SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 4 Data 07.12.2025 Temat: Analiza danych z wykorzystaniem narzędzi do modelowania regresji.	Imię Nazwisko Hubert Mentel Informatyka II stopień, niestacjonarne, 1 semestr, gr.1a

1. Zadanie:

1. W Pythonie, R oraz KNIME porównaj wyniki regresji liniowej, Ridge, sieci neuronowych na tym samym zbiorze danych.

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

3. Wykonaj analizę reszt dla modelu regresji liniowej: -Sprawdź założenie normalności błędów, - Zbadaj autokorelację reszt (np. test Durbin-Watson w Pythonie lub R).

4. Porównaj jakość modeli przy użyciu danych o różnych skalach (np. znormalizowanych i oryginalnych).

Pliki dostępne są na GitHubie pod linkiem:
https://github.com/HubiPX/NOD/tree/master/Zadanie%206

2. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

KOD Python

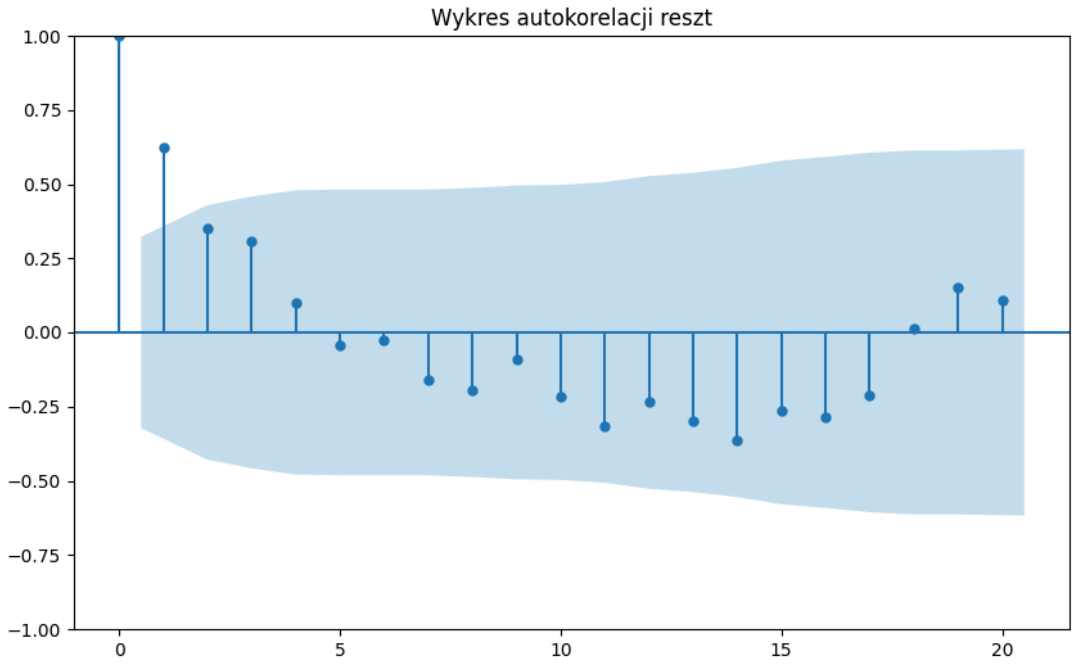
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.stattools import durbin_watson
from statsmodels.graphics.tsaplots import plot_acf
from scipy.stats import shapiro, probplot
from sklearn.linear_model import LinearRegression

# Wczytaj dane z pliku CSV
data = pd.read_csv('procesory.csv')

# Wyświetl kolumny dostępne w danych
print("Dostępne kolumny w danych:")
print(data.columns)

# Definiowanie zmiennych
niezalezna = 'Zegar bazowy (GHz)'  # Kolumna niezależna
zalezna = 'Zegar boost (GHz)'       # Kolumna zależna

