Analiza zmian klimatycznych metodami uczenia maszynowego

Niniejszy projekt ma na celu przedstawienie technik uczenia maszynowego w analizie zestawów danych klimatycznych, a także analizę i wizualizację tych danych.

Import bibliotek:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import r2_score
from sklearn.linear_model import Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from pmdarima import model_selection
import pmdarima as pm
from statsmodels.tsa.ar_model import AutoReg
from tabulate import tabulate
from xgboost import XGBRegressor
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
import xgboost as xgb
from matplotlib.dates import DateFormatter
```

Ustawienie opcji, które wyświetlają wszystkie rekordy danych **(opcjonalnie)**.

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Wyłączenie powyższych opcji umożliwia poniższy kod.

```
pd.reset_option('display.max_columns')
pd.reset_option('display.max_rows')
```

1. Emisja CO2

Pierwszym etapem niniejszej pracy jest przedstawienie danych w formie czytelnych wykresów odnośnie emisji CO2 w latach 1960 - 2019 dla całego świata i pojedyńczych państw, a także próba predykcji tych wartości dla lat późniejszych (2020 - 2100). Zostaną użyte i porównane różne algorytmy uczenia maszynowego takie jak:

- regresja liniowa
- regresja wielomianowa

a także metody statystyczne takie jak:

- autoregresja
- model ARIMA

Wyjaśnienie danych znajdujących się w pliku co2_emissions_kt_by_country.csv:

- country_code: To kod kraju, który identyfikuje konkretny kraj.
- country_name: To pełna nazwa kraju odpowiadająca kodowi kraju.
- year: To rok, dla którego zgromadzono dane dotyczące emisji CO2.
- value: To wartość emisji CO2 zgromadzona dla danego kraju i roku. Ta wartość wskazuje ilość CO2 emitowanego przez dany kraj w danym roku.

Wniosek z tych danych mógłby obejmować analizę trendów emisji CO2 w danym kraju lub regionie, porównanie poziomów emisji między różnymi krajami, zrozumienie wpływu polityk ekologicznych na poziomy emisji w poszczególnych krajach oraz prognozowanie przyszłych emisji CO2 na podstawie danych historycznych.

Wczytanie danych:

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
print(data_CO2)
```

```
      country_code country_name  year         value
0              ABW        Aruba  1960  11092.675000
1              ABW        Aruba  1961  11576.719000
2              ABW        Aruba  1962  12713.489000
3              ABW        Aruba  1963  12178.107000
4              ABW        Aruba  1964  11840.743000
...            ...          ...   ...           ...
13948          ZWE     Zimbabwe  2015  12430.000305
13949          ZWE     Zimbabwe  2016  11020.000458
13950          ZWE     Zimbabwe  2017  10340.000153
13951          ZWE     Zimbabwe  2018  12380.000114
13952          ZWE     Zimbabwe  2019  11760.000229

[13953 rows x 4 columns]
```

Dane w tabeli reprezentują ilość CO2 wyemitowaną w danym roku dla różnych państw. Kolumna "value" zawiera ilość CO2 wyrażoną w kilotonach.

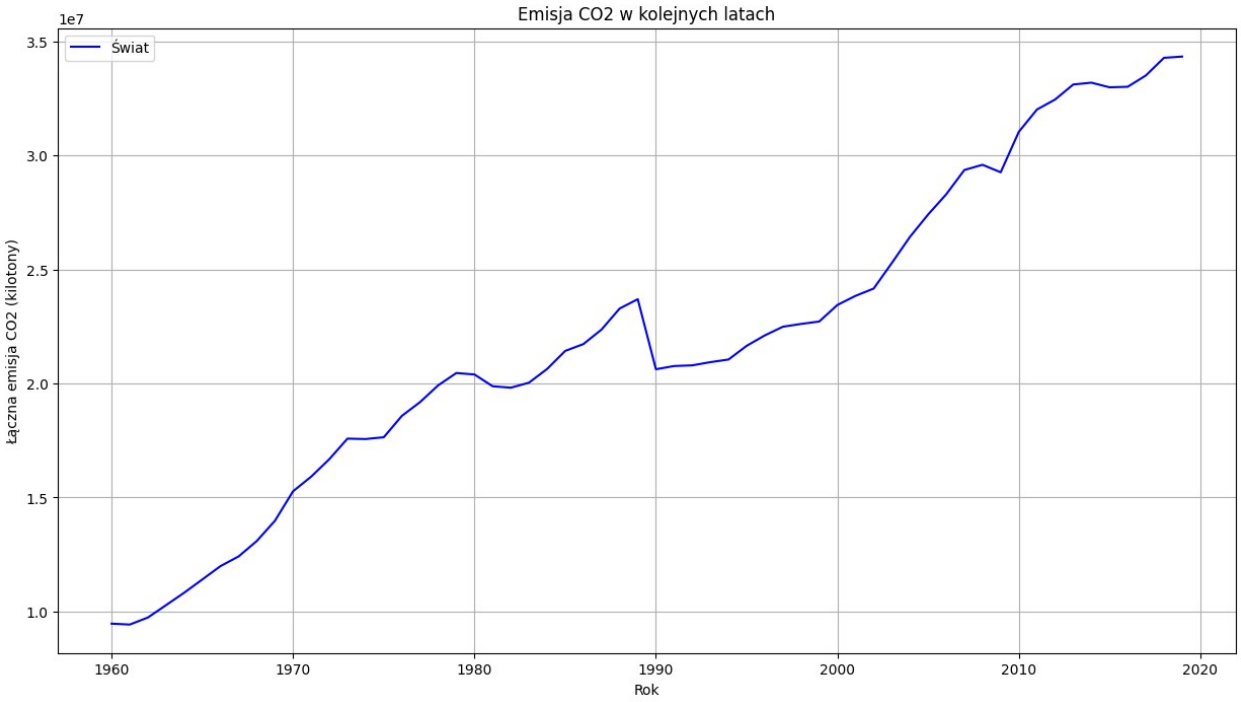
Łączna emisja CO2 wg. roku i kraju:

```
world_data = data_CO2[data_CO2['country_name'] == 'World']
usa_data = data_CO2[data_CO2['country_name'] == 'United States']
european_data = data_CO2[data_CO2['country_name'] == 'European Union']
china_data = data_CO2[data_CO2['country_name'] == 'China']
india_data = data_CO2[data_CO2['country_name'] == 'India']
russia_data = data_CO2[data_CO2['country_name'] == 'Russian
Federation']
poland_data = data_CO2[data_CO2['country_name'] == 'Poland']

world_co2_by_year = world_data.groupby('year')['value'].sum()
usa_co2_by_year = usa_data.groupby('year')['value'].sum()
european_co2_by_year = european_data.groupby('year')['value'].sum()
china_co2_by_year = china_data.groupby('year')['value'].sum()
india_co2_by_year = india_data.groupby('year')['value'].sum()
russia_co2_by_year = russia_data.groupby('year')['value'].sum()
poland_co2_by_year = poland_data.groupby('year')['value'].sum()

pd.set_option('display.float_format', lambda x: '%.2f' % x)

total_co2_by_year = world_co2_by_year

total_co2_by_year.tail(10)
```

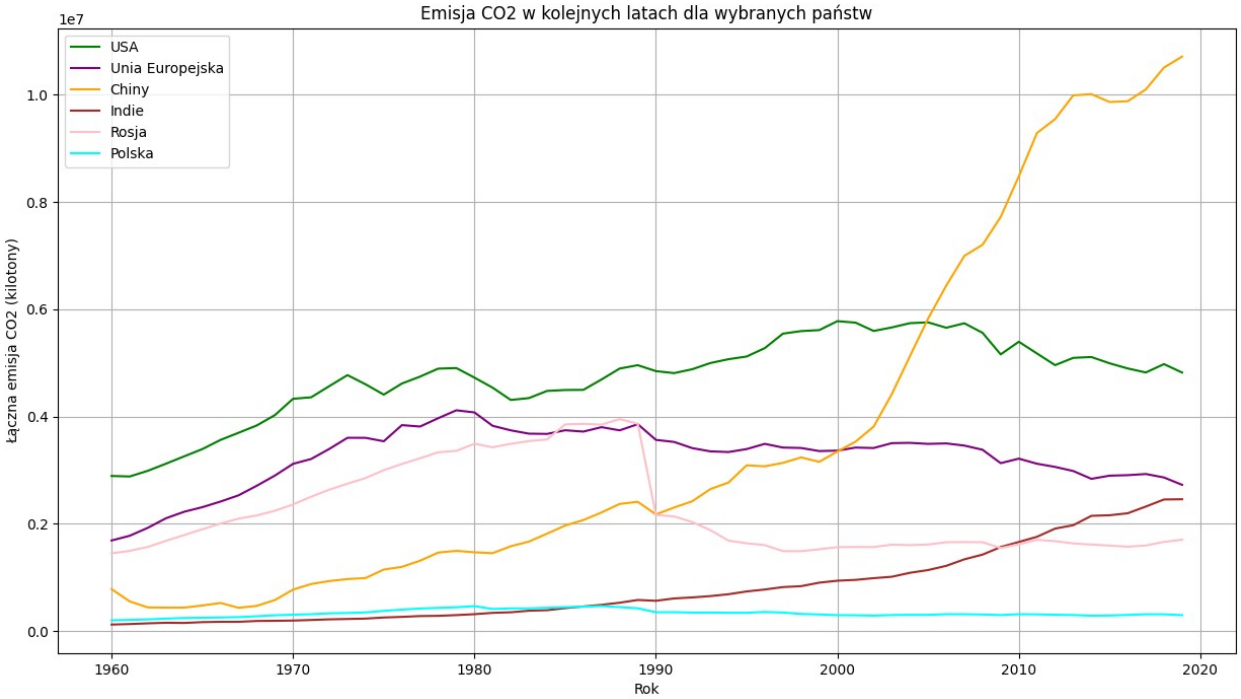
```
year
2010   31043476.98
2011   32021108.26
2012   32460316.86
2013   33119382.99
2014   33198729.82
2015   32995536.02
2016   33018556.40
2017   33514537.91
2018   34289350.66
2019   34344006.07
Name: value, dtype: float64
```

```
plt.figure(figsize=(15, 8))
plt.plot(total_co2_by_year.index, total_co2_by_year.values,
label='Świat', color='blue')


plt.title('Emisja CO2 w kolejnych latach')
plt.xlabel('Rok')
plt.ylabel('Łączna emisja CO2 (kilotony)')
plt.legend()
```

```
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(15, 8))

plt.plot(usa_co2_by_year.index, usa_co2_by_year.values, label='USA',
color='green')
plt.plot(european_co2_by_year.index, european_co2_by_year.values,
label='Unia Europejska', color='purple')
plt.plot(china_co2_by_year.index, china_co2_by_year.values,
label='Chiny', color='orange')
plt.plot(india_co2_by_year.index, india_co2_by_year.values,
label='Indie', color='brown')
plt.plot(russia_co2_by_year.index, russia_co2_by_year.values,
label='Rosja', color='pink')
plt.plot(poland_co2_by_year.index, poland_co2_by_year.values,
label='Polska', color='cyan')

plt.title('Emisja CO2 w kolejnych latach dla wybranych państw')
plt.xlabel('Rok')
plt.ylabel('Łączna emisja CO2 (kilotony)')
plt.legend()
plt.grid(True)
plt.show()
```

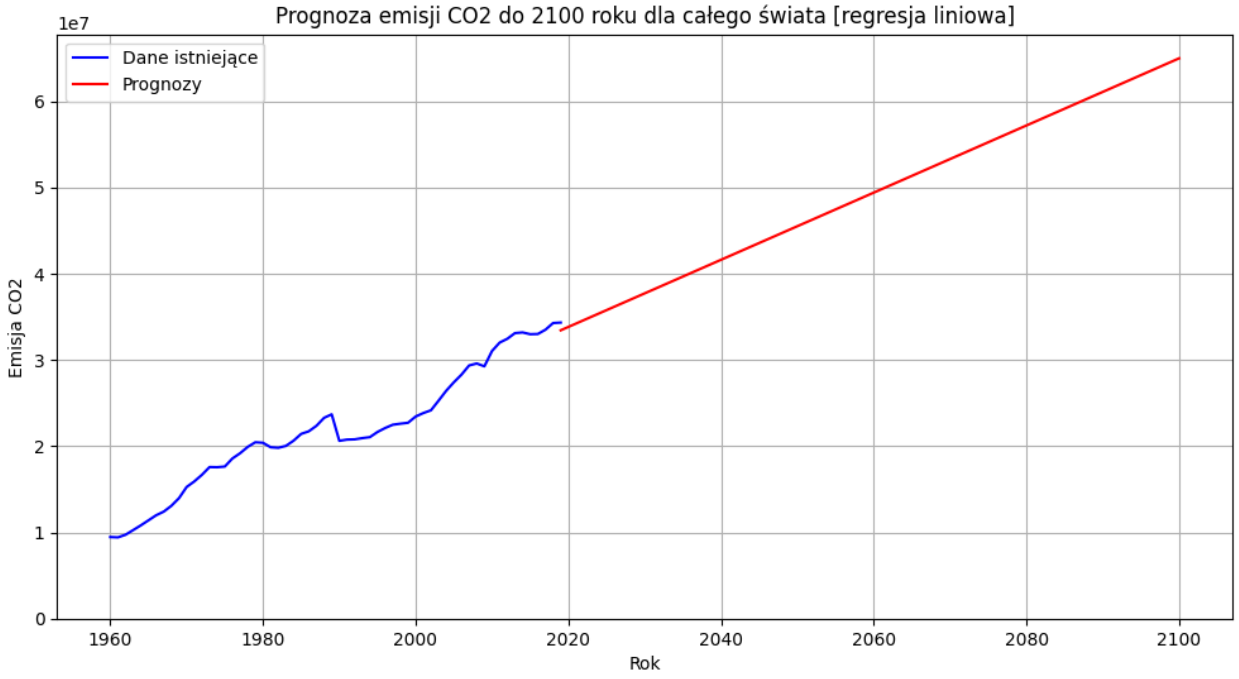


Próba predykcji emisji CO2 dla *całego świata* metodą **regresji liniowej**

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
world_data=data_CO2[data_CO2["country_name"]=="World"]
X = world_data[['year']]
y = world_data['value']


model_lr = LinearRegression()
model_lr.fit(X, y)

r2_train_lr = model_lr.score(X, y)
mse_lr = mean_squared_error(y, model_lr.predict(X))
mae_lr = mean_absolute_error(y, model_lr.predict(X))

future_years = [[year] for year in range(2019, 2101)]
future_predictions = model_lr.predict(future_years)

plt.figure(figsize=(12, 6))
plt.plot(world_data['year'], world_data['value'], color='blue',
label='Dane istniejące')
plt.plot(range(2019, 2101), future_predictions, color='red',
label='Prognozy')

plt.title('Prognoza emisji CO2 do 2100 roku dla całego świata
[regresja liniowa]')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
```

```
plt.legend()
plt.grid(True)
plt.show()

print('Regresja liniowa - Metryka R2 dla danych
trenowanych:',round(r2_train_lr,2))
print('Regresja liniowa - Metryka MSE dla danych trenowanych:',mse_lr)
print('Regresja liniowa - Metryka MAE dla danych trenowanych:',mae_lr)
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
```



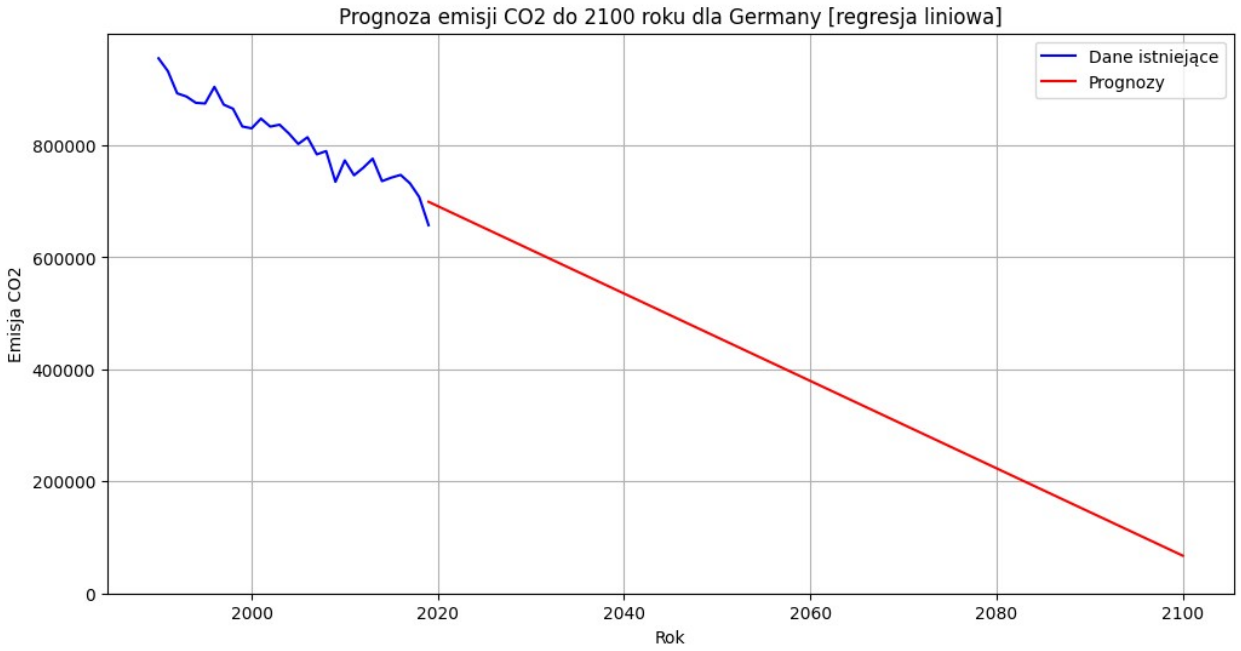

```
Regresja liniowa - Metryka R2 dla danych trenowanych: 0.94
Regresja liniowa - Metryka MSE dla danych trenowanych:
2769226987816.5386
Regresja liniowa - Metryka MAE dla danych trenowanych:
1495133.2619871527
```

Próba predykcji emisji CO2 dla *wybranego państwa* metodą **regresji liniowej**

```
try:
    data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
    panstwo = input("Podaj nazwę państwa dla którego ma być dokonana
predykcja:")
```

```
    data_country = data_CO2[data_CO2['country_name'] == panstwo]

    X = data_country[['year']]
    y = data_country['value']


    model_lr = LinearRegression()
    model_lr.fit(X, y)

    r2_train_lr2 = model_lr.score(X, y)
    mse_lr2 = mean_squared_error(y, model_lr.predict(X))
    mae_lr2 = mean_absolute_error(y, model_lr.predict(X))

    future_years = [[year] for year in range(2019, 2101)]
    future_predictions = model_lr.predict(future_years)

    plt.figure(figsize=(12, 6))
    plt.plot(data_country['year'], data_country['value'],
color='blue', label='Dane istniejące')
    plt.plot(range(2019, 2101), future_predictions, color='red',
label='Prognozy')

    plt.title('Prognoza emisji CO2 do 2100 roku dla ' + panstwo + '
[regresja liniowa]')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.legend()
    plt.grid(True)
    plt.show()

    print('Regresja liniowa - Metryka R2 dla danych
trenowanych:',round(r2_train_lr2,2))
    print('Regresja liniowa - Metryka MSE dla danych
trenowanych:',mse_lr2)
    print('Regresja liniowa - Metryka MAE dla danych
trenowanych:',mae_lr2)
except:
    print("Nie ma takiego państwa, tylko angielskie nazwy z wielkiej
litery")
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
```




```
Regresja liniowa - Metryka R2 dla danych trenowanych: 0.93
Regresja liniowa - Metryka MSE dla danych trenowanych:
330210033.3469252
Regresja liniowa - Metryka MAE dla danych trenowanych:
14427.430607141509
```

Należy oczywiście wziąść pod uwagę, że regresja liniowa jest prostym modelem, który zakłada liniową zależność między zmiennymi. Jednak emisje CO2 mogą podlegać wpływowi wielu skomplikowanych czynników, takich jak technologiczny postęp, polityka rządowa i zmieniające się wzorce konsumpcji. Rządy prawdopodobnie będą kontynuować wysiłki w kierunku ograniczenia emisji CO2, co może wpłynąć na przyszłe trendy. Dlatego aby tego typu prognozy były jak najbardziej rzetelne powinno się monitorować i brać pod uwagę inne czynniki, by prognozy te były jak najbardziej rzetelne.

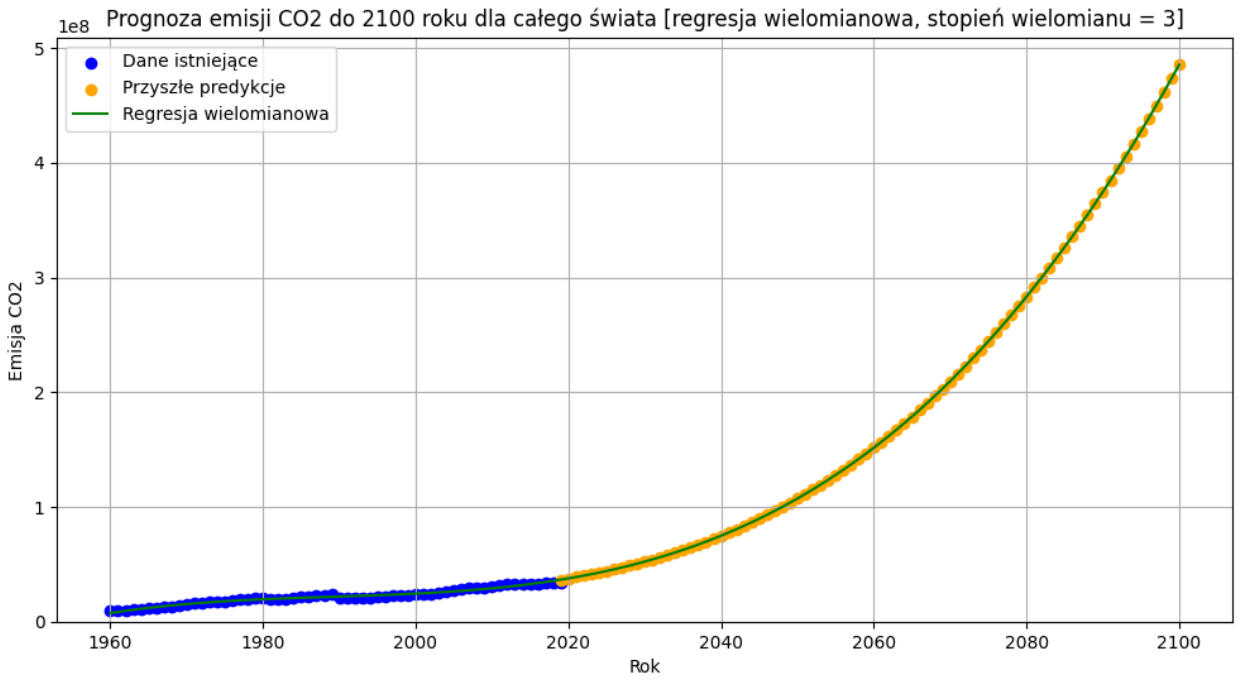
Próba predykcji emisji CO2 dla *całego świata* metodą **regresji wielomianowej**

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
world_data=world_data[world_data["country_name"]=="World"]
X = world_data[['year']]
y = world_data['value'].values


poly_features = PolynomialFeatures(degree=3)
X_poly = poly_features.fit_transform(X)


X_train, X_test, y_train, y_test = train_test_split(X_poly, y,
test_size=0.2, random_state=42)
```

```
model = LinearRegression()
model.fit(X_train, y_train)

y_pred_train = model.predict(X_train)

r2_train_pr = r2_score(y_train, y_pred_train)
mse_pr = mean_squared_error(y, model_lr.predict(X))
mae_pr = mean_absolute_error(y, model_lr.predict(X))

future_years = [[year] for year in range(2019, 2101)]
future_years_poly = poly_features.transform(future_years)
future_emission = model.predict(future_years_poly)

X_range = np.concatenate((X, future_years), axis=0)
X_range_poly = poly_features.transform(X_range)


plt.figure(figsize=(12, 6))
plt.title('Prognoza emisji CO2 do 2100 roku dla całego świata
[regresja wielomianowa, stopień wielomianu = 3]')
plt.scatter(X, y, color='blue', label='Dane istniejące')
plt.scatter(future_years, future_emission, color='orange',
label='Przyszłe predykcje')
plt.plot(X_range, model.predict(X_range_poly), color='green',
label='Regresja wielomianowa')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
plt.grid(True)
plt.legend()
plt.show()

print(f"Regresja wielomianowa - Metryka R2 dla trenowanych danych:
{round(r2_train_pr,2)}")
print('Regresja wielomianowa - Metryka MSE dla danych
trenowanych:',mse_pr)
print('Regresja wielomianowa - Metryka MAE dla danych
trenowanych:',mae_pr)
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
```

```
with feature names
  warnings.warn(
```

