Laboratorium 6 – Klasteryzacja

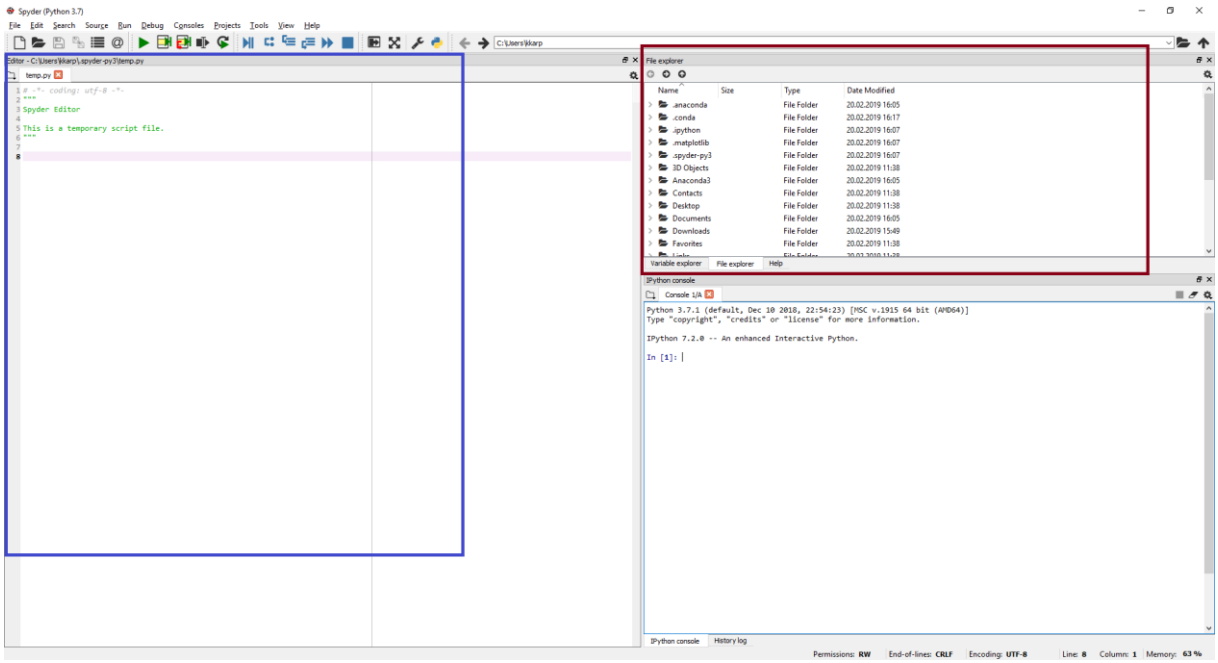
Zadania laboratoryjne:

Klasteryzacja (grupowanie, analiza skupień) danych jest techniką, w której zbiory danych (elementów) dzieli się na względnie jednorodne klasy. Elementy grupuje się na podstawie podobieństwa wyrażanego za pomocą metryk (funkcji podobieństwa). Podobnie jak w przypadku klasyfikacji, zastosowanie tej techniki pozwala na dokonanie pewnego uogólnienia (np. mężczyźni po 50 roku życia i z pensją 150 tys. rocznie, kupią mecedesa). Różnica pomiędzy obiema technikami jest taka, że klasyfikując spodziewamy się jakiegoś wyniku (chociażby ustalając, które zmienne są niezależne, a które zależne), natomiast grupując dane nie zakładamy, że otrzymamy odpowiedź na konkretne pytanie, raczej oczekujemy, że zobaczymy w jakie klastry układają się zebrane dane i na tej podstawie wyciągamy wnioski.

1. Przygotowanie środowiska do pracy.
   1.1. Pobrać i zainstalować platformę Anaconda (https://www.anaconda.com/distribution/).
   1.2. Uruchomić platformę Anaconda, a następnie aplikację Spyder.
   1.3. Na komputerze utworzyć nowy folder i zapisać w nim otrzymany od prowadzącego plik z danymi (Mall_Customers.csv).

UWAGA! Pliki mogą być wykorzystywane podczas kolejnych laboratoriów, dlatego po zakończeniu zajęć należy skopiować folder na prywatny nośnik.

   1.4. W programie Spyder ustawić swój folder jako katalog roboczy (obszar zaznaczony czerwonym prostokątem na Rysunku 1.). Zapisać pod dowolną nazwą plik temp.py (widoczny w obszarze zaznaczonym niebieskim prostokątem na Rysunku 1.) w tym samym folderze co plik ze zbiorem danych.



Rysunek 1. Interfejs programu Spyder.

2. **Metoda k-średnich (*k-means*)**
   Wstęp.

Metod k-średnich należy do algorytmów niehierarchicznych. Grupowanie odbywa się poprzez wstępn podział populacji na z góry założoną liczbę klas. Uzyskany podział jest następnie poprawiany w ten sposób, że niektóre elementy są przenoszone do innych klas, tak, aby uzyskać minimalną wariancję wewnątrz każdej z nich. Wykonuje się to, żeby zapewnić jak największe podobieństwa elementów w ramach każdej z klas, przy jednoczesnej maksymalnej różnicy pomiędzy samymi klasami. Podstawowy algorytm wygląda następująco:

- Ustalenie liczby klas
- Wybór środków (centrów) klas, np. poprzez losowy wybór k obserwacji, wybór k pierwszych obserwacji ze zbioru czy dobór pozwalający na maksymalizację odległości skupień
- przypisanie punktów do najbliższych centrów klas - każdy element przypisujemy do klasy, do którego środka ma najbliżej
- wyliczenie nowych centrów skupień - najczęściej nowym środkiem klasy jest ten punkt, którego współrzędne stanowią średnią arytmetyczną współrzędnych elementów należących do tej klasy,
- powtarzanie algorytmu aż do osiągnięcia kryterium zbieżności

Zadania.