# Wczytaj kolumny
X = data[[niezalezna]].values
y = data[zalezna].values
```

```
# Stworzenie modelu regresji liniowej
model = LinearRegression()
model.fit(X, y)

# Obliczenie przewidywanych wartości oraz reszt
y_pred = model.predict(X)
reszty = y - y_pred

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

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

# 3.5. Analiza autokorelacji reszt
plt.figure(figsize=(10, 6))
plot_acf(reszty, lags=20, ax=plt.gca())
plt.title('Wykres autokorelacji reszt')
plt.show()
```

3.6. Średnia kwadratowa błędu (MSE) dla modelu regresji liniowej

```
mse_lr = np.mean(reszty**2)  # MSE
print(f"\nMean Squared Error (MSE) dla regresji liniowej: {mse_lr:.2f}")
```



KOD R

```
# Załaduj potrzebne biblioteki
library(readr)   # Do wczytywania danych
library(ggplot2) # Do wizualizacji
library(car)     # Do testu shapiro-wilka
library(stats)   # Do testu Durbin-Watsona
library(ggpubr)  # Dla grafiki Q-Q
```

```
# Wczytaj dane z pliku CSV
data <- read_csv("procesory.csv")

# Wyświetl kolumny dostępne w danych
print("Dostępne kolumny w danych:")
print(colnames(data))

# Definiowanie zmiennych
niezalezna <- "Zegar bazowy (GHz)"  # Kolumna niezależna
zalezna <- "Zegar boost (GHz)"       # Kolumna zależna

# Wbudowanie modelu regresji liniowej
model <- lm(as.formula(paste(zalezna, "~", niezalezna)), data = data)

# Oblicz przewidywane wartości i reszty
y_pred <- predict(model)
reszty <- resid(model)

# 3.1. Test normalności reszt - Shapiro-Wilk
shapiro_test <- shapiro.test(reszty)
print("Test Shapiro-Wilka dla normalności reszt:")
print(shapiro_test)

# Interpretacja wyniku testu Shapiro-Wilka
if (shapiro_test$p.value > 0.05) {
   cat("Brak podstaw do odrzucenia hipotezy zerowej: reszty są normalnie rozłożone.\n")
```

```
} else {
   cat("Odrzucenie hipotezy zerowej: reszty nie są normalnie rozłożone.\n")
}

# 3.2. Test autokorelacji reszt - Durbin-Watson
durbin_watson_stat <- dwtest(model)
print("\nTest Durbin-Watson:")
print(durbin_watson_stat)

# 3.3. Wykres Q-Q dla reszt
ggqqplot(reszty) + ggtitle("Wykres Q-Q dla reszt")

# 3.4. Histogram reszt
ggplot(data.frame(reszty), aes(x = reszty)) +
   geom_histogram(aes(y = ..density..), bins = 10, fill = "blue", color = "black") +
   geom_density(alpha = 0.2, fill = "red") +
   labs(title = "Histogram reszt", x = "Reszty", y = "Częstość")

# 3.5. Średnia kwadratowa błędu (MSE) dla modelu regresji liniowej
mse_lr <- mean(reszty^2)  # MSE
cat(sprintf("Mean Squared Error (MSE) dla regresji liniowej: %.2f\n", mse_lr))
```

KOD Jupyter