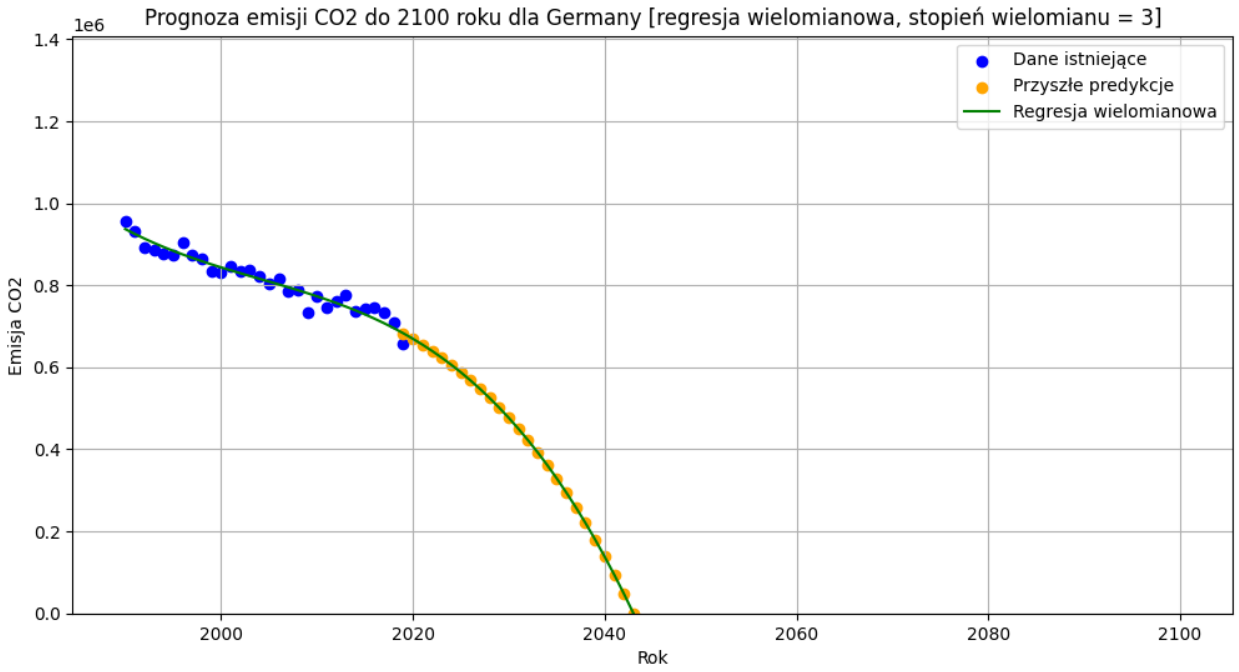



```
Regresja wielomianowa - Metryka R2 dla trenowanych danych: 0.97
Regresja wielomianowa - Metryka MSE dla danych trenowanych:
492367245795759.1
Regresja wielomianowa - Metryka MAE dla danych trenowanych:
21030052.25705492
```

```
try:
    panstwo = input("Podaj nazwę państwa dla którego ma być dokonana
predykcja:")
    data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
    degree = int(input("Podaj stopień wielomianu"))
    data_country = data_CO2[data_CO2['country_name'] == panstwo]

    X = data_country[['year']]
    y = data_country['value'].values

    poly_features = PolynomialFeatures(degree=degree)
    X_poly = poly_features.fit_transform(X)

    X_train, X_test, y_train, y_test = train_test_split(X_poly, y,
test_size=0.2, random_state=42)

    model = LinearRegression()
    model.fit(X_train, y_train)
```

```
    y_pred_train = model.predict(X_train)

    r2_train_pr2 = r2_score(y_train, y_pred_train)
    mse_pr2 = mean_squared_error(y, model_lr.predict(X))
    mae_pr2 = mean_absolute_error(y, model_lr.predict(X))

    future_years = [[year] for year in range(2019, 2101)]
    future_years_poly = poly_features.transform(future_years)
    future_emission = model.predict(future_years_poly)

    X_range = np.concatenate((X, future_years), axis=0)
    X_range_poly = poly_features.transform(X_range)

    plt.figure(figsize=(12, 6))
    plt.title('Prognoza emisji CO2 do 2100 roku dla '+ panstwo + '
[regresja wielomianowa, stopień wielomianu = '+ str(degree) + ']')
    plt.scatter(X, y, color='blue', label='Dane istniejące')
    plt.scatter(future_years, future_emission, color='orange',
label='Przyszłe predykcje')
    plt.plot(X_range, model.predict(X_range_poly), color='green',
label='Regresja wielomianowa')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.grid(True)
    plt.legend()
    plt.show()

    print(f"Regresja wielomianowa - Metryka R2 dla trenowanych danych:
{round(r2_train_pr2,2)}")
    print('Regresja wielomianowa - Metryka MSE dla danych
trenowanych:',mse_pr2)
    print('Regresja wielomianowa - Metryka MAE dla danych
trenowanych:',mae_pr2)
except:
    print("Nie ma takiego państwa, tylko angielskie nazwy z wielkiej
litery")
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
```




```
Regresja wielomianowa - Metryka R2 dla trenowanych danych: 0.95
Regresja wielomianowa - Metryka MSE dla danych trenowanych:
330210033.3469252
Regresja wielomianowa - Metryka MAE dla danych trenowanych:
14427.430607141509
```

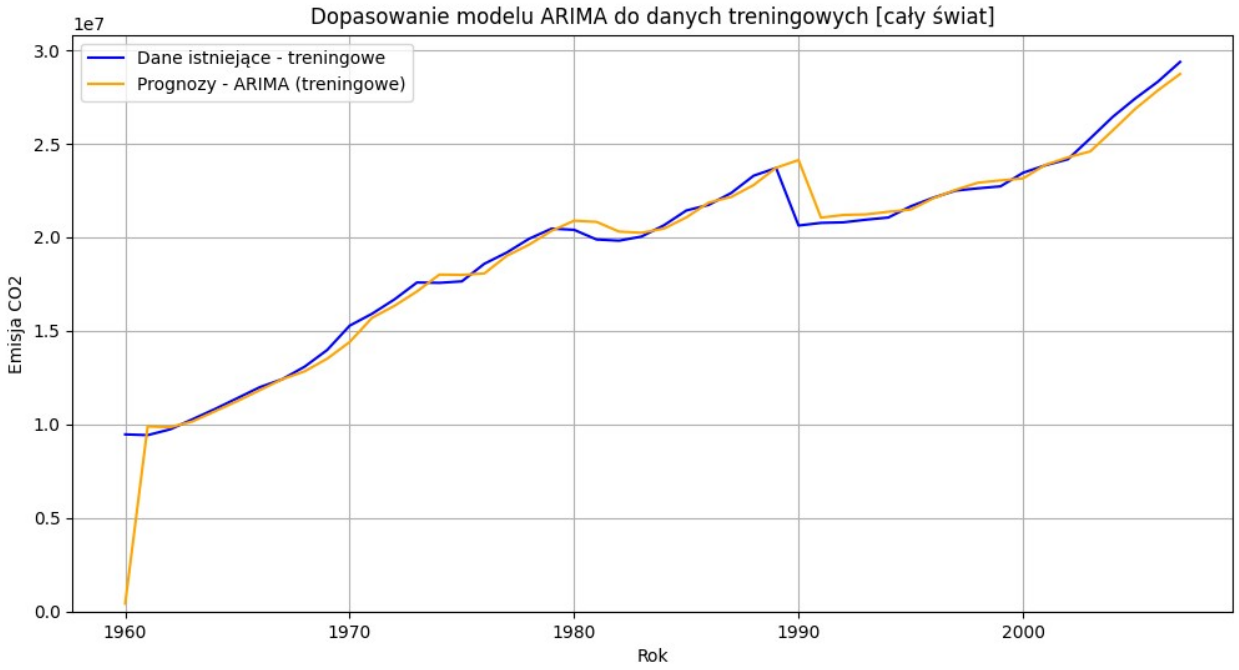
Dane z regresji liniowej przewidują bardziej optymistyczny scenariusz w którym to do roku 2100 przewidywana emisja dla całego świata będzie cały rząd wielkości mniejsza, niż za pomocą regresji wielomianowej. Różnice między przewidywaniami regresji liniowej, a wielomianowej mogą wynikać z ograniczeń liniowego modelu, który zakłada stałą, jednostajną zmianę na przestrzeni lat. W przypadku zjawisk złożonych, takich jak emisja CO2, regresja wielomianowa jest bardziej elastyczna i może lepiej odzwierciedlać nieliniowe wzorce, co czasami prowadzi do bardziej realistycznych prognoz. Jednak i tak należy wziąść pod uwagę, iż modele te operują tylko na dostarczonych danych liczbowych, nie są więc w stanie wziąść pod uwagę wysiłku rządów w celu ograniczania tejże emisji.

Model ARIMA

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
world_data = data_CO2[data_CO2["country_name"]=="World"]
X = world_data[['year']]
y = world_data['value']


train_size = int(0.8 * len(y))
train, test = y[:train_size], y[train_size:]


model_arima = pm.auto_arima(train, seasonal=False, m=1)
```

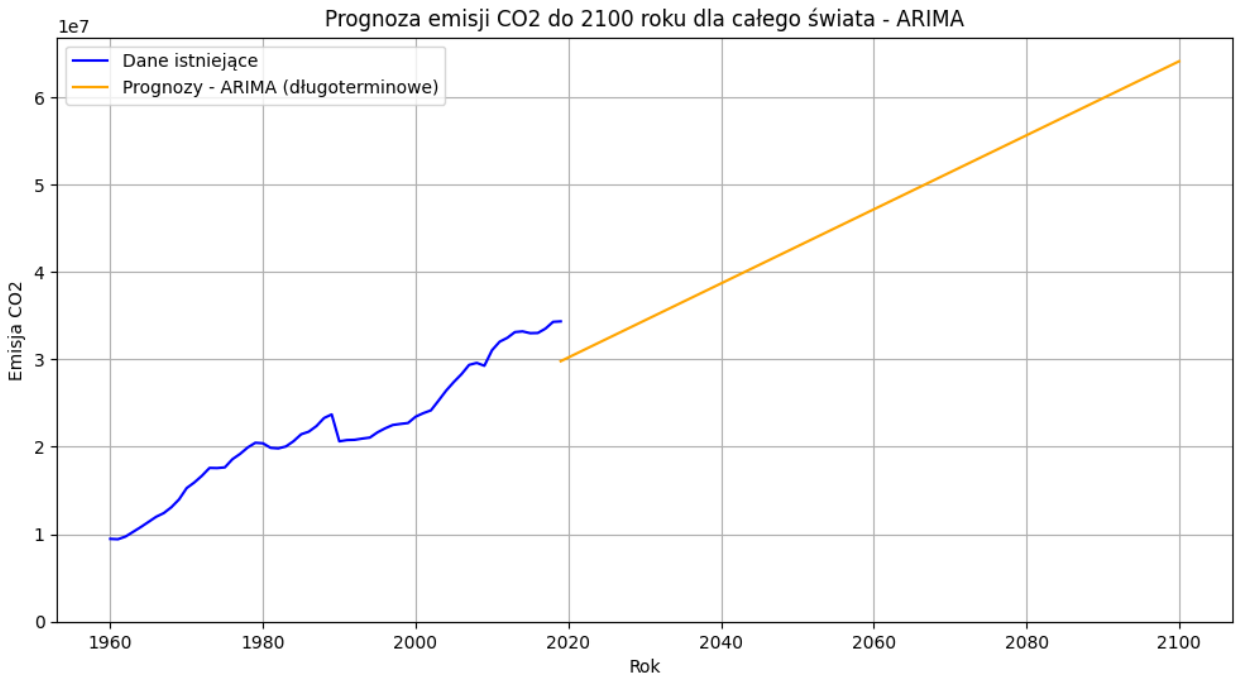
```
future_predictions_arima = model_arima.predict(n_periods=len(test))  # Predict for test period

predicted_train = model_arima.predict_in_sample()
mse_arima = mean_squared_error(train, predicted_train)
mae_arima = mean_absolute_error(train, predicted_train)


print('ARIMA - Metryka MSE dla danych trenowanych:', mse_arima)
print('ARIMA - Metryka MAE dla danych trenowanych:', mae_arima)

plt.figure(figsize=(12, 6))
plt.plot(world_data['year'][:train_size], train, color='blue',
label='Dane istniejące - treningowe')
plt.plot(world_data['year'][:train_size], predicted_train,
color='orange', label='Prognozy - ARIMA (treningowe)')

plt.title('Dopasowanie modelu ARIMA do danych treningowych [cały
świat]')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
plt.legend()
plt.grid(True)
plt.show()

future_years = np.arange(2019, 2101)
future_predictions_long_term =
model_arima.predict(n_periods=len(future_years))

plt.figure(figsize=(12, 6))
plt.plot(world_data['year'], world_data['value'], color='blue',
label='Dane istniejące')
plt.plot(future_years, future_predictions_long_term, color='orange',
label='Prognozy - ARIMA (długoterminowe)')

plt.title('Prognoza emisji CO2 do 2100 roku dla całego świata -
ARIMA')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
plt.legend()
plt.grid(True)
plt.show()


ARIMA - Metryka MSE dla danych trenowanych: 2104158295007.1677
ARIMA - Metryka MAE dla danych trenowanych: 570585.3400432427
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
```




```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
```

```
index will result in an exception.
  return get_prediction_index(
```



Model ARIMA to model statystyczny używany do analizy i prognozowania szeregów czasowych.

Pierwszy wykres pokazuje, że model ten dobrze dopasowuje się do historycznych danych co widać po bliskim dopasowaniu niebieskiej linii do pomarańczowej. Sugeruje to jego skuteczność do krótkoterminowych prognoz. Długoterminowa prognoza modelu ARIMA pokazuje znaczący wzrost emisji CO2, przewidując liniowy trend wzrostowy aż do roku 2100. Pomimo dobrego dopasowania do danych historycznych, prognozy długoterminowe mogą być mniej wiarygodne ze względu na założenia liniowego trendu (podobnie jak regresja liniowa).

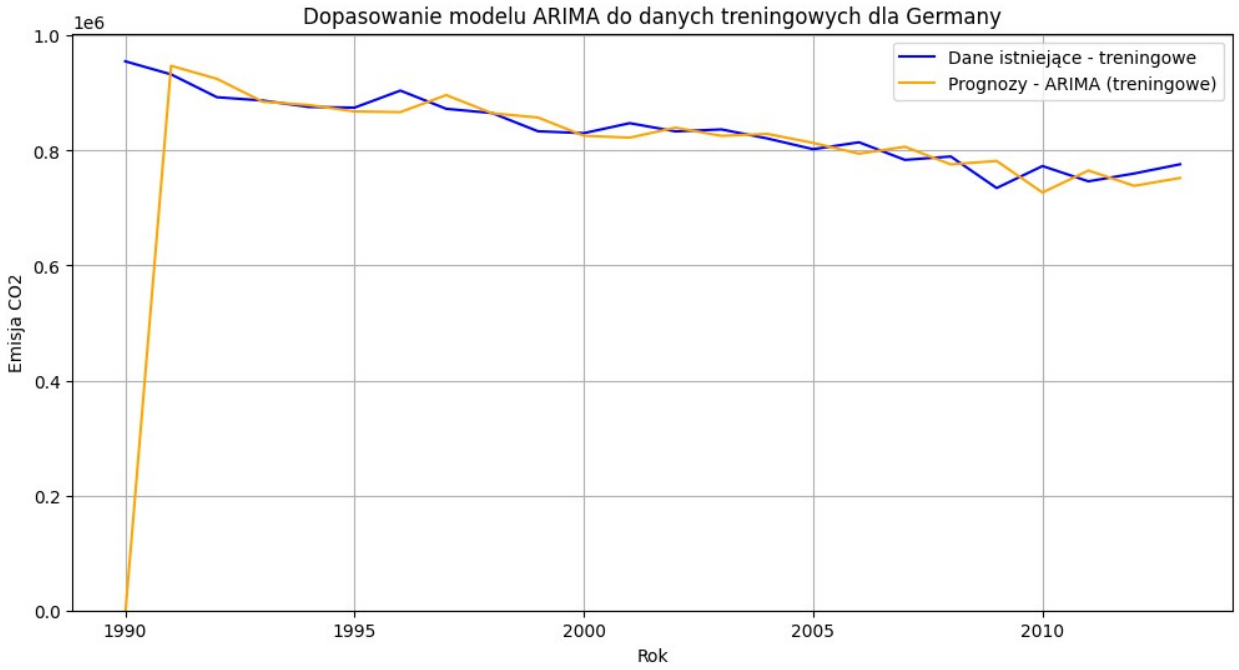
Model ARIMA dla wybranego państwa

```
try:
    data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
    panstwo = input("Podaj nazwę państwa dla którego ma być dokonana
predykcja:")
    data_country = data_CO2[data_CO2['country_name'] == panstwo]
    X = data_country[['year']]
    y = data_country['value']


    train_size = int(0.8 * len(y))
    train, test = y[:train_size], y[train_size:]


    model_arima = pm.auto_arima(train, seasonal=False, m=1)
```

```
    future_predictions_arima =
model_arima.predict(n_periods=len(test))  # Predict for test period

    predicted_train = model_arima.predict_in_sample()
    mse_arima2 = mean_squared_error(train, predicted_train)
    mae_arima2 = mean_absolute_error(train, predicted_train)

    print('ARIMA - Metryka MSE dla danych trenowanych:', mse_arima2)
    print('ARIMA - Metryka MAE dla danych trenowanych:', mae_arima2)

    plt.figure(figsize=(12, 6))
    plt.plot(data_country['year'][:train_size], train, color='blue',
label='Dane istniejące - treningowe')
    plt.plot(data_country['year'][:train_size], predicted_train,
color='orange', label='Prognozy - ARIMA (treningowe)')

    plt.title(f'Dopasowanie modelu ARIMA do danych treningowych dla
{panstwo}')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.legend()
    plt.grid(True)
    plt.show()

    future_years = np.arange(2019, 2101)
    future_predictions_long_term =
model_arima.predict(n_periods=len(future_years))

    plt.figure(figsize=(12, 6))
    plt.plot(data_country['year'], data_country['value'],
color='blue', label='Dane istniejące')
    plt.plot(future_years, future_predictions_long_term,
color='orange', label='Prognozy - ARIMA (długoterminowe)')

    plt.title(f'Arima - Prognoza emisji CO2 do 2100 roku dla
{panstwo}')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.legend()
    plt.grid(True)
    plt.show()

except:
    print("Nie ma takiego państwa, tylko angielskie nazwy z wielkiej
litery")
```

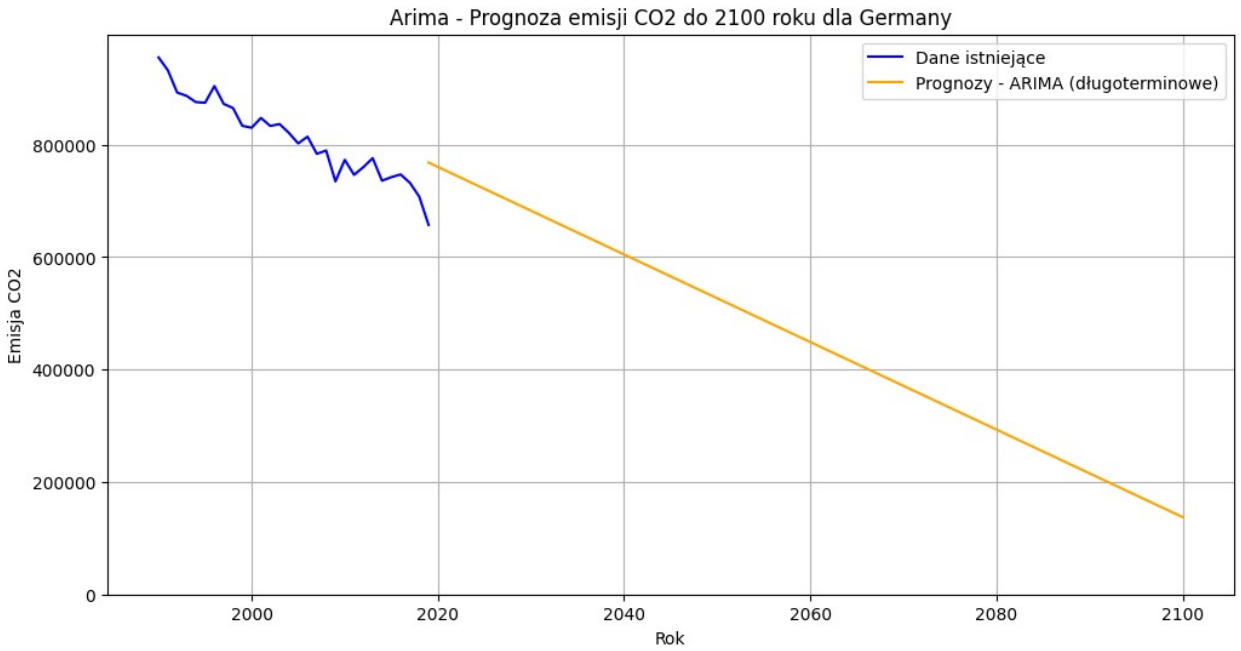
```
ARIMA - Metryka MSE dla danych trenowanych: 39134646079.85395
ARIMA - Metryka MAE dla danych trenowanych: 57804.546809541906
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
```




```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
```

```
index will result in an exception.
  return get_prediction_index(
```



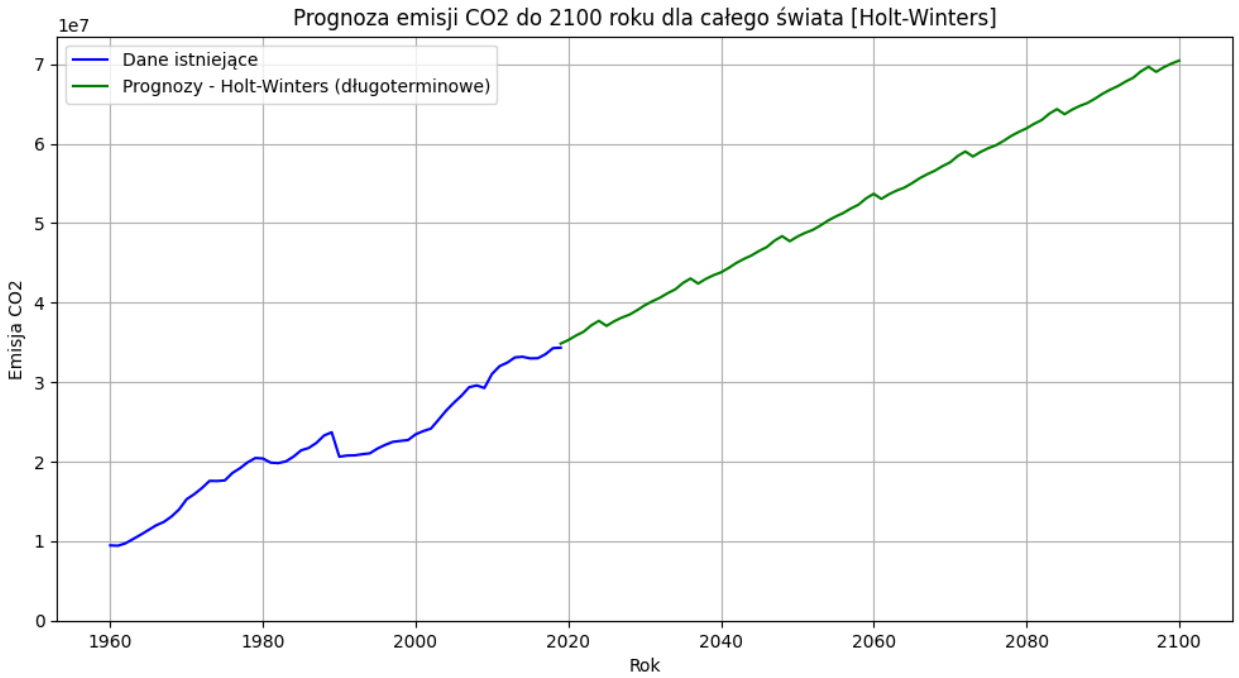
Model Holt-Winters

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
world_data = data_CO2[data_CO2['country_name'] == 'World']
X = world_data['year']
y = world_data['value']

world_data.set_index('year', inplace=True)

model_hw = ExponentialSmoothing(world_data['value'], trend='add',
seasonal='add', seasonal_periods=12)
fit_hw = model_hw.fit()


future_years = np.arange(2019, 2101)
future_predictions_hw = fit_hw.forecast(len(future_years))

y_true = world_data['value']
y_pred = fit_hw.fittedvalues

mse_hw = mean_squared_error(y_true, y_pred)
mae_hw = mean_absolute_error(y_true, y_pred)

print('Holt-Winters - Metryka MSE dla danych testowych:', mse_hw)
print('Holt-Winters - Metryka MAE dla danych testowych:', mae_hw)

plt.figure(figsize=(12, 6))
```

```
plt.plot(world_data.index, world_data['value'], color='blue',
label='Dane istniejące')
plt.plot(future_years, future_predictions_hw, color='green',
label='Prognozy - Holt-Winters (długoterminowe)')

plt.title('Prognoza emisji CO2 do 2100 roku dla całego świata [Holt-
Winters]')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\holtwinters\
model.py:917: ConvergenceWarning: Optimization failed to converge.
Check mle_retvals.
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
```

```
Holt-Winters - Metryka MSE dla danych testowych: 495119191035.0865
Holt-Winters - Metryka MAE dla danych testowych: 477964.33186931466
```



