

Patryk Ostrowski

Moduł 4, zadanie 1 – Irisy

1. Zacytanie pliku i ‘przegląd’ zawartości:

```
import pandas as pd
import matplotlib.pyplot as plt

# Konfiguracja: wyświetlaj wszystko
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

df = pd.read_csv('25_iris.csv')
print(df)
print()
```

```
C:\Users\pos\AppData\Local\Programs\Python\Python313\python.exe C:\Users\pos\Documents\Python\Python-Lessons\akademia\zadania\mod_4\mod_4_zad_1.py
długość kielicha (sepal length) szerokość kielicha (sepal width) \
0                  5.1                 3.5
1                  4.9                 3.0
2                  4.7                 3.2
3                  4.6                 3.1
4                  5.0                 3.0
5                  5.4                 3.9
6                  4.6                 3.4
7                  5.0                 3.4
8                  4.4                 2.9
9                  4.9                 3.1
10                 5.4                 3.7
11                 4.8                 3.4
12                 4.8                 3.0
13                 4.3                 3.0
14                 5.8                 4.0
```

	długość płatka (petal length)	szerokość płatka (petal width) \
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2
5	1.7	0.4
6	1.4	0.3
7	1.5	0.2
8	1.4	0.2
9	1.5	0.1
10	1.5	0.2
11	1.6	0.2
12	1.4	0.1
13	1.1	0.1
14	1.2	0.2
15	1.5	0.4

149	5.1	1.8
klasa (class)		
0	Iris-setosa	
1	Iris-setosa	
2	Iris-setosa	
3	Iris-setosa	
4	Iris-setosa	
5	Iris-setosa	
6	Iris-setosa	
7	Iris-setosa	
8	Iris-setosa	
9	Iris-setosa	
10	Iris-setosa	
11	Iris-setosa	
12	Iris-setosa	
13	Iris-setosa	

2. Wylistowanie kolumn zastanych i zastąpienie ich własnymi:

```
print('Kolumny:')
print(df.columns)
df.columns = ['Dł_kiel', 'Szer_kiel', 'Dł_płat', 'Szer_płat', 'Klasa']
print()
print('Po zmianie nazw kolumn:')
print(df)
print()
```

```
Kolumny:
Index(['długość kielicha (sepal length)', 'szerokość kielicha (sepal width)',
       'długość płatka (petal length)', 'szerokość płatka (petal width)',
       'klasa (class)'],
      dtype='object')

Po zmianie nazw kolumn:
   Dł_kiel  Szer_kiel  Dł_płat  Szer_płat      Klasa
0      5.1       3.5     1.4      0.2  Iris-setosa
1      4.9       3.0     1.4      0.2  Iris-setosa
2      4.7       3.2     1.3      0.2  Iris-setosa
3      4.6       3.1     1.5      0.2  Iris-setosa
4      5.0       3.6     1.4      0.2  Iris-setosa
5      5.4       3.9     1.7      0.4  Iris-setosa
6      4.6       3.4     1.4      0.3  Iris-setosa
7      5.0       3.4     1.5      0.2  Iris-setosa
```

3. Zliczenie wartości dla każdej klasy i zliczenie unikatów w każdej z kolumn:

```
print('Łącznie w każdej klasie jest:')
print(df.groupby('Klasa').size())
print()

print('Unikaty:')
print(df.nunique())
print()
```

```
Łącznie w każdej klasie jest:
Klasa
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64

Unikaty:
Dł_kiel        35
Szer_kiel      23
Dł_płat        43
Szer_płat      22
Klasa          3
dtype: int64
```

4. Praca na danych dot. kolumny długości kielichów:

```
# długości kielichów
lista_dlugosci_kielichow = df["Dł_kiel"].to_list()
print('Lista wartości długości kielichów:', lista_dlugosci_kielichow)

dlugosc_kielicha_max = max(lista_dlugosci_kielichow)
print('Najdłuższy kielich:', dlugosc_kielicha_max)

dlugosc_kielicha_min = min(lista_dlugosci_kielichow)
print('Najkrótszy kielich:', dlugosc_kielicha_min)

potencjal_wzrostu_wzdluz = dlugosc_kielicha_max - dlugosc_kielicha_min
procentowy_potencjal_wzrostu_wzdluz = potencjal_wzrostu_wzdluz / dlugosc_kielicha_min * 100
print('Irys wzduż może wzrosnąć o', round(potencjal_wzrostu_wzdluz, 1), 'cm, czyli o', round(procentowy_potencjal_wzrostu_wzdluz, 1), 'procent.')
print()

Listy wartości długości kielichów: [5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5.0, 5.0, 5.2, 4.7, 4.8, 5.4, 5.1]
Najdłuższy kielich: 7.9
Najkrótszy kielich: 4.3
Irys wzduż może wzrosnąć o 3.6 cm, czyli o 83.7 procent.
```

