

Sieci Neuronowe

Sprawozdanie z ćwiczenia 1

Cel ćwiczenia

Celem ćwiczenia jest wprowadzenie/przypomnienie narzędzi i zapoznanie się z danymi z których będziemy korzystać w dalszej części kursu do ewaluacji sieci neuronowych jako metody uczenia maszynowego.

Technologia

Python 3.11, Jupyter Notebook.

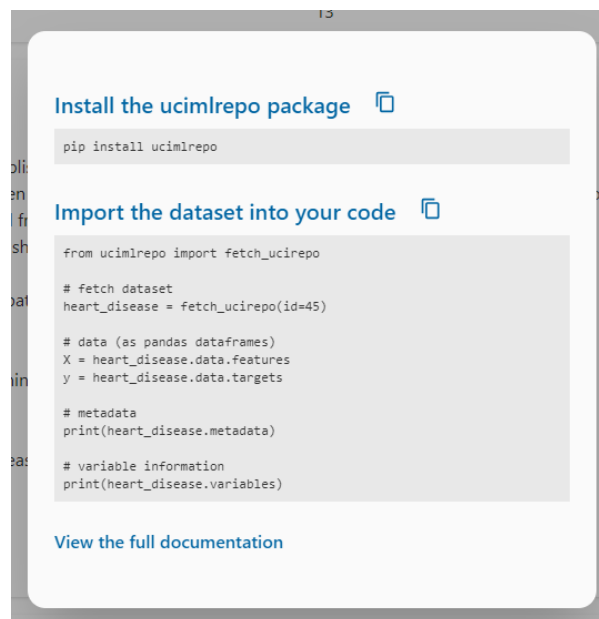
Realizacja ćwiczeń

1. Analiza eksploracyjna zbioru danych

Wybór zbioru danych

Zbiór danych pochodzi ze źródła: [Heart Disease - UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/Heart+Disease)

Można było pobrać cały zbiór, który zawierał też dane z innych regionów, ale tak jak w informacji użyto przetworzonego zbioru „cleveland”. Co ciekawe przy porównaniu rzekomych tych samych zbiorów widać pewne różnice. Zatem końcowy postawiono na użycie kodu z „Import in python”.



```
Install the ucimlrepo package

pip install ucimlrepo

Import the dataset into your code

from ucimlrepo import fetch_ucirepo

# fetch dataset
heart_disease = fetch_ucirepo(id=45)

# data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets

# metadata
print(heart_disease.metadata)

# variable information
print(heart_disease.variables)

View the full documentation
```



Jeżeli połączymy dane w pełen zbiór danych, możemy wpisać kilka pierwszych wierszy.

```
df = pd.concat([X, y], axis=1)
df.head(20)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0
5	56	1	2	120	236	0	0	178	0	0.8	1	0.0	3.0	0
6	62	0	4	140	268	0	2	160	0	3.6	3	2.0	3.0	3
7	57	0	4	120	354	0	0	163	1	0.6	1	0.0	3.0	0
8	63	1	4	130	254	0	2	147	0	1.4	2	1.0	7.0	2
9	53	1	4	140	203	1	2	155	1	3.1	3	0.0	7.0	1
10	57	1	4	140	192	0	0	148	0	0.4	2	0.0	6.0	0
11	56	0	2	140	294	0	2	153	0	1.3	2	0.0	3.0	0
12	56	1	3	130	256	1	2	142	1	0.6	2	1.0	6.0	2
13	44	1	2	120	263	0	0	173	0	0.0	1	0.0	7.0	0
14	52	1	3	172	199	1	0	162	0	0.5	1	0.0	7.0	0
15	57	1	3	150	168	0	0	174	0	1.6	1	0.0	3.0	0
16	48	1	2	110	229	0	0	168	0	1.0	3	0.0	7.0	1
17	54	1	4	140	239	0	0	160	0	1.2	1	0.0	3.0	0
18	48	0	3	130	275	0	0	139	0	0.2	1	0.0	3.0	0
19	49	1	2	130	266	0	0	171	0	0.6	1	0.0	3.0	0

Możemy sprawdzić typy danych.