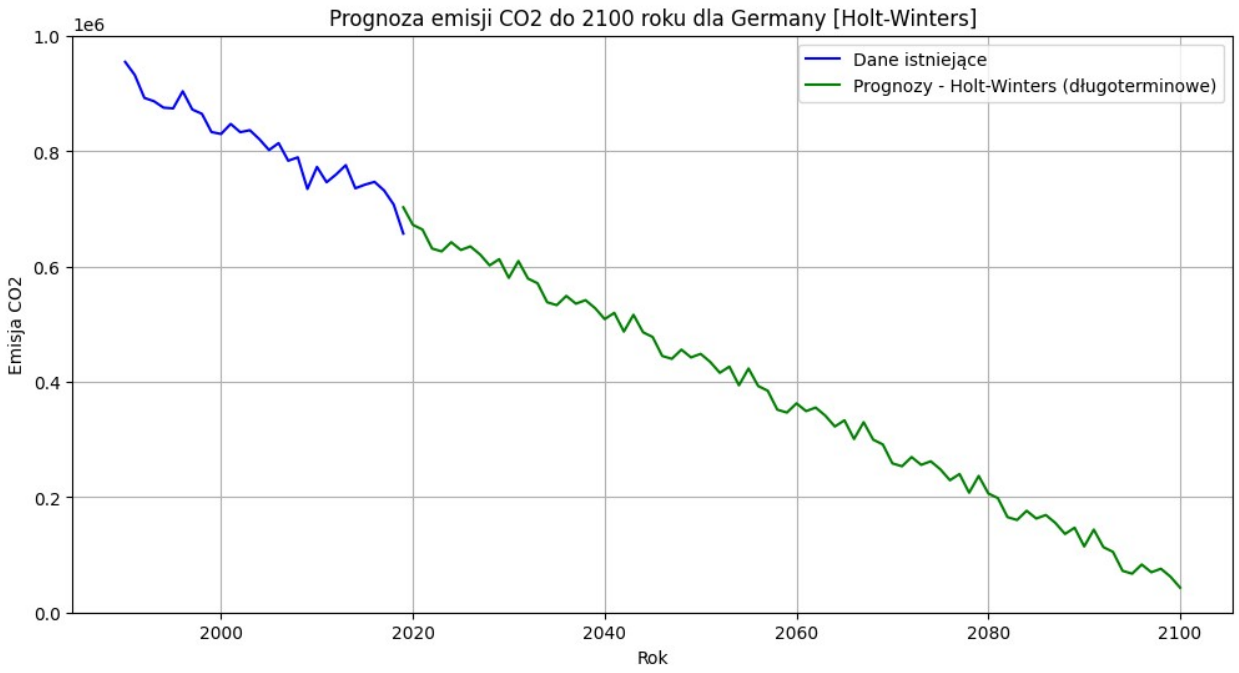
Holt-Winters to metoda wygładzania wykładniczego używana do analizy i prognozowania szeregów czasowych. Składa się z trzech komponentów: poziomu, trendu i sezonowości, co pozwala na modelowanie danych z trendami i okresowymi wzorcami. Wykres pokazuje, że model Holt-Winters przewiduje kontynuację rosnącego trendu emisji CO2 aż do 2100 roku w przypadku wszystkich państw. Widać, pewne fluktuacje świadczące o sezonowych zmianach w emisji CO2, co jest wynikiem uwzględnienia przez model Holt-Winters komponentu sezonowego.

```
try:
    panstwo = input("Podaj nazwę państwa dla którego ma być dokonana
predykcja:")
    data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
    data = data_CO2[data_CO2['country_name'] == panstwo]
    X = data['year']
    y = data['value']

    data.set_index('year', inplace=True)

    model_hw = ExponentialSmoothing(data['value'], trend='add',
seasonal='add', seasonal_periods=12)
    fit_hw = model_hw.fit()


    future_years = np.arange(2019, 2101)
    future_predictions_hw = fit_hw.forecast(len(future_years))

    y_true = data['value']
    y_pred = fit_hw.fittedvalues
```

```
    mse_hw2 = mean_squared_error(y_true, y_pred)
    mae_hw2 = mean_absolute_error(y_true, y_pred)

    print('Holt-Winters - Metryka MSE dla danych testowych:', mse_hw2)
    print('Holt-Winters - Metryka MAE dla danych testowych:', mae_hw2)

    plt.figure(figsize=(12, 6))
    plt.plot(data.index, data['value'], color='blue', label='Dane
istniejące')
    plt.plot(future_years, future_predictions_hw, color='green',
label='Prognozy - Holt-Winters (długoterminowe)')

    plt.title(f'Prognoza emisji CO2 do 2100 roku dla {panstwo} [Holt-
Winters]')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.legend()
    plt.grid(True)
    plt.show()
except:
    print("Nie ma takiego państwa, tylko angielskie nazwy z wielkiej
litery")
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\holtwinters\
model.py:917: ConvergenceWarning: Optimization failed to converge.
Check mle_retvals.
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
```

```
Holt-Winters - Metryka MSE dla danych testowych: 499099739.3976793
Holt-Winters - Metryka MAE dla danych testowych: 15375.114161076206
```



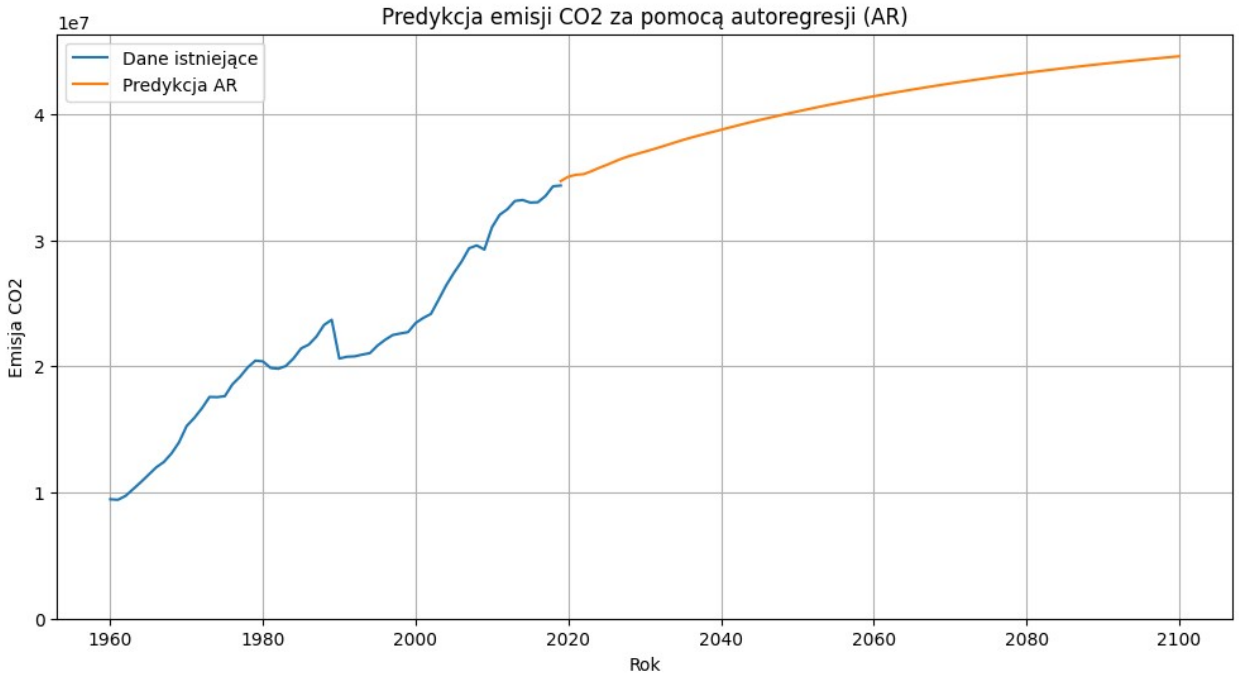
Próba predykcji emisji CO2 dla całego świata za pomocą **autoregresji**

```
data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
world_data = data_CO2[data_CO2['country_name'] == 'World']
X = world_data['year']
y = world_data['value']

train_size = int(0.8 * len(y))
train, test = y[:train_size], y[train_size:]

lags = int(input("Podaj liczbę opóźnień: "))
model = AutoReg(train, lags=lags)
model_fit = model.fit()

predicted_test = model_fit.predict(start=len(train), end=len(train) +
len(test) - 1)

mse_ar = mean_squared_error(test, predicted_test)
mae_ar = mean_absolute_error(test, predicted_test)
r2_ar = r2_score(test, predicted_test)

print('AR - Metryka MSE dla danych testowych:', mse_ar)
print('AR - Metryka MAE dla danych testowych:', mae_ar)
print('AR - Metryka R^2 dla danych testowych:', r2_ar)
```

```
model_full = AutoReg(y, lags=lags)
model_full_fit = model_full.fit()

future_years = np.arange(2019, 2101)
future_emission = model_full_fit.predict(start=len(y), end=len(y) +
len(future_years) - 1)

plt.figure(figsize=(12, 6))
plt.plot(X, y, '-', label='Dane istniejące')
plt.plot(future_years, future_emission, '-', label='Predykcja AR')
plt.title('Predykcja emisji CO2 za pomocą autoregresji (AR)')
plt.xlabel('Rok')
plt.ylabel('Emisja CO2')
plt.ylim(0, None)
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
```

```
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(

AR - Metryka MSE dla danych testowych: 581852072258.9084
AR - Metryka MAE dla danych testowych: 712277.142113369
AR - Metryka R^2 dla danych testowych: 0.7684006815774143
```



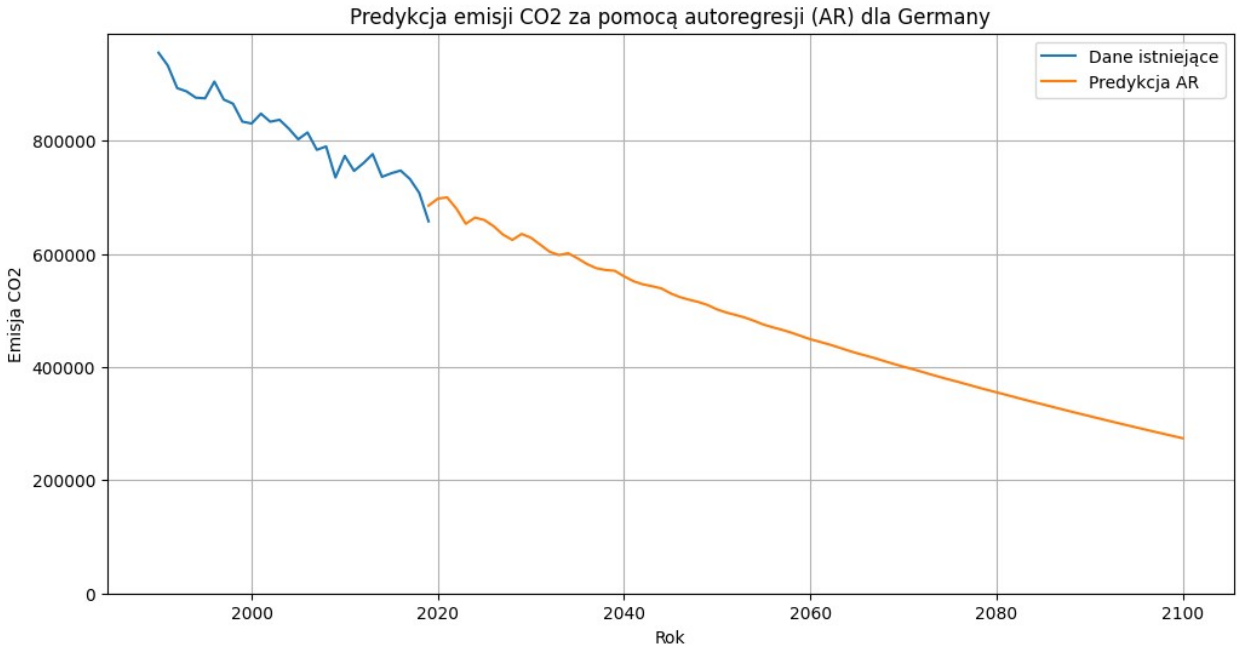
Autoregresja to model statystyczny stosowany w analizie szeregów czasowych, który prognozuje wartość zmiennej na podstawie jej wcześniejszych wartości. Model AR zakłada, że aktualna wartość szeregu czasowego jest liniową kombinacją przeszłych wartości i szumu błędu. Jest często stosowany do przewidywania przyszłych wartości na podstawie przeszłych danych. Prognoza modelu AR (pomarańczowa linia) na przyszłe lata do 2100 roku wskazuje na kontynuację wzrostowego trendu emisji CO2. Model przewiduje systematyczny, choć nieco spłaszczony wzrost emisji w przyszłości, co może sugerować stabilizację tempa wzrostu w dalszych latach.

```
try:
    panstwo = input("Podaj nazwę państwa dla którego ma być dokonana
predykcja:")

    data_CO2 = pd.read_csv('dane/co2_emissions_kt_by_country.csv')
```

```
    data = data_CO2[data_CO2['country_name'] == panstwo]
    X = data['year']
    y = data['value']

    train_size = int(0.8 * len(y))
    train, test = y[:train_size], y[train_size:]

    lags = int(input("Podaj liczbę opóźnień: "))
    model = AutoReg(train, lags=lags)
    model_fit = model.fit()

    predicted_test = model_fit.predict(start=len(train),
end=len(train) + len(test) - 1)

    mse_ar2 = mean_squared_error(test, predicted_test)
    mae_ar2 = mean_absolute_error(test, predicted_test)
    r2_ar2 = r2_score(test, predicted_test)

    print('AR - Metryka MSE dla danych testowych:', mse_ar2)
    print('AR - Metryka MAE dla danych testowych:', mae_ar2)
    print('AR - Metryka R^2 dla danych testowych:', r2_ar2)

    model_full = AutoReg(y, lags=lags)
    model_full_fit = model_full.fit()

    future_years = np.arange(2019, 2101)
    future_emission = model_full_fit.predict(start=len(y), end=len(y)
+ len(future_years) - 1)

    plt.figure(figsize=(12, 6))
    plt.plot(X, y, '-', label='Dane istniejące')
    plt.plot(future_years, future_emission, '-', label='Predykcja AR')
    plt.title(f'Predykcja emisji CO2 za pomocą autoregresji (AR) dla
{panstwo}')
    plt.xlabel('Rok')
    plt.ylabel('Emisja CO2')
    plt.ylim(0, None)
    plt.legend()
    plt.grid(True)
    plt.show()
except:
    print("Nie ma takiego państwa, tylko angielskie nazwy z wielkiej
litery")
```

```
AR - Metryka MSE dla danych testowych: 954953177.8884035
AR - Metryka MAE dla danych testowych: 26462.4600024189
AR - Metryka R^2 dla danych testowych: -0.0036664151832004332

C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
```

```
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:473: ValueWarning: An unsupported index was provided and
will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: ValueWarning: No supported index is available.
Prediction results will be given with an integer index beginning at
`start`.
  return get_prediction_index(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\statsmodels\tsa\base\
tsa_model.py:836: FutureWarning: No supported index is available. In
the next version, calling this method in a model without a supported
index will result in an exception.
  return get_prediction_index(
```



Podsumowanie

Oto zastosowane modele i ich metryki:

```
data = {
    "Model": ["Linear Regression (Example - World)", "Linear
Regression (Example - Selected Country)",
              "Polynomial Regression (Example - World)", "Polynomial
Regression (Example - Selected Country)",
              "ARIMA", "ARIMA (Example - Selected Country)", "Holt-
Winters", "Holt-Winters (Example - Selected Country)",
              "AR", "AR (Example - Selected Country)"],
    "R2": [r2_train_lr, r2_train_lr2, r2_train_pr, r2_train_pr2, '-',
'-', '-', '-', r2_ar, r2_ar2],
    "MSE": [mse_lr, mse_lr2, mse_pr, mse_pr2, mse_arima, mse_arima2,
mse_hw, mse_hw2, mse_ar, mse_ar2],
    "MAE": [mae_lr, mae_lr2, mae_pr, mae_pr2, mae_arima, mae_arima2,
mae_hw, mae_hw2, mae_ar, mae_ar2]
}

metrics_df = pd.DataFrame(data)
#metrics_df = metrics_df.sort_values(by="MAE")
metrics_df
```

```
                                                Model        R2
MSE  \
0                Linear Regression (Example - World)  0.942623
2.769227e+12
1     Linear Regression (Example - Selected Country)   0.93244
3.302100e+08
```

```
2            Polynomial Regression (Example - World)  0.970563
4.923672e+14
3  Polynomial Regression (Example - Selected Coun...  0.945339
3.302100e+08
4                                              ARIMA         -
2.104158e+12
5                  ARIMA (Example - Selected Country)         -
3.913465e+10
6                                       Holt-Winters         -
4.951192e+11
7          Holt-Winters (Example - Selected Country)         -
4.990997e+08
8                                                 AR  0.768401
5.818521e+11
9                       AR (Example - Selected Country) -0.003666
9.549532e+08

             MAE
0  1.495133e+06
1  1.442743e+04
2  2.103005e+07
3  1.442743e+04
4  5.705853e+05
5  5.780455e+04
6  4.779643e+05
7  1.537511e+04
8  7.122771e+05
9  2.646246e+04
```

Wnioski z analizy metryk modeli:

1. **Linear Regression (Example - World)**:
   - Bardzo wysokie (R^2 = 0.94) wskazuje na dobrą zgodność modelu z danymi.
   - MSE i MAE są wysokie, co sugeruje, że chociaż model dobrze dopasowuje się do danych, prognozy mogą być znacznie rozproszone.
2. **Linear Regression (Example - Selected Country)**:
   - Wysokie (R^2 = 0.93) również wskazuje na dobrą zgodność modelu z danymi.
   - MSE i MAE są znacznie niższe w porównaniu do modelu dla całego świata, co sugeruje lepszą dokładność prognoz dla wybranego kraju.
3. **Polynomial Regression (Example - World)**:
   - Najwyższe (R^2 = 0.97) wskazuje na najlepszą zgodność modelu z danymi.
   - Bardzo wysokie MSE i MAE sugerują, że mimo dobrej zgodności, model może mieć problemy z dokładnością prognoz.
4. **Polynomial Regression (Example - Selected Country)**:
   - Wysokie (R^2 = 0.95) wskazuje na dobrą zgodność modelu.
   - MSE i MAE są takie same jak w przypadku regresji liniowej dla wybranego kraju, co sugeruje podobną dokładność.
5. **ARIMA**:

   - Bardzo wysokie MSE i MAE sugerują, że model ma problemy z dokładnością prognoz.
6. **ARIMA (Example - Selected Country)**:
   - MSE i MAE są znacznie niższe niż w przypadku modelu dla całego świata, co wskazuje na lepszą dokładność prognoz dla wybranego kraju.
7. **Holt-Winters**:
   - Wysokie MSE i MAE sugerują, że model nie jest dokładny.
8. **Holt-Winters (Example - Selected Country)**:
   - MSE i MAE są znacznie niższe niż w przypadku modelu dla całego świata, co wskazuje na lepszą dokładność prognoz dla wybranego kraju.
9. **AR**:
   - (R^2 = 0.77) wskazuje na umiarkowaną zgodność modelu.
   - Wysokie MSE i MAE sugerują problemy z dokładnością prognoz.
10. **AR (Example - Selected Country)**:
    - Negatywne (R^2 = -1.77) wskazuje na bardzo złą zgodność modelu.
    - Bardzo wysokie MSE i MAE sugerują, że model jest nieodpowiedni do prognoz dla wybranego kraju.

W przypadku ARIMA i Holt-Winters obliczanie R^2 jest niezalecane.

Najlepszym modelem wydaje się być regresja wielomianowa, można jeszcze próbować dostosować odpowiedni stopień wielomianów, w powyższym przykładzie jest to 3.

2. Globalny poziom morza

Następny zestaw danych do analizy dotyczy pomiarów poziomu morza w latach 1993 - 2021.

Plik sealevel.csv zawiera następujące cechy:

- Year: Rok pomiaru.
- TotalWeightedObservations: Suma obserwacji ważonych.
- GMSL_noGIA: Globalny poziom morza (Global Mean Sea Level) bez korekty związanej z działalnością lądolodów (Glacial Isostatic Adjustment - GIA). Jest to średnia wysokość poziomu morza w [mm] na całym świecie w stosunku do określonego odniesienia, nie uwzględniając wpływu zmian masy lądolodów.
- StdDevGMSL_noGIA: Odchylenie standardowe globalnego poziomu morza bez korekty GIA. Wskazuje ono na zmienność w wysokości poziomu morza w ciągu danego roku.
- SmoothedGSML_noGIA: Wygładzony globalny poziom morza bez korekty GIA.
- GMSL_GIA: Globalny poziom morza z uwzględnieniem korekty GIA. Korekta GIA uwzględnia wpływ zmian masy lądolodów na wysokość poziomu morza.
- StdDevGMSL_GIA: Odchylenie standardowe globalnego poziomu morza z uwzględnieniem korekty GIA.

- SmoothedGSML_GIA: Wygładzony globalny poziom morza z uwzględnieniem korekty GIA.
- SmoothedGSML_GIA_sigremoved: Wygładzony globalny poziom morza z uwzględnieniem korekty GIA, z usuniętymi sygnałami, które nie są istotne z punktu widzenia analizy lub nieodpowiednio wyznaczonymi.