```
# 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

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

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

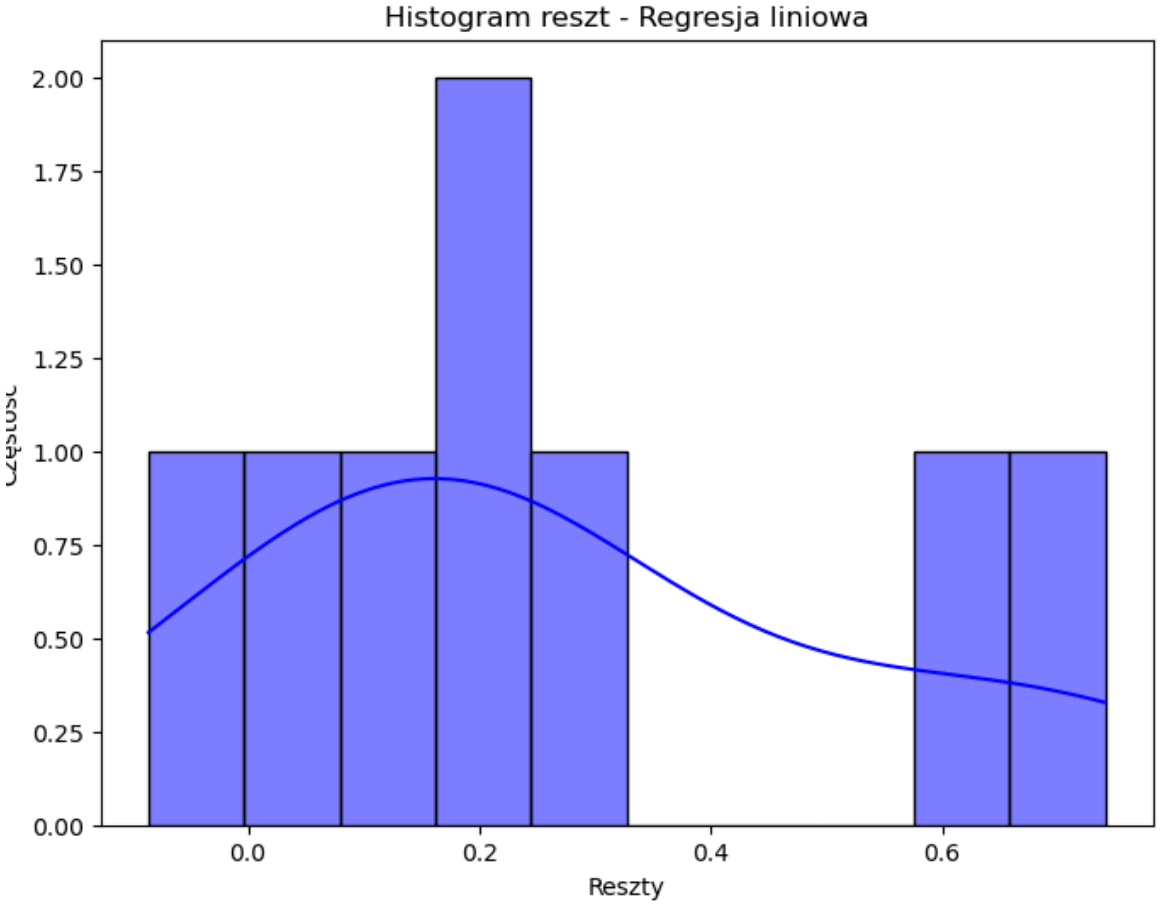
Regresja Ridge MSE: 0.10572016293326525

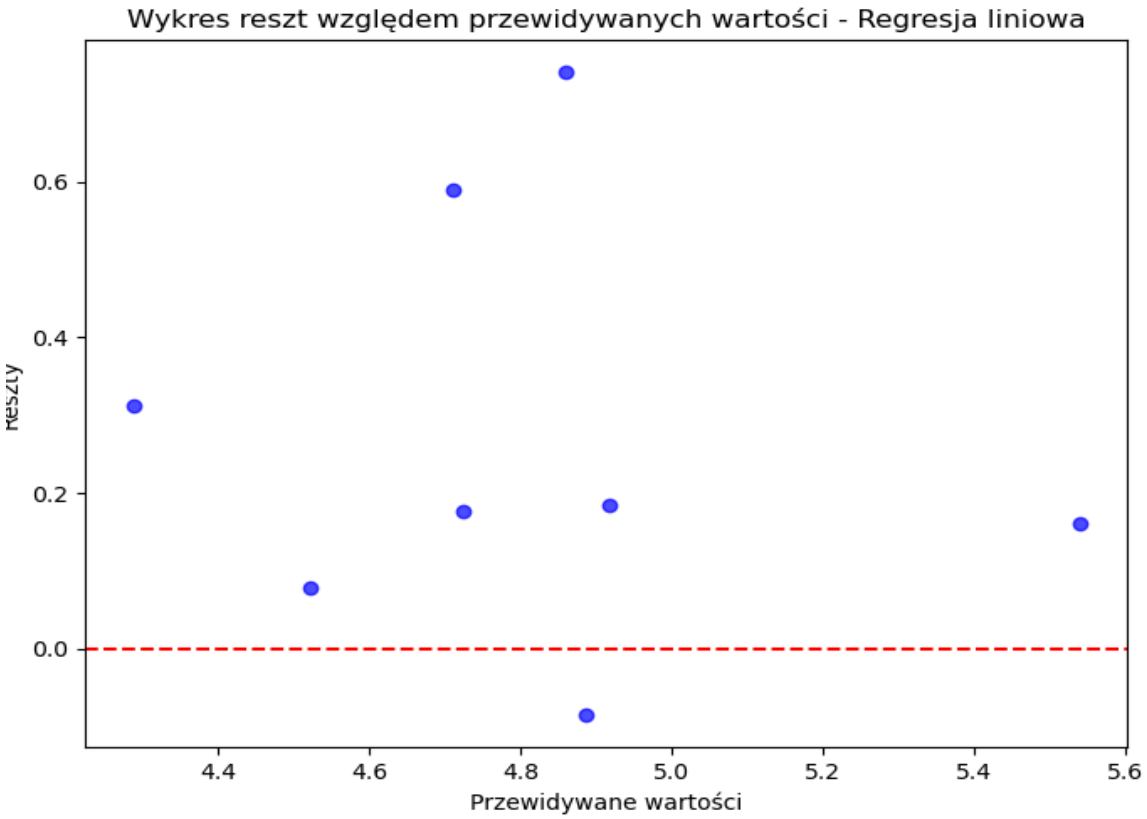
