alessandro.pomponio2

February 6, 2021

1 Alessandro Pomponio - 0000920265

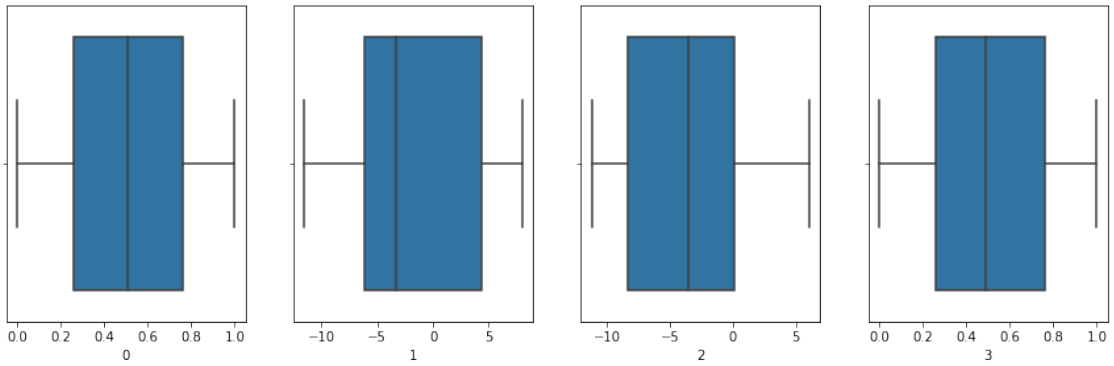
[1]:
```
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples
from plot_silhouette import plot_silhouette

# Variables
file_name = 'lab_exercise.csv'
separator = ','
random_state = 42

# Directives
%matplotlib inline
np.random.seed(random_state)
```

1. the boxplots of the attributes and a comment on remarkable situations, if any (2pt)
2. a pairplot of the data (see Seaborn pairplot) and a comment on remarkable situations, if any (2pt)
3. a clustering schema using a method of your choice exploring a range of parameter values (5pt)
4. the plot of the global inertia (SSD) and silhouette index for the parameter values you examine (4pt)
5. the optimal parameters of your choice (4pt)
6. a pairplot of the data using as hue the cluster assignment with the optimal parameter (3pt)
7. a plot of the silhouette index for the data points, grouped according to the clusters (4pt)
8. A sorted list of the discovered clusters for decreasing sizes (7pt)

1.1 1. the boxplots of the attributes and a comment on remarkable situations, if any (2pt)

[2]:
```
# Read the file
X = pd.read_csv(file_name, delimiter = separator, header = None)
```

[3]:
```
# Produce the boxplots
plt.figure(figsize=(15,15))
pos = 1
for i in X.columns:
    plt.subplot(3, 4, pos) # It looks bad otherwise
    sns.boxplot(X[i])
    pos += 1
```

