Побудова моделі

Ознайомитись з різновидами регресійних моделей. Після завершення цієї лабораторної роботи ви зможете:

- Будувати регресійні моделі кількох видів
- Оцінювати якість моделі візуально
- Оцінювати якість моделі за допомогою числових мір, без використання тестової вибірки
- Виконувати прогнозування відгуку, використовуючи побудовану модель

1. Скачати дані із файлу "clean_data2.csv" (Data2.csv з виправленими помилками та заповненими пропусками). Записати дані у dataframe. В попередній роботі ви визначили ознаки, що можуть бути предикторами для 'CO2 emission'. Побудуйте моделі лінійної регресії для кожного з цих предикторів.
2. Побудуйте модель множинної лінійної регресії для всіх доречних предикторів разом.
3. Побудуйте кілька поліноміальних моделей другого порядку.
4. Побудуйте візуалізації для оцінки всіх моделей.
5. Порахуйте значення R^2 та MSE для оцінки якості кожної моделі. Оберіть найкращу модель.

Завдання #1:

Імпортую бібліотеки

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from termcolor import colored
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
```

Зчитую дані з файлу у датафрейм

```
df = pd.read_csv("../Data2-clean.csv", sep=';' ,encoding='cp1252')
df
```

```
    Country Name                        Region  GDP per capita
Population  \
0    Afghanistan                    South Asia      561.778746
34656032.0
1        Albania       Europe & Central Asia     4124.982390
```

```
2876101.0
2        Algeria  Middle East & North Africa     3916.881571
40606052.0
3        Andorra       Europe & Central Asia    36988.622030
77281.0
4         Angola          Sub-Saharan Africa     3308.700233
28813463.0
..           ...                         ...             ...          .
..
178      Vanuatu         East Asia & Pacific     2860.566475
270402.0
179      Vietnam         East Asia & Pacific     2170.648054
92701100.0
180  Yemen, Rep.  Middle East & North Africa      990.334774
27584213.0
181       Zambia          Sub-Saharan Africa     1269.573537
16591390.0
182     Zimbabwe          Sub-Saharan Africa     1029.076649
16150362.0

     CO2 emission       Area  Population density
0         9809.225   652860.0           53.083405
1         5716.853    28750.0          100.038296
2       145400.217  2381740.0           17.048902
3          462.042      470.0          164.427660
4        34763.160  1246700.0           23.111786
..             ...        ...                 ...
178        154.014    12190.0           22.182281
179     166910.839   330967.0          280.091671
180      22698.730   527970.0           52.245796
181       4503.076   752610.0           22.045136
182      12020.426   390760.0           41.330643

[183 rows x 7 columns]
```

З попередньої роботи знаємо, що хорошими показниками для `CO2 emission` можуть бути:

- `Population`.
- `Area`.

Розробимо моделі, використовуючи ці ознаки як змінні-предиктори.

Створюю об'єкт лінійної регресії та навчаю першу (`Population`) модель

```
x_p = df[['Population']]
x_a = df[['Area']]
y = df['CO2 emission']

model_1 = LinearRegression()
model_1.fit(x_p, y)
```

```
Yhat1 = model_1.predict(x_p)
Yhat1
```

```
array([ 1.58214516e+05,  4.03076403e+03,  1.87081678e+05, -
9.54801216e+03,
        1.29868665e+05, -9.43311639e+03,  2.02807572e+05,
4.26711009e+03,
        1.07132557e+05,  3.25157987e+04,  3.74397728e+04, -
8.02484512e+03,
       -3.00857888e+03,  7.80654066e+05, -8.54026050e+03,
3.62018657e+04,
        4.51338644e+04, -8.14263245e+03,  4.28251736e+04, -
6.05250656e+03,
        4.29007810e+04,  7.13926174e+03,  9.94429119e+02,
9.97527254e+05,
       -7.86976834e+03,  2.46584456e+04,  8.05422255e+04,
4.11359360e+04,
       -7.30521588e+03,  6.65498837e+04,  1.03794797e+05,
1.66124543e+05,
        1.23683486e+04,  6.01951172e+04,  7.69681481e+04,
6.67881854e+06,
        2.26124345e+05, -6.06300544e+03,  3.72073963e+05,
1.49455225e+04,
        1.36426379e+04,  1.05040350e+05,  1.03111651e+04, -
4.24596198e+03,
        4.13179489e+04,  1.78821860e+04, -9.56614746e+03,
4.17408053e+04,
        6.95709700e+04,  4.54320976e+05,  2.08591518e+04, -
3.99675916e+03,
       -3.53590011e+03,  4.86897167e+05, -5.56251845e+03,
1.67370999e+04,
        3.14630717e+05, -3.17804576e+02, -3.29424396e+01,
8.12163464e+03,
        3.91148214e+05,  1.26925025e+05,  4.22160154e+04, -
9.40228927e+03,
        7.05286822e+04,  5.02172090e+04, -1.11389515e+03, -
6.17118658e+03,
        4.27040581e+04,  3.42891049e+04,  2.57203558e+04,
3.77099297e+04,
       -8.30128972e+03,  6.41443676e+06,  1.25690676e+06,
3.79551625e+05,
        1.70569329e+05,  1.32342344e+04,  3.15442255e+04,
2.84087329e+05,
        4.05625437e+03,  6.06204576e+05,  3.59528909e+04,
7.64212652e+04,
        2.25193554e+05, -9.36794959e+03,  2.38701109e+05,
9.73859780e+03,
        1.95879239e+04,  2.28659271e+04, -4.11741401e+02,
1.92190464e+04,
```

```
        7.69125317e+02,  1.24615092e+04,  4.01231336e+03, -
7.09459789e+03,
       -6.95295521e+03,  1.74245460e+02,  1.10855642e+05,
7.78502722e+04,
        1.41385420e+05, -7.89744191e+03,  7.73809374e+04, -
7.80305388e+03,
       -9.66549383e+03,  1.09439021e+04, -3.79307412e+03,
6.08853127e+05,
       -9.41383610e+03,  7.30996070e+03,  4.76479769e+03, -
6.90146025e+03,
        1.61226171e+05,  1.29946354e+05,  2.46655397e+05,
2.10764502e+03,
       -9.85964061e+03,  1.30690081e+05,  7.26436871e+04,
1.28441894e+04,
        1.99140877e+04,  8.92425788e+05,  1.54651685e+04,
1.15442594e+04,
        9.27424489e+05, -9.81862512e+03,  9.64901287e+03,
2.93022540e+04,
        2.27056057e+04,  1.44231247e+05,  4.91346218e+05,
1.74185931e+05,
        4.01680112e+04,  2.54473119e+03,  8.56794374e+04,
6.90369706e+05,
        4.78961223e+04, -8.97627932e+03, -8.95306438e+03,
1.46666017e+05,
        6.48481533e+04,  2.43168439e+04, -9.46361359e+03,
2.59604618e+04,
        1.72813872e+04,  1.64149920e+04,  9.48683091e+01, -
7.01480350e+03,
        5.95423479e+04,  2.61324921e+05,  2.15404912e+05,
9.29456852e+04,
       -9.65697926e+03, -9.05929032e+03, -9.39100443e+03,
1.82097991e+05,
       -7.21396684e+03, -3.40676487e+03,  3.81231120e+04,
3.06951844e+04,
        3.24556048e+04,  2.59691559e+05,  3.24175794e+05, -
3.76785546e+03,
        2.69801921e+04, -9.40323534e+03, -3.30068932e+03,
4.54011346e+04,
        3.75840135e+05,  1.75494919e+04, -9.86911095e+03,
1.91360374e+05,
        2.08421925e+05,  3.50495701e+04,  3.08523174e+05,
1.55776493e+06,
        6.78601620e+03,  1.44592018e+05, -8.61106486e+03,
4.39826405e+05,
        1.23904826e+05,  7.05719634e+04,  6.84322686e+04])
```

Знаходжу коефіцієнти моделі

```
print(f"coefficient: {colored(model_1.coef_, 'green')}")
print(f"interception point: {colored(model_1.intercept_, 'blue')}")
```

```
coefficient: [0.00485161]
interception point: -9922.949236649234
```

Отримали кінцеву лінійну модель зі структурою:

$$ Yhat1 = -9922.9492 + 0.00485 \cdot `Population` $$

Підставляючи фактичні значення, маємо:

```
test_population = 1e9

model_1.predict([[test_population]])
predicted_co2_m1 = model_1.predict([[test_population]])[0]
predicted_co2_m1
```

```
C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(

4841684.578578547
```

Створюю об'єкт лінійної регресії та навчаю другу модель

```
model_2 = LinearRegression()
model_2.fit(x_a, y)

Yhat2 = model_2.predict(x_a)
Yhat2
```

```
array([ 1.69753080e+05,  6.16484104e+03,  6.22917481e+05, -
1.24775513e+03,
        3.25407115e+05, -1.25561857e+03,  7.27412021e+05,
6.42433433e+03,
        2.02771454e+06,  2.06149476e+04,  2.13281610e+04,
2.26719942e+03,
       -1.16885869e+03,  3.73250049e+04, -1.25823971e+03,
5.30440074e+04,
        6.63140473e+03,  4.64981961e+03,  2.87093034e+04,
```