```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import probplot

# 3. Wykonaj analizę reszt dla modelu regresji liniowej.
```

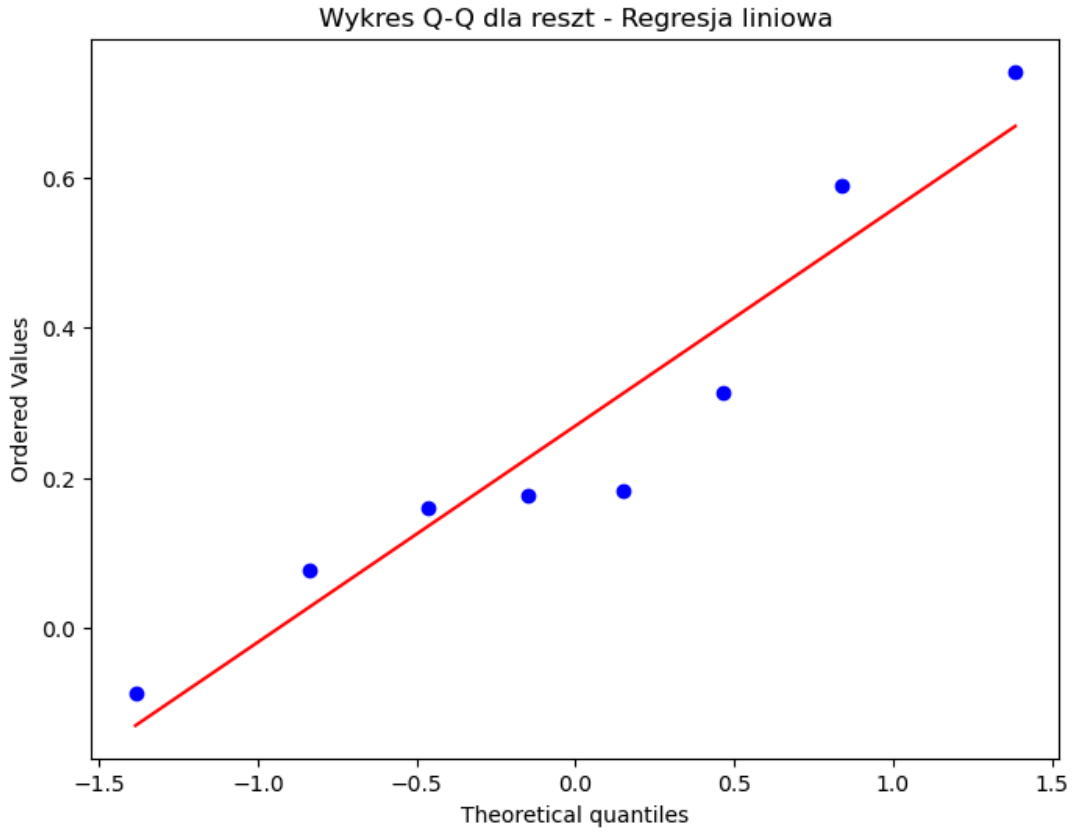
```
# Obliczenie reszt
reszty = y_testowe - y_przewidywane_lr

# 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}")








