```python
# 0. Przygotowanie danych

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import probplot
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error

# Wczytanie danych
df = pd.read_csv('procesory.csv')
print(df.to_string())
```

```
    Marka             Model  HT  Rdzenie  Wątki  Cache L1  Cache L2
Cache L3  Zegar bazowy (GHz)  Zegar boost (GHz)
0   Intel     Core i5-9600K   0        6      6       384       1.5
9                 3.7                4.6
1   Intel     Core i7-9700K   0        8      8       512       2.0
12                3.6                4.9
2   Intel     Core i9-9900K   1        8     16       512       2.0
16                3.6                5.0
3   Intel    Core i5-10600K   1        6     12       384       1.5
12                4.1                4.8
4   Intel    Core i7-10700K   1        8     16       512       2.0
16                3.8                5.1
5   Intel    Core i9-10900K   1       10     20       640       2.5
20                3.7                5.3
6   Intel    Core i5-11600K   1        6     12       384       3.0
12                3.9                4.9
7   Intel    Core i7-11700K   1        8     16       512       4.0
16                3.6                5.0
8   Intel    Core i9-11900K   1        8     16       512       4.0
16                3.5                5.3
9   Intel    Core i5-12600K   1       10     16       640       9.5
20                3.7                4.9
10  Intel    Core i7-12700K   1       12     20       768      12.0
25                3.6                5.0
11  Intel    Core i9-12900K   1       16     24      1250      14.0
30                3.2                5.2
12  Intel    Core i5-13600K   1       14     20      1000      20.0
24                3.5                5.1
13  Intel    Core i7-13700K   1       16     24      1250      24.0
30                3.4                5.3
14  Intel    Core i9-13900K   1       24     32      2000      32.0
36                3.0                5.4
15  Intel    Core i5-14600K   1       14     20      1000      20.0
```

```
24                  3.5                 5.3
16  Intel    Core i7-14700K   1       20     28       1500       25.0
