

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:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Oraz podstawowych statystyk o danych:

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

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

Obraz zawierający tekst, zrzut ekranu, linia, Czcionka

Opis wygenerowany automatycznie

def remove\_outliers\_zscore(data, threshold=3):  
 z\_scores = (data - data.mean()) / data.std()  
 return data[abs(z\_scores) < threshold]

- współczynnik iqrObraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, diagram, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, zrzut ekranu, numer, Równolegle

Opis wygenerowany automatycznie

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.

Obraz zawierający tekst, mapa, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

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()