902  0.707107  0.565685  1.122497  0.458258
23   0.469042  0.556776  0.648074  0.648074  0.529150  0.678233  0.624500
24   0.591608  0.648074  0.640312  0.538516  0.574456  0.830662  0.547723
25   0.547723  0.223607  0.469042  0.424264  0.632456  1.009950  0.608276
26   0.316228  0.500000  0.509902  0.547723  0.346410  0.648074  0.458258
27   0.141421  0.591608  0.616441  0.721110  0.244949  0.529150  0.624500
28   0.141421  0.500000  0.547723  0.678233  0.282843  0.648074  0.608276
29   0.538516  0.346410  0.300000  0.173205  0.538516  1.014889  0.316228
..        ...       ...       ...       ...       ...       ...       ...
120  5.121523  5.190376  5.348832  5.229723  5.164301  4.727579  5.274467
121  4.028647  4.002499  4.143670  3.986226  4.060788  3.748333  4.062019
122  6.211280  6.261789  6.446705  6.322974  6.265780  5.836095  6.399219
123  4.109745  4.106093  4.281355  4.143670  4.160529  3.801316  4.228475
124  4.969909  5.042817  5.194228  5.069517  5.007994  4.576024  5.113707
125  5.312250  5.389805  5.558777  5.438750  5.359104  4.917316  5.496362
126  3.977436  3.981206  4.149699  4.012481  4.024922  3.663332  4.090232
127  4.007493  4.031129  4.185690  4.047221  4.047221  3.674235  4.112177
128  4.840455  4.851804  5.014978  4.873397  4.883646  4.506662  4.947727
129  5.097058  5.158488  5.338539  5.217279  5.149757  4.722288  5.288667
130  5.546170  5.591959  5.777543  5.655086  5.601785  5.178803  5.731492
131  6.014150  6.154673  6.312686  6.215304  6.057227  5.559676  6.240192
132  4.880574  4.891830  5.053712  4.913247  4.923413  4.545327  4.984977
133  4.160529  4.168933  4.341659  4.198809  4.208325  3.845777  4.287190
134  4.570558  4.547527  4.716991  4.555217  4.614109  4.288356  4.662617
135  5.788782  5.860034  6.040695  5.932116  5.843800  5.391660  5.988322
136  4.891830  4.959839  5.092151  4.962862  4.920366  4.502222  4.993996
137  4.606517  4.650806  4.806246  4.669047  4.645428  4.247352  4.731807
138  3.896152  3.915354  4.066940  3.926831  3.934463  3.569314  3.991240
139  4.796874  4.860041  5.026927  4.910193  4.844585  4.412482  4.961854
140  5.019960  5.072475  5.228767  5.104900  5.061620  4.641121  5.152669
141  4.636809  4.702127  4.868265  4.760252  4.686150  4.249706  4.803124
142  4.208325  4.180909  4.334743  4.177320  4.246175  3.925557  4.263801
143  5.257376  5.320714  5.475400  5.349766  5.297169  4.868265  5.397222
144  5.136146  5.206726  5.353504  5.232590  5.173007  4.739198  5.267827
145  4.654031  4.700000  4.864155  4.745524  4.701064  4.284857  4.796874
146  4.276681  4.249706  4.430576  4.288356  4.330127  3.988734  4.384062
