Analiza danych w języku Python

Mateusz Zimoch

Co nieco o mnie...

- Mieszkam we Wrocławiu
- CEO/Founder w Trasee / Gallio
- Data Scientist/ Machine Learning Engineer
- Konsultant w Opera Software
- Pasjonat systemów wizyjnych
- Zwycięzca konkursu US Navy na prototyp autonomicznego podwodnego drona
- Z branżą szkoleniową związany od ponad pięciu lat

Kontakt: mateusz@trasee.io, LinkedIN

Co nieco o Was!

- Gdzie mieszkacie i czym się zajmujecie?
- Jakie macie doświadczenie związane z Pythonem i analizą danych?
- Czego chcecie się dowiedzieć na tym kursie?

Organizacja szkolenia

- Możecie przerywać i zadawać pytania w każdej chwili, ~~najlepiej za pomocą mikrofonu~~
- ~~Poza zadawaniem pytań proszę o wyciszenie mikrofonu ze względu na szumy~~
- Hosty
- Slack
- ~~Reakcje~~
- Mówcie mi kiedy za bardzo odpływam ;)
- Przerwy:
  - 15 minut ok. 11:00
  - 50 minut ok. 13:00
  - 15 minut ok. 15:00

Jak będzie wyglądał dzisiejszy dzień?

1. Dlaczego Python?
    - Główne cechy języka
    - Ekosystem do data science
    - Krótkie porównanie z R, MATLAB i innymi językami
2. Python - podstawy
3. Interaktywny zeszyt Jupyter Notebook
    - Interaktywna praca z kodem i danymi
    - ~~Google Colab do pracy w chmurze~~
    - Kluczowe różnice ze środowiskami developerskimi (Pycharm, Visual Studio Code)
4. Pandas - kluczowy pakiet do danych tabelarycznych
    - Wczytywanie danych tabelarycznych (CSV, TSV, XLS)
    - Operacje na kolumnach i wierszach
    - Filtrowanie i procesowanie danych
    - Odczytywanie danych statystycznych
    - Modyfikacja danych z użyciem funkcji Pythonowych
    - Agregacja, podobieństwa do SQL
    - Pandas Profiling - raporty

Wstęp do Pythona

- Napisany przez Guido van Rossum i wydany w 1991 (34 lata temu!)
- Trzy główne wersje, aktualnie najnowsza wersja to 3.13 (07.10.2024r.), release 3.14 w 2025
- Wersja 2.7 przestała być wspierana w 2020.
- Wbrew pozorom nazwa nie pochodzi od węża, a nazwy programu “Latający cyrk Monty Pythona.”
- Napisany jako język skryptowy dla rozproszonego systemu operacyjnego Amoeba, który nie zdobył wielkiej popularności.

Python - najważniejsze cechy

- Interpretowalny - nie trzeba go kompilować oraz posiada własny dedykowany interpreter.
- Wysokopoziomowy - operujesz na pojęciach z domeny problemu, który rozwiązujesz (np. dla banku będą to waluty, kredyty, konta, przelewy) a nie na bitach i rejestrach procesora.
- Dynamicznie typowany - deklarując zmienne nie musisz podawać ich typu oraz możesz w nich trzymać co zechcesz.
- Wspiera wiele stylów (Paradygmatów) programowania: ○ Obiektowy (klasy, metody, dziedziczenie) ○ Funkcyjny (wyrażenia lambda) ○ Imperatywny (pętle) ○ Refleksyjny (zaglądanie do zmiennych, modułów, klas)

Python - najważniejsze cechy

- Używanie wcięć zamiast klamr
- Pojedyncze wyrażenie nie musi kończyć się średnikiem
- Zmienne nie mają deklaracji typu.

```
some_list = [1, "foo", False]

for element in some_list:
    print(element)
```

Porównanie programowania

„Hello World!” w różnych językach

Java

```
public class Hello World {
public static void main(Strings[] args){
System.out.println("Hello World!");
}
}
```

C

```
#include <stdio.h>
int main( )
{
printf("Hello World!");
return 0;
}
```

C++

```
#include <iostream>
using namespace std;
int main( )
{
cout<<"Hello World!";
return 0;
}
```

Python

```
print("Hello World!")
```

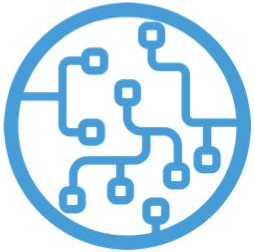
Dlaczego Python?

Python jest prosty

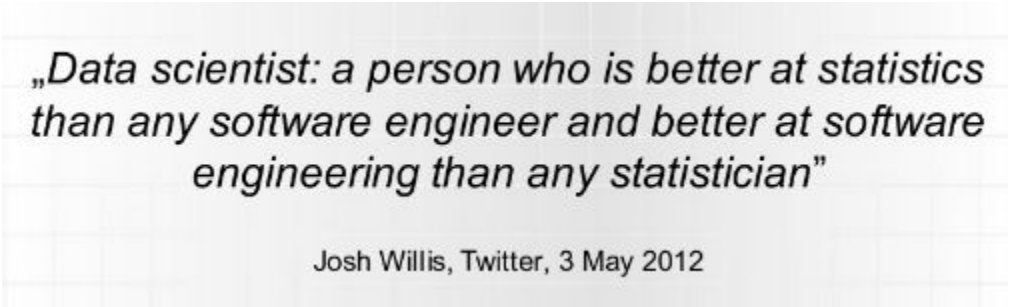
Ma prostą i intuicyjną składnię i słowa kluczowe są zrozumiałe dla każdego początkującego.

Łatwo jest po nim opanować następny język programowania.

Python

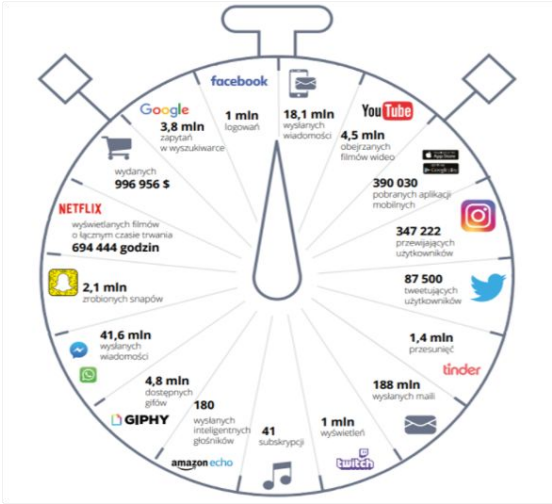


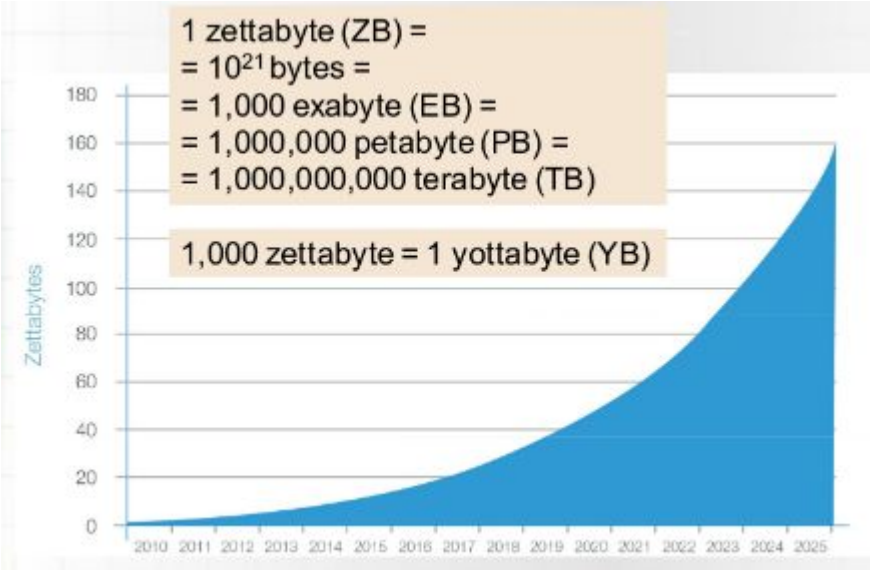
Kim jest analityk danych?