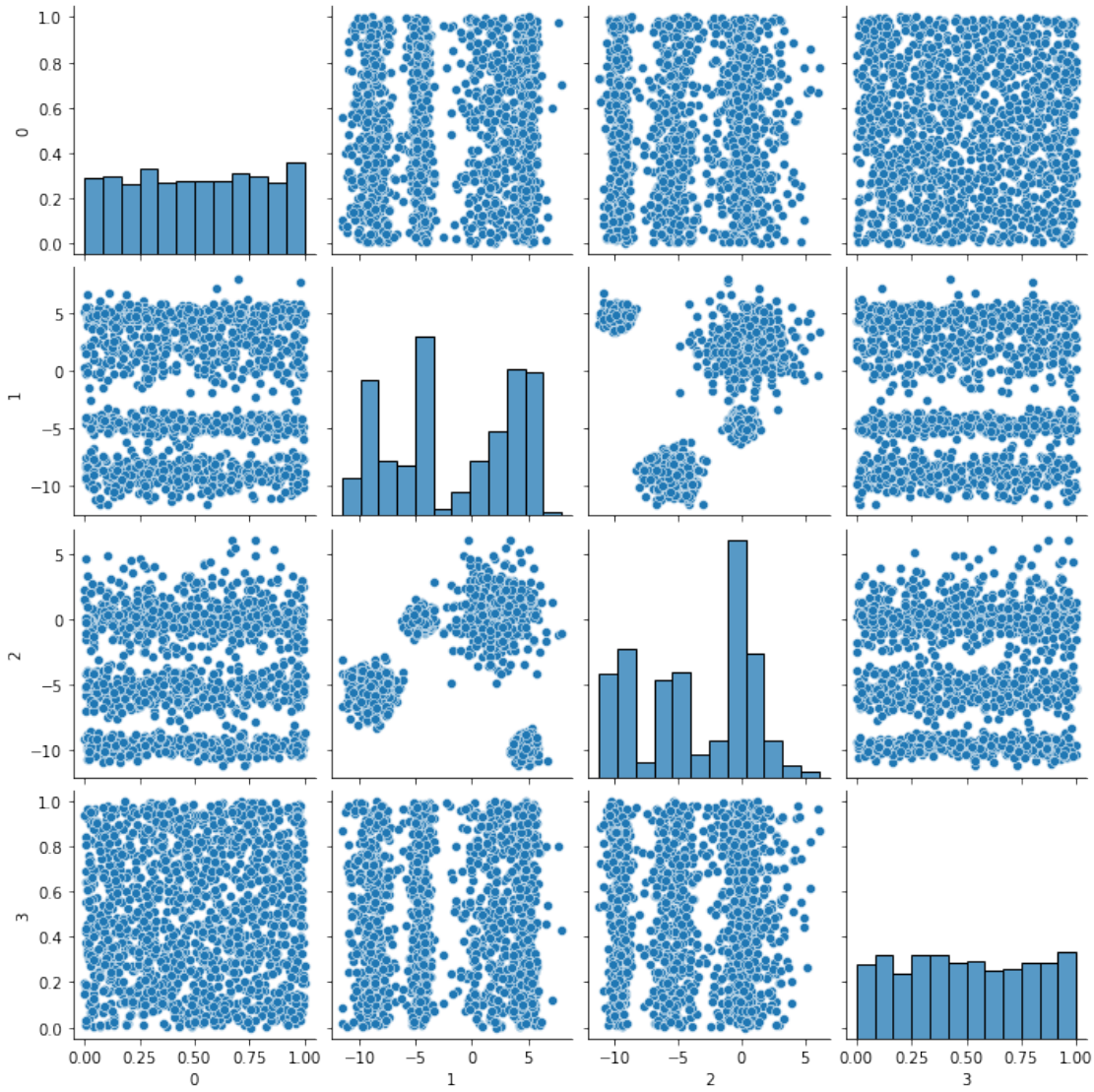


The boxplots show that there are no outliers, the distribution of 0 and 3 is very similar. 1 and 2 have a similar median value but different distribution of values. There doesn't seem to be any particular situaion showing.

1.2 2. a pairplot of the data (see Seaborn pairplot) and a comment on remarkable situations, if any (2pt)

[4]:
```
sns.pairplot(X)
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x2588e1ca340>
```



From the pairplot it is clear that the columns 1 and 2 tend to form quite distinct clusters. They're probably our best bet for our clustering efforts.

1.3 3. a clustering schema using a method of your choice exploring a range of parameter values (5pt)

