Wizualizacja Danych za Pomocą Biblioteki Matplotlib

Laboratorium z Analizy Danych

Contents

1 Wprowadzenie

Celem tego tutorialu jest zapoznanie się z biblioteką Matplotlib w języku Python, która jest popularnym narzędziem do tworzenia wykresów i wizualizacji danych. Matplotlib oferuje szeroki zakres możliwości, od prostych wykresów słupkowych po bardziej złożone wykresy 3D. Praca z wizualizacjami jest kluczowa dla analizy i interpretacji danych.

2 Instalacja Matplotlib

Przed rozpoczęciem pracy upewnij się, że masz zainstalowaną bibliotekę Matplotlib. Możesz ją zainstalować za pomocą poniższej komendy:

```
pip install matplotlib
```

3 Podstawy Matplotlib

Matplotlib najczęściej używa się wraz z modułem `pyplot`. Moduł ten oferuje zestaw funkcji do tworzenia wykresów podobnych do funkcji z Matlab.

```
import matplotlib.pyplot as plt
```

4 Podstawowe Typy Wykresów

4.1 Wykres Liniowy

Wykresy liniowe są używane do wizualizacji zmian wartości na osi czasu lub w zależności od zmiennej.

```
import matplotlib.pyplot as plt

# Dane
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Tworzenie wykresu
plt.plot(x, y, marker='o', linestyle='-', color='b', label="Liniowy")
plt.xlabel("O   X")
plt.ylabel("O   Y")
```

```
plt.title("Wykres Liniowy")
plt.legend()
plt.show()
```

4.2 Wykres Słupkowy

Wykresy słupkowe są używane do porównywania wartości między różnymi kategoriami.

```
# Dane
kategorie = ['A', 'B', 'C', 'D']
wartosci = [5, 7, 3, 8]

# Tworzenie wykresu s upkowego
plt.bar(kategorie, wartosci, color='orange')
plt.xlabel("Kategorie")
plt.ylabel("Warto ci")
plt.title("Wykres S upkowy")
plt.show()
```

4.3 Histogram

Histogramy są używane do przedstawiania rozkładu danych liczbowych.

```
import numpy as np

# Dane
dane = np.random.normal(0, 1, 1000)

# Tworzenie histogramu
plt.hist(dane, bins=30, color='purple', alpha=0.7)
plt.xlabel("Warto ci")
plt.ylabel("Cz stotliwo  ")
plt.title("Histogram")
plt.show()
```

4.4 Wykres Kołowy

Wykresy kołowe są używane do przedstawienia procentowego udziału kategorii.

```
# Dane
kategorie = ['A', 'B', 'C', 'D']
wartosci = [15, 30, 45, 10]

# Tworzenie wykresu ko owego
plt.pie(wartosci, labels=kategorie, autopct='%1.1f%%', startangle=90)
```

```
plt.title("Wykres Ko owy")
plt.show()
```

5 Dostosowywanie Wykresów

5.1 Kolory, Linie i Style Markerów

Matplotlib pozwala na łatwe dostosowanie kolorów, stylów linii i markerów. Przykład:

```
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]

plt.plot(x, y, color='green', marker='x', linestyle='--', linewidth=2)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Dostosowany Wykres Liniowy")
plt.show()
```

5.2 Dodawanie Siatki

Siatka pomaga w odczytywaniu wartości na wykresie. Można ją dodać za pomocą funkcji `plt.grid()`.

```
plt.plot(x, y, color='blue', marker='o')
plt.grid(True)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Wykres z Siatk ")
plt.show()
```

5.3 Dodawanie Adnotacji

Adnotacje pozwalają dodać opisy lub oznaczenia do punktów na wykresie.

```
plt.plot(x, y, marker='o')
plt.annotate('Najwy szy punkt', xy=(5, 35), xytext=(3, 30),
             arrowprops=dict(facecolor='black', arrowstyle='->'))
plt.title("Wykres z Adnotacj ")
plt.show()
```

