Kierunek studiów: Informatyka
Przedmiot: Big Data i hurtowanie danych II
Rok akademicki: 2023/2024

Analiza Importu Towarów z Ukrainy

Autorzy projektu:
Grzegorz Urban,
Dawid Kusion,
Przemysław Szewczyk,
Jakub Rynduch

Prowadzący:
mgr inż. Tomasz Potempa

Spis treści

1. Wprowadzenie

Celem tego projektu jest stworzenie aplikacji, która umożliwia użytkownikom analizę danych importowych towarów z Ukrainy oraz prognozowanie przyszłych wartości importu dla różnych towarów.

2. Opis Aplikacji

Aplikacja jest napisana w języku Python i wykorzystuje biblioteki do przetwarzania danych, wizualizacji oraz tworzenia modeli prognostycznych. Główne komponenty aplikacji obejmują:

a) Baza Danych

b) Analiza Danych

c) Prognozowanie

d) Wizualizacja

e) Interfejs Użytkownika

3. Struktura Plików

Aplikacja składa się z następujących plików:

1. main.py
2. Add_data_to_DataBase.py
3. db_config.py
4. fav_item_data.py
5. fav_item_predictions.py
6. fav_items_all_year.py
7. sum_prediction.py
8. sum_data.py
9. ware_sum_prediction.py

4. Opis Plików

main.py

Ten plik jest głównym modułem aplikacji, który tworzy interfejs użytkownika za pomocą Tkinter. Zawiera funkcje obsługujące różne przyciski w interfejsie, takie jak przyciski do wyświetlania rzeczywistych danych, prognozowania wartości oraz wyświetlania najpopularniejszych towarów.

- **Interfejs Tkinter:** Tworzy główne okno aplikacji, dodaje przyciski, pola wyboru.
- **Funkcje przycisków:** Definiuje funkcje obsługujące kliknięcia przycisków, które uruchamiają odpowiednie moduły backendowe do generowania wykresów.

Add_data_to_DataBase.py

Ten plik zajmuje się dodawaniem danych do bazy danych. Używa biblioteki pandas do wczytywania danych z pliku CSV.

```python
import pandas as pd
from sqlalchemy import create_engine


# Interface implementation for datasource connection and data import
data = pd.read_csv( filepath_or_buffer: 'data/Import.csv', header=0, encoding='latin1')

data = data.apply(lambda x: x.str.encode('latin1').str.decode('utf-8') if x.dtype == "object" else x)


# Utwórz połączenie do bazy danych PostgreSQL
engine = create_engine('postgresql://2023_urban_grzegorz:35240@195.150.230.208:5432/2023_urban_grzegorz')

#data.to_sql('data', engine, schema='import_ukraine', if_exists='replace', index=False)
```

db_config.py

Plik konfiguracyjny bazy danych, zawiera funkcję get_db_engine(), która tworzy połączenie z bazą danych.

```python
import sqlalchemy

12 usages  Gosqu248 +1
def get_db_engine():
    # Utwórz połączenie do bazy danych PostgreSQL
    engine = sqlalchemy.create_engine('postgresql://2023_urban_grzegorz:35240@195.150.230.208:5432/2023_urban_grzegorz')
    return engine
```

fav_item_data.py

Zajmuje się analizą i wizualizacją najpopularniejszych towarów dla danego roku.

- **Analiza danych:** Grupuje dane według nazwy towaru i sumuje wartości.
- **Wizualizacja:** Tworzy wykresy kolumnowe dla top 5 towarów w wybranym roku.

fav_item_predictions.py

Moduł odpowiedzialny za prognozowanie wartości importu dla wybranych towarów na podstawie modelu SARIMA.

- **Prognozowanie:** Używa modelu SARIMA do prognozowania wartości importu na przyszłe lata.
- **Wizualizacja:** Tworzy wykresy pokazujące prognozowane wartości importu dla top 5

towarów.

fav_items_all_year.py

Podobny do fav_item_data.py, ale analizuje dane dla wszystkich lat.

- **Analiza danych:** Grupuje dane według roku i sumuje wartości.
- **Wizualizacja:** Tworzy wykresy kolumnowe dla top 5 towarów z uwzględnieniem wszystkich lat.

sum_prediction.py

Ten plik zajmuje się prognozowaniem sumarycznej wartości importu dla różnych okresów.

- **Prognozowanie:** Używa modelu SARIMA do prognozowania sumarycznej wartości importu na przyszłe miesiące.
- **Wizualizacja:** Tworzy wykresy liniowe z rzeczywistymi i prognozowanymi wartościami importu.

sum_data.py

Zajmuje się wyświetlaniem rzeczywistych danych sumarycznych importu dla danych lat.