```
8.69267263e+03,
        2.86582725e+05,  1.20519312e+04,  1.51108880e+05,
2.23073528e+06,
        1.41451362e+02,  2.77237532e+04,  7.05060710e+04,
5.92369576e+03,
       -3.14627752e+02,  4.60822481e+04,  1.23248737e+05,
2.61575517e+06,
        1.61921101e+05,  3.35183983e+05,  1.96812726e+05,
2.50520605e+06,
        2.97897943e+05, -8.83153958e+02,  6.13250701e+05,
8.82721872e+04,
        1.20230986e+04,  8.31504712e+04,  1.34621069e+04,
1.05360959e+03,
        1.93020164e+04,  9.87952679e+03, -1.17436309e+03,
1.13861605e+04,
        6.58273283e+04,  2.61123550e+05,  4.14393876e+03,
5.98136093e+03,
        1.04844869e+04,  2.88082020e+05,  3.41788178e+03,
8.73338176e+04,
        1.42552679e+05,  6.87892214e+04,  1.59094418e+03,
1.68984271e+04,
        9.23035072e+04,  6.11538280e+04,  3.32176717e+04, -
1.28183001e+03,
        2.71706917e+04,  6.30725056e+04,  8.09924555e+03,
5.49757908e+04,
        5.90272660e+03,  2.81143037e+04, -1.08131247e+03,
2.30135568e+04,
        2.56268377e+04,  8.60267081e+05,  4.99511647e+05,
4.56058054e+05,
        1.12661935e+05,  1.70504534e+04,  4.41391662e+03,
7.76146144e+04,
        1.50968870e+03,  9.76983465e+04,  2.20411123e+04,
7.12865194e+05,
        1.50752405e+05, -1.15863623e+03,  2.49138864e+04,
3.29993028e+03,
        5.10385699e+04,  6.06977489e+04,  1.55328109e+04,
1.36814691e+03,
        6.58684527e+03,  2.78207355e+04,  1.57414540e+04, -
6.92072536e+02,
       -1.36300685e+03,  5.36801316e+03,  1.52567547e+05,
2.96843691e+04,
        8.53365056e+04, -1.29231459e+03,  3.23700750e+05, -
1.28707230e+03,
       -1.32376832e+03,  2.68790397e+05, -8.36235474e+02,
5.13521401e+05,
       -1.18746881e+03,  7.50162465e+03,  4.08607478e+05,
2.24885140e+03,
        1.15676251e+05,  2.08158087e+05,  1.75973055e+05,
2.14687357e+05,
```

```
       -1.36570663e+03,  3.72070534e+04,  9.51728464e+03,
6.87997060e+04,
        3.28009097e+04,  2.40762501e+05,  9.95897642e+04,
7.97534682e+04,
        2.07298351e+05, -1.25037628e+03,  1.83977216e+04,
1.19946095e+05,
        1.05244621e+05,  3.35503763e+05,  7.72633811e+04,
8.05869921e+04,
        2.28025547e+04,  1.67219965e+03,  6.11145108e+04,
4.48032716e+06,
        5.53314525e+03, -6.26543928e+02, -1.11931906e+03,
5.62093827e+05,
        5.01895812e+04,  2.17894824e+04, -1.25037628e+03,
1.75799246e+04,
       -1.18248864e+03,  1.14818323e+04,  3.94211064e+03,
6.20415820e+03,
        1.65768941e+05,  3.18170135e+05,  1.31243227e+05,
1.58263790e+04,
       -1.30279917e+03, -1.20843797e+03, -1.26872429e+03,
4.91235777e+05,
        4.15686375e+04,  3.17935764e+03,  1.15904291e+05,
9.45175603e+03,
        3.56857412e+04,  2.46930054e+05,  1.33125209e+05,
2.52669270e+03,
        1.35145297e+04, -1.17436309e+03, -2.63018756e+01,
4.15135935e+04,
        2.04480621e+05,  1.26567106e+05, -1.36308549e+03,
6.19427924e+04,
        1.56828217e+05,  2.05418177e+04,  6.24827482e+04,
2.57560972e+06,
        4.48188565e+04,  1.15899049e+05,  1.82422602e+03,
8.53802787e+04,
        1.37017608e+05,  1.95898995e+05,  1.01052887e+05])
```

Знаходжу коефіцієнти моделі

```
print(f"coefficient: {colored(model_2.coef_, 'green')}")
print(f"interception point: {colored(model_2.intercept_, 'blue')}")

coefficient: [0.26211443]
interception point: -1370.948918094975
```

Отримали кінцеву лінійну модель зі структурою:

$$ Yhat2 = 1370.9489 + 0.26211443 \cdot `Area` $$

Підставляючи фактичні значення, маємо:

```
test_area = 0.5e7

model_2.predict([[test_area]])
predicted_co2_m2 = model_2.predict([[test_area]])[0]
predicted_co2_m2
```

```
C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(

1309201.2172565707
```

Завдання #2:

Створюю об'єкт лінійної регресії та навчаю множинну лінійну модель із структурою

$$ Yhat3 = -68459.37215735792 + 0.00408196 \cdot `Population` + 0.12675393 \cdot `Area` $$

```
x = df[['Population', 'Area']]

model_3 = LinearRegression()
model_3.fit(x, y)

Yhat3 = model_3.predict(x)
Yhat3
```

```
array([ 1.55757671e+05, -5.30750728e+04,  3.99187744e+05, -
6.80843400e+04,
        2.07180105e+05, -6.79914737e+04,  4.62950636e+05, -
5.27507339e+04,
        1.01125677e+06, -2.21210302e+04, -1.76332882e+04, -
6.51030349e+04,
       -6.25441566e+04,  6.15414749e+05, -6.72415262e+04, -
3.33759018e+03,
       -1.82668652e+04, -6.40499435e+04, -9.53282600e+03, -
6.03363383e+04,
        1.15233842e+05, -4.76128077e+04,  1.44626607e+04,
1.85857826e+06,
```

```
       -6.60005336e+04, -2.52942149e+04,  4.24130495e+04, -
2.19728057e+04,
       -6.57460925e+04,  1.88294937e+04,  8.74823051e+04,
1.34525649e+06,
        2.92608438e+04,  1.53287353e+05,  1.00485635e+05,
6.77132996e+06,
        2.74863022e+05, -6.49758732e+04,  5.50158533e+05, -
4.18614038e+03,
       -4.21550573e+04,  6.91394494e+04, -4.42621527e+04, -
6.25104971e+04,
       -1.53501462e+04, -3.96246564e+04, -6.80641073e+04, -
1.88223397e+04,
        3.09196946e+04,  4.49075540e+05, -3.98935801e+04, -
5.99178533e+04,
       -5.73524715e+04,  4.89520551e+05, -6.24748771e+04, -
3.13255439e+03,
        2.74206678e+05, -2.64497433e+04, -5.87059771e+04, -
4.44425963e+04,
        3.14285974e+05,  7.69151926e+04, -7.86518084e+03, -
6.79782123e+04,
        1.30318071e+04,  1.33039922e+04, -5.64681494e+04, -
3.80544886e+04,
       -2.06635878e+04, -1.70024810e+04, -3.83303876e+04, -
1.65909607e+04,
       -5.40393143e+04,  5.75342560e+06,  1.23962100e+06,
4.80434351e+05,
        1.38544266e+05, -4.00675320e+04, -3.07730089e+04,
2.17105727e+05,
       -5.53047761e+04,  4.97835069e+05, -1.85395234e+04,
3.49579414e+05,
        2.02922894e+05, -6.78897459e+04,  1.53434340e+05, -
4.96581390e+04,
       -1.82857234e+04, -1.08567270e+04, -5.22826424e+04, -
4.26158265e+04,
       -5.56152178e+04, -3.55093545e+04, -4.84595148e+04, -
6.57514122e+04,
       -6.59566915e+04, -5.67051329e+04,  1.07601093e+05,
2.04074848e+04,
        1.00775938e+05, -6.67171611e+04,  1.62193759e+05, -
6.66352116e+04,
       -6.82199433e+04,  7.97424824e+04, -6.30433503e+04,
7.01148181e+05,
       -6.79422960e+04, -4.96696364e+04,  1.42156702e+05, -
6.41667344e+04,
        1.32140959e+05,  1.50545900e+05,  2.33176335e+05,
4.61447123e+04,
       -6.84035716e+04,  6.85027282e+04,  6.27441439e+03, -
1.53706719e+04,
       -2.68307135e+04,  8.07834024e+05,  1.72405133e+03, -
```

```
1.11673367e+04,
        8.21097925e+05, -6.83132910e+04, -4.24324858e+04,
2.32100129e+04,
        1.05504692e+04,  2.24146797e+05,  3.91315624e+05,
1.26076259e+05,
       -1.46248613e+04, -5.64979268e+04,  4.21937108e+04,
2.68801068e+06,
       -1.64739054e+04, -6.73028989e+04, -6.75216641e+04,
3.35770293e+05,
        1.93839565e+04, -2.84513346e+04, -6.80145978e+04, -
2.91041252e+04,
       -4.54795419e+04, -4.00842508e+04, -5.74614592e+04, -
6.23493802e+04,
        7.08120007e+04,  3.14282724e+05,  1.85252807e+05,
2.64067107e+04,
       -6.82026391e+04, -6.76541349e+04, -6.79623800e+04,
3.31315701e+05,
       -4.54153080e+04, -6.07764541e+04,  2.86770017e+04, -
2.90511491e+04,
       -1.48837042e+04,  2.78458024e+05,  2.77678584e+05, -
6.13958793e+04,
       -3.02121164e+04, -6.79270392e+04, -6.22374068e+04, -
1.17358070e+03,
        3.55653220e+05,  1.65234901e+04, -6.84102721e+04,
1.31510175e+05,
        1.91750039e+05, -2.00245756e+04,  2.30347612e+05,
2.49671616e+06,
       -3.20645058e+04,  1.18253355e+05, -6.58104721e+04,
3.51894002e+05,
        1.11060503e+05,  9.46622644e+04,  4.69960957e+04])
```

Знаходжу коефіцієнти моделі

```
print(f"coefficient: {colored(model_3.coef_, 'green')}")
print(f"interception point: {colored(model_3.intercept_, 'blue')}")

coefficient: [0.00408196 0.12675393]
interception point: -68459.37215735792
```

Підставляючи фактичні значення, маємо:

