Akademia Ekonomiczno-Humanistyczna w Warszawie

SPRAWOZDANIE

INTELIGENTNA ANALIZA DANYCH

LAB5

REGRESJA LINIOWA

09.01.2022

JOANNA PRAJZENDANC

36358

MIŁOSZ SAKOWSKI

36381

Spis treści

1. Cel i przebieg ćwiczenia

Celem ćwiczenia było zapoznanie się z regresją liniową oraz sposobami jej zastosowania.

2. Definicje i założenia

2.1. Wyjaśnienie pojęć

W sprawozdaniu pojawiają się następujące pojęcia:

- regresja liniowa -
- współczynnik regresji liniowej -
- metryki oceny regresji liniowej -
- metryka MAE - średnie oddalenie się przewidywanego wyniku od prawdziwej danej, które pozwala oszacować uporządkowanie zbioru, liczone według wzoru:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

- metryka MSE - uśrednione kwadraty oddalenia się przewidywanego wyniku od prawdziwej danej, liczone według wzoru:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- metryka RMSE - pierwiastek kwadratowy z metryki MSE, bardziej czuła metryka pozwalająca wykryć znacząco odstające dane w zbiorze danych

3. Metryki oceny regresji liniowej

3.1. Omówienie kodu

i. Najprostszy, przykładowy zestaw danych

```
# MAE - Mean Absolute Error       - (odleglosc1 + odleglosc2 + odleglosc3) / liczba punktów
# MSE - Mean Squared Error        - (odleglosc1^2 + odleglosc2^2 + odleglosc3^2) / liczba punktów
# RMSE - Root Mean Squared Error  - sqrt(MSE)
```

Obraz 1: Przypomnienie sposobu obliczania metryk

```python
y_true_1 = [3, 5, 3, 3, 2, 3, 3, 3, 3, 4]
y_pred_1 = [5, 3, 5, 5, 4, 5, 5, 5, 5, 6]
#MAE = ???
#MSE = ???
#RMSE = ???
print('MAE for first dataset: %.2f' % mean_absolute_error(y_true_1, y_pred_1))
print('MSE for first dataset: %.2f' % mean_squared_error(y_true_1, y_pred_1))
print('RMSE for first dataset: %.2f' % math.sqrt(mean_squared_error(y_true_1, y_pred_1)))
```

```
MAE for first dataset: 2.00
MSE for first dataset: 4.00
RMSE for first dataset: 2.00
```

Obraz 2: Obliczenie metryk dla pierwszego przykładowego zestawu danych

```python
y_true_2 = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
y_pred_2 = [4, 4, 4, 4, 4, 6, 6, 6, 6, 6]
print('MAE for second dataset: %.2f' % mean_absolute_error(y_true_2, y_pred_2))
print('MSE for second dataset: %.2f' % mean_squared_error(y_true_2, y_pred_2))
print('RMSE for second dataset: %.2f' % math.sqrt(mean_squared_error(y_true_2, y_pred_2)))
```

```
MAE for second dataset: 2.00
MSE for second dataset: 5.00
RMSE for second dataset: 2.24
```

Obraz 3: Obliczenie metryk dla drugiego przykładowego zestawu danych

```python
y_true_3 = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
y_pred_3 = [3, 3, 3, 3, 3, 3, 3, 3, 3, 23]
#MAE = ???
#MSE = ???
#RMSE = ???
print('MAE for third dataset: %.2f' % mean_absolute_error(y_true_3, y_pred_3))
print('MSE for third dataset: %.2f' % mean_squared_error(y_true_3, y_pred_3))
print('RMSE for third dataset: %.2f' % math.sqrt(mean_squared_error(y_true_3, y_pred_3)))
```

```
MAE for third dataset: 2.00
MSE for third dataset: 40.00
RMSE for third dataset: 6.32
```

Obraz 4: Obliczenie metryk dla trzeciego przykładowego zestawu danych

Można zauważyć, że metryka RMSE jest bardzo czuła na duże odchylenia danych.