Jak duże ilości danych są w tej chwili produkowane

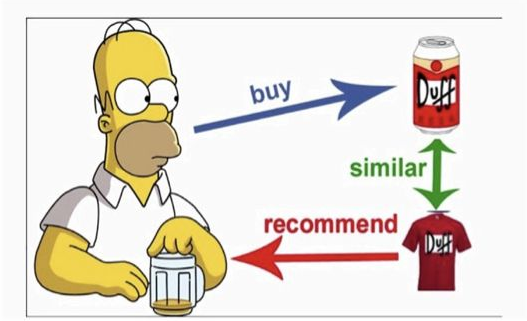
Minuta w świecie danych





Data science,

gdzie się (nie) wykorzystuje?



Google

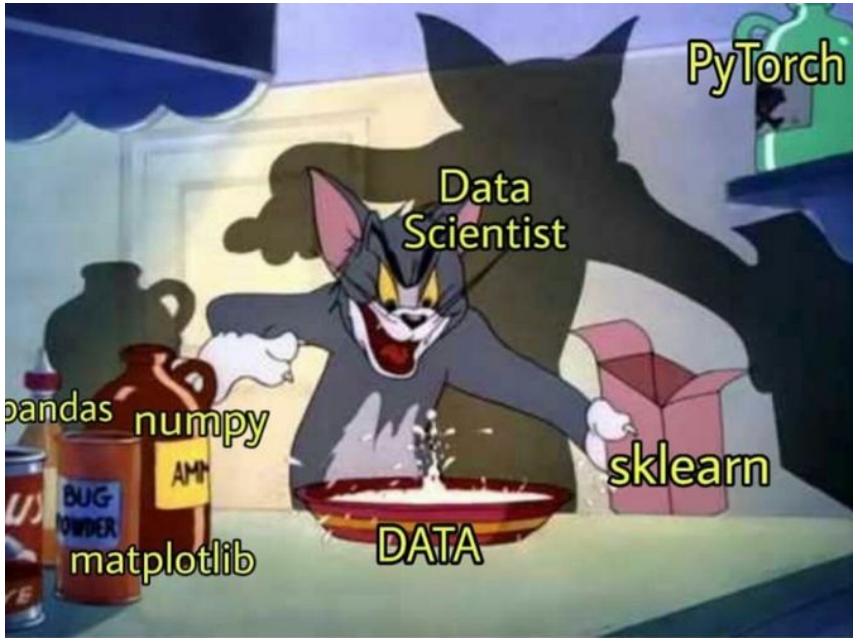
NETFLIX

YouTube

Zadania analityka danych

- znalezienie użytecznych informacji z dostępnych danych lub pozyskanie danych dających użyteczną informację rozwiązującą problem
- radzenie sobie z ogromną ilością danych (przechowywanie, moc obliczeniowa, pozyskanie, itp.)
- łączenie źródeł danych
- wyciąganie tylko koniecznych informacji (czyszczenie danych)
- tworzenie wizualizacji w celu lepszego zrozumienia danych
- budowanie modeli matematycznych do obsługi danych
- prezentacja i przekazywanie wyników

Co jest potrzebne do analizy danych?



Czym są dane

- obraz
- dźwięk
- aktywność w internecie
- zakupy w sklepach
- odwiedzane miejsca
- usługi z których się korzysta
- wiele, wiele innych

Skąd brać dane?

kaggle

Środowisko w jakim będziemy pracować...

Notatniki Jupyter

Notatniki Jupyter, wcześniej znany jako notatnik IPython, zapewnia narzędzia do tworzenia i udostępniania stron internetowych z tekstem (Markdown), wykresami i kodem Pythona w specjalnym formacie.

Często notebooki są używane jako:

- narzędzie edukacyjne,
- demonstracje oprogramowania w języku Python.

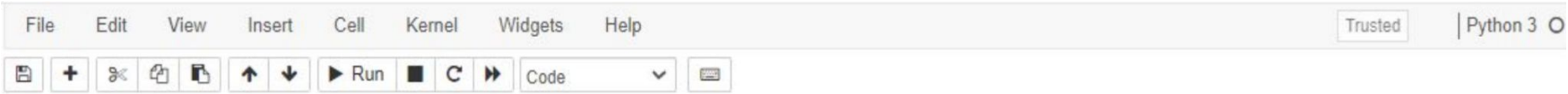
Notatniki możemy importować lub eksportować ze zwykłego kodu Pythona lub ze specjalnego formatu notatnika. Notebooki można uruchamiać lokalnie lub możemy udostępniać je online, uruchamiając dedykowany serwer notebooków

Jupyter Notebook/Lab/Google Colaboratory

- ***Jupyter notebook / Jupyter Lab***
  - interaktywny zeszyt do tworzenia programów, dokumentów, równań i wizualizacji
  - dobry do prototypowania
  - instalacja za pomocą condy/pip-a
  - Jupyter Notebook rozpoczyna pracę po wpisaniu w terminalu komendy jupyter-notebook.
- ***Google Colabolatory***
  - bezpłatna usługa w chmurze Google
  - skonfigurowane środowisko (biblioteki)
  - darmowy dostęp do wydajnej karty graficznej
  - możliwość współpracy z innymi użytkownikami

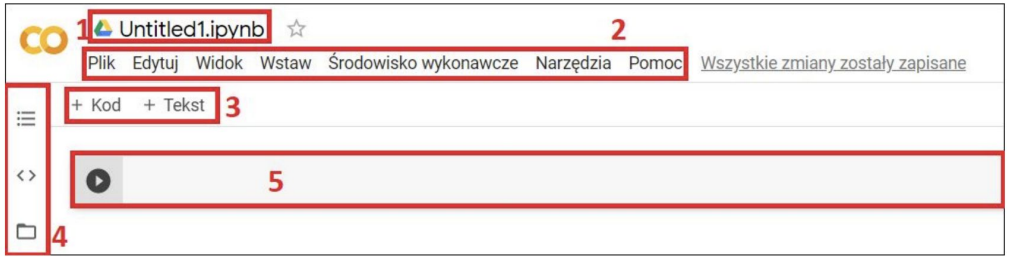
W celu wykorzystania wszystkich funkcji (w szczególności wgrywania i pobierania plików) najlepiej pracować w przeglądarce Google Chrome