35                  3.4                 5.5
17  Intel    Core i9-14900K   1       24     32       2000       32.0
36                  3.2                 5.7
18    AMD     Ryzen 5 1600X   1        6     12        576        3.0
16                  3.6                 4.0
19    AMD     Ryzen 7 1700X   1        8     16        768        4.0
16                  3.4                 3.8
20    AMD     Ryzen 5 2600X   1        6     12        384        3.0
16                  3.6                 4.2
21    AMD     Ryzen 7 2700X   1        8     16        576        4.0
20                  3.7                 4.3
22    AMD     Ryzen 5 3600X   1        6     12        384        3.0
32                  3.8                 4.4
23    AMD     Ryzen 7 3700X   1        8     16        512        4.0
32                  3.6                 4.4
24    AMD     Ryzen 9 3900X   1       12     24        768        6.0
70                  3.8                 4.6
25    AMD     Ryzen 5 5600X   1        6     12        384        3.0
32                  3.7                 4.6
26    AMD   Ryzen 7 5700X3D   1        8     16        512        4.0
96                  3.4                 4.6
27    AMD     Ryzen 7 5800X   1        8     16        512        4.0
32                  3.8                 4.7
28    AMD   Ryzen 7 5800X3D   1        8     16        512        4.0
96                  3.4                 4.5
29    AMD     Ryzen 9 5900X   1       12     24        768        6.0
70                  3.7                 4.8
30    AMD     Ryzen 9 5950X   1       16     32       1000        8.0
64                  3.4                 4.9
31    AMD     Ryzen 5 7600X   1        6     12        384        3.0
32                  4.3                 5.0
32    AMD     Ryzen 7 7700X   1        8     16        512        4.0
40                  4.5                 5.2
33    AMD   Ryzen 7 7800X3D   1        8     16        512        4.0
96                  4.2                 5.0
34    AMD     Ryzen 9 7900X   1       12     24       1000        6.0
64                  4.7                 5.6
35    AMD   Ryzen 9 7900X3D   1       12     24       1000        6.0