Wczytanie danych:

```
data_sealevel = pd.read_csv('dane/sealevel.csv')
data_sealevel.head(10)
```

```
   Year  TotalWeightedObservations  GMSL_noGIA  StdDevGMSL_noGIA  \
0  1993                  327401.31      -38.59             89.86
1  1993                  324498.41      -41.97             90.86
2  1993                  333018.19      -41.93             87.27
3  1993                  297483.19      -42.67             90.75
4  1993                  321635.81      -37.86             90.26
5  1993                  291945.91      -36.09             89.99
6  1993                  327830.00      -36.11             88.74
7  1993                  326320.41      -35.52             89.49
8  1993                  322331.00      -35.47             88.79
9  1993                  331127.31      -39.25             98.10

   SmoothedGSML_noGIA  GMSL_GIA  StdDevGMSL_GIA  SmoothedGSML_GIA  \
0              -38.76    -38.59           89.86             -38.75
1              -39.78    -41.97           90.86             -39.77
2              -39.62    -41.91           87.27             -39.61
3              -39.67    -42.65           90.74             -39.64
4              -38.75    -37.83           90.25             -38.72
5              -37.71    -36.05           89.99             -37.67
6              -36.85    -36.06           88.74             -36.81
7              -36.32    -35.47           89.49             -36.27
8              -36.11    -35.41           88.78             -36.05
9              -36.17    -39.19           98.09             -36.11

   SmoothedGSML_GIA_sigremoved
0                       -38.57
1                       -39.11
2                       -38.58
3                       -38.34
4                       -37.21
5                       -35.98
6                       -34.94
7                       -34.19
8                       -33.72
9                       -33.48
```

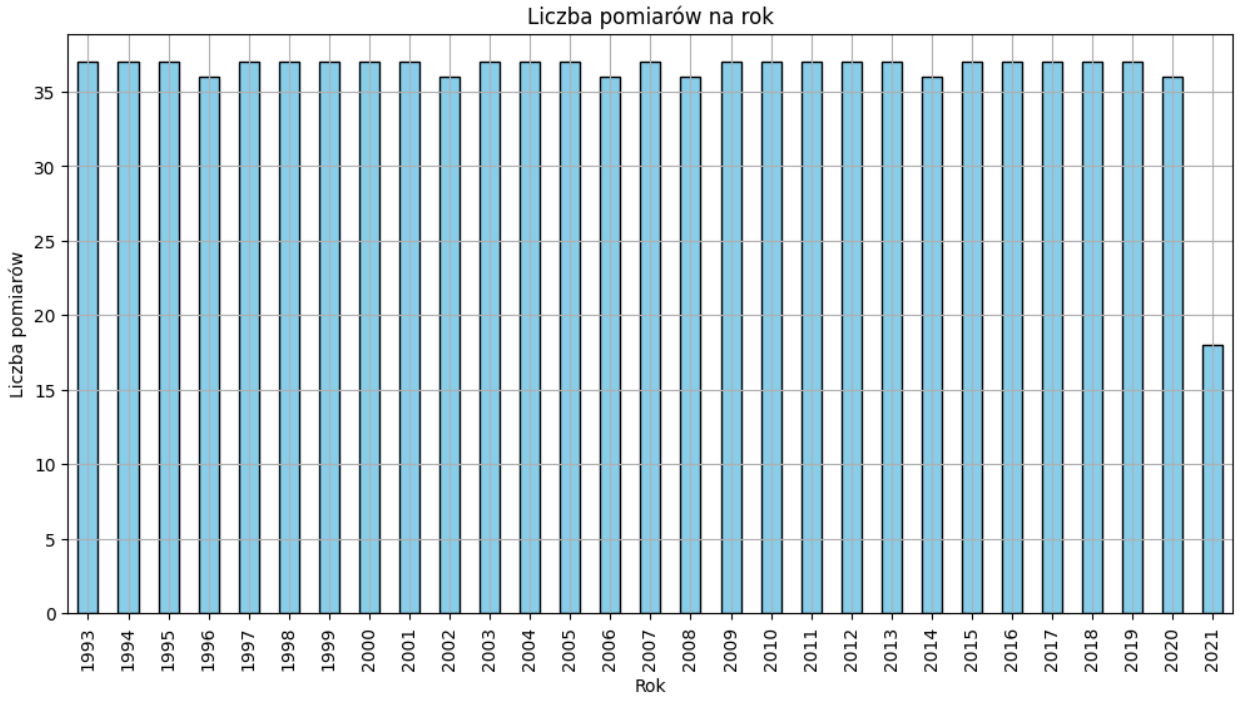
Sprawdźmy czy są braki w danych.

```
print(data_sealevel.isnull().sum())
```

```
Year                            0
TotalWeightedObservations       0
GMSL_noGIA                      0
StdDevGMSL_noGIA                0
SmoothedGSML_noGIA              0
GMSL_GIA                        0
StdDevGMSL_GIA                  0
SmoothedGSML_GIA                0
SmoothedGSML_GIA_sigremoved     0
dtype: int64
```

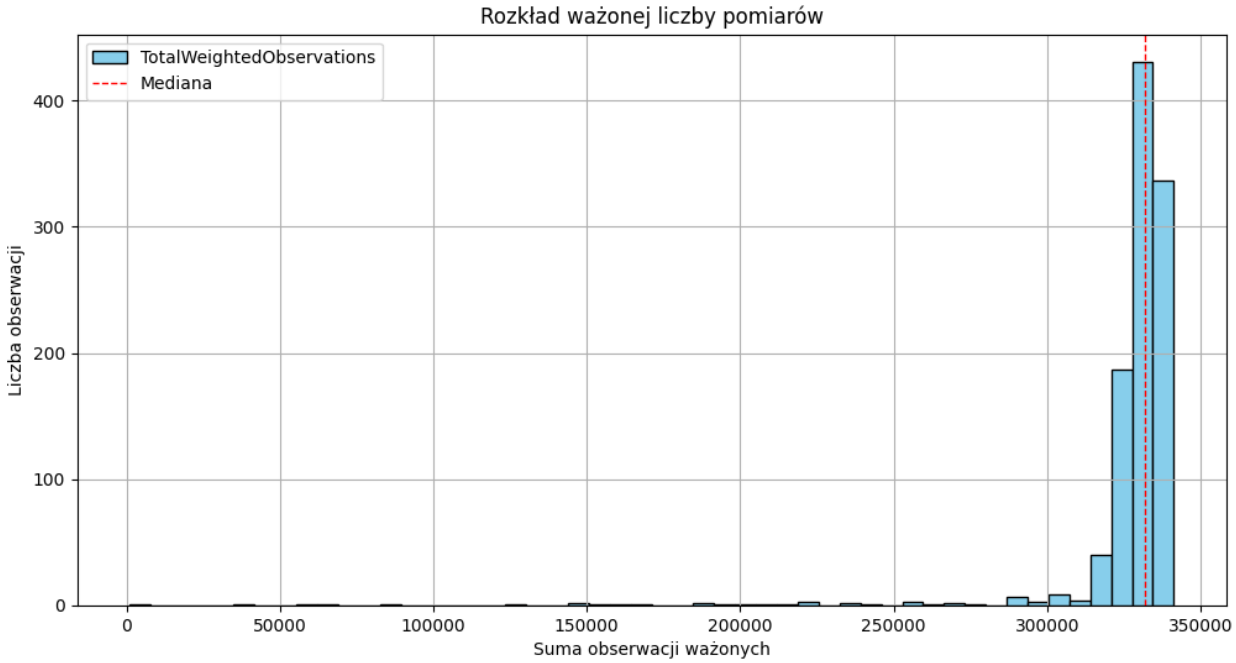
Liczba obserwacji w danym roku

```
plt.figure(figsize=(12, 6))
data_sealevel.Year.value_counts().sort_index().plot(kind='bar',
color='skyblue', edgecolor='black')
plt.title('Liczba pomiarów na rok')
plt.xlabel('Rok')
plt.ylabel('Liczba pomiarów')
plt.grid(True)
plt.show()
```

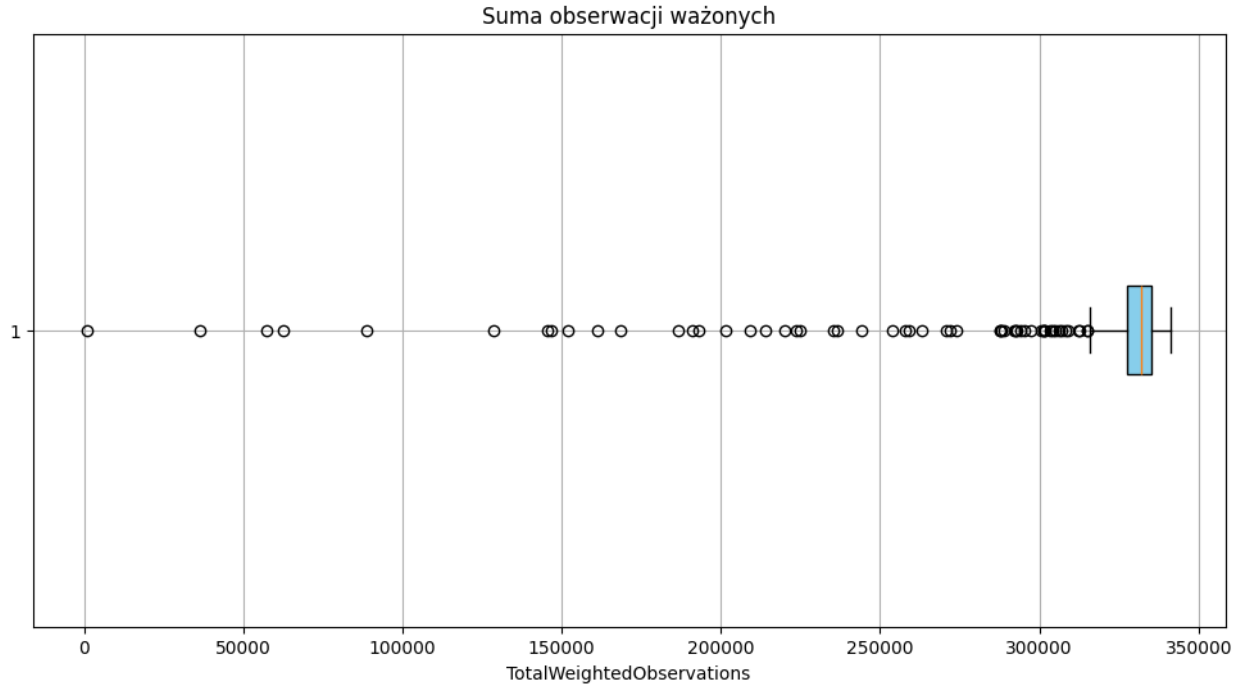



```
plt.figure(figsize=(12, 6))
data_sealevel.TotalWeightedObservations.plot(kind='hist', bins=50,
color='skyblue', edgecolor='black')
```

```
plt.axvline(data_sealevel.TotalWeightedObservations.median(),
color='red', linestyle='dashed', linewidth=1, label='Mediana')
plt.title('Rozkład ważonej liczby pomiarów')
plt.xlabel('Suma obserwacji ważonych')
plt.ylabel('Liczba obserwacji')
plt.legend()
plt.grid(True)
plt.show()
```

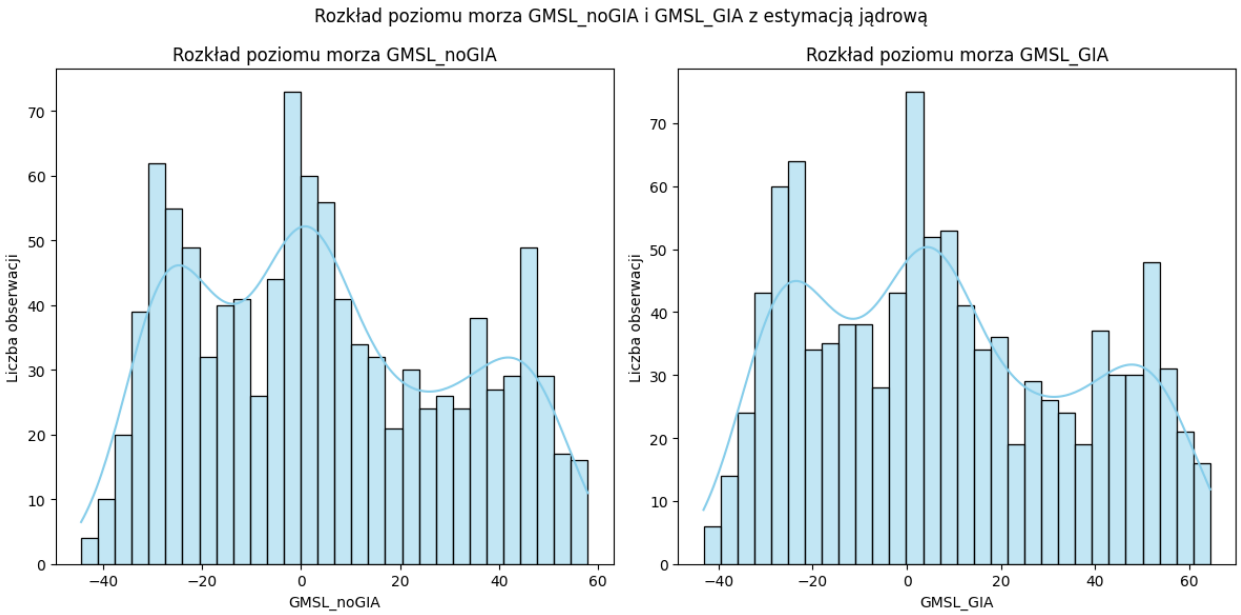



```
plt.figure(figsize=(12, 6))
plt.boxplot(data_sealevel.TotalWeightedObservations, vert=False,
patch_artist=True, boxprops=dict(facecolor='skyblue', color='black'))
plt.title('Suma obserwacji ważonych')
plt.xlabel('TotalWeightedObservations')
plt.grid(True)
plt.show()
```



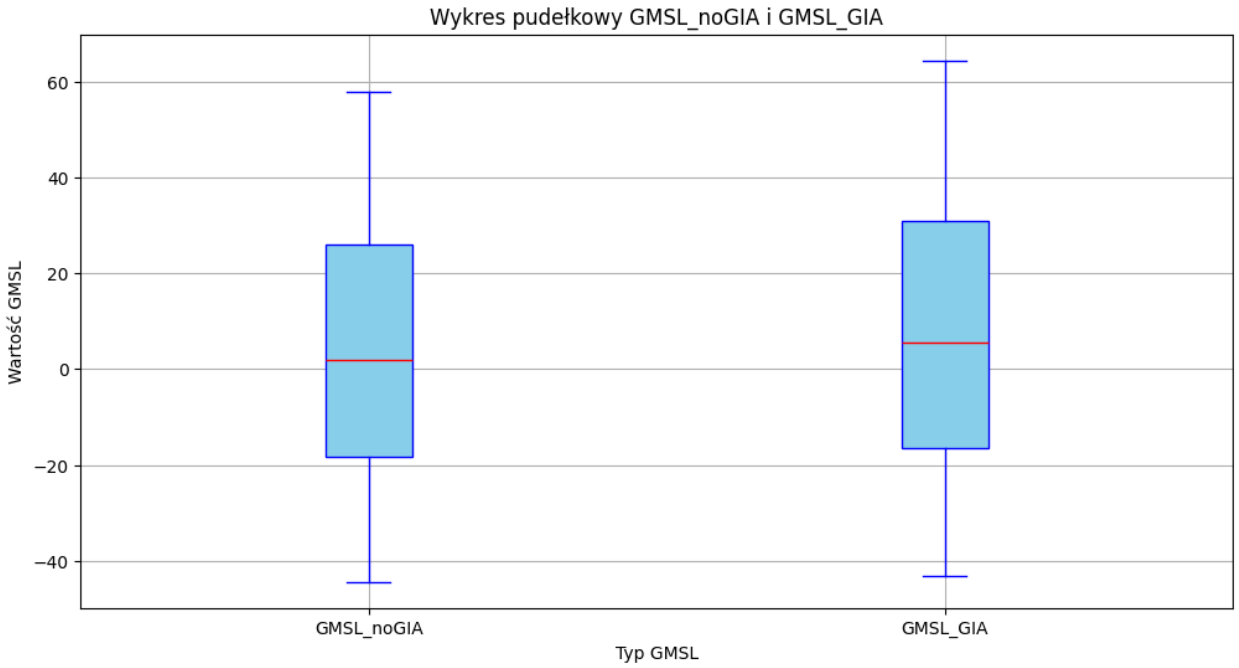
Wykresy z rozkładami zmiennych:

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(data_sealevel['GMSL_noGIA'], bins=30, kde=True,
color='skyblue', edgecolor='black')
plt.title('Rozkład poziomu morza GMSL_noGIA')
plt.xlabel('GMSL_noGIA')
plt.ylabel('Liczba obserwacji')

plt.subplot(1, 2, 2)
sns.histplot(data_sealevel['GMSL_GIA'], bins=30, kde=True,
color='skyblue', edgecolor='black')
plt.title('Rozkład poziomu morza GMSL_GIA')
plt.xlabel('GMSL_GIA')
plt.ylabel('Liczba obserwacji')

plt.suptitle('Rozkład poziomu morza GMSL_noGIA i GMSL_GIA z estymacją
jądrową')
plt.tight_layout()
plt.show()
```



Wizualizacja z użyciem wykresu pudełkowego dla GMSL

```
plt.figure(figsize=(12, 6))
boxplot = data_sealevel.boxplot(column=['GMSL_noGIA', 'GMSL_GIA'],
patch_artist=True,
                                boxprops=dict(facecolor='skyblue',
color='blue'),
                                whiskerprops=dict(color='blue'),
                                capprops=dict(color='blue'),
                                medianprops=dict(color='red'),

flierprops=dict(markeredgecolor='blue', markerfacecolor='blue'))

plt.title('Wykres pudełkowy GMSL_noGIA i GMSL_GIA')
plt.xlabel('Typ GMSL')
plt.ylabel('Wartość GMSL')
plt.grid(True)
plt.show()
```

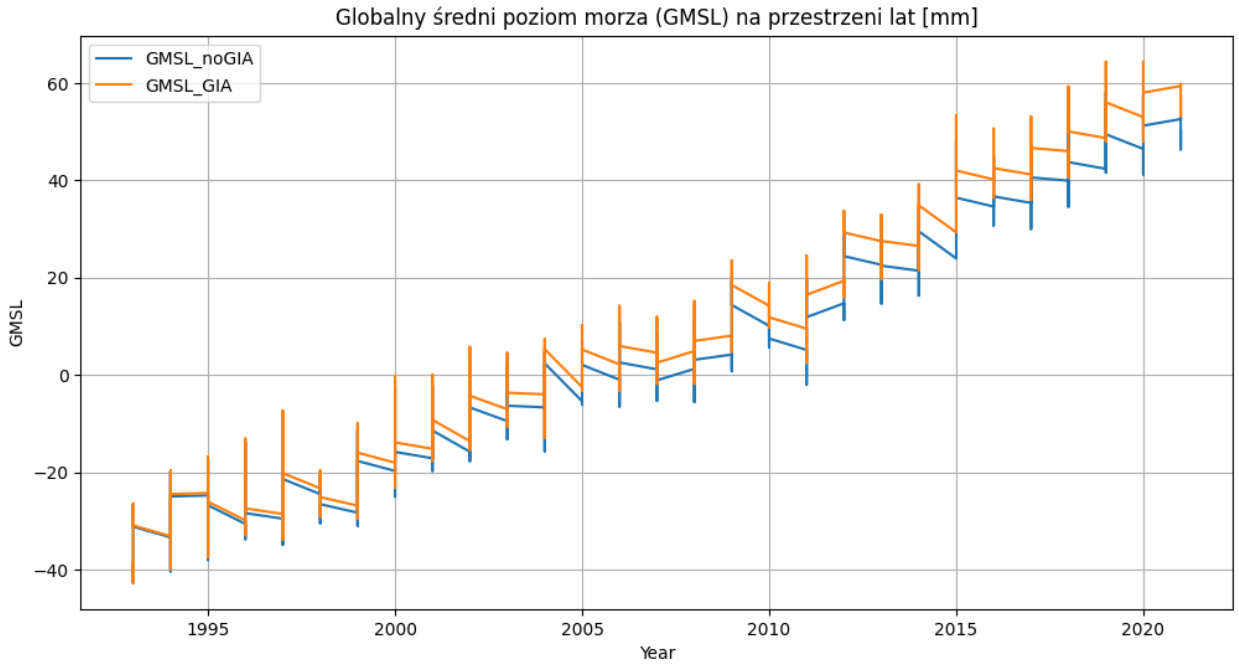


Usunięcie obserwacji odstającej

```
data_sealevel[data_sealevel.TotalWeightedObservations<25000]
```

```
     Year  TotalWeightedObservations  GMSL_noGIA  StdDevGMSL_noGIA  \
107  1995                     906.10      -44.39             77.41

     SmoothedGSML_noGIA  GMSL_GIA  StdDevGMSL_GIA  SmoothedGSML_GIA  \
107              -25.34    -43.14           77.42            -24.63

     SmoothedGSML_GIA_sigremoved
107                       -26.87
```

```
data_sealevel =
data_sealevel[data_sealevel.TotalWeightedObservations>25000]

plt.figure(figsize=(12, 6))
plt.plot(data_sealevel['Year'], data_sealevel['GMSL_noGIA'],
label='GMSL_noGIA')
plt.plot(data_sealevel['Year'], data_sealevel['GMSL_GIA'],
label='GMSL_GIA')
plt.xlabel('Year')
plt.ylabel('GMSL')
plt.title('Globalny średni poziom morza (GMSL) na przestrzeni lat
[mm]')
plt.legend()
plt.grid(True)
plt.show()
```



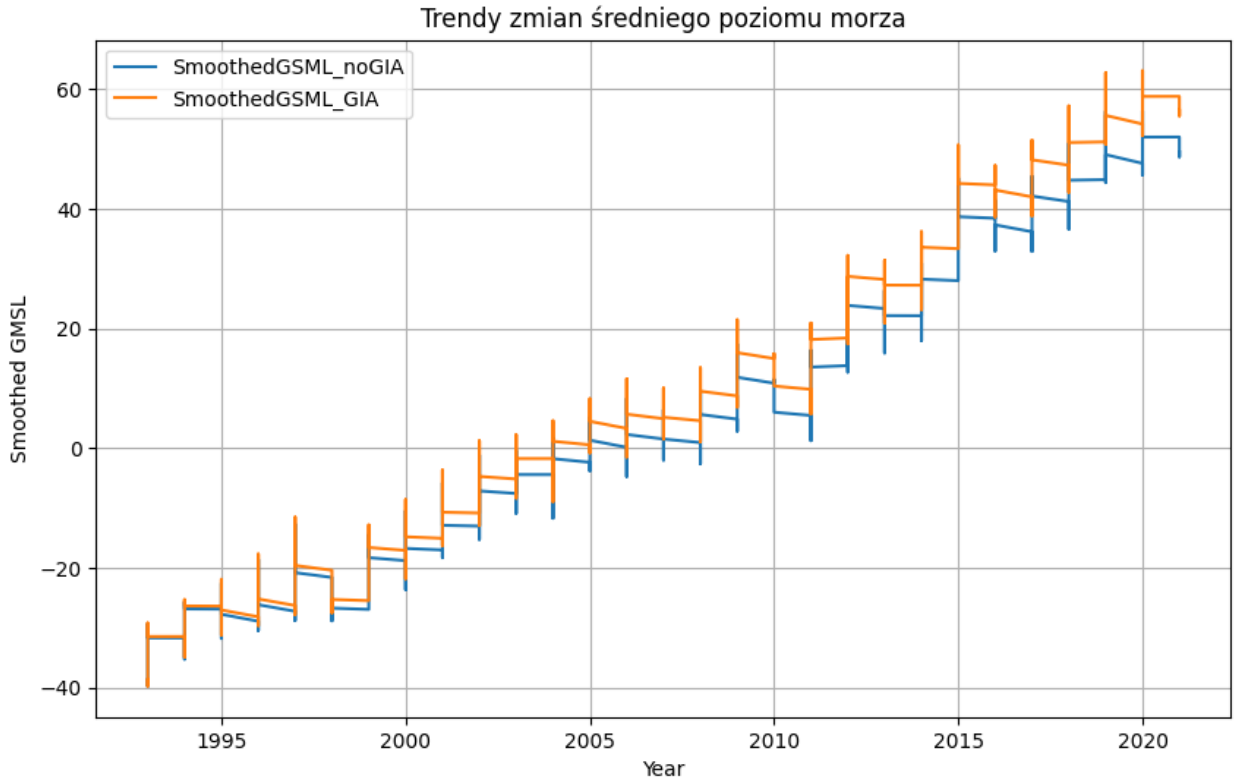
Statystyki opisowe

```
print("\nStatystyki opisowe dla kolumny GMSL_noGIA:")
print(data_sealevel['GMSL_noGIA'].describe())

print("\nStatystyki opisowe dla kolumny GMSL_GIA:")
print(data_sealevel['GMSL_GIA'].describe())
```

```
Statystyki opisowe dla kolumny GMSL_noGIA:
count   1047.00
mean       4.69
std       26.32
min      -42.67
25%      -18.21
50%        1.94
75%       25.87
max       57.92
Name: GMSL_noGIA, dtype: float64

Statystyki opisowe dla kolumny GMSL_GIA:
count   1047.00
mean       8.16
std       28.28
min      -42.65
25%      -16.61
50%        5.55
75%       30.95
```

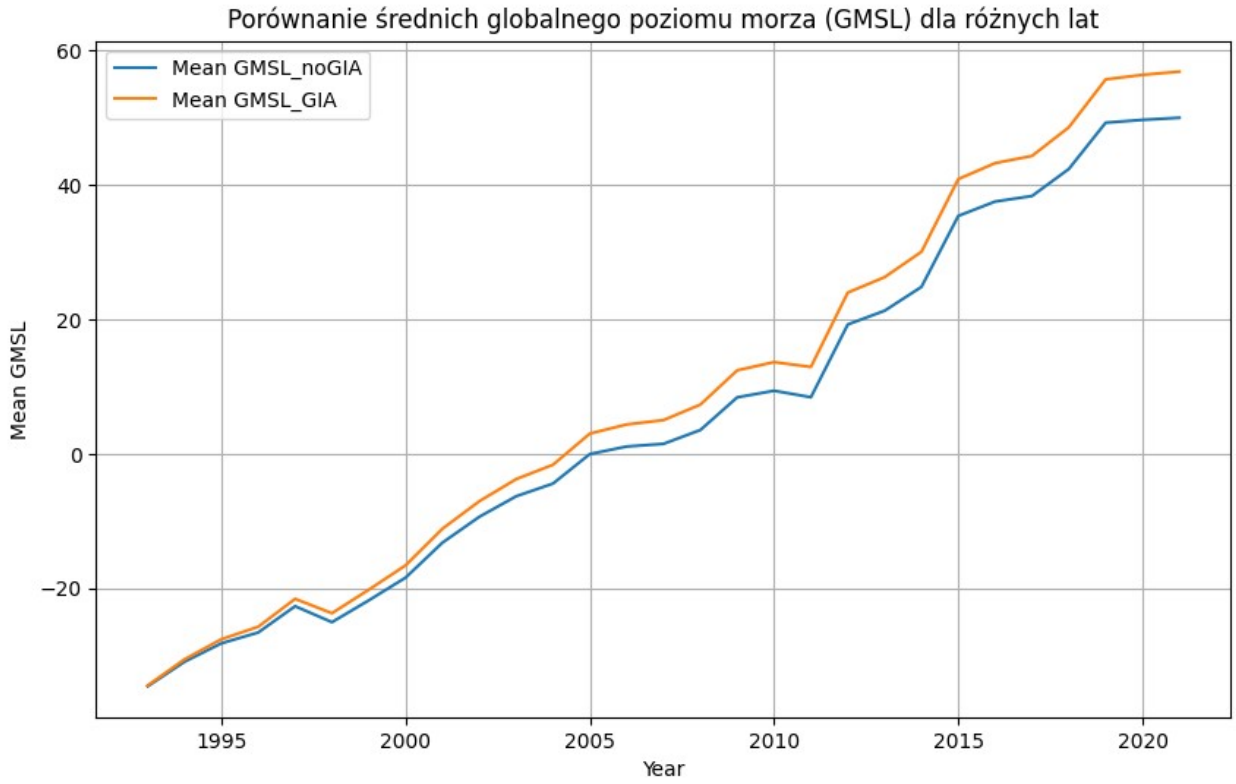
```
max       64.39
Name: GMSL_GIA, dtype: float64
```