Jupyter Notebook - interfejs



- File – w tym menu można stworzyć nowy Notebook, zapisać aktualny, zmienić mu nazwę, zrobić kopię, pobrać do określonego formatu, bądź zrobić tzw. checkpointa, do którego będzie można się później cofnąć, gdy coś pójdzie nie tak.
- Edit – edycja komórek (scalanie, rozłączanie, usuwanie, itd.); niektóre operacje nie będą dostępne dla wybranych rodzajów komórek.
- View – ukrywa dostępne narzędzia (np. umożliwia włączenie/wyłączenie wyświetlania linii kodu czy paska narzędzi).
- Insert – menu pozwalające dodawać nowe komórki powyżej/poniżej aktualnie wybranej.
- Cell – uruchomienie jednej, kilku lub wszystkich komórek naraz; umożliwia zmianę typu komórki na inny.
- Kernel - Menu Kernel służy do pracy z jądrem działającym w tle. Tutaj możemy zrestartować proces, ponownie się z nim połączyć, zamknąć, a nawet zmienić proces, którego używa Notebook (do analizy kodu, obliczeń itp).
- Widgets - Menu Widgets służy do zapisywania i czyszczenia stanu widżetów. Widżety to w zasadzie małe apki napisane w języku JavaScript, które możemy dodać do Jupytera, aby tworzyć dynamiczną zawartość za pomocą Pythona.
- Help – menu pomocy ze skrótami klawiszowymi i odnośnikami do różnych tutoriali.

Google Colab - interfejs



Interfejs Google Colaba jest intuicyjny i nie różni się zbytnio od tego z Jupytera.
1 – tutaj możemy doprecyzować nazwę naszego notebooka
2 – pasek narzędzi
3 – szybkie dodawanie komórek (z kodem lub tekstowych z możliwym formatowaniem)
4 – boczny pasek – zawiera spis treści naszego notebooka, gotowe snippety (fragmenty kodu do wykorzystania i wstawienia) oraz drzewko plików widocznych i dostępnych dla naszego notatnika;
5 – komórka kodu, tutaj możemy pisać nasz kod; kliknięcie na strzałce uruchamia listing. Po kliknięciu na komórce uwidacznia się ponadto pasek w jej prawej górnej części umożliwiający przesunięcie komórki w górę, w dół, usunięcie jej, pobranie linku do niej, dodanie komentarza i inne ustawienia kodu (jak czcionka, długość linii itd.).

Biblioteka pandas

Pandas

Pandas to biblioteka niezbędna do analizy danych w Pythonie. Dostarcza wydajne struktury danych, dzięki którym praca z danymi tabularycznymi staje się prosta i ituicyjna. Celem twórców jest utrzymanie statusu biblioteki niezbędnej do codziennych analiz oraz zdobycie fotela lidera w kategorii najpotężniejszego narzędzia open-source do analizy danych w jakimkolwiek języku programowania. Obecnie, projekt wciąż prężnie się rozwija i jego znajomość jest niezbędna dla każdego analityka danych.

Kiedy stosować Pandas?

Pandas będzie dobrym wyborem do następujących zastosowań:

- Dane tabelaryczne (kolumny jak w SQLu lub Excelu)
- Dane reprezentujące szeregi czasowe
- Macierze,
- Wyniki pomiarów i statystyk.

Mocne strony

Mocnymi stronami Pandas są między innymi:

- Prosta obsługa brakujących wartości (NaN),
- Możliwość modyfikowania rozmiaru DataFrame'a - możemy dodawać i usuwać kolumny i wiersze,
- Automatyczne wyrównywanie danych w obliczeniach (jak w NumPy),
- Metoda groupBy działająca analogicznie jak w SQLu,
- Łatwo stworzyć DataFrame na podstawie innego obiektu,
- Cięcie, indeksowanie i tworzenie podzbiorów,
- Łączenie (join i merge) zbiorów.

Pandas - struktury danych

Series to jednowymiarowa struktura danych (jednowymiarowa macierz numpy), która oprócz danych przechowuje też unikalny indeks.

Tworzenie - *pd.Series(zawartosc)*

```
pd.Series(np.random.random(10))
```

```
0    0.661750
1    0.072319
2    0.758071
3    0.035651
4    0.992312
5    0.488344
6    0.083446
7    0.319852
8    0.419058
9    0.310146
dtype: float64
```

Pandas - struktury danych

Drugą strukturą w pandas jest **DataFrame** (ramka danych) - czyli dwu lub więcej wymiarowa struktura danych, najczęściej w formie tabeli z wierszami i kolumnami. Kolumny mają nazwy, a wiersze mają indeksy.

Tworzenie - pd.DataFrame(zawartosc)

Date	Open	High	Low	Close	Volume	Adj Close
2014-09-16	99.80	101.26	98.89	100.86	66818200	100.86
2014-09-15	102.81	103.05	101.44	101.63	61216500	101.63
2014-09-12	101.21	102.19	101.08	101.66	62626100	101.66
...	...	...	...	...	...	...

Pandas - wczytywanie danych

Pierwszy krok jest zwykle ten sam. Dane są przechowywane w plikach csv, tsv, bazach danych, plikach excel itd. Wczytać je można np. z użyciem funkcji pd.read_csv
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

Najważniejsze argumenty (warto przejrzeć wszystkie, to chyba najważniejsza funkcja pandasa, odpowiednie wczytanie pliku ułatwia pracę na nim):

- ścieżka do pliku
- separator kolumn (domyślnie przecinek)
- nagłówek
- kolumna indeksu
- nazwy kolumn

Pandas - wyświetlanie danych

Po załadowaniu danych kolejnym krokiem jest ich wyświetlenie, można to robić na wiele sposobów:

- df.head(n) - wyświetla pierwsze n rekordów danych
- df.tail(n) - wyświetla końcowe n rekordów danych
- df.sample(n) - wyświetla losowe rekordy danych w liczbie n
- df[“kolumna”] lub df.kolumna - wyświetlenie danej kolumny (jako Series)
- df[[“kolumna”]] - wyświetlenie danej kolumny (jako DataFrame)
- df[[“kolumna1”, “kolumna2”]] - wyświetlenie kilku kolumn
- df.index - wyświetlanie nazw indeksów (wierszy)
- df.columns - wyświetlanie nazw kolumn
- df.info() - ogólne informacje o zbiorze

Pandas - wyświetlanie danych - loc i iloc

Dla dataframe’a występują dwie funkcje do pobierania określonych danych:

- loc szuka po nazwach kolumn i indeksów [[wiersze],[kolumny]]
- iloc po ich numerach porządkowych [[numer wiersza],[numer kolumny]]

Pandas - wyświetlanie danych - warunki

Można również w nawiasach podawać warunek logiczny.

```
print(read.how == 'SEO')  # Series z wartościami True/False według wierszy
display(read[read.how == 'SEO'])  # Wyświetla wszystkie wiersze dla których było True
```

```
time
2018-01-01 00:01:01     True
2018-01-01 00:03:20     True
2018-01-01 00:04:01    False
2018-01-01 00:04:02    False
2018-01-01 00:05:03    False
                       ...
2018-01-01 23:57:14    False
2018-01-01 23:58:33     True
2018-01-01 23:59:36    False
2018-01-01 23:59:36    False
2018-01-01 23:59:38    False
Name: how, Length: 1795, dtype: bool
```

time	status	country	identifier	how	continent
2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2018-01-01 00:08:57	read	country_7	2458151272	SEO	Australia
2018-01-01 00:11:22	read	country_7	2458151276	SEO	North America
2018-01-01 00:13:05	read	country_8	2458151277	SEO	North America

Pandas - nadpisywanie danych

Za pomocą komendy przypisania (=) można też nadpisywać lub dołączać dane:

- df[“kolumna”] = seria_danych
- df[[“kolumna”]] = dataframe
- df[[“kolumna1”, “kolumna2”]] = dataframe
- df.loc[wiersz,kolumna] = dana
- df.iloc[numer_wiersza,numer_kolumny] = dana

Pandas - operacje na danych

Wszystkie poniższe operacje dla dataframe'u zwrócą wartości dla każdej z kolumn, możemy wybierać kolumny dla których chcemy je wykonać:

- df.count() - zliczanie liczby elementów (nie NaNów)
- df.kolumna.value_counts() - zliczanie liczby unikalnych elementów
- df.sum() - suma wszystkich elementów
- df.min() - element minimalny
- df.max() -element maksymalny
- df.mean() - średnia zbioru
- df.median() - mediana zbioru

Zadanie 1

Stwórz dataframe z dziesięcioma imionami uczniów i uczennic oraz liczbą punktów, jakie uzyskali z egzaminu. Następnie sprawdź, jaka była średnia i mediana wyników.