ii. Regresja liniowa na przykładowym zestawie danych

```python
x = [80, 90, 100, 100, 110, 120]
x = np.reshape(x, (-1, 1))
y = [12, 9, 10, 9, 8, 6]
```

Obraz 5: Przygotowanie zestawu danych do regresji liniowej

```
regr = linear_model.LinearRegression()
regr.fit(x, y)
```

```
LinearRegression()
```

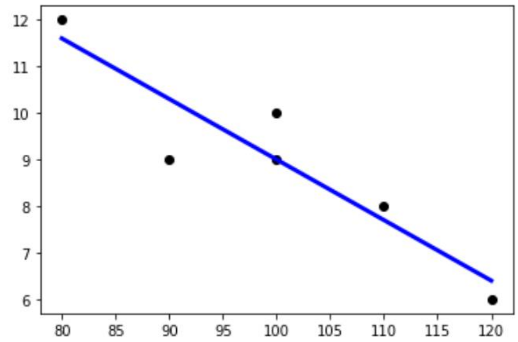
```
y_pred = regr.predict(x)
```

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(y, y_pred))
print('MSE: %.2f' % mean_squared_error(y, y_pred))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(y, y_pred)))
```

```
Coefficients:
 [-0.13]
MAE: 0.57
MSE: 0.52
RMSE: 0.72
```

Obraz 6: Regresja liniowa i obliczenie metryk

```
plt.scatter(x, y, color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.show()
```



Obraz 7: Wykres dopasowanej funkcji do zestawu danych dzięki regresji liniowej

3.2. Zadanie #1

i. Pytanie

Kiedy stosować dane metryki oceny regresji?

ii. Odpowiedź

Kiedy analizowane dane mają charakter liniowy i można dopasować do nich wykres funkcji, w omawianym przypadku funkcji liniowej.

3.3. Zadanie #2

i. Polecenie

Proszę zaproponować tak zbiór danych, żeby współczynnik regresji liniowej był dodatni

ii. Rozwiązanie

```
# Proszę zaproponować tak zbiór danych, żeby współczynnik był dodatni

x = [80, 90, 100, 100, 110, 120]
x = np.reshape(x, (-1, 1))
y = [10, 12, 9, 18, 20, 13]
```

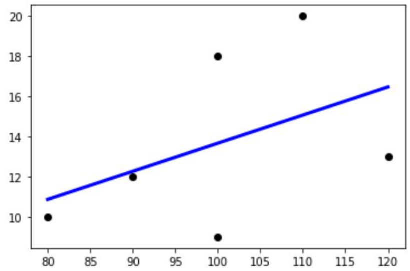
Obraz 8: Przykładowy zestaw danych

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(y, y_pred))
print('MSE: %.2f' % mean_squared_error(y, y_pred))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(y, y_pred)))
```

```
Coefficients:
 [0.14]
MAE: 3.09
MSE: 12.96
RMSE: 3.60
```

Obraz 9: Obliczenie metryk

```
plt.scatter(x, y, color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.show()
```



Obraz 10: Wykres pokazujący dodatni współczynnik regresji liniowej

3.4. Zadanie #3

i. Polecenie

Proszę zaproponować taki zbiór danych, aby błędy MAE, MSE, RMSE były równe 0

ii. Rozwiązanie

```
# Proszę zaproponować taki zbiór danych, aby błędy MAE, MSE, RMSE były równe 0

x = [80, 90, 100, 110, 120, 130]
x = np.reshape(x, (-1, 1))
y = [10, 12, 14, 16, 18, 20]
```

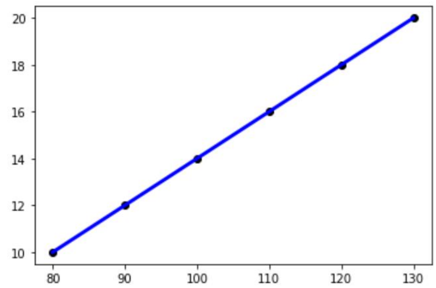
Obraz 11: Przykładowy zestaw danych

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(y, y_pred))
print('MSE: %.2f' % mean_squared_error(y, y_pred))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(y, y_pred)))
```

```
Coefficients:
 [0.2]
MAE: 0.00
MSE: 0.00
RMSE: 0.00
```

Obraz 12: Obliczenie metryk

```
plt.scatter(x, y, color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.show()
```



Obraz 13: Wykres pokazujący zerowe metryki

3.5. Zadanie #4

i. Polecenie

Proszę zaproponować taki zbiór danych, aby współczynnik MAE był niski, a RMSE - wysoki.

ii. Rozwiązanie

```
# Proszę zaproponować taki zbiór danych, aby współczynnik MAE był niski,
# a RMSE - wysoki.
```

```
x = [80, 90, 100, 110, 120, 130]
x = np.reshape(x, (-1, 1))
y = [-10, 12, -114, 1116, -8, 20]
```

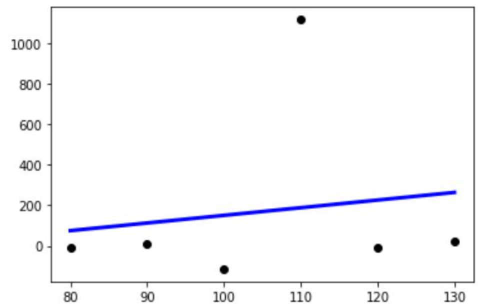
Obraz 14: Przykład zestaw danych

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(y, y_pred))
print('MSE: %.2f' % mean_squared_error(y, y_pred))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(y, y_pred)))
```

```
Coefficients:
 [3.77142857]
MAE: 309.27
MSE: 177037.65
RMSE: 420.76
```

Obraz 15: Obliczenie metryk

```
plt.scatter(x, y, color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.show()
```



Obraz 16: Wykres dla metryki RMSE dużo wyższej od MAE

3.6. Zadanie #5

i. Polecenie

Proszę zaproponować taki zbiór danych, aby wartość MAE była większa niż RMSE.

ii. Rozwiązanie

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \left| Y_i - \hat{Y}_i \right|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}$$

Jeśli MAE ma być większe od RMSE to tak naprawdę należy sprawdzić czy:

$$\left| Y_i - \hat{Y}_i \right| > \sqrt{(Y_i - \hat{Y}_i)^2}$$

$$\left| Y_i - \hat{Y}_i \right| > (Y_i - \hat{Y}_i)^{2 * \frac{1}{2}}$$

$$\left| Y_i - \hat{Y}_i \right| > Y_i - \hat{Y}_i$$

Otrzymany wzór byłby prawdą dla Y_i i $\hat{Y}_i$ mniejszych od zera; ponieważ jednak w naszych zadaniach RMSE liczymy jako pierwiastek z MSE to ostatecznie otrzymujemy tak naprawdę wartość bezwzględną z różnicy Y_i i $\hat{Y}_i$. Różnica pomiędzy MAE i RMSE wynika jedynie z niedokładności MSE.

3.7. Zadanie #6

i. Polecenie

Proszę wygenerować zbiór 30 punktów w przestrzeni 2d i powtórzyć eksperyment z regresją liniową.

ii. Rozwiązanie

A. Generowanie zbioru

```
import math
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn import linear_model
import numpy as np
import random
```

Obraz 17: Dodanie bibliotek

```
random.seed()
temp_x = random.sample(range(50), 30)
temp_y_true = random.sample(range(500), 30)
temp_x.sort()
temp_x = np.reshape(temp_x, (-1, 1))
```

Obraz 18:Wygenerowanie zestawu danych 30 punktów

```
regr = linear_model.LinearRegression()
regr.fit(temp_x, temp_y_true)
temp_y_predict = regr.predict(temp_x)
```

Obraz 19: Regresja liniowa

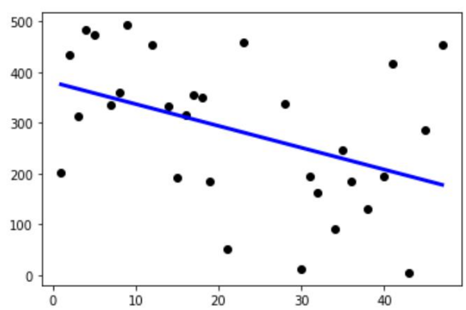
B. Wynik

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(temp_y_true, temp_y_predict))
print('MSE: %.2f' % mean_squared_error(temp_y_true, temp_y_predict))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(temp_y_true, temp_y_predict)))
```

```
Coefficients: 
 [-4.30095594]
MAE: 104.25
MSE: 16558.18
RMSE: 128.68
```

Obraz 20: Obliczenie metryk

```
plt.scatter(temp_x, temp_y_true, color='black')
plt.plot(temp_x, temp_y_predict, color='blue', linewidth=3)
plt.show()
```



Obraz 21: Wykres regresji liniowej wygenerowanych przypadkowych punktów

3.8. Zadanie #7

i. Polecenie

Proszę powtórzyć regresję liniową w przestrzeni jednowymiarowej na innym, rzeczywistym zbiorze danych.

ii. Rozwiązanie

A. Zbiór danych

Zbiór danych pobrany został ze strony: https://www.kaggle.com/maajdl/yeh-concret-data, dla pokazania przykładu regresji liniowej bez wstępnej analizy, kolumna Y została wybrana na podstawie opracowania https://www.kaggle.com/emrearslan123/concrete-compressive-strength-prediction.

```
concrete_data = pd.read_csv(
    "Concrete_Data_Yeh.csv", header=0, index_col=False)
```

Obraz 22: Import danych z pliku

```
print(concrete_data.head())
```

```
   cement   slag  flyash  water  superplasticizer  coarseaggregate  \
0   540.0    0.0     0.0  162.0               2.5           1040.0
1   540.0    0.0     0.0  162.0               2.5           1055.0
2   332.5  142.5     0.0  228.0               0.0            932.0
3   332.5  142.5     0.0  228.0               0.0            932.0
4   198.6  132.4     0.0  192.0               0.0            978.4

   fineaggregate  age  csMPa
0          676.0   28  79.99
1          676.0   28  61.89
2          594.0  270  40.27
3          594.0  365  41.05
4          825.5  360  44.30
```

Obraz 23: Podgląd danych z pliku

```
x = concrete_data['cement'].values.tolist()
x = np.reshape(x, (-1, 1))
y = concrete_data['csMPa']
```

```
regr = linear_model.LinearRegression()
regr.fit(x, y)
y_pred = regr.predict(x)
```

Obraz 24: Regresja liniowa

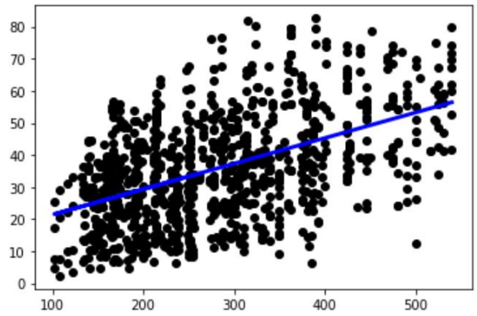
B. Wynik

```
print('Coefficients: \n', regr.coef_)
print('MAE: %.2f' % mean_absolute_error(y, y_pred))
print('MSE: %.2f' % mean_squared_error(y, y_pred))
print('RMSE: %.2f' % math.sqrt(mean_squared_error(y, y_pred)))
```

```
Coefficients:
 [0.07958034]
MAE: 11.85
MSE: 209.71
RMSE: 14.48
```

Obraz 25: Obliczenie metryk

```
plt.scatter(x, y, color='black')
plt.plot(x, y_pred, color='blue', linewidth=3)
plt.show()
```



Obraz 26: Wykres regresji liniowej wybranych danych z pliku