
	<p>Politechnika Bydgoska im. J.J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki</p>		
Przedmiot	Algorytmy i eksploracja danych		
Prowadzący	dr inż. Michał Kruczkowski		
Temat	drzewa decyzyjne - eksploracja zbiorów danych		
Studenci	Adam Szreiber, Cezary Naskręt		
Nr ćw.	1	Data wykonania	10.10.2023

Zadanie 1

Należało wczytać dane do obiektu "DataFrame", co uczyniono metodą `pd.read_csv()`

Następnie odwołano się do pola `shape` które reprezentuje wielowymiarowość.

Zliczanie ilości wystąpień wartości `null`, odbywa się za pomocą metody `isnull()`

w ten sposób otrzymano DataFrame zawierający wartości true/false w zależności od wartości w tym polu w pierwotnej tablicy.

Następnie `sum()` zlicza ilość wystąpień `true`, i zwraca wektor ilością wystąpień `null` w danym wierszu.

Kolejne wywołanie `sum()` zlicza wszystkie wartości w wektorze i zwraca ilość wszystkich wystąpień wartości 'null' w DataFrame.

```
zad1.py
1 import pandas as pd
2 # Wczytanie danych z pliku CSV
3 df = pd.read_csv('netflix_titles.csv')
4
5 # Wyświetlenie pierwszych kilku wierszy ramki danych
6 [wiersze_count, kolumny_count] = df.shape
7 print('Ilość wczytanych wierszy: ' + str(wiersze_count))
8
9 size_str = str(kolumny_count) + ' x ' + str(wiersze_count)
10 print('Wymiar wczytanych danych (' + size_str + ")")
11
12 ilosc_null = df.isnull().sum().sum()
13 print('Ilość wystąpień wartości "null": ' + str(ilosc_null))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● adam-pc@adampc-MS-7A34:~/Documents/mgr/AED-AlgorytmyEksploracjiDanych/lab1$ python3 zad1.py
Ilość wczytanych wierszy: 8807
Wymiar wczytanych danych (12 x 8807)
Ilość wystąpień wartości "null": 4307
```

Ryc 1. Kod realizujący założenia.

Zadanie 2

Należało wczytać zbiór danych "titanic".

Następnie po raz kolejny wykorzystano składnie `isnull()` oraz podwójne `sum()` aby uzyskać ilość pustych wartości w zbiorze.

Otrzymanie skumulowanej sumy ilości wartości `null` w poszczególnych kolumnach wykonano poprzez uruchomienie metody `.cumsum()` na wektorze ilości `null` w poszczególnych kolumnach.

W kolejnym kroku należało usunąć kolumny które posiadają więcej niż 30% danych pustych, posłużyła w tym metoda `dropna()` z parametrem `thresh=minimalna ilość wartości non-Null`

W ostatnim kroku zaimplementowano mapowanie pola `sex` z wartościami "female" | "male" na wartości odpowiadające 0 | 1.

```
zad2.py
1 import seaborn as sns
2 import pandas as pd
3
4 titanic_data = sns.load_dataset("titanic")
5 df = pd.DataFrame(titanic_data)
6 ilosc_nan = df.isnull().sum().sum()
7 print("Ilość wartości pustych (NaN) w zbiorze danych:", ilosc_nan)
8
9 ilosc_nan_w_kolumnach = df.isnull().sum()
10 skumulowana_ilosc_nan = ilosc_nan_w_kolumnach.cumsum()
11 print("Ilość wartości pustych (null) w każdej kolumnie:")
12 print(ilosc_nan_w_kolumnach.to_frame().T)
13 print("\nSuma skumulowana ilości wartości pustych (null) w kolumnach:")
14 print(skumulowana_ilosc_nan)
15
16 prog = 0.3 # 30%
17 column_count = str(df.shape[1])
18 minimalna_liczba_wartosci = int((1 - prog) * len(df))
19 df = df.dropna(axis=1, thresh=minimalna_liczba_wartosci)
20 print("\nIlość kolumn przed usunięciem: " + column_count)
21 print("Ilość kolumn po usunięciu: " + str(df.shape[1]))
22
23 mapowanie = {"female": 0, "male": 1}
24 df["sex"] = df["sex"].map(mapowanie)
25 print("\nRamka danych po zamianie danych kategorycznych:")
26 print(df["sex"].head(10).to_frame(name='sex').T)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Ilość wartości pustych (NaN) w zbiorze danych: 869
Ilość wartości pustych (null) w każdej kolumnie:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone	
0	0	0	0	177	0	0	0	embarked	2	0	0	688	embark_town	2	0	0

Suma skumulowana ilości wartości pustych (null) w kolumnach:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	0	0	177	177	177	177	179	179	179	179	867	869	869	869

dtype: int64

Ilość kolumn przed usunięciem: 15
Ilość kolumn po usunięciu: 14

Ramka danych po zamianie danych kategorycznych:

	0	1	2	3	4	5	6	7	8	9
sex	1	0	0	0	1	1	1	1	0	0

Ryc 2. Ukazanie wyników i kodu realizującego zadanie 2

Zadanie 3

Wymagało pobrania użytkownika oraz repozytoriów użytkownika z github, użyto do tego biblioteki requests.

Następnie odbywało się liczenie ilości wystąpień konkretnego języka programowania.

Ostatnim krokiem było utworzenie wykresu kołowego z otrzymanymi danymi - w tym celu użyto bibliotekę pyplot