Zadanie 2

Bazując na ramce danych utworzonej w poprzednim zadaniu, wyświetl czwarty wiersz. Następnie zapisz do osobnej ramki tych uczniów, którzy uzyskali wynik powyżej średniej.

Pandas - operacje na danych

Wszystkie poniższe operacje dla dataframe'u zwrócą wartości dla każdej z kolumn, możemy wybierać kolumny dla których chcemy je wykonać:

- df.apply(funkcja) - aplikacja funkcji dla każdej komórki zbioru

```
print(df.apply(lambda x: x.max() - x.min()))
```

```
A    0.863928
B    0.697781
C    1.431593
D    0.000000
F    4.000000
dtype: float64
```

Zadanie 3

Dodaj do ramki trzecią kolumnę, w której znajdzie się procentowy wynik uczniów i uczennic z tego egzaminu (to znaczy, że każda wartość liczbowa musi zostać podzielona przez maksimum punktów, jakie można było na tym egzaminie uzyskać).

Następnie, chcemy zanonimizować kolumnę z imionami: chcemy, by została tylko pierwsza i ostatnia litera imienia. Np. dla imienia “Marcelina” chcemy zachować “M...a”

Pandas - operacje na danych - grupowanie

Od czasu do czasu trzeba wykonać segmentację bazy danych. Oprócz wyznaczania statystyk dla wszystkich wartości, czasem można te wartości pogrupować. W pandasie służy do tego metoda groupby.

- df.groupby(“kolumna”).operacja().kolumna(y)

```
display(cars.groupby('cylinders').mean().horsepower)
```

```
cylinders
3     99.250000
4     78.281407
5     82.333333
6    101.506024
8    158.300971
Name: horsepower, dtype: float64
```

Pandas - usuwanie danych

Do usuwania danych służy nam komenda df.drop() - jako parametr dajemy spis indexów lub kolumn do usunięcia oraz axis - czy ma być usunięty index czy kolumna.

Możemy również usuwać dane niepełne (NaNy) komendą df.dropna() - parametr axis usuwa kolumny lub wiersze.

```
wine = wine.drop(wine.columns[[0,3,6,8,11,12,13]], axis = 1)

wine.head()
```

	Malic acid	Ash	Magnesium	Total phenols	Nonflavanoid phenols	Color intensity	Hue
0	1.71	2.43	127	2.80	0.28	5.64	1.04
1	1.78	2.14	100	2.65	0.26	4.38	1.05
2	2.36	2.67	101	2.80	0.30	5.68	1.03
3	1.95	2.50	113	3.85	0.24	7.80	0.86
4	2.59	2.87	118	2.80	0.39	4.32	1.04

Pandas - NaNy

- metoda df.fillna(wartość) - wypełnia dane określoną wartością
- metoda df.dropna() - usuwa wiersze z pustymi danymi z tabeli

Pandas - łączenie danych

W rzeczywistości często zamiast korzystać z jednej dużej bazy, lecz łączymy wiele mniejszych (łatwiej jest nimi zarządzać, unikać redundancji, dodatkowo oszczędzamy miejsce na dysku i osiągamy większą szybkość). Dane w bibliotece możemy łączyć na dwa sposoby:

- metoda pd.concat([df, series/df]) dodaje nowe wiersze na końcu istniejącego dataframe'a
- metoda df.merge(), która w swoich założeniach jest bardzo podobna do SQL-owego JOINa - można wybrać metodę łączenia - inner (część wspólna), outer (suma), left, right. Parametr on - nazwa kolumny, która ma być łącznikiem.

Dodatkowo, możemy sami podać nową nazwę kolumny i w ten sposób dodać dane do istniejących danych: df["nowa kolumna"] = dane

Pandas - sortowanie danych

Metoda df.sort_values(by=”nazwa kolumny”)

- parametr ascending - czy rosnąco czy malejąco
- często po sortowaniu chcemy zresetować indexy - służy do tego metoda df.reset_index(drop=True) - parametr ten usuwa stary index

Zadanie 4

Posortuj ramkę danych względem liczby punktów otrzymanych z egzaminu, od najwyższego do najniższego. W przypadku takiej samej liczby punktów, osoby mają zostać wyświetlone w kolejności alfabetycznej.

Zadanie 5

Połącz ze sobą dwie poniższe ramki:

student_data1 = pd.DataFrame({ 'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'], 'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed Bernal', 'Kwame Morin'] })

student_data2 = pd.DataFrame({ 'student_id': ['S6', 'S7', 'S8', 'S9', 'S10'], 'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser William', 'Madeeha Preston'] })

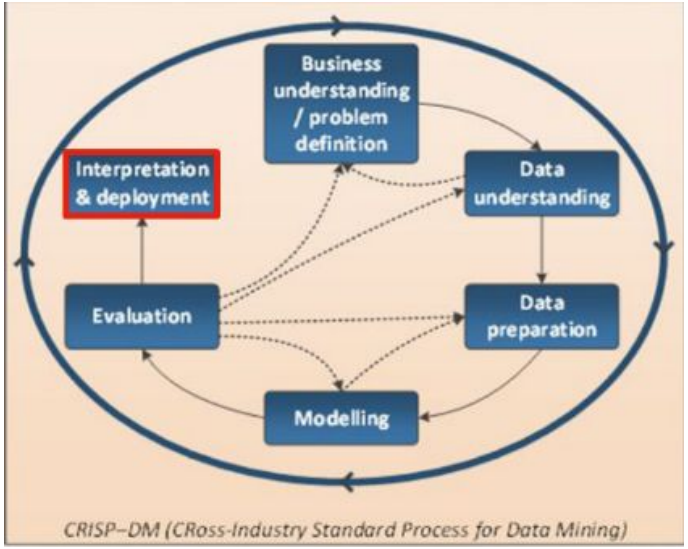
Zadanie 6

Do ramki utworzonej w poprzednim zadaniu dodaj informacje o rezultatach studentów z ramki poniżej:

exam_results = pd.DataFrame({ 'student_id': ['S2', 'S10', 'S3', 'S1', 'S7', 'S9', 'S5', 'S4', 'S8', 'S6'], 'marks': [200, 210, 190, 222, 199, 201, 200, 198, 219, 201] })

