

## SPRAWOZDANIE

Zajęcia: Nauka o danych I

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 6 Data 07.12.2024 Temat: „Analiza danych z wykorzystaniem narzędzi do modelowania regresji” Wariant 10	Anna Więzik Informatyka II stopień, niestacjonarne, 1 semestr, gr.1b
---	---

### 1. Polecenie:

Premise Child Health COVID-19 Health Services Disruption Survey 2020

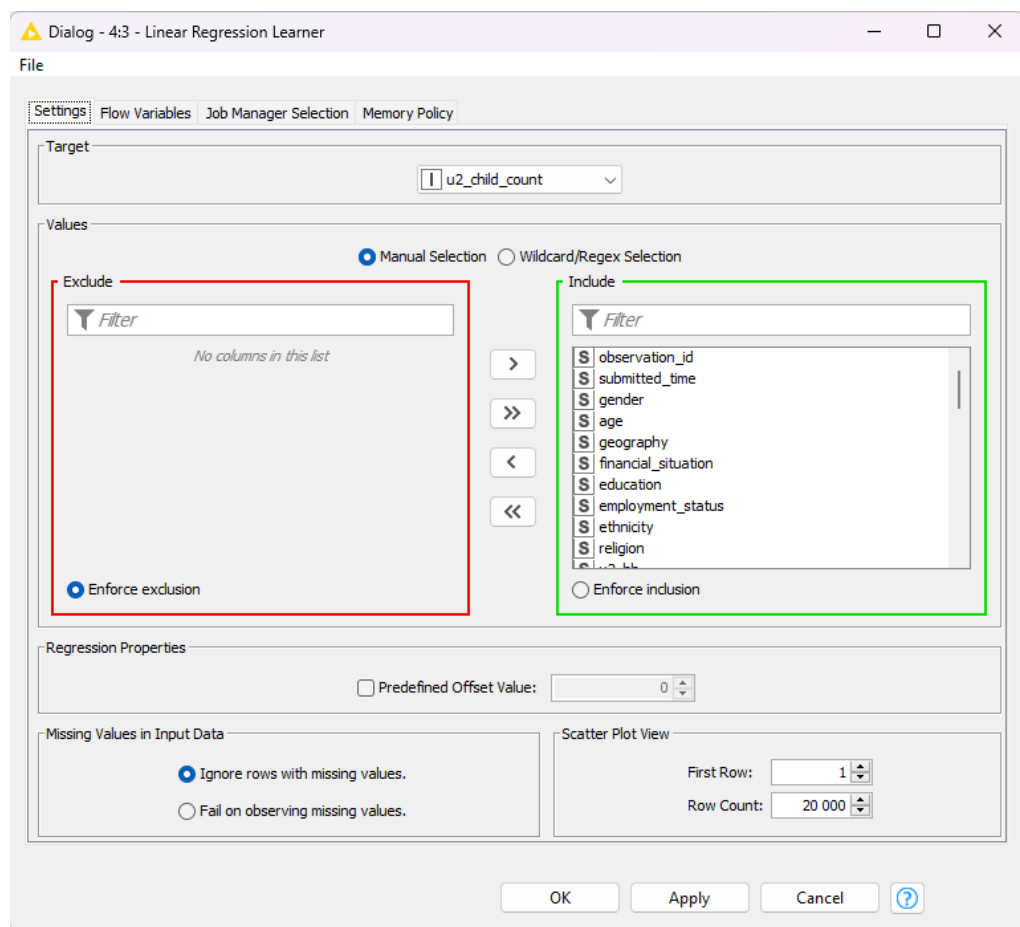
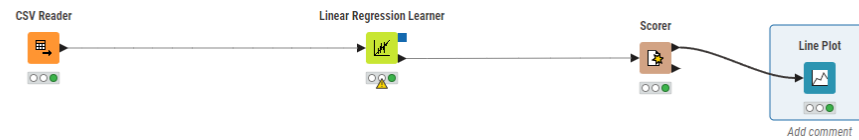
<http://ghdx.healthdata.org/record/ihme-data/premise-child-health-covid-19-health-services-disruption-survey-2020>

Link do repozytorium: [https://github.com/AnaShiro/NoD1\\_2024](https://github.com/AnaShiro/NoD1_2024)

## 2. Opis programu opracowanego

W Pythonie oraz KNIME porównaj wyniki regresji liniowej i Ridge na tym samym zbiorze danych.

