Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи регресії даних у машинному навчанні.

Хід роботи:

Завдання 4.1. Створення регресора однієї змінної.

Лістинг програми:

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt

input_file = "data_singlevar_regr.txt"

data = np.loadtxt(input_file, delimiter=",")
X, y = data[:, :-1], data[:, -1]

num_training = int(0.8 * len(X))
num_test = len(X) - num_training

X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
y_test_pred = regressor.predict(X_test)

plt.scatter(X_test, y_test, color="green")
plt.plot(X_test, y_test_pred, color="black", linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()
```

```
print("Linear regressor performance:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))

output_model_file = "model.pkl"
with open(output_model_file, "wb") as f:
    pickle.dump(regressor, f)

y_test_pred_new = regressor.predict(X_test)
print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
print("Linear regressor performance:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))

output_model_file = "model.pkl"
with open(output_model_file, "wb") as f:
    pickle.dump(regressor, f)

y_test_pred_new = regressor.predict(X_test)
print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

Результат виконання програми:

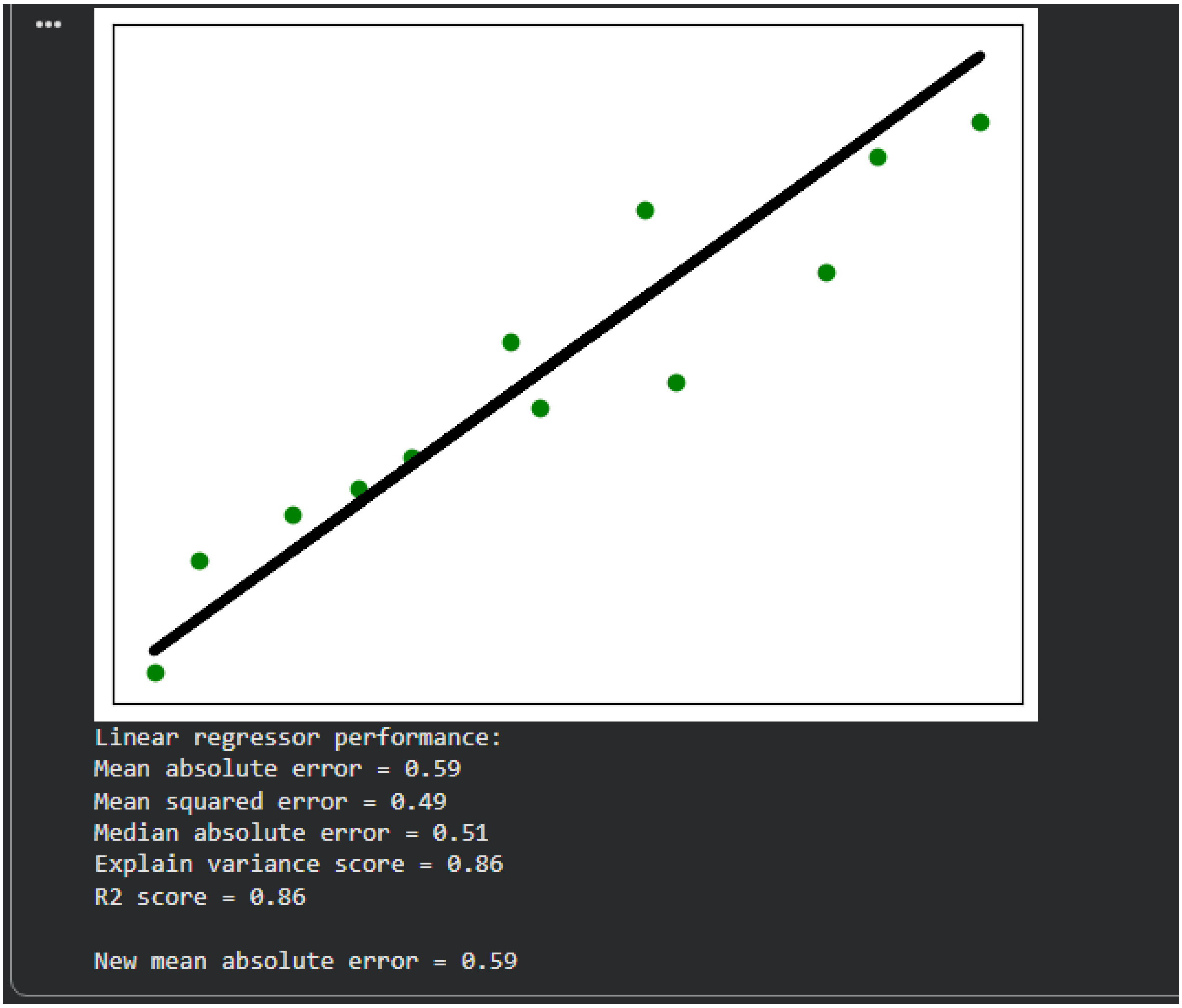


Рис. 1

У завданні було побудовано, навчено і протестовано лінійну регресійну модель. На графіку (рис. 1) відображені зелені точки (відображають реальні дані) і чорна лінія (лінія регресії, прогноз моделі). Виміряно наступні метрики:

- середня абсолютна похибка (Mean Absolute Error) ≈ 0.59, тобто в середньому прогнозовані значення відрізняються від реальних на ~ 0.59 одиниць;
- середній квадрат похибки (Mean Squared Error) ≈ 0.49, тобто великі похибки зустрічаються рідко, тому можна вважати, що модель добре підходить до даних;
- медіанна абсолютна похибка (Median Absolute Error) ≈ 0.51, тобто половина прогнозів моделі відрізняється від реальних даних на 0.51 одиниць;
- пояснена оцінка дисперсії (Explained Variance Score) ≈ 0.86 означає те, що модель добре узгоджується з даними;

- коефіцієнт детермінації (R^2 Score) ≈ 0.86 показує, що лінія регресії добре описує дані.
- New mean absolute error ≈ 0.59, дане значення збігається з MAE, тобто модель було коректно збережено у файл model.pkl і завантажено з нього.

Отже, модель має високу точність прогнозу та гарну узгодженість з даними. Лінія регресії проходить біля точок тестової вибірки, що підтверджує якість моделі.

Завдання 4.2. Передбачення за допомогою регресії однієї змінної.

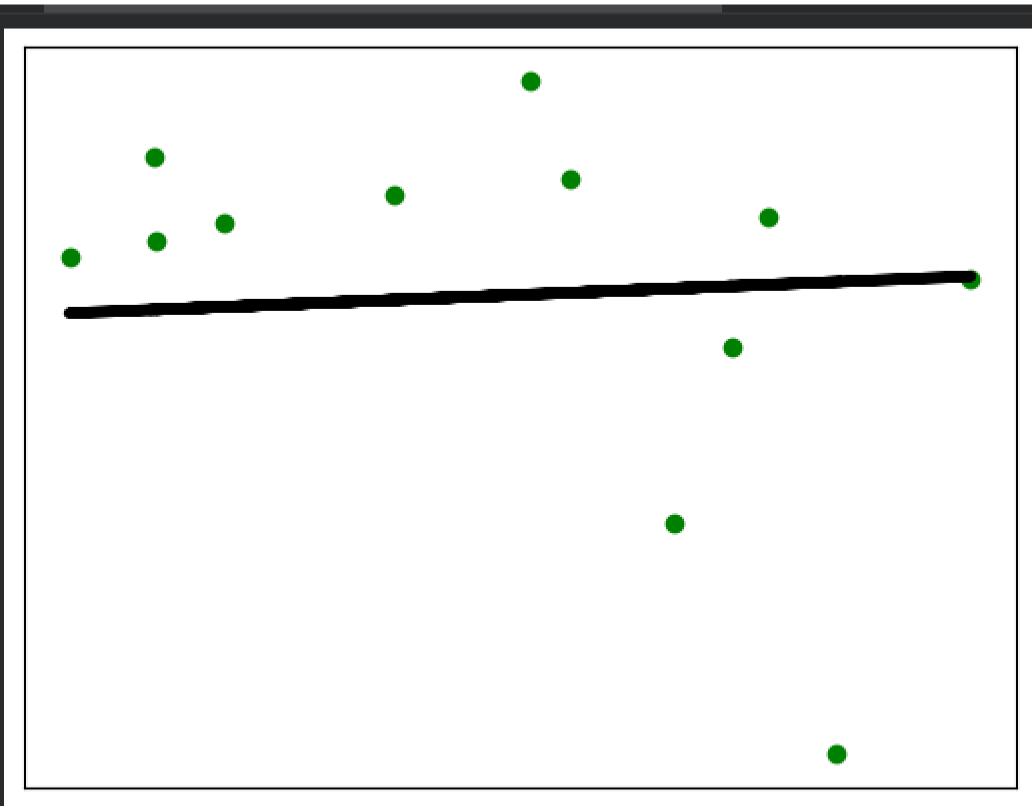
Варіант 4 файл: data_regr_4.txt

Лістинг програми:

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt

input_file = "data_regr_4.txt"

data = np.loadtxt(input_file, delimiter=",")
X, y = data[:, :-1], data[:, -1]

num_training = int(0.8 * len(X))
num_test = len(X) - num_training

X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
y_test_pred = regressor.predict(X_test)

plt.scatter(X_test, y_test, color="green")
plt.plot(X_test, y_test_pred, color="black", linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()

print("Linear regressor performance:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
```

```
print("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))

output_model_file = "model2.pkl"
with open(output_model_file, "wb") as f:
    pickle.dump(regressor, f)

y_test_pred_new = regressor.predict(X_test)
print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

Результат виконання програми:



```
Linear regressor performance:
Mean absolute error = 2.72
Mean squared error = 13.16
Median absolute error = 1.9
Explain variance score = -0.07
R2 score = -0.07

New mean absolute error = 2.72
```

Рис. 2

У завданні було побудовано, навчено і протестовано лінійну регресійну модель. На графіку (рис. 1)видно, що точки тестової вибірки хаотично розкидані і знаходяться на значній відстані від побудованої лінії регресії. Виміряні метрики показали наступні результати:

- Середня абсолютна похибка (Mean Absolute Error) ≈ 2.72. Це означає, що в середньому прогноз моделі відхиляється від реального значення на 2.72 одиниці.
- Середній квадрат похибки (Mean Squared Error) ≈ 13.16. Це значення підтверджує, що у прогнозах є суттєві відхилення.
- Медіанна абсолютна похибка (Median Absolute Error) ≈ 1.9. Це означає, що половина прогнозів моделі відхиляється від реальних даних на 1.9 одиниці або менше.
- Пояснена оцінка дисперсії (Explain Variance Score) ≈ -0.07. Від'ємне значення (близьке до 0) вказує на те, що модель абсолютно не пояснює варіацію даних.
- Коефіцієнт детермінації (R2 Score) ≈ -0.07. Від'ємний R^2 є яскравим індикатором того, що модель не просто погана, а працює гірше, ніж просте усереднення вихідних значень.
- New mean absolute error ≈ 2.72. Це значення повністю збігається з оригінальною MAE, що підтверджує коректне збереження та подальше завантаження моделі з файлу.

Отже, побудована лінійна модель має вкрай низьку якість прогнозу і є абсолютно непридатною для цього набору даних, оскільки не може вловити жодної залежності.

Завдання 4.3. Створення багатовимірного регресора.

Лістинг програми:

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures

input_file = "data_multivar_regr.txt"

data = np.loadtxt(input_file, delimiter=",")
X, y = data[:, :-1], data[:, -1]

num_training = int(0.8 * len(X))
num_test = len(X) - num_training

X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
y_test_pred = regressor.predict(X_test)

print("Linear regressor performance:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Mean squared error =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))

polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)

datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.fit_transform(datapoint)

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)

print("\nLinear regression:\n", regressor.predict(datapoint))
print("\nPolynomial regression:\n", poly_linear_model.predict(poly_datapoint))
```

Результат виконаня програми:

```
••• Linear regressor performance:
    Mean absolute error = 3.58
    Mean squared error = 20.31
    Median absolute error = 2.99
    Explain variance score = 0.86
    R2 score = 0.86

    Linear regression:
     [36.05286276]

    Polynomial regression:
     [41.08312885]
```

Рис. 3

У завданні було створено моделі лінійної та поліноміальної регресії, за допомогою яких спрогнозовано результат (рис. 3) для контрольної точки [7.75, 6.35, 5.56]. Результат лінійного регресора ≈ 36.05, в той час як поліноміальний регресор дав результат ≈ 41.08, який є набагато ближчим до очікуваного 41.35. Отже, поліноміальний регресор ступеня 10 продемонстрував вищу якість прогнозування та більш точні результати порівняно з лінійним регресором.

Завдання 4.4. Регресія багатьох змінних.

Лістинг програми:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.5,
random_state=0)

regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)
ypred = regr.predict(Xtest)

print("Коефіцієнти регресії =", np.round(regr.coef_, 2))
```

```
print("Точка перетину =", regr.intercept_)
print("R2 score =", round(r2_score(ytest, ypred), 2))
print("Mean absolute error =", round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =", round(mean_squared_error(ytest, ypred), 2))

fig, ax = plt.subplots()
ax.scatter(ytest, ypred, edgecolors=(0, 0, 0))
ax.plot([y.min(), y.max()], [y.min(), y.max()], "k--", lw=4)
ax.set_xlabel("Виміряно")
ax.set_ylabel("Передбачено")
plt.show()
```

Результат виконання програми:

```
Коефіцієнти регресії = [ -20.4  -265.89  564.65  325.56 -692.16  395.56   23.5   116.36  843.95
   12.72]
Точка перетину = 154.35892852801342
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```

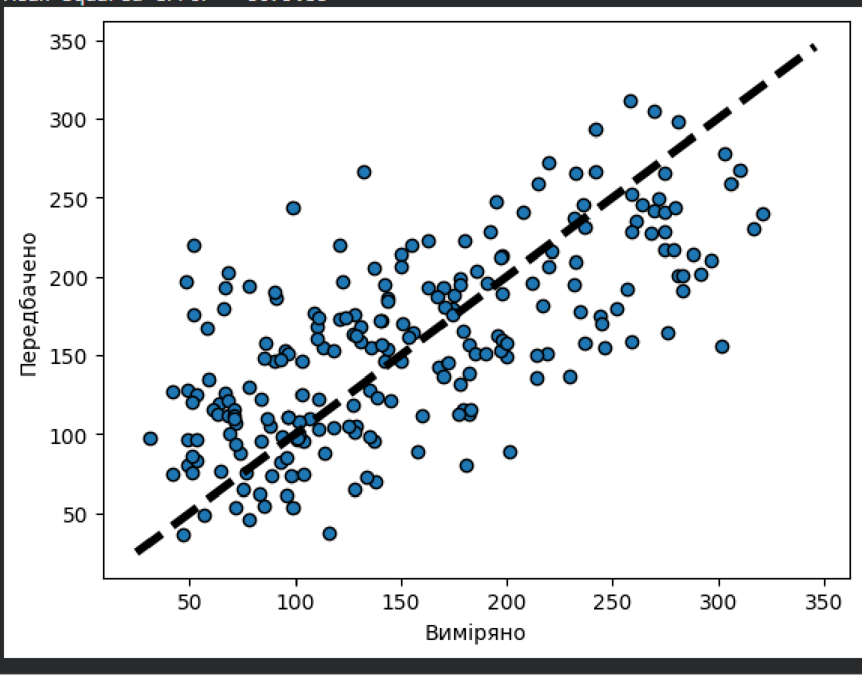


Рис. 4

У завданні було побудовано багатовимірну модель лінійної регресії з використанням набору даних про діабет. На діаграмі пунктирна лінія – це лінія, яка зображує ідеальний випадок, коли передбачення дорівнюють реальним значенням; а сині точки – це приклади з тестових даних, у яких координата x – реальне значення, а y – спрогнозоване; їх положення демонструє, наскільки добре передбачення моделі збігаються з справжніми даними. У нашому випадку більшість точок хаотично розкидані по графіку, це означає, що модель часто помиляється і не є ідеальною для передбачення прогресування захворювання. Метрики MAE, MSE та R^2 score показали посередній результат, який означає, що лінійна регресія лише частково розуміє закономірності у даних. Коефіцієнти регресії показують, як кожна ознака впливає на результат: якщо коефіцієнт позитивний, то ця збільшення цієї ознаки підвищує рівень прогресування захворювання; якщо коефіцієнт ознаки негативний, то відповідно її збільшення зменшує прогресування.

Завдання 4.5. Самостійна побудова регресії.

Варіант 9

m = 100

X = np.linspace(-3, 3, m)

y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, m)

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

def generate_data(m):
    np.random.seed(42)
    X = np.linspace(-3, 3, m).reshape(-1, 1)
    y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, (m, 1))
    return X, y

m = 100
X, y = generate_data(m)

lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_pred_lin = lin_reg.predict(X)

print("Лінійна регресія:")
```

```
print("intercept =", lin_reg.intercept_)
print("coef =", lin_reg.coef_)

print("MAE:", mean_absolute_error(y, y_pred_lin))
print("MSE:", mean_squared_error(y, y_pred_lin))
print("R2:", r2_score(y, y_pred_lin))

poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)

poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)
y_pred_poly = poly_reg.predict(X_poly)

print("\nПоліноміальна регресія 2-го ступеня:")
print("intercept =", poly_reg.intercept_)
print("coef =", poly_reg.coef_)

print("MAE:", mean_absolute_error(y, y_pred_poly))
print("MSE:", mean_squared_error(y, y_pred_poly))
print("R2:", r2_score(y, y_pred_poly))

X_sorted = X
y_pred_lin_sorted = y_pred_lin
y_pred_poly_sorted = y_pred_poly

plt.scatter(X, y, color='green', label='Дані')
plt.plot(X_sorted, y_pred_lin_sorted, color='blue', linewidth=2,
label='Лінійна регресія')
plt.plot(X_sorted, y_pred_poly_sorted, color='red', linewidth=2,
label='Поліноміальна регресія 2-го ступеня')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.title('Лінійна та поліноміальна регресії')
plt.show()
```

Результат виконання програми:

```
Лінійна регресія:
intercept = [2.97018074]
coef = [[0.34014935]]
MAE: 0.42912784039118684
MSE: 0.2788579635044964
R2: 0.5594486415622438

Поліноміальна регресія 2-го ступеня:
intercept = [2.97579626]
coef = [[ 0.34014935 -0.00183477]]
MAE: 0.4288509903286737
MSE: 0.2788327438755414
R2: 0.5594884845764284
```

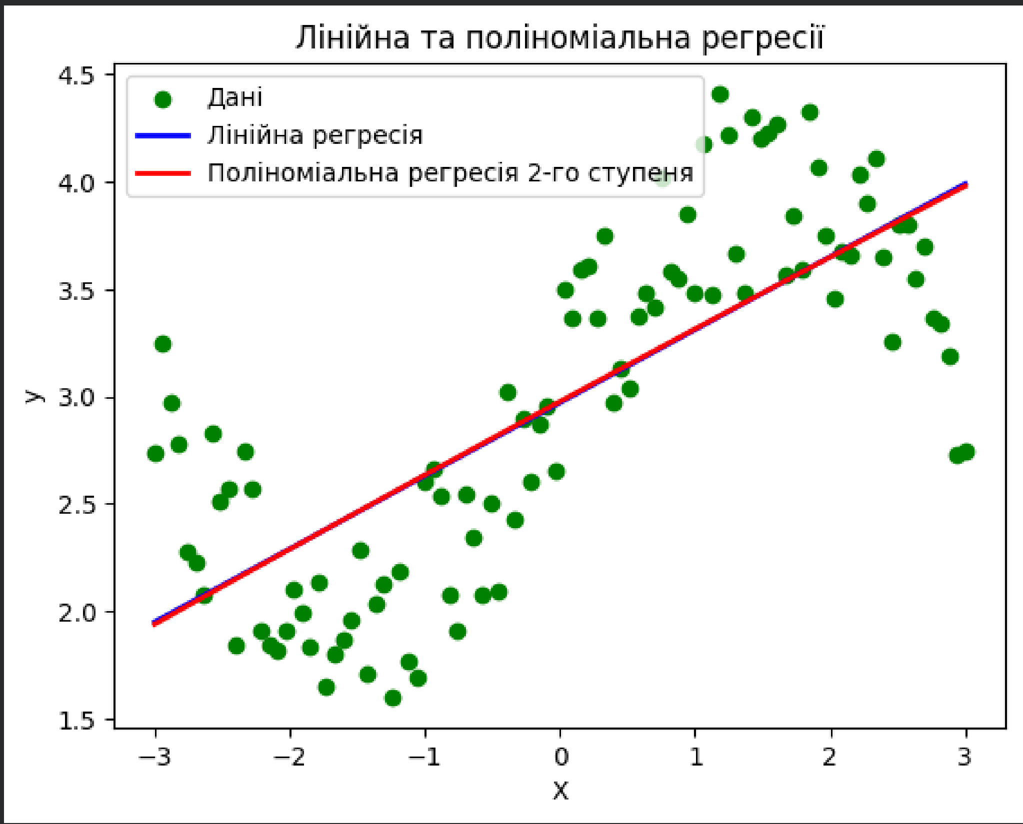


Рис. 5

У завданні було згенеровано випадковий набір даних за наданою формулою (на основі синусоїди), який використовувася для побудови моделей лінійної та поліноміальної регресій. На графіку видно, що дані мають чітку нелінійну (синусоїдальну) залежність. Лінійна регресія (синя лінія) не здатна описати цю залежність і показує лише загальний слабкий висхідний тренд. Поліноміальна регресія 2-го ступеня (червона лінія) у цьому випадку не дала жодного покращення. Її графік практично повністю збігається з лінійною моделлю, оскільки проста параболічна форма не підходить для опису синусоїдальної залежності.

Середня абсолютна похибка (MAE) ≈ 0.428, що свідчить про помітну неточність прогнозів. Коефіцієнт детермінації (R^2 Score) ≈ 0.559 (або 55.9%), тобто моделі пояснюють лише трохи більше половини варіації даних.

Для моделі поліноміальної регресії було отримано наступні коефіцієнти:

- Перетин (intercept) ≈ 2.975;
- Коефіцієнт для x ≈ 0.34;
- Коефіцієнт для x^2 ≈ -0.001;

Модель у вигляді математичного рівняння:

$y = 3 + \sin(X) + \text{шум}$

Модель регресії з передбаченими коефіцієнтами:

$y = -0.001x^2 + 0.34x + 2.975$

Отриманий коефіцієнт для x^2 виявився практично нульовим. Це підтверджує, що модель 2-го ступеня не змогла вловити нелінійність і, по суті, звелася до звичайної лінійної моделі, яка є невідповідною для цих даних.

Завдання 4.6. Побудова кривих навчання.

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
    train_errors, val_errors = [], []
```

```
    for m in range(1, len(X_train)):
        model.fit(X_train[:m], y_train[:m])
        y_train_predict = model.predict(X_train[:m])
        y_val_predict = model.predict(X_val)
        train_errors.append(mean_squared_error(y_train_predict, y_train[:m]))
        val_errors.append(mean_squared_error(y_val_predict, y_val))

    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="Навчальний набір (помилка)")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Перевірочний набір (помилка)")
    plt.legend()
    plt.xlabel("Кількість навчальних даних")
    plt.ylabel("RMSE (середньоквадратична помилка)")
    plt.grid(True)

def generate_data(m):
    np.random.seed(42)
    X = np.linspace(-3, 3, m).reshape(-1, 1)
    y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, (m, 1))

    return X, y

m = 100
X, y = generate_data(m)

lin_reg = LinearRegression()
plt.figure(figsize=(10, 5))
plt.title("Криві навчання: Лінійна регресія")
plot_learning_curves(lin_reg, X, y)
plt.show()

polynomial_regression_10 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plt.figure(figsize=(10, 5))
plt.title("Криві навчання: Поліном 10-го ступеня")
plot_learning_curves(polynomial_regression_10, X, y)
plt.show()

polynomial_regression_2 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plt.figure(figsize=(10, 5))
plt.title("Криві навчання: Поліном 2-го ступеня")
plot_learning_curves(polynomial_regression_2, X, y)
plt.show()
```

Результат виконання програми:

Рис. 6

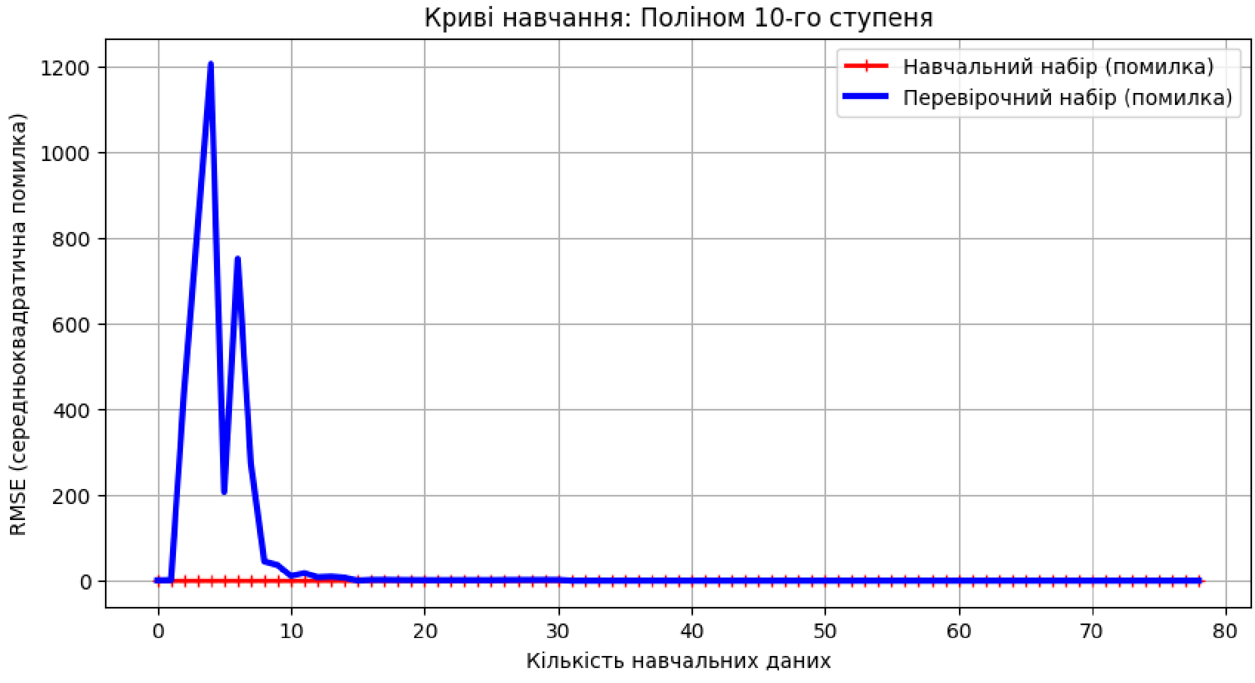


Рис. 7

Рис. 8

У завданні порівнювалися криві навчання трьох моделей:

- Лінійна модель (рис. 6) – її графік показує, що обидві криві швидко сходяться і стабілізуються на високому рівні похибки. Можна зробити висновок, що модель недонавчена. Дана модель занадто проста, щоб уловити синусоїдальну залежність, тому додавання більшої кількості навчальних зразків не допоможе.
- Поліноміальна модель 10-го ступеня (рис. 7) – навчальна помилка на графіку для усіх даних дорівнює нулю, а перевірочна помилка на початку є надзвичайно високою. Можна зробити висновок, що модель перенавчена, тобто добре запам'ятовує навчальні дані, але погано узагальнює, що є прикладом високої дисперсії.
- Поліноміальна модель 2-го ступеня (рис. 8) – її графік показує результати, практично ідентичні лінійній моделі. Обидві криві також сходяться на високій похибці. Це означає, що параболічна модель так само нездатна описати синусоїдальну залежність, як і пряма лінія, і також є недонавченою.

Отже, жодна з трьох протестованих моделей не є оптимальною для цього набору даних. Лінійна та поліноміальна моделі 2-го ступеня є недонавченими, а модель 10-го ступеня – перенавченою. Для знаходження оптимального балансу слід було б протестувати моделі проміжних ступенів.

Завдання 4.7. Експериментально отримані N-значень величини Y при значеннях величини X. Відшукати параметри функції за методом найменших квадратів. Побудувати графіки, де в декартовій системі координат нанести експериментальні точки і графік апроксимуючої функції.

9	X	0,3	1,0	1,5	2,2	3,6	4,5
	Y	5	10	13	16	17	18

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([0.3, 1.0, 1.5, 2.2, 3.6, 4.5])
Y = np.array([5, 10, 13, 16, 17, 18])
n = len(X)

beta1 = (n * np.sum(X*Y) - np.sum(X) * np.sum(Y)) / (n * np.sum(X**2) -
(np.sum(X))**2)
beta0 = np.mean(Y) - beta1 * np.mean(X)

print(f"β0 = {beta0:.2f}")
print(f"β1 = {beta1:.2f}")
print(f"Рівняння прямої: y = {beta0:.2f} + {beta1:.2f}x")

Y_pred = beta0 + beta1 * X
S = np.sum((Y - Y_pred)**2)
print(f"S({beta0:.2f}; {beta1:.2f}) = {S:.6f}")

plt.scatter(X, Y, color='blue', label='Експериментальні точки')
plt.plot(X, Y_pred, color='red', label='Апроксимація (МНК)')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Метод найменших квадратів')
plt.legend()
plt.grid(True)
```

```
plt.show()
```

Результат виконання програми:

```
•••  β0 = 7.03
     β1 = 2.81
     Рівняння прямої: y = 7.03 + 2.81x
     S(7.03; 2.81) = 21.959599
```

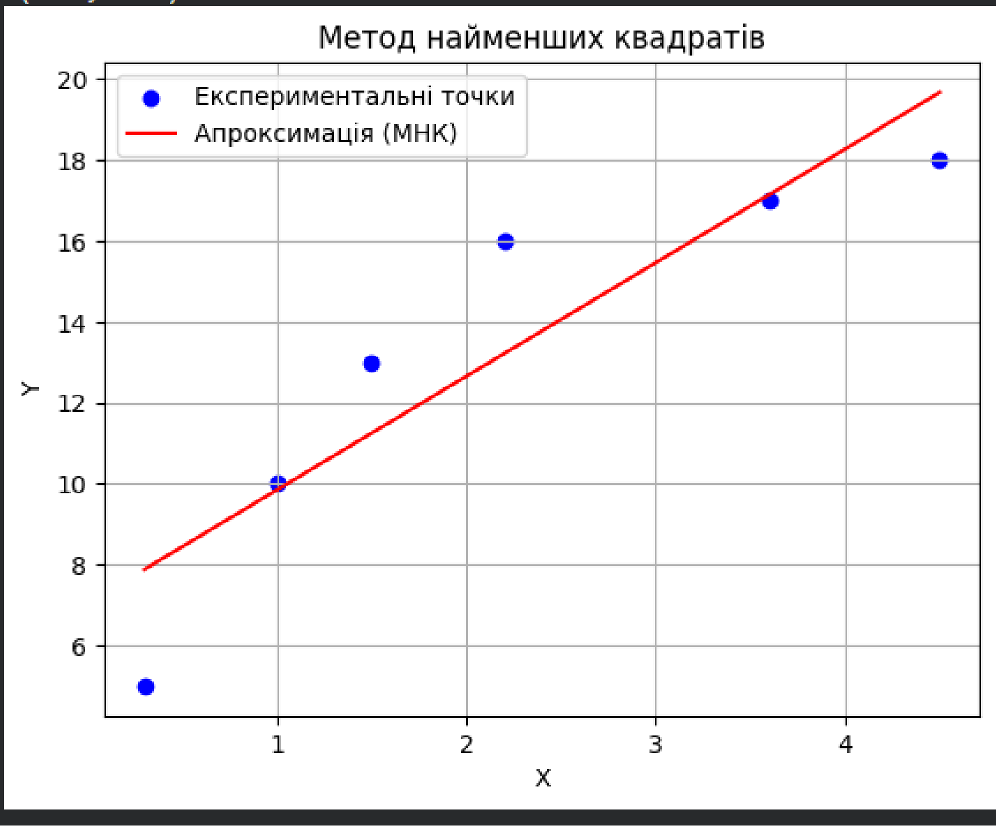


Рис. 9

У завданні було отримано наступні коефіцієнти (рис. 9):

$\beta_0 = 7.03$ і $\beta_1 = 2.81$,

тобто рівняння апроксимуючої функції виглядає наступним чином: $y = 7.03 + 2.81x$

Мінімальна сума квадратів похибок дорівнює 21.96.

На графіку (рис. 9) видно, що апроксимуюча функція добре узгоджується з експериментальними даними, оскільки лінія регресії проходить близько до більшості точок.

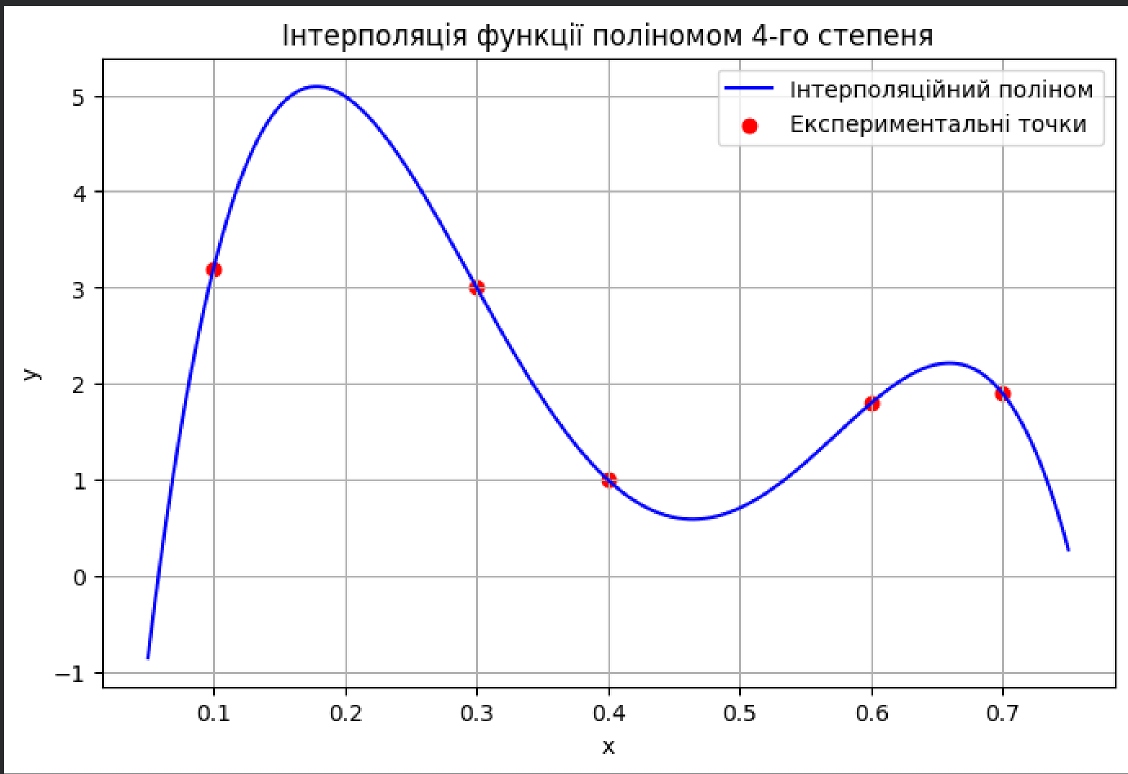
Завдання 4.8. Виконати інтерполяцію функції, задану в табличній формі в п'яти точках. Розрахунки виконати в середовищі Python.

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([0.1, 0.3, 0.4, 0.6, 0.7])
y = np.array([3.2, 3, 1, 1.8, 1.9])

X = np.vander(x, increasing=True)
print(f"Матриця Вандермонда:\n{X}")

coeffs = np.linalg.solve(X, y)
print(f"Коефіцієнти інтерполяційного полінома:\n{coeffs}")

polynomial = np.poly1d(coeffs[::-1])
print(f"Функція інтерполяційного полінома:\n{polynomial}")

x_plot = np.linspace(min(x)-0.05, max(x)+0.05, 200)
y_plot = polynomial(x_plot)

plt.figure(figsize=(8,5))
plt.plot(x_plot, y_plot, label="Інтерполяційний поліном", color='blue')
plt.scatter(x, y, color='red', label="Експериментальні точки")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Інтерполяція функції поліномом 4-го степеня")
plt