147  4.459821  4.498889  4.661545  4.533211  4.504442  4.102438  4.593474
148  4.650806  4.718050  4.848711  4.719110  4.678675  4.264974  4.749737
149  4.140048  4.153312  4.298837  4.149699  4.173727  3.818377  4.217819

          7         8         9      ...         140       141       142  \
0    0.173205  0.921954  0.469042    ...    5.019960  4.636809  4.208325
1    0.424264  0.509902  0.173205    ...    5.072475  4.702127  4.180909
2    0.412311  0.435890  0.316228    ...    5.228767  4.868265  4.334743
3    0.500000  0.300000  0.316228    ...    5.104900  4.760252  4.177320
4    0.223607  0.921954  0.529150    ...    5.061620  4.686150  4.246175
5    0.700000  1.459452  1.009950    ...    4.641121  4.249706  3.925557
6    0.424264  0.547723  0.479583    ...    5.152669  4.803124  4.263801
7    0.000000  0.787401  0.331662    ...    4.962862  4.590207  4.120680
8    0.787401  0.000000  0.556776    ...    5.273519  4.938623  4.310452
9    0.331662  0.556776  0.000000    ...    5.033885  4.669047  4.143670
10   0.500000  1.284523  0.787401    ...    4.868265  4.469899  4.124318
11   0.223607  0.670820  0.346410    ...    4.953786  4.600000  4.077990
12   0.469042  0.424264  0.173205    ...    5.152669  4.790616  4.244997
13   0.905539  0.346410  0.728011    ...    5.595534  5.254522  4.645428
14   1.044031  1.791647  1.311488    ...    5.081338  4.651881  4.448595
15   1.236932  1.997498  1.555635    ...    4.847680  4.438468  4.255585
```

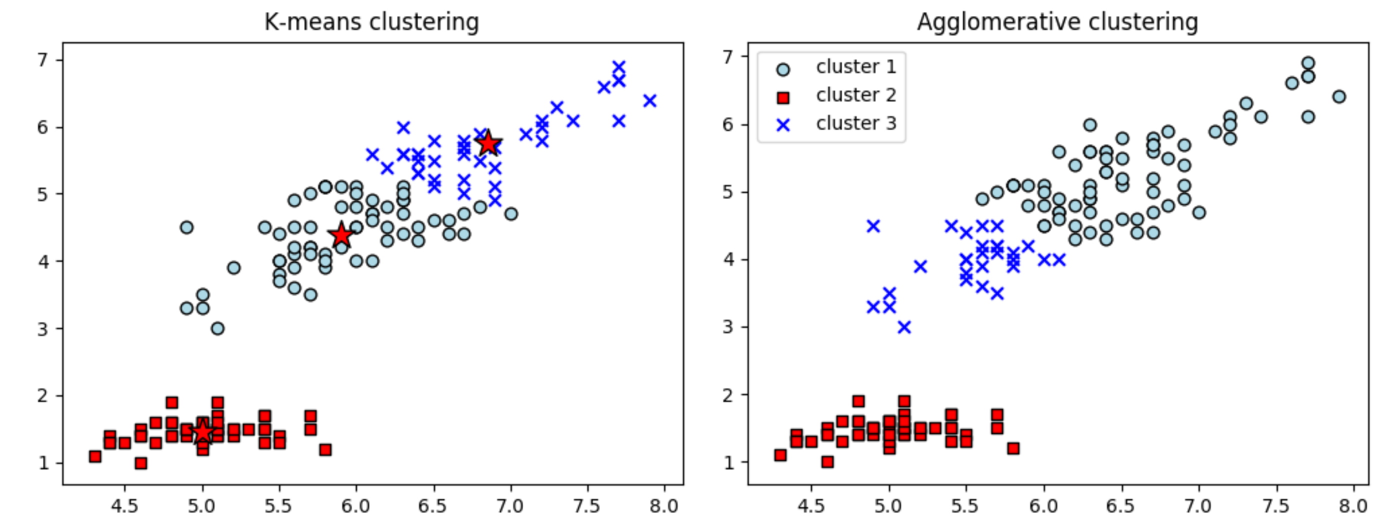
```
16   0.700000  1.431782  1.009950    ...     4.981967  4.576024  4.276681
17   0.200000  0.927362  0.500000    ...     4.976947  4.592385  4.168933
18   0.836660  1.612452  1.100000    ...     4.594562  4.182105  3.916631
19   0.424264  1.148913  0.754983    ...     4.926459  4.548626  4.149699
20   0.447214  1.157584  0.624500    ...     4.672259  4.279019  3.885872
21   0.374166  1.086278  0.700000    ...     4.871345  4.491102  4.086563
22   0.670820  0.830662  0.774597    ...     5.537147  5.173007  4.685083
23   0.387298  0.911043  0.529150    ...     4.628175  4.252058  3.790778
24   0.447214  0.812404  0.519615    ...     4.714870  4.376071  3.823611
25   0.412311  0.640312  0.200000    ...     4.872371  4.503332  3.983717
26   0.223607  0.830662  0.447214    ...     4.793746  4.422669  3.953479
27   0.223607  1.004988  0.509902    ...     4.905099  4.518849  4.104875
28   0.223607  0.943398  0.447214    ...     4.981967  4.591296  4.174925
29   0.374166  0.469042  0.264575    ...     4.984977  4.637887  4.074310
..        ...       ...       ...    ...          ...       ...       ...
120  5.069517  5.407402  5.146844    ...     0.264575  0.608276  1.407125
121  3.939543  4.115823  3.968627    ...     1.396424  1.382027  0.316228
122  6.158734  6.498461  6.211280    ...     1.568439  1.838478  2.487971
123  4.037326  4.296510  4.070626    ...     1.081665  0.900000  0.547723
124  4.914265  5.248809  4.991994    ...     0.374166  0.692820  1.252996
125  5.262129  5.625833  5.332917    ...     0.883176  1.077033  1.740690
126  3.905125  4.167733  3.944617    ...     1.157584  0.959166  0.519615
127  3.935734  4.208325  3.987480    ...     1.104536  0.969536  0.479583
128  4.768648  5.025933  4.811445    ...     0.519615  0.793725  0.812404
129  5.044799  5.400926  5.102940    ...     0.969536  1.039230  1.621727
130  5.492722  5.830094  5.544367    ...     1.039230  1.224745  1.889444
131  5.984981  6.426508  6.091798    ...     1.652271  1.808314  2.705550
132  4.809366  5.064583  4.853864    ...     0.469042  0.774597  0.842615
133  4.086563  4.358899  4.119466    ...     1.144552  1.044031  0.648074
134  4.483302  4.696807  4.493328    ...     1.268858  1.396424  0.774597
135  5.746303  6.115554  5.818075    ...     1.126943  1.284523  2.204541
136  4.831149  5.132251  4.914265    ...     0.500000  0.842615  1.113553
137  4.539824  4.838388  4.597826    ...     0.678233  0.812404  0.830662
138  3.822303  4.085340  3.872983    ...     1.224745  1.077033  0.479583
139  4.745524  5.089204  4.817676    ...     0.412311  0.360555  1.224745
140  4.962862  5.273519  5.033885    ...     0.000000  0.547723  1.212436
141  4.590207  4.938623  4.669047    ...     0.547723  0.000000  1.236932
142  4.120680  4.310452  4.143670    ...     1.212436  1.236932  0.000000
143  5.200961  5.523586  5.274467    ...     0.346410  0.812404  1.431782
144  5.082322  5.406478  5.165269    ...     0.244949  0.692820  1.374773
145  4.598913  4.914265  4.666905    ...     0.424264  0.244949  1.034408
146  4.200000  4.429447  4.220190    ...     1.063015  0.943398  0.547723
147  4.397727  4.701064  4.457578    ...     0.608276  0.519615  0.774597
148  4.589118  4.888763  4.672259    ...     0.624500  0.818535  0.948683
149  4.060788  4.302325  4.106093    ...     1.122497  1.122497  0.331662

          143       144       145       146       147       148       149
0    5.257376  5.136146  4.654031  4.276681  4.459821  4.650806  4.140048
1    5.320714  5.206726  4.700000  4.249706  4.498889  4.718050  4.153312
2    5.475400  5.353504  4.864155  4.430576  4.661545  4.848711  4.298837
3    5.349766  5.232590  4.745524  4.288356  4.533211  4.719110  4.149699
4    5.297169  5.173007  4.701064  4.330127  4.504442  4.678675  4.173727
5    4.868265  4.739198  4.284857  3.988734  4.102438  4.264974  3.818377
6    5.397222  5.267827  4.796874  4.384062  4.593474  4.749737  4.217819
7    5.200961  5.082322  4.598913  4.200000  4.397727  4.589118  4.060788
8    5.523586  5.406478  4.914265  4.429447  4.701064  4.888763  4.302325