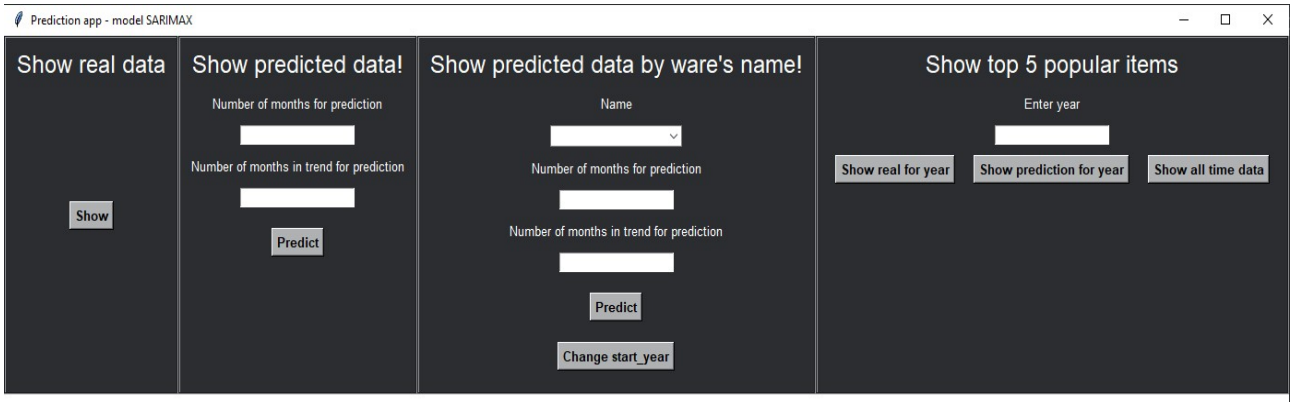
- **Analiza danych:** Grupuje dane według roku i miesiąca, a następnie sumuje wartości.
- **Wizualizacja:** Tworzy wykresy liniowe pokazujące rzeczywiste dane sumaryczne importu.

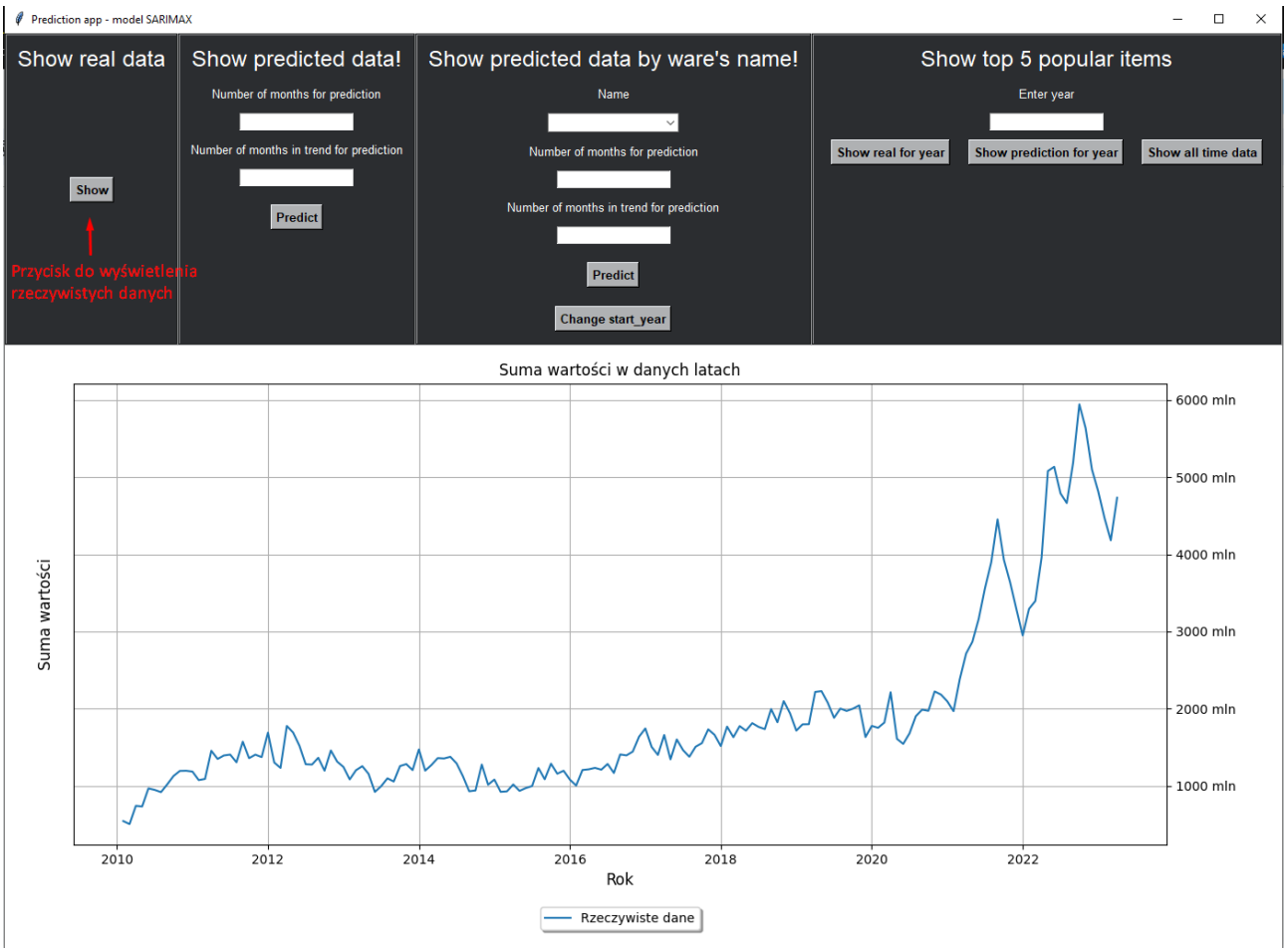
ware_sum_prediction.py

Podobny do sum_prediction.py, ale koncentruje się na prognozowaniu wartości importu dla wybranych towarów.

