
UMB W07: redukcja wymiarowości

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

import umb_tools as umb
```

```
# konfiguracja
plt.rcParams["figure.figsize"] = [5, 4]
pd.set_option("display.float_format", lambda x: "%.4f" % x)
```

1. Zbiór danych

Wczytanie i normalizacja danych

```
# odczyt pliku TSV (zwracane są: zbiór danych w postaci DataFrame biblioteki Pandas oraz lista kolumn)
(df, c_names) = umb.read_data("data/BreastCancer.txt")
df
```

#BreastCancer	labels	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cor
842302	1	17.9900	10.3800	122.8000	1001.0000	0.1184	
874858	1	14.2200	23.1200	94.3700	609.9000	0.1075	
875263	1	12.3400	26.8600	81.1500	477.4000	0.1034	
87556202	1	14.8600	23.2100	100.4000	671.4000	0.1044	
875938	1	13.7700	22.2900	90.6300	588.9000	0.1200	
...
8910720	2	10.7100	20.3900	69.5000	344.9000	0.1082	
8910506	2	12.8700	16.2100	82.3800	512.2000	0.0943	
8910499	2	13.5900	21.8400	87.1600	561.0000	0.0796	
8912055	2	11.7400	14.0200	74.2400	427.3000	0.0781	
92751	2	7.7600	24.5400	47.9200	181.0000	0.0526	

569 rows × 31 columns

```

# pobranie etykiet klas
labels = df.iloc[:, 0].to_numpy()

# pobranie macierzy danych
data = df.iloc[:, 1:].to_numpy()

# normalizacja
data = StandardScaler().fit_transform(data)

```

Informacje o klasach

```

# pobranie indeksów próbek z klas
(c_labels, c_index) = umb.class_info(labels)

# liczba klas
c_n = len(c_labels)

# informacje o klasach
print(f"\nClasses: {c_n}")
for i in range(c_n):
    print(f"  name = {c_names[i]}, label = {c_labels[i]}, samples = {len(c_index[i])}")

```

```

Classes: 2
  name = Malignant, label = 1, samples = 212
  name = Benign, label = 2, samples = 357

```

2. Analiza składowych głównych (PCA, *Principal Component Analysis*)

```
from sklearn.decomposition import PCA
```

```
# wyznaczenie rzutu danych w przestrzeni trójwymiarowej
n_components = 3

pca_model = PCA(n_components)
pc = pca_model.fit_transform(data)

print(f"\nData size in the original space: {data.shape[0]} samples, {data.shape[1]} features")
print(f"Data size in principal components space: {pc.shape[0]} samples, {pc.shape[1]} features")
```

Data size in the original space: 569 samples, 30 features
Data size in principal components space: 569 samples, 3 features

```
fig = plt.figure(figsize=(10, 4))

# wykres 2D
ax = fig.add_subplot(1, 2, 1)

for i in range(c_n):
    ax.scatter(pc[c_index[i], 0], pc[c_index[i], 1], color=umb.class_color(i), label=c_names[i])

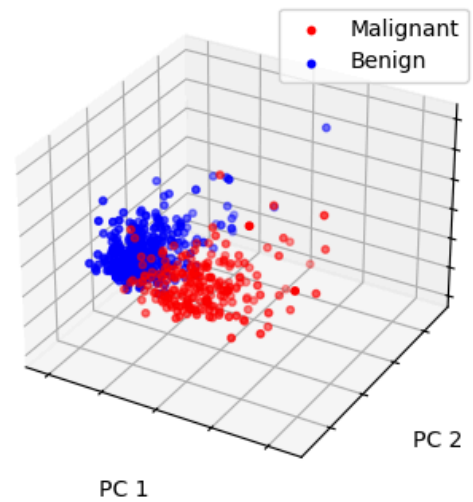
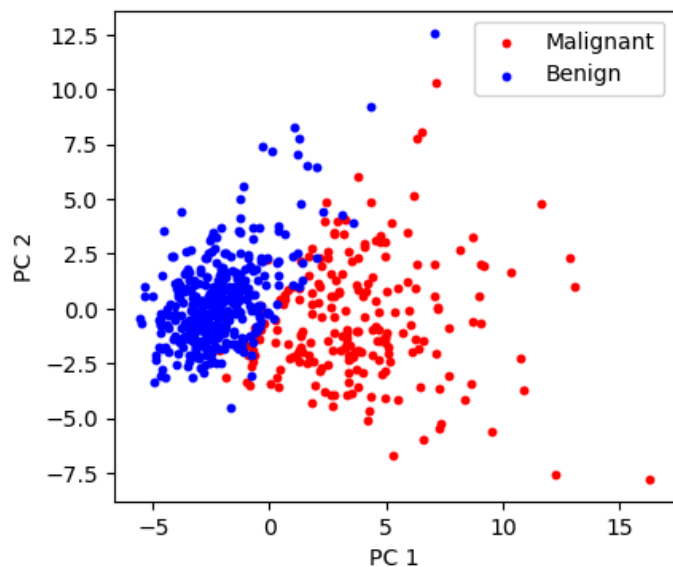
ax.legend()
ax.set_xlabel(f"PC 1")
ax.set_ylabel(f"PC 2")

# wykres 3D
ax = fig.add_subplot(1, 2, 2, projection="3d")

for i in range(c_n):
    ax.scatter(pc[c_index[i], 0], pc[c_index[i], 1], pc[c_index[i], 2], color=umb.class_color(i))

ax.legend()
ax.set(xticklabels=[], yticklabels=[], zticklabels=[])
ax.set_xlabel("PC 1")
ax.set_ylabel("PC 2")
ax.set_zlabel("PC 3")

plt.show()
```



```
fig = plt.figure(figsize=(10, 4))

# wykres 2D
ax = fig.add_subplot(1, 2, 1)
umb.pca_plot(data, labels, c_names, ax)

# wykres 3D
ax = fig.add_subplot(1, 2, 2, projection="3d")
umb.pca_plot_3d(data, labels, c_names, ax)

plt.show()
```