6 Zaawansowane Typy Wykresów

6.1 Wykres Punktowy (Scatter Plot)

Wykresy punktowe są przydatne do analizy zależności między dwoma zmiennymi.

```
x = np.random.rand(50)
y = np.random.rand(50)
sizes = 1000 * np.random.rand(50)
colors = np.random.rand(50)

plt.scatter(x, y, s=sizes, c=colors, alpha=0.5, cmap='viridis')
plt.colorbar()
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Wykres Punktowy")
plt.show()
```

6.2 Wykres 3D

Matplotlib umożliwia tworzenie wykresów 3D przy użyciu modułu `Axes3D`.

```
from mpl_toolkits.mplot3d import Axes3D
```

```
2
3  fig = plt.figure()
4  ax = fig.add_subplot(111, projection='3d')
5
6  x = np.linspace(-5, 5, 100)
7  y = np.linspace(-5, 5, 100)
8  X, Y = np.meshgrid(x, y)
9  Z = np.sin(np.sqrt(X**2 + Y**2))
10
11 ax.plot_surface(X, Y, Z, cmap='plasma')
12 plt.title("Wykres 3D")
13 plt.show()
```

7 Podsumowanie `Matplotlib`

Biblioteka Matplotlib oferuje wiele możliwości wizualizacji danych, od prostych wykresów liniowych po zaawansowane wykresy 3D. Zrozumienie podstawowych typów wykresów i sposobu ich dostosowywania umożliwia lepszą interpretację danych oraz komunikację wyników. Zachęcamy do dalszej eksploracji Matplotlib oraz eksperymentowania z różnymi rodzajami wykresów.

8 Wizualizacja Danych za Pomocą Biblioteki Plotly

8.1 Wprowadzenie do `plotly`

Plotly to potężna biblioteka do wizualizacji danych w języku Python, która pozwala tworzyć interaktywne wykresy o wysokiej jakości. Jest szczególnie przydatna w analizie danych, ponieważ oferuje szeroką gamę typów wykresów, takich jak wykresy liniowe, słupkowe, kołowe, wykresy 3D, mapy geoprzestrzenne i wiele innych.

8.2 Instalacja Plotly

Przed rozpoczęciem pracy z Plotly należy upewnić się, że biblioteka jest zainstalowana. Możesz ją zainstalować za pomocą poniższej komendy:

```
pip install plotly
```

9 Podstawy Plotly

Plotly oferuje dwa główne sposoby tworzenia wykresów:

- `plotly.express` — do szybkiego tworzenia wykresów opartego na uproszczonej składni.
- `plotly.graph_objects` — umożliwia bardziej zaawansowane i złożone ustawienia wykresów.

10 Tworzenie Podstawowych Wykresów

10.1 Wykres Liniowy

Wykresy liniowe są używane do prezentacji zmian w wartościach ciągłych. Plotly pozwala łatwo stworzyć wykresy liniowe za pomocą modułu `plotly.express`.

```
import plotly.express as px

# Dane
x = [1, 2, 3, 4, 5]
y = [10, 15, 13, 17, 10]

# Tworzenie wykresu
fig = px.line(x=x, y=y, title="Wykres Liniowy", labels={'x':"O   X", 'y':"O   Y"})
fig.show()
```

10.2 Wykres Słupkowy

Wykresy słupkowe są przydatne do porównania danych między kategoriami.

```
# Dane
kategorie = ['A', 'B', 'C', 'D']
```

```
wartosci = [4, 7, 1, 8]

# Tworzenie wykresu s upkowego
fig = px.bar(x=kategorie, y=wartosci, title="Wykres S upkowy", labels={'x':"Kategorie", 'y':"Warto ci"})
fig.show()
```

10.3 Wykres Kołowy

Wykresy kołowe są używane do wizualizacji procentowego udziału kategorii.

```
# Dane
kategorie = ['A', 'B', 'C', 'D']
wartosci = [15, 30, 45, 10]

# Tworzenie wykresu ko owego
fig = px.pie(names=kategorie, values=wartosci, title="Wykres Ko owy")
fig.show()
```

10.4 Histogram

Histogramy są używane do wizualizacji rozkładu danych.

```
import numpy as np

# Dane
dane = np.random.normal(0, 1, 1000)

# Tworzenie histogramu
fig = px.histogram(x=dane, nbins=30, title="Histogram", labels={'x':"Warto ci", 'y':"Cz stotliwo   "})
fig.show()
```

11 Dostosowywanie Wykresów

11.1 Kolory i Styl

Plotly umożliwia dostosowanie koloru wykresu oraz stylów poprzez parametry w funkcjach.

```
fig = px.line(x=x, y=y, title="Wykres Liniowy z Dostosowanymi Kolorami")
fig.update_traces(line=dict(color="purple", width=4))
fig.show()
```

11.2 Dodawanie Adnotacji

Adnotacje pozwalają na wyróżnienie określonych punktów na wykresie.

```
fig = px.line(x=x, y=y, title="Wykres z Adnotacjami")
fig.add_annotation(x=3, y=13, text="Wyr  niony Punkt", showarrow=True, arrowhead=1)
fig.show()
```

12 Tworzenie Zaawansowanych Wykresów

12.1 Wykres Punktowy (Scatter Plot)

Wykresy punktowe są przydatne do wizualizacji zależności między dwoma zmiennymi.

```
# Dane
x = np.random.rand(50)
y = np.random.rand(50)
sizes = 1000 * np.random.rand(50)
colors = np.random.rand(50)

fig = px.scatter(x=x, y=y, size=sizes, color=colors, title="Wykres Punktowy")
fig.show()
```

12.2 Wykres 3D

Plotly umożliwia tworzenie wykresów 3D, które mogą być szczególnie przydatne do wizualizacji zależności trójwymiarowych.

```
import plotly.graph_objects as go

# Dane
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

# Tworzenie wykresu 3D
fig = go.Figure(data=[go.Surface(z=Z, x=X, y=Y)])
fig.update_layout(title="Wykres 3D", scene=dict(
                    xaxis_title='X',
                    yaxis_title='Y',
                    zaxis_title='Z'))
fig.show()
```

13 Tworzenie Wykresów z Subplotami

W Plotly można tworzyć wykresy z subplotami, aby jednocześnie przedstawić różne dane.

```
from plotly.subplots import make_subplots

# Dane
x = [1, 2, 3, 4, 5]
y1 = [10, 15, 13, 17, 10]
y2 = [5, 6, 2, 3, 8]

# Tworzenie subplot w
fig = make_subplots(rows=1, cols=2, subplot_titles=("Wykres 1", "Wykres 2"))
fig.add_trace(go.Scatter(x=x, y=y1, mode='lines+markers', name="Liniowy 1"), row=1, col=1)
fig.add_trace(go.Scatter(x=x, y=y2, mode='lines+markers', name="Liniowy 2"), row=1, col=2)
fig.update_layout(title="Subploty w Plotly")
fig.show()
```

14 Podsumowanie `plotly`

Biblioteka Plotly oferuje zaawansowane możliwości interaktywnej wizualizacji danych, które są użyteczne zarówno w analizie danych, jak i prezentacji wyników. Plotly pozwala na łatwe tworzenie atrakcyjnych i dynamicznych wykresów, które można dostosować do specyficznych

potrzeb. Dalsze zgłębianie możliwości tej biblioteki oraz eksperymentowanie z różnymi wykresami pomoże w lepszym zrozumieniu i prezentacji danych.

15 Wprowadzenie do biblioteki `plotnine`

Biblioteka `plotnine` jest inspirowana biblioteką `ggplot2` z języka R, która opiera się na filozofii tzw. "Grammar of Graphics". Umożliwia ona tworzenie eleganckich i czytelnych wizualizacji w Pythonie, zachowując składnię podobną do `ggplot2`. `plotnine` jest szczególnie przydatna do eksploracyjnej analizy danych i tworzenia różnorodnych typów wykresów.

15.1 Instalacja Biblioteki plotnine

Aby korzystać z `plotnine`, należy najpierw ją zainstalować. Można to zrobić za pomocą poniższej komendy:

```
pip install plotnine
```

15.2 Podstawy plotnine

Podobnie jak w `ggplot2`, tworzenie wykresów w `plotnine` opiera się na komponowaniu różnych elementów wykresu, takich jak dane, estetyki oraz geomy (elementy graficzne wykresu). Wszystkie wykresy zaczynają się od funkcji `ggplot()`.

```
from plotnine import *
import pandas as pd
```

```

# Przykładowe dane
data = pd.DataFrame({
    'x': [1, 2, 3, 4, 5],
    'y': [2, 4, 6, 8, 10]
})

# Tworzenie podstawowego wykresu
(ggplot(data) + aes(x='x', y='y') + geom_point())
```

15.3 Tworzenie Podstawowych Typów Wykresów

15.4 Wykres Punktowy (Scatter Plot)

Wykres punktowy jest używany do wizualizacji zależności między dwoma zmiennymi.

```
# Wykres punktowy
(ggplot(data) + aes(x='x', y='y') + geom_point() +
 ggtitle("Wykres Punktowy") +
 xlab("O   X") + ylab("O   Y"))
```

15.5 Wykres Liniowy

Wykres liniowy jest przydatny do przedstawienia trendów na przestrzeni danych ciągłych.

```
# Wykres liniowy
(ggplot(data) + aes(x='x', y='y') + geom_line(color='blue') +
 ggtitle("Wykres Liniowy") +
 xlab("O   X") + ylab("O   Y"))
```

15.6 Wykres Słupkowy

Wykresy słupkowe są stosowane do porównywania wartości między różnymi kategoriami.

```
# Dane do wykresu s upkowego
data_bar = pd.DataFrame({
    'kategorie': ['A', 'B', 'C', 'D'],
    'warto ci': [4, 7, 1, 8]
})

# Wykres s upkowy
(ggplot(data_bar) + aes(x='kategorie', y='warto ci') +
 geom_bar(stat='identity', fill='skyblue') +
 ggtitle("Wykres S upkowy") +
```

```
 xlab("Kategorie") + ylab("Warto ci"))
```

15.7 Histogram

Histogramy są używane do przedstawienia rozkładu danych.

```
import numpy as np

# Dane
data_hist = pd.DataFrame({
    'warto ci': np.random.normal(0, 1, 1000)
})

# Histogram
(ggplot(data_hist) + aes(x='warto ci') +
 geom_histogram(bins=30, fill='purple', alpha=0.7) +
 ggtitle("Histogram") +
 xlab("Warto ci") + ylab("Cz stotliwo  "))
```

16 Dostosowywanie Wykresów

16.1 Kolory i Style

Możemy dostosować kolory punktów, linii oraz styl wykresów za pomocą różnych parametrów w funkcjach geom.

```
# Wykres punktowy z dostosowanymi kolorami
(ggplot(data) + aes(x='x', y='y') +
 geom_point(color='red', size=3) +
 ggtitle("Wykres Punktowy z Dostosowanymi Kolorami") +
 xlab("O   X") + ylab("O   Y"))
```

16.2 Dodawanie Linii Trendu

Linia trendu jest użyteczna do analizy wzorców w danych punktowych.

```
# Wykres punktowy z lini  trendu
(ggplot(data) + aes(x='x', y='y') +
 geom_point() + geom_smooth(method='lm') +
 ggtitle("Wykres z Lini  Trendu") +
 xlab("O   X") + ylab("O   Y"))
```

17 Zaawansowane Wykresy

17.1 Facetowanie Wykresów

Facetowanie umożliwia podział danych na podgrupy i tworzenie osobnych wykresów dla każdej z nich.

```
# Przyk adowe dane
data_facet = pd.DataFrame({
    'x': np.tile([1, 2, 3, 4], 2),
    'y': [1, 2, 3, 4, 2, 3, 4, 5],
    'grupa': ['A']*4 + ['B']*4
})

# Wykres z facetowaniem
(ggplot(data_facet) + aes(x='x', y='y') +
 geom_point() + facet_wrap('~grupa') +
 ggtitle("Wykres z Facetowaniem") +
 xlab(" O   X") + ylab(" O   Y"))
```

17.2 Wykres Pudłowy (Box Plot)

Wykres pudłowy przedstawia rozkład danych i pomaga w identyfikacji wartości odstających.

```
# Przyk adowe dane
data_box = pd.DataFrame({
    'kategorie': np.random.choice(['A', 'B', 'C'], 100),
    'warto ci': np.random.randn(100)
})

# Wykres pud owy
(ggplot(data_box) + aes(x='kategorie', y='warto ci') +
 geom_boxplot(fill='lightblue') +
 ggtitle("Wykres Pud owy") +
 xlab("Kategorie") + ylab("Warto ci"))
```

18 Dodawanie Motywów do Wykresów

Plotnine oferuje różne motywy, które można dodać do wykresów, aby zmienić ich styl.

```
# Dodanie motywu
(ggplot(data) + aes(x='x', y='y') + geom_point() +
 ggtitle("Wykres z Motywem") +
 theme_minimal())
```

19 Podsumowanie

Biblioteka `plotnine` to potężne narzędzie do tworzenia wizualizacji w Pythonie. Dzięki składni inspirowanej `ggplot2`, pozwala na szybkie i efektywne tworzenie wykresów o wysokiej jakości. Zachęcamy do dalszej eksploracji `plotnine` oraz eksperymentowania z różnymi rodzajami wykresów.

20 Wizualizacja Geoprzestrzenna Danych za Pomocą `Plotly` w Pythonie

20.1 Wprowadzenie

Biblioteka `Plotly` umożliwia tworzenie interaktywnych map geoprzestrzennych w Pythonie. Mapy te mogą być używane do wizualizacji lokalizacji, rozmieszczenia punktów, a także do przedstawienia danych o zmiennych kolorach na mapie (choropleth). W tym tutorialu omówimy, jak używać `Plotly` do nakładania danych na mapy.

20.2 Instalacja Biblioteki Plotly

Jeśli jeszcze nie zainstalowałeś biblioteki `Plotly`, możesz to zrobić za pomocą poniższej komendy:

```
pip install plotly
```

21 Mapy Punktowe (Scatter Maps)

Mapy punktowe są używane do wyświetlania lokalizacji punktów na mapie na podstawie ich szerokości i długości geograficznej.

21.1 Przykład: Wyświetlanie Lokalizacji Miast na Mapie

W tym przykładzie na mapie zaznaczymy lokalizacje kilku miast za pomocą punktów.

```
import plotly.express as px
import pandas as pd

# Przyk adowe dane miast z ich wsp  rz dnymi
data = pd.DataFrame({
    'city': ['New York', 'London', 'Tokyo', 'Sydney'],
    'latitude': [40.7128, 51.5074, 35.6895, -33.8688],
    'longitude': [-74.0060, -0.1278, 139.6917, 151.2093]
})

# Tworzenie mapy punktowej
fig = px.scatter_mapbox(data, lat="latitude", lon="longitude", hover_name="city",
                        zoom=1, height=500)

# Konfiguracja stylu mapy
```

```
fig.update_layout(mapbox_style="open-street-map")
fig.update_layout(title="Lokalizacja Wybranych Miast na Mapie")
fig.show()
```

W tym przykładzie wykorzystujemy funkcję `px.scatter_mapbox`, aby nałożyć punkty reprezentujące miasta na mapę. Użyliśmy stylu mapy `open-street-map`, ale `Plotly` oferuje również inne style map.

22 Mapy Choropleth (Mapy o Zmiennych Kolorach Obszarów)

Mapy choropleth są używane do reprezentowania danych związanych z regionami, gdzie kolor obszaru reprezentuje wartości zmiennej.

22.1 Przykład: Wizualizacja Populacji Krajów na Mapie Świata

W tym przykładzie na mapie świata przedstawimy dane o populacji różnych krajów, gdzie intensywność koloru wskazuje wielkość populacji.

```
import plotly.express as px

# Dane kraj w z ich kodami ISO oraz populacj
data = px.data.gapminder().query("year == 2007")

# Tworzenie mapy choropleth
fig = px.choropleth(data, locations="iso_alpha", color="pop",
                    hover_name="country", color_continuous_scale="Viridis",
```

```
                    labels={'pop': 'Populacja'})

# Ustawienia mapy
fig.update_layout(title="Populacja Kraj w na  wiecie  w 2007 roku")
fig.show()
```

W tym kodzie używamy gotowego zestawu danych `gapminder` dostępnego w `Plotly`, który zawiera dane o populacji krajów. Funkcja `px.choropleth` generuje mapę choropleth, a kolor zmienia się zgodnie z wartościami zmiennej populacji.

23 Mapy Cieplne (Heatmaps)

Mapy cieplne mogą być używane do przedstawiania gęstości występowania punktów w określonych obszarach.

23.1 Przykład: Gęstość Lokalizacji na Mapie

Poniżej przykład mapy cieplnej, która przedstawia gęstość rozmieszczenia punktów.

```
import plotly.graph_objects as go

# Przyk adowe dane z losowymi wsp  rz dnymi
import numpy as np
np.random.seed(0)
lats = np.random.normal(37.7749, 0.5, 100) #  rednia  szeroko  geogr.
```

```
lons = np.random.normal(-122.4194, 0.5, 100) #   rednia   d  u g o     geogr.

# Tworzenie mapy cieplnej
fig = go.Figure(go.Densitymapbox(lat=lats, lon=lons, radius=10))

# Ustawienia stylu mapy
fig.update_layout(mapbox_style="stamen-terrain", mapbox_center_lon=-122.4194,
                  mapbox_center_lat=37.7749, mapbox_zoom=10)
fig.update_layout(title="Mapa Cieplna Przedstawiaj ca  G  s t o     Punkt w")
fig.show()
```

W tym przykładzie użyliśmy funkcji `Densitymapbox` do stworzenia mapy cieplnej, w której dane o szerokości i długości geograficznej określają położenie punktów. Parametr `radius` kontroluje wielkość rozmycia ciepła wokół punktów.

23.2 Mapy z Danymi na Poziomie Stanów

‘Plotly‘ pozwala również na wizualizację danych specyficznych dla regionów wewnątrz kraju (np. stanów USA).

23.3 Przykład: Mapy Stanów w USA

Poniższy przykład ilustruje mapę stanów USA z danymi populacyjnymi.

```
import plotly.express as px
```

```
2
3  # Dane stan w USA z populacj
4  data = px.data.gapminder().query("year == 2007 & continent == 'Americas'")
5  data = data[data['country'] != 'Canada']  # Usuni cie Kanady, je li potrzebne
6
7  # Tworzenie mapy choropleth dla stan w USA
8  fig = px.choropleth(data, locations="iso_alpha", color="pop",
9                      hover_name="country", locationmode="USA-states",
10                     color_continuous_scale="Viridis", labels={'pop': 'Populacja'})
11
12 # Ustawienia mapy
13 fig.update_layout(title="Populacja Stan w USA", geo_scope='usa')
14 fig.show()
```

24 Podsumowanie - mapy geoprzestrzenne

Biblioteka `Plotly` umożliwia tworzenie interaktywnych map w Pythonie, co jest przydatne do eksploracji i analizy danych geoprzestrzennych. Omówiliśmy różne typy map, takie jak mapy punktowe, choropleth, cieplne i mapy stanów. Te mapy można dostosowywać i integrować z różnorodnymi danymi, co czyni `Plotly` potężnym narzędziem do wizualizacji danych.

25 Zadanie

Zadanie dotyczy tworzenia wszystkich możliwych wykresów w celu eksploracji zbioru danych