```
model_3.predict([[test_population, test_area]])
predicted_co2_m3 = model_3.predict([[test_population, test_area]])[0]
predicted_co2_m3

C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
```

```
with feature names
  warnings.warn(
C:\Users\local_gud2i5y\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-
packages\Python312\site-packages\sklearn\base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(

4647268.381848969
```

Завдання #3:

Створюю об'єкт поліноміальної регресії та навчаю поліноміальну модель із структурою

$$ Yhat4 = - 2.149 \cdot 10^{4} + 0.005384 \cdot `Population` - 4.224 \cdot 10^{-13} \cdot `Population`^2 $$

```
f4 = np.polyfit(df['Population'], df['CO2 emission'], 2)
p4 = np.poly1d(f4)
p4(test_population)

4939878.49945833
```

Знаходжу коефіцієнти моделі

```
print(p4)

            2
-4.224e-13 x + 0.005384 x - 2.149e+04
```

Створюю об'єкт поліноміальної регресії та навчаю поліноміальну модель із структурою

$$ Yhat5 = -68883.26503891358 + 4.60675779 \cdot 10^{-1} \cdot `Area` - 1.65261389 \cdot 10^{-8} \cdot `Area`^2 $$

```
f5 = np.polyfit(df['Area'], df['CO2 emission'], 2)
p5 = np.poly1d(f5)
p5(test_area)

1821342.154418972
```

Знаходжу коефіцієнти моделі

```
print(p5)
```

```
            2
-1.653e-08 x + 0.4607 x - 6.888e+04
```

Створюю об'єкт поліноміальної регресії та навчаю поліноміальну модель із структурою

$$ Yhat6 = 2367.3231821453373 + 2.35469535 \cdot 10^{-3} \cdot `Population` + 2.74701507 \cdot 10^{-2} \cdot `Area` \\ - 3.16730279 \cdot 10^{-12} \cdot `Population` \cdot `Area` + 1.01806703 \cdot 10^{-9} \cdot `Population`^2 - 4.46783405 \cdot 10^{-9} \cdot `Area`^2 $$

```
Input = [('scale', StandardScaler()),
         ('polynomial', PolynomialFeatures(degree=2,
include_bias=False)),
         ('model', LinearRegression())]
pipe = Pipeline(Input)
pipe.fit(x, y)

ypipe = pipe.predict(x)
ypipe
```

```
array([1.19231819e+05, 9.98369387e+03, 2.31302305e+05, 2.56219622e+03,
       1.31458295e+05, 2.61713103e+03, 2.65480082e+05, 1.01288183e+04,
       1.92394653e+05, 2.57420224e+04, 2.82587258e+04, 3.67399586e+03,
       5.73900145e+03, 3.30415813e+05, 3.05005266e+03, 3.19869795e+04,
       2.98680880e+04, 3.86814678e+03, 3.19577337e+04, 5.32305885e+03,
       6.45928166e+04, 1.21875334e+04, 2.34508819e+04, 2.06495547e+06,
       3.52406552e+03, 2.27898514e+04, 5.75752397e+04, 2.78567997e+04,
       3.74971838e+03, 4.64278947e+04, 7.92151046e+04, 2.81359988e+05,
       3.14128086e+04, 8.25351547e+04, 7.55254198e+04, 1.05049197e+07,
       1.91527208e+05, 4.29130227e+03, 3.95940418e+05, 2.50107456e+04,
       1.53747205e+04, 7.25580405e+04, 1.39132038e+04, 5.38298255e+03,
       2.98702459e+04, 1.71795803e+04, 2.56110545e+03, 2.89368260e+04,
       5.11242237e+04, 3.19272116e+05, 1.78915828e+04, 6.04071087e+03,
       6.75567304e+03, 3.50295826e+05, 4.99814825e+03, 2.58889396e+04,
       1.96845206e+05, 1.45890057e+04, 7.48747646e+03, 1.32381844e+04,
       2.14703567e+05, 7.94140916e+04, 3.22977214e+04, 2.62933383e+03,
       4.53195634e+04, 4.06557089e+04, 7.68572035e+03, 1.00543253e+04,
       2.86020954e+04, 2.76395066e+04, 1.95342004e+04, 2.86270433e+04,
       5.97107928e+03, 2.04030551e+06, 9.45431187e+05, 3.47944454e+05,
       1.13167052e+05, 1.57843710e+04, 2.30578674e+04, 1.59895010e+05,
       9.45930941e+03, 3.08930453e+05, 2.76274776e+04, 1.34321430e+05,
       1.52112801e+05, 2.65896114e+03, 1.22659165e+05, 1.24195006e+04,
       2.31252261e+04, 2.60201866e+04, 8.85300484e+03, 1.67473733e+04,
       8.43923248e+03, 1.66910812e+04, 1.10698345e+04, 3.81159513e+03,
       3.80842628e+03, 8.01196195e+03, 8.85001418e+04, 4.93049392e+04,
       9.18245123e+04, 3.35817769e+03, 9.36306255e+04, 3.40449983e+03,
       2.49719451e+03, 4.05165607e+04, 5.39597790e+03, 5.42950803e+05,
       2.63365647e+03, 1.17383663e+04, 4.63238196e+04, 4.21979299e+03,
       1.08905176e+05, 1.10185655e+05, 1.71006301e+05, 2.98754051e+04,
       2.39857048e+03, 7.62413945e+04, 4.33762526e+04, 2.16602384e+04,
```

```
       2.10503107e+04, 5.27231780e+05, 2.65725854e+04, 2.21924956e+04,
       5.14700515e+05, 2.43057189e+03, 1.41710083e+04, 3.67647993e+04,
       3.12793975e+04, 1.43486815e+05, 2.51238833e+05, 1.07394763e+05,
       2.98058106e+04, 8.74618580e+03, 5.86145455e+04, 1.95237128e+06,
       3.10196011e+04, 2.90517772e+03, 2.86445820e+03, 1.84109244e+05,
       4.62218295e+04, 2.18548388e+04, 2.60288174e+03, 2.21169527e+04,
       1.54950088e+04, 1.66641606e+04, 7.81346760e+03, 4.58539833e+03,
       6.04274230e+04, 2.20353238e+05, 1.41573675e+05, 5.40693413e+04,
       2.50352864e+03, 2.80350788e+03, 2.63618898e+03, 2.02174427e+05,
       8.15447588e+03, 6.02344143e+03, 4.12827227e+04, 2.33375913e+04,
       2.77453288e+04, 1.99049241e+05, 1.98392781e+05, 5.77623058e+03,
       2.20801325e+04, 2.64018020e+03, 5.72339722e+03, 3.50808064e+04,
       2.51961788e+05, 3.07568864e+04, 2.39424904e+03, 1.11184317e+05,
       1.44529833e+05, 2.69764784e+04, 1.65983139e+05, 3.50497769e+06,
       1.57592296e+04, 1.00049652e+05, 3.34133081e+03, 2.33269623e+05,
       9.29945166e+04, 7.14191738e+04, 5.60473069e+04])
```

Знаходжу коефіцієнти моделі

```
print(f"coefficient: {colored(pipe.named_steps['model'].coef_,
'green')}")
print(f"interception point:
{colored(pipe.named_steps['model'].intercept_, 'blue')}")
```

