

SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 5

Data 28.02.2025

Temat: Wykorzystanie narzędzi do eksploracyjnej analizy danych (EDA)

Wariant 14

Krzysztof Świerczek

Informatyka

II stopień, stacjonarne,

1semestr, gr. A

1. Polecenie: Wykorzystanie narzędzi do eksploracyjnej analizy danych (EDA)

2. Github:

<https://github.com/Krzycho165/STUDIA>

In [4]: *# 0. Krzysztof Świerczek Przygotowanie danych*

```
import pandas as pd
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv('final_solution.csv')
#print(df.info())
#print(df.describe())
numerical_features = ['age', 'income', 'outcome']
numerical_df = df[numerical_features]

if numerical_df.isnull().sum().any():
    numerical_df.fillna(numerical_df.mean(), inplace=True)
```

In [5]: *# 1. Krzysztof Świerczek Zidentyfikować wartości odstające za pomocą algorytmu I*

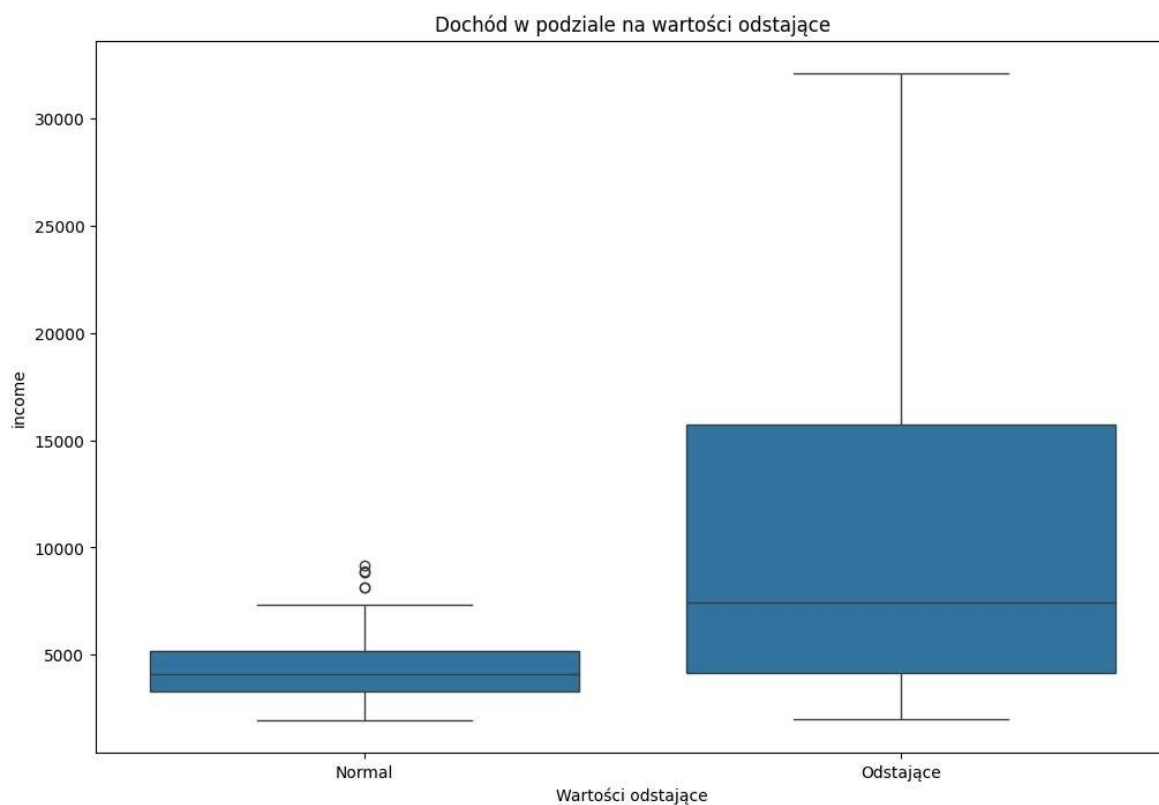
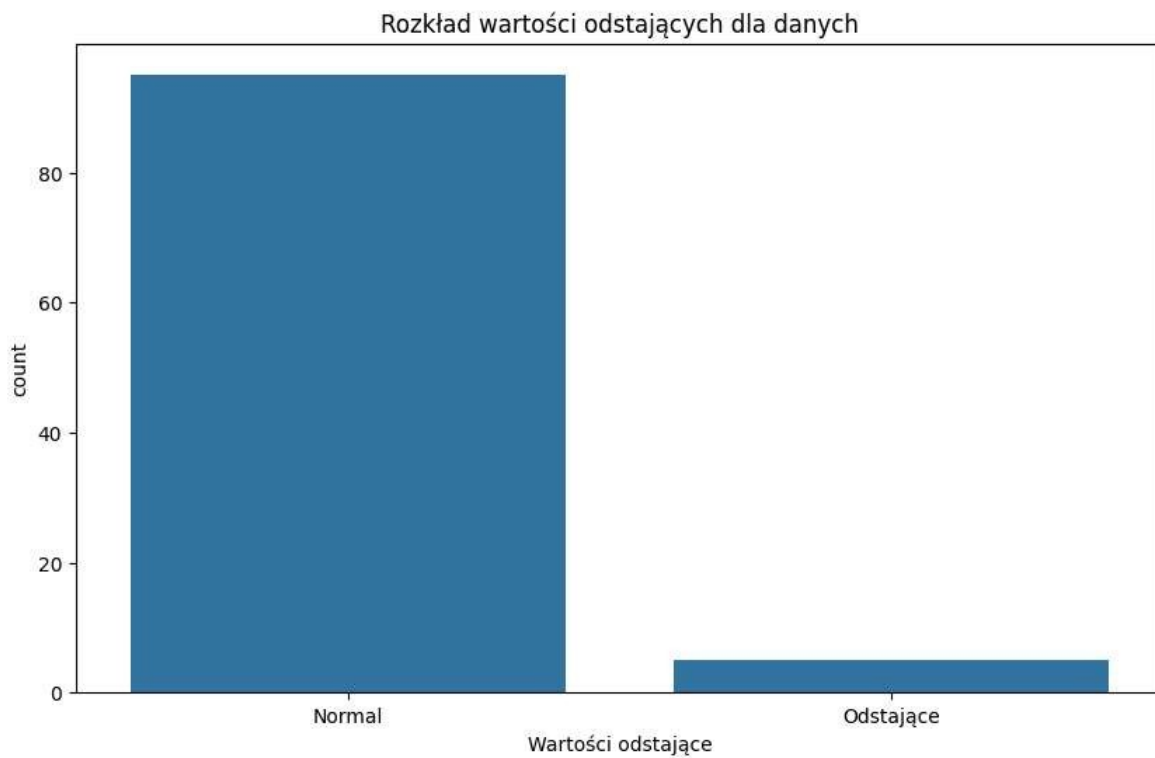
```
scaler = StandardScaler()
numerical_df_scaled = scaler.fit_transform(numerical_df)

# Model Isolation Forest
iso_forest = IsolationForest(contamination=0.05, random_state=42)
df['Wartości odstające'] = iso_forest.fit_predict(numerical_df_scaled)

# Oznaczenie wartości odstających (-1 oznacza odstające, 1 oznacza normalne)
df['Wartości odstające'] = df['Wartości odstające'].apply(lambda x: 'Odstające'

# Wizualizacja wyników
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Wartości odstające')
plt.title("Rozkład wartości odstających dla danych")
plt.show()

# Analiza zmiennych z uwzględnieniem odstających wartości
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='Wartości odstające', y='income')