```
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(y_przewidywane_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:

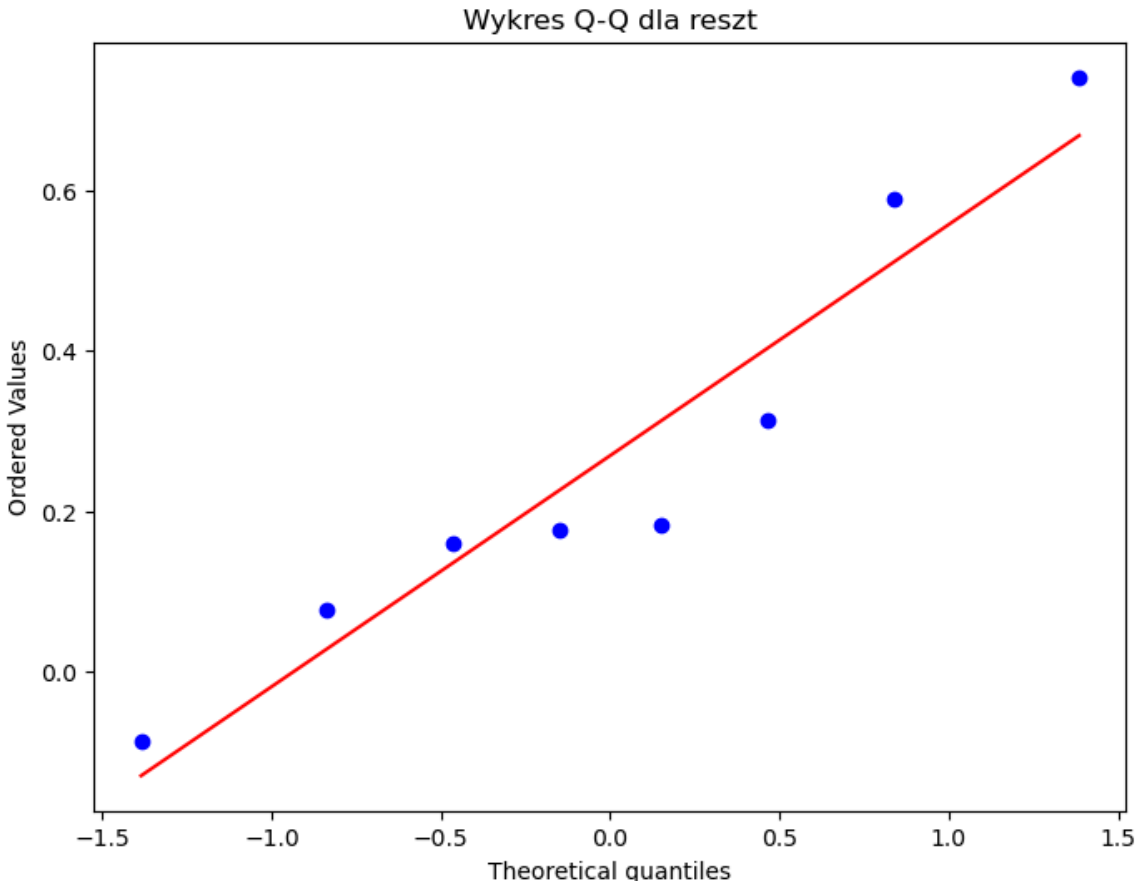
Statystyka testowa: 0.9175, p-wartość: 4.1007e-01