5. Praca na danych dot. kolumny szerokości kielichów:

6. Praca na danych dot. kolumny długości płatków – na Data Frame a nie na liście tym

razem:

```
# długości platków - liczone na data frame
najdluzszy_platek = df['Dł_plat'].max()
print('Najdłuzszy platek ma', najdluzszy_platek, 'cm.')

najkrotszy_platek = df['Dł_plat'].min()
print('Najkrótszy platek ma', najkrotszy_platek, 'cm.')

potencjal_wzrostu_platka_wzdluz = najdluzszy_platek - najkrotszy_platek
procentowy_potencjal_wzrostu_platka_wzdluz = potencjal_wzrostu_platka_wzdluz / najkrotszy_platek * 100
print('Platek IRSYNA wzduż może przyrosnąć o', round(procentowy_potencjal_wzrostu_platka_wzdluz, 1), 'cm, czyli o', round(procentowy_potencjal_wzrostu_platka_wzdluz, 1), 'procent.')
print()
```

Najdłuższy płatek ma 6,9 cm.

Najkrótszy płatek ma 1.0 cm.

Płatek irysa wzdłuż może przyrosnąć o 5.9 cm, czyli o 590.0 procent.

7. Praca na danych dot. kolumny szerokości płatków – na Data Frame a nie na liście tym razem:

```
# szerokości płatków - liczzone na data frame
najszerszy_platek = df['Szer_płat'].max()
print('Najszerszy płatek ma', najszerszy_platek, 'cm.')

najwęższy_platek = df['Szer_płat'].min()
print('Najwęższy płatek ma', najwęższy_platek, 'cm.')
potencjal_wzrostu_platka_wszerz = najszerszy_platek - najwęższy_platek
procentowy_potencjal_wzrostu_platka_wszerz = potencjal_wzrostu_platka_wszerz / najwęższy_platek * 100
print('Płatek irysa wszerz może przyrosnąć o', round(potencjal_wzrostu_platka_wszerz, 1), 'cm, czyli o', round(procentowy_potencjal_wzrostu_platka_wszerz, 1), 'procent')
print()
```

Najszerszy płatek ma 2.5 cm.

Najwęższy płatek ma 0.1 cm.

Płatek irysa wszerz może przyrosnąć o 2.4 cm, czyli o 2400.0 procent

8. Analiza grup w ramach agregacji:

```
# analiza grup na data frame
print('Zbiorczo - wskaźniki agregujące jako data frame:')
print(
    df.groupby("Klasa")[["Dł_kiel", "Szer_kiel", "Dł_płat", "Szer_płat"]].agg(['min', 'max', 'mean'])
)
```

Zbiorczo - wskaźniki agregujące jako data frame:

Klasa	Dł_kiel			Szer_kiel			Dł_płat		
	min	max	mean	min	max	mean	min	max	mean
Iris-setosa	4.3	5.8	5.006	2.3	4.4	3.418	1.0	1.9	1.464
Iris-versicolor	4.9	7.0	5.936	2.0	3.4	2.770	3.0	5.1	4.260
Iris-virginica	4.9	7.9	6.588	2.2	3.8	2.974	4.5	6.9	5.552

Szer_płat

Klasa	min	max	mean
Iris-setosa	0.1	0.6	0.244
Iris-versicolor	1.0	1.8	1.326
Iris-virginica	1.4	2.5	2.026

9. Napisałem funkcję, która jest następnie wywoływana w pętli i pokazuje wyliczenia – w tym przypadku średniej dla każdego z wymiarów irysa – dł. i szer. kielicha i odpowiednio płatka.

```
# wyliczenie średniej generyczną funkcją
def wylicz_srednia_wymiaru_dla_klasy(wymiar): # przy wywołaniu funkcji jako 'wymiar' podany będzie string wynikowy ze znanej już listy/obiektu kolumn z wywołania df.columns i usage new*
    # ta zmienne dla podanego wymiaru irysa - dł. lub szer. łodygi, dł. lub szer. kielicha - przechowią średnia dla zadeklarowanego przy wywołaniu funkcji wymiaru kwiatka
    srednia_wymiaru = df.groupby(['Klasa'])[wymiar].mean()
    print(srednia_wymiaru) # tu wyświetlamy średnia

# poniższa pętla wywołuje funkcje "wylicz_srednia_wymiaru_dla_klasy", ale najpierw zagląda do struktury "df.columns" i leci po jej pozycjach, które podaje funkcji jako parametry.
# Ponieważ ostatnia dana "Klasa" bo to jest typu, stąd -1
for i in range(len(df.columns) - 1):
    wymiar = df.columns[i]
    print("Wyliczam średnią dla:", wymiar)
    wylicz_srednia_wymiaru_dla_klasy(wymiar)
    print()
```

```
Wyliczam średnią dla: Dł_kiel
Klasa
Iris-setosa      5.006
Iris-versicolor  5.936
Iris-virginica   6.588
Name: Dł_kiel, dtype: float64
```