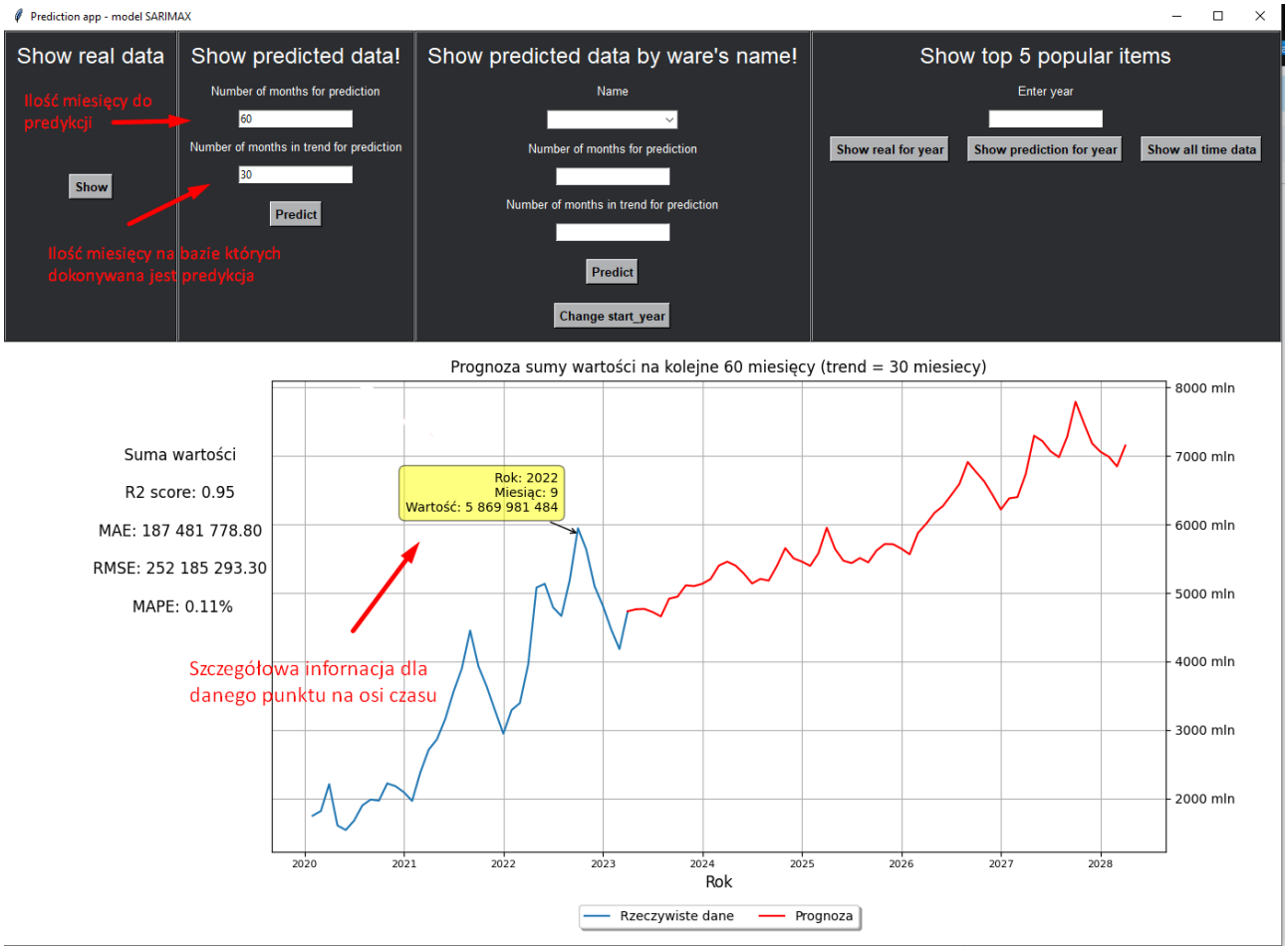
- **Prognozowanie:** Używa modelu SARIMA do prognozowania wartości importu dla konkretnych towarów.
- **Wizualizacja:** Tworzy wykresy liniowe z rzeczywistymi i prognozowanymi wartościami importu dla wybranych towarów.

