SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

|  |  |
| --- | --- |
| Laboratorium Nr 1  Data 27.09.2025  Temat: "Ustalenia platformy Jupyter. Użycie biblioteki pandas"  Wariant 9 | Tomasz Dziubiński  Informatyka  II stopień, zaoczne,  1semestr, gr.1A |

1. Polecenie: wariant 1 zadania

Dane do zadania będą pobrane ze strony

http://ghdx.healthdata.org/ihme\_data

która przedstawia sobą repozytorium danych socjoekonomicznych.

One zawierają dane badań statystycznych w zakresie gospodarki, demografii

oraz służby zdrowia.

Dane mogą być przedstawione w postaci plików formatów .csv lub .xslx

Zadanie dotyczy pobrania danych z pliku, tworzenia ramki danych, wykonania

poszczególnych zadań poniżej na podstawie odpowiedniego zbioru danych:

Wariant 9. Premise Women’s Health COVID-19 Health Services Disruption Survey

2020

http://ghdx.healthdata.org/record/ihme-data/premise-women%E2%80%99s-health-covid-19-health-services-disruption-survey-2020

2. Opis programu opracowanego (kody źródłowe, rzuty ekranu)

• ładowanie biblioteki Pandas

import pandas as pd # ładowanie biblioteki Pandas

• tworzenie ramki danych ze słownika

dict\_city = {"City" : ["Warszawa", "Łódź", "Poznań"],

"Population" : [12678079, 5398064, 1625631]}

df = pd.DataFrame(dict\_city) # tworzenie ramki danych ze słownika

df

• zachowanie ramki danych pobranych z pliku w formacie csv (xlsx)

df.to\_csv("example.csv")

• tworzenie ramki danych z listy list

# IHME\_PREM\_WMN\_HEALTH\_2020\_Y2011M10D11.csv

df = pd.read\_csv("IHME\_PREM\_WMN\_HEALTH\_2020\_Y2011M10D11.csv", encoding = "utf-8")

• transponowanie (wymieniamy kolumny a wierszy)

df\_T = df.T # zamienia wiersze z kolumnami

• wyświetlić pierwsze 10 wierszy ramki danych

df.head(10) # pierwsze 10 wierszy ramki danych

• wyświetlić ostatnie 10 wierszy ramki danych

df.tail(10) # ostatnie 10 wierszy ramki danych

• wyświetlić informacje˛ o ramce danych

df.info() # informacja o ramce danych

• wyświetlić, ile wierszy i kolumn znajduje się˛ w ramce danych

df.shape # pokazuje, ile wierszy i kolumn znajduje się w ramce danych

• wyświetlić informacje˛ statystyczna˛ o kolumnach liczbowych (wartości

niepowtarzalne, średnia, odchylenie standardowe, minimum, kwartyle,

maksimum)

df.describe()

• wyświetlić informacje˛ statystyczna˛ o kolumnach kategoryzowanych (ile

unikalnych wartości, top - jaka jest najpopularniejsza wartość, freq -

jak często najpopularniejsza)

df.describe(include = 'all')

• usunąć brakujące wartości w ramce danych

df.dropna(inplace=True) # usuwanie brakujących wartości (NA)

df.dropna(how='all', inplace=True) # usuwa jeśli wszystkie kolumny sa NaN – zastosowane w moim przypadku, gdyż nie byłoby żadnych danych

• przedstawić wybór wierszy i kolumny używając nazw oraz indeksów na

różne sposoby

df.age

df["age"] # wybór kolumny

df[["observation\_id","gender","age"]] # wybór kilku kolumn jednocześnie

df["age"][0] #indeks

df.loc[1:4, "observation\_id":"geography"] # zakres

df.iloc[100:110, 0:3] #zakres2

• przedstawić wybór wierszy z ramki danych pod warunkiem odnośnie

określonej wartości kolumny

df[df["gender"] == "Female"] # wg wartości

• przedstawić wybór wierszy z ramki danych pod warunkiem spełnienia

kilku warunków jednocześnie

df[(df["gender"].str.startswith("F")) & (df["geography"] == "Rural")] # wiecej wartości # and

df[(df["location"] == "Poland") | (df["location"] == "Polska")] # or

• wybrać wiersze które zawierają˛ w kolumnie kategoryzowanej określone

słowo

df[df["financial\_situation"].str.contains("food", case=False, na=False)] # zawiera konkretne slowo