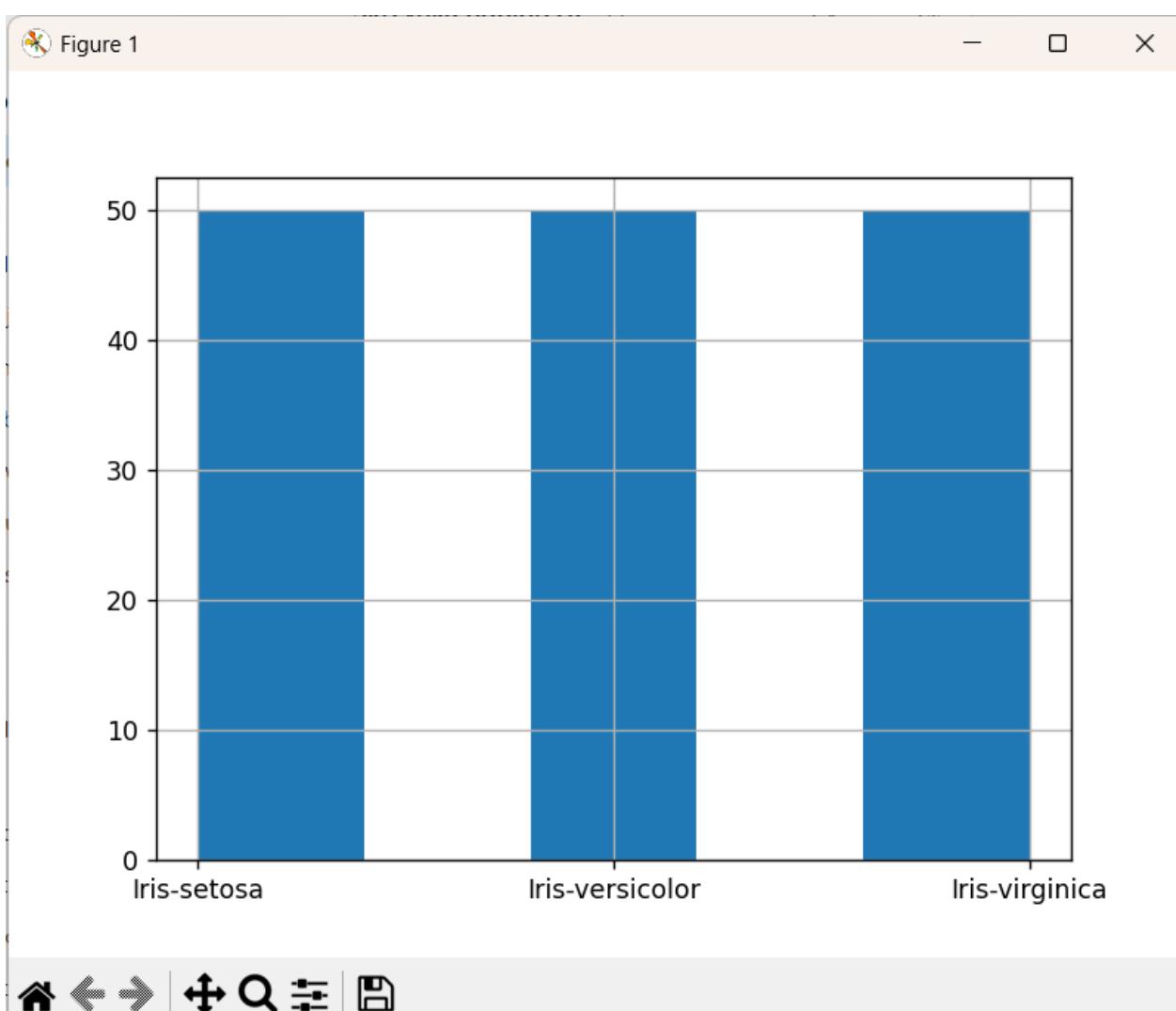
```
Wyliczam średnią dla: Szer_kiel
Klasa
Iris-setosa      3.418
Iris-versicolor   2.770
Iris-virginica   2.974
Name: Szer_kiel, dtype: float64
```

```
Wyliczam średnią dla: Dł_płat
Klasa
Iris-setosa      1.464
Iris-versicolor   4.260
Iris-virginica   5.552
Name: Dł_płat, dtype: float64
```

```
Wyliczam średnią dla: Szer_płat
Klasa
Iris-setosa      0.244
Iris-versicolor   1.326
Iris-virginica   2.026
Name: Szer_płat, dtype: float64
```