```
zad3.py
1  import requests
2  import matplotlib.pyplot as plt
3
4  # ----- API -----
5  githubApi = "https://api.github.com"
6  userName = "MikiKru"
7
8  getUserUri = githubApi + "/users/" + userName
9  userResp = requests.get(getUserUri).json()
10
11 getReposUri = githubApi + "/users/" + userName + "/repos"
12 reposResp = requests.get(getReposUri).json()
13 # -----
14 print('Użytkownik -> [' + userResp.get('login') + '] używał następujących języków programowania:')
15
16 language_counts = {}
17
18 for repo in reposResp:
19     language = repo.get("language", "Brak języka")
20     if language in language_counts:
21         language_counts[language] += 1
22     else:
23         language_counts[language] = 1
24
25 sorted_data = dict(sorted(language_counts.items(), key=lambda item: item[1], reverse=True))
26 for language, count in sorted_data.items():
27     print(f"{language} -> {count}")
28
29
30 languages = list(language_counts.keys())
31 counts = list(language_counts.values())
32
33 plt.figure(figsize=(9, 6))
34 plt.pie(counts, labels=languages, autopct='%1.1f%%', startangle=140)
35 plt.axis('equal')
36
37 plt.title( userName + "- wykres języków programowania" )
38
39 plt.show()
40
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

python3 +

```
sex 1 0 0 0 1 1 1 1 0 0
adam-pc@adampc-MS-7A34:~/Documents/mgr/AED-AlgorytmyEksploracjiDanych/lab1$
adam-pc@adampc-MS-7A34:~/Documents/mgr/AED-AlgorytmyEksploracjiDanych/lab1$ python3 zad3.py
Użytkownik -> [MikiKru] używał następujących języków programowania:
Java -> 13
Python -> 4
None -> 4
C++ -> 4
Jupyter Notebook -> 2
TSQL -> 1
C# -> 1
CMake -> 1
```