• wybrać wiersze które nie zawierają˛ w kolumnie kategoryzowanej określone

słowo

df[~df["financial\_situation"].str.contains("food", case=False, na=False)] # nie zawiera konkretne slowo

• utwórz kolumnę˛ na podstawie istniejacych

df["education2"] = df["education"]

# utwórz nową kolumnę na podst. obecnej

df["observation\_id2"] = df["observation\_id"].str.split('\_', n=1).str[1]

# utwórz nową kolumnę, która będzie miała zaokrąglone wartości z kolumny val

• usuń kolumnę˛

df.drop("education2", axis=1, inplace = True) # usuń kolumnę education2

• zmień nazwę˛ kolumny

df.rename(columns = {"gender": "gender2"}, inplace = True) # zmień nazwę kolumny

• zachowaj ramkę˛ danych jako plik csv na komputerze

df.to\_csv("lab1result.csv")

df.loc[:10].to\_csv("deaths\_10.csv")

• wyświetlić średnią (maksymalna˛, minimalna˛) wartość z jednej kolumny

df['submitted\_time'].max() # maksymalna wartość jedna kolumna na raz

df['submitted\_time'].min() # min wartość jedna kolumna na raz

• wyświetlić liczbę˛ wierszy

df['observation\_id'].count() # liczba rekordów

• wyświetlić wartości unikatowe w kolumnie

df['geography'].unique() # wartości unikatowe

• wyświetlić liczby rekordów odpowiadających do wartości

df['geography'].value\_counts() # liczba rekordów pasujących do unikalnych wartości

• sortowanie wierszy ramki danych według wartości określonej kolumny

(malejąco, rosnąco)

df.sort\_values(['submitted\_time'], ascending = True) # sortowanie

• wyświetlić wierszy dla 10 największych (najmniejszych) wartości określonej

Kolumny

df.nlargest(10,'wmn\_post\_injectable\_missed') # 10 najwyższych wartości dla kolumny

• wyświetlić wierszy dla 10 największych wartości określonej kolumny

pod warunkiem określonych wartości innej kolumny

df[df['gender'] == 'Female'].nlargest(10, 'wmn\_post\_injectable\_missed') # 10 najwyższych wartości dla kolumny

• grupowanie wierszy według wartości kolumny kategoryzowanej, potem

- uśrednienie wartości wszystkich kolumn w grupie – MultiIndex

grupa = df.groupby("education").mean(numeric\_only=True)

grupa

grupa.columns # multiindeks

• grupowanie wierszy według wartości kolumny kategoryzowanej, potem

- uśrednienie wartości dla pewnych kolumn, liczba wartości i mediana

dla pozostałych kolumn w grupach

df\_t = df.groupby('education').agg({'country': ['count'],

'observation\_id2': ['mean', 'median']})

# dla kolumny 'education' count (liczba unikalnych wartości), dla 'observation\_id2' - średnia i mediana

df\_t

• wyświetlić nazwy kolumn indeksu złożonego

grupa.index # indeksy

grupa.columns # multiindeks

• sortować kolumnę˛ indeksu złożonego

df.sort\_values(['wmn\_post\_injectable\_missed'], ascending = False) # sortowanie

• stworzyć tabele˛ przystawna˛ (pivot table) na podstawie ramki danych

pivot = pd.pivot\_table(

df,

values="geography", # kolumna z wartościami

index="wmn\_safe\_place\_no\_access\_why", # wiersze

columns="wmn\_post\_safe\_place", # kolumny

aggfunc="count" # funkcja agregująca (np. mean, sum, count)

)

Pivot

• wyświetlić indeksy i kolumny tabeli przystawnej

pivot.index

pivot.columns

• utwórz indeks złożony tabeli przystawnej i wyświetl go

pivot2 = df.pivot\_table(values='geography', index=['wmn\_safe\_place\_no\_access\_why','wmn\_post\_injectable\_missed'], columns='wmn\_post\_safe\_place', aggfunc='count',

margins=False, dropna=True, fill\_value=None)

# ustaw multi-indeks

pivot2

• zaimportuj moduł pyplot z biblioteki matplotlib

import matplotlib.pyplot as plt # zaimportuj moduł pyplot z biblioteki matplotlib