- KNIME



Dialog - 4:4 - Scorer

File

Scorer

Flow Variables

Job Manager Selection

Memory Policy

First Column

S Variable

Second Column

S Variable

Sorting of values in tables

Sorting strategy: Insertion order

Reverse order

Provide scores as flow variables

Use name prefix

Missing values

In case of missing values: Ignore

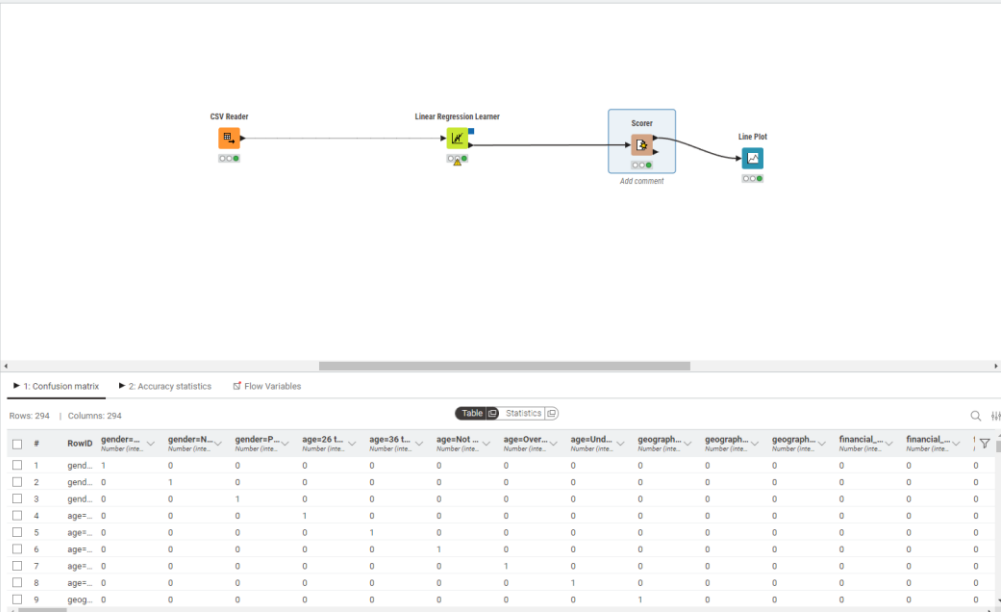
Fail

OK

Apply

Cancel

?





- Python

```
from sklearn.datasets import load_diabetes
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

data = load_diabetes()
X, y = data.data, data.target

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

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("R^2:", r2_score(y_test, y_pred_lr))
```

[1] ✓ 2.0s

... R^2: 0.551421067415138

```
from sklearn.linear_model import Ridge
from sklearn.datasets import load_diabetes

# Załaduj dane
data = load_diabetes()
X, y = data.data, data.target

# Trenuj model Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X, y)

# Współczynniki regresji jako miara ważności cech
feature_importance = abs(ridge.coef_)
print("Ważność cech:", feature_importance)
```

[2] ✓ 0.0s

... Ważność cech: [ 29.46611189 83.15427636 306.35268015 201.62773437 5.90961437  
29.51549508 152.04028006 117.3117316 262.94429001 111.87895644]

```

from scipy.stats import shapiro
import matplotlib.pyplot as plt
import numpy as np

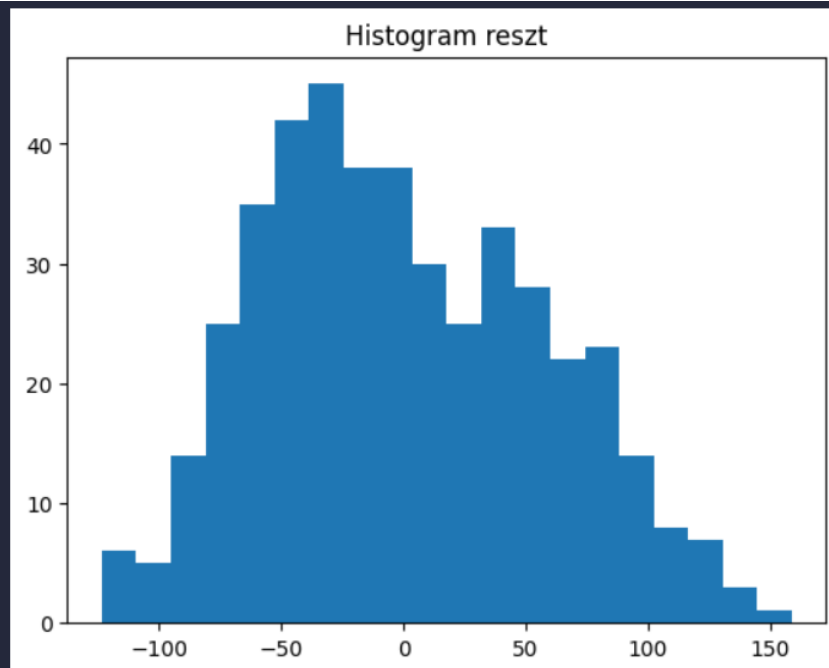
# Oblicz reszty
residuals = y - ridge.predict(X)

# Histogram reszt
plt.hist(residuals, bins=20)
plt.title("Histogram reszt")
plt.show()

# Test Shapiro-Wilka
stat, p = shapiro(residuals)
print("Statystyka Shapiro-Wilka:", stat, "P-wartość:", p)