128                  4.4                 5.6
36    AMD     Ryzen 9 7950X   1       16     32       1000        8.0
80                  4.5                 5.7
```

```
# Wybór cech numerycznych do analizy
numeryczne_kolumny = ['HT', 'Rdzenie', 'Wątki', 'Cache L1', 'Cache
L2', 'Cache L3', 'Zegar bazowy (GHz)']
df[['Cache L1', 'Cache L2', 'Cache L3']] = df[['Cache L1', 'Cache L2',
'Cache L3']].replace(' MB', '', regex=True).astype(float)
```

```
# Sprawdzenie brakujących wartości i uzupełnienie ich średnimi
if df[numeryczne_kolumny].isnull().sum().any():
    df[numeryczne_kolumny] =
df[numeryczne_kolumny].fillna(df[numeryczne_kolumny].mean())

# Definicja cech (X) i wartości docelowej (y)
X = df[numeryczne_kolumny]
y = df['Zegar boost (GHz)']

# Podział na zbiór treningowy i testowy
X_treningowe, X_testowe, y_treningowe, y_testowe = train_test_split(X,
y, test_size=0.2, random_state=42)

# Liniowa regresja
lr = LinearRegression()
lr.fit(X_treningowe, y_treningowe)
y_przewidywane_lr = lr.predict(X_testowe)
mse_lr = mean_squared_error(y_testowe, y_przewidywane_lr)

# Regresja Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X_treningowe, y_treningowe)
y_przewidywane_ridge = ridge.predict(X_testowe)
mse_ridge = mean_squared_error(y_testowe, y_przewidywane_ridge)

# Sieć neuronowa
siec_neuronowa = MLPRegressor(hidden_layer_sizes=(10,), max_iter=500,
random_state=42)
siec_neuronowa.fit(X_treningowe, y_treningowe)
y_przewidywane_siec_neuronowa = siec_neuronowa.predict(X_testowe)
mse_siec_neuronowa = mean_squared_error(y_testowe,
y_przewidywane_siec_neuronowa)

# Wyniki
print(f"Regresja liniowa MSE: {mse_lr}")
print(f"Regresja Ridge MSE: {mse_ridge}")
print(f"Sieć Neuronowa MSE: {mse_siec_neuronowa}")
```

```
Regresja liniowa MSE: 0.1372718296056184
Regresja Ridge MSE: 0.10572016293326525
Sieć Neuronowa MSE: 23.127786419331983

C:\Hubert\Programy\anaconda\Lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

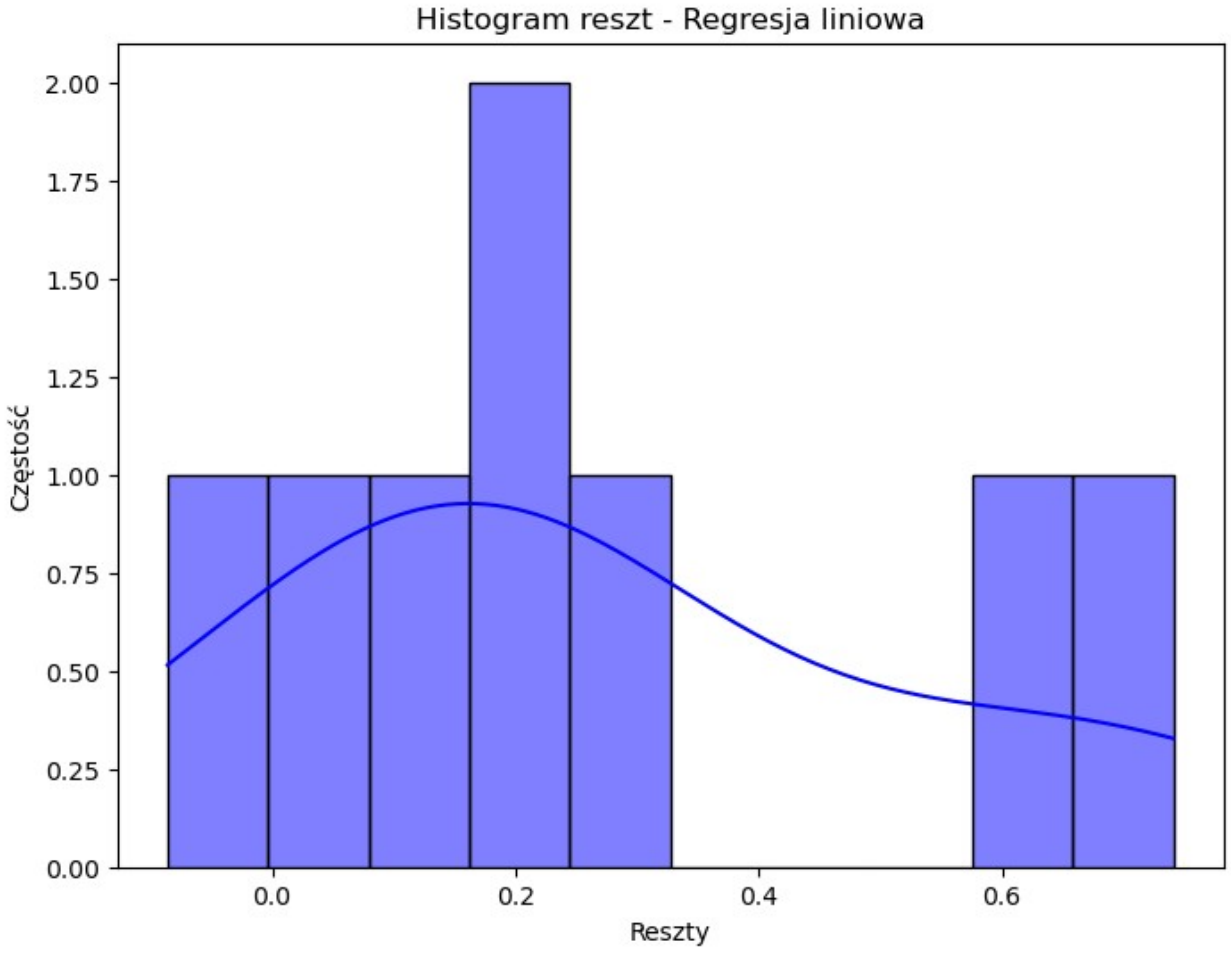
```
# 2. Zbadaj wpływ zmiennych objaśniających na predykcję (analiza
ważności cech w Ridge).
```

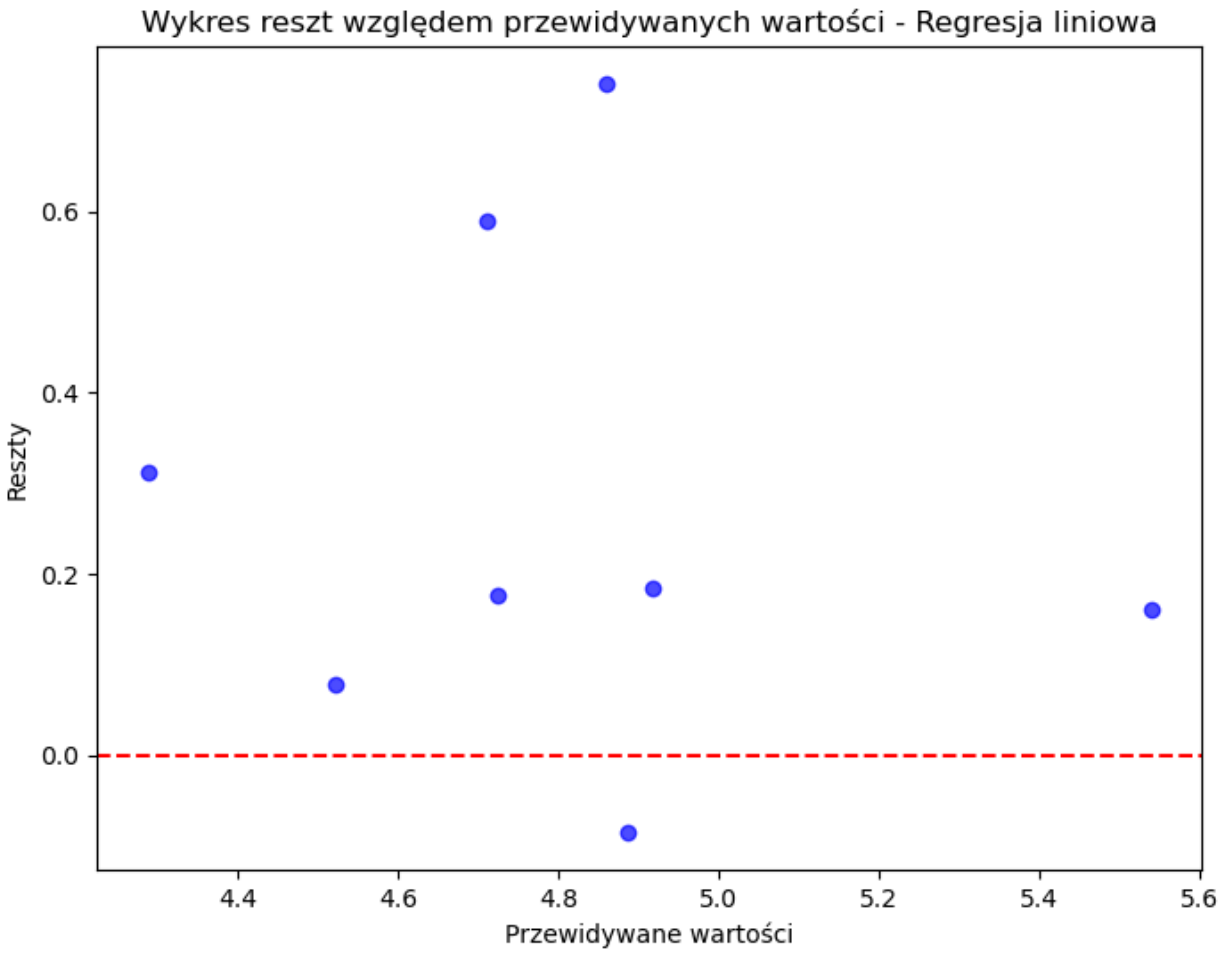
```python
from sklearn.linear_model import Ridge
import numpy as np