```
plt.figure(figsize=(10, 6))
plt.plot(data_sealevel['Year'], data_sealevel['SmoothedGSML_noGIA'],
label='SmoothedGSML_noGIA')
plt.plot(data_sealevel['Year'], data_sealevel['SmoothedGSML_GIA'],
label='SmoothedGSML_GIA')
plt.xlabel('Year')
plt.ylabel('Smoothed GMSL')
plt.title('Trendy zmian średniego poziomu morza')
plt.legend()
plt.grid(True)
plt.show()
```




```
mean_gmsl_noGIA = data_sealevel.groupby('Year')['GMSL_noGIA'].mean()
mean_gmsl_GIA = data_sealevel.groupby('Year')['GMSL_GIA'].mean()

plt.figure(figsize=(10, 6))
plt.plot(mean_gmsl_noGIA.index, mean_gmsl_noGIA.values, label='Mean
GMSL_noGIA')
plt.plot(mean_gmsl_GIA.index, mean_gmsl_GIA.values, label='Mean
GMSL_GIA')
plt.xlabel('Year')
plt.ylabel('Mean GMSL')
```

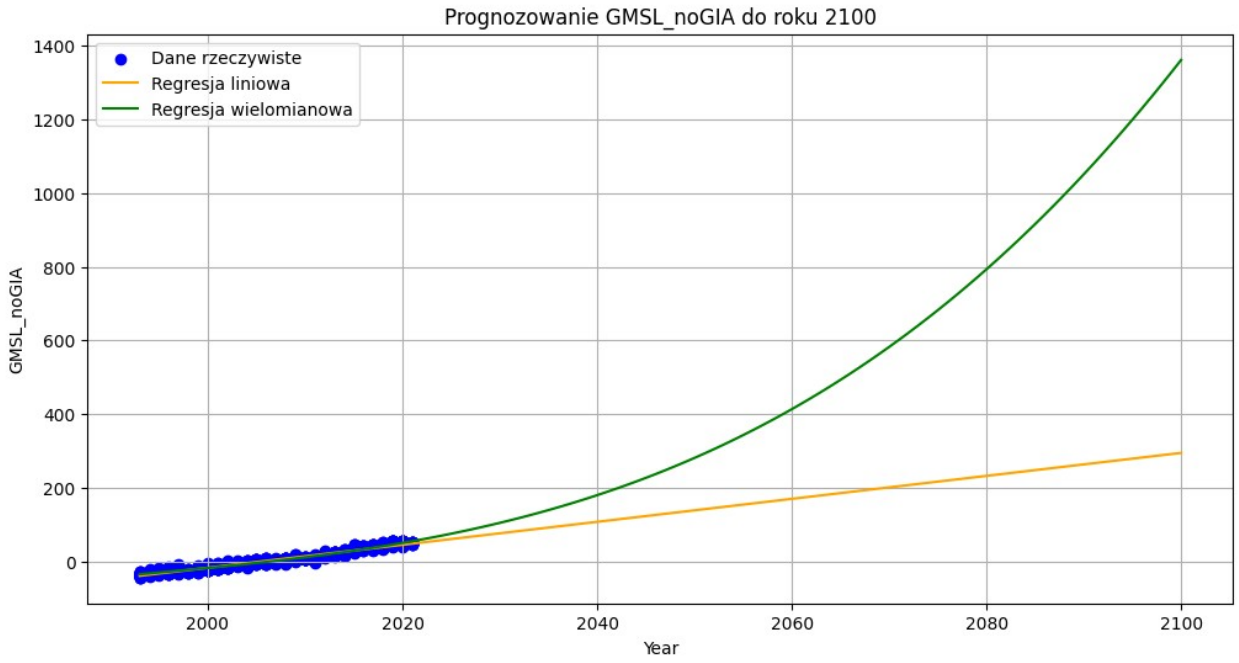
```
plt.title('Porównanie średnich globalnego poziomu morza (GMSL) dla różnych lat')
plt.legend()
plt.grid(True)
plt.show()
```




```
lin_reg = LinearRegression()
poly_features = PolynomialFeatures(degree=3)
poly_reg = LinearRegression()

lin_reg.fit(data_sealevel['Year'].values.reshape(-1, 1),
data_sealevel['GMSL_noGIA'])

X_poly =
poly_features.fit_transform(data_sealevel['Year'].values.reshape(-1,
1))
poly_reg.fit(X_poly, data_sealevel['GMSL_noGIA'])

X_future = np.arange(1993, 2101).reshape(-1, 1)

lin_pred_future = lin_reg.predict(X_future)

X_poly_future = poly_features.transform(X_future)
poly_pred_future = poly_reg.predict(X_poly_future)
```

```
plt.figure(figsize=(12, 6))

plt.scatter(data_sealevel['Year'], data_sealevel['GMSL_noGIA'],
label='Dane rzeczywiste', color='blue')

plt.plot(X_future, lin_pred_future, label='Regresja liniowa',
color='orange')
plt.plot(X_future, poly_pred_future, label='Regresja wielomianowa',
color='green')

plt.xlabel('Year')
plt.ylabel('GMSL_noGIA')
plt.title('Prognozowanie GMSL_noGIA do roku 2100')
plt.legend()
plt.grid(True)

r2_lin = r2_score(data_sealevel['GMSL_noGIA'],
lin_reg.predict(data_sealevel['Year'].values.reshape(-1, 1)))
r2_poly = r2_score(data_sealevel['GMSL_noGIA'],
poly_reg.predict(poly_features.transform(data_sealevel['Year'].values.
reshape(-1, 1))))

print(f'R2 dla regresji liniowej: {r2_lin}')
print(f'R2 dla regresji wielomianowej: {r2_poly}')


plt.show()
```

```
R2 dla regresji liniowej: 0.9478215456552407
R2 dla regresji wielomianowej: 0.9573563573935061
```



//DODAĆ OPIS

3. Średnia temperatura w poszczególnych państwach w latach 1743 - 2013

// OPIS DANYCH

Wczytanie danych:

```
data_temperature =
pd.read_csv('dane/GlobalLandTemperaturesByCountry.csv')
print(data_temperature.head(10))


           dt  AverageTemperature  AverageTemperatureUncertainty
Country
0  1743-11-01               4.384                          2.294
Åland
1  1743-12-01                 NaN                            NaN
Åland
2  1744-01-01                 NaN                            NaN
Åland
3  1744-02-01                 NaN                            NaN
Åland
4  1744-03-01                 NaN                            NaN
Åland
5  1744-04-01               1.530                          4.680
Åland
6  1744-05-01               6.702                          1.789
Åland
7  1744-06-01              11.609                          1.577
Åland
8  1744-07-01              15.342                          1.410
Åland
9  1744-08-01                 NaN                            NaN
Åland
```

Mamy dużą liczbę wartości pustych zwłaszcza w danych z XVIII i XIX w. można więc usunąć wartości puste.

```
data_temperature = data_temperature.dropna()
print(data_temperature.head(10))

            dt  AverageTemperature  AverageTemperatureUncertainty
Country
0   1743-11-01               4.384                          2.294
Åland
5   1744-04-01               1.530                          4.680
Åland
```

```
6   1744-05-01                6.702                                1.789
Åland
7   1744-06-01               11.609                                1.577
Åland
8   1744-07-01               15.342                                1.410
Åland
10  1744-09-01               11.702                                1.517
Åland
11  1744-10-01                5.477                                1.862
Åland
12  1744-11-01                3.407                                1.425
Åland
13  1744-12-01               -2.181                                1.641
Åland
14  1745-01-01               -3.850                                1.841
Åland
```

Można by wybrać Państwo i miesiąc dla którego przeprowadzano pomiary, następnie zwizualizować dane. Poniższe trzy wykresy przedstawiają średnią temperature dla danego państwa, a także niepewności pomiarowe i regresje liniową, która rezprezentuje linie trendu. Ponadto na ostatnim wykresie znajduje się sama linia trendu (wielominan drugiego stopnia).

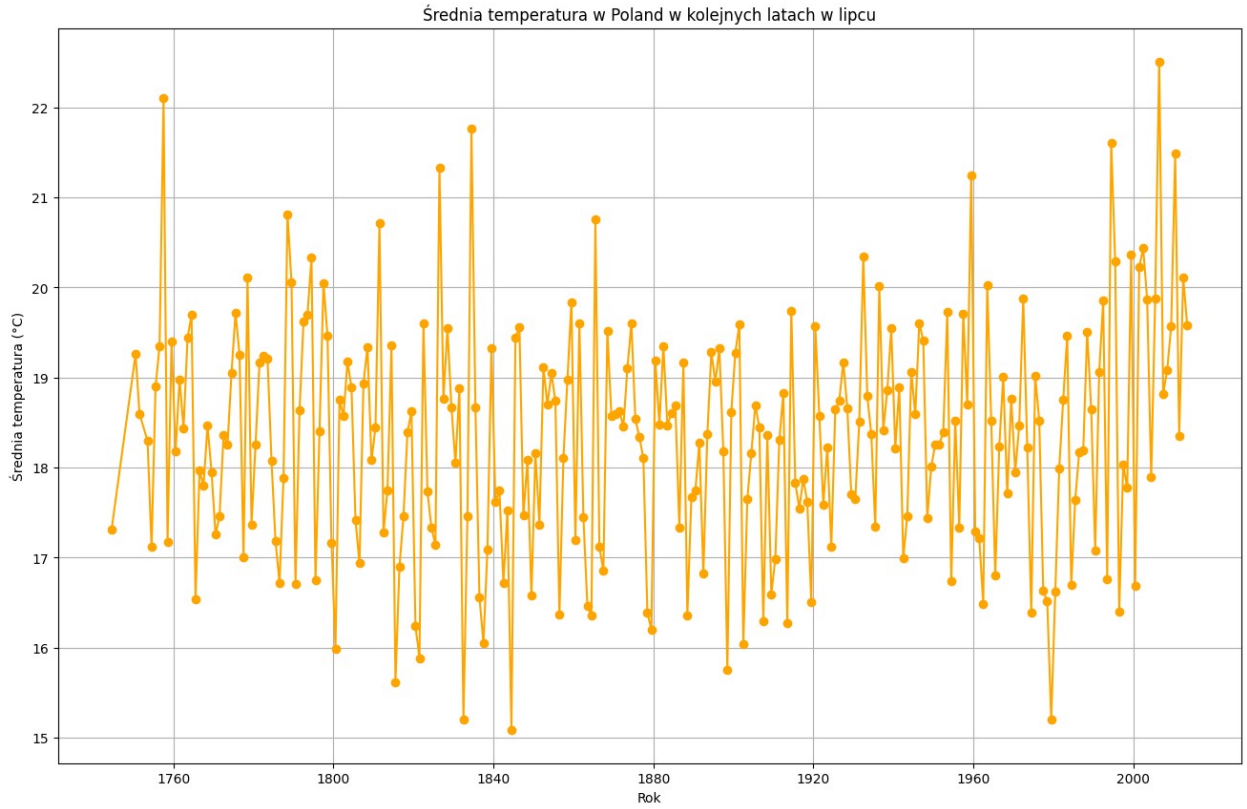
```
try:
    data_temperature['dt'] = pd.to_datetime(data_temperature['dt'])
    month = int(input("Podaj miesiąc pomiarów (1 - 12): "))
    country_name = input("Podaj państwo dla którego chcesz uzyskać
dane odnośnie temperatury: ")
    country_temperature =
data_temperature[(data_temperature['dt'].dt.month == month) &
(data_temperature['Country'] == country_name)]

    month_name = ['styczniu', 'lutym', 'marcu', 'kwietniu', 'maju',
'czerwcu', 'lipcu', 'sierpniu', 'wrześniu', 'październiku',
'listopadzie', 'grudniu']

    country_temperature['dt'] =
pd.to_datetime(country_temperature['dt'])


    country_temperature['Year'] = country_temperature['dt'].dt.year


    trend_line_calculate = np.polyfit(country_temperature['Year'],
country_temperature['AverageTemperature'], 2)
    trend_line_calcualte_fn = np.poly1d(trend_line_calculate)

    temperature_1850 = trend_line_calcualte_fn(1850)
    temperature_1900 = trend_line_calcualte_fn(1900)
    temperature_1950 = trend_line_calcualte_fn(1950)
```

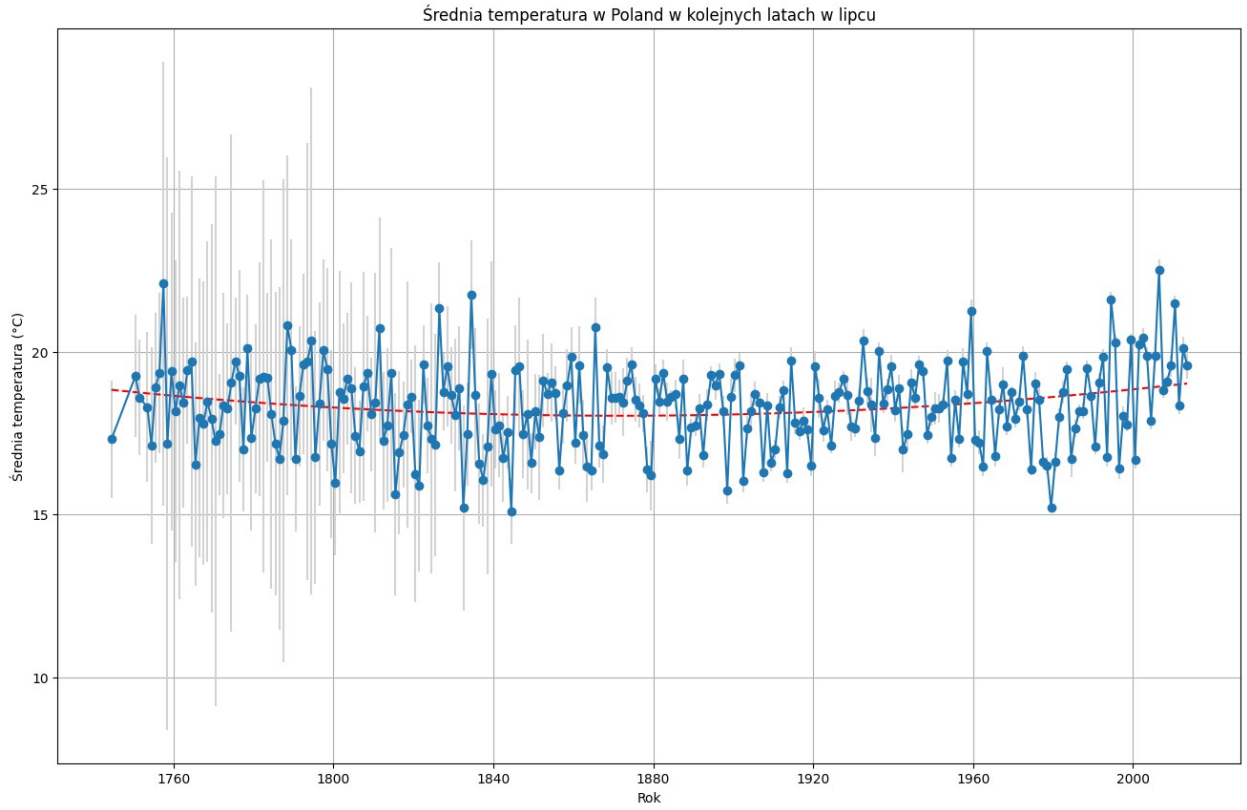
```
    temperature_2000 = trend_line_calcualte_fn(2000)
    temperature_2024 = trend_line_calcualte_fn(2024)
    temperature_2050 = trend_line_calcualte_fn(2050)
    temperature_2100 = trend_line_calcualte_fn(2100)

    trend_line_draw = np.polyfit(country_temperature.index,
country_temperature['AverageTemperature'], 2)
    trend_line_draw_fn = np.poly1d(trend_line_draw)

    predictions = pd.DataFrame({'Rok': [1850, 1900, 1950, 2000,2024,
2050, 2100],'Przewidywana temperatura (°C)': [temperature_1850,
temperature_1900, temperature_1950, temperature_2000,temperature_2024,
temperature_2050, temperature_2100]})
    print(predictions)

    future_trend = []
    for i in range(1950,2100):
        future_trend.append(trend_line_calcualte_fn(i))

    future_years=np.arange(1950,2100)

    plt.figure(figsize=(16, 10))
    plt.plot(country_temperature['dt'],
country_temperature['AverageTemperature'], marker='o',
linestyle='-',color='orange')
    plt.title('Średnia temperatura w ' + country_name + ' w kolejnych
latach w ' + month_name[month-1])
    plt.xlabel('Rok')
    plt.ylabel('Średnia temperatura (°C)')
    plt.grid(True)
    plt.show()

    plt.figure(figsize=(16, 10))
    plt.errorbar(country_temperature['dt'],
country_temperature['AverageTemperature'],
yerr=country_temperature['AverageTemperatureUncertainty'], marker='o',
linestyle='-', label='Średnia temperatura', ecolor='lightgray')
    plt.plot(country_temperature['dt'],
trend_line_draw_fn(country_temperature.index), linestyle='--',
color='red', label='Linia trendu')
    plt.title('Średnia temperatura w ' + country_name + ' w kolejnych
latach w ' + month_name[month-1])
    plt.xlabel('Rok')
    plt.ylabel('Średnia temperatura (°C)')
    plt.grid(True)
    plt.show()

    plt.figure(figsize=(16, 10))
    plt.plot(future_years, future_trend, linestyle='-',
color='orange', label='Linia trendu')
```

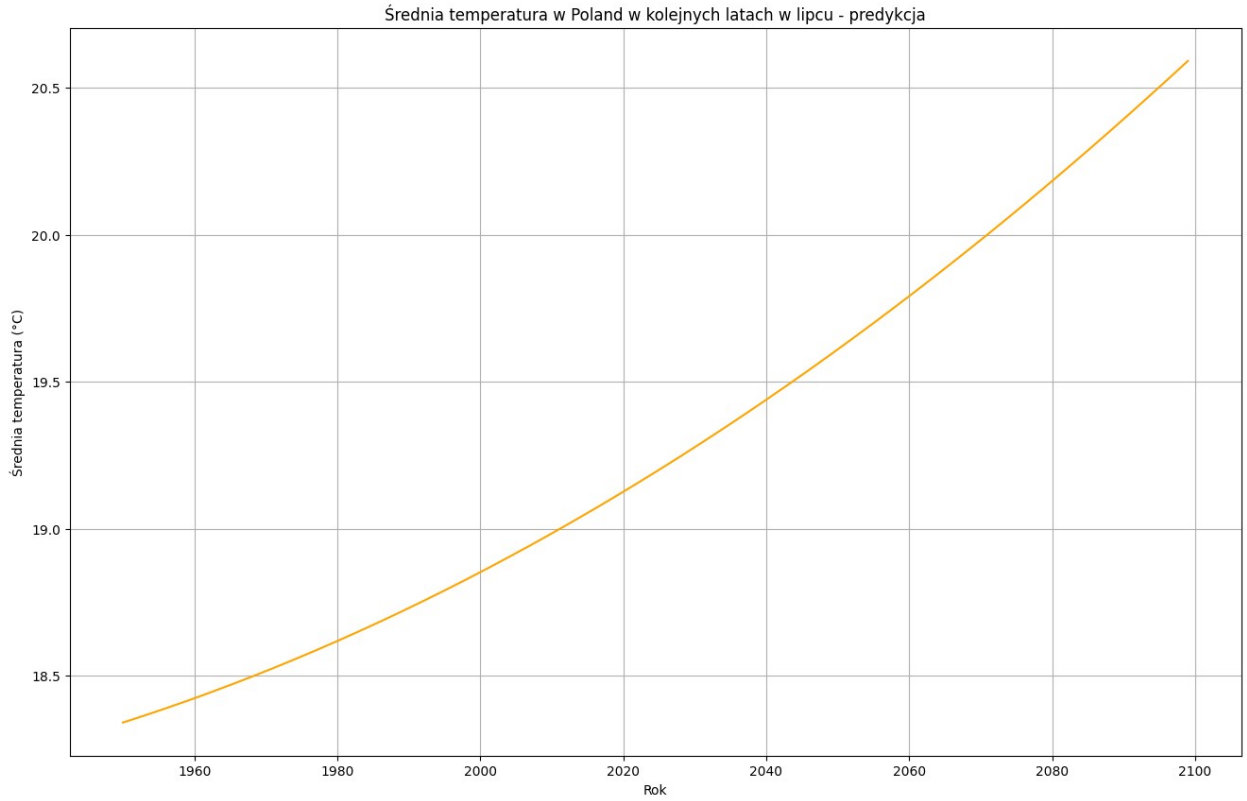
```
    plt.title('Średnia temperatura w ' + country_name + ' w kolejnych
latach w ' + month_name[month-1] + ' - predykcja')
    plt.xlabel('Rok')
    plt.ylabel('Średnia temperatura (°C)')
    plt.grid(True)
    plt.show()


except:
    print("Wystąpił błąd. Upewnij się, że nazwa państwa (duża litera i
angielska nazwa) oraz miesiąc (liczba 1 - 12) są poprawne.")
```

```
C:\Users\kubar\AppData\Local\Temp\ipykernel_19596\471621561.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  country_temperature['dt'] =
pd.to_datetime(country_temperature['dt'])
C:\Users\kubar\AppData\Local\Temp\ipykernel_19596\471621561.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  country_temperature['Year'] = country_temperature['dt'].dt.year
```

	Rok	Przewidywana temperatura (°C)
0	1850	18.056832
1	1900	18.075862
2	1950	18.341003
3	2000	18.852255
4	2024	19.185075
5	2050	19.609618
6	2100	20.613093


Średnia temperatura w Poland w kolejnych latach w lipcu
Średnia temperatura (°C)
Rok
22
21
20
19
18
17
16
15
1760
1800
1840
1880
1920
1960
2000


Średnia temperatura w Poland w kolejnych latach w lipcu
Średnia temperatura (°C)
Rok
25
20
15
10
1760
1800
1840
1880
1920
1960
2000



4. Stopień zalesienia danego państwa

//OPIS DANYCH ITP.

Wczytanie danych:

```
data_forest = pd.read_csv('dane/forest_percent_by_country.csv')
print(data_forest.head(10))
```

```
  country_code country_name  year     value
0          ABW        Aruba  1990  2.333333
1          ABW        Aruba  1991  2.333333
2          ABW        Aruba  1992  2.333333
3          ABW        Aruba  1993  2.333333
4          ABW        Aruba  1994  2.333333
5          ABW        Aruba  1995  2.333333
6          ABW        Aruba  1996  2.333333
7          ABW        Aruba  1997  2.333333
8          ABW        Aruba  1998  2.333333
9          ABW        Aruba  1999  2.333333
```

Znajdźmy 10 państw o największej powierzchni lasów i 10 państw o najmniejszej powierzchni lasów odpowiednio w 1990 i 2020.