Proces analizy danych

Schemat procesu analizy danych



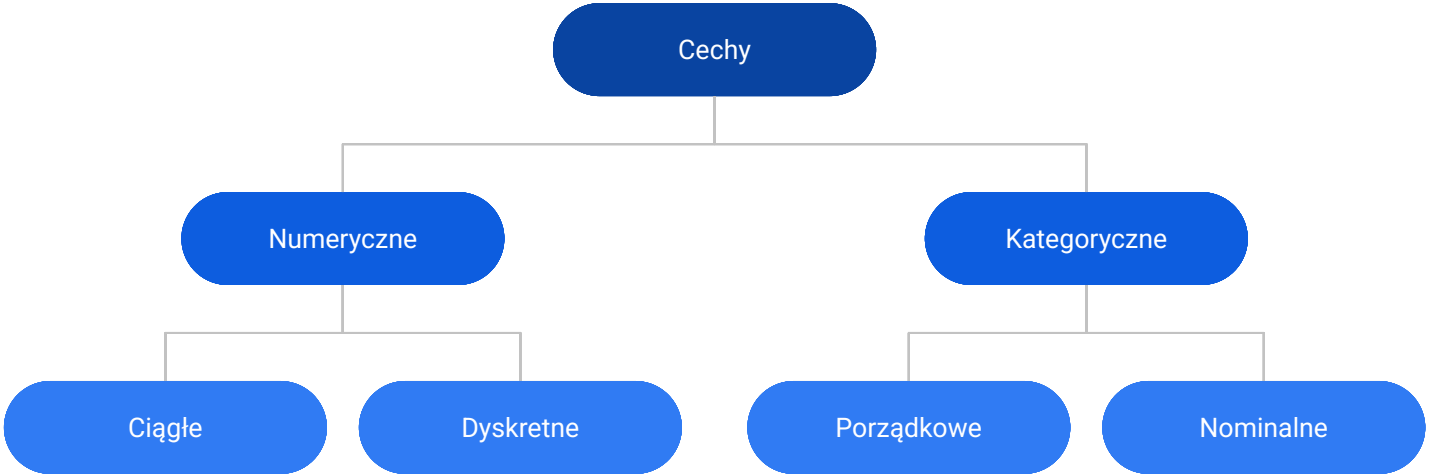
Schemat procesu analizy danych

Data understanding - zebranie określonych danych potrzebnych do rozwiązania problemu, opis danych (czasem również etykietyzacja), eksloracja i weryfikacja jakości

Typy danych - arkusze CSV (na takich będziemy pracować), tekst, obraz, dźwięk, wideo, dane pogodowe, giełdowe, archiwa publiczne itd.

Najważniejsze zadanie - EKSTRAKCJA CECH (feature engineering)

Cechy - podział



Metody agregacji cech

Numerycznych:

- średnia
- suma,
- max, min, odchylenie standardowe

Kategorycznych:

- liczenie wystąpień
- szukanie najczęstszych
- wyniki procentowe np. 23% populacji to blondyni

Data preparation - przygotowanie danych

Wartości niekompletne - jak sobie z tym poradzić:

- ignorować
- wstawić wartość “nieznana”
- ręcznie uzupełniać na podstawie zgromadzonej wiedzy
- usuwać rekordy z niekompletnymi danymi
- uzupełniać algorytmicznie - poszukiwać najbliższego sąsiada, wyciągać średnią, wstawiać wartości losowe, zadania klasyfikacji

Data preparation - przygotowanie danych

```
threshold = 0.7


#Dropping columns with missing value rate higher than threshold
data = data[data.columns[data.isnull().mean() < threshold]]

#Dropping rows with missing value rate higher than threshold
data = data.loc[data.isnull().mean(axis=1) < threshold]
```

Data preparation - przygotowanie danych

Wprowadzanie nowych wartości w miejsce NaNów

```
#Filling all missing values with 0
data = data.fillna(0)


#Filling missing values with medians of the columns
data = data.fillna(data.median())
```

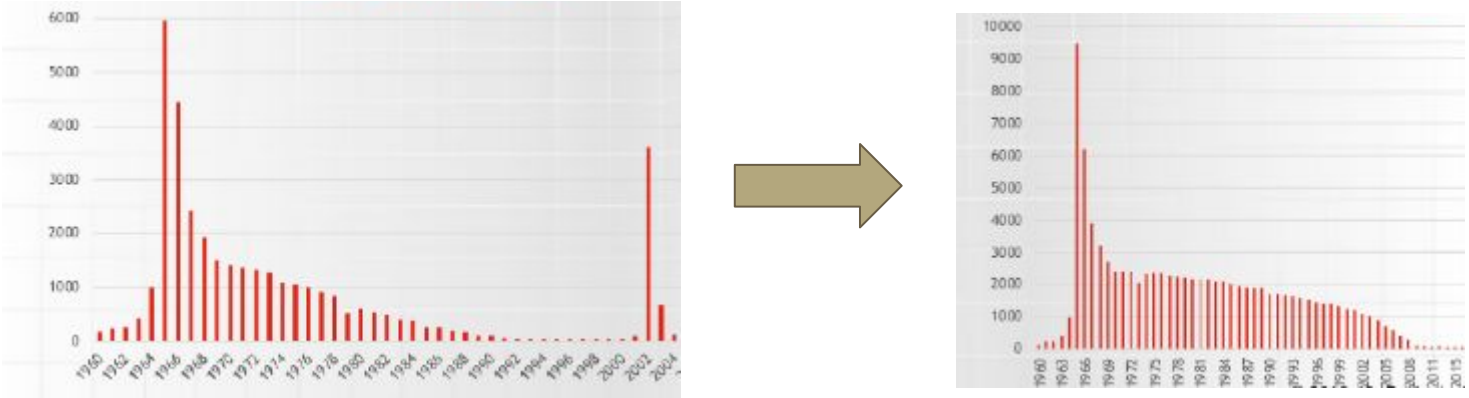
Data preparation - przygotowanie danych

Wprowadzanie nowych wartości w miejsce NaNów - cechy kategoryczne

```
#Max fill function for categorical columns
data['column_name'].fillna(data['column_name'].value_counts()
.idxmax(), inplace=True)
```

Data preparation - przygotowanie danych

Czyszczenie danych - usuwanie błędów, szumów, wartości odstających



Data preparation - przygotowanie danych

```
#Dropping the outlier rows with standard deviation
factor = 3
upper_lim = data['column'].mean () + data['column'].std () *
factor
lower_lim = data['column'].mean () - data['column'].std () *
factor

data = data[(data['column'] < upper_lim) & (data['column'] >
lower_lim)]
```

Data preparation - przygotowanie danych

```
#Dropping the outlier rows with Percentiles
upper_lim = data['column'].quantile(.95)
lower_lim = data['column'].quantile(.05)

data = data[(data['column'] < upper_lim) & (data['column'] >
lower_lim)]
```

Data preparation - przygotowanie danych

Usuwać wartości odstające czy je ograniczać?