model_ridge = Ridge(alpha=1.0)  # Możesz dostosować wartość alpha
model_ridge.fit(X_treningowe, y_treningowe)  # Dopasowanie modelu do danych treningowych
waznosci = model_ridge.coef_  # Współczynniki ważności cech
nazwy_cech = X.columns

# Wypisanie ważności cech
for feature, waznosc in zip(nazwy_cech, waznosci):
    print(f'Cecha: {feature}, ważność: {waznosc}')  # Wyniki ważności cech

# Predykcja na zbiorze testowym
y_przewidywane_ridge = model_ridge.predict(X_testowe)
mse_ridge = np.mean((y_testowe - y_przewidywane_ridge)**2)  # Obliczenie MSE
print(f"Regresja Ridge MSE: {mse_ridge}")
```

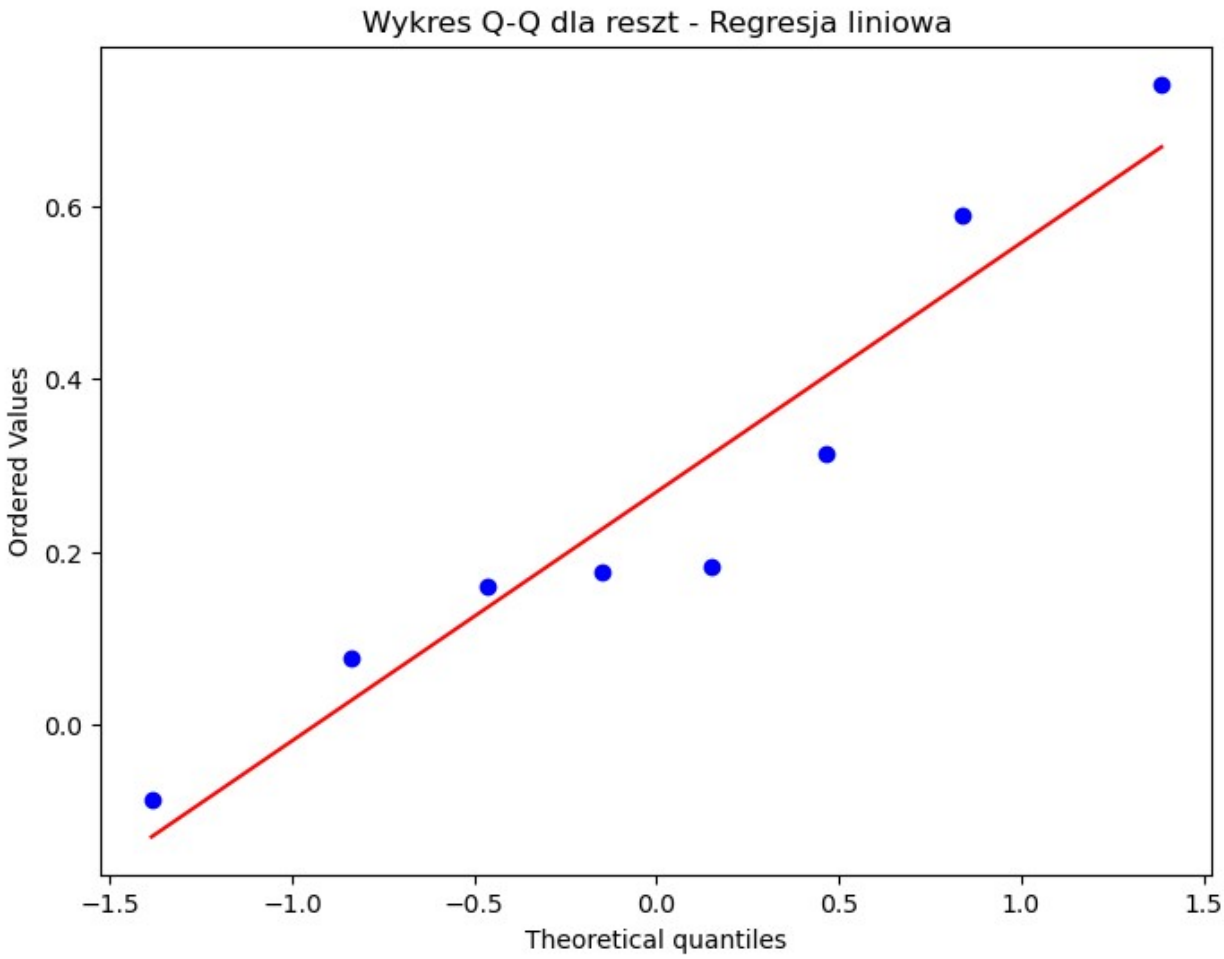
```
Cecha: HT, ważność: 0.10713334012991896
Cecha: Rdzenie, ważność: 0.26058991170363016
Cecha: Wątki, ważność: -0.03648659326375331
Cecha: Cache L1, ważność: -0.0014705934913741671
Cecha: Cache L2, ważność: -0.011273933273744265
Cecha: Cache L3, ważność: -0.002232405384124981
Cecha: Zegar bazowy (GHz), ważność: 0.5346284890367008
Regresja Ridge MSE: 0.10572016293326525
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import probplot

