Filip Wałęga
Dokumentacja Pracy z Systemów Sztucznej Inteligencji
K-NN i Zbiory Miękkie

K-**NN:**

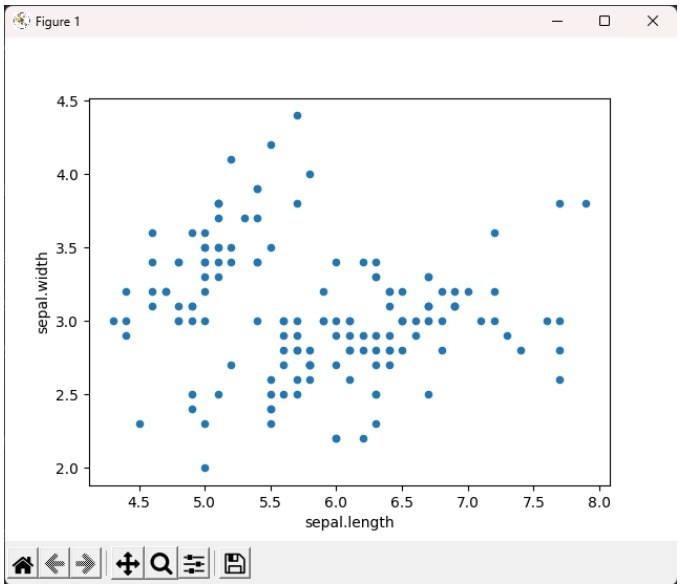
```
# Zadanie 1. Wczytaj do programu zbiór danych o kwiatach Iris.

df = pd.read_csv('pliki/iris/iris.csv')
```

Jak widać na obrazku poniżej DF został wczytany

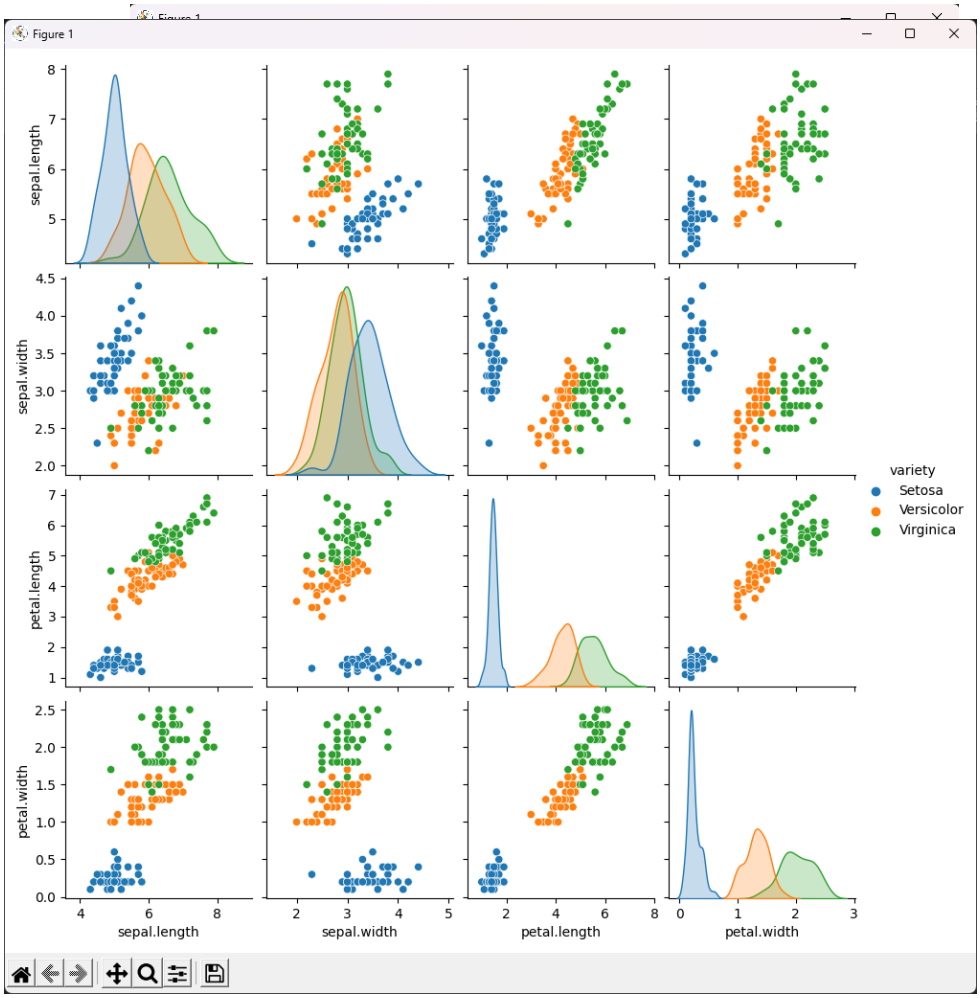
```
K-NN:
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
```