```
coefficient: [409749.22941313 121284.82415744 -66927.69067616
291103.03583485
 -17287.0721734 ]
interception point: 136537.42783411834
```

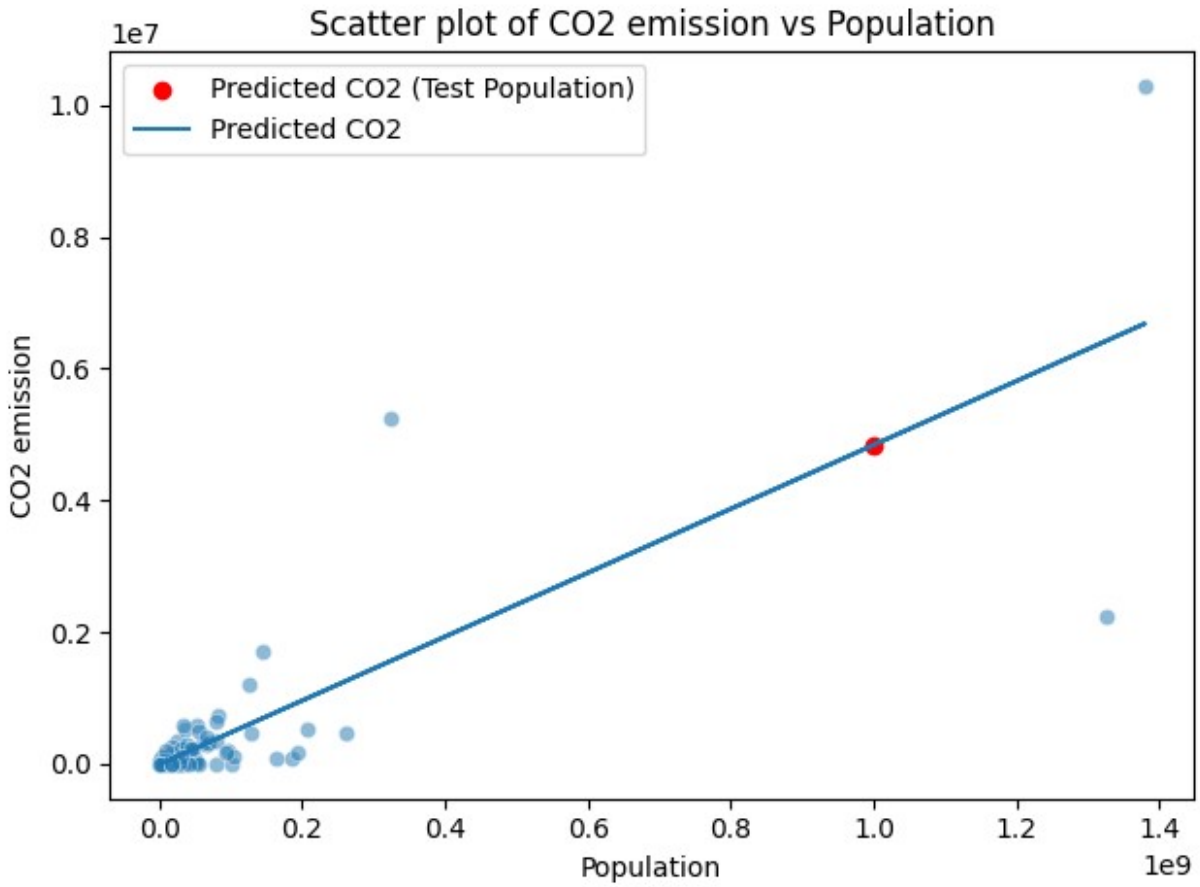
Завдання #4:

Для простої лінійної регресії чудовим способом візуалізації відповідності моделі є використання графіків регресії. Цей графік покаже комбінацію розсіяних точок даних (діаграма розсіювання, scatterplot) та підігнану лінію лінійної регресії, що проходить через дані.

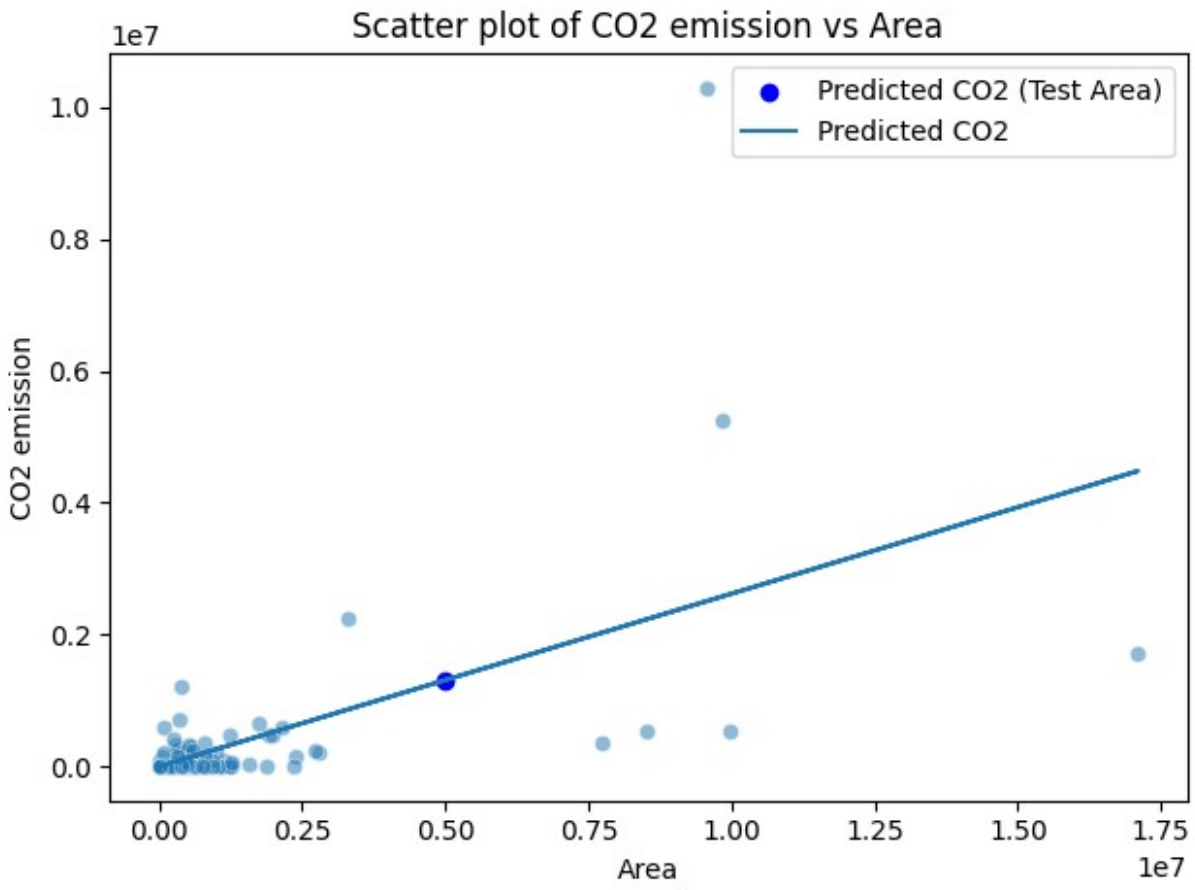
Таким способом візуалізую перші моделі.

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

sns.scatterplot(data=df, x='Population', y='CO2 emission', alpha=0.5)
plt.scatter(x=test_population, y=predicted_co2_m1, color='red',
marker='o', label='Predicted CO2 (Test Population)')
plt.plot(df['Population'], Yhat1, label='Predicted CO2')
plt.title(f'Scatter plot of CO2 emission vs Population')
```

```
plt.legend()
plt.tight_layout()
```

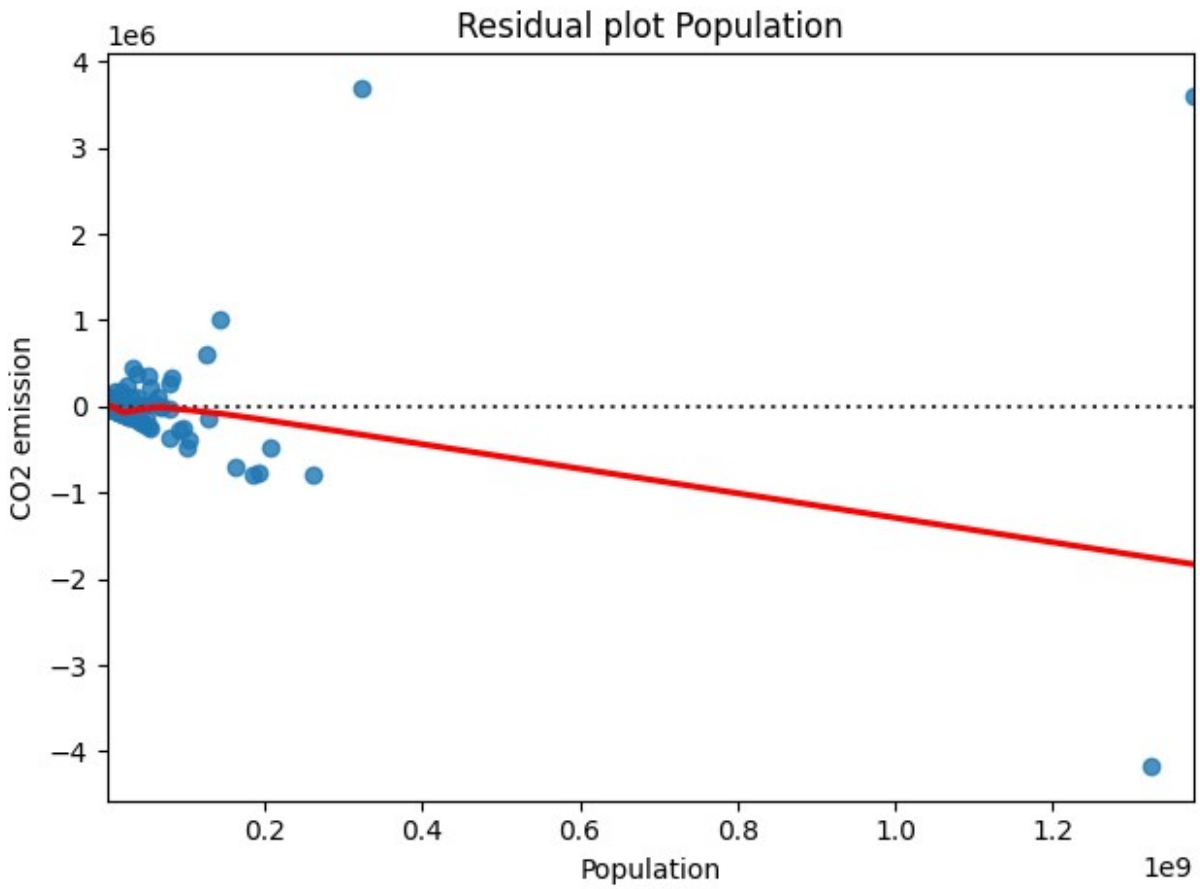



```
sns.scatterplot(data=df, x='Area', y='CO2 emission', alpha=0.5)
plt.scatter(x=test_area, y=predicted_co2_m2, color='blue', marker='o',
label='Predicted CO2 (Test Area)')
plt.plot(df['Area'], Yhat2, label='Predicted CO2')
plt.title(f'Scatter plot of CO2 emission vs Area')
plt.legend()
plt.tight_layout()
```