10. Wykresy

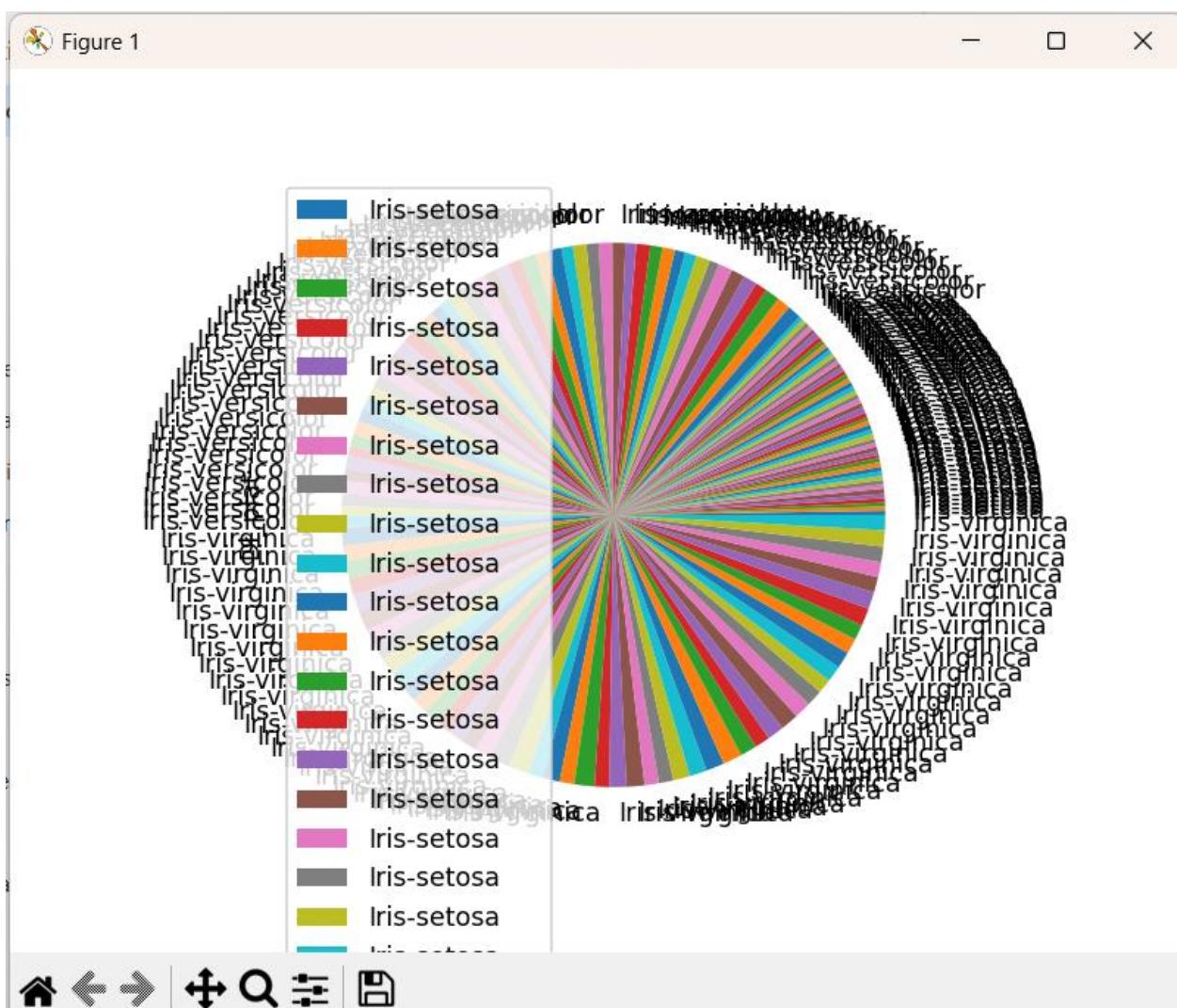
```
# histogram  
df[ "Klasa" ].hist(bins=5)  
plt.show()
```



Podział na pięć binsów przejłyście pokazuje udział próbek każdej klasy – akurat tutaj jest ich po 50 co sugeruje, że wcześniejsze i dalsze analizy są miarodajne, gdyż nie mamy klasy, której ilość próbek jest nadzwyczajnie duża lub nadzwyczajnie niewielka.

Wykres kołowy:

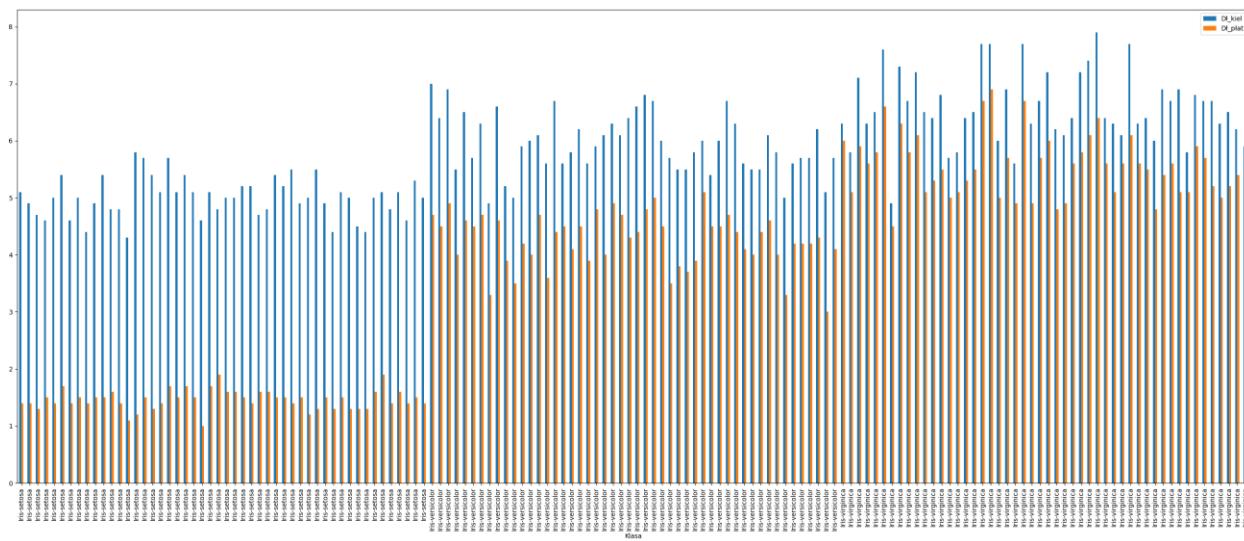
```
# wykres kołowy  
df.plot(kind="pie", y="Dł_płat", labels=df[ 'Klasa' ], legend=True)  
plt.show()
```