Wynik df.head() powyżej. Do wykresów w PyCharm muszę użyć dodatkowo plt.show()

```
# Zadanie 2. Wykonaj analizę danych zbioru Iris.

# Drukuje pierwsze pare wierszy, można podać konkretną ilość
print(df.head())

# Wbudowana metoda do robienia wykresu z danych DataFrame z modułu pandas
df.plot(kind = "scatter", x="sepal.length", y="sepal.width")
plt.show()     # To moje do wyświetlania wykresu w pycharm

# Piękna metoda z modułu seaborn do Analizy Danych, daje nam zbiór wykresów z całego DataFrame
sns.pairplot(data=df, hue="variety")
plt.show()
```



Wykres z DataFrame



Wykres Seaborn pairplot

Kolejno następuje Tasacja, Normalizacja oraz Podział Danych na zbiór do uczenia i do walidacji. Następnie Klasteryzacja oraz wyświetlenie wyniku K-NN oraz prawdziwego wyniku dla porównania.

```
DataProcessor.shuffle(df)               # Tasacja bd Irysów
DataProcessor.normalization(df)         # Normalizacja bd Irysów
print(df.head())                        # Sprawdzenie danych

train, test = DataProcessor.split(df, 0.7)      # Podział zbioru na train i test
print(len(test), len(train))                    # Sprawdzenie czy poprawnie dzieli
print(KNN.clustering(test.iloc[0], train, 4))   # K-NN pierwszego sampla z części test max(wyników_z_min_odległości)
print(f"Variety: {test.iloc[0].variety}")       # Wydrukowanie prawdziwej odmiany sampla
```

```
   sepal.length  sepal.width  petal.length  petal.width     variety
0      0.583333     0.500000      0.728814     0.916667   Virginica
1      0.500000     0.416667      0.661017     0.708333   Virginica
2      0.555556     0.291667      0.661017     0.708333   Virginica
3      0.361111     0.416667      0.525424     0.500000  Versicolor
4      0.916667     0.416667      0.949153     0.833333   Virginica
45 105
(('Versicolor', 4), {'Setosa': 0, 'Versicolor': 4, 'Virginica': 0})
Variety: Versicolor
```

Zbiory Miękkie:

Inicjacja zmiennych potrzebnych do mojej implementacji algorytmu opierającego się na zbiorach miękkich

```
#       tzn 4. (BrainTiredException)
# Zadanie i+1. Zaimplementuj algorytm inferencji zbiorami miękkimi dla dla klasyfikacji wybranych produktów w sklepie.

# Tuple cech produktów wersja 'Human Readable'
traits = ("czerwone", "zielone", "okrągłe", "szpiczaste", "słodkie", "ostre")

# Dict produktów, w wersji, która jest wygodniejsza dla mnie i dla komputera
products = {
    "onion" : (0, 1, 1, 0, 0, 0),
    "paprica" : (1, 0, 0, 0, 1, 1),
    "carrot" : (0, 0, 0, 1, 1, 0)
}
# Są to dwa tuple reprezentujace wejście użytkownika
client = ("czerwone", "słodkie", "ostre")
client_w_input = (0.3, 0.3, 0.4)

# Inicjacja zmiennych 'przetłumaczonych' z 'clientInput' na 'shopInput'
client_traits = [0, 0, 0, 0, 0, 0]
client_weights = [0, 0, 0, 0, 0, 0]
```

```
# Translacja zachcianek klienta na te sklepowe
for i in range(len(traits)):
    if traits[i] in client:
        index_trait = client.index(traits[i])
        client_weights[i] = client_w_input[index_trait]
        client_traits[i] = 1
```

```
print(client_traits) #DEBUG :D

# Liczenie wszystkich produktów wg preferencji klienta
cpSum ={}
for product in products:
    cpSum[product] = 0
    for i in range(len(traits)):
        if products[product][i] == client_traits[i]:
            if products[product][i] == 1:
                cpSum[product] += 1*client_weights[i]
```

```
[1, 0, 0, 0, 1, 1]
```

```
#Wydrukowanie najbardziej korzystnego produktu wg preferencji klienta
print(max(cpSum), cpSum)
```

```
paprica {'onion': 0, 'paprica': 1.0, 'carrot': 0.3}
```