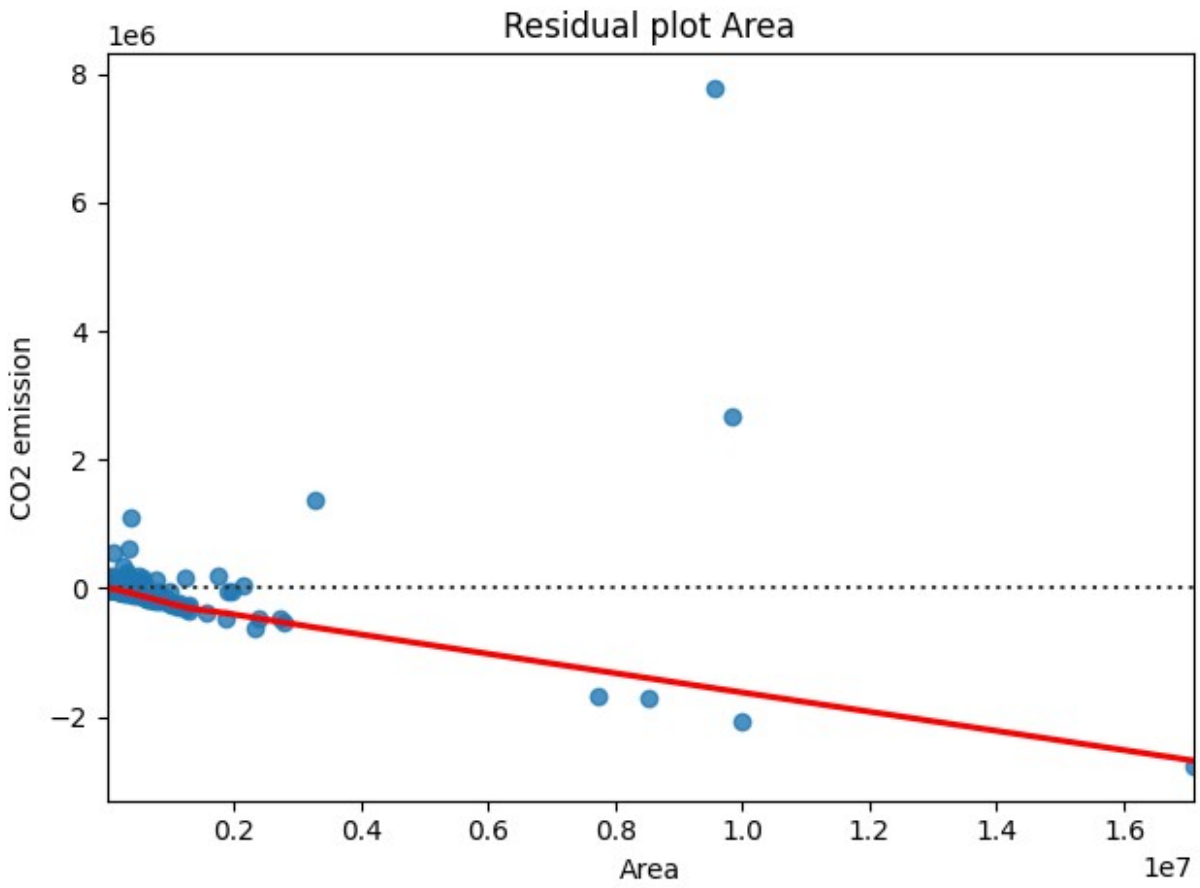


Для візуалізації дисперсії даних використаю діаграми залишків - по ній зможу прийняти рішення про необхідність ускладнення моделі.

```
sns.residplot(data=df, x='Population', y='CO2 emission', lowess=True,
line_kws={'color': 'red'})
plt.title('Residual plot Population')
plt.tight_layout()
```

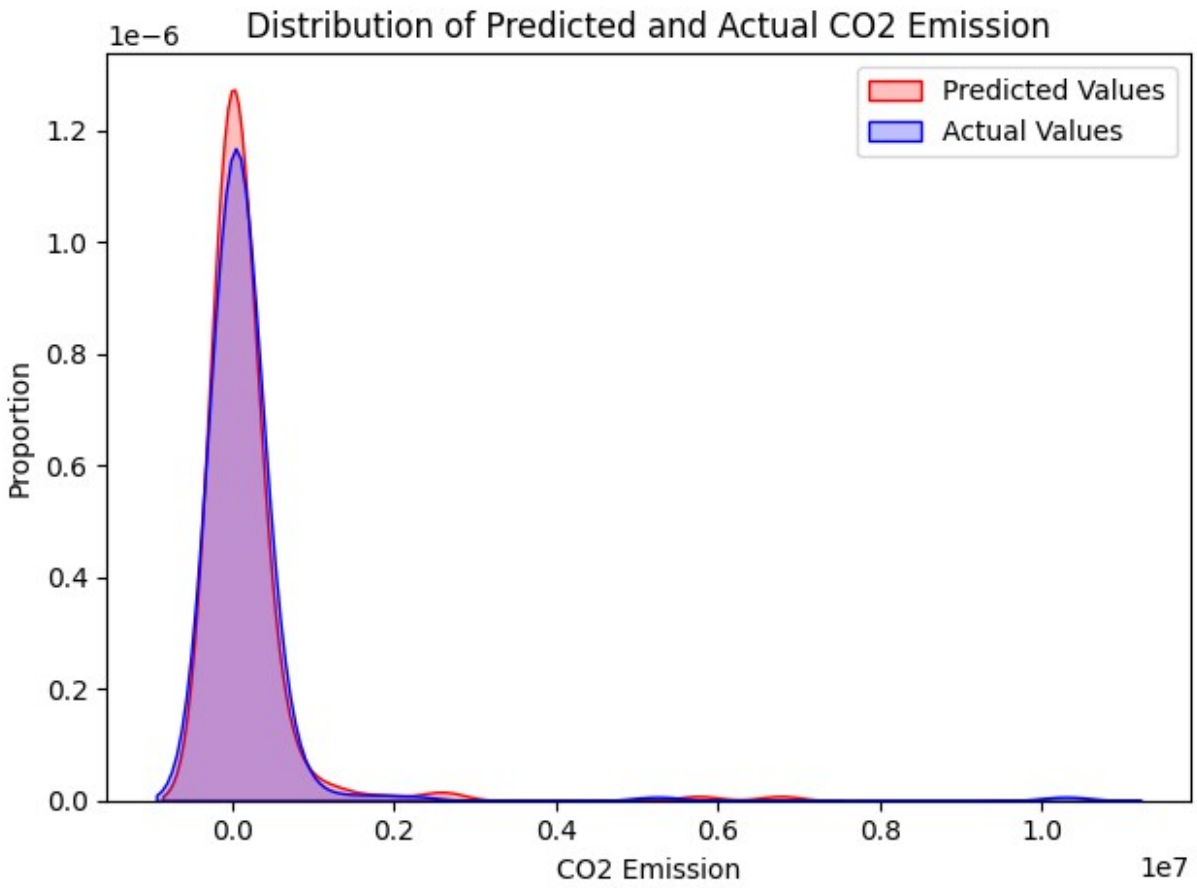



```
sns.residplot(data=df, x='Area', y='CO2 emission', lowess=True,
line_kws={'color': 'red'})
plt.title('Residual plot Area')
plt.tight_layout()
```



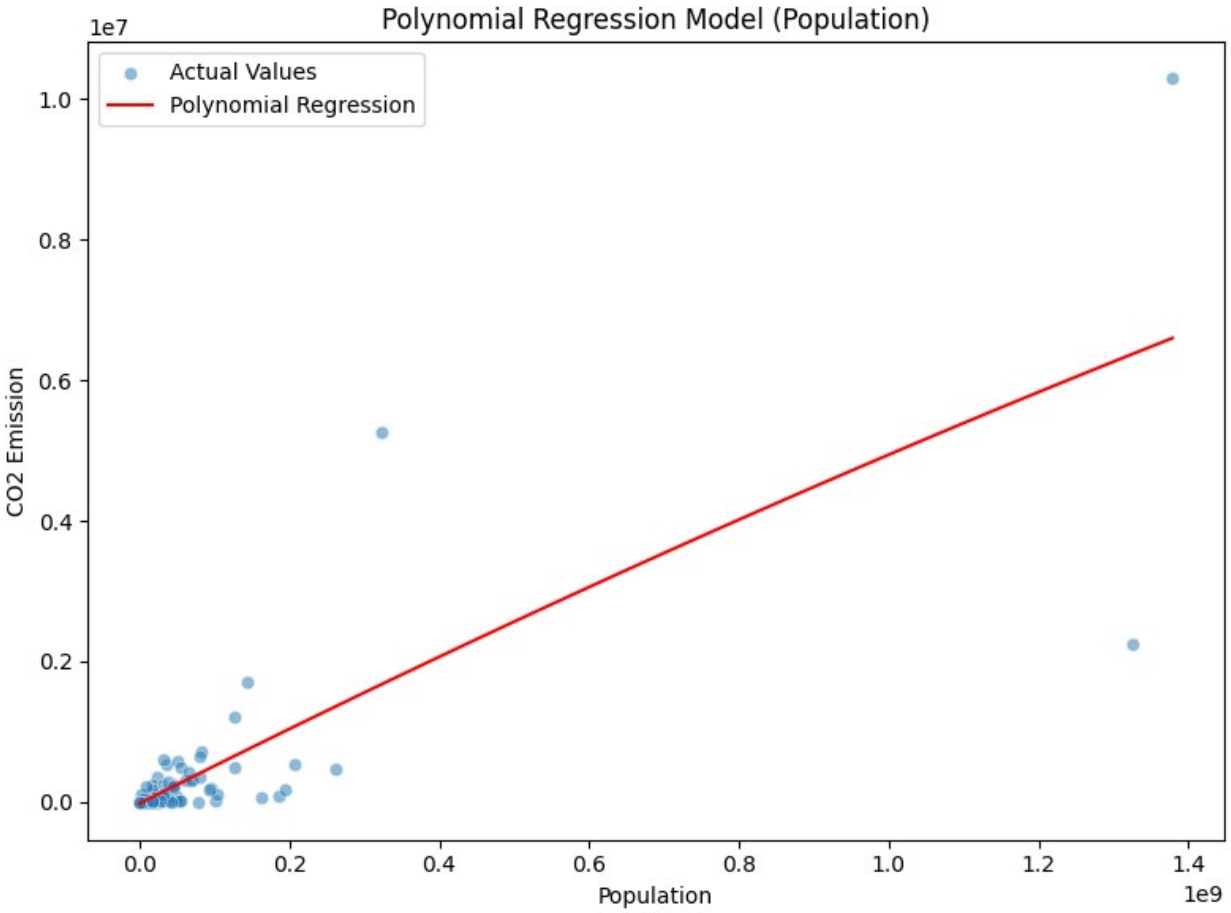
Для візуалізації моделі множинної лінійної регресії використаю діаграму розподілу.

```
sns.kdeplot(x=Yhat3, fill=True, color='r', label='Predicted Values')
sns.kdeplot(x=df['CO2 emission'], fill=True, color='b', label='Actual
Values')

plt.title('Distribution of Predicted and Actual CO2 Emission')
plt.xlabel('CO2 Emission')
plt.ylabel('Proportion')
plt.legend()
# plt.xscale('log')
plt.tight_layout()
```