```
df.info()

[25] ✓ 0.0s

... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0    age         303 non-null    int64
1    sex         303 non-null    int64
2    cp          303 non-null    int64
3    trestbps    303 non-null    int64
4    chol        303 non-null    int64
5    fbs         303 non-null    int64
6    restecg     303 non-null    int64
7    thalach     303 non-null    int64
8    exang       303 non-null    int64
9    oldpeak     303 non-null    float64
10   slope       303 non-null    int64
11   ca          299 non-null    float64
12   thal        301 non-null    float64
13   num         303 non-null    int64
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
```

Mimo, że niektóre kolumny są kategoriyczne to są zapisane w postaci liczbowej. Zbiór danych posiada 303 próbek i 14 kolumn w tym kolumna 'num' jest kolumną klasy próbki.

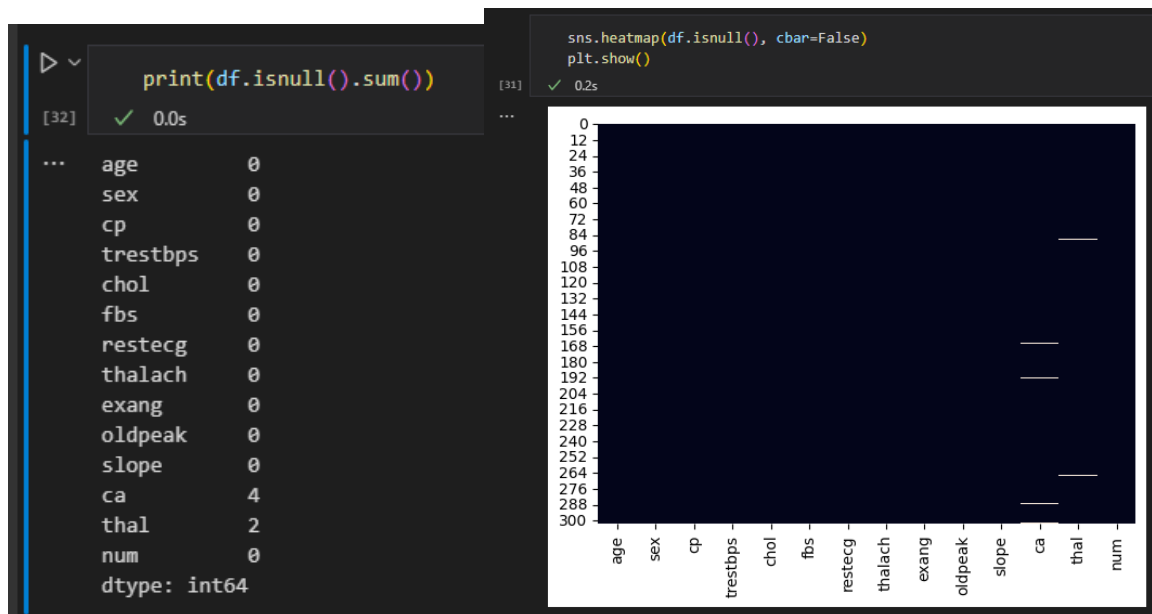
Możemy uzyskać podstawowe statystyki zbioru.

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	299.000000	301.000000	303.000000
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.672241	4.734219	0.937294
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.937438	1.939706	1.228536
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	3.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.000000	2.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.000000	4.000000

Teraz możemy zauważyć, że niektórych danych brakuje. Count(thal) = 301 oraz Count(ca) = 299

Czy występują cechy brakujące i jaką strategię możemy zastosować, żeby je zastąpić?



W przypadku cechy liczbowej 'ca' możemy wstawić średnią w miejsce braków. W przypadku kategorycznej cechy 'thal' możemy wstawić modalną. Byłoby to błędem w przypadku 'ca', ponieważ, mimo że to są liczby to są w przedziale liczb całkowitych 0-3 zatem wstawimy medianę.