Przy zastanej konfiguracji Data Frame program pokazał co umie, czyli po prostu poszalał, ale to sugeruje poprawne wczytanie rzędów i serii danych. Dalszym krokiem byłoby wyłuskanie od biznesu co byłoby dla nich wartościowe i głębsze poznanie modułu wykresów przez developera 😊

Wykres słupkowy:

```
# wykres słupkowy
df.plot(kind="bar", x="Klasa", y=["Dł_kiel", "Dł_płat"])
plt.show()
```



Wykres – pomimo swojej pozornej nieczytelności – jest bardzo czytelny w pokazaniu tendencji wzrostowych dla poszczególnych odmian/typów irysów.

Co można z tego wyczytać? Na przykład przekrojowy zakres rozmiarów kwiata, co przy prowadzeniu biznesu może wpłynąć na strategię kształtowania cen, gdyż mniejsze egzemplarze wystarczy wpakować do mniejszego opakowania, mniej też miejsca mogą one zająć na ekspozycji, a co za tym idzie: 1) będziemy mogli/musimy zamówić różnego rodzaju/wymiaru opakowania, 2) będziemy różnicować cenę wliczoną za transport w zależności od tej wartości. Otóż większych kwiatów zmieści się mnie na ciężarówce transportowej, a więc trzeba rozważyć konwersję konkretnego wolumenu zamówienia na zysk; do rozważenia jest także większa – a co za tym idzie droższa – skrzynka w automacie paczkowym, gdyż gdzieś na którejś wartości osi Y możemy przekraczać progową wartość dla rozmiaru przesyłki M i zamawiać tylko skrytki L lub w ogóle być zmuszeni do omijania automatów i bardzo duże egzemplarze wysyłać wyłącznie kurierem z dostawą osobistą.

Wykresy punktowe:

```
# wykres punktowy
df.plot(kind="scatter", x="Klasa", y="Dł_kiel")
plt.show()

df.plot(kind="scatter", x="Klasa", y="Szer_kiel")
plt.show()

df.plot(kind="scatter", x="Klasa", y="Dł_płat")
plt.show()

df.plot(kind="scatter", x="Klasa", y="Szer_płat")
plt.show()
```