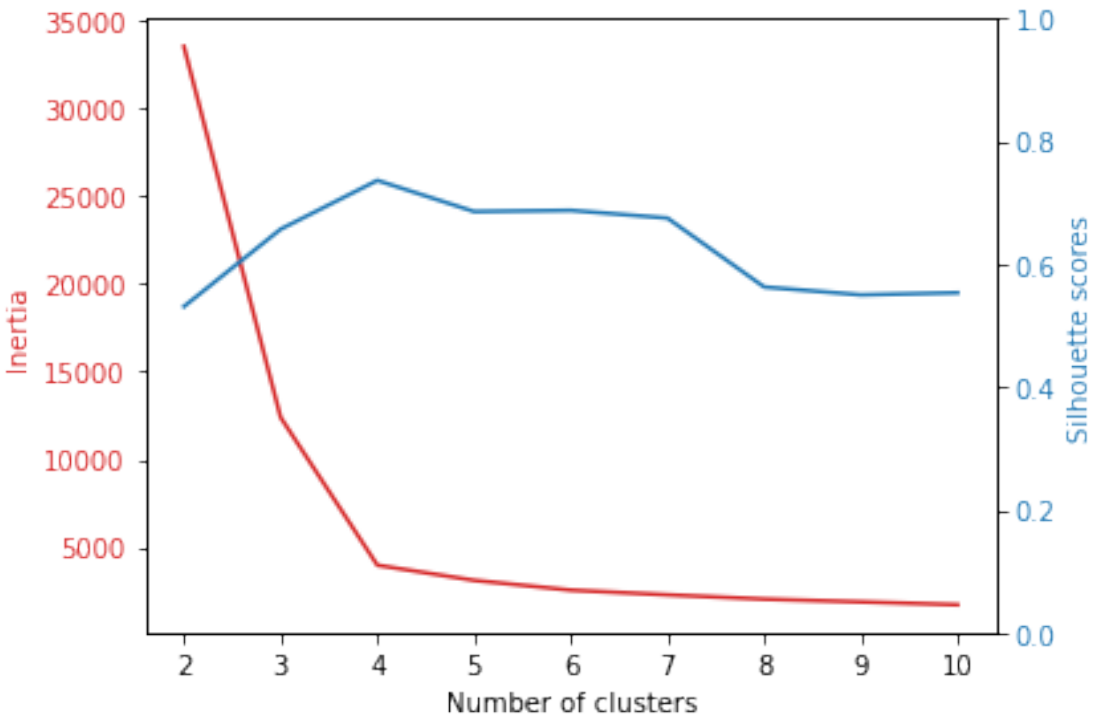
In order to find a clustering scheme, we will use K-means with the elbow method, ranging from 2 to 10 clusters

[5]:
```
k_range = range(2,11)

# Distortion and Silhouette Score as measures
distortions = []
silhouette_scores = []
```

```
for i in k_range:

    km = KMeans(n_clusters = i,
                init = 'k-means++',
                n_init = 10,
                max_iter = 300,
                random_state = random_state)

    y_km = km.fit_predict(X)
    distortions.append(km.inertia_)
    silhouette_scores.append(silhouette_score(X,y_km))
```

1.4 4. the plot of the global inertia (SSD) and silhouette index for the parameter values you examine (4pt)