```

[3] ✓ 1.3s



Statystyka Shapiro-Wilka: 0.982184886932373 P-wartość: 2.9706814530072734e-05

```

from statsmodels.stats.stattools import durbin_watson

# Test Durbin-Watsona
dw_stat = durbin_watson(residuals)
print("Statystyka Durbin-Watsona:", dw_stat)

```

[4] ✓ 0.8s

... Statystyka Durbin-Watsona: 1.927616883684219

- R

```
[4]: data(mtcars)
X <- as.matrix(mtcars[, -1])
y <- mtcars$mpg

model_lm <- lm(mpg ~ ., data = mtcars)
summary(model_lm)

library(glmnet)
ridge <- glmnet(X, y, alpha = 0)
lasso <- glmnet(X, y, alpha = 1)
```

Call:

lm(formula = mpg ~ ., data = mtcars)

Residuals:

	Min	1Q	Median	3Q	Max
	-3.4506	-1.6044	-0.1196	1.2193	4.6271

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	12.30337	18.71788	0.657	0.5181
cyl	-0.11144	1.04502	-0.107	0.9161
disp	0.01334	0.01786	0.747	0.4635
hp	-0.02148	0.02177	-0.987	0.3350
drat	0.78711	1.63537	0.481	0.6353
wt	-3.71530	1.89441	-1.961	0.0633
qsec	0.82104	0.73084	1.123	0.2739
vs	0.31776	2.10451	0.151	0.8814
am	2.52023	2.05665	1.225	0.2340
gear	0.65541	1.49326	0.439	0.6652
carb	-0.19942	0.82875	-0.241	0.8122

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.65 on 21 degrees of freedom

Multiple R-squared: 0.869, Adjusted R-squared: 0.8066

F-statistic: 13.93 on 10 and 21 DF, p-value: 3.793e-07

```
[2]: # Normalność błędów
model <- lm(mpg ~ ., data = mtcars)
residuals <- residuals(model)

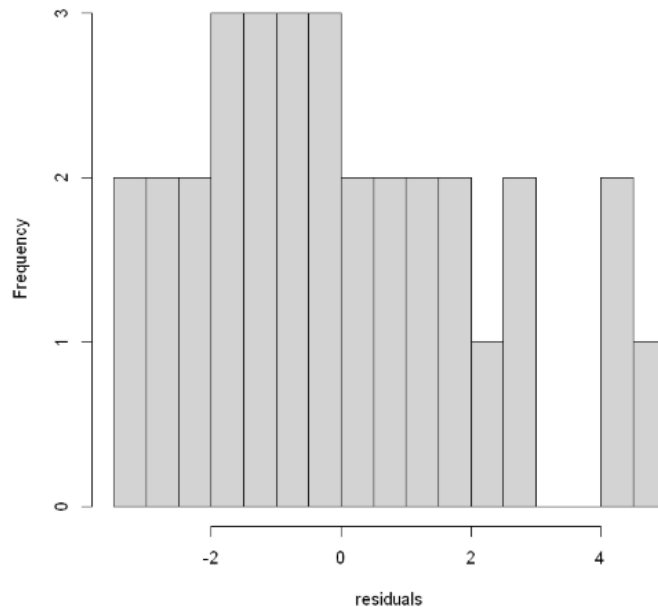
# Histogram reszt
hist(residuals, breaks = 20, main = "Histogram reszt")

# Test Shapiro-Wilka
shapiro.test(residuals)
```

#### Shapiro-Wilk normality test

```
data: residuals  
W = 0.95694, p-value = 0.2261
```

Histogram reszt



```
# Autokorelacja reszt  
library(lmtest)  
dwtest(model)
```

#### Durbin-Watson test

```
data: model  
DW = 1.8609, p-value = 0.1574  
alternative hypothesis: true autocorrelation is greater than 0
```

### 3. Wnioski

Regresja liniowa opisuje zależność zmiennej zależnej  $y$  od zmiennych niezależnych  $x_1, x_2, \dots, x_p$ . W regresji grzbietowej (Ridge) współczynniki  $\beta_j$  są regularizowane, aby podkreślić zmienne mające największy wpływ na wynik modelu. Reszty, czyli różnice między wartościami rzeczywistymi a przewidywanymi, oceniają jakość modelu. Regresja za pomocą sieci neuronowych wykorzystuje uczenie głębokie do modelowania złożonych relacji między zmiennymi. W Pythonie można to osiągnąć za pomocą bibliotek TensorFlow i PyTorch. Wizualizacja sieci neuronowych, na przykład przy użyciu `plot_model` w TensorFlow, jest kluczowa dla zrozumienia architektury modelu. Rozmiar partii (batch size) określa liczbę próbek przetwarzanych jednocześnie w jednej aktualizacji wag, epoka to jedno pełne przejście przez zestaw danych treningowych, a iteracja to pojedyncza aktualizacja wag.