5. Wizualizacja



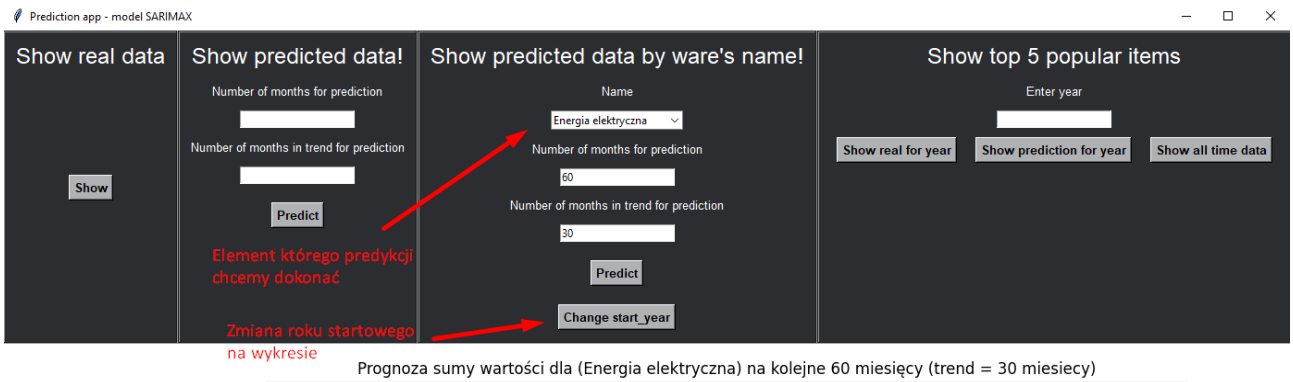


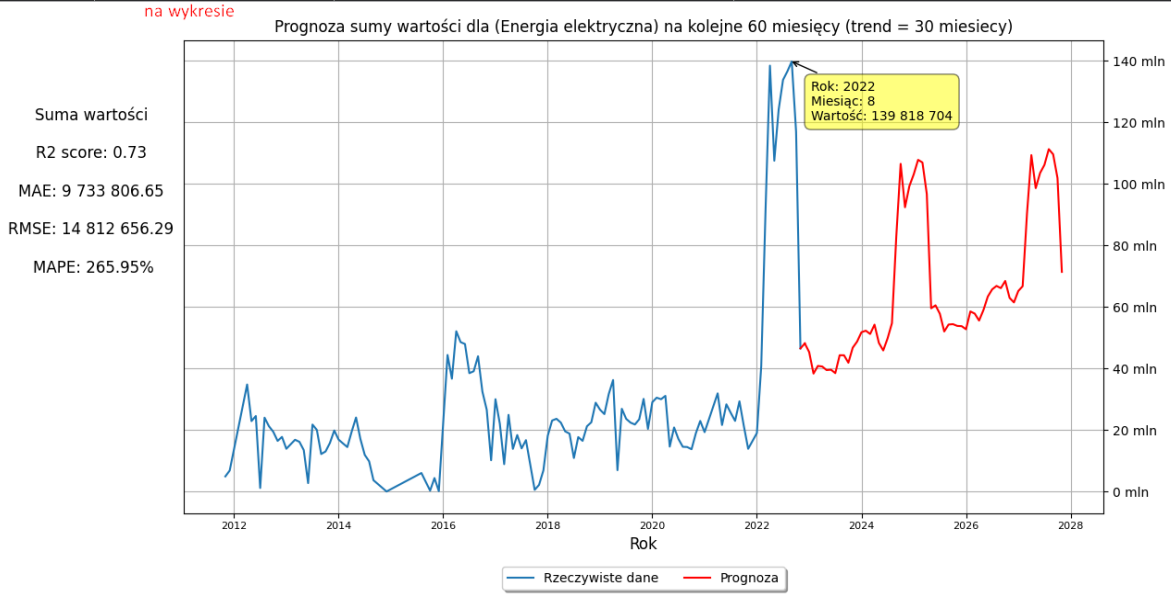

```python
    'The two args are the value and tick position'
    return '%1.0f mln' % (x * 1e-6)  # Change the scale factor to 1e-9

formatter = FuncFormatter(millions)

1 usage  Gosqu248
def plot_year_values():
    # Wykonaj zapytanie SQL, aby pobrać dane z tabeli importUkraine.data
    query = "SELECT * FROM import_ukraine.data"

    # Wczytaj dane do DataFrame
    df = pd.read_sql_query(query, engine)

    # Convert 'Wartosc' and 'miesiac' columns to numeric
    df['Wartosc'] = pd.to_numeric(df['Wartosc'], errors='coerce')
    month_dict = {'Styczeń': 1, 'Luty': 2, 'Marzec': 3, 'Kwiecień': 4, 'Maj': 5, 'Czerwiec': 6,
                  'Lipiec': 7, 'Sierpień': 8, 'Wrzesień': 9, 'Październik': 10, 'Listopad': 11, 'Grudzień': 12}
    df['miesiac'] = df['miesiac'].map(month_dict)

    # Group by year and month and calculate sum
    grouped = df.groupby(['rok', 'miesiac']).sum().reset_index()

    # Tworzenie wykresu
    fig = plt.figure(figsize=(10, 6))
    x = grouped['rok'] + grouped['miesiac'] / 12
    line, = plt.plot( *args: x, grouped['Wartosc'], label='Rzeczywiste dane')  # Removed marker='o'

    # Dostosowanie wykresu
    plt.xlabel( xlabel: 'Rok', fontsize=12)
    plt.ylabel( ylabel: 'Suma wartości', rotation=90, labelpad=15, fontsize=12)
    plt.title('Suma wartości w danych latach')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.12), fancybox=True, shadow=True, ncol=5)
    plt.gca().yaxis.tick_right()
    plt.gca().yaxis.tick_right()
    plt.gca().yaxis.set_label_position("left")

    # Increase the size of the y-axis values
    plt.tick_params(axis='y', labelsize=10)
    plt.tick_params(axis='x', labelsize=10)

    plt.grid(True)

    plt.gca().yaxis.set_major_formatter(formatter)

    cursor = mplcursors.cursor(line, hover=True)
    cursor.connect("add", lambda sel: sel.annotation.set_text(
        f'Rok: {int(sel.target[0])}\nMiesiąc: {int((sel.target[0] % 1) * 12) + 1}\nWartość: {int(sel.target[1]):,}'.replace(
            __old: ',', __new: ' ')))

    # plt.show()  # Commented out to prevent displaying the plot

    return fig
```