Figure 1

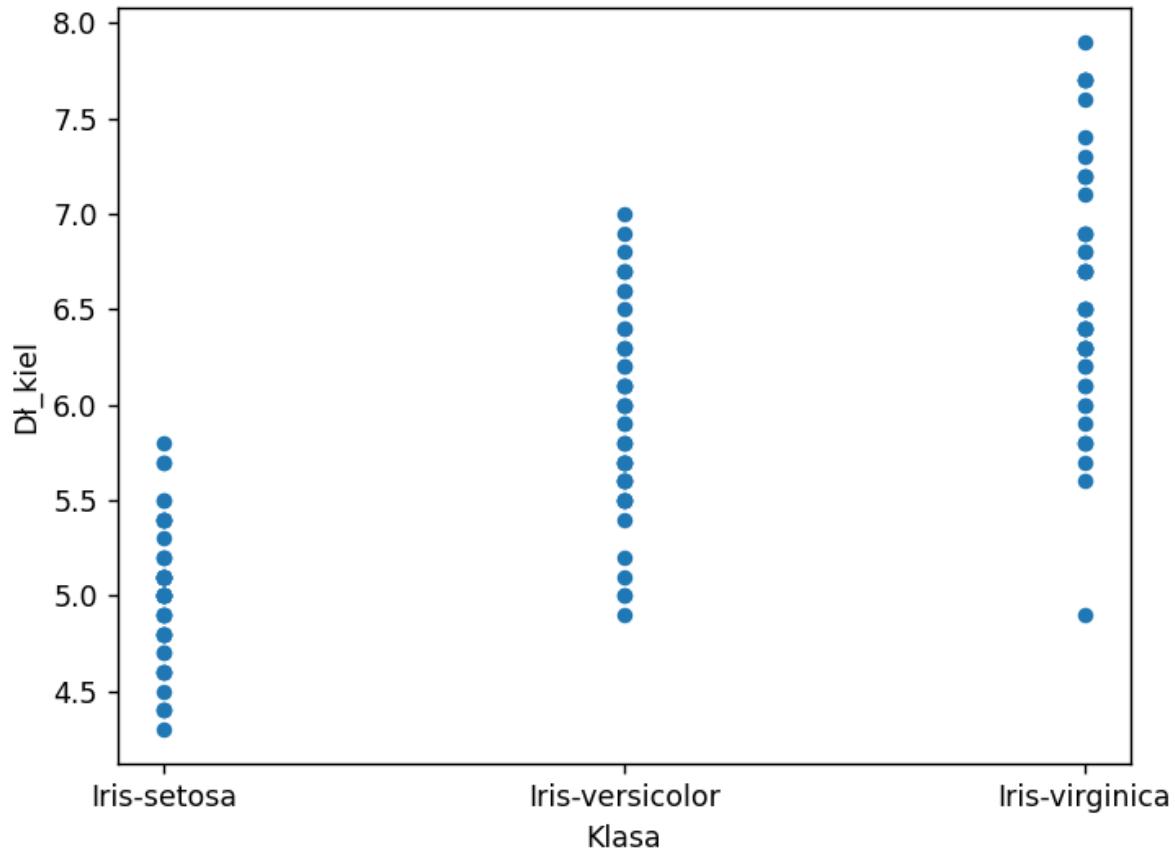


Figure 1

— □ ×

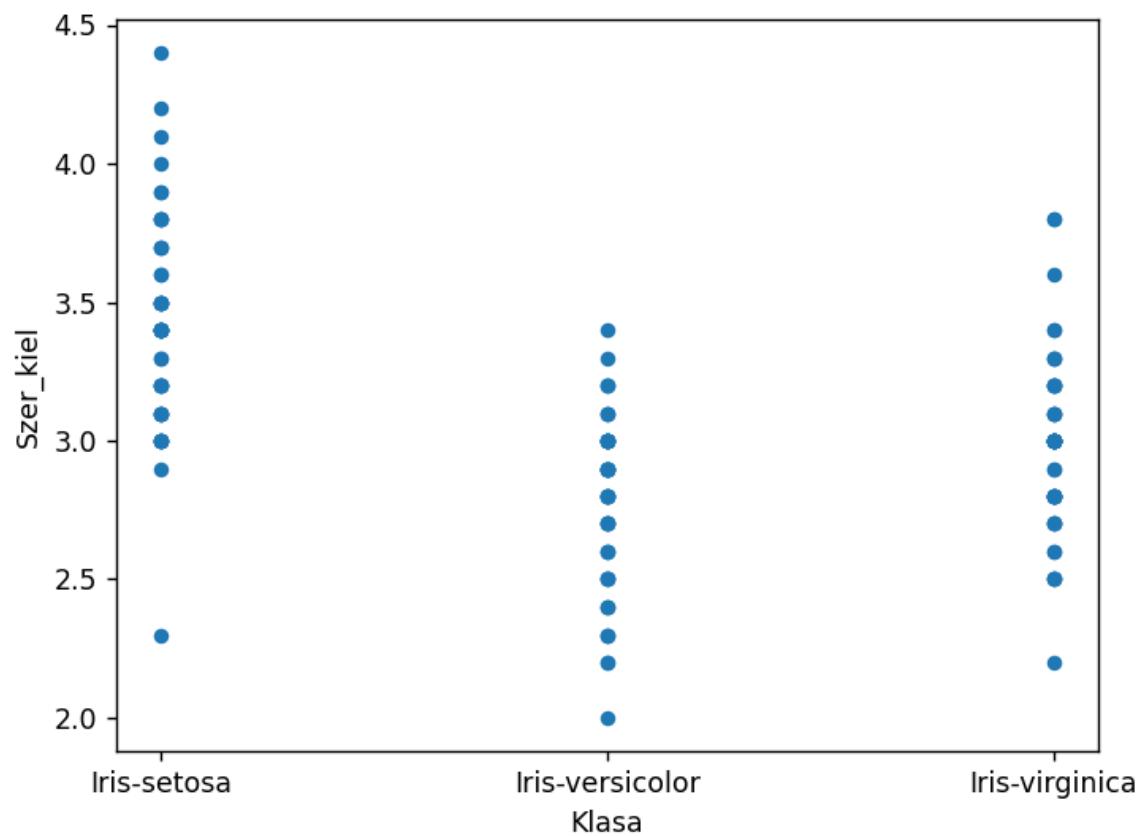