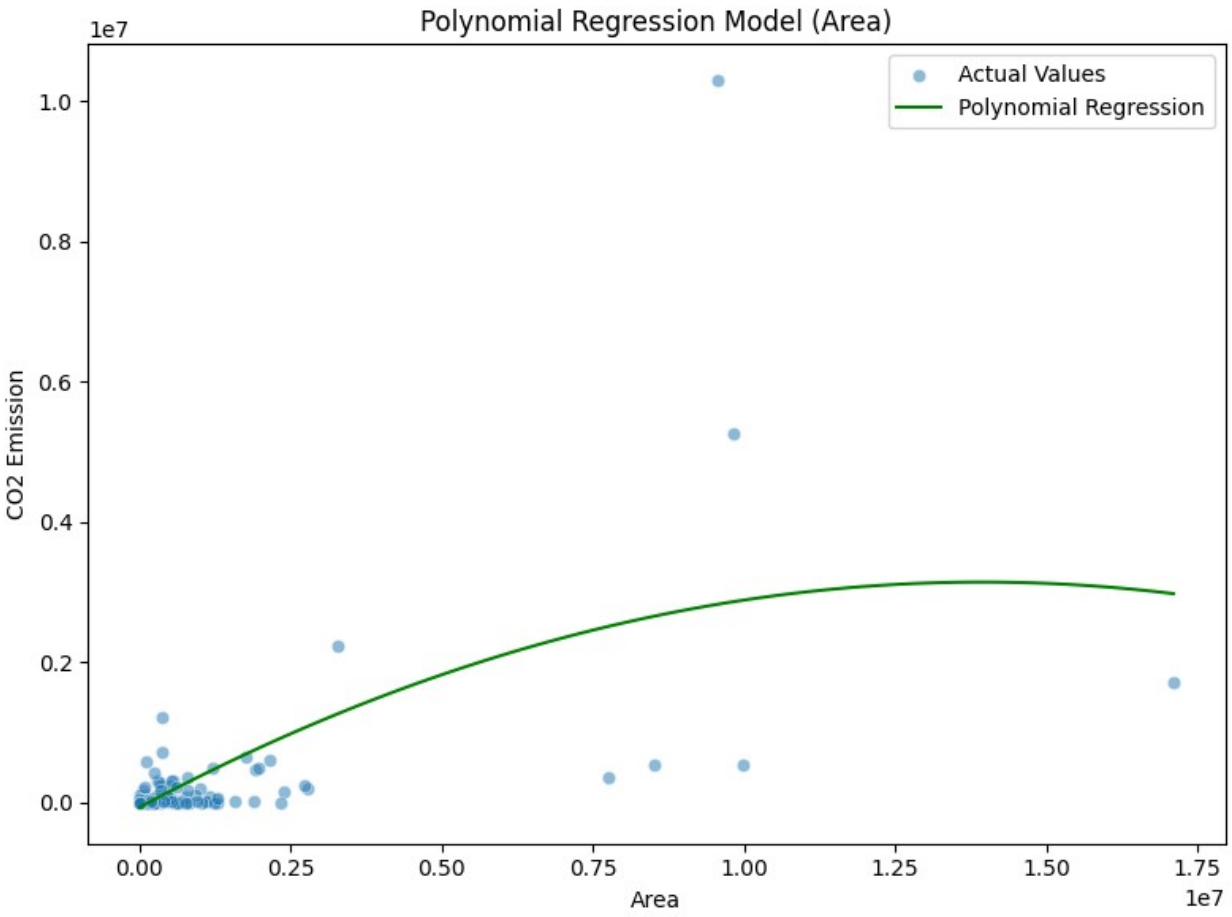


Для візуалізації поліноміальних моделей з одним предиктором використаю функцію `poly1d`. Зручно розміщувати всі моделі на одному графіку для обрання найкращої.

```
x_range_population = np.linspace(df['Population'].min(),
df['Population'].max(), 100)
x_range_area = np.linspace(df['Area'].min(), df['Area'].max(), 100)

y_values_p4 = p4(x_range_population)
y_values_p5 = p5(x_range_area)

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Population', y='CO2 emission',
label='Actual Values', alpha=0.5)
plt.plot(x_range_population, y_values_p4, 'r-', label='Polynomial
Regression')
plt.title('Polynomial Regression Model (Population)')
plt.xlabel('Population')
plt.ylabel('CO2 Emission')
plt.legend()
plt.tight_layout()
```

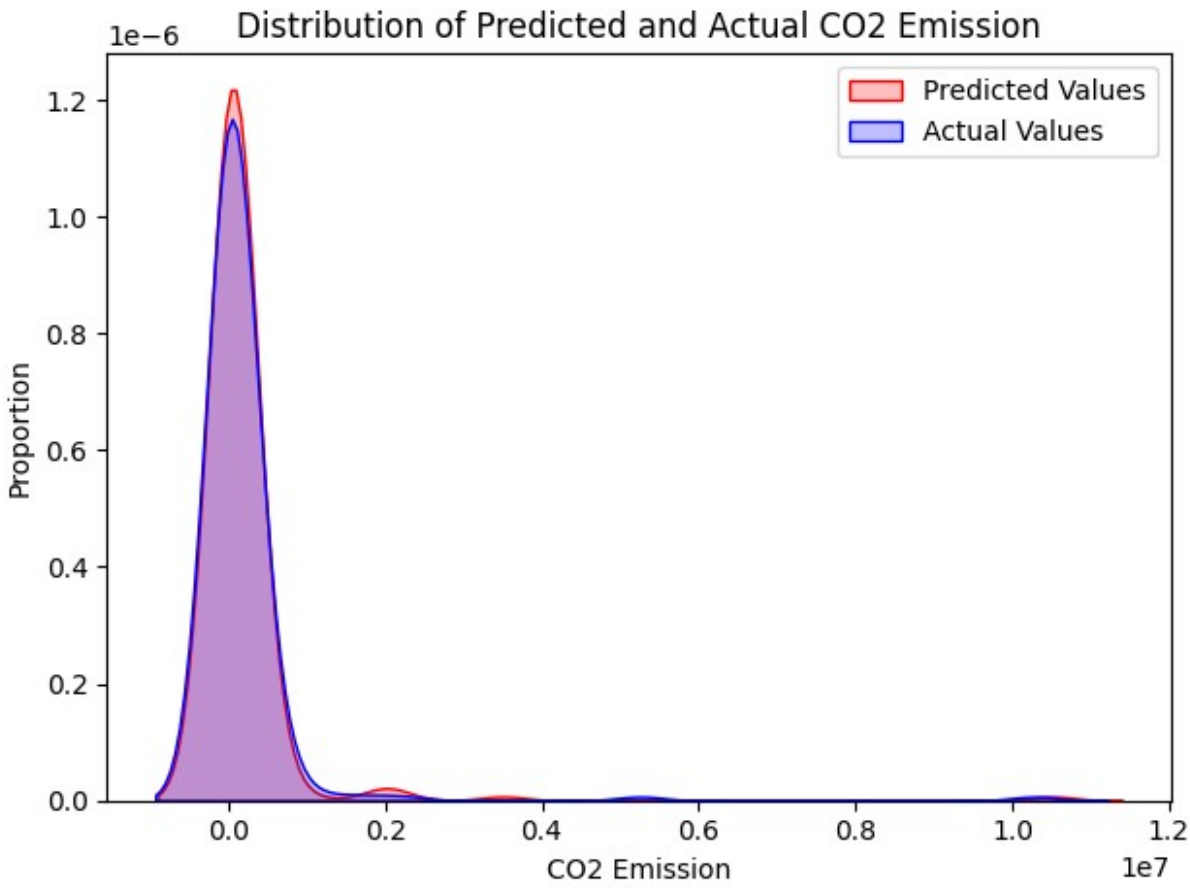



```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Area', y='CO2 emission', label='Actual
Values', alpha=0.5)
plt.plot(x_range_area, y_values_p5, 'g-', label='Polynomial
Regression')
plt.title('Polynomial Regression Model (Area)')
plt.xlabel('Area')
plt.ylabel('CO2 Emission')
plt.legend()
plt.tight_layout()
```



Для візуалізації поліноміальних моделей з кількома предикторами використаю діаграму розподілу або діаграму залишків

```
sns.kdeplot(x=ypipe, fill=True, color='r', label='Predicted Values')
sns.kdeplot(x=df['CO2 emission'], fill=True, color='b', label='Actual
Values')

plt.title('Distribution of Predicted and Actual CO2 Emission')
plt.xlabel('CO2 Emission')
plt.ylabel('Proportion')
plt.legend()
# plt.xscale('log')
plt.tight_layout()
```



Завдання #5:

Значення R^2 отримую безпосередньо з моделі .score(X, Y), а для розрахунку MSE спочатку формую прогнозовані значення .predict(X) і порівнюю їх з фактичними

Модель 1:

$$Yhat1 = a + bX$$

```
mse = mean_squared_error(y_true=y, y_pred=Yhat1)
score = model_1.score(x_p, y)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")

R-square: 6.4481e-01
MSE of actual and predicted values: 2.7398e+11
```

Модель 2:

$$Yhat2 = a + bX$$

```
mse = mean_squared_error(y_true=y, y_pred=Yhat2)
score = model_2.score(x_a, y)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 3.4463e-01
MSE of actual and predicted values: 5.0552e+11
```

Модель 3:

$$Yhat3 = a + b_1 X_1 + b_2 X_2$$

```
mse = mean_squared_error(y_true=y, y_pred=Yhat3)
score = model_3.score(x, y)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 7.0918e-01
MSE of actual and predicted values: 2.2433e+11
```

Модель 4:

$$Yhat4 = a + b_1 X_1 + b_2 X_1^2$$

```
from sklearn.metrics import r2_score

mse = mean_squared_error(y_true=y, y_pred=p4(x_p))
score = r2_score(y_true=y, y_pred=p4(x_p))

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 6.4542e-01
MSE of actual and predicted values: 2.7351e+11
```

Модель 5:

$$Yhat5 = a + b_1 X_2 + b_2 X_2^2$$

```
mse = mean_squared_error(y_true=y, y_pred=p5(x_a))
score = r2_score(y_true=y, y_pred=p5(x_a))

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
```

```
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 3.7686e-01
MSE of actual and predicted values: 4.8066e+11
```

Модель 6:

$$Yhat6 = a + b_1 X_1 + b_2 X_2 + b_3 X_1 X_2 + b_4 X_1^2 + b_5 X_2^2$$

```
mse = mean_squared_error(y_true=y, y_pred=ypipe)
score = r2_score(y_true=y, y_pred=ypipe)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 9.3788e-01
MSE of actual and predicted values: 4.7920e+10
```

Висновок

Порівнюючи всі моделі, роблю висновок, що **модель #6, поліноміальна множинна регресія, - є найкращою моделлю** для прогнозування `CO2 emission` на основі нашого набору даних. Ця модель має найвищий коефіцієнт детермінації 0.93788 та найменшу середньоквадратичну похибку $4.7920 \cdot 10^{10}$.

Додаткове завдання:

1. Побудуйте кілька поліноміальних моделей різних порядків.
2. Побудуйте візуалізації для оцінки всіх моделей (зручно розміщувати всі моделі на одному графіку для обрання найкращої).
3. Порахуйте значення R^2 та MSE для оцінки якості кожної моделі (теж доцільно побудувати графік залежності R^2 або MSE від порядку поліному моделі). Оберіть найкращу модель.

1. Будую поліноміальну модель 3 та 4 порядків

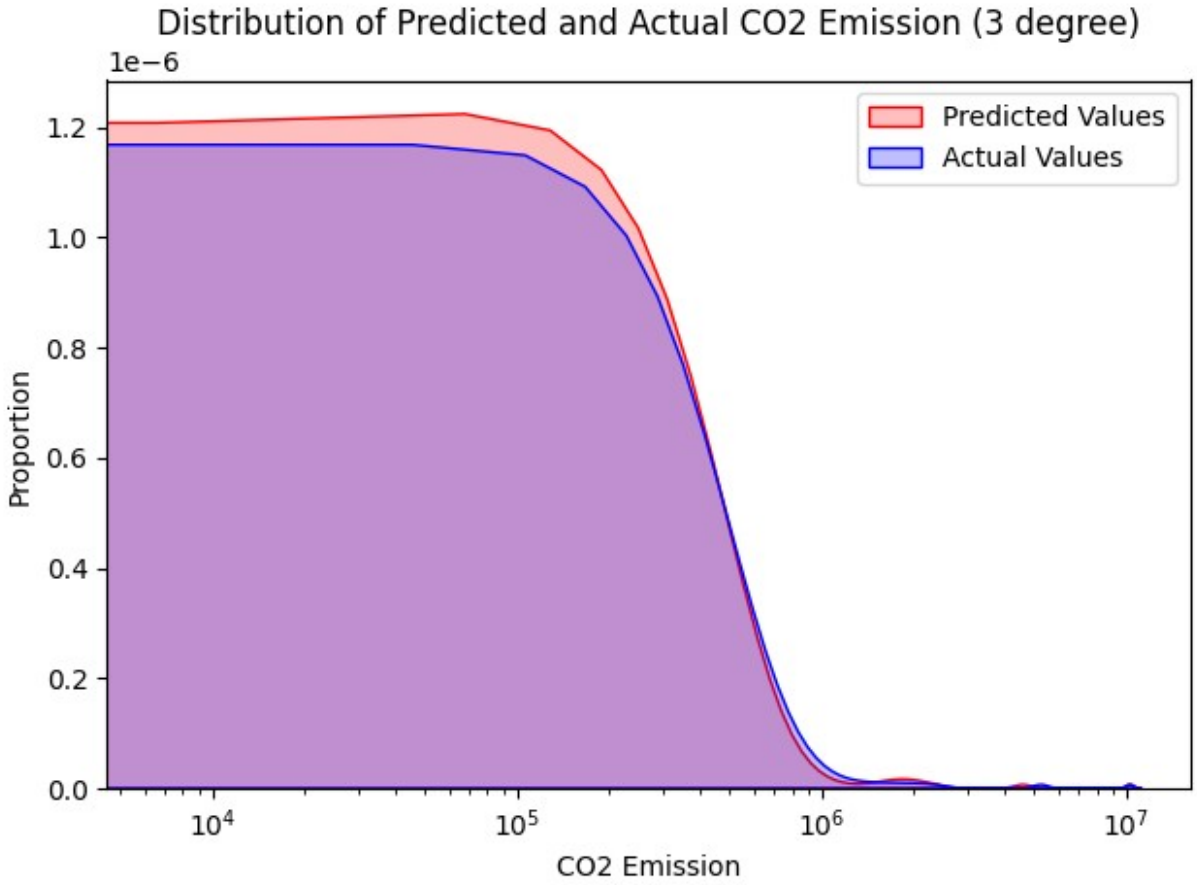
```
Input_3 = [('scale', StandardScaler()),
         ('polynomial', PolynomialFeatures(degree=3,
include_bias=False)),
         ('model', LinearRegression())]
pipe_3 = Pipeline(Input_3)
pipe_3.fit(x, y)

Input_4 = [('scale', StandardScaler()),
         ('polynomial', PolynomialFeatures(degree=4,
include_bias=False)),
         ('model', LinearRegression())]
```

```
pipe_4 = Pipeline(Input_4)
pipe_4.fit(x, y)

ypipe_3 = pipe_3.predict(x)
ypipe_4 = pipe.predict(x)
```

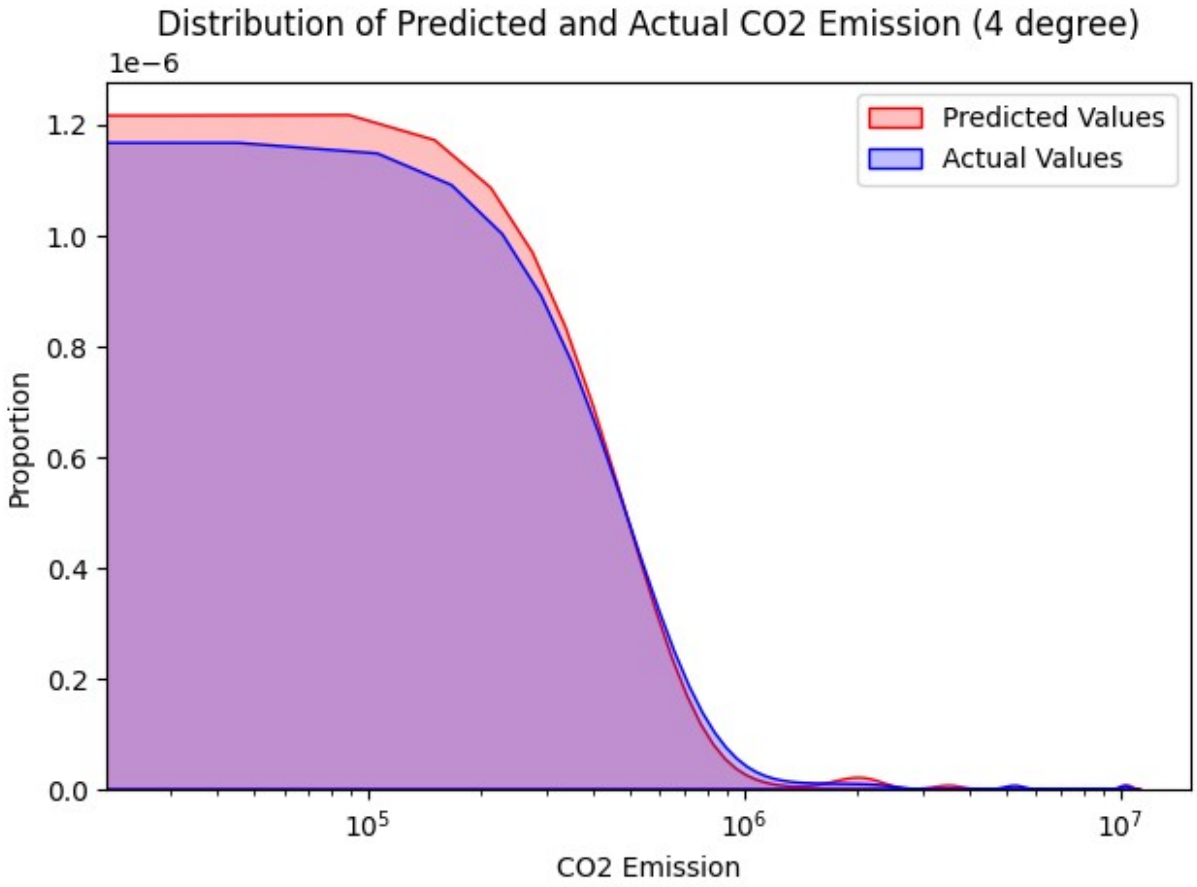
1. Створюю візуалізацію для оцінки всіх моделей