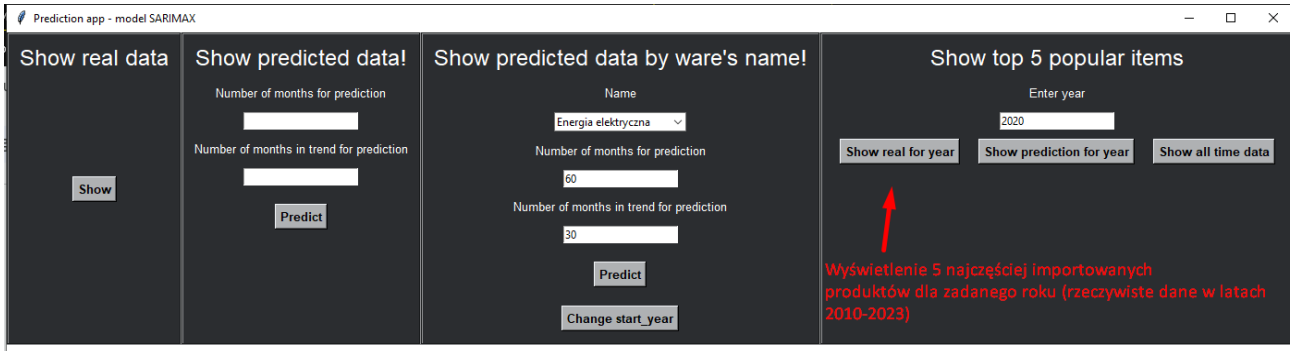

```python
1 usage   Gosqu248
def plot_year_values_with_forecast(trend=12, months_prediction=80):
    # Wykonaj zapytanie SQL, aby pobrać dane z tabeli importUkraine.data
    query = "SELECT * FROM import_ukraine.data"

    # Wczytaj dane do DataFrame
    df = pd.read_sql_query(query, engine)

    # Convert 'Wartosc' and 'miesiac' columns to numeric
    df['Wartosc'] = pd.to_numeric(df['Wartosc'], errors='coerce')
    month_dict = {'Styczeń': 1, 'Luty': 2, 'Marzec': 3, 'Kwiecień': 4, 'Maj': 5, 'Czerwiec': 6,
                  'Lipiec': 7, 'Sierpień': 8, 'Wrzesień': 9, 'Październik': 10, 'Listopad': 11, 'Grudzień': 12}
    df['miesiac'] = df['miesiac'].map(month_dict)

    # Group by year and month and calculate sum
    grouped = df.groupby(['rok', 'miesiac']).sum().reset_index()

    # Define x here before using it in np.arange
    x = grouped['rok'] + grouped['miesiac'] / 12

    # SARIMAX model
    model = SARIMAX(grouped['Wartosc'], order=(1, 1, 1), seasonal_order=(1, 1, 1, trend))
    results = model.fit()

    # Forecast next months_prediction months
    forecast = results.get_forecast(steps=months_prediction)
    forecast_index = np.arange(x.iloc[-1] + 1 / 12, x.iloc[-1] + months_prediction / 12 + 1 / 12,
                               1 / 12)  # Forecast up to the end of months_prediction months

    # Filter the data to include only years 2020 and later for plotting
    grouped_plot = grouped[grouped['rok'] >= 2020]
    x_plot = grouped_plot['rok'] + grouped_plot['miesiac'] / 12

    # Tworzenie wykresu
    fig = plt.figure(figsize=(10, 6))
    line, = plt.plot( *args: x_plot, grouped_plot['Wartosc'], label='Rzeczywiste dane')  # Removed marker='o'
    forecast_plot, = plt.plot( *args: forecast_index, forecast.predicted_mean, label='Prognoza', linestyle='-', color='red')


    # Connect the real data plot with the forecast using a red line
    last_real_data_point = x_plot.iloc[-1], grouped_plot['Wartosc'].iloc[-1]
    first_forecast_point = forecast_index[0], forecast.predicted_mean.iloc[0]
    plt.plot( *args: [last_real_data_point[0], first_forecast_point[0]], [last_real_data_point[1], first_forecast_point[1]],
             color='red')

    # Obliczanie R^2
    predicted = results.fittedvalues
    mae = mean_absolute_error(grouped['Wartosc'], predicted)
    rmse = np.sqrt(mean_squared_error(grouped['Wartosc'], predicted))
    mape = mean_absolute_percentage_error(grouped['Wartosc'], predicted)
    r2 = r2_score(grouped['Wartosc'], predicted)

    # Format the MAE, RMSE, MAPE and R^2 values
    formatted_mae = f'{mae:,.2f}'.replace( __old: ',', __new: ' ')
    formatted_rmse = f'{rmse:,.2f}'.replace( __old: ',', __new: ' ')
```