Figure 1

— □ ×

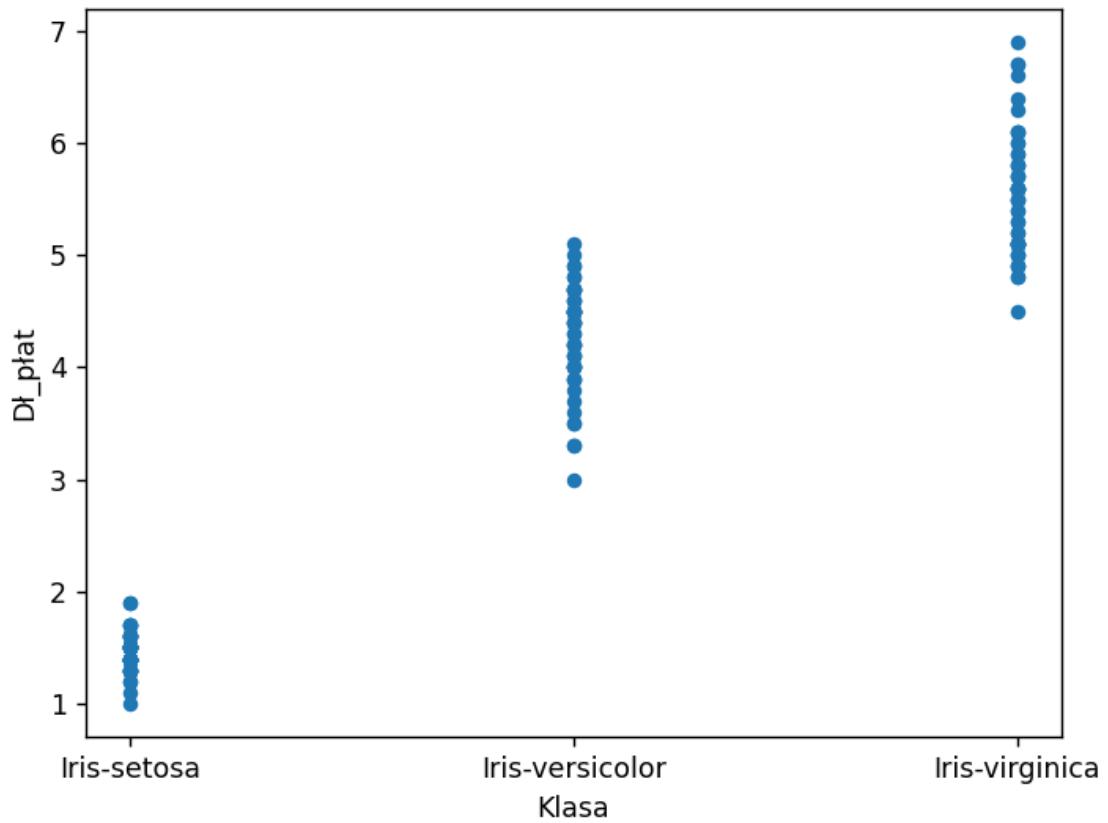
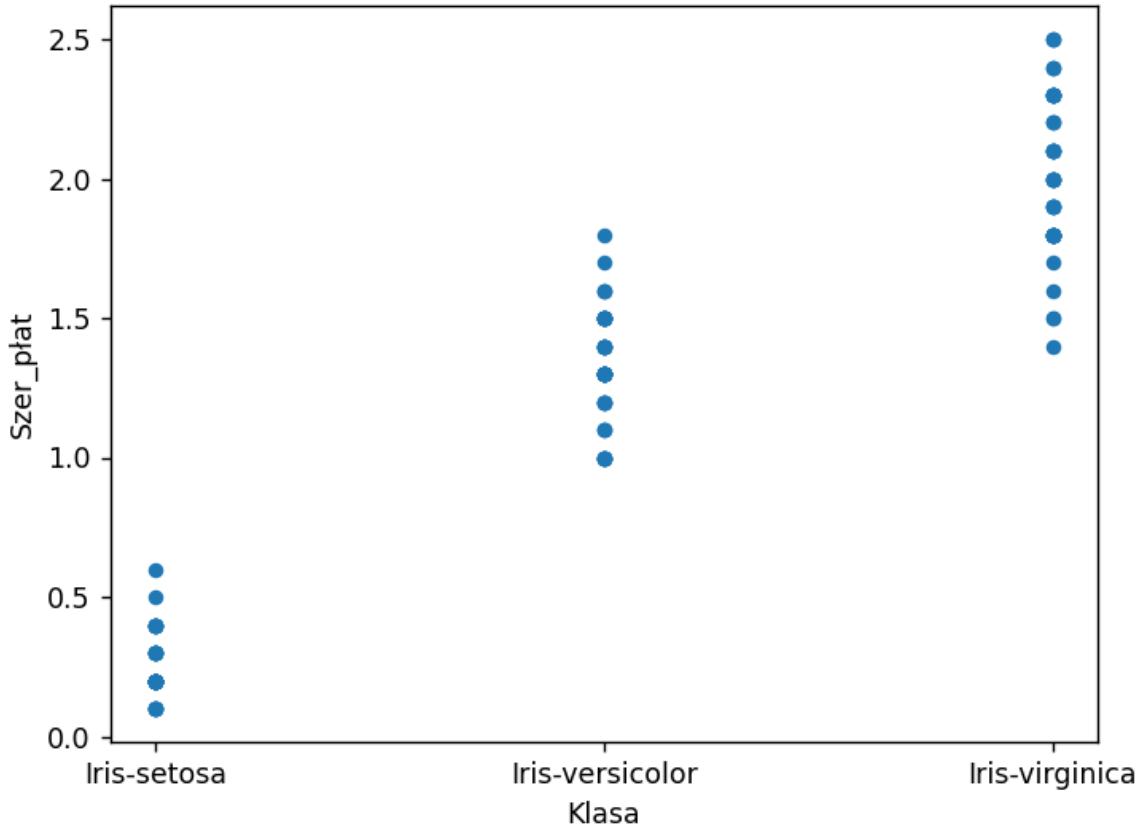