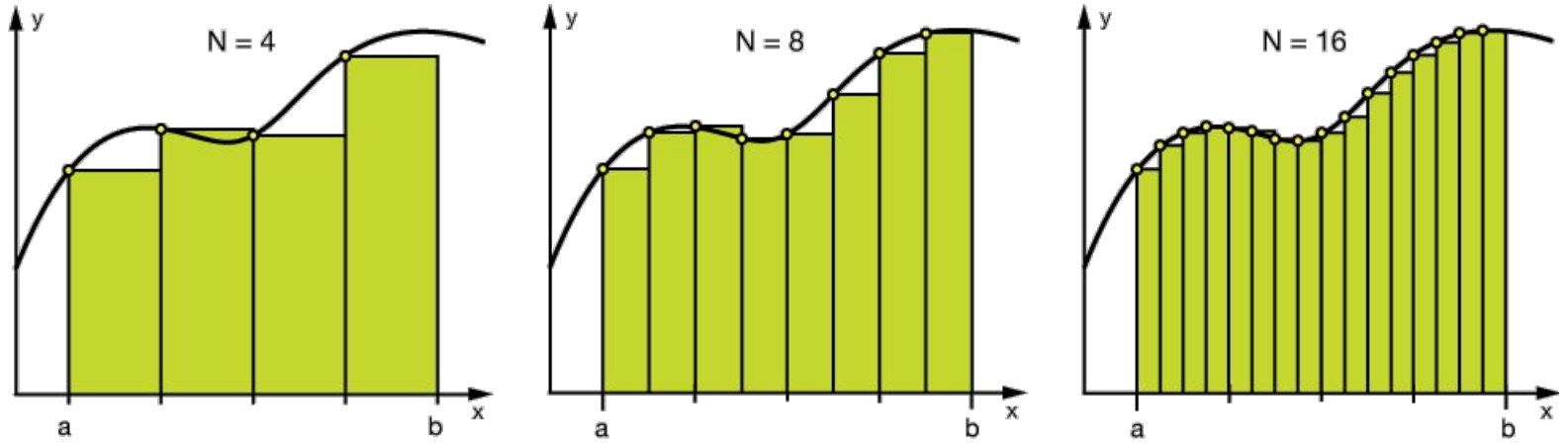
```
#Capping the outlier rows with Percentiles
upper_lim = data['column'].quantile(.95)
lower_lim = data['column'].quantile(.05)


data.loc[(df[column] > upper_lim),column] = upper_lim
data.loc[(df[column] < lower_lim),column] = lower_lim
```

Data preparation - przygotowanie danych

Standaryzacja:

- styl kodowania języka: UTF-8, ANSI itp.
- daty: 2019-09-12, 12 wrz 2019, 120919 itp.
- Zielona Góra, ZG, Z. Góra
- AGH, Akademia Górniczo Hutnicza
- Nierozpoznawalne znaki

Dyskretyzacja (binning)



Dyskretyzacja (binning)

```
#Numerical Binning Example

Value      Bin
0-30   ->  Low
31-70  ->  Mid
71-100 ->  High

#Categorical Binning Example

Value      Bin
Spain  ->  Europe
Italy  ->  Europe
Chile  ->  South America
Brazil ->  South America
```

Tranformacja logarytmiczna

- pomaga poradzić sobie z wypaczonymi danymi
- porządkuje wielkość danych
- zmniejsza wpływ wartości odstających
- zwiększa niezawodność modelu
- trzeba zająć się wartościami ujemnymi

```
#Log Transform Example
data = pd.DataFrame({'value':[2,45, -23, 85, 28, 2, 35, -12]})

data['log+1'] = (data['value']+1).transform(np.log)

#Negative Values Handling
#Note that the values are different
data['log'] = (data['value']-data['value'].min()+1)
.transform(np.log)
```

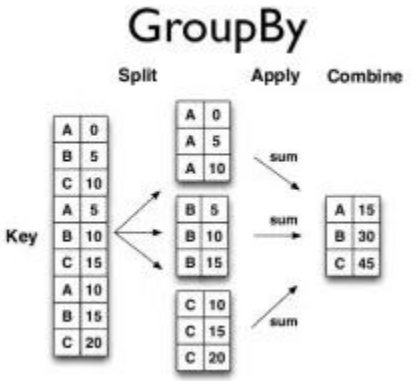
	value	log(x+1)	log(x-min(x)+1)
0	2	1.09861	3.25810
1	45	3.82864	4.23411
2	-23	nan	0.00000
3	85	4.45435	4.69135
4	28	3.36730	3.95124
5	2	1.09861	3.25810
6	35	3.58352	4.07754
7	-12	nan	2.48491

One - hot encoding (kodowanie 1 z n)

User	City
1	Roma
2	Madrid
1	Madrid
3	Istanbul
2	Istanbul
1	Istanbul
1	Roma

User	Istanbul	Madrid
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

Grupowanie


GroupBy
Split
Apply
Combine
Key
A 0
B 5
C 10
A 5
B 10
C 15
A 10
B 15
C 20
A 0
A 5
A 10
B 5
B 10
B 15
C 10
C 15
C 20
sum
sum
sum
A 15
B 30
C 45

Dzielenie cech

```
data.name
0  Luther N. Gonzalez
1    Charles M. Young
2        Terry Lawson
3       Kristen White
4      Thomas Logsdon

#Extracting first names
data.name.str.split(" ").map(lambda x: x[0])
0     Luther
1    Charles
2      Terry
3    Kristen
4     Thomas

#Extracting last names
data.name.str.split(" ").map(lambda x: x[-1])
0    Gonzalez
1       Young
2      Lawson
3       White
4     Logsdon
```

Skalowanie

W większości przypadków cechy numeryczne zestawu danych nie mają określonego zakresu i różnią się od siebie. W rzeczywistości nie ma sensu oczekiwać, że kolumny wieku i dochodów będą miały ten sam zakres. Ale z punktu widzenia uczenia maszynowego, ten sam zakres pomaga usprawnić model

Skalowanie rozwiązuje ten problem. Funkcje ciągłe stają się identyczne pod względem zakresu po procesie skalowania. Ten proces nie jest obowiązkowy w przypadku wielu algorytmów, ale nadal warto go zastosować. Jednak algorytmy oparte na obliczeniach odległości, takich jak k-NN lub k-Means, muszą mieć skalowane ciągłe funkcje jako dane wejściowe modelu.

Zasadniczo istnieją dwa popularne sposoby skalowania

Normalizacja

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
data = pd.DataFrame({'value':[2,45, -23, 85, 28, 2, 35, -12]})

data['normalized'] = (data['value'] - data['value'].min()) /
(data['value'].max() - data['value'].min())
```

	value	normalized
0	2	0.23
1	45	0.63
2	-23	0.00
3	85	1.00
4	28	0.47
5	2	0.23
6	35	0.54
7	-12	0.10

Standaryzacja

$$z = \frac{x - \mu}{\sigma}$$

```
data = pd.DataFrame({'value':[2,45, -23, 85, 28, 2, 35, -12]})

data['standardized'] = (data['value'] - data['value'].mean()) /
data['value'].std()
```

	value	standardized
0	2	-0.52
1	45	0.70
2	-23	-1.23
3	85	1.84
4	28	0.22
5	2	-0.52
6	35	0.42
7	-12	-0.92

Wyodrębnianie daty

```
from datetime import date

data = pd.DataFrame({'date':
['01-01-2017',
'04-12-2008',
'23-06-1988',
'25-08-1999',
'20-02-1993',
]})

#Transform string to date
data['date'] = pd.to_datetime(data.date, format="%d-%m-%Y")

#Extracting Year
data['year'] = data['date'].dt.year

#Extracting Month
data['month'] = data['date'].dt.month

#Extracting passed years since the date
data['passed_years'] = date.today().year - data['date'].dt.year

#Extracting passed months since the date
data['passed_months'] = (date.today().year - data['date'].dt.year)
* 12 + date.today().month - data['date'].dt.month