• wskazać, że wykresy należy rysować bezpośrednio w zeszycie, a nie w

osobnej zakładce

%matplotlib inline

# wskazanie, że wykresy należy rysować bezpośrednio w zeszycie, a nie w osobnej zakładce

• wyświetlić wykres na podstawie tabeli przystawnej

pivot.plot(kind = 'line')

• narysować histogram na podstawie wartości kolumny

df\_bar = df[(df['country']== 'Poland') & (df['gender2'] == 'Female')].pivot\_table(values='wmn\_pre\_injectable\_missed',

index='geography', columns='gender2', aggfunc='count',

fill\_value=None, margins=False, dropna=True)

df\_bar.plot(kind = 'line')

plt.ylabel('Y')

plt.title('tytul')

• przedstawić sposoby łączenia ramek danych za pomocą˛ metod merge i

Concat

dict\_city = {"City" : ["Warszawa", "Łódź", "Poznań"],

"Population" : [12678079, 5398064, 1625631]}

df1 = pd.DataFrame(dict\_city) # tworzenie ramki danych ze słownika

dict\_city2 = {"City" : ["Warszawa", "Łódź", "Poznań"],

"Population" : [1267807933, 539806433, 162563133]}

df2 = pd.DataFrame(dict\_city2) # tworzenie ramki danych ze słownika

merged\_df = pd.merge(df1, df2, on=['City'], how='inner')

merged\_df

concat\_df = pd.concat([df1, df2], axis=0)

concat\_df

• pokazać dodawanie nowych kolumn za pomoca˛ operacji matematycznych

df\_all = df

df\_all["info"] = df\_all["geography"] + df\_all["financial\_situation"] + df\_all["education"]

df\_all

• przedstawić na przykładzie dodawanie nowych kolumn z pomocą˛ funkcji

Lambda

# tworzymy listę

religions = ['Catholism', 'Muslim']

# za pomocą funkcji lambda określamy, że jeśli religia („religion”) jest zawarty w liście, to nowa kolumna będzie miała postać „sprreligia”

# będzie true, jeśli nie, to false

df\_all['sprreligia'] = df\_all['religion'].apply(lambda x: True if x in religions else False )

df\_all[df\_all['sprreligia'] == True]

• przedstawić możliwości pracy z dużymi plikami przy użyciu argumentu

Chunksize

for chunk\_df in pd.read\_csv('IHME\_PREM\_WMN\_HEALTH\_2020\_Y2011M10D11.csv',

chunksize = 50000):

print("CHUNK DF")

print(chunk\_df.head())

#filtrowanie lambda

# zastosuj metodę groupby oddzielnie do każdej części, a następnie połącz wynik w nową ramkę danych

new\_df = pd.DataFrame() # pusta ramka danych

for chunk\_df in pd.read\_csv('IHME\_PREM\_WMN\_HEALTH\_2020\_Y2011M10D11.csv',

chunksize = 50000):

result = chunk\_df.groupby(['geography', 'age']).agg(

country\_poland=('country', lambda x: (x == 'Poland').sum()),

religion\_count=('religion', 'count')

)

new\_df = pd.concat([new\_df,result])

3. Wnioski

Do tworzenia kodu użyty został program Jupyter Lab oraz język programowania Python wersja 3.

Zaimportowano biblioteki pandas oraz matplotlib moduł pyplot.

Biblioteka pandas jest narzędziem do analizy i przetwarzania danych. Pozwala łatwo manipulować dużymi zbiorami danych w formie tabelarycznej (jak arkusze Excela). Możemy odczytać i zapisać dane z pliku, otrzymać dane statystyczne, filtrować oraz sortować, usuwać oraz dodawać kolumny, łączyć dane oraz grupować wg klucza.

Moduł pyplot z biblioteki **Matplotlib** w Pythonie, używany w **JupyterLab**, służy do **tworzenia wykresów i wizualizacji danych.**

Wczytano plik o nazwie: IHME\_PREM\_WMN\_HEALTH\_2020\_Y2011M10D11.csv z podanej strony w wariancie nr 9.

Pliki z dokumentacją i analizą danych w postaci: lab1.ipynb, lab1.pdf umieszczono na zdalnym publicznym repozytorium:

https://github.com/kurt82/NOD/tree/main/Lab1