Import wymaganych pakietów

In [1]:
```
import numpy as np
import pandas as pd
```

Wczytanie pliku

In [2]:
```
df = pd.read_csv("Zbiór danych Titanic.arff.txt", header = 0, na_values = "?")
df.head(20)
```

Out[2]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	NaN	135.0	Montreal, PQ / Chesterville, ON
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	NaN	NaN	Montreal, PQ / Chesterville, ON
5	1	1	Anderson, Mr. Harry	male	48.0000	0	0	19952	26.5500	E12	S	3	NaN	New York, NY
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0000	1	0	13502	77.9583	D7	S	10	NaN	Hudson, NY
7	1	0	Andrews, Mr. Thomas Jr	male	39.0000	0	0	112050	0.0000	A36	S	NaN	NaN	Belfast, NI
8	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53.0000	2	0	11769	51.4792	C101	S	D	NaN	Bayside, Queens, NY
9	1	0	Artagaveytia, Mr. Ramon	male	71.0000	0	0	PC 17609	49.5042	NaN	C	NaN	22.0	Montevideo, Uruguay
10	1	0	Astor, Col. John Jacob	male	47.0000	1	0	PC 17757	227.5250	C62 C64	C	NaN	124.0	New York, NY
11	1	1	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female	18.0000	1	0	PC 17757	227.5250	C62 C64	C	4	NaN	New York, NY
12	1	1	Aubart, Mme. Leontine Pauline	female	24.0000	0	0	PC 17477	69.3000	B35	C	9	NaN	Paris, France
13	1	1	Barber, Miss. Ellen 'Nellie'	female	26.0000	0	0	19877	78.8500	NaN	S	6	NaN	NaN
14	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0000	0	0	27042	30.0000	A23	S	B	NaN	Hessle, Yorks
15	1	0	Baumann, Mr. John D	male	NaN	0	0	PC 17318	25.9250	NaN	S	NaN	NaN	New York, NY
16	1	0	Baxter, Mr. Quigg Edmond	male	24.0000	0	1	PC 17558	247.5208	B58 B60	C	NaN	NaN	Montreal, PQ
17	1	1	Baxter, Mrs. James (Helene DeLaudeniere Chaput)	female	50.0000	0	1	PC 17558	247.5208	B58 B60	C	6	NaN	Montreal, PQ
18	1	1	Bazzani, Miss. Albina	female	32.0000	0	0	11813	76.2917	D15	C	8	NaN	NaN
19	1	0	Beattie, Mr. Thomson	male	36.0000	0	0	13050	75.2417	C6	C	A	NaN	Winnipeg, MN

Ilość wartości brakujących w danych kolumnach:

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
pclass          0
survived        0
name            0
sex             0
age           263
sibsp           0
parch           0
ticket          0
fare            1
cabin        1014
embarked        2
boat          823
body         1188
home.dest     564
dtype: int64
```

Procentowy udział wartości brakujących:

In [4]:
```
df.isnull().mean() * 100
```

Out[4]:
```
pclass        0.000000
survived      0.000000
name          0.000000
sex           0.000000
age          20.091673
sibsp         0.000000
parch         0.000000
ticket        0.000000
fare          0.076394
cabin        77.463713
embarked      0.152788
boat         62.872422
body         90.756303
home.dest    43.086325
dtype: float64
```

Wartości brakujące mają widoczny udział dla kolumn: **age**, **cabin**, **boat**, **body**, **home.dest**

Ilość cech:

In [5]:
```
len(df.columns)
```

Out[5]:
```
14
```

Zbiór zawiera 14 cech:

- **pclass** - klasa, którą podrożował dany pasażer
- **survived** - 0 = zginął w katastrofie; 1 = przeżył
- **name** - dane osobowe pasażera
- **sex** - płeć
- **age** - wiek
- **sibsp** - ilość rodzeństwa lub małżonków danego pasażera na pogładzie
- **parch** - ilość rodziców i dzieci danego pasażera na pokładzie
- **ticket** - nr biletu
- **fare** - opłata za podróż
- **cabin** - nr kabiny
- **embarked** - port, w którym pasażer wsiadł (S - Southampton, C - Cherbourg, Q - Queenstown)
- **boat** - oznaczenie łodzi ratunkowej, do której wsiadł pasażer
- **body** - nr identyfikacyjny ciała (jeśli zginął w katastrofie i udało się odnaleźć ciało)
- **home.dest** - miejsce zamieszkania lub cel podróży

Już na pierwszy rzut oka można zauważyć sporo wartości NaN w kolumnach body i boat, co jest zrozumiałe, gdyż najczęściej brakuje ich dla pasażerów, którzy zginęli w katastrofie (nie zdążyli wsiąść na łódź ratunkową, nie udało się znaleźć ich ciał)

Można również dostrzec powtarzające się wartości w kolumnach: **ticket, fare, czy też cabin**

Przed użyciem funkcji **isnull()** należało przy wczytaniu ustalić, że wartości NA w zbiorze danych są oznaczone przez znaki zapytania (normalnie za wartości NA uznaje się po prostu puste pola)

Dla upewnienia się, że wartości brakujące nie były oznaczone w inny sposób:

In [6]:
```
df.isnull().sum() + df.notnull().sum() == len(df)
```

Out[6]:
```
pclass       True
survived     True
name         True
sex          True
age          True
sibsp        True
parch        True
ticket       True
fare         True
cabin        True
embarked     True
boat         True
body         True
home.dest    True
dtype: bool
```

Możemy stworzyć zmienną kategoryczną, np. dla zmiennej **body**, która poinformuje nas o wartości brakującej. Dodatkowo możemy użyć operacji **groupby**, aby zmapować brakujące wartości danej kolumny i powiązanie z wartością **survived** (0 lub 1)

In [7]:
```
df['BodyNull'] = np.where(df['body'].isnull(), 1, 0)
df.groupby(['survived'])['BodyNull'].mean()
```

Out[7]:
```
survived
0    0.850433
1    1.000000
Name: BodyNull, dtype: float64
```

In [8]:
```
#to samo w jednej linijce
df.groupby(['survived'])['body'].apply(lambda x: np.where(x.isnull(), 1, 0).mean())
```

Out[8]:
```
survived
0    0.850433
1    1.000000
Name: body, dtype: float64
```

In [9]:
```
df.head(10)[['survived', 'body','BodyNull']]
```

Out[9]:

	survived	body	BodyNull
0	1	NaN	1
1	1	NaN	1
2	0	NaN	1
3	0	135.0	0
4	0	NaN	1
5	1	NaN	1
6	1	NaN	1
7	0	NaN	1
8	1	NaN	1
9	0	22.0	0

Operacja ta pozwoliła nam zobaczyć jaki jest odsetek brakującej wartości zmiennej **body** w zależności czy ktoś przeżył katastrofę.

Dochodzimy do oczywistego wniosku, że dla tych, którzy przeżyli, wartości brakujące body mają 100% udziału w tej grupie (nie szukano i identyfikowano ciał ocalałych).

Dla tych, którzy zginęli udział ten jednak też jest bardzo duży (85%), co oznacza, że nie odnaleziono i zidentyfikowano aż 85% ciał ofiar.