Figure 1

-

□

×

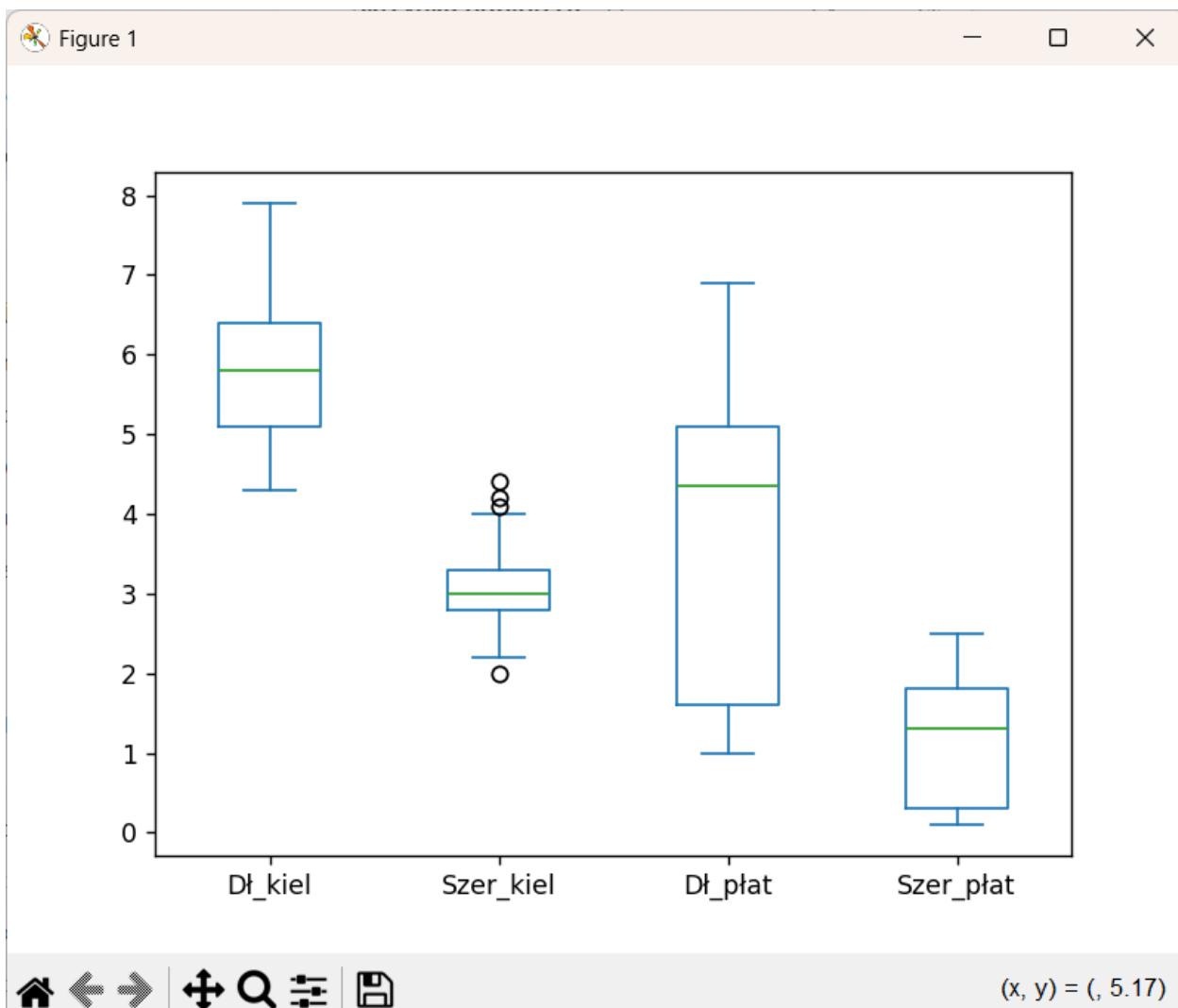


(x, y) = (Iris-virginica, 2.277)

Powyższe wykresy ukazują rozstrzał wartości dla każdego z analizowanych wymiarów każdej z klas irysów. Możemy dokładniej przyjrzeć się wartościom skrajnym i pogłębić analizę w oparciu o poprzedni wykres słupkowy rozważając opakowania ‘nie mniejsze niż’ i ‘nie większe niż’.

Wykres pudełkowy:

```
# wykres pudełkowy
df.plot(kind="box")
plt.show()
```



Ten wykres pokazuje wartości min. i max, oznakowuje kwartyle 1-wszy (percentyl 25-ty), 2-gi (percentyl 50-ty) i 3-ci (percentyl 75-ty), a także wartości odstające dla każdego z analizowanych wymiarów irysa z poddanego obserwacji zbioru próbek.

Analizę przeprowadzono w IDE PyCharm firmy JetBrains.

Repozytorium dostępne jest pod adresem: https://github.com/racibornio/Python-lessons/blob/master/akademia/zadania/mod_4/mod_4_zad_1.py