# 3. Wykonaj analizę reszt dla modelu regresji liniowej.

# Obliczenie reszt
reszty = y_testowe - y_przewidywane_lr

# Histogram reszt
plt.figure(figsize=(8, 6))
sns.histplot(reszty, kde=True, bins=10, color='blue')
plt.title('Histogram reszt - Regresja liniowa')
plt.xlabel('Reszty')
plt.ylabel('Częstość')
plt.show()

# Wykres reszt względem przewidywanych wartości
plt.figure(figsize=(8, 6))
plt.scatter(y_przewidywane_lr, reszty, alpha=0.7, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
```

```
plt.title('Wykres reszt względem przewidywanych wartości - Regresja
liniowa')
plt.xlabel('Przewidywane wartości')
plt.ylabel('Reszty')
plt.show()

# Normalność reszt - wykres Q-Q
plt.figure(figsize=(8, 6))
probplot(reszty, dist="norm", plot=plt)
plt.title('Wykres Q-Q dla reszt - Regresja liniowa')
plt.show()

# Średnia kwadratowa błędu (MSE) dla modelu regresji liniowej
print(f"Mean Squared Error (MSE) dla regresji liniowej: {mse_lr:.2f}")
```




Wykres reszt względem przewidywanych wartości - Regresja liniowa
Reszty
0.6
0.4
0.2
0.0
4.4
4.6
4.8
5.0
5.2
5.4
5.6
Przewidywane wartości




```
Mean Squared Error (MSE) dla regresji liniowej: 0.14
```

```
from statsmodels.stats.stattools import durbin_watson
from scipy.stats import shapiro

# 3.1. Sprawdzenie normalności reszt (Shapiro-Wilk)
shapiro_test_stat, shapiro_p_value = shapiro(reszty)
print("Test Shapiro-Wilka dla normalności reszt:")
print(f"Statystyka testowa: {shapiro_test_stat:.4f}, p-wartość:
{shapiro_p_value:.4e}")

if shapiro_p_value > 0.05:
    print("Brak podstaw do odrzucenia hipotezy zerowej: reszty są
normalnie rozłożone.")
else:
    print("Odrzucenie hipotezy zerowej: reszty nie są normalnie
rozłożone.")

# 3.2. Test autokorelacji reszt (Durbin-Watson)
durbin_watson_stat = durbin_watson(reszty)
print("\nTest Durbin-Watson:")
```

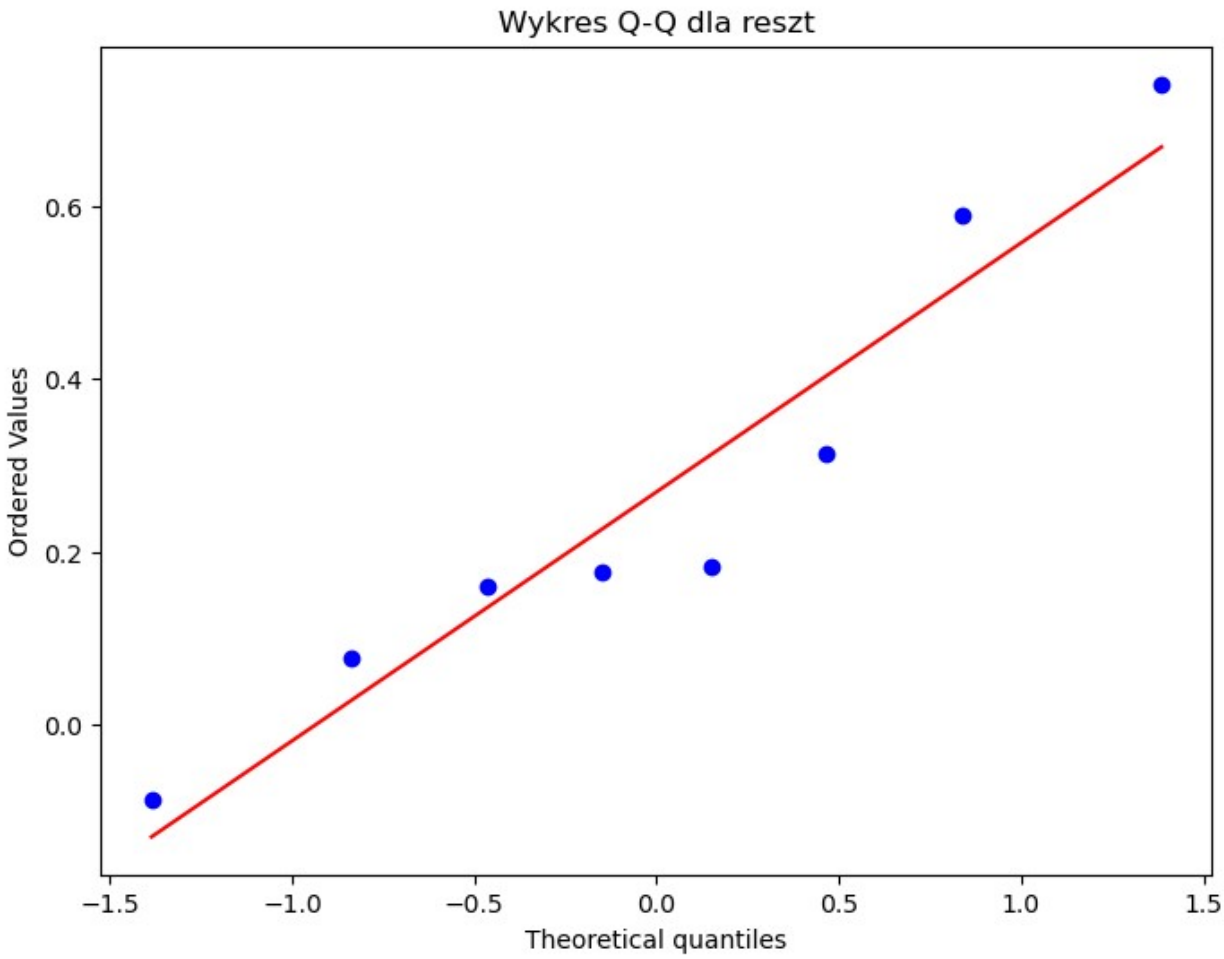
```
print(f"Statystyka Durbin-Watson: {durbin_watson_stat:.4f}")