[6]:
```
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Number of clusters')
ax1.set_ylabel('Inertia', color=color)
ax1.plot(k_range, distortions, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('Silhouette scores', color=color)  # we already handled the␣
 ↪x-label with ax1
ax2.plot(k_range, silhouette_scores, color=color)
ax2.tick_params(axis='y', labelcolor=color)
ax2.set_ylim(0,1) # the axis for silhouette is [0,1]

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```



1.5 5. the optimal parameters of your choice (4pt)

Both the silhouette scores and the inertia elbow suggest that the best number of clusters is 4, which is in line with what we were expecting, given the initial pairplots

[7]:
```
best_k = 4
```

1.6 6. a pairplot of the data using as hue the cluster assignment with the optimal parameter (3pt)

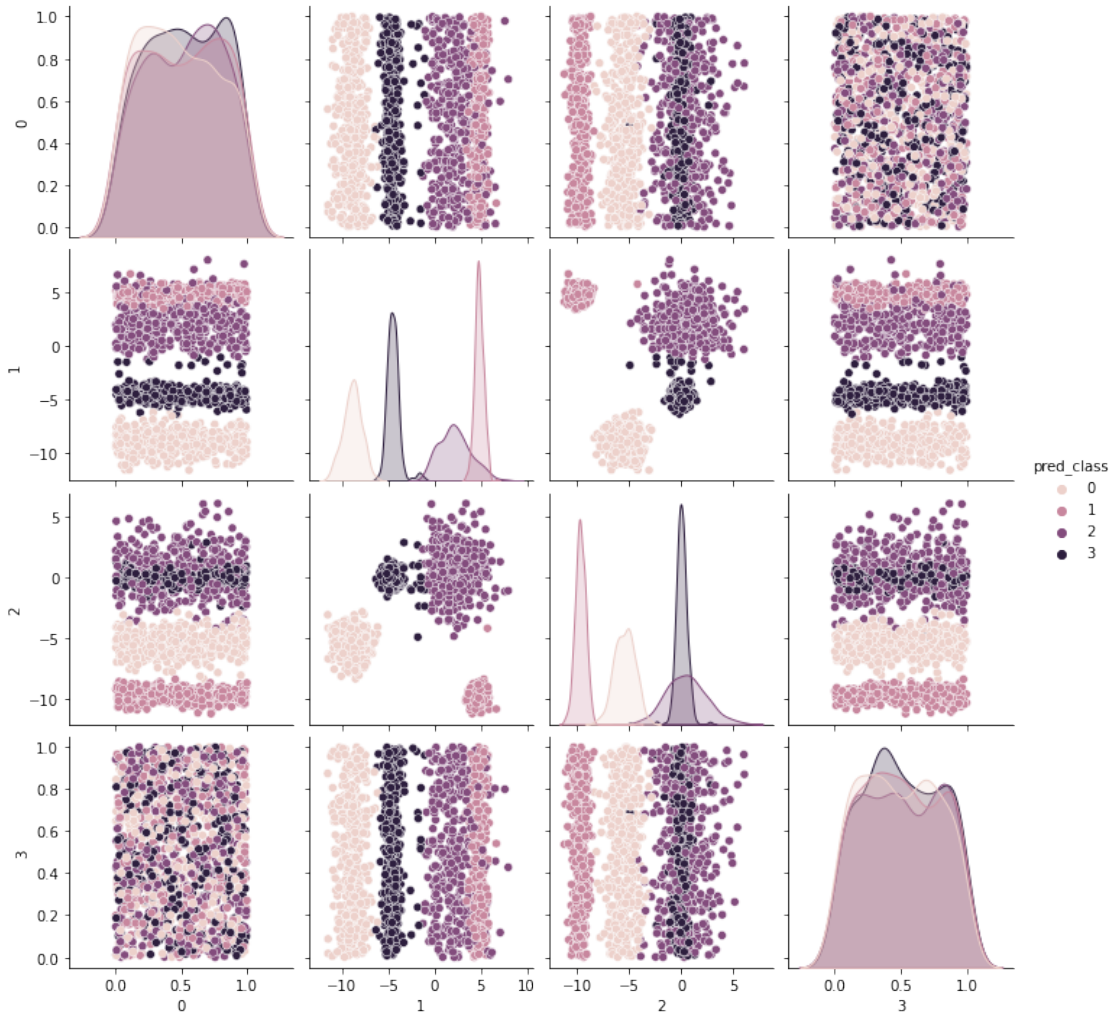
[8]:
```
# Create a new Kmeans classifier with the best parameter we found
km = KMeans(n_clusters = 4,
                init = 'k-means++',
                n_init = 10,
                max_iter = 300,
                random_state = random_state)