```
top_10_1990 =
data_forest[data_forest['year']==1990].nlargest(10,'value')
top_10_1990.head(10)
```

	country_code	country_name	year	value
6671	SUR	Suriname	1990	98.574551
2706	GNQ	Equatorial Guinea	1990	96.226381
2892	GUY	Guyana	1990	94.499111
2458	GAB	Gabon	1990	92.217255
1517	CSS	Caribbean small states	1990	91.385056
6351	SLB	Solomon Islands	1990	90.922115
341	ASM	American Samoa	1990	90.350000
3977	LBR	Liberia	1990	88.509551
5731	PNG	Papua New Guinea	1990	80.377335
2675	GNB	Guinea-Bissau	1990	79.421408

```
top_10_2020 =
data_forest[data_forest['year']==2020].nlargest(10,'value')
top_10_2020.head(10)
```

	country_code	country_name	year	value
6701	SUR	Suriname	2020	97.412115
2922	GUY	Guyana	2020	93.550114
2457	FSM	Micronesia, Fed. Sts.	2020	92.028571
2488	GAB	Gabon	2020	91.320681
6381	SLB	Solomon Islands	2020	90.138264
5730	PLW	Palau	2020	90.021739
1547	CSS	Caribbean small states	2020	89.820724
2736	GNQ	Equatorial Guinea	2020	87.287701
371	ASM	American Samoa	2020	85.650000
5761	PNG	Papua New Guinea	2020	79.176258

```
bottom_10_1990 =
data_forest[data_forest['year']==1990].nsmallest(10,'value')
bottom_10_1990.head(10)
```

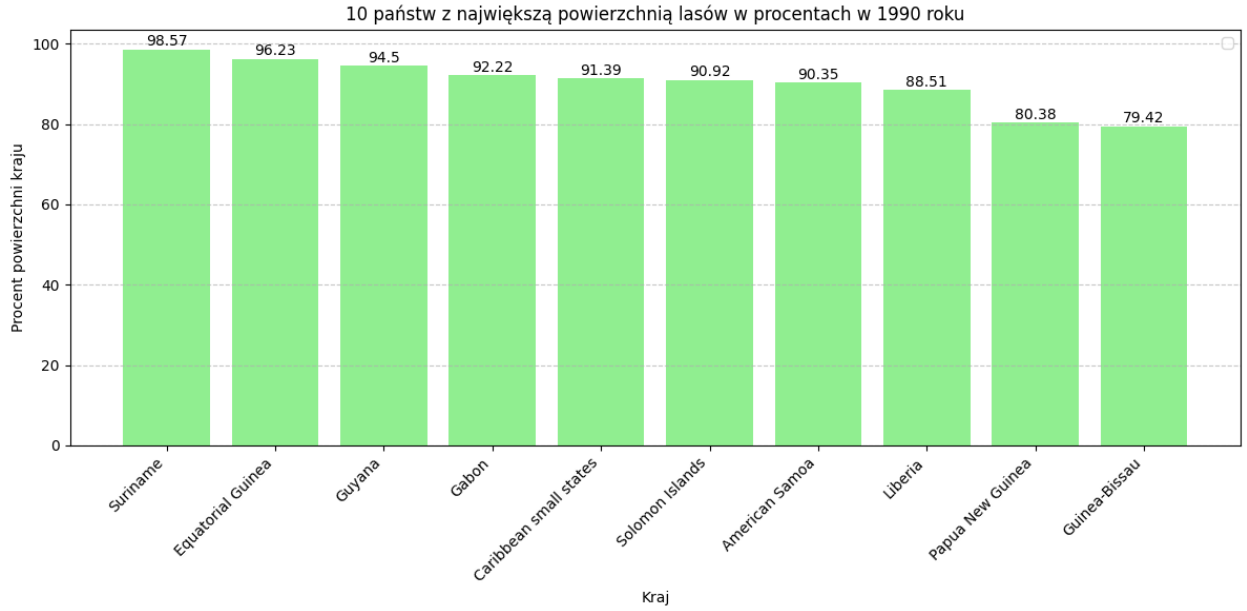
	country_code	country_name	year	value
2582	GIB	Gibraltar	1990	0.000000
4494	MCO	Monaco	1990	0.000000
5422	NRU	Nauru	1990	0.000000
6072	QAT	Qatar	1990	0.000000
2799	GRL	Greenland	1990	0.000644
5515	OMN	Oman	1990	0.009693
2056	EGY	Egypt, Arab Rep.	1990	0.044010
2397	FRO	Faroe Islands	1990	0.057307
4008	LBY	Libya	1990	0.123328
3450	ISL	Iceland	1990	0.170274

```
bottom_10_2020 =
data_forest[data_forest['year']==2020].nsmallest(10,'value')
bottom_10_2020.head(10)
```

```
     country_code      country_name  year     value
2612          GIB         Gibraltar  2020  0.000000
4524          MCO            Monaco  2020  0.000000
5452          NRU             Nauru  2020  0.000000
6102          QAT             Qatar  2020  0.000000
2829          GRL         Greenland  2020  0.000536
5545          OMN              Oman  2020  0.008078
2086          EGY  Egypt, Arab Rep.  2020  0.045186
2427          FRO     Faroe Islands  2020  0.058565
4038          LBY             Libya  2020  0.123328
1590          CUW           Curacao  2020  0.157658
```

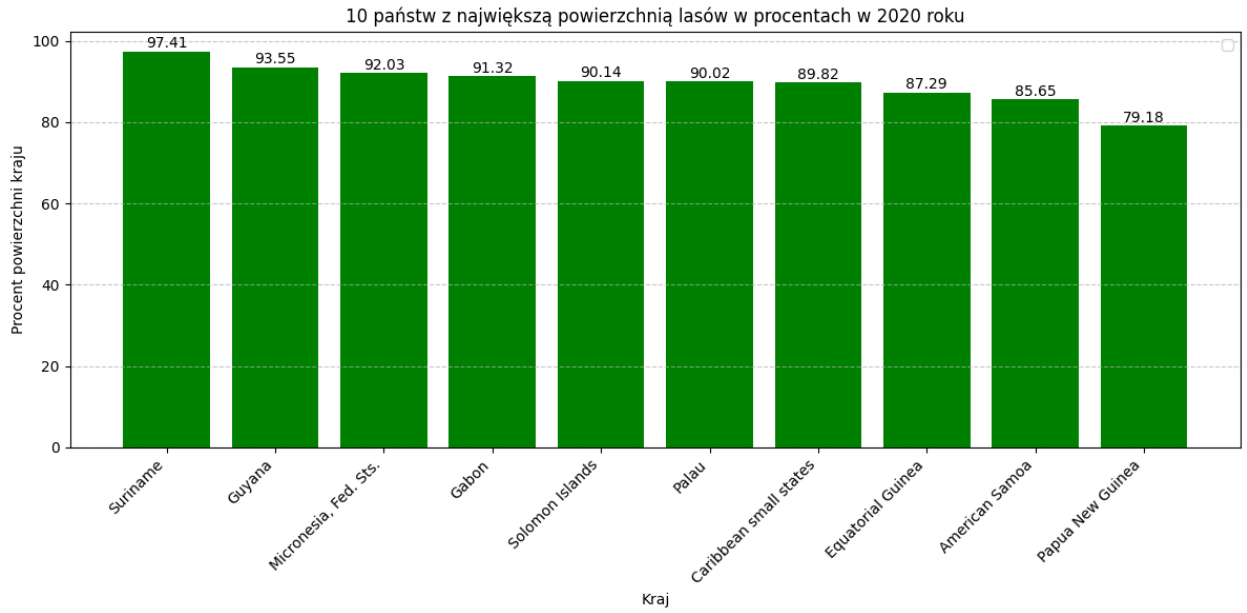
```
plt.figure(figsize=(12, 6))

bars = plt.bar(top_10_1990['country_name'], top_10_1990['value'],
color='lightgreen')
plt.title('10 państw z największą powierzchnią lasów w procentach w
1990 roku')
plt.xlabel('Kraj')
plt.ylabel('Procent powierzchni kraju')

plt.xticks(rotation=45, ha='right')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.1, round(yval,
2), ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



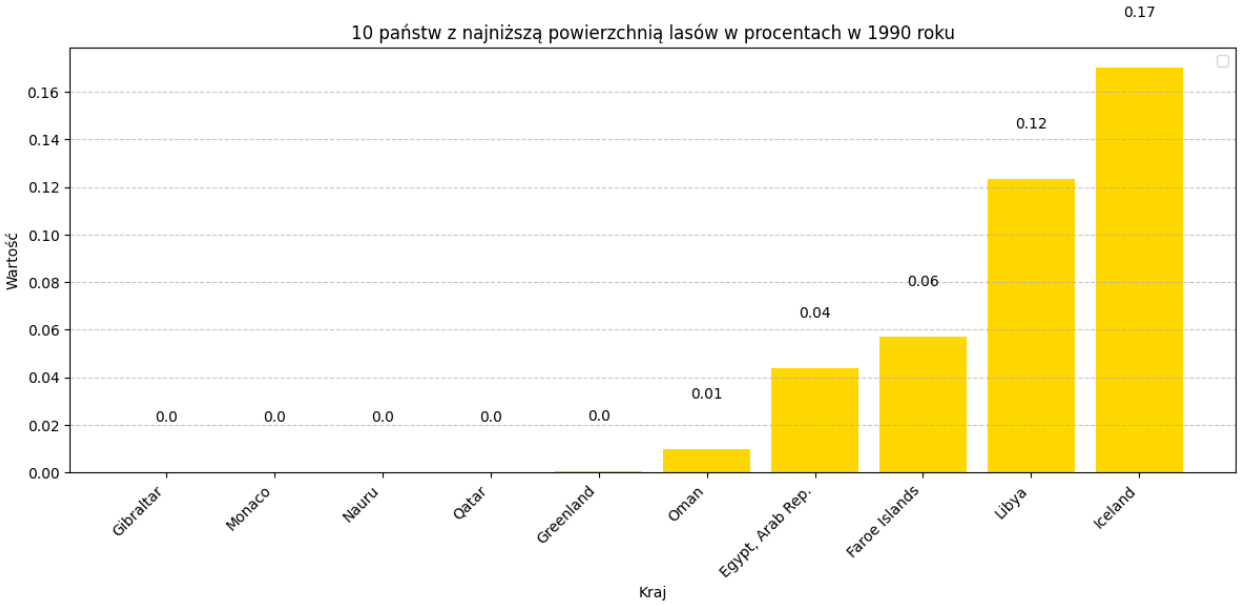

```
plt.figure(figsize=(12, 6))

bars = plt.bar(top_10_2020['country_name'], top_10_2020['value'],
color='green')
plt.title('10 państw z największą powierzchnią lasów w procentach w
2020 roku')
plt.xlabel('Kraj')
plt.ylabel('Procent powierzchni kraju')

plt.xticks(rotation=45, ha='right')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.1, round(yval,
2), ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



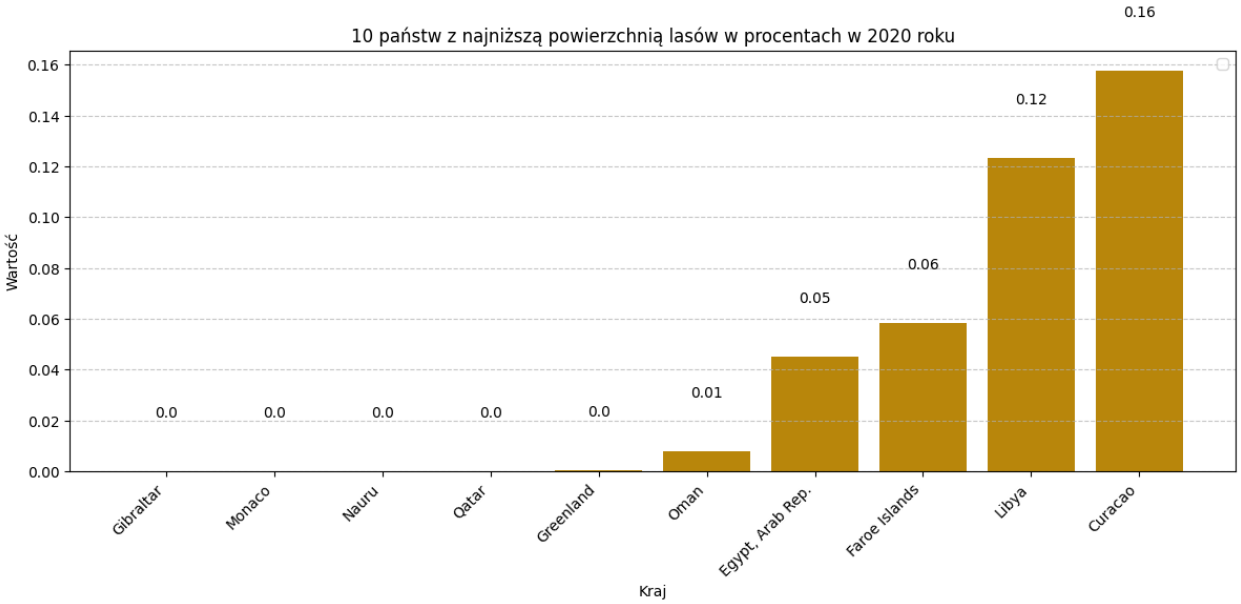

```
plt.figure(figsize=(12, 6))

bars = plt.bar(bottom_10_1990['country_name'],
bottom_10_1990['value'], color='gold')
plt.title('10 państw z najniższą powierzchnią lasów w procentach w
1990 roku')
plt.xlabel('Kraj')
plt.ylabel('Wartość')

plt.xticks(rotation=45, ha='right')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.02, round(yval,
2), ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```




```
plt.figure(figsize=(12, 6))

bars = plt.bar(bottom_10_2020['country_name'],
bottom_10_2020['value'], color='darkgoldenrod')
plt.title('10 państw z najniższą powierzchnią lasów w procentach w
2020 roku')
plt.xlabel('Kraj')
plt.ylabel('Wartość')

plt.xticks(rotation=45, ha='right')

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.02, round(yval,
2), ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



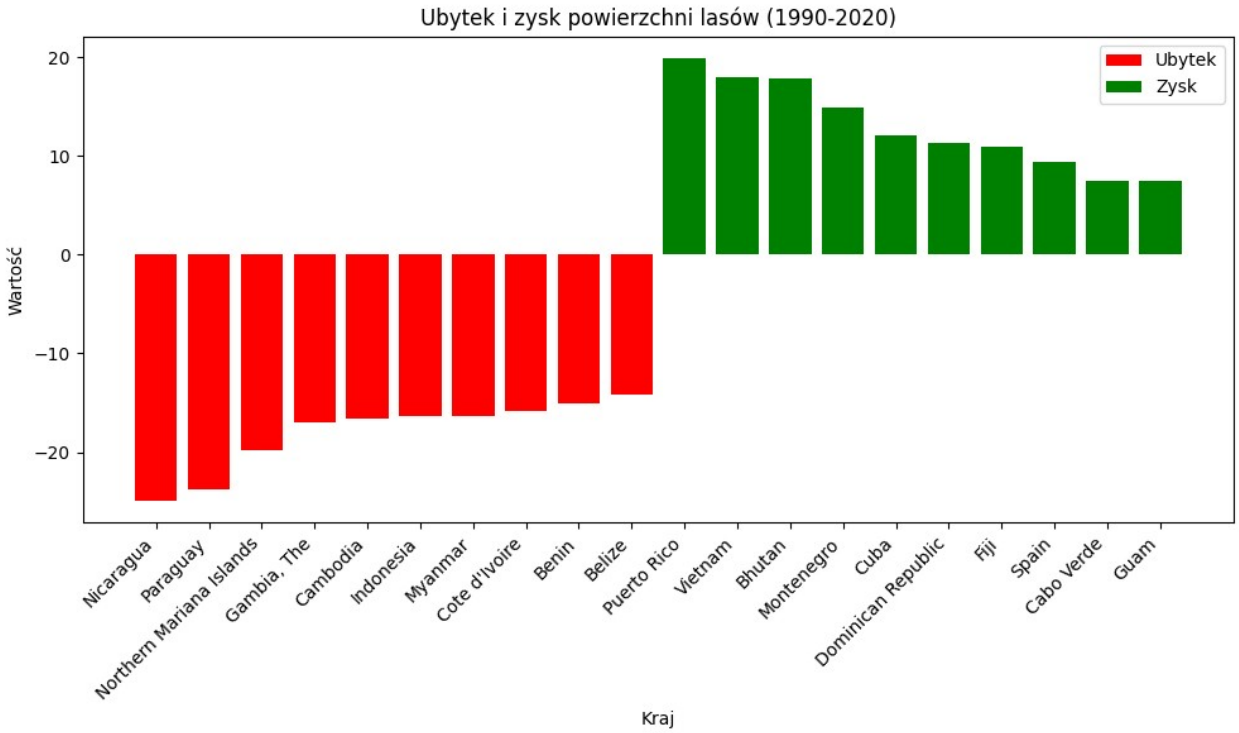
Znajdźmy 10 państw, które w latach 1990 - 2020 straciły najwięcej powierzchni lasów i 10 państw, które najwięcej zyskały.

```
grouped_data_forest = data_forest.groupby('country_name')

forest_loss = grouped_data_forest.apply(lambda x: x.iloc[0]['value'] -
x.iloc[-1]['value'])
forest_gain = grouped_data_forest.apply(lambda x: x.iloc[0]['value'] -
x.iloc[-1]['value'])

top_10_forest_loss = forest_loss.nlargest(10)
top_10_forest_gain = forest_gain.nsmallest(10)

print("10 państw z największym ubytkiem lasów w latach 1990-2020:\n")
print(top_10_forest_loss.head(10))

print('\n')

print("10 państw z największym zyskiem lasów w latach 1990-2020:\n")
print(top_10_forest_gain)
```

```
10 państw z największym ubytkiem lasów w latach 1990-2020:

country_name
Nicaragua                    24.861143
Paraguay                     23.769444
Northern Mariana Islands     19.750000
Gambia, The                  16.995059
Cambodia                     16.635056
Indonesia                    16.365934
Myanmar                      16.275272
Cote d'Ivoire                15.767767
```

```
Benin                          15.076268
Belize                         14.159579
dtype: float64


10 państw z największym zyskiem lasów w latach 1990-2020:

country_name
Puerto Rico           -19.842165
Vietnam               -17.913324
Bhutan                -17.798732
Montenegro            -14.944238
Cuba                  -12.071129
Dominican Republic    -11.368454
Fiji                  -10.964970
Spain                  -9.336804
Cabo Verde             -7.528536
Guam                   -7.407407
dtype: float64
```

```
plt.figure(figsize=(10, 6))

plt.bar(top_10_forest_loss.index, top_10_forest_loss.values*-1,
color='red', label='Ubytek')

plt.bar(top_10_forest_gain.index, top_10_forest_gain.values*-1,
color='green', label='Zysk')

plt.title('Ubytek i zysk powierzchni lasów (1990-2020)')
plt.xlabel('Kraj')
plt.ylabel('Wartość')
plt.xticks(rotation=45, ha='right')

plt.legend()

plt.tight_layout()
plt.show()
```



Procent zalesienia całego świata w roku 1990 i 2016 - porównanie

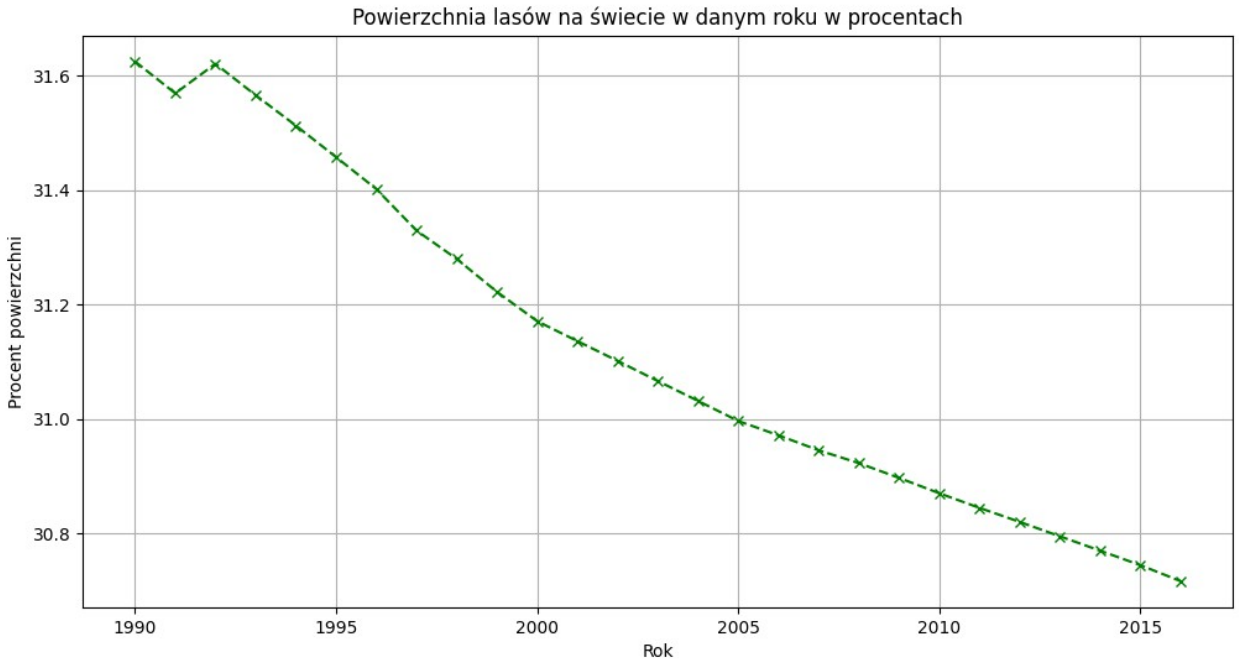
```
world_forest = data_forest[data_forest['country_name']=='World']
#world_forest.head(1)
world_forest_1990 = world_forest.nsmallest(1,'year')
world_forest_2016 = world_forest.nlargest(1,'year')
print(world_forest_1990)
print(world_forest_2016)
```

```
     country_code country_name  year      value
7861          WLD        World  1990  31.624509
     country_code country_name  year      value
7887          WLD        World  2016  30.716421
```

Procent powierzchni lądów zalesionych na przestrzeni lat 1990 - 2020 dla **całego świata**

```
world_forest_all_years = data_forest[data_forest['country_name'] ==
'World']

world_forest_all_years = world_forest_all_years.sort_values(by='year')

plt.figure(figsize=(12, 6))
plt.plot(world_forest_all_years['year'],
world_forest_all_years['value'], marker='x', linestyle='--',
color='green')
plt.xlabel('Rok')
```

```
plt.ylabel('Procent powierzchni')
plt.title('Powierzchnia lasów na świecie w danym roku w procentach')
plt.grid(True)
plt.show()
```



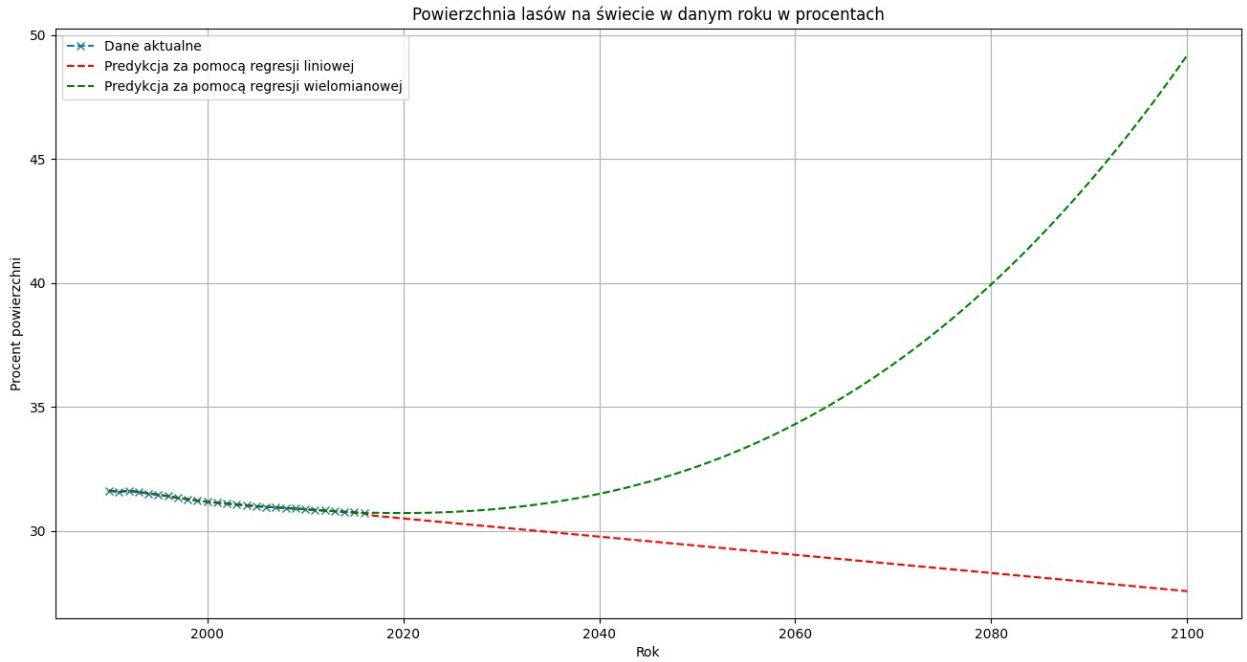
Prognozy uzyskane za pomocą **regresji liniowej**