```
median = df['ca'].median()
df['ca'].fillna(median, inplace=True)
mode_category = df['thal'].mode()[0]
df['thal'].fillna(mode_category, inplace=True)
```

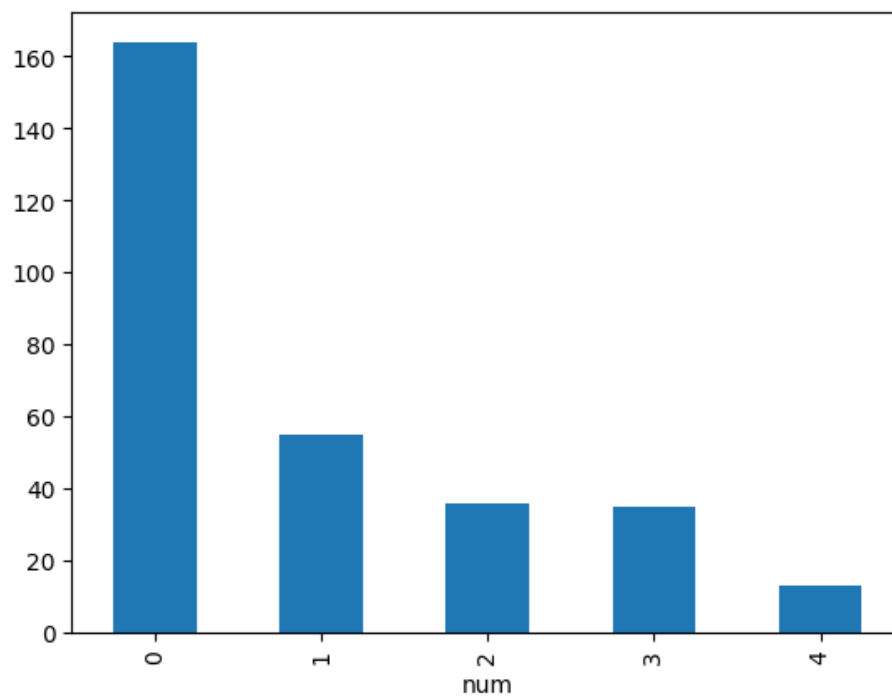
```
print(df.isnull().sum())
```

[62] ✓ 0.0s

...	age	0
	sex	0
	cp	0
	trestbps	0
	chol	0
	fbs	0
	restecg	0
	thalach	0
	exang	0
	oldpeak	0
	slope	0
	ca	0
	thal	0
	num	0
	dtype:	int64

Czy zbiór jest zbalansowany pod względem liczby próbek na klasy?

```
df['num'].value_counts().plot.bar()
```



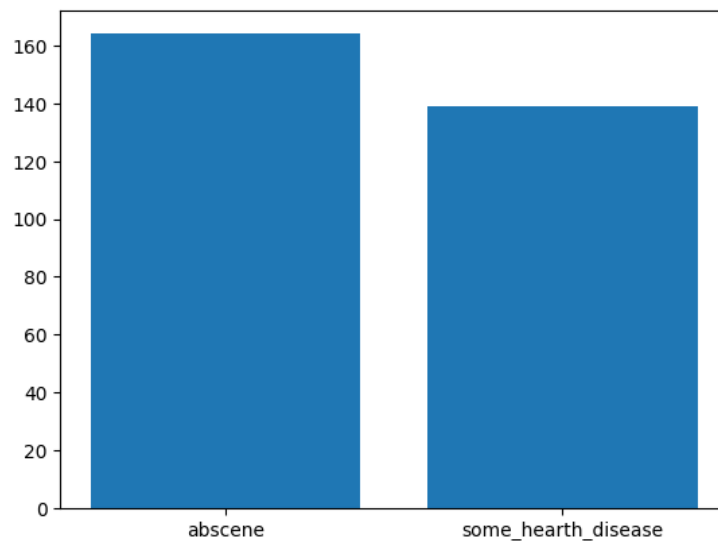
0 – brak choroby, [1,4] – choroba

Z informacji dotyczących zbioru wynika, że zbiór miał koncentrować się na wykrywaniu braku lub obecności jakiejś choroby. Dlatego też możemy wyświetlić wykres:

```
abscene = df['num'].value_counts()[0]
some_hearth_disease = df['num'].value_counts()[1:].sum()
print('abscene: ', abscene)
print('some_hearth_disease: ', some_hearth_disease)
plt.bar(['abscene', 'some_hearth_disease'], [abscene, some_hearth_disease])
```

abscene: 164

some_hearth_disease: 139



Dopiero teraz można zaakceptować zbalansowanie pod względem liczby próbek na klasy, aczkolwiek to czy będziemy rozróżniać kategorie choroby trzeba gruntownie ustalić.

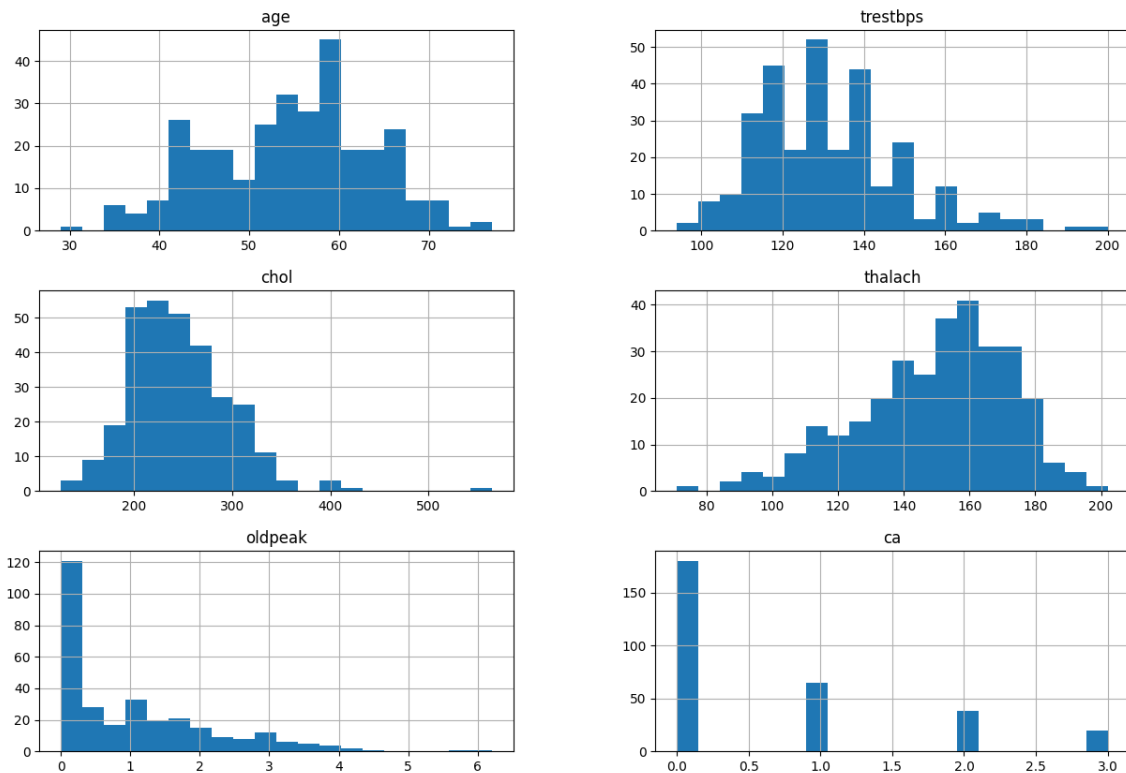
Średnie i odchylenia cech liczbowych.

```
numeric_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca']  
df[numeric_features].describe().loc[['mean', 'std']]
```

	age	trestbps	chol	thalach	oldpeak	ca
mean	54.438944	131.689769	246.693069	149.607261	1.039604	0.663366
std	9.038662	17.599748	51.776918	22.875003	1.161075	0.934375

Rozkłady cech liczbowych.

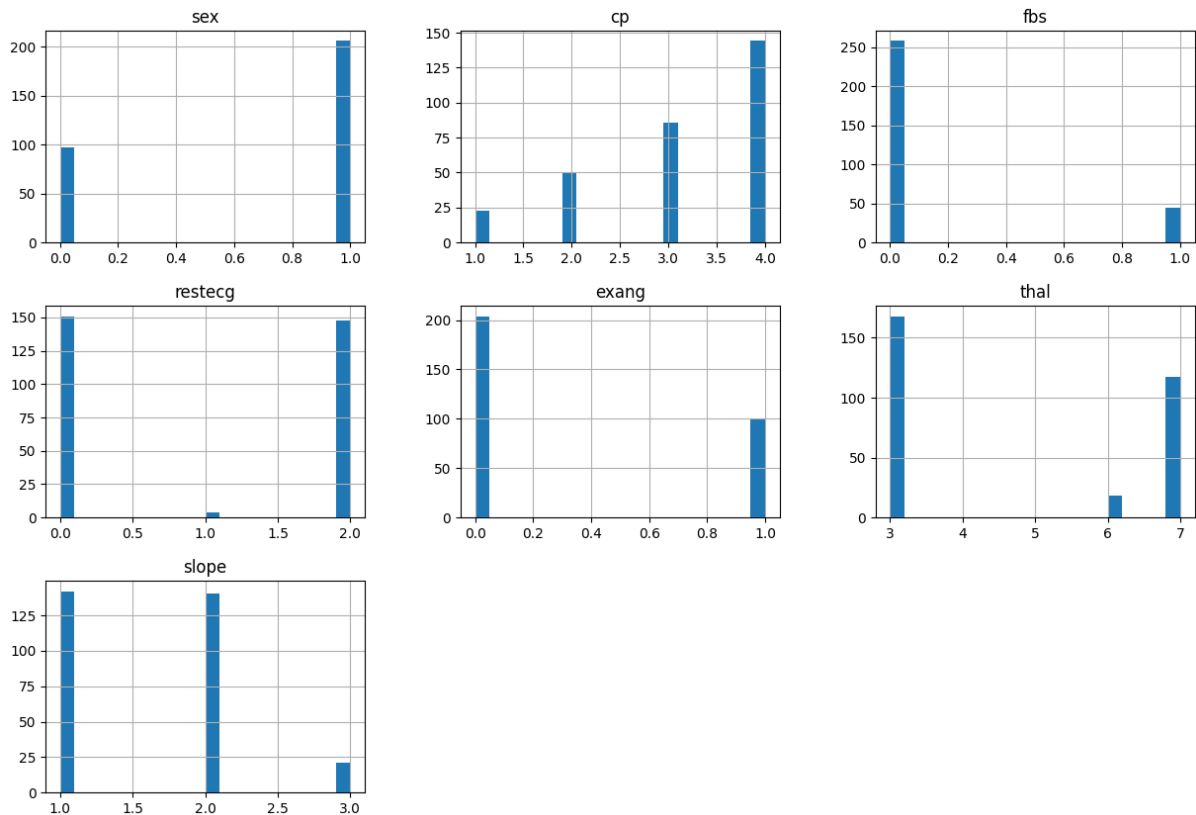
```
df[numeric_features].hist(bins=20, figsize=(15, 10))
```



Jedynymi cechami, które mogły być o rozkładzie normalnym to 'chol', 'thalach' i 'age' reszta nie przypomina rozkładów normalnych.

Rozkłady cech katerycznych:

```
categories_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'thal',  
'slope']  
df[categories_features].hist(bins=20, figsize=(15, 10))
```



Tutaj trzeba zauważyć, że dla każdej cechy istnieje jedna klasa, której licznosc jest bardzo mala, co sprawia, że rozkłady nie są idealnie równomierne.

Kod przekształcający dane do macierzy cech liczbowych (przykłady × cechy)

```
def one_hot_encode(df, column, column_names):  
    dummies = pd.get_dummies(df[column], prefix=column)  
    column_names = [column + '_' + str(name) for name in column_names]  
    dummies.columns = column_names  
    dummies = dummies.astype('int64')  
    df = pd.concat([df, dummies], axis=1)  
    df.drop(column, axis=1, inplace=True)  
    return df
```

```
df = one_hot_encode(df, 'cp', ['typical_angina', 'atypical_angina', 'non-anginal_pain',  
'asymptomatic'])  
df = one_hot_encode(df, 'thal', ['normal', 'ST-T_wave_abnormality', 'left_ventricular_hypertrophy'])  
df = one_hot_encode(df, 'slope', ['upsloping', 'flat', 'downsloping'])  
df = one_hot_encode(df, 'restecg', ['normal', 'fixed_defect', 'reversable_defect'])
```

```
df.info()
[163] ✓ 0.0s

... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 23 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   age                                       303 non-null    int64
1   sex                                       303 non-null    int64
2   trestbps                                  303 non-null    int64
3   chol                                      303 non-null    int64
4   fbs                                       303 non-null    int64
5   thalach                                   303 non-null    int64
6   exang                                     303 non-null    int64
7   oldpeak                                  303 non-null    float64
8   ca                                       303 non-null    float64
9   num                                       303 non-null    int64
10  cp_typical_angina                        303 non-null    int64
11  cp_atypical_angina                      303 non-null    int64
12  cp_non-anginal_pain                     303 non-null    int64
13  cp_asymptomatic                         303 non-null    int64
14  thal_normal                             303 non-null    int64
15  thal_ST-T_wave_abnormality              303 non-null    int64
16  thal_left_ventricular_hypertrophy        303 non-null    int64
17  slope_upsloping                         303 non-null    int64
18  slope_flat                              303 non-null    int64
19  slope_downsloping                       303 non-null    int64
...
21  restecg_fixed_defect                    303 non-null    int64
22  restecg_reversable_defect                303 non-null    int64
dtypes: float64(2), int64(21)
memory usage: 54.6 KB

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output set
```

- 2.
- 3.
- 4.
- 5.

Wnioski