9    5.274467  5.165269  4.666905  4.220190  4.457578  4.672259  4.106093
10   5.097058  4.977951  4.503332  4.170132  4.316248  4.511097  4.032369
11   5.190376  5.071489  4.597826  4.184495  4.387482  4.561798  4.022437
12   5.397222  5.286776  4.784349  4.324350  4.576024  4.791659  4.217819
13   5.845511  5.724509  5.235456  4.764452  5.025933  5.205766  4.631414
14   5.304715  5.180734  4.713809  4.453089  4.553021  4.750789  4.333590
15   5.061620  4.925444  4.501111  4.297674  4.341659  4.479955  4.113393
16   5.217279  5.081338  4.617359  4.325506  4.448595  4.616276  4.178516
17   5.218237  5.092151  4.609772  4.237924  4.420407  4.606517  4.102438
18   4.817676  4.700000  4.229657  3.937004  4.052160  4.254409  3.806573
19   5.158488  5.028916  4.570558  4.235564  4.379498  4.536518  4.060788
20   4.904080  4.794789  4.302325  3.924283  4.106093  4.328972  3.811824
21   5.109795  4.976947  4.511097  4.168933  4.324350  4.485532  4.006245
22   5.781003  5.646238  5.178803  4.794789  4.984977  5.135173  4.628175
23   4.876474  4.749737  4.259108  3.863936  4.068169  4.260282  3.738984
24   4.944694  4.832184  4.366921  3.944617  4.144876  4.315090  3.764306
25   5.116640  5.007994  4.501111  4.048456  4.295346  4.522168  3.952215
26   5.037857  4.911212  4.429447  4.038564  4.234383  4.414748  3.896152
27   5.140039  5.021952  4.538722  4.165333  4.343961  4.542026  4.032369
28   5.221111  5.102940  4.610857  4.227292  4.419276  4.627094  4.110961
29   5.225897  5.109795  4.627094  4.183300  4.413615  4.597826  4.036087
..        ...       ...       ...       ...       ...       ...       ...
120  0.223607  0.300000  0.574456  1.224745  0.734847  0.787401  1.284523
121  1.640122  1.532971  1.195826  0.774597  0.969536  1.029563  0.458258
122  1.303840  1.581139  1.838478  2.224860  1.931321  2.095233  2.424871
123  1.322876  1.284523  0.768115  0.244949  0.509902  1.000000  0.538516
124  0.316228  0.400000  0.616441  1.153256  0.624500  0.624500  1.086278
125  0.648074  0.916515  1.086278  1.519868  1.100000  1.284523  1.593738
126  1.407125  1.341641  0.836660  0.387298  0.574456  0.984886  0.469042
127  1.334166  1.256981  0.836660  0.556776  0.538516  0.818535  0.282843
128  0.670820  0.714143  0.574456  0.707107  0.469042  0.692820  0.793725
129  0.836660  1.077033  1.048809  1.337909  1.004988  1.345362  1.489966
130  0.848528  1.122497  1.224745  1.584298  1.292285  1.565248  1.816590
131  1.382027  1.558846  1.900000  2.493993  2.009975  2.034699  2.523886
132  0.648074  0.663325  0.547723  0.741620  0.500000  0.670820  0.836660
```

```
133  1.300000  1.330413  0.921954  0.509902  0.583095  1.048809  0.538516
134  1.322876  1.438749  1.220656  0.812404  0.916515  1.224745  0.781025
135  0.943398  1.135782  1.345362  1.892089  1.529706  1.702939  2.118962
136  0.624500  0.435890  0.700000  1.191638  0.721110  0.244949  0.964365
137  0.761577  0.812404  0.663325  0.793725  0.387298  0.624500  0.648074
138  1.462874  1.371131  0.948683  0.624500  0.670820  0.900000  0.316228