#Extracting the weekday name of the date
data['day_name'] = data['date'].dt.day_name()
```

	date	year	month	passed_years	passed_months	day_name
0	2017-01-01	2017	1	2	26	Sunday
1	2008-12-04	2008	12	11	123	Thursday
2	1988-06-23	1988	6	31	369	Thursday
3	1999-08-25	1999	8	20	235	Wednesday
4	1993-02-20	1993	2	26	313	Saturday

Czy wiesz że?

Proces pozyskania, preprocessingu danych i wydobycia z nich cech zajmuje nawet 80% czasu pracy inżyniera danych

Pandas Profiling

Generuje automatyczny raport na podstawie dostarczonej ramki danych umożliwiający przeprowadzenie EDA (Exploratory Data Analysis) Dla każdej kolumny obliczane są podstawowe statystyki, a wszystko przedstawione jest w formie interaktywnego HTMLowego raportu:

- typ kolumny
- unikalne i brakujące wartości
- kwantyle, minima, maksima, zakres
- średnia, odchylenie standardowe
- najczęściej występujące wartości
- histogram i wiele wiele innych!

Jak będzie wyglądał ten dzień?

1. NumPy - biblioteka do obliczeń numerycznych
    - Podstawy obliczeń numerycznych
    - Porównanie szybkości z czystym Pythonem
    - Kluczowe różnice pomiędzy tablicami Numpy i listami Pythonowymi
    - Typowe operacje na wektorach i macierzach
    - Algebra liniowa
2. Wizualizacja danych - hasłowo
    - ~~Eksploracja danych~~
    - ~~Profesjonalne wykresy w ramach Pandas~~
    - ~~Fragmenty Matplotlib - bazowej biblioteki do wykresów w Pythonie~~
    - ~~Seaborn - nieco bardziej estetyczne wykresy~~
    - ~~Plotly - tworzenie interaktywnych wykresów~~
    - ~~Plotly Dash - interaktywne dashboardy~~

Biblioteka NumPy

NumPy

NumPy jest jedną z podstawowych bibliotek wykorzystywanych w analizie danych w Pythonie. Zaletą NumPy jest wsparcie dla dużych, wielowymiarowych tablic i macierzy oraz mnóstwo zaimplementowanych funkcji matematycznych.

NumPy rozwiązuje również problem wydajności standardowych operacji matematycznych w Pythonie. Został zaimplementowany w C i Fortran, dzięki czemu efektywność operacji została przeniesiona na inny poziom. Wydajność operacji w numpy jest zbliżona do tej z MATLAB, ale osiągamy ją pracując w środowisku pełnoprawnego języka programowania.

Podstawy wizualizacji biblioteka Matplotlib

Jeden obraz wart więcej niż tysiąc słów.

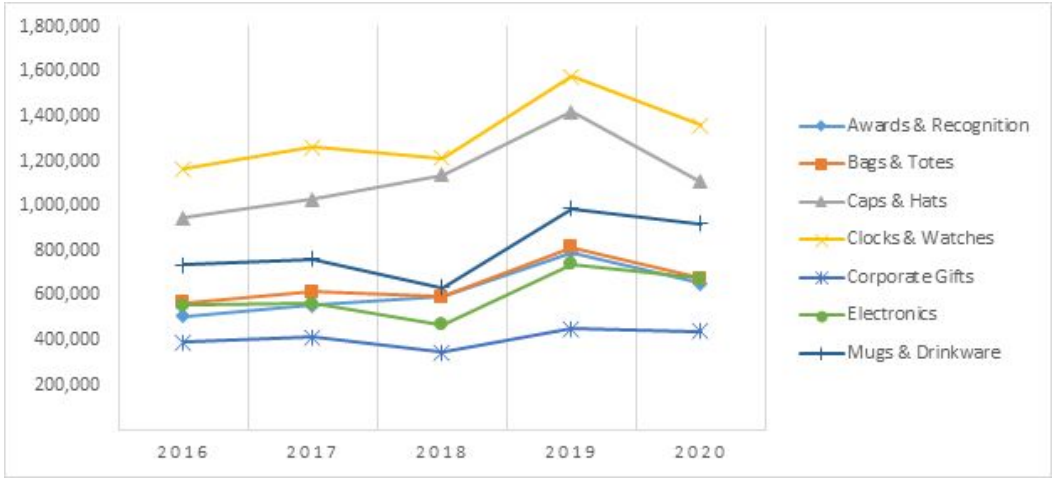
Matplotlib

- biblioteka do tworzenia dwu- i trójwymiarowych wykresów
- interfejs zbliżony do MATLABa

- ok. 70 tys. linii kodu
- moduł **pyplot**
- strona domowa z dokumentacją: https://matplotlib.org/index.html

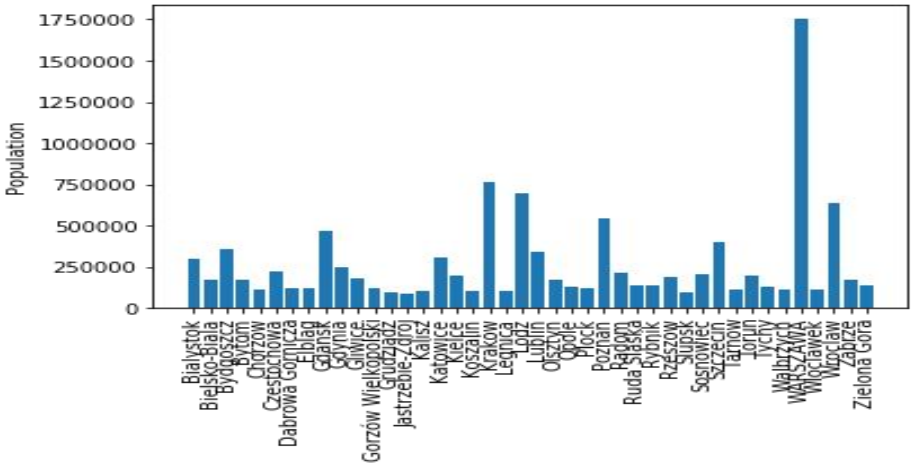
Wybór właściwego wykresu

Wykres liniowy - dane prezentowane w postaci znaczników połączonych liniami prostymi; wizualizacja zmian wartości w kolejnych przedziałach czasowych



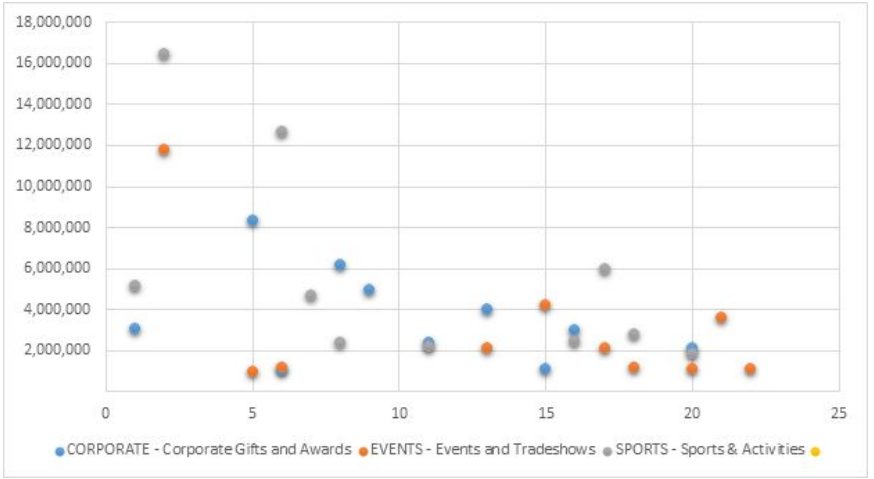
Wybór właściwego wykresu

Wykres słupkowy - zgrupowane dane w formie prostokątnych słupków o wysokości proporcjonalnej do długości. Wykorzystywany np. w celu porównania danych obejmujących wiele kategorii.



Wybór właściwego wykresu

Wykres punktowy - wyświetla dwie zmienne z zestawu danych przy użyciu współrzędnych kartezjańskich (lub trzech, gdy uwzględnimy zmienne kolory).



Seaborn

Na oficjalnej stronie projektu możemy przeczytać opis:

If matplotlib “tries to make easy things easy and hard things possible”, seaborn tries to make a well-defined set of hard things easy too.

Plotly

Plotly

https://plotly.com/python/

Plotly is a technical computing company headquartered in Montreal, Quebec, that develops online data analytics and visualization tools. Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST.

Jak będzie wyglądał ten dzień?

1. Dane tekstowe i API
    - Proste operacje tekstowe
    - Łączenie z API i pobieranie danych
    - Pozyskiwanie ze stron (requests)
    - Parsowanie stron internetowych (BeautifulSoup)
    - Przetwarzanie i zapisywanie
2. Inne
    - Podstawy SQLalchemy
    - Propozycje od grupy :)

Biblioteki, które poznamy

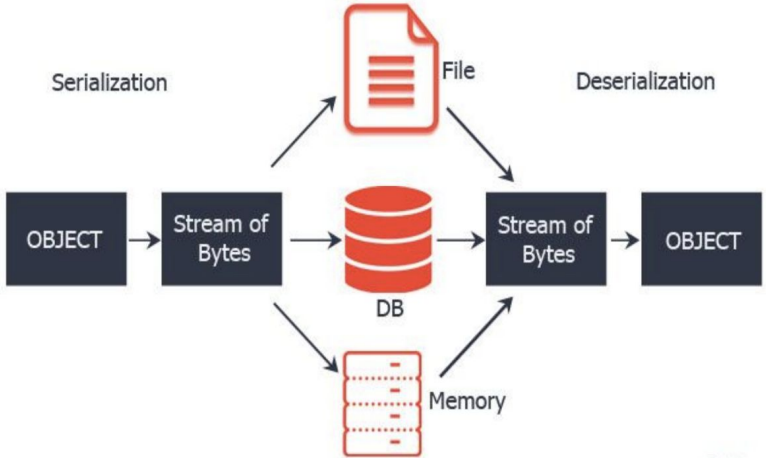
1. Pickle
2. Requests
3. Pretty printer
4. Beautiful soup
5. Re

Serializacja

Proces przekształcania obiektów, tj. instancji określonych klas, do postaci szeregowej, czyli w strumień bajtów, z zachowaniem aktualnego stanu obiektu.

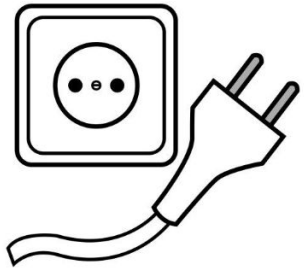
Serializowany obiekt może zostać utrwalony w pliku dyskowym, przesłany do innego procesu lub innego komputera poprzez sieć. Procesem odwrotnym do serializacji jest deserializacja.

Biblioteka pickle



API

Skrót **API (Application Programming Interface)** oznacza interfejs przeznaczony dla programistów tworzących aplikacje. Większość dużych firm tworzy takie interfejsy dla swoich klientów, a także do użytku wewnętrznego. Zapewne korzystałeś z API, nawet o tym nie wiedząc, na przykład komentując post na Facebooku na stronie internetowej innej firmy, płacąc za modne różowe buty w ekskluzywnym sklepie internetowym przy użyciu konta PayPal, albo nawet używając wyszukiwarki na stronie sklepu lub korzystając z Google Maps, aby sprawdzić jego lokalizację



Co można pobierać z API?

API można używać przede wszystkim do pobierania aktualnych i historycznych informacji, na przykład o:

- pogodzie
- natężeniu ruchu
- lotach
- najnowszych wiadomościach
- twittach
- mapach
- filmach
- wiele, wiele innych

Trzeba pamiętać o tym, że większość dobrych API jest płatna, ewentualnie posiadają darmową wersję testową/demo. Zawsze musimy posiadać wygenerowany dla nas klucz API.

Co można pobierać z API?

Użyjemy serwisu OpenWeatherAPI: https://openweathermap.org

Aby uzyskać klucz trzeba się zarejestrować - klucz powinien przyjść na maila. W ramach darmowej wersji możemy wykonywać 60 zapytań na minutę, sprawdzać aktualną pogodę oraz jej prognozy do 7 dni dla całego świata. Dane są zwracane jako JSON lub XML.

Do API jest załączona bardzo dobra dokumentacja: https://openweathermap.org/current

JSON

JSON (ang. JavaScript Object Notation) to otwarty format zapisu struktur danych. Jego przeznaczeniem jest najczęściej wymiana danych pomiędzy aplikacjami. JSON składa się z par atrybut – wartość oraz typów danych tablicowych. Notacja JSONa jest zbieżna z obiektami w języku JavaScript. Jego zaletą jest popularność, prostota działania, zwięzłość syntaktyki, a jako że dane są zapisywane do tekstu – po sformatowaniu czytelne dla ludzi. JSON może być alternatywą dla XML lub CSV. Pliki JSON zapisujemy z rozszerzeniem .json. Logo JSON to torus Mobiusa.