Ryc 3. Prezentacja kodu i wyników

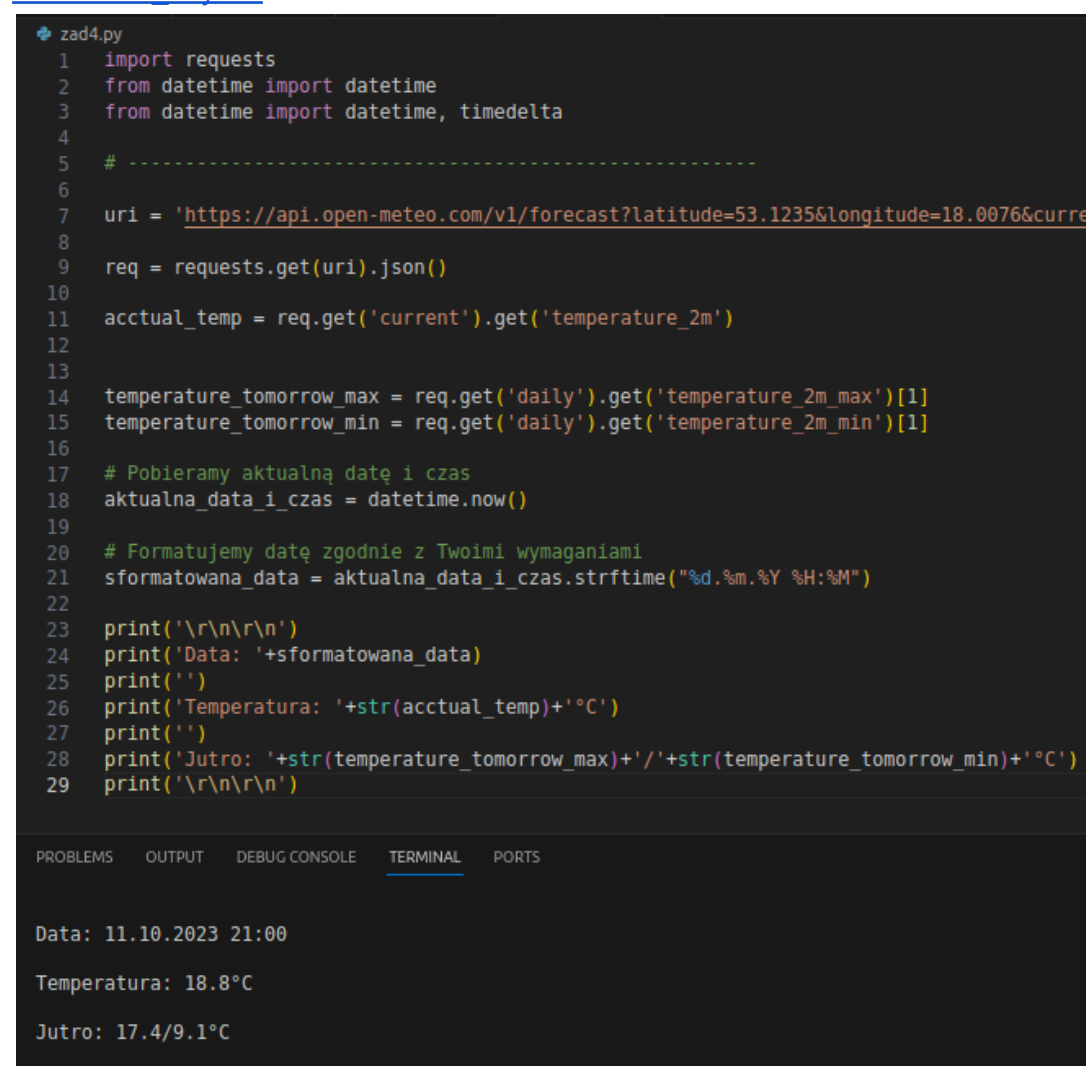
Zadanie 4

Należało pobrać dane odnośnie pogody - aktualna temperatura oraz najwyższa i najniższa temperatura na kolejny dzień.

W query umieszczono lokalizację która nas interesuje (Bydgoszcz, latitude=53.1235 longitude=18.0076), następnie parametr current - aktualna temperatura, daily z parametrami temperature_2m_min oraz temperature_2m_max - zwraca najniższą i najwyższą dzienną temperature, forecast_days - specyfikuje ilość dni.

Precyzyjnie skonstruowane zapytanie pozwoliło otrzymać wszystkie wymagane informacje, w kolejnych krokach pozostało tylko wyciąganie i prezentowanie danych.

https://api.open-meteo.com/v1/forecast?latitude=53.1235&longitude=18.0076¤t=temperature_2m&daily=temperature_2m_max,temperature_2m_min&timezone=Europe%2FBerlin&forecast_days=2



```
zad4.py
1  import requests
2  from datetime import datetime
3  from datetime import datetime, timedelta
4
5  # -----
6
7  uri = 'https://api.open-meteo.com/v1/forecast?latitude=53.1235&longitude=18.0076&current=temperature_2m&daily=temperature_2m_max,temperature_2m_min&timezone=Europe%2FBerlin&forecast_days=2'
8
9  req = requests.get(uri).json()
10
11  actual_temp = req.get('current').get('temperature_2m')
12
13
14  temperature_tomorrow_max = req.get('daily').get('temperature_2m_max')[1]
15  temperature_tomorrow_min = req.get('daily').get('temperature_2m_min')[1]
16
17  # Pobieramy aktualną datę i czas
18  aktualna_data_i_czas = datetime.now()
19
20  # Formatujemy datę zgodnie z Twoimi wymaganiami
21  sformatowana_data = aktualna_data_i_czas.strftime("%d.%m.%Y %H:%M")
22
23  print('\r\n\r\n')
24  print('Data: '+sformatowana_data)
25  print('')
26  print('Temperatura: '+str(actual_temp)+'°C')
27  print('')
28  print('Jutro: '+str(temperature_tomorrow_max)+'/'+str(temperature_tomorrow_min)+'°C')
29  print('\r\n\r\n')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Data: 11.10.2023 21:00
Temperatura: 18.8°C
Jutro: 17.4/9.1°C
```

Ryc 4. Prezentacja kodu i wyników w terminalu.

Wnioski

Podczas tego laboratorium, eksplorowaliśmy publiczne API GitHub, co pozwoliło na zdobycie danych związanych z repozytoriami użytkownika. Przeprowadziliśmy analizę, w której zliczaliśmy ilość repozytoriów napisanych w konkretnym języku programowania. Do komunikacji z API wykorzystaliśmy bibliotekę Requests, która umożliwia pobieranie danych z zewnętrznych źródeł. Następnie wykorzystaliśmy bibliotekę Pandas do przetwarzania i analizy tych danych, co pozwoliło na wyodrębnienie istotnych informacji oraz przygotowanie ich do dalszej analizy.

Ponadto, w trakcie tego laboratorium nauczyliśmy się tworzyć wykresy kołowe przy pomocy biblioteki Matplotlib pyplot. Dzięki temu stworzyliśmy wykres, który wizualizował udział procentowy repozytoriów w różnych językach programowania, co pozwoliło na lepsze zrozumienie preferencji użytkownika w kontekście programowania.

Biblioteki Pythona, takie jak Pandas, Matplotlib i Requests, stanowią potężne narzędzia do przeprowadzania analizy danych i komunikacji z zewnętrznymi źródłami danych. Eksploracja danych jest kluczowym etapem w analizie danych, pozwalającym zrozumieć i przygotować dane przed przystąpieniem do modelowania. Pozwala na zrozumienie charakterystyki zbioru danych i identyfikowanie problemów z brakującymi danymi lub innymi anomaliami.

Repozytorium dostępne pod adresem

<https://github.com/AdamSzz/AED-AlgorytmyEksploracjiDanych/tree/main/lab1>