139  0.556776  0.574456  0.360555  0.959166  0.469042  0.787401  1.090871
140  0.346410  0.244949  0.424264  1.063015  0.608276  0.624500  1.122497
141  0.812404  0.692820  0.244949  0.943398  0.519615  0.818535  1.122497
142  1.431782  1.374773  1.034408  0.547723  0.774597  0.948683  0.331662
143  0.000000  0.316228  0.734847  1.307670  0.842615  0.806226  1.319091
144  0.316228  0.000000  0.616441  1.284523  0.793725  0.624500  1.256981
145  0.734847  0.616441  0.000000  0.781025  0.360555  0.670820  0.948683
146  1.307670  1.284523  0.781025  0.000000  0.583095  1.067708  0.655744
147  0.842615  0.793725  0.360555  0.583095  0.000000  0.616441  0.640312
148  0.806226  0.624500  0.670820  1.067708  0.616441  0.000000  0.768115
149  1.319091  1.256981  0.948683  0.655744  0.640312  0.768115  0.000000

[150 rows x 150 columns]
```

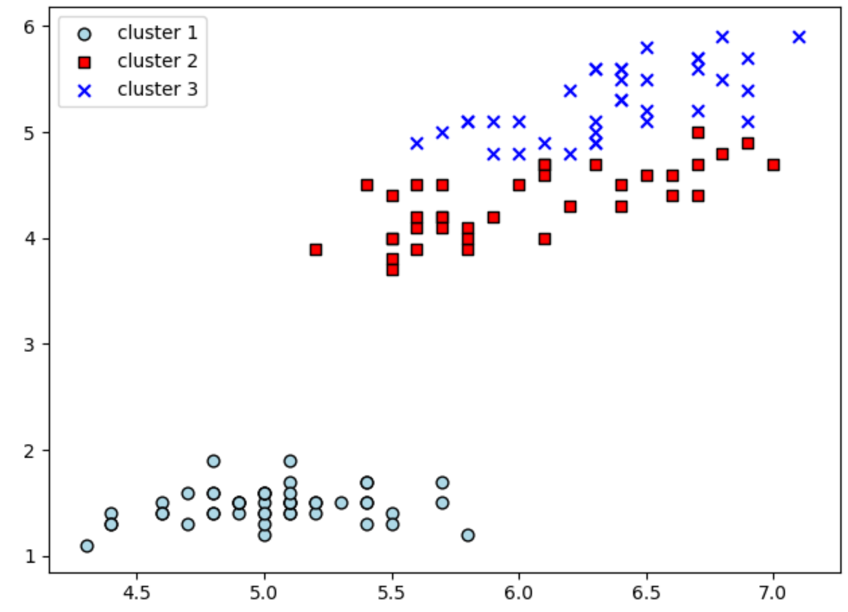
Code:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd

np.random.seed(5)
iris = datasets.load_iris()
X = iris.data
y = iris.target
print(len(X))
variables = ['W','X', 'Y', 'Z']
df = pd.DataFrame(X, columns=variables)
print(df)
from scipy.spatial.distance import pdist, squareform
row_dist = pd.DataFrame(squareform(pdist(df, metric='euclidean')), )
print(row_dist)

from scipy.cluster.hierarchy import linkage

row_clusters = linkage(row_dist, method='complete', metric='euclidean')
pd.DataFrame(row_clusters,
             columns=['row label 1', 'row label 2',
                      'distance', 'no. of items in clust.'],
             index=['cluster %d' % (i + 1)
                    for i in range(row_clusters.shape[0])])

print(row_clusters)

row_clusters = linkage(pdist(df, metric='euclidean'), method='complete')
pd.DataFrame(row_clusters,
             columns=['row label 1', 'row label 2',
                      'distance', 'no. of items in clust.'],
             index=['cluster %d' % (i + 1)
                    for i in range(row_clusters.shape[0])])

print(row_clusters)

from scipy.cluster.hierarchy import dendrogram
row_dendr = dendrogram(row_clusters, )
plt.tight_layout()
plt.ylabel('Euclidean distance')
plt.show()



# plot row dendrogram
fig = plt.figure(figsize=(8, 8), facecolor='white')
axd = fig.add_axes([0.09, 0.1, 0.2, 0.6])

row_dendr = dendrogram(row_clusters, orientation='left')
# reorder data with respect to clustering
df_rowclust = df.iloc[row_dendr['leaves'][::-1]]
axd.set_xticks([])
axd.set_yticks([])
# remove axes spines from dendrogram
for i in axd.spines.values():
    i.set_visible(False)
# plot heatmap
axm = fig.add_axes([0.23, 0.1, 0.6, 0.6])  # x-pos, y-pos, width, height
cax = axm.matshow(df_rowclust, interpolation='nearest', cmap='hot_r')

fig.colorbar(cax)
axm.set_xticklabels([''] + list(df_rowclust.columns))
axm.set_yticklabels([''] + list(df_rowclust.index))
axm.set_aspect('auto')
plt.show()
```

III) Performing the DBSCAN and compare it to K-means and agglomerative clustering, similar to inputs [24] to [25] of this notebook.