Wykorzystując napisany na poprzednich zajęciach skrypt, przygotować zbiór danych do analizy.

2.1. Wczytać zbiór danych Mall_Customers.csv.

2.2. Utworzyć macierz danych Annual Income oraz Spending Score

2.3. Zidentyfikować optymalną liczbę klastrów za pomocą metody „łokcia” (elbow method).

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Przeanalizować otrzymany wykres. Zaobserwować, że wraz ze wzrostem podziału na coraz większą liczbę klastrów suma wariancji będzie malała. Jest to naturalne, ponieważ klastry stają się coraz mniejsze (w związku z tym naturalnie odległości pomiędzy punktami też będą mniejsze). Optymalna liczba klastrów to taka, dla której suma wariancji przestaje gwałtownie maleć, a dokładanie kolejnych klastrów nie wprowadza już dużej poprawy. Objawia się to pojawieniem się na wykresie silnego zagięcia, tzw. "łokcia".

2.4. Dopasować model KMeans do zestawu danych wykorzystując klasę KMeans.

```
# Tworzenie obiektu klasy KMeans z argumentem n_clusters = 5
(tyle wyznaczyliśmy korzystając z metody „łokcia”)

kmeans = KMeans(n_clusters = 5, init = 'k-means++',
random_state = 42)
```

```
# Dopasowanie modelu

y_kmeans = kmeans.fit_predict(X)
```

2.5. Wykreślić utworzone klastry.

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

2.6. Wykonać sprawozdanie z realizacji podpunktów 2.1 – 2.5. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wykres z podpunktu 2.3
- Wykres z podpunktu 2.4

3. Metoda hierarchiczna

Wstęp.

W tej metodzie algorytm tworzy dla zbioru elementów hierarchię klasyfikacji w taki sposób, że najpierw ustala podział, w którym każdy element stanowi samodzielny klaster, a kończąc na ustaleniu podziału, w którym wszystkie obiekty należą do jednego klastra.

Zadania.

Wykorzystując napisany na poprzednich zajęciach skrypt, przygotować zbiór danych do analizy.

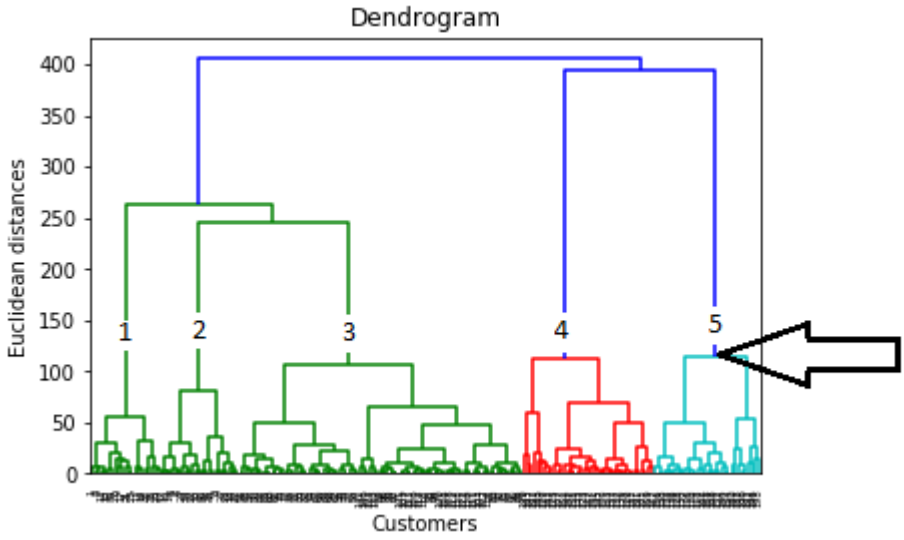
3.1. Wczytać zbiór danych Mall_Customers.csv.

3.2. Utworzyć macierz danych Annual Income oraz Spending Score

3.3. Zidentyfikować optymalną liczbę klastrów za pomocą metody dendrogram.

```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```

Przeanalizować otrzymany wykres. W tej metodzie wyznacza optymalnej liczby klastrów, na wykresie należy odnaleźć najdłuższą pionową linię, a następnie miejsce jej przecięcia z pierwszą linią pionową. Następnie należy policzyć ile pionowych linii jest w tym obszarze. Liczba ta mówi o optymalnej liczbie klastrów.



3.4. Dopasować model hierarchiczny do zestawu danych wykorzystując klasę AgglomerativeClustering.

```
from sklearn.cluster import AgglomerativeClustering
# Tworzenie obiektu klasy KMeans z argumentem n_clusters = 5 (tyle wyznaczyliśmy korzystając z metody „łokcia”)
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
# Dopasowanie modelu

y_hc = hc.fit_predict(X)
```

3.5. Wykreślić utworzone klastry.

```
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

3.6. Wykonać sprawozdanie z realizacji podpunktów 3.1 – 3.5. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wykres z podpunktu 3.3
- Wykres z podpunktu 3.4