# Predict the cluster labels
y_km = km.fit_predict(X)
```

In order to use the predicted labels as hue we will add it to a new dataframe using the `assign` method

[9]:
```
X_pred = X.assign(pred_class = y_km)
```

[10]:
```
sns.pairplot(X_pred, hue = 'pred_class')
```

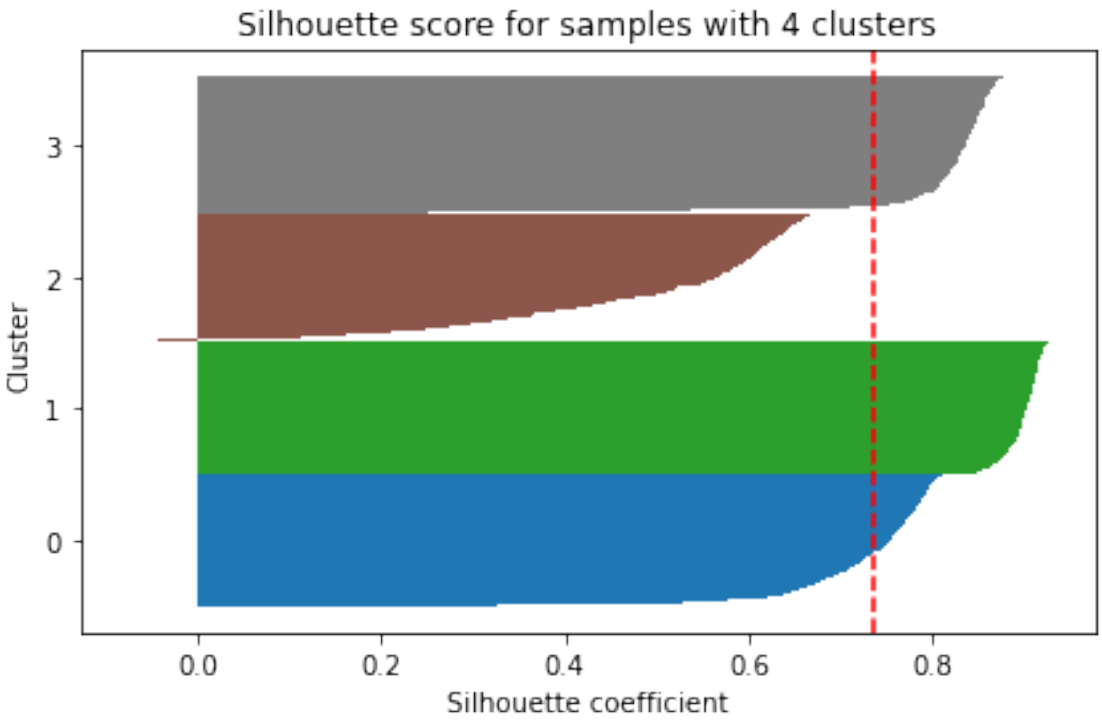
[10]:
```
<seaborn.axisgrid.PairGrid at 0x2588e9e3250>
```



1.7 7. a plot of the silhouette index for the data points, grouped according to the clusters (4pt)

In order to perform this task, we will use the `plot_silhouette` function that was introduced in the exercises in class

[11]:
```
# Compute the Silhouette Coefficient for each sample, with the euclidean metric
silhouette_score_samples = silhouette_samples(X, y_km, metric='euclidean')
plt.title(f"Silhouette score for samples with {best_k} clusters")
plot_silhouette(silhouette_score_samples, y_km)
```



1.8 7. A sorted list of the discovered clusters for decreasing sizes (7pt)

To make this task easier, we leverage numpy's function `bincount`

[12]:
```
occurrences = np.bincount(y_km)
```

`bincount` created an array that contains as index the cluster numbers, as value, the elements in that cluster. We can then create tuples to have this association in an explicit way

[13]:
```
item_cluster_tuples = [(qty, idx) for idx, qty in enumerate(occurrences)]
```

[14]:
```
item_cluster_tuples
```

[14]:
```
[(375, 0), (376, 1), (359, 2), (390, 3)]
```

We can now sort the tuples and extract the cluster index to obtain what was requested

[15]:
```
sorted_clusters = [i[1] for i in sorted(item_cluster_tuples, reverse = True)]
```

[16]:
```
sorted_clusters
```

[16]:
```
[3, 1, 0, 2]
```