Ćwiczenie laboratoryjne: 5. Wykorzystanie narzędzi do eksploracyjnej analizy danych (EDA)

Katedra Informatyki i Automatyki

Cel ćwiczenia

Celem ćwiczenia jest wykorzystanie zaawansowanych metod eksploracyjnej analizy danych (EDA) z użyciem nowoczesnych narzędzi i bibliotek do:

- detekcji wartości odstających z wykorzystaniem metod statystycznych i uczenia maszynowego,
- analizy głównych składowych (PCA) dla redukcji wymiarowości,
- interaktywnej wizualizacji danych,
- badania rozkładów danych oraz testów statystycznych.

Wymagania wstępne

Studenci powinni mieć podstawową znajomość:

- Python i bibliotek takich jak Pandas oraz Matplotlib,
- statystyki (rozumienie testów statystycznych i podstawowych miar),
- metod redukcji wymiarowości (np. PCA).

Narzędzia

Do realizacji ćwiczenia będą wykorzystane:

- **Python 3.x**
- Biblioteki: **Scikit-learn**, **Pycaret**, **Plotly**, **Statsmodels**, **Altair**
- Środowisko: **Jupyter Notebook**

Instrukcja

1. Przygotowanie środowiska pracy

1. Uruchom środowisko Jupyter Notebook lub IDE.
2. Zainstaluj wymagane biblioteki:

```
pip install scikit-learn pycaret plotly statsmodels altair
```

3. Pobierz plik `housing.csv` zawierający dane o cenach domów.

2. Wczytanie i wstępne przetwarzanie danych

1. Wczytaj dane:

```python
import pandas as pd

# Wczytanie danych
df = pd.read_csv('housing.csv')

# Podstawowe informacje o danych
print(df.info())
print(df.describe())
```

2. Sprawdź rozkłady danych:

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Histogram rozkladu cen
sns.histplot(df['price'], kde=True)
plt.title('Rozklad cen domow')
plt.show()
```

3. Detekcja wartości odstających

1. Wykorzystaj algorytm Isolation Forest:

```python
from sklearn.ensemble import IsolationForest

# Dopasowanie modelu Isolation Forest
isolation_forest = IsolationForest(contamination=0.05)
df['outliers'] = isolation_forest.fit_predict(df[['price', 'area']])
```

```
6
7       # Wyswietlenie wartosci odstajacych
8       print(df[df['outliers'] == -1])
```

2. Wizualizacja wartości odstających:

```
1       import plotly.express as px
2
3       fig = px.scatter(df, x='size', y='price', color='
            outliers',
4                        title='Wartosci odstajace w danych')
5       fig.show()
```

4. Analiza głównych składowych (PCA)

1. Zastosuj PCA dla redukcji wymiarowości:

```
1       from sklearn.decomposition import PCA
2       from sklearn.preprocessing import StandardScaler
3
4       # Skalowanie danych
5       scaler = StandardScaler()
6       scaled_data = scaler.fit_transform(df[['area', 'price
            ', 'bedrooms']])
7
8       # PCA
9       pca = PCA(n_components=2)
10      principal_components = pca.fit_transform(scaled_data)
11
12      # Wynik PCA
13      df['PC1'] = principal_components[:, 0]
14      df['PC2'] = principal_components[:, 1]
15      print(pca.explained_variance_ratio_)
```

2. Wizualizacja komponentów głównych:

```
1       fig = px.scatter(df, x='PC1', y='PC2', color='price',
2                        title='Wizualizacja glownych
                             skladowych')
3       fig.show()
```

4a. Wizualizacja redukcji wymiarowości - t-SNE

1. Redukcja wymiarowości za pomocą t-SNE:

```
from sklearn.manifold import TSNE

# t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(df[['price','area','
    bedrooms','bathrooms','stories','parking']])

# Dodanie wynikow do ramki danych
df['tSNE1'] = tsne_results[:, 0]
df['tSNE2'] = tsne_results[:, 1]

# Wizualizacja
sns.scatterplot(data=df, x='tSNE1', y='tSNE2', hue='
    prefarea', palette='coolwarm')
plt.title('Wizualizacja t-SNE')
plt.show()
```

4b. Wizualizacja redukcji wymiarowości - UMAP

1. Redukcja wymiarowości za pomocą UMAP:

```
    import umap

    # UMAP
    reducer = umap.UMAP(n_neighbors=10, min_dist=0.1,
        random_state=42)
    umap_results = reducer.fit_transform(df[['price','
        area','bedrooms','bathrooms','stories','parking'
        ]])

    # Dodanie wynikow do ramki danych
    df['UMAP1'] = umap_results[:, 0]
    df['UMAP2'] = umap_results[:, 1]

    # Wizualizacja
    sns.scatterplot(data=df, x='UMAP1', y='UMAP2', hue='
        species', palette='Spectral')
    plt.title('Wizualizacja UMAP')
    plt.show()
```

5. Interaktywna analiza zależności

1. Stwórz interaktywny wykres zależności:

```
    import altair as alt

```

```
    chart = alt.Chart(df).mark_circle(size=60).encode(
        x='size',
        y='price',
        color='bedrooms',
        tooltip=['area', 'price', 'rooms']
    ).interactive()

    chart.show()
```

5a. Analiza macierzy korelacji

1. Oblicz i wizualizuj macierz korelacji:

```
    import numpy as np

    # Macierz korelacji
    correlation_matrix = df.corr()

    # Heatmap
    sns.heatmap(correlation_matrix, annot=True, cmap='
       coolwarm')
    plt.title('Macierz korelacji')
    plt.show()
```

6. Testy statystyczne

1. Sprawdź, czy cena domów różni się w zależności od liczby spalni (`bedrooms`):

```
    from statsmodels.formula.api import ols
    from statsmodels.stats.anova import anova_lm

    # Model ANOVA
    model = ols('price ~ C(bedrooms)', data=df).fit()
    anova_results = anova_lm(model)

    print(anova_results)
```

2. Interpretuj wyniki testu ANOVA: *Czy różnice są istotne statystycznie?*

Pytania kontrolne

1. Jak działa algorytm Isolation Forest i jak interpretować jego wyniki?
2. W jaki sposób analiza PCA może pomóc w eksploracyjnej analizie danych?

3. Jakie są zalety wykorzystania interaktywnych wizualizacji?
4. Jak interpretować wyniki testu ANOVA?
5. Jak działa algorytm t-SNE i kiedy warto go stosować?
6. W jaki sposób algorytm UMAP różni się od t-SNE?
7. Jak interpretować macierz korelacji?

Podsumowanie

Podczas ćwiczenia studenci nauczyli się:

- identyfikować wartości odstające za pomocą algorytmu Isolation Forest,
- redukować wymiarowość danych z użyciem PCA,
- tworzyć zaawansowane interaktywne wizualizacje danych,
- wizualizować dane wielowymiarowe za pomocą zaawansowanych algorytmów (t-SNE, UMAP),
- tworzyć interaktywne wizualizacje danych w 2D i 3D,
- analizować zależności między zmiennymi za pomocą macierzy korelacji.
- przeprowadzać testy statystyczne dla analizy różnic w grupach.

Appendices

A t-SNE: Redukcja Wymiarowości i Wizualizacja

A.1 Czym jest t-SNE?

t-SNE (t-Distributed Stochastic Neighbor Embedding) to technika redukcji wymiarowości zaprojektowana do wizualizacji danych o wysokiej wymiarowości w przestrzeni o niższej wymiarowości (najczęściej 2D lub 3D). Metoda ta koncentruje się na zachowaniu lokalnych struktur danych, co czyni ją wyjątkowo przydatną do uwidaczniania klastrów i wzorców w złożonych zbiorach danych.

Główne etapy działania t-SNE:

- **Przekształcenie podobieństw w przestrzeni o wysokiej wymiarowości:** t-SNE mierzy podobieństwa między punktami, obliczając prawdopodobieństwo, że dany punkt jest bliski innemu punktowi w tej przestrzeni. Prawdopodobieństwa te są wyrażone w postaci rozkładu Gaussa.

- **Modelowanie podobieństw w przestrzeni o niskiej wymiarowości:** Podobieństwa między punktami w przestrzeni o niskiej wymiarowości są modelowane za pomocą rozkładu t-Studenta. Jest to kluczowy element metody, który zmniejsza efekt tzw. "zanikania odległości" (ang. crowding problem).

- **Minimalizacja różnicy pomiędzy rozkładami:** t-SNE minimalizuje dywergencję Kullbacka-Leiblera (KL) między rozkładami w przestrzeni o wysokiej i niskiej wymiarowości, aby zachować relacje lokalne.

A.2 Zrozumienie Wizualizacji t-SNE

Wyniki wizualizacji t-SNE przedstawiają dane w przestrzeni 2D lub 3D w sposób, który umożliwia intuicyjne zrozumienie ich struktury. Kluczowe aspekty wizualizacji to:

- **Klastery:** Punkty, które były blisko siebie w przestrzeni o wysokiej wymiarowości, są również blisko siebie na wykresie t-SNE, co pozwala na łatwe identyfikowanie grup w danych.

- **Relacje lokalne:** t-SNE kładzie nacisk na zachowanie lokalnych sąsiedztw, co oznacza, że punkty bliskie sobie w oryginalnej przestrzeni sąsiadują ze sobą na wykresie.

- **Relacje globalne:** Ze względu na specyfikę algorytmu, t-SNE nie zawsze dokładnie zachowuje globalną strukturę danych (np. odległości między klastrami mogą nie odpowiadać ich rzeczywistym odległościom w przestrzeni o wysokiej wymiarowości).

A.3 Zastosowania Wizualizacji t-SNE

Wizualizacje uzyskane za pomocą t-SNE znajdują zastosowanie w różnych dziedzinach, w tym:

- **Eksploracyjnej Analizie Danych:** t-SNE jest często używane do identyfikowania ukrytych klastrów, wzorców i punktów odstających w danych.

- **Uczeniu Maszynowym:** Do oceny separacji między klasami w zbiorach danych używanych do klasyfikacji.
- **Bioinformatyce i Genomice:** Do analizy danych genomowych i proteomicznych, w których wymiarowość danych jest szczególnie wysoka.

A.4 Porównanie t-SNE z innymi metodami

t-SNE wyróżnia się na tle innych metod redukcji wymiarowości, takich jak PCA czy UMAP, ze względu na:

- Skupienie na lokalnych relacjach pomiędzy danymi, co jest szczególnie przydatne w przypadku analiz klastrowych.
- Brak założenia liniowości, co pozwala na uchwycenie bardziej złożonych struktur danych.
- Wyższe wymagania obliczeniowe, co sprawia, że t-SNE jest mniej efektywne dla bardzo dużych zbiorów danych w porównaniu z UMAP.

A.5 Podsumowanie

t-SNE jest potężnym narzędziem do redukcji wymiarowości i wizualizacji danych. Dzięki swojej zdolności do zachowania lokalnych relacji w danych o wysokiej wymiarowości, umożliwia odkrywanie klastrów i wzorców w sposób wizualnie intuicyjny. Jednak w interpretacji wyników należy uwzględniać ograniczenia dotyczące globalnych relacji i konieczności dostosowania hiperparametrów.

B UMAP: Redukcja Wymiarowości i Wizualizacja

B.1 Czym jest UMAP?

UMAP (Uniform Manifold Approximation and Projection) to technika redukcji wymiarowości, która służy przede wszystkim do osadzania danych o wysokiej wymiarowości w przestrzeni o niższej wymiarowości. UMAP zachowuje lokalną i globalną strukturę danych, co czyni go potężnym narzędziem do wizualizacji i analizy.

Podstawowa idea UMAP obejmuje:

- **Budowę grafu w wysokowymiarowej przestrzeni:** Graf jest tworzony na podstawie podobieństwa pomiędzy punktami danych, mierzonego przy użyciu metryki odległości (np. odległości euklidesowej lub cosinusowej).
- **Optymalizację w przestrzeni o niższej wymiarowości:** UMAP dąży do zachowania relacji z tego grafu podczas projekcji danych do przestrzeni o niższej wymiarowości (np. 2D lub 3D).

UMAP jest szczególnie skuteczny w wizualizacji złożonych danych o wysokiej wymiarowości, ponieważ równoważy zachowanie zarówno lokalnej, jak i globalnej struktury, co uwidacznia klastery i wzorce w danych.

B.2 Zrozumienie Wizualizacji UMAP

Kiedy UMAP redukuje dane z przestrzeni o wysokiej wymiarowości (np. 100 cech) do przestrzeni o niższej wymiarowości (np. 2D), uzyskana wizualizacja przedstawia:

- **Klastery:** Punkty bliskie sobie w przestrzeni o wysokiej wymiarowości często tworzą klastery na wykresie 2D lub 3D. Klastery te mogą reprezentować grupy lub kategorie w danych, takie jak różne klasy lub typy obserwacji.
- **Odległości:** Względne odległości pomiędzy klastrami wskazują na stopień podobieństwa pomiędzy nimi. Klastery bliżej siebie na wykresie UMAP są prawdopodobnie bardziej podobne w przestrzeni o wysokiej wymiarowości.
- **Kształt i Struktura:** UMAP zachowuje topologię danych w miarę możliwości, co oznacza, że kształty i wzorce w wizualizacji często odzwierciedlają istotne relacje w oryginalnym zbiorze danych.

B.3 Zastosowania Wizualizacji UMAP

Wizualizacje UMAP są szeroko stosowane w:

- **Eksploracyjnej Analizie Danych:** Aby identyfikować wzorce, klastery i punkty odstające w danych o wysokiej wymiarowości.
- **Uczeniu Nienadzorowanym:** Do oceny skuteczności algorytmów klasteryzacji lub zrozumienia struktury danych bez etykiet.
- **Uczeniu Nadzorowanym:** Do wizualizacji separacji pomiędzy klasami w problemach klasyfikacji.

B.4 Podsumowanie

Wizualizacje UMAP są cennym narzędziem do interpretacji danych o wysokiej wymiarowości, dostarczając intuicyjnego, niskowymiarowego przedstawienia. Poprzez zachowanie zarówno lokalnych sąsiedztw, jak i globalnych relacji, UMAP pozwala wizualnie wykrywać istotne struktury i związki w złożonych zbiorach danych.

C Analiza ANOVA

C.1 Czym jest ANOVA?

ANOVA (Analysis of Variance) to statystyczna metoda służąca do porównywania średnich wartości w więcej niż dwóch grupach. Jej głównym celem jest określenie, czy istnieją statystycznie istotne różnice pomiędzy średnimi w różnych grupach.

Metoda ta opiera się na dekompozycji wariancji i porównuje wariancję wewnątrz grup (ang. *within-group variance*) z wariancją pomiędzy grupami (ang. *between-group variance*).

C.2 Założenia metody ANOVA

Przed zastosowaniem ANOVA należy upewnić się, że dane spełniają następujące założenia:

- Dane w każdej grupie są niezależne.
- Dane w każdej grupie mają rozkład normalny.
- Wariancje pomiędzy grupami są homogeniczne (jednorodne).

C.3 Funkcja testowa i hipotezy

W ANOVA oblicza się wartość statystyki F jako stosunek wariancji pomiędzy grupami do wariancji wewnątrz grup:

$$F = \frac{\text{Wariancja pomiędzy grupami}}{\text{Wariancja wewnątrz grup}}$$

Hipotezy testowe:

- Hipoteza zerowa (H_0): Wszystkie grupy mają tę samą średnią, tj. nie ma różnic pomiędzy średnimi.

- Hipoteza alternatywna (H_1): Przynajmniej jedna grupa ma średnią różniącą się od innych.

C.4 Funkcja `anova_lm` w Pythonie

W Pythonie do przeprowadzania analizy ANOVA można użyć funkcji `anova_lm` z pakietu `statsmodels`. Funkcja ta przyjmuje jako argumenty dopasowany model regresji liniowej i zwraca tabelę ANOVA, która zawiera następujące informacje:

- **sum_sq** (Sum of Squares): Suma kwadratów dla każdego źródła wariancji (np. model, reszty).
- **df** (Degrees of Freedom): Liczba stopni swobody dla każdego źródła wariancji.
- **mean_sq** (Mean Squares): Średnia kwadratów, obliczana jako stosunek sumy kwadratów do stopni swobody.
- **F-statistic**: Wartość statystyki F, wskazująca stosunek wariancji pomiędzy grupami do wariancji wewnątrz grup.
- **PR(>F)**: Wartość p (ang. *p-value*), która określa, czy statystyka F jest statystycznie istotna.

C.5 Interpretacja wyników funkcji `anova_lm`

Po uzyskaniu wyników w tabeli ANOVA należy zwrócić szczególną uwagę na:

- **Wartość p (PR(>F))**: Jeśli wartość p jest mniejsza od ustalonego poziomu istotności (np. $\alpha = 0.05$), odrzucamy hipotezę zerową. Oznacza to, że istnieją istotne różnice pomiędzy średnimi grup.
- **Wartość F**: Wyższa wartość F sugeruje większą różnicę pomiędzy grupami w stosunku do wariancji wewnątrz grup.

Przykładowo:

- Jeśli $p < 0.05$: Możemy stwierdzić, że różnice między grupami są statystycznie istotne.
- Jeśli $p \geqslant 0.05$: Nie mamy dowodów na odrzucenie hipotezy zerowej, co oznacza brak istotnych różnic pomiędzy grupami.

C.6 Przykład w Pythonie

```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm

# Dane przykładowe
data = pd.DataFrame({
    'Group': ['A', 'A', 'B', 'B', 'C', 'C'],
    'Value': [5, 7, 8, 9, 6, 10]
})

# Dopasowanie modelu
model = ols('Value ~ Group', data=data).fit()

# Tabela ANOVA
anova_results = anova_lm(model)
print(anova_results)
```

C.7 Podsumowanie

ANOVA jest potężnym narzędziem do analizy różnic między grupami, a funkcja `anova_lm` w Pythonie pozwala na łatwe przeprowadzenie tej analizy. Kluczową rolę w interpretacji wyników odgrywają wartości p oraz F, które wskazują, czy różnice pomiędzy grupami są istotne statystycznie.