Діаграми розподілу спостережень

```
sns.kdeplot(x=ypipe_3, fill=True, color='r', label='Predicted Values')
sns.kdeplot(x=df['CO2 emission'], fill=True, color='b', label='Actual
Values')

plt.title('Distribution of Predicted and Actual CO2 Emission (3
degree)')
plt.xlabel('CO2 Emission')
plt.ylabel('Proportion')
plt.legend()
plt.xscale('log')
plt.tight_layout()
```

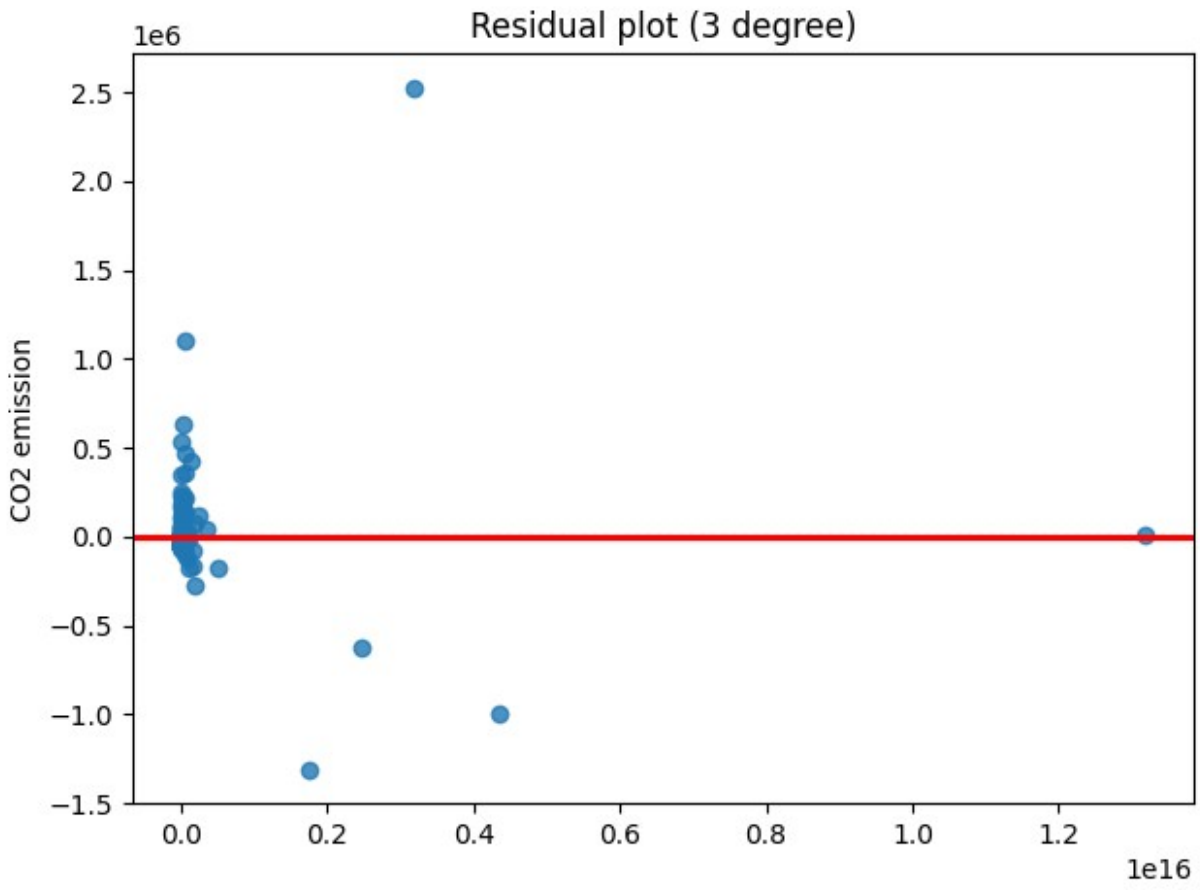


```
sns.kdeplot(x=ypipe_4, fill=True, color='r', label='Predicted Values')
sns.kdeplot(x=df['CO2 emission'], fill=True, color='b', label='Actual
Values')

plt.title('Distribution of Predicted and Actual CO2 Emission (4
degree)')
plt.xlabel('CO2 Emission')
plt.ylabel('Proportion')
plt.legend()
plt.xscale('log')
plt.tight_layout()
```

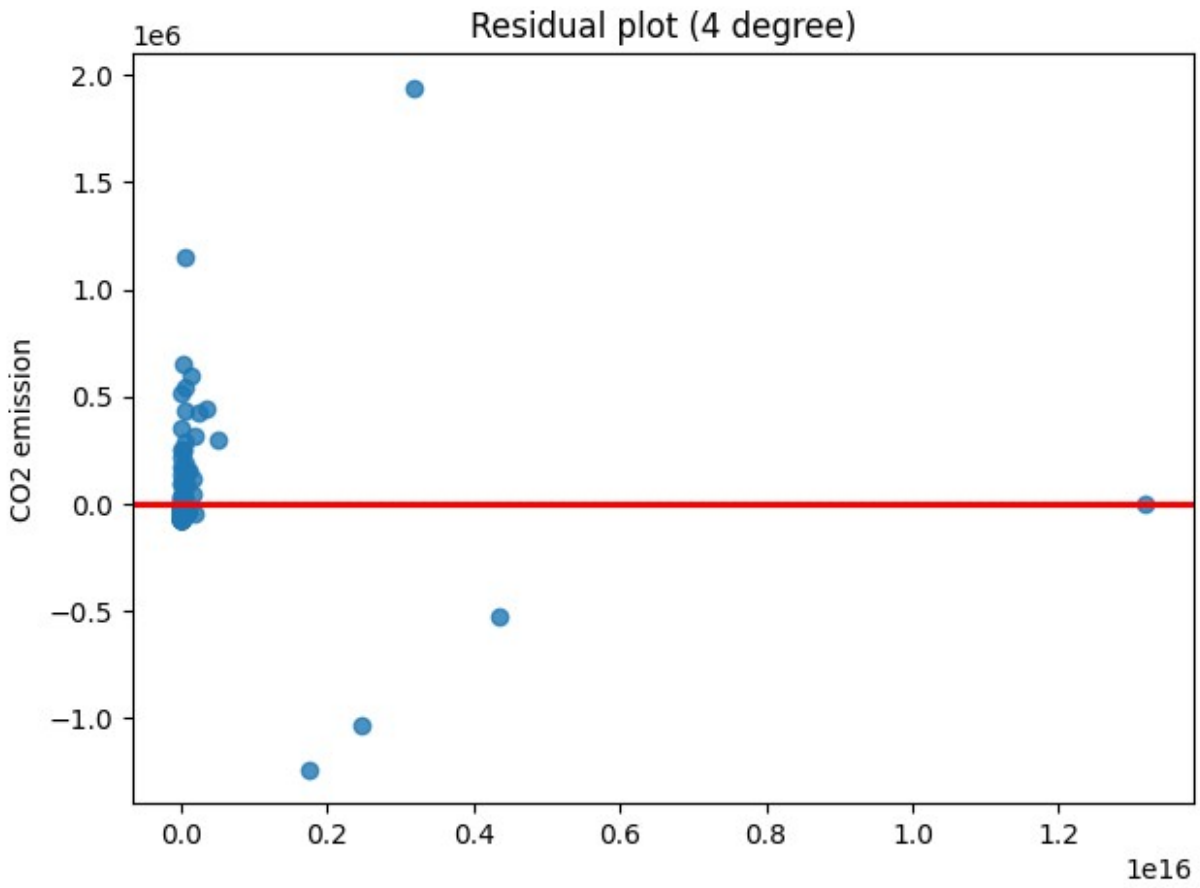


Діаграми залишків

```
sns.residplot(x=df['Area'] * df['Population'], y=y, lowess=True,
line_kws={'color': 'red'}, order=3)
plt.title('Residual plot (3 degree)')
plt.tight_layout()
```




```
sns.residplot(x=df['Area'] * df['Population'], y=y, lowess=True,
line_kws={'color': 'red'}, order=4)
plt.title('Residual plot (4 degree)')
plt.tight_layout()
```



1. Оцінювання моделей

Модель **третього** порядку

```
mse = mean_squared_error(y_true=y, y_pred=ypipe_3)
score = r2_score(y_true=y, y_pred=ypipe_3)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 9.5912e-01
MSE of actual and predicted values: 3.1535e+10
```

Модель **четвертого** порядку

```
mse = mean_squared_error(y_true=y, y_pred=ypipe_4)
score = r2_score(y_true=y, y_pred=ypipe_4)

print(f"R-square: {colored(f'{score:.4e}', 'green')}")
print(f"MSE of actual and predicted values: {colored(f'{mse:.4e}',
'blue')}")
```

```
R-square: 9.8479e-01
MSE of actual and predicted values: 1.1735e+10
```

Можна зробити висновки, що множинна поліноміальна модель четвертого порядку є найкращим представником серед усіх інших моделей. З коефіцієнтом детермінації більше ніж 0.985 та середньоквадратичною похибкою $1.173 \cdot 10^{10}$

Виконав студент групи ІП-24

Піддубний Борис