```
years_after_2016 = np.arange(2017, 2101).reshape(-1, 1)
linear_regression = LinearRegression()
linear_regression.fit(world_forest_all_years[['year']],
world_forest_all_years['value'])
predicted_forest_coverages_linear =
linear_regression.predict(years_after_2016)

# Regresja wielomianowa
poly_features = PolynomialFeatures(degree=3)  # Można dostosować
stopień wielomianu
X_poly = poly_features.fit_transform(world_forest_all_years[['year']])
poly_regression = LinearRegression()
poly_regression.fit(X_poly, world_forest_all_years['value'])

# Prognozy dla regresji wielomianowej
years_after_2016_poly = poly_features.transform(years_after_2016)
predicted_forest_coverages_poly =
poly_regression.predict(years_after_2016_poly)

# Łączenie aktualnych danych z prognozami
extended_years =
np.concatenate((world_forest_all_years['year'].values,
```

```
years_after_2016.flatten()))
extended_forest_coverages_linear =
np.concatenate((world_forest_all_years['value'].values,
predicted_forest_coverages_linear))
extended_forest_coverages_poly =
np.concatenate((world_forest_all_years['value'].values,
predicted_forest_coverages_poly))

# Prognozy na wybrane lata
years_to_predict = [2024, 2050, 2075, 2100]
predicted_coverages_linear =
linear_regression.predict(np.array(years_to_predict).reshape(-1, 1))
predicted_coverages_poly =
poly_regression.predict(poly_features.transform(np.array(years_to_pred
ict).reshape(-1, 1)))

predictions_df = pd.DataFrame({
    'Rok': years_to_predict,
    'Powierzchnia lasów (%) - Liniowa': predicted_coverages_linear,
    'Powierzchnia lasów (%) - Wielomianowa': predicted_coverages_poly
})
print(predictions_df)

# Wizualizacja
plt.figure(figsize=(16, 8))
plt.plot(world_forest_all_years['year'],
world_forest_all_years['value'], marker='x', linestyle='--',
label='Dane aktualne')
plt.plot(extended_years, extended_forest_coverages_linear,
linestyle='--', color='red', label='Predykcja za pomocą regresji
liniowej')
plt.plot(extended_years, extended_forest_coverages_poly,
linestyle='--', color='green', label='Predykcja za pomocą regresji
wielomianowej')
plt.xlabel('Rok')
plt.ylabel('Procent powierzchni')
plt.title('Powierzchnia lasów na świecie w danym roku w procentach')
plt.grid(True)
plt.legend()
plt.show()
```

```
    Rok  Powierzchnia lasów (%) - Liniowa  \
0  2024                          30.355088
1  2050                          29.401759
2  2075                          28.485096
3  2100                          27.568433

   Powierzchnia lasów (%) - Wielomianowa
0                              30.749194
1                              32.596102
```

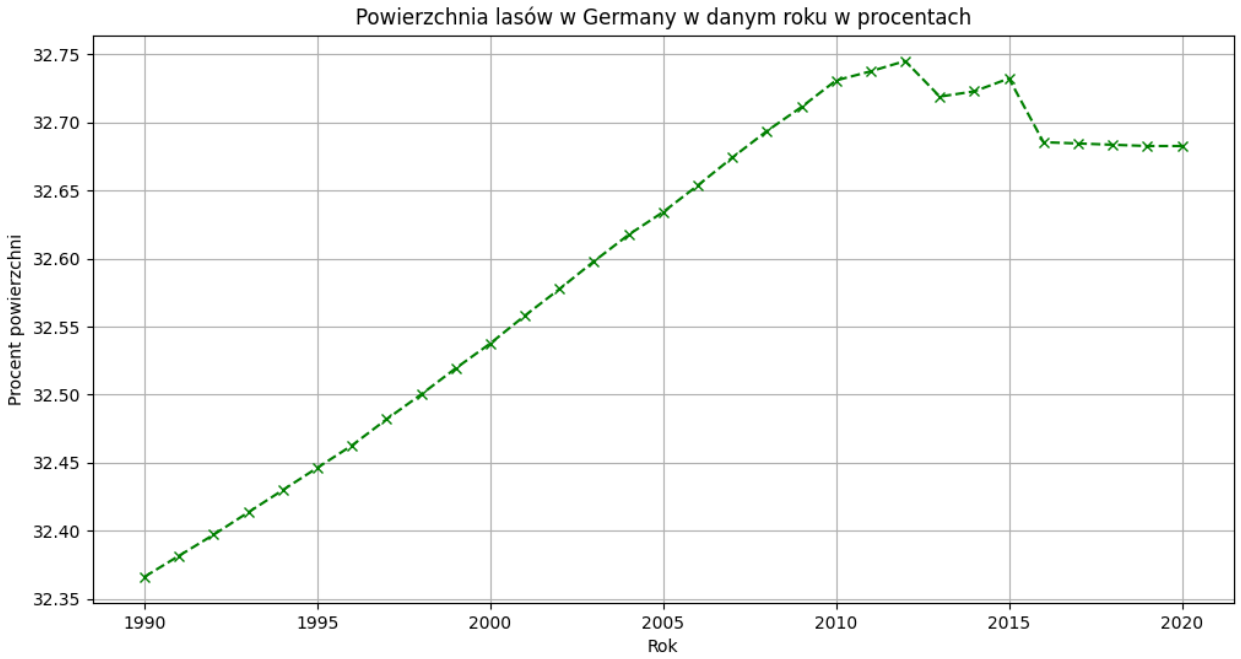
```
2                                   38.225217
3                                   49.182666

C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
```



Powierzchnia lasów dla **wybranego państwa** w procentach i jej prognozy do roku 2100 za pomocą **regresjii liniowej**.

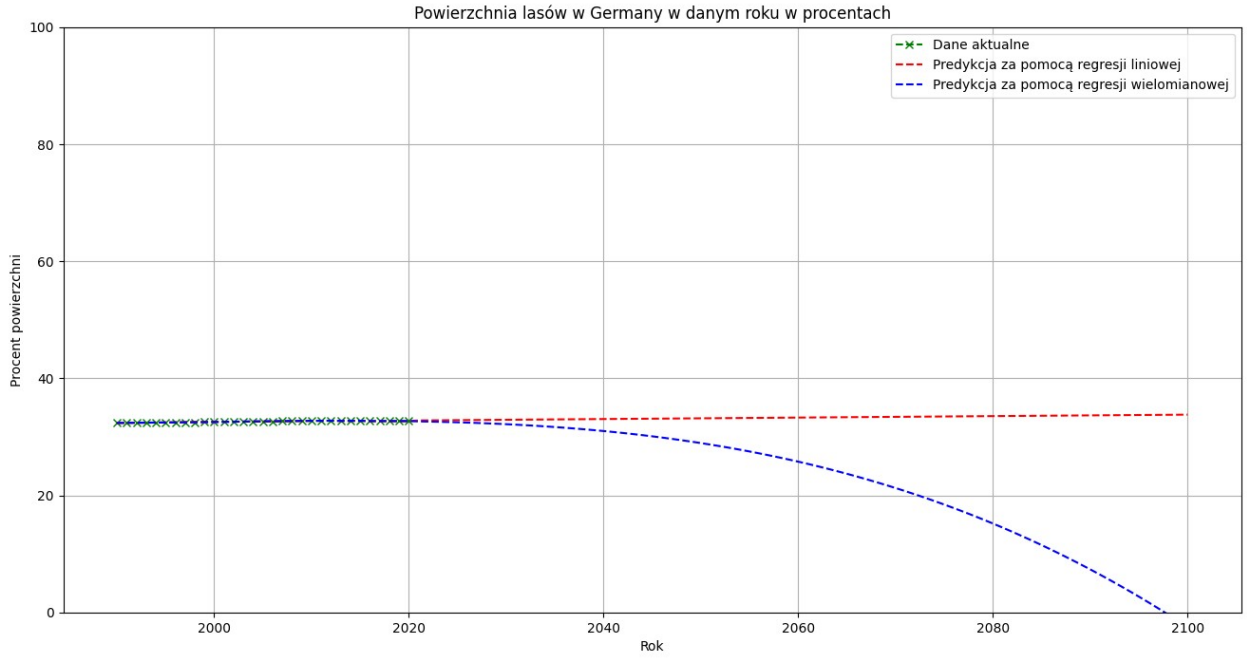
```
try:
    country_name = input("Podaj nazwę państwa (angielska nazwa, z
wielkiej litery):")
    country_forest_all_years = data_forest[data_forest['country_name']
== country_name]

    country_forest_all_years =
country_forest_all_years.sort_values(by='year')

    plt.figure(figsize=(12, 6))
    plt.plot(country_forest_all_years['year'],
country_forest_all_years['value'], marker='x', linestyle='--',
color='green')
    plt.xlabel('Rok')
    plt.ylabel('Procent powierzchni')
    plt.title('Powierzchnia lasów w ' + country_name + ' w danym roku
w procentach')
    plt.grid(True)
    plt.show()

    years_after_2020 = np.arange(2021, 2101).reshape(-1, 1)
    linear_regression = LinearRegression()
    linear_regression.fit(country_forest_all_years[['year']],
country_forest_all_years['value'])
    predicted_forest_coverages_linear =
linear_regression.predict(years_after_2020)

    poly_features = PolynomialFeatures(degree=3)  # Możesz dostosować
stopień wielomianu
    X_poly =
poly_features.fit_transform(country_forest_all_years[['year']])
    poly_regression = LinearRegression()
    poly_regression.fit(X_poly, country_forest_all_years['value'])

    years_after_2020_poly = poly_features.transform(years_after_2020)
    predicted_forest_coverages_poly =
poly_regression.predict(years_after_2020_poly)

    extended_years =
np.concatenate((country_forest_all_years['year'].values,
years_after_2020.flatten()))
    extended_forest_coverages_linear =
np.concatenate((country_forest_all_years['value'].values,
predicted_forest_coverages_linear))
    extended_forest_coverages_poly =
np.concatenate((country_forest_all_years['value'].values,
predicted_forest_coverages_poly))
```

```
    years_to_predict = [2024, 2050, 2075, 2100]
    predicted_coverages_linear =
linear_regression.predict(np.array(years_to_predict).reshape(-1, 1))
    predicted_coverages_poly =
poly_regression.predict(poly_features.transform(np.array(years_to_pred
ict).reshape(-1, 1)))

    predicted_coverages_linear = np.where(predicted_coverages_linear <
0, 0, predicted_coverages_linear)
    predicted_coverages_linear = np.where(predicted_coverages_linear >
100, 100, predicted_coverages_linear)

    predicted_coverages_poly = np.where(predicted_coverages_poly < 0,
0, predicted_coverages_poly)
    predicted_coverages_poly = np.where(predicted_coverages_poly >
100, 100, predicted_coverages_poly)

    predictions_df = pd.DataFrame({
        'Rok': years_to_predict,
        'Powierzchnia lasów (%) - Liniowa':
predicted_coverages_linear,
        'Powierzchnia lasów (%) - Wielomianowa':
predicted_coverages_poly
    })
    print(predictions_df)

    plt.figure(figsize=(16, 8))
    plt.plot(country_forest_all_years['year'],
country_forest_all_years['value'], marker='x', linestyle='--',
color='green', label='Dane aktualne')
    plt.plot(extended_years, extended_forest_coverages_linear,
linestyle='--', color='red', label='Predykcja za pomocą regresji
liniowej')
    plt.plot(extended_years, extended_forest_coverages_poly,
linestyle='--', color='blue', label='Predykcja za pomocą regresji
wielomianowej')
    plt.xlabel('Rok')
    plt.ylabel('Procent powierzchni')
    plt.ylim(0, 100)
    plt.title('Powierzchnia lasów w ' + country_name + ' w danym roku
w procentach')
    plt.grid(True)
    plt.legend()
    plt.show()

except Exception as e:
    print("Wystąpił błąd. Upewnij się, że nazwa państwa (duża litera i
angielska nazwa) jest poprawna")
```




```
    Rok  Powierzchnia lasów (%) - Liniowa  \
0  2024                          32.832946
1  2050                          33.157846
2  2075                          33.470250
3  2100                          33.782654

   Powierzchnia lasów (%) - Wielomianowa
0                              32.520937
1                              28.923570
2                              18.422025
3                               0.000000

C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
```

```
  warnings.warn(
C:\Users\kubar\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-
packages\Python311\site-packages\sklearn\base.py:465: UserWarning: X
does not have valid feature names, but PolynomialFeatures was fitted
with feature names
  warnings.warn(
```



// ZESTAWIĆ CO2 Z LASAMI I KORELACJA

5. Katastrofy naturalne w latach 1900 - 2021

Analizowany w tym podpunkcie plik CSV zawiera dane dotyczące klęsk żywiołowych z różnych regionów i krajów w latach 1900-2021. Oto opis poszczególnych kolumn:

- Year: Rok, w którym wystąpiła klęska.
- Seq: Numer sekwencyjny.
- Glide: Kod zdarzenia z Global Disaster Identifier (Glide).
- Disaster Group: Grupa klęski, tutaj głównie naturalne.
- Disaster Subgroup: Podgrupa klęski, np. klimatyczne, geofizyczne.
- Disaster Type: Rodzaj klęski, np. susza, trzęsienie ziemi.
- Disaster Subtype: Podtyp klęski, np. powódź rzeczna, pożar lasu.
- Disaster Subsubtype: Podpodtyp klęski, szczegółowa kategoria klęski.

- Event Name: Nazwa zdarzenia klęski.
- Country: Kraj, w którym wystąpiła klęska.
- ISO: Kod ISO kraju.
- Region: Region geograficzny, do którego należy kraj.
- Continent: Kontynent, na którym znajduje się kraj.
- Location: Szczegółowa lokalizacja klęski.
- Origin: Pochodzenie klęski.
- Associated Dis: Powiązane klęski.
- Associated Dis2: Druga powiązana klęska.
- OFDA Response: Odpowiedź Urzędu ds. Pomocy Zagranicznej (OFDA).
- Appeal: Apel o pomoc.
- Declaration: Deklaracja o klęsce.
- Aid Contribution: Wkład w pomoc.
- Dis Mag Value: Wartość magnitudy klęski.
- Dis Mag Scale: Skala magnitudy klęski.
- Latitude: Szerokość geograficzna.
- Longitude: Długość geograficzna.
- Local Time: Lokalny czas.
- River Basin: Obszar dorzecza rzeki.
- Start Year: Rok rozpoczęcia klęski.
- Start Month: Miesiąc rozpoczęcia klęski.
- Start Day: Dzień rozpoczęcia klęski.
- End Year: Rok zakończenia klęski.
- End Month: Miesiąc zakończenia klęski.
- End Day: Dzień zakończenia klęski.
- Total Deaths: Całkowita liczba ofiar śmiertelnych.
- No Injured: Liczba rannych.

- No Affected: Liczba dotkniętych klęską.
- No Homeless: Liczba bezdomnych.
- Total Affected: Całkowicie dotknięci klęską.
- Insured Damages ('000 US$): Szacowane ubezpieczone szkody w tysiącach dolarów amerykańskich.
- Total Damages ('000 US$): Całkowite szkody w tysiącach dolarów amerykańskich.
- CPI: Indeks cen towarów i usług.
- Adm Level: Poziom administracyjny.
- Admin1 Code: Kod administracyjny regionu/prowincji.
- Admin2 Code: Kod administracyjny podregionu.
- Geo Locations: Lokalizacje geograficzne.

Ten zbiór danych zawiera informacje o różnych rodzajach klęsk żywiołowych, takich jak susze, trzęsienia ziemi, powodzie czy erupcje wulkanów, wraz z danymi geograficznymi i skutkami tych klęsk. Może być wykorzystywany do analizy trendów w występowaniu klęsk naturalnych oraz ich wpływu na społeczeństwo i gospodarkę.

Wczytanie danych:

```
disasters_data = pd.read_csv("dane/1900_2021_DISASTERS.xlsx - emdat
data.csv")
disasters_data.head(10)

#data_check = disasters_data.loc[disasters_data['Disaster
Group']=='Natural']
#data_check.head(100)

#unikalne_wartosci = disasters_data['Origin'].unique()
#print(unikalne_wartosci)
```

```
   Year   Seq Glide Disaster Group Disaster Subgroup        Disaster
Type  \
0  1900  9002   NaN        Natural    Climatological
Drought
1  1900  9001   NaN        Natural    Climatological
Drought
2  1902    12   NaN        Natural       Geophysical
Earthquake
3  1902     3   NaN        Natural       Geophysical    Volcanic
activity
4  1902    10   NaN        Natural       Geophysical    Volcanic
activity
```

```
5  1903     6   NaN        Natural       Geophysical  Mass movement
(dry)
6  1903    12   NaN        Natural       Geophysical    Volcanic
activity
7  1904     3   NaN        Natural    Meteorological
Storm
8  1905     5   NaN        Natural       Geophysical  Mass movement
(dry)
9  1905     3   NaN        Natural       Geophysical
Earthquake

   Disaster Subtype Disaster Subsubtype      Event Name        Country
...  \
0           Drought                 NaN             NaN     Cabo Verde
...
1           Drought                 NaN             NaN          India
...
2   Ground movement                 NaN             NaN      Guatemala
...
3          Ash fall                 NaN     Santa Maria      Guatemala
...
4          Ash fall                 NaN     Santa Maria      Guatemala
...
5          Rockfall                 NaN             NaN         Canada
...
6          Ash fall                 NaN  Mount Karthala  Comoros (the)
...
7  Tropical cyclone                 NaN             NaN     Bangladesh
...
8          Rockfall                 NaN             NaN         Canada
...
9   Ground movement                 NaN             NaN          India
...

  No Affected No Homeless Total Affected Insured Damages ('000 US$)  \
0         NaN         NaN            NaN                         NaN
1         NaN         NaN            NaN                         NaN
2         NaN         NaN            NaN                         NaN
3         NaN         NaN            NaN                         NaN
4         NaN         NaN            NaN                         NaN
5         NaN         NaN           23.0                         NaN
6         NaN         NaN            NaN                         NaN
7         NaN         NaN            NaN                         NaN
8         NaN         NaN           18.0                         NaN
9         NaN         NaN            NaN                         NaN

  Total Damages ('000 US$)       CPI Adm Level Admin1 Code Admin2 Code
\
0                      NaN  3.221647       NaN         NaN         NaN
```

```
1                      NaN  3.221647       NaN          NaN          NaN

2                  25000.0  3.350513       NaN          NaN          NaN

3                      NaN  3.350513       NaN          NaN          NaN

4                      NaN  3.350513       NaN          NaN          NaN

5                      NaN  3.479379       NaN          NaN          NaN

6                      NaN  3.479379       NaN          NaN          NaN

7                      NaN  3.479379       NaN          NaN          NaN

8                      NaN  3.479379       NaN          NaN          NaN

9                  25000.0  3.479379       NaN          NaN          NaN


  Geo Locations
0           NaN
1           NaN
2           NaN
3           NaN
4           NaN
5           NaN
6           NaN
7           NaN
8           NaN
9           NaN

[10 rows x 45 columns]
```

Do analizy danych dokonam selekcji odpowiednich kolumn: rok, rodzaj klęski, podtyp klęski, podtyp podtypu klęski, nazwa zdarzenia, kraj, kontynent, pochodzenie, liczba ofiar śmiertelnych, liczba osób dotkniętych klęską, szkody w tysiącach dolarów amerykańskich.

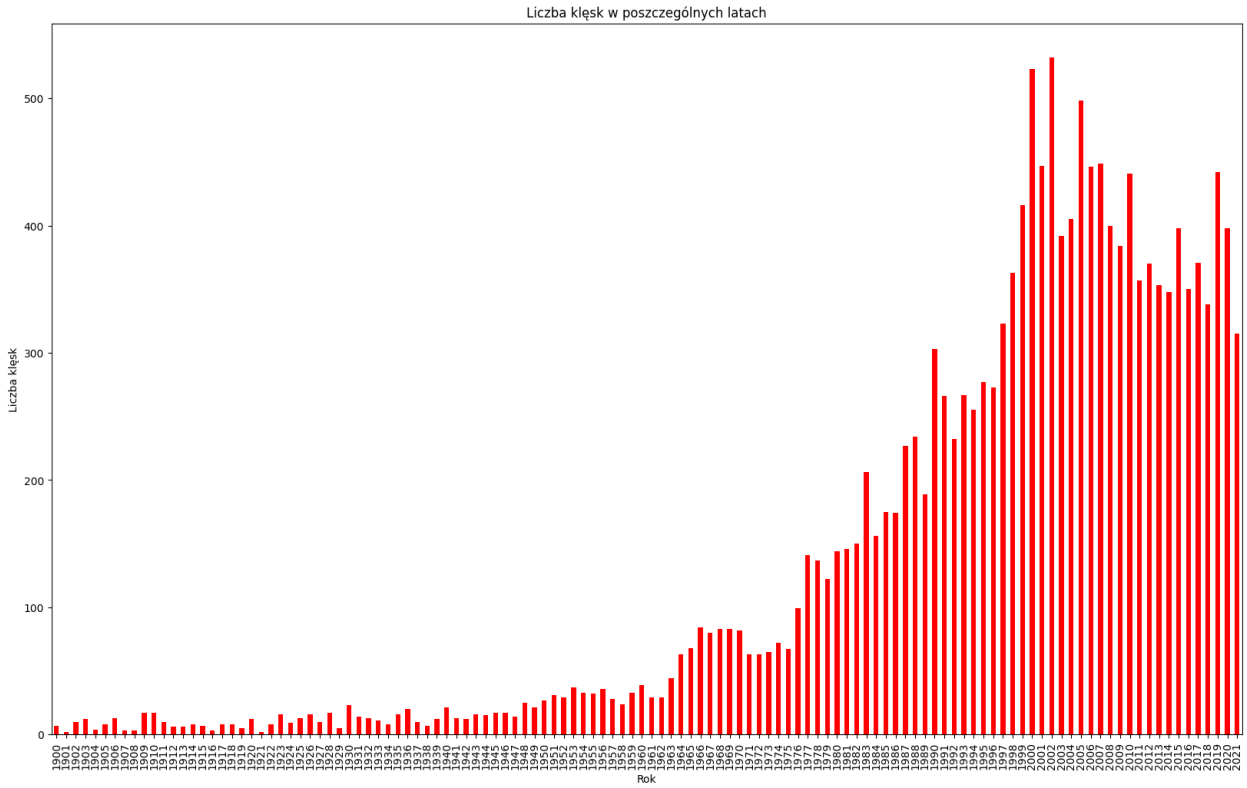
```
selected_disaster_data = disasters_data[['Year', 'Disaster Type',
'Disaster Subtype', 'Disaster Subsubtype',
                  'Event Name', 'Country', 'Continent', 'Origin',
'Total Deaths',
                  'Total Affected', 'Total Damages (\'000 US$)']]

selected_disaster_data.head(10)
```

```
   Year        Disaster Type  Disaster Subtype Disaster Subsubtype  \
0  1900              Drought           Drought                 NaN
1  1900              Drought           Drought                 NaN
2  1902           Earthquake   Ground movement                 NaN
3  1902    Volcanic activity          Ash fall                 NaN
```

```
4  1902    Volcanic activity          Ash fall                 NaN
5  1903  Mass movement (dry)          Rockfall                 NaN
6  1903    Volcanic activity          Ash fall                 NaN
7  1904                Storm  Tropical cyclone                 NaN
8  1905  Mass movement (dry)          Rockfall                 NaN
9  1905           Earthquake   Ground movement                 NaN

       Event Name        Country Continent Origin  Total Deaths  \
0             NaN     Cabo Verde    Africa    NaN       11000.0
1             NaN          India      Asia    NaN     1250000.0
2             NaN      Guatemala  Americas    NaN        2000.0
3     Santa Maria      Guatemala  Americas    NaN        1000.0
4     Santa Maria      Guatemala  Americas    NaN        6000.0
5             NaN         Canada  Americas    NaN          76.0
6  Mount Karthala  Comoros (the)    Africa    NaN          17.0
7             NaN     Bangladesh      Asia    NaN           NaN
8             NaN         Canada  Americas    NaN          18.0
9             NaN          India      Asia    NaN       20000.0

   Total Affected  Total Damages ('000 US$)
0             NaN                       NaN
1             NaN                       NaN
2             NaN                   25000.0
3             NaN                       NaN
4             NaN                       NaN
5            23.0                       NaN
6             NaN                       NaN
7             NaN                       NaN
8            18.0                       NaN
9             NaN                   25000.0
```

Pogrupujmy poszczególne klęski wg. roku, aby utworzyć wykres słupkowy, z którego będzie można odczytać liczbę klęsk w poszczególnych latach.