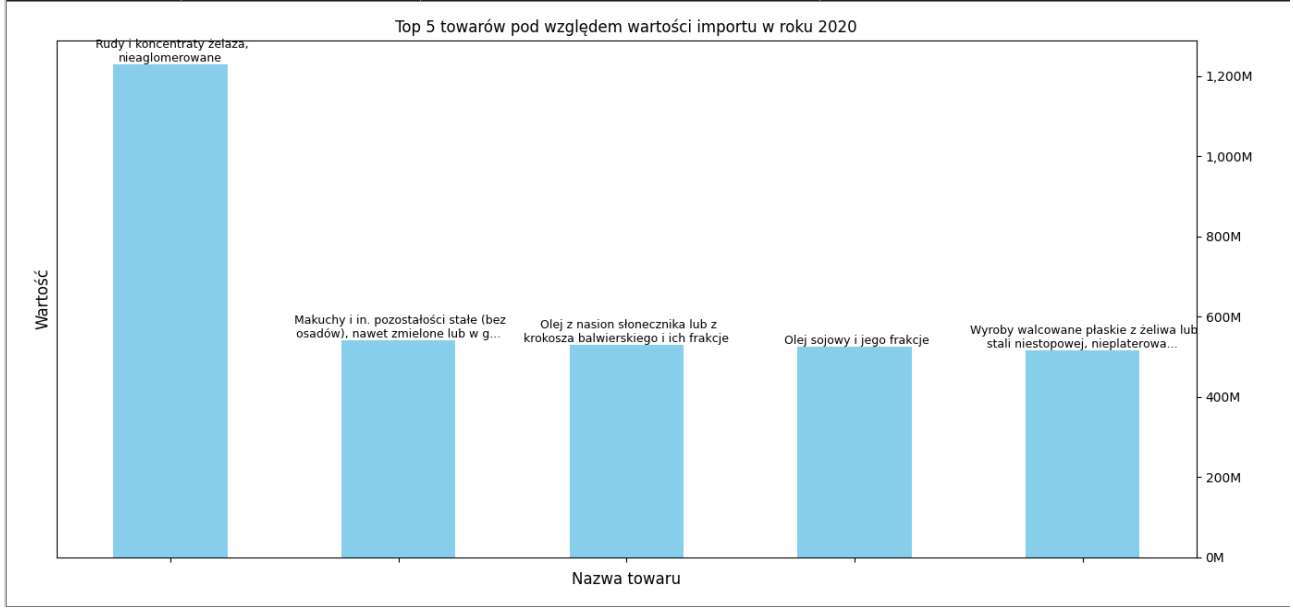
```python
# Format the MAE, RMSE, MAPE and R^2 values
formatted_mae = f'{mae:,.2f}'.replace( __old: ',', __new: ' ')
formatted_rmse = f'{rmse:,.2f}'.replace( __old: ',', __new: ' ')
formatted_mape = f'{mape:,.2f}'
formatted_r2 = f'{r2:,.2f}'

# Create a string for the x-label
xlabel_string = f'Suma wartości \n\nR2 score: {formatted_r2} \n\nMAE: {formatted_mae} \n\nRMSE: {formatted_rmse} \n\nMAPE: {formatted_mape}%'

# Dostosowanie wykresu
plt.xlabel(xlabel_string, fontsize=12, ha='center')

# Dostosowanie wykresu
plt.xlabel( xlabel: 'Rok', fontsize=12)
plt.ylabel(xlabel_string, rotation=0, labelpad=70, fontsize=12)
plt.title(f'Prognoza sumy wartości na kolejne {months_prediction} miesięcy (trend = {trend} miesiecy)')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.10), fancybox=True, shadow=True, ncol=5)
plt.gca().yaxis.tick_right()
plt.gca().yaxis.set_label_position("left")

# Increase the size of the y-axis values
plt.tick_params(axis='y', labelsize=10)
plt.tick_params(axis='x', labelsize=8)  # Adjust label size for better spacing

plt.grid(True)

plt.gca().yaxis.set_major_formatter(formatter)

# Add interactive cursor
cursor = mplcursors.cursor( pickables: [line, forecast_plot], hover=True)
cursor.connect("add", lambda sel: sel.annotation.set_text(
    f'Rok: {int(sel.target[0])}\nMiesiąc: {int((sel.target[0] % 1) * 12) + 1}\nWartość: {int(sel.target[1]):,}'.replace(
        __old: ',', __new: ' ')))

#plt.show()  # Commented out to prevent displaying the plot

return fig
```