DBSCAN:



```
from sklearn.cluster import AgglomerativeClustering

f, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 3))

km = KMeans(n_clusters=3, random_state=0)
y_km = km.fit_predict(X)
ax1.scatter(X[y_km == 0, 0], X[y_km == 0, 2],
            edgecolor='black',
            c='lightblue', marker='o', s=40, label='cluster 1')
ax1.scatter(X[y_km == 1, 0], X[y_km == 1, 2],
            edgecolor='black',
            c='red', marker='s', s=40, label='cluster 2')
ax1.scatter(X[y_km == 2, 0], X[y_km == 2, 2],
            edgecolor='black',
            c='blue', marker='x', s=40, label='cluster 3')

ax1.scatter(km.cluster_centers_[:, 0],
            km.cluster_centers_[:, 2],
            s=250, marker='*',
            c='red', edgecolor='black',
            label='centroids')

ax1.set_title('K-means clustering')



ac = AgglomerativeClustering(n_clusters=3,
                             affinity='euclidean',
                             linkage='complete')
y_ac = ac.fit_predict(X)
ax2.scatter(X[y_ac == 0, 0], X[y_ac == 0, 2], c='lightblue',
            edgecolor='black',
            marker='o', s=40, label='cluster 1')
ax2.scatter(X[y_ac == 1, 0], X[y_ac == 1, 2], c='red',
            edgecolor='black',
            marker='s', s=40, label='cluster 2')
ax2.scatter(X[y_ac == 2, 0], X[y_ac == 2, 2], c='blue',
            edgecolor='black',
            marker='x', s=40, label='cluster 3')
# ax2.scatter(km.cluster_centers_[:, 0],
#             km.cluster_centers_[:, 2],
#             s=250, marker='*',
#             c='red', edgecolor='black',
#             label='centroids')
ax2.set_title('Agglomerative clustering')

plt.legend()
plt.tight_layout()
plt.show()



from sklearn.cluster import DBSCAN

db = DBSCAN(eps=0.8, min_samples=10, metric='euclidean')
y_db = db.fit_predict(X)
plt.scatter(X[y_db == 0, 0], X[y_db == 0, 2],
            c='lightblue', marker='o', s=40,
            edgecolor='black',
            label='cluster 1')
plt.scatter(X[y_db == 1, 0], X[y_db == 1, 2],
            c='red', marker='s', s=40,
            edgecolor='black',
            label='cluster 2')
plt.scatter(X[y_db == 2, 0], X[y_db == 2, 2],
            c='blue', marker='x', s=40,
            edgecolor='black',
            label='cluster 3')

plt.legend()
plt.tight_layout()
plt.show()
```

hw5

This private post is only visible to Instructors and MD SHIRAJUM MUNIR

Updated 2 years ago by MD SHIRAJUM MUNIR

followup discussions *for lingering questions and comments*