plt.title("Dochód w podziale na wartości odstające")
plt.show()
```



```
In [6]: # 2. Krzysztof Świerczek Redukować wymiarowość danych z użyciem PCA,
# Redukcja do 2 wymiarów
pca = PCA(n_components=2)
pca_result = pca.fit_transform(numerical_df_scaled)
df['PC1'] = pca_result[:, 0]
df['PC2'] = pca_result[:, 1]

# Wynik w formie DataFrame
pca_df = pd.DataFrame(pca_result, columns=['PC1', 'PC2'])

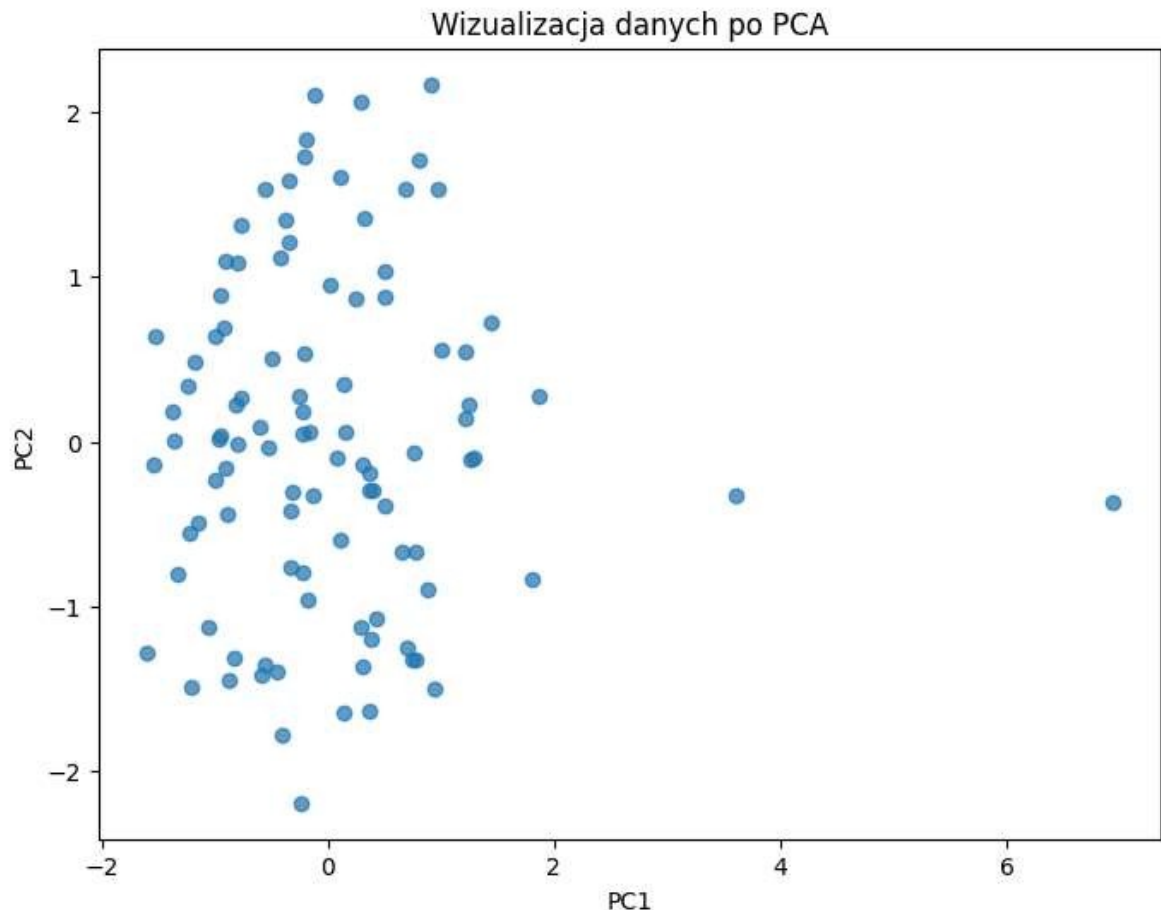
explained_variance = pca.explained_variance_ratio_
print(f"Wariancja wyjaśniana przez każdą składową: {explained_variance}")
```

```
print(f"Łączna wariancja: {sum(explained_variance):.2f}")

plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PC1'], pca_df['PC2'], alpha=0.7)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Wizualizacja danych po PCA')
plt.show()
```

Wariancja wyjaśniana przez każdą składową: [0.41987303 0.33992199]

Łączna wariancja: 0.76



```
In [7]: # 3. Krzysztof Świerczek Tworzyć zaawansowane interaktywne wizualizacje danych,
df['Różnica między dochodami a wydatkami'] = df['income'] - df['outcome']
fig_pca = px.scatter(df, x='Różnica między dochodami a wydatkami', y='savings',
fig_pca.show()
```

```
In [8]: fig_box = px.box(df, x='Wartości odstające', y='income', color='Wartości odstające',  
                        title='Rozkład dochodów w zależności od wartości odstających')  
fig_box.show()
```

```
In [9]: fig_hist = px.histogram(df, x='Wartości odstające', color='Wartości odstające',  
                                title='Rozkład wartości odstających w danych', marginal=  
                                fig_hist.show()
```

```
In [10]: fig_children_income = px.scatter(df, x='children', y='income', color='income', h
        title='Zależność między liczbą dzieci a dochodem'
fig_children_income.show()
```

```
In [13]: # 4. Zwizualizować dane wielowymiarowe za pomocą zaawansowanych algorytmów (t-SNE)

from sklearn.manifold import TSNE
from umap import UMAP

columns_of_interest = ['age', 'income', 'outcome', 'savings', 'credit_score', 's
X = df[columns_of_interest]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Użycie UMAP do redukcji wymiarowości
umap_model = UMAP(n_components=2, random_state=42)
umap_results = umap_model.fit_transform(X_scaled)

tsne_model = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=300)
tsne_results = tsne_model.fit_transform(X_scaled)

umap_df = pd.DataFrame(umap_results, columns=['UMAP1', 'UMAP2'])
tsne_df = pd.DataFrame(tsne_results, columns=['t-SNE1', 't-SNE2'])
df_umap = pd.concat([df, umap_df], axis=1)
df_tsne = pd.concat([df, tsne_df], axis=1)

# UMAP
plt.figure(figsize=(10, 8))
sns.scatterplot(data=df_umap, x='UMAP1', y='UMAP2', hue='city', palette='viridis')
plt.title('UMAP Visualization')
```



```
plt.show()

# t-SNE
plt.figure(figsize=(10, 8))
sns.scatterplot(data=df_tsne, x='t-SNE1', y='t-SNE2', hue='city', palette='virid
plt.title('t-SNE Visualization')
plt.show()
```

```
-----
ImportError                                Traceback (most recent call last)
Cell In[13], line 4
      1 # 4. Zwizualizować dane wielowymiarowe za pomocą zaawansowanych algorytmó
w (t-SNE, UMAP)
      3 from sklearn.manifold import TSNE
----> 4 from umap import UMAP
      7 columns_of_interest = ['age', 'income', 'outcome', 'savings', 'credit_sco
re', 'spending_score']
      8 X = df[columns_of_interest]

ImportError: cannot import name 'UMAP' from 'umap' (C:\Users\krzys\AppData\Local
\Programs\Python\Python310\lib\site-packages\umap\__init__.py)
```

```
In [14]: # 6. Analizować zależności między zmiennymi za pomocą macierzy korelacji.
import plotly.figure_factory as ff

#Numeric dataframe:
numerical_features = ['income', 'outcome', 'savings']
numeric_df = df[numerical_features]
correlation_matrix = numeric_df.corr()
fig = ff.create_annotated_heatmap(
    z=correlation_matrix.values,
    x=correlation_matrix.columns.tolist(),
    y=correlation_matrix.columns.tolist(),
    colorscale='Viridis',
    showscale=True,
)

fig.update_layout(title="Macierz korelacji zmiennych")
fig.show()
```

```
In [16]: # 7. Przeprowadzać testy statystyczne dla analizy różnic w grupach.

from scipy import stats
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
from scipy.stats import chi2_contingency

# Test T-studenta
group1 = df[df['employment_status'] == 'employed']['income']
group2 = df[df['employment_status'] == 'retired']['income']

t_stat, p_value = stats.ttest_ind(group1, group2)

print(f"T-statystyka: {t_stat}")
print(f"P-wartość: {p_value}")
if p_value < 0.05:
    print("Istnieje istotna różnica w średnich income między grupą zatrudnionych a")
else:
    print("Brak istotnej różnicy w średnich income między grupą zatrudnionych a")

# Test ANOVA
model = ols('outcome ~ employment_status', data=df).fit()
anova_table = anova_lm(model)

print(anova_table)

if anova_table['PR(>F)'][0] < 0.05:
```

```

    print("Istnieje istotna różnica w średnich outcome między przynajmniej dwoma
else:
    print("Brak istotnej różnicy w średnich outcome między przynajmniej dwoma gr

# Test Chi-kwadrat
contingency_table = pd.crosstab(df['employment_status'], df['city'])

chi2_stat, p_val, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-kwadrat statystyka: {chi2_stat}")
print(f"P-wartość: {p_val}")

if p_val < 0.05:
    print("Istnieje istotna zależność między employment_status a city. (np. w Wa
else:
    print("Brak istotnej zależności między employment_status a city.")

```

T-statystyka: 1.9184754519058442

P-wartość: 0.06258664760296179

Brak istotnej różnicy w średnich income między grupą zatrudnionych a emerytami.

	df	sum_sq	mean_sq	F	PR(>F)
employment_status	3.0	4.239691e+06	1.413230e+06	2.035066	0.114095
Residual	96.0	6.666619e+07	6.944395e+05	NaN	NaN

Brak istotnej różnicy w średnich outcome między przynajmniej dwoma grupami employment_status.

Chi-kwadrat statystyka: 32.33963867527266

P-wartość: 0.6434063139648425

Brak istotnej zależności między employment_status a city.

C:\Users\krzys\AppData\Local\Temp\ipykernel_6404\1644763415.py:28: FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`

In []:

3. Wnioski z wykonanego kodu:

Podczas realizacji ćwiczenia wykorzystano kilka popularnych bibliotek języka Python, co pozwoliło na zdobycie praktycznej wiedzy na temat ich obsługi i zastosowania w analizie danych oraz wizualizacji.

1. Obsługa pandas – manipulacja danymi

Podstawową operacją było wczytanie danych z pliku CSV przy użyciu biblioteki `pandas`. Pozwoliło to na zapoznanie się z metodami takimi jak `read_csv()`, `info()` i `describe()`, które dostarczają kluczowych informacji o strukturze danych. Ponadto zastosowano metody sprawdzania brakujących wartości i ich uzupełniania (`fillna()`), co jest istotnym elementem przygotowania danych do analizy.

2. Standaryzacja danych z `sklearn.preprocessing`

W celu ujednolicenia wartości numerycznych w zbiorze danych użyto `StandardScaler()` z biblioteki `sklearn.preprocessing`. Pozwoliło to na zrozumienie potrzeby normalizacji danych, zwłaszcza przed zastosowaniem algorytmów uczących się, takich jak `IsolationForest`.

3. Wykorzystanie `IsolationForest` do detekcji anomalii

W celu identyfikacji wartości odstających zastosowano algorytm `IsolationForest` z biblioteki `sklearn.ensemble`. Dzięki temu można było zapoznać się z koncepcją detekcji anomalii oraz parametrem `contamination`, który określa procent wartości odstających w zbiorze danych.

4. Wizualizacja wyników – `matplotlib`, `seaborn` i `plotly`

Do prezentacji wyników analizy wykorzystano różne biblioteki do wizualizacji danych: `matplotlib` – podstawowe narzędzie do tworzenia wykresów. `seaborn` – bardziej zaawansowana biblioteka do wizualizacji statystycznych, pozwalająca na szybkie tworzenie czytelnych wykresów. `plotly` – biblioteka do interaktywnych wizualizacji, umożliwiającą dynamiczną analizę danych.

Porównanie tych narzędzi pozwoliło na lepsze zrozumienie ich możliwości oraz wybór odpowiedniego narzędzia do konkretnych zastosowań.

5. Wykorzystanie PCA do redukcji wymiarowości

W kodzie zastosowano analizę głównych składowych (PCA) z biblioteki `sklearn.decomposition`. Dzięki temu możliwe było poznanie sposobu redukcji wymiarowości danych w celu ich lepszej wizualizacji i analizy.

Podsumowanie

Ćwiczenie pozwoliło na zdobycie praktycznych umiejętności w zakresie pracy z danymi, ich wstępnego przetwarzania, wykrywania anomalii oraz wizualizacji wyników. Zastosowanie różnych bibliotek umożliwiło porównanie ich funkcjonalności i dobór najbardziej efektywnych narzędzi do analizy.