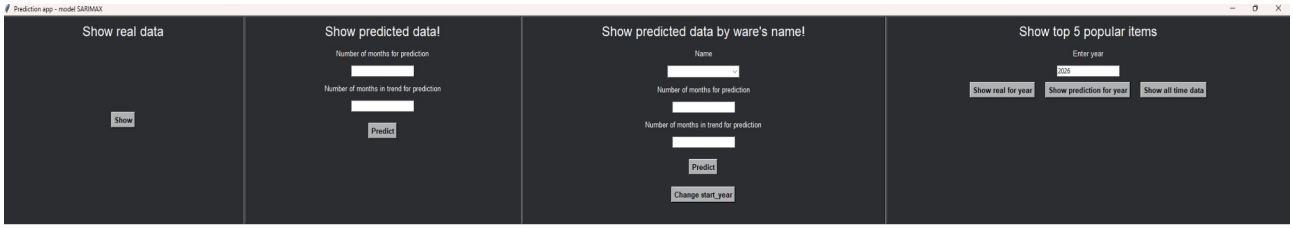
```python
def wrap_labels(labels, width, max_length=50):
    wrapped_labels = []
    for label in labels:
        if len(label) > max_length:
            label = label[:max_length] + '...'  # Przytnij i dodaj '...'
        wrapped_labels.append('\n'.join(textwrap.wrap(label, width)))
    return wrapped_labels



# Funkcja do generowania wykresu kolumnowego dla 5 najczęściej występujących przedmiotów w danym roku
1 usage  Gosqu248
def plot_top_items(year=2020):
    # Wybieramy dane dla danego roku
    df_year = df[df["rok"] == year]

    # Grupujemy dane po nazwie przedmiotu i sumujemy wartości
    grouped = df_year.groupby("SITC-R4.nazwa")["Wartosc"].sum()

    # Sortujemy dane malejąco i wybieramy 5 najczęściej występujących przedmiotów
    top_items = grouped.sort_values(ascending=False).head(6)
    top_items = top_items.iloc[1:]  # Usuń pierwszy element (największy)

    # Tworzymy wykres
    fig, ax = plt.subplots(figsize=(10, 6))  # Utwórz obiekt figury i osi
    bars = top_items.plot(kind="bar", color="skyblue", ax=ax)
    ax.set_title(f"Top 5 towarów pod względem wartości importu w roku {year}")
    ax.set_ylabel("Wartość", fontsize=12)
    ax.set_xlabel("Nazwa towaru", fontsize=12)  # Ustawienie etykiety osi X

    plt.gca().yaxis.tick_right()



    # Dodajemy niestandardowy format dla etykiet osi Y
    formatter = FuncFormatter(lambda x, pos: '{:,.0f}'.format(x / 1e6) + 'M')
    ax.yaxis.set_major_formatter(formatter)

    # Usunięcie etykiet osi X
    ax.set_xticklabels([])

    # Dodaj nazwy nad kolumnami, z maksymalną długością 50 znaków
    for i, (item, value) in enumerate(top_items.items()):
        wrapped_label = wrap_labels( labels: [item],  width: 40, max_length=70)[0]
        ax.text(i, value, wrapped_label, ha='center', va='bottom', fontsize=9, rotation=0)

        # Dodanie interaktywności
    cursor = mplcursors.cursor(bars, hover=True)
    cursor.connect("add", lambda sel: sel.annotation.set_text(
        f'{top_items.index[sel.target.index]}: {sel.target[1]:,.0f}'))



    return fig  # Zwróć obiekt figury
```