# Interpretacja wyników testu Durbin-Watson
if durbin_watson_stat < 1.5:
    print("Wskazanie na autokorelację dodatnią reszt.")
elif durbin_watson_stat > 2.5:
    print("Wskazanie na autokorelację ujemną reszt.")
else:
    print("Brak istotnej autokorelacji reszt.")
```

```
Test Shapiro-Wilka dla normalności reszt:
Statystyka testowa: 0.9175, p-wartość: 4.1007e-01
Brak podstaw do odrzucenia hipotezy zerowej: reszty są normalnie
rozłożone.

Test Durbin-Watson:
Statystyka Durbin-Watson: 1.4322
Wskazanie na autokorelację dodatnią reszt.
```

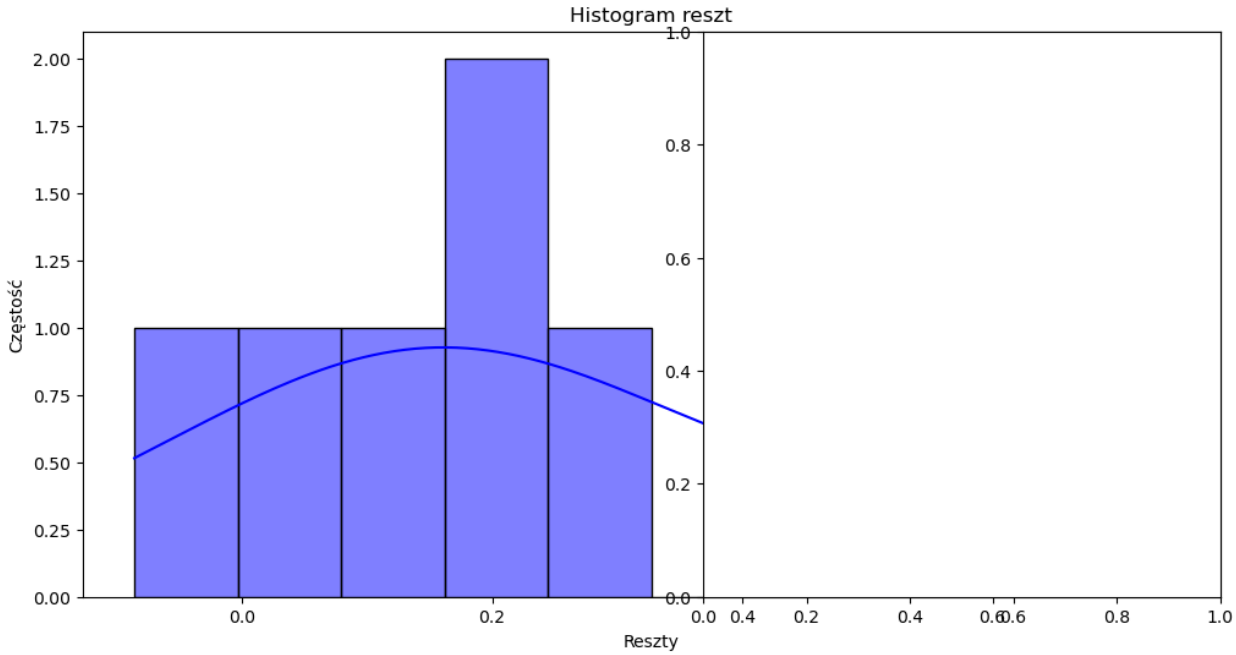
```
import matplotlib.pyplot as plt
from scipy.stats import shapiro, probplot
from statsmodels.stats.stattools import durbin_watson