Możemy zrobić takie mapowanie według zmiennej survived dla wszystkich zmiennych, dla których przedtem zauważyliśmy znaczny udział wartości brakujących:

In [10]:
```
#dla czytelności pokazuję z dokładnością do 2 miejsc po przecinku
df.groupby(["survived"])[['boat', 'body', 'home.dest', 'age', 'cabin']].apply(lambda x: x.isnull().mean()).styl
```

Out[10]:

survived	boat	body	home.dest	age	cabin
0	0.99	0.85	0.51	0.23	0.87
1	0.05	1.00	0.31	0.15	0.61

Albo też pogrupować według innej zmiennej, np. **pclass**

In [11]:
```
df.groupby("pclass")[['boat', 'body', 'home.dest', 'age', 'cabin']].apply(lambda x: x.isnull().mean()).style.fo
```

Out[11]:

pclass	boat	body	home.dest	age	cabin
1	0.38	0.89	0.11	0.12	0.21
2	0.60	0.89	0.06	0.06	0.92
3	0.76	0.92	0.72	0.29	0.98

Można pójść jeszcze krok dalej i pogrupować według obu tych zmiennych:

In [12]:
```
df.groupby(["survived", "pclass"])[['boat', 'body', 'home.dest', 'age', 'cabin']].apply(lambda x: x.isnull().mea
```

Out[12]:

survived	pclass	boat	body	home.dest	age	cabin
0	1	0.98	0.72	0.07	0.16	0.28
	2	0.99	0.80	0.09	0.08	0.96
	3	0.99	0.90	0.73	0.30	0.99
1	1	0.01	1.00	0.12	0.10	0.17
	2	0.07	1.00	0.02	0.03	0.86
	3	0.08	1.00	0.70	0.28	0.95

Na podstawie analizy powyższych tabel można dojść do wniosków:

- dla zmiennych **home.dest, age i cabin** rozkład udziału wartości brakujących według klasy jest niezależny od tego czy pasażerowie przeżyli katastrofę
- wartości brakujące dla **home.dest** są wyraźnie zależne od klasy (najwięcej dla 3). Nieznacznie wyższy udział wartości NA dla zmiennej home.dest przy grupowaniu według survived dla zmarłych może być związany z tym, że większość ofiar katastrofy to pasażerowie 3 klasy (sprawdzenie poniżej)
- wartości NA dla zmiennej **boat** są jednoznacznie zależne od tego czy ktoś przeżył katastrofę (NA związane ze śmiercią)
- wartości NA dla **age** nie są wyraźnie zależne od żadnej ze zmiennych, według których grupowaliśmy
- wartości NA dla zmiennej **body** nie występują jeśli ktoś przeżył katastrofę (oczywiste - nie szukano i identyfikowano ciał ocalałych). Dla zmarłych jednak także jest on niezależny od klasy (bardzo wysokie udziały procentowe wartości brakujących w każdej)

In [13]:
```
# ilość ofiar według klasy
df[df['survived'] == 0]['pclass'].value_counts()
```

Out[13]:
```
pclass
3    528
2    158
1    123
Name: count, dtype: int64
```

Podsumowanie:

- możemy usunąć wartości brakujące z kolumn **fare i embarked** - ze względu na małą ilość zakładamy, że są to wartości typu **MCAR**
- wartości brakujące typu **MAR - boat, body** (zależne, przynajmniej częściowo, od survived), **cabin, home.dest** (zależne od klasy). Do tej grupy zaliczymy też **age**, ponieważ brakujące wartości w tej kolumnie są związane z brakami np. w cabin (pokazane poniżej) Kolumnę boat w zasadzie w całości można usunąć ze zbioru (brak nowej informacji - bardzo silne powiązanie z survived). Innych danych brakujących ze względu na dużą ilość nie można po prostu usunąć. Rozwiązaniem jest np. wypełnienie braków (imputacja).

In [14]:
```
# odsetek brakujących wartości dla innych zmiennych, gdy wartości brakuje dla age
df[df['age'].isnull()].isnull().mean()
```

Out[14]:
```
pclass       0.000000
survived     0.000000
name         0.000000
sex          0.000000
age          1.000000
sibsp        0.000000
parch        0.000000
ticket       0.000000
fare         0.000000
cabin        0.912548
embarked     0.000000
boat         0.737643
body         0.996198
home.dest    0.771863
BodyNull     0.000000
dtype: float64
```

In [15]:
```
# odsetek wartości not-null dla innych zmiennych, gdy wartości brakuje dla age
df[df['age'].isnull()].notnull().mean()
```

Out[15]:
```
pclass       1.000000
survived     1.000000
name         1.000000
sex          1.000000
age          0.000000
sibsp        1.000000
parch        1.000000
ticket       1.000000
fare         1.000000
cabin        0.087452
embarked     1.000000
boat         0.262357
body         0.003802
home.dest    0.228137
BodyNull     1.000000
dtype: float64
```

In [ ]: