SPRAWOZDANIE

Laboratoria 3 – EDA w Streamlit

KRZYSZTOF KOWALSKI 407142

Cały kod Python wykorzystany w tych laboratoriach znajduje się na końcu sprawozdania. W sprawozdaniu korzystane będzie z fragmentów implementujących opisywaną funkcjonalność

Całość pracy rozpoczęto od instalacji oraz zaimportowania wymaganych bibliotek.

```
import streamlit as st
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
import matplotlib.pyplot as plt
import seaborn as sns
```

Następnie należało wybrać zbiór danych wykorzystany do przeprowadzenia procesu EDA. Skorzystano z dostępnego w bibliotece **scikit-learn** zbioru danych **fetch_california_housing** zawierającego informacje o domach w Kalifornii.

```
def read_data():
    housing = fetch_california_housing()
    df = pd.DataFrame(housing.data, columns=housing.feature_names)
    df['PRICE'] = housing.target
    return df
```

Następnie można było przejść do pierwszej funkcjonalności streamlit. Była to możliwość wyświetlenia 5 pierwszych wierszy danych:

Eksploracyjna Analiza Danych (EDA) - Ceny Mieszkań w Kalifornii

Dane - podstawowe info

☑ Wyświetl 5 pierwszych rekordów danych

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	PRICE
0	8.3252	41	6.9841	1.0238	322	2.5556	37.88	-122.23	4.526
1	8.3014	21	6.2381	0.9719	2,401	2.1098	37.86	-122.22	3.585
2	7.2574	52	8.2881	1.0734	496	2.8023	37.85	-122.24	3.521
3	5.6431	52	5.8174	1.0731	558	2.5479	37.85	-122.25	3.413
4	3.8462	52	6.2819	1.0811	565	2.1815	37.85	-122.25	3.422

Oraz podstawowych statystyk o danych:

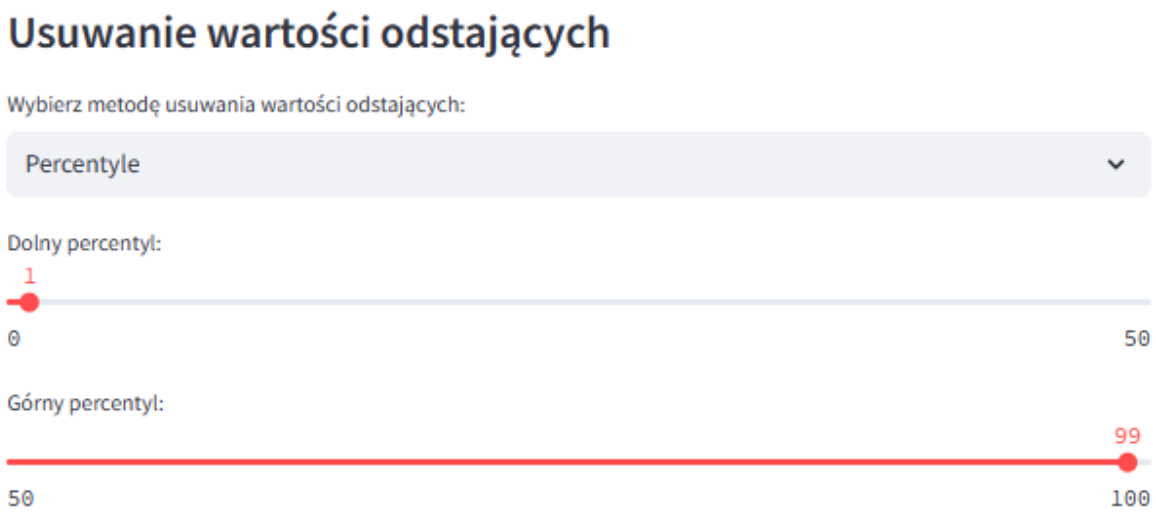
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	PRICE
count	20,640	20,640	20,640	20,640	20,640	20,640	20,640	20,640	20,640
mean	3.8707	28.6395	5.429	1.0967	1,425.4767	3.0707	35.6319	-119.5697	2.0686
std	1.8998	12.5856	2.4742	0.4739	1,132.4621	10.386	2.136	2.0035	1.154
min	0.4999	1	0.8462	0.3333	3	0.6923	32.54	-124.35	0.15
25%	2.5634	18	4.4407	1.0061	787	2.4297	33.93	-121.8	1.196
50%	3.5348	29	5.2291	1.0488	1,166	2.8181	34.26	-118.49	1.797
75%	4.7433	37	6.0524	1.0995	1,725	3.2823	37.71	-118.01	2.6472
max	15.0001	52	141.9091	34.0667	35,682	1,243.3333	41.95	-114.31	5

Jeśli użytkownik zaznaczy checkbox. Kod, który odpowiadał za tą funkcjonalność:

```
def basic_info(df):
    st.subheader('Dane - podstawowe info')
    if st.checkbox('Wyświetl 5 pierwszych rekordów danych'):
        st.write(df.head())
    if st.checkbox("Wyświetl podstawowe statystyki"):
        st.write(df.describe())
```

Drugą z opcji była możliwość usunięcia wartości odstających, korzystając z trzech znanych metod:

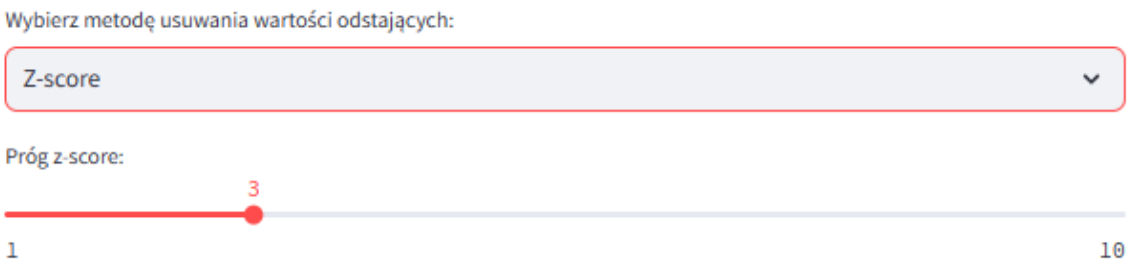
- percyntyle: użytkownik wybiera dolny oraz górny percentyl odcięcia wartości odstających




```
def remove_outliers_percentile(data, lower_percentile=1,
upper_percentile=99):
    lower_bound = np.percentile(data, lower_percentile)
```

```
    upper_bound = np.percentile(data, upper_percentile)
    return data[(data >= lower_bound) & (data <= upper_bound)]
```

- z-score: użytkownik wybiera z-score




```
def remove_outliers_zscore(data, threshold=3):
    z_scores = (data - data.mean()) / data.std()
    return data[abs(z_scores) < threshold]
```

- współczynnik iqr

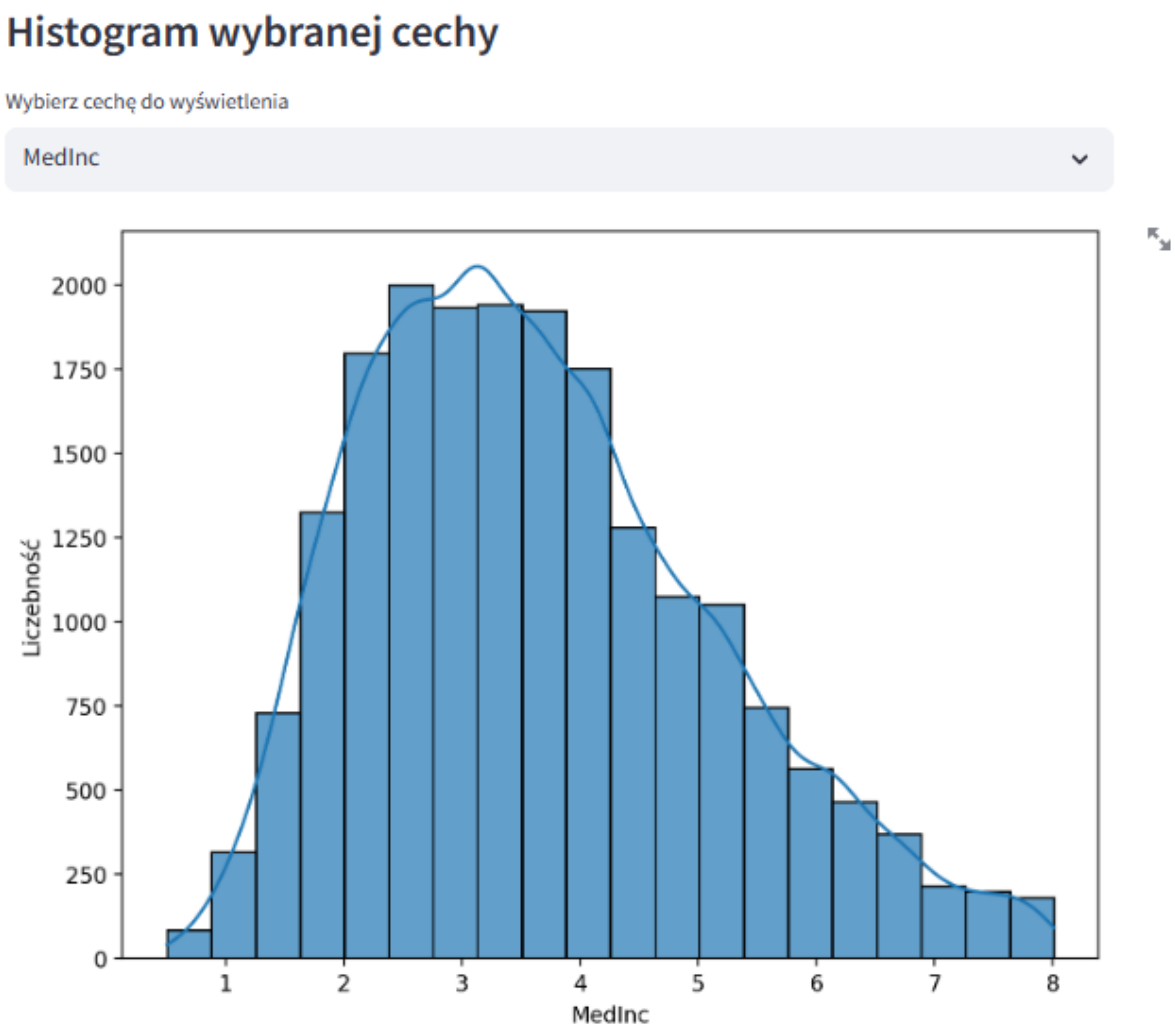

```
def remove_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    return data[(data >= (Q1 - 1.5 * IQR)) & (data <= (Q3 + 1.5 * IQR))]
```

Oraz kod odpowiedzialny za stworzenie wyglądu w streamlit oraz obsługę wyboru metody przez użytkownika:

```
def removing_outliers(df):
    st.subheader("Usuwanie wartości odstających")
    outliers_method = st.selectbox("Wybierz metodę usuwania wartości
odstających:",
                                   ['Percentyle', 'Z-score', 'Współczynnik
IQR'])
    if outliers_method == 'Percentyle':
        lower_percentile = st.slider("Dolny percentyl:", min_value=0,
max_value=50, value=1)
        upper_percentile = st.slider("Górny percentyl:", min_value=50,
max_value=100, value=99)
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_percentile(df[feature],
lower_percentile, upper_percentile)
    elif outliers_method == 'Z-score':
        threshold = st.slider("Próg z-score:", min_value=1, max_value=10,
value=3)
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_zscore(df[feature],
threshold)
    else:
```

```
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_iqr(df[feature])
```

Kolejno dodano histogram wraz z krzywą. Istnieje możliwość doboru zmiennej dla której ma być przedstawiony histogram.

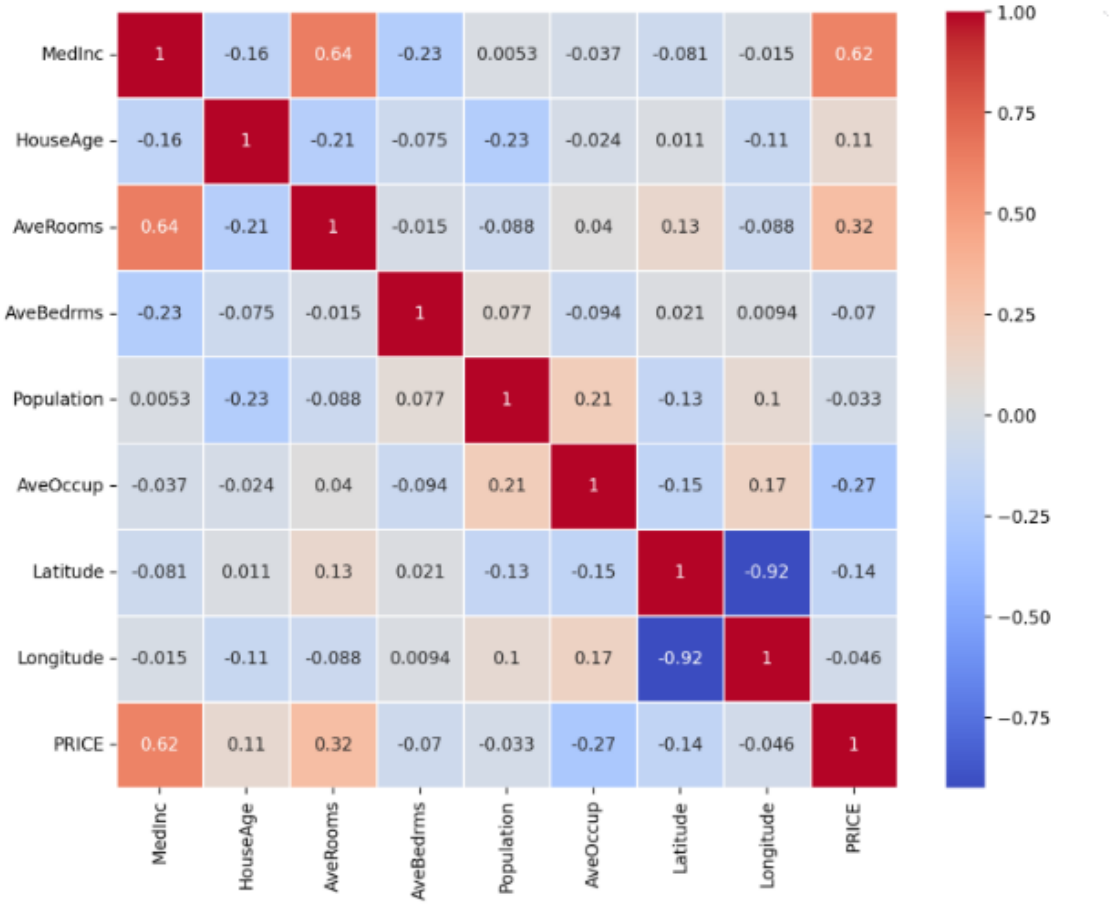


Kod Python:

```
def display_hist(df):
    st.subheader('Histogram wybranej cechy')
    selected_feature = st.selectbox('Wybierz cechę do wyświetlenia',
df.columns[:-1])
    fig = plt.figure(figsize=(8, 6))
    sns.histplot(df[selected_feature], bins=20, alpha=0.7, kde=True)
    plt.xlabel(selected_feature)
    plt.ylabel('Liczebność')
    st.pyplot(fig)
```

Następnie dodano macierz korelacji, wraz z liczbowym wyświetleniem wartości.

Macierz korelacji



Kod Python:

```
def display_corr_matrix(df):
    st.subheader('Macierz korelacji')
    fig = plt.figure(figsize=(10, 8))
    corr = df.corr()
    sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
    st.pyplot(fig)
    return corr
```

Ostatnią z funkcjonalności było dodanie wykresu punktowego, w którym użytkownik wybiera zmienną X i Y dla których takowy wykres ma zostać stworzony.

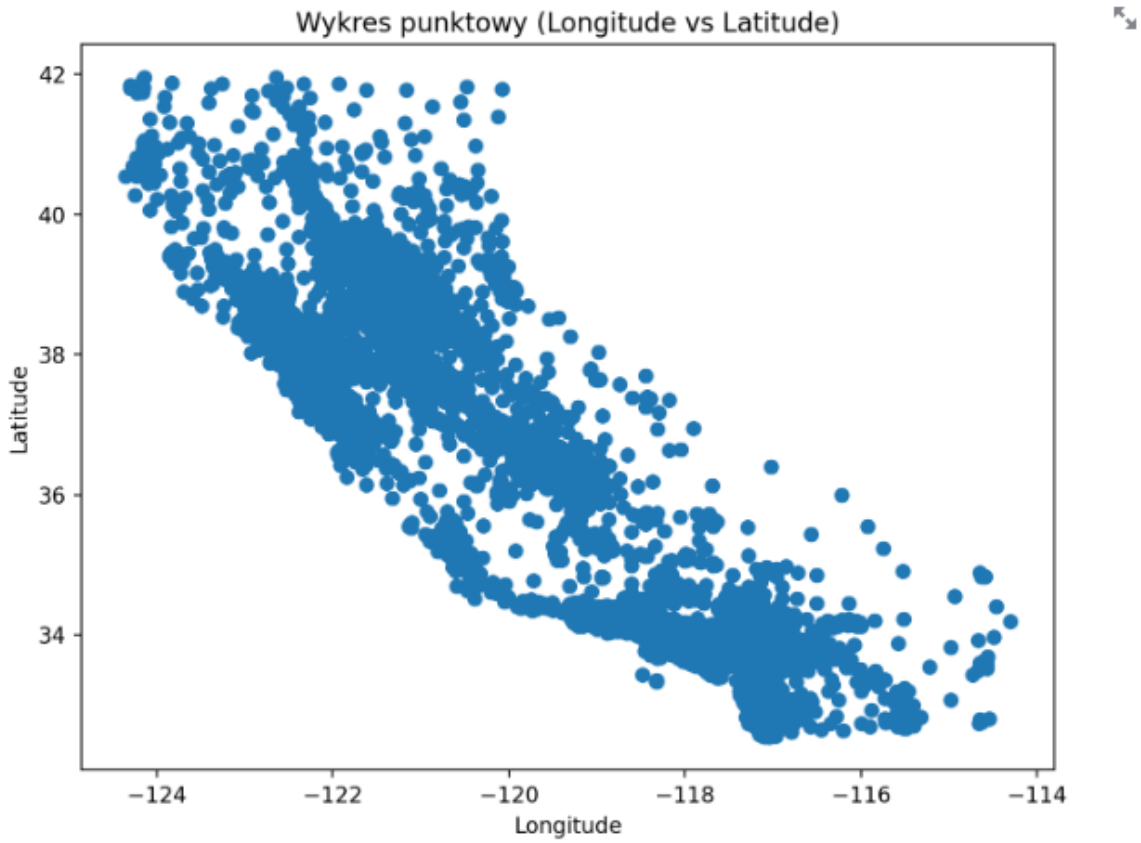
Wykres punktowy

Wybierz cechę do osi X:

Longitude

Wybierz cechę do osi Y:

Latitude



Wartość korelacji między **Longitude** a **Latitude**: **-0.92**

Kod Python:

```
def display_scatter(df):
    st.subheader('Wykres punktowy')
    x_feature = st.selectbox('Wybierz cechę do osi X:', df.columns[:-1])
    y_feature = st.selectbox('Wybierz cechę do osi Y:', df.columns[:-1])
    fig = plt.figure(figsize=(8, 6))
    plt.scatter(df[x_feature], df[y_feature])
    plt.xlabel(x_feature)
    plt.ylabel(y_feature)
    plt.title(f'Wykres punktowy ({x_feature} vs {y_feature})')
    st.pyplot(fig)
    return x_feature, y_feature
```

Poniżej wykresu punktowego wyświetla się również informacja o korelacji dla zmiennych, które wybrano do stworzenia wykresu (niebieskie zdanie na powyższej grafice). Kod Python, którym to zaimplementowano:

```
def display_corr_value(df, corr, x_feature, y_feature):
    correlation_value = corr.loc[x_feature, y_feature]
    st.write(f'<div style="text-align: center; font-size: 18px; color:
blue;">'
             f'Wartość korelacji między <b>{x_feature}</b> a
<b>{y_feature}</b>: '
             f'<span style="font-weight: bold; font-size:
20px;">{correlation_value:.2f}</span>'
             '</div>', unsafe_allow_html=True)
```

Podsumowując, streamlit pozwala na dokładną i interaktywną możliwość eksploracji zbioru danych. Dzięki wykorzystaniu interakcji użytkownika istnieje możliwość zbadania wykresów, zależności między kolejnymi cechami w zbiorze bez zmiany kodu. Wydaje się być to przejrzysty i łatwy w obsłudze dodatek podczas wgłębianiu się w zbiór danych.

Cały wykorzystany kod Python:

```
import streamlit as st
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
import matplotlib.pyplot as plt
import seaborn as sns

def read_data():
    housing = fetch_california_housing()
    df = pd.DataFrame(housing.data, columns=housing.feature_names)
    df['PRICE'] = housing.target
    return df

def basic_info(df):
    st.subheader('Dane - podstawowe info')
    if st.checkbox('Wyświetl 5 pierwszych rekordów danych'):
        st.write(df.head())
    if st.checkbox("Wyświetl podstawowe statystyki"):
        st.write(df.describe())

def removing_outliers(df):
    st.subheader("Usuwanie wartości odstających")
    outliers_method = st.selectbox("Wybierz metodę usuwania wartości
odstających:",
                                   ['Percentyle', 'Z-score', 'Współczynnik
IQR'])
    if outliers_method == 'Percentyle':
        lower_percentile = st.slider("Dolny percentyl:", min_value=0,
max_value=50, value=1)
        upper_percentile = st.slider("Górny percentyl:", min_value=50,
max_value=100, value=99)
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_percentile(df[feature],
lower_percentile, upper_percentile)
    elif outliers_method == 'Z-score':
        threshold = st.slider("Próg z-score:", min_value=1, max_value=10,
```

```
value=3)
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_zscore(df[feature],
threshold)
    else:
        for feature in df.columns:
            if feature != 'PRICE':  # Ignoruj kolumnę 'PRICE'
                df[feature] = remove_outliers_iqr(df[feature])

def display_hist(df):
    st.subheader('Histogram wybranej cechy')
    selected_feature = st.selectbox('Wybierz cechę do wyświetlenia',
df.columns[:-1])
    fig = plt.figure(figsize=(8, 6))
    sns.histplot(df[selected_feature], bins=20, alpha=0.7, kde=True)
    plt.xlabel(selected_feature)
    plt.ylabel('Liczebność')
    st.pyplot(fig)

def display_corr_matrix(df):
    st.subheader('Macierz korelacji')
    fig = plt.figure(figsize=(10, 8))
    corr = df.corr()
    sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
    st.pyplot(fig)
    return corr

def display_scatter(df):
    st.subheader('Wykres punktowy')
    x_feature = st.selectbox('Wybierz cechę do osi X:', df.columns[:-1])
    y_feature = st.selectbox('Wybierz cechę do osi Y:', df.columns[:-1])
    fig = plt.figure(figsize=(8, 6))
    plt.scatter(df[x_feature], df[y_feature])
    plt.xlabel(x_feature)
    plt.ylabel(y_feature)
    plt.title(f'Wykres punktowy ({x_feature} vs {y_feature})')
    st.pyplot(fig)
    return x_feature, y_feature

def display_corr_value(df, corr, x_feature, y_feature):
    correlation_value = corr.loc[x_feature, y_feature]
    st.write(f'<div style="text-align: center; font-size: 18px; color:
blue;">'
             f'Wartość korelacji między <b>{x_feature}</b> a
<b>{y_feature}</b>: '
             f'<span style="font-weight: bold; font-size:
20px;">{correlation_value:.2f}</span>'
             '</div>', unsafe_allow_html=True)
def remove_outliers_percentile(data, lower_percentile=1,
upper_percentile=99):
    lower_bound = np.percentile(data, lower_percentile)
    upper_bound = np.percentile(data, upper_percentile)
    return data[(data >= lower_bound) & (data <= upper_bound)]

def remove_outliers_zscore(data, threshold=3):
    z_scores = (data - data.mean()) / data.std()
    return data[abs(z_scores) < threshold]

def remove_outliers_iqr(data):
    Q1 = data.quantile(0.25)
```

```
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    return data[(data >= (Q1 - 1.5 * IQR)) & (data <= (Q3 + 1.5 * IQR))]
def main():
    data = read_data()
    st.title('Eksploracyjna Analiza Danych (EDA) - Ceny Mieszkań w
Kalifornii')
    basic_info(data)
    removing_outliers(data)
    display_hist(data)
    corr = display_corr_matrix(data)
    x, y = display_scatter(data)
    display_corr_value(data, corr, x, y)

if __name__ == "__main__":
    main()
```