# 3.1. Sprawdzenie normalności reszt (Shapiro-Wilk)
shapiro_test_stat, shapiro_p_value = shapiro(reszty)
print("Test Shapiro-Wilka dla normalności reszt:")
print(f"Statystyka testowa: {shapiro_test_stat:.4f}, p-wartość:
{shapiro_p_value:.4e}")

if shapiro_p_value > 0.05:
    print("Brak podstaw do odrzucenia hipotezy zerowej: reszty są
normalnie rozłożone.")
else:
    print("Odrzucenie hipotezy zerowej: reszty nie są normalnie
rozłożone.")

# 3.2. Test autokorelacji reszt (Durbin-Watson)
durbin_watson_stat = durbin_watson(reszty)
print("\nTest Durbin-Watson:")
print(f"Statystyka Durbin-Watson: {durbin_watson_stat:.4f}")

# Interpretacja wyników testu Durbin-Watson
if durbin_watson_stat < 1.5:
    print("Wskazanie na autokorelację dodatnią reszt.")
elif durbin_watson_stat > 2.5:
    print("Wskazanie na autokorelację ujemną reszt.")
else:
    print("Brak istotnej autokorelacji reszt.")

# 3.3. Normalność reszt - wykres Q-Q
```

```
plt.figure(figsize=(8, 6))
probplot(reszty, dist="norm", plot=plt)
plt.title('Wykres Q-Q dla reszt')
plt.show()

# 3.4. Histogram reszt oraz średnia kwadratowa błędu (MSE)
plt.figure(figsize=(12, 6))

# Histogram reszt
sns.histplot(reszty, kde=True, bins=10, color='blue')
plt.title('Histogram reszt')
plt.xlabel('Reszty')
plt.ylabel('Częstość')
plt.subplot(1, 2, 2)  # Podział okna na dwie części

# Średnia kwadratowa błędu (MSE)
plt.scatter(przewidywany_y_lr, reszty, alpha=0.7, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Wykres reszt względem przewidywanych wartości')
plt.xlabel('Przewidywane wartości')
plt.ylabel('Reszty')

plt.tight_layout()
plt.show()

# 3.5. Średnia kwadratowa błędu (MSE) dla modelu regresji liniowej
print(f"Mean Squared Error (MSE) dla regresji liniowej: {mse_lr:.2f}")
```

```
Test Shapiro-Wilka dla normalności reszt:
Statystyka testowa: 0.9175, p-wartość: 4.1007e-01
Brak podstaw do odrzucenia hipotezy zerowej: reszty są normalnie
rozłożone.

Test Durbin-Watson:
Statystyka Durbin-Watson: 1.4322
Wskazanie na autokorelację dodatnią reszt.
```




```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[69], line 45
     42 plt.subplot(1, 2, 2)  # Podział okna na dwie części
     44 # Średnia kwadratowa błędu (MSE)
---> 45 plt.scatter(przewidywany_y_lr, reszty, alpha=0.7, color='blue')
     46 plt.axhline(y=0, color='red', linestyle='--')
     47 plt.title('Wykres reszt względem przewidywanych wartości')

NameError: name 'przewidywany_y_lr' is not defined
```


Histogram reszt
Częstość
Reszty
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
0.2
0.4
0.6
0.8
1.0