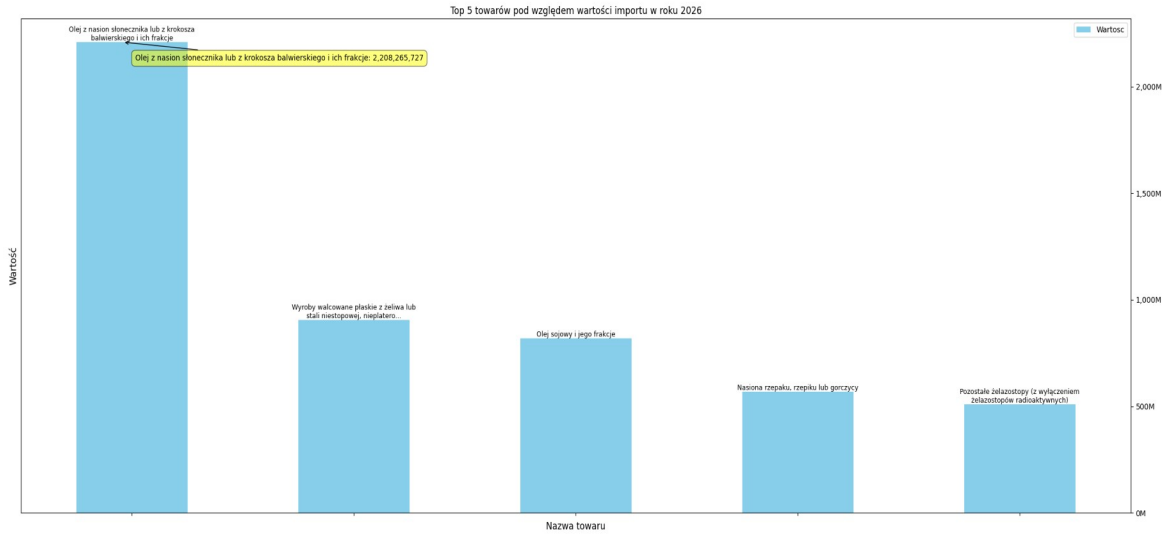





```
    formatted_mape = f'{mape:,.2f}'
    formatted_r2 = f'{r2:,.2f}'

    # Print the MAE, RMSE, MAPE and R^2 values
    print(f"R2 score for {name}: {formatted_r2}")
    print(f"MAE for {name}: {formatted_mae}")
    print(f"RMSE for {name}: {formatted_rmse}")
    print(f"MAPE for {name}: {formatted_mape}%")

    # Forecast next years_prediction years
    forecast = results.get_forecast(steps=years_prediction)
    forecast_index = np.arange(x.iloc[-1] + 1, x.iloc[-1] + years_prediction + 1)

    # Append the forecast to the forecast_df DataFrame
    forecast_df = pd.concat([forecast_df, pd.DataFrame({'Nazwa': name, 'rok': forecast_index, 'Wartosc': forecast.predicted_mean})])

# Reset the index of forecast_df
forecast_df.reset_index(drop=True, inplace=True)

data_for_year = forecast_df[forecast_df['rok'] == year]

top_items = data_for_year.sort_values(by='Wartosc', ascending=False).head(5)

# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Create a bar plot
bars = top_items.plot(kind='bar', x='Nazwa', y='Wartosc', color='skyblue', ax=ax)

# Set the title and labels
ax.set_title(f"Top 5 towarów pod względem wartości importu w roku {year}")
ax.set_xlabel("Nazwa towaru", fontsize=12)
ax.set_ylabel("Wartość", fontsize=12)

plt.gca().yaxis.tick_right()

# Add custom format for y-axis labels
formatter = FuncFormatter(lambda x, pos: '{:,.0f}'.format(x / 1e6) + 'M')
ax.yaxis.set_major_formatter(formatter)

# Remove x-axis labels
ax.set_xticklabels([])

# Add names above the bars
for i, (item, value) in enumerate(zip(top_items['Nazwa'], top_items['Wartosc'])):
    wrapped_label = wrap_labels( labels: [item], width: 40, max_length=70)[0]
    ax.text(i, value, wrapped_label, ha='center', va='bottom', fontsize=9, rotation=0)

# Add interactivity
cursor = mplcursors.cursor(bars, hover=True)
cursor.connect("add", lambda sel: sel.annotation.set_text(
    f'{top_items.iloc[sel.target.index]["Nazwa"]}: {sel.target[1]:,.0f}'))

plt.show()
```


```
1 usage   Gosqu248
def plot_top_items_with_year():
    engine = get_db_engine()
    query = "SELECT * FROM import_ukraine.data"
    df = pd.read_sql_query(query, engine)
    df = df.dropna(axis=1, how='all')

    df_filtered = df[df['rok'] < 2024]
    df_filtered['Wartosc'] = pd.to_numeric(df_filtered['Wartosc'], errors='coerce')
    grouped = df_filtered.groupby(['rok'])['Wartosc'].sum().reset_index()  # Group by 'rok' only
    grouped['date'] = pd.to_datetime(grouped['rok'], format='%Y')  # Convert 'rok' to datetime
    grouped.set_index('date', inplace=True)
    grouped = grouped.asfreq('YS')  # Use 'YS' frequency for yearly start
    grouped.sort_index(inplace=True)

    top_items = df_filtered.groupby("SITC-R4.nazwa")["Wartosc"].sum().nlargest(6)  # Top 5 przedmiotów
    top_items = top_items.iloc[1:]  # Usuń pierwszy element (największy)

    fig, ax = plt.subplots(figsize=(10, 6))
    bars = top_items.plot(kind="bar", color="skyblue", ax=ax)
    ax.set_title(f"Top 5 towarów pod względem wartości ze wszystkich lat")
    ax.set_ylabel("Wartość", fontsize=12)
    ax.set_xlabel("Nazwa towaru", fontsize=12)
    formatter = FuncFormatter(lambda x, pos: '{:,.0f}'.format(x / 1e6) + 'M')
    ax.yaxis.set_major_formatter(formatter)
    ax.yaxis.tick_right()
    ax.set_xticklabels([])

    for i, v in enumerate(top_items):
        wrapped_text = textwrap.fill(top_items.index[i], width=30)  # zawijanie do maksymalnie 20 znaków
        ax.text(i, v, wrapped_text, color='black', ha='center', va='bottom',
                bbox=dict(facecolor='none', edgecolor='none', boxstyle='round,pad=0.5'))

    cursor = mplcursors.cursor(bars, hover=True)
    cursor.connect("add", lambda sel: sel.annotation.set_text(
        f'{top_items.index[sel.target.index]}: {sel.target[1]:,.0f}'))
    return fig
```






```python
import pandas as pd
from sqlalchemy import create_engine


# Interface implementation for datasource connection and data import
data = pd.read_csv( filepath_or_buffer: 'data/Import.csv', header=0, encoding='latin1')

data = data.apply(lambda x: x.str.encode('latin1').str.decode('utf-8') if x.dtype == "object" else x)


# Utwórz połączenie do bazy danych PostgreSQL
engine = create_engine('postgresql://2023_urban_grzegorz:35240@195.150.230.208:5432/2023_urban_grzegorz')

#data.to_sql('data', engine, schema='import_ukraine', if_exists='replace', index=False)
```

6. Link do projektu (github)

https://github.com/Gosqu248/ImportFromUkraineProjekt