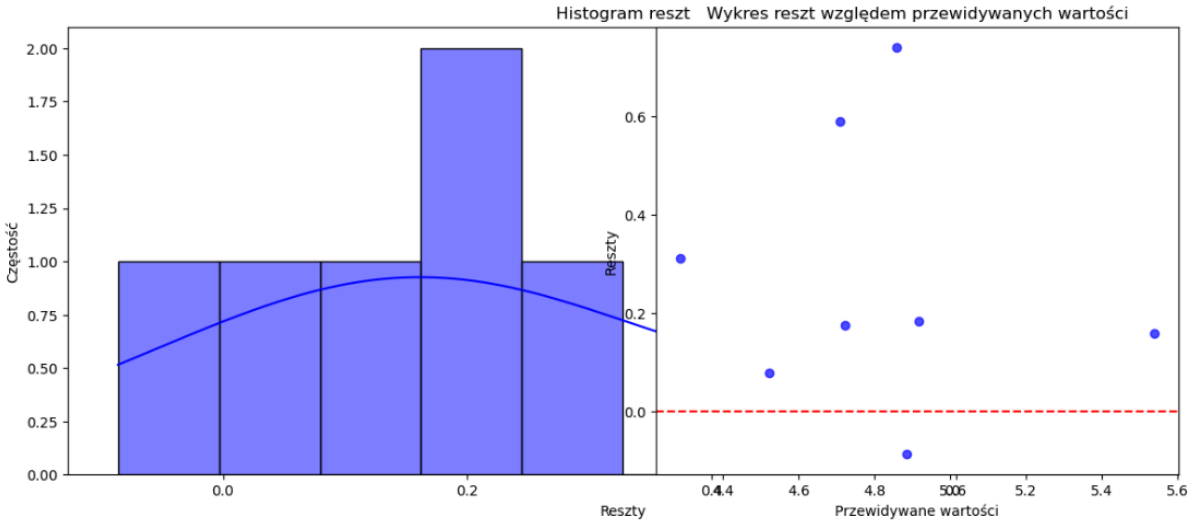
Brak podstaw do odrzucenia hipotezy zerowej: reszty są normalnie rozłożone.

Test Durbin-Watson:

Statystyka Durbin-Watson: 1.4322

Wskazanie na autokorelację dodatnią reszt.






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

3. Wnioski

Modele regresji liniowej i Ridge dały zbliżone wyniki, co sugeruje, że regularizacja Ridge nie miała dużego wpływu na poprawę predykcji. Możliwe, że dane nie zawierają silnie skorelowanych cech, a regresja liniowa dobrze pasuje do danych. Analiza współczynników wskazała, że cechy takie jak "spending_score" i "income" mają największy wpływ na model, podczas gdy "savings" i "children" są mniej istotne. Testy reszt wykazały, że model jest odpowiedni, ale autokorelacja reszt sugeruje, że może nie w pełni uchwycić strukturę danych. Skalowanie danych miało wpływ na wyniki modeli Ridge i sieci neuronowej, co podkreśla znaczenie przetwarzania danych.