```
{
"title" : "This Is What You Came For",
"artist" : "Calvin Harris",
"length" : "3:41",
"released" : "2016.04.29"
}
```

XML

XML (ang. Extensible Markup Language, w wolnym tłumaczeniu Rozszerzalny Język Znaczników) – uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. Obecnie zastępowany przez JSONa

```
<?xml version="1.0" encoding="iso-8859-8" standalone="yes" ?>
<CURRENCIES>
  <LAST_UPDATE>2004-07-29</LAST_UPDATE>
  <CURRENCY>
    <NAME>dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>4.527</RATE>
    <CHANGE>0.044</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>euro</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>EUR</CURRENCYCODE>
    <COUNTRY>European Monetary Union</COUNTRY>
    <RATE>5.4417</RATE>
    <CHANGE>-0.013</CHANGE>
  </CURRENCY>
</CURRENCIES>
```

RegEx

Wyrażenia regularne (RegEx - ang. Regular Expressions) to ciąg znaków, który definiuje wzorzec wyszukiwania. W Pythonie RegEx obsługuje wbudowana biblioteka re. Jej podstawowe funkcje:

- findall() - zwraca listę zawierającą wszystkie dopasowania
- search() - zwraca obiekt match, jeśli istnieje dopasowanie w dowolnym miejscu ciągu znaków.
- split() - zwraca listę, w której ciąg znaków został podzielony po każdym dopasowaniu.
- sub() - zastępuje jedno lub wiele dopasowań ciągiem znaków

https://www.w3schools.com/python/python_regex.asp

Dziękuję za uwagę

Mateusz Zimoch