```
disasters_years = selected_disaster_data.groupby('Year').size()

disasters_years.plot(kind='bar', figsize=(20, 12), color='RED')

plt.title('Liczba klęsk w poszczególnych latach')
plt.xlabel('Rok')
plt.ylabel('Liczba klęsk')

plt.show()
```



Sprawdźmy 10 największych klęsk wg. ofiar śmiertelnych i według strat wyrażonych w dolarach amerykańskich.

```
top_10_disaster_death = selected_disaster_data.nlargest(10, 'Total
Deaths')

top_loss = selected_disaster_data.nlargest(10, 'Total Damages (\'000
US$)')

print("Top 10 klęsk według liczby ofiar śmiertelnych:")


display(top_10_disaster_death[['Year', 'Disaster Type',
                          'Country', 'Continent',  'Total Deaths',
                              'Total Affected', 'Total Damages (\'000
US$)']])

print("\nTop 10 klęsk według strat w dolarach amerykańskich:")
display(top_loss[['Year', 'Disaster Type',
                  'Country', 'Continent',  'Total Deaths',
                 'Total Affected', 'Total Damages (\'000 US$)']])
```

```
Top 10 klęsk według liczby ofiar śmiertelnych:
```

```
      Year Disaster Type       Country Continent  Total Deaths  \
111   1931         Flood         China      Asia     3700000.0
```

```
97    1928       Drought         China      Asia     3000000.0
902   1917      Epidemic  Soviet Union    Europe     2500000.0
58    1920      Epidemic         India      Asia     2000000.0
276   1959         Flood         China      Asia     2000000.0
152   1943       Drought    Bangladesh      Asia     1900000.0
20    1909      Epidemic         China      Asia     1500000.0
145   1942       Drought         India      Asia     1500000.0
1355  1965       Drought         India      Asia     1500000.0
16    1907      Epidemic         India      Asia     1300000.0

      Total Affected  Total Damages ('000 US$)
111              NaN                1400000.0
97               NaN                      NaN
902              NaN                      NaN
58               NaN                      NaN
276              NaN                      NaN
152              NaN                      NaN
20               NaN                      NaN
145              NaN                      NaN
1355     100000000.0                 100000.0
16               NaN                      NaN


Top 10 klęsk według strat w dolarach amerykańskich:

       Year Disaster Type                          Country Continent  \
11933  2011    Earthquake                            Japan      Asia
9844   2005         Storm  United States of America (the)  Americas
5530   1995    Earthquake                            Japan      Asia
14871  2017         Storm  United States of America (the)  Americas
10490  2008    Earthquake                            China      Asia
14353  2017         Storm                      Puerto Rico  Americas
14894  2017         Storm  United States of America (the)  Americas
12923  2012         Storm  United States of America (the)  Americas
11943  2011         Flood                         Thailand      Asia
5102   1994    Earthquake  United States of America (the)  Americas

       Total Deaths  Total Affected  Total Damages ('000 US$)
11933       19846.0        368820.0               210000000.0
9844         1833.0        500000.0               125000000.0
5530         5297.0        541636.0               100000000.0
14871          88.0        582024.0                95000000.0
10490       87476.0      45976596.0                85000000.0
14353          64.0        750000.0                68000000.0
14894          97.0         70000.0                57000000.0
12923          54.0             NaN                50000000.0
11943         813.0       9500000.0                40000000.0
5102           60.0         27000.0                30000000.0
```

6. Zbiór danych klimatycznych w Jena

```
jena_dataset = pd.read_csv("dane/jena_climate_2009_2016.csv")
jena_dataset.head()
```

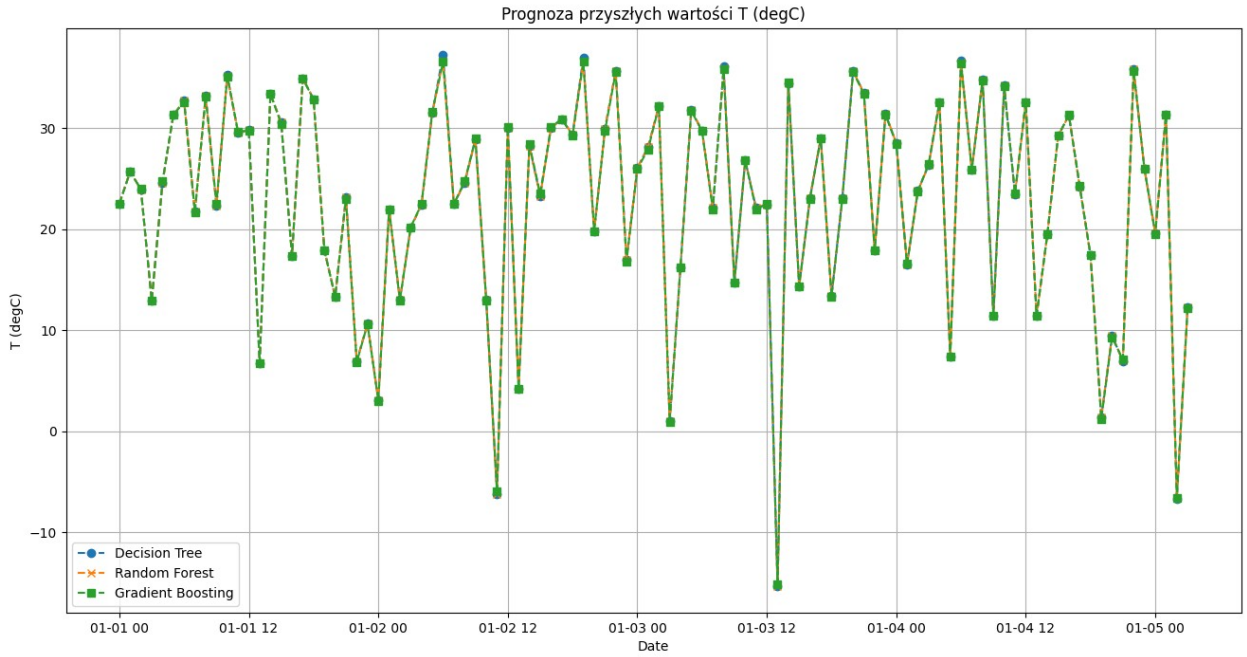
```
             Date Time  p (mbar)  T (degC)  Tpot (K)  Tdew (degC)  rh
(%)  \
0  01.01.2009 00:10:00    996.52     -8.02    265.40        -8.90
93.3
1  01.01.2009 00:20:00    996.57     -8.41    265.01        -9.28
93.4
2  01.01.2009 00:30:00    996.53     -8.51    264.91        -9.31
93.9
3  01.01.2009 00:40:00    996.51     -8.31    265.12        -9.07
94.2
4  01.01.2009 00:50:00    996.51     -8.27    265.15        -9.04
94.1

   VPmax (mbar)  VPact (mbar)  VPdef (mbar)  sh (g/kg)  H2OC
(mmol/mol)  \
0          3.33          3.11          0.22       1.94
3.12
1          3.23          3.02          0.21       1.89
3.03
2          3.21          3.01          0.20       1.88
3.02
3          3.26          3.07          0.19       1.92
3.08
4          3.27          3.08          0.19       1.92
3.09

   rho (g/m**3)  wv (m/s)  max. wv (m/s)  wd (deg)
0       1307.75      1.03           1.75     152.3
1       1309.80      0.72           1.50     136.1
2       1310.24      0.19           0.63     171.6
3       1309.19      0.34           0.50     198.0
4       1309.00      0.32           0.63     214.3
```

```
# Konwersja kolumny 'Date Time' na typ datetime
jena_dataset['Date Time'] = pd.to_datetime(jena_dataset['Date Time'],
errors='coerce')

# Usunięcie wierszy z niepoprawnymi datami
jena_dataset = jena_dataset.dropna(subset=['Date Time'])

# Pobranie listy dostępnych kolumn
columns = list(jena_dataset.columns)
columns.remove('Date Time')

# Wyświetlenie dostępnych kolumn i pobranie wyboru użytkownika
```

```
print("Dostępne kolumny: ", columns)
target_variable = input(f"Podaj nazwę zmiennej docelowej ({columns}):
")

# Sprawdzenie, czy wybrana zmienna jest dostępna
if target_variable not in columns:
    raise ValueError(f"Zmienna {target_variable} nie jest dostępna.
Wybierz jedną z dostępnych kolumn.")

# Ustawienie cech (features) jako wszystkich pozostałych kolumn oprócz
wybranej zmiennej docelowej
features = [col for col in columns if col != target_variable]

# Podział danych na cechy (X) i zmienną docelową (y)
X = jena_dataset[features]
y = jena_dataset[target_variable]

# Podział danych na zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Trenowanie modeli
tree_reg = DecisionTreeRegressor()
forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)
gboost_reg = GradientBoostingRegressor(n_estimators=100,
random_state=42)

tree_reg.fit(X_train, y_train)
forest_reg.fit(X_train, y_train)
gboost_reg.fit(X_train, y_train)

# Prognozowanie na zbiorze testowym
y_pred_tree = tree_reg.predict(X_test)
y_pred_forest = forest_reg.predict(X_test)
y_pred_gboost = gboost_reg.predict(X_test)

# Obliczanie błędów
print('Decision Tree MSE:', mean_squared_error(y_test, y_pred_tree))
print('Random Forest MSE:', mean_squared_error(y_test, y_pred_forest))
print('Gradient Boosting MSE:', mean_squared_error(y_test,
y_pred_gboost))

print('Decision Tree MAE:', mean_absolute_error(y_test, y_pred_tree))
print('Random Forest MAE:', mean_absolute_error(y_test,
y_pred_forest))
print('Gradient Boosting MAE:', mean_absolute_error(y_test,
y_pred_gboost))

print('Decision Tree R2:', r2_score(y_test, y_pred_tree))
print('Random Forest R2:', r2_score(y_test, y_pred_forest))
```

```
print('Gradient Boosting R2:', r2_score(y_test, y_pred_gboost))

# Prognozowanie przyszłych wartości
future_dates = pd.date_range(start='2017-01-01', periods=100,
freq='H')
future_data = pd.DataFrame({
    feature: np.random.uniform(X[feature].min(), X[feature].max(),
size=len(future_dates))
    for feature in features
})

future_predictions_tree = tree_reg.predict(future_data)
future_predictions_forest = forest_reg.predict(future_data)
future_predictions_gboost = gboost_reg.predict(future_data)

# Wizualizacja wyników
plt.figure(figsize=(16, 8))
plt.plot(future_dates, future_predictions_tree, label='Decision Tree',
linestyle='--', marker='o')
plt.plot(future_dates, future_predictions_forest, label='Random
Forest', linestyle='--', marker='x')
plt.plot(future_dates, future_predictions_gboost, label='Gradient
Boosting', linestyle='--', marker='s')
plt.xlabel('Date')
plt.ylabel(target_variable)
plt.title(f'Prognoza przyszłych wartości {target_variable}')
plt.legend()
plt.grid(True)
plt.show()
```

```
Dostępne kolumny:  ['p (mbar)', 'T (degC)', 'Tpot (K)', 'Tdew (degC)',
'rh (%)', 'VPmax (mbar)', 'VPact (mbar)', 'VPdef (mbar)', 'sh (g/kg)',
'H2OC (mmol/mol)', 'rho (g/m**3)', 'wv (m/s)', 'max. wv (m/s)', 'wd
(deg)']
Decision Tree MSE: 4.957021079288081e-05
Random Forest MSE: 2.8346462293873304e-05
Gradient Boosting MSE: 0.006845633637664787
Decision Tree MAE: 0.003293029449187653
Random Forest MAE: 0.002901947426625253
Gradient Boosting MAE: 0.06255295771641649
Decision Tree R2: 0.9999992993104894
Random Forest R2: 0.9999995993144174
Gradient Boosting R2: 0.9999032349548841
```



7. Zbiór danych dotyczący zmian klimatu

```
climate_change_data = pd.read_csv("dane/climate_change_data.csv")
climate_change_data.head()
```

```
                             Date          Location        Country  \
0  2000-01-01 00:00:00.000000000   New Williamtown         Latvia
1  2000-01-01 20:09:43.258325832      North Rachel   South Africa
2  2000-01-02 16:19:26.516651665  West Williamland  French Guiana
3  2000-01-03 12:29:09.774977497       South David        Vietnam
4  2000-01-04 08:38:53.033303330    New Scottburgh        Moldova

   Temperature  CO2 Emissions  Sea Level Rise  Precipitation
Humidity  \
0    10.688986     403.118903        0.717506      13.835237
23.631256
1    13.814430     396.663499        1.205715      40.974084
43.982946
2    27.323718     451.553155       -0.160783      42.697931
96.652600
3    12.309581     422.404983       -0.475931       5.193341
47.467938
4    13.210885     410.472999        1.135757      78.695280
61.789672

   Wind Speed
0   18.492026
1   34.249300
2   34.124261
```

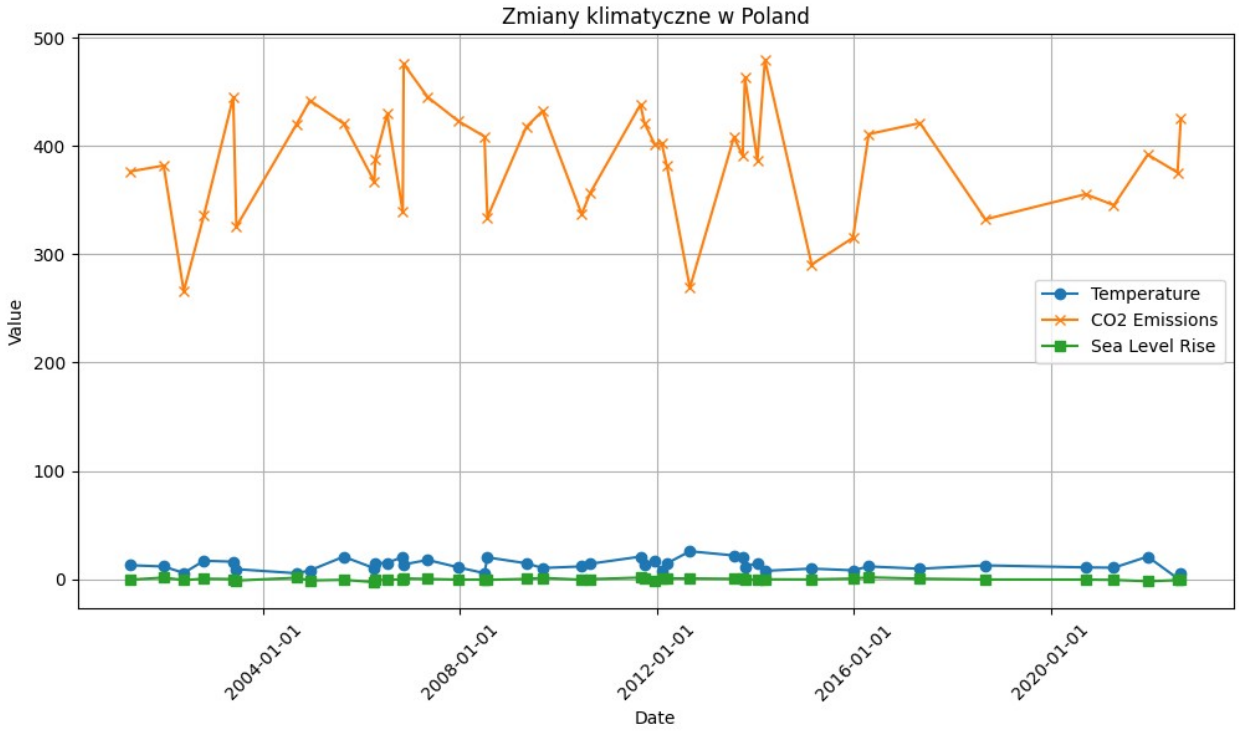
```
3    8.554563
4    8.001164
```

Pogrupujmy dane wg. Państwa

```
climate_change_data['Date'] =
pd.to_datetime(climate_change_data['Date'], errors='coerce')

print(climate_change_data[climate_change_data['Date'].isnull()])

climate_change_data = climate_change_data.dropna(subset=['Date'])

def show_country_data(country_name):
    country_data = climate_change_data[climate_change_data['Country']
== country_name]

    if country_data.empty:
        print(f"Brak danych dla kraju: {country_name}")
        return

    print(f"Dane dla kraju: {country_name}")
    print(country_data.head())

    plt.figure(figsize=(12, 6))
    plt.plot(country_data['Date'], country_data['Temperature'],
label='Temperature', marker='o')
    plt.plot(country_data['Date'], country_data['CO2 Emissions'],
label='CO2 Emissions', marker='x')
    plt.plot(country_data['Date'], country_data['Sea Level Rise'],
label='Sea Level Rise', marker='s')
    plt.xlabel('Date')
    plt.ylabel('Value')
    plt.title(f'Zmiany klimatyczne w {country_name}')
    plt.legend()
    plt.grid(True)

    plt.gca().xaxis.set_major_formatter(DateFormatter('%Y-%m-%d'))
    plt.gca().xaxis.set_tick_params(rotation=45)

    plt.show()

country_name = input("Podaj nazwę państwa (angielska nazwa, z wielkiej
litery):")
show_country_data(country_name)
```

```
Empty DataFrame
Columns: [Date, Location, Country, Temperature, CO2 Emissions, Sea
Level Rise, Precipitation, Humidity, Wind Speed]
Index: []
Dane dla kraju: Poland
```

```
                                Date          Location Country
Temperature  \
563  2001-04-17 23:12:54.437443744        Henryburgh  Poland
12.922686
859  2001-12-22 15:10:18.901890184  West Sharonhaven  Poland
11.724248
1037 2002-05-21 04:00:38.883888384       South Alvin  Poland
5.798536
1216 2002-10-18 13:00:42.124212416    Michaelborough  Poland
17.043241
1474 2003-05-23 06:48:42.772277216       East Javier  Poland
16.153522

      CO2 Emissions  Sea Level Rise  Precipitation   Humidity  Wind
Speed
563      376.232171       -0.460922      20.839448  20.821533
4.235199
859      381.706960        1.554425       8.163170  65.066638
12.596190
1037     265.786269       -0.831447      94.605102  49.763020
46.792132
1216     336.328398        0.515082      97.166092  75.954817
7.391963
1474     444.749433        0.115398      74.418231  71.447522
39.955686
```



Wykorzystanie modeli:

- drzew decyzyjnych
- lasów losowych
- gradient boosting

do predykcji jednej z następujących cech (zmienna wyjaśniana):

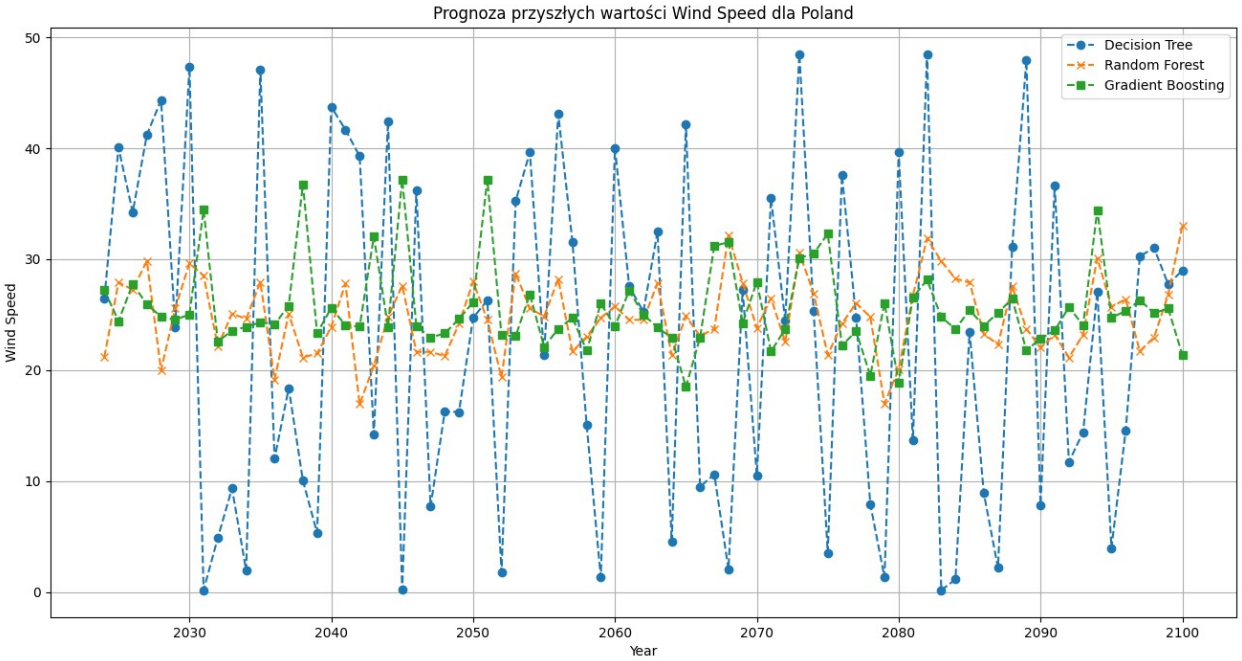
- Temperature
- CO2 Emissions
- Sea Level Rise
- Precipitation
- Humidity
- Wind Speed

(pozostałe zmienne stają się zmiennymi wyjaśniającymi) dla danego państwa.

```
columns = list(climate_change_data.columns)
columns.remove('Date')
columns.remove('Location')
columns.remove('Country')

print("Dostępne kolumny: ", columns)
target_variable = input(f"Podaj nazwę zmiennej docelowej ({columns}):
")


if target_variable not in columns:
    raise ValueError(f"Zmienna {target_variable} nie jest dostępna.
Wybierz jedną z dostępnych kolumn.")


features = [col for col in columns if col != target_variable]


X = climate_change_data[features]
y = climate_change_data[target_variable]


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


tree_reg = DecisionTreeRegressor()
forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)
gboost_reg = GradientBoostingRegressor(n_estimators=100,
random_state=42)

tree_reg.fit(X_train, y_train)
forest_reg.fit(X_train, y_train)
gboost_reg.fit(X_train, y_train)
```

```
y_pred_tree = tree_reg.predict(X_test)
y_pred_forest = forest_reg.predict(X_test)
y_pred_gboost = gboost_reg.predict(X_test)


print('Decision Tree MSE:', mean_squared_error(y_test, y_pred_tree))
print('Random Forest MSE:', mean_squared_error(y_test, y_pred_forest))
print('Gradient Boosting MSE:', mean_squared_error(y_test,
y_pred_gboost))
print('--------------------')
print('Decision Tree MAE:', mean_absolute_error(y_test, y_pred_tree))
print('Random Forest MAE:', mean_absolute_error(y_test,
y_pred_forest))
print('Gradient Boosting MAE:', mean_absolute_error(y_test,
y_pred_gboost))
print('--------------------')
print('Decision Tree R2:', r2_score(y_test, y_pred_tree))
print('Random Forest R2:', r2_score(y_test, y_pred_forest))
print('Gradient Boosting R2:', r2_score(y_test, y_pred_gboost))

future_years = np.arange(2024, 2101).reshape(-1, 1)


future_data = pd.DataFrame({
    feature: np.random.uniform(X[feature].min(), X[feature].max(),
size=future_years.shape[0])
    for feature in features
})

future_predictions_tree = tree_reg.predict(future_data)
future_predictions_forest = forest_reg.predict(future_data)
future_predictions_gboost = gboost_reg.predict(future_data)

plt.figure(figsize=(16, 8))
plt.plot(future_years, future_predictions_tree, label='Decision Tree',
linestyle='--', marker='o')
plt.plot(future_years, future_predictions_forest, label='Random
Forest', linestyle='--', marker='x')
plt.plot(future_years, future_predictions_gboost, label='Gradient
Boosting', linestyle='--', marker='s')
plt.xlabel('Year')
plt.ylabel(target_variable)
plt.title(f'Prognoza przyszłych wartości {target_variable} dla
{country_name}')
plt.legend()
plt.grid(True)
plt.show()

Dostępne kolumny:  ['Temperature', 'CO2 Emissions', 'Sea Level Rise',
'Precipitation', 'Humidity', 'Wind Speed']
```

```
Decision Tree MSE: 430.67392585167687
Random Forest MSE: 216.931935262709
Gradient Boosting MSE: 208.687413823388
--------------------
Decision Tree MAE: 16.859642038664976
Random Forest MAE: 12.665407561424141
Gradient Boosting MAE: 12.502882233916848
--------------------
Decision Tree R2: -1.0816641041627455
Random Forest R2: -0.04854135710658469
Gradient Boosting R2: -0.008691430500752828
```



//Analiza danych zestawionych, można by spojrzeć na korlacje między np. emisją CO2 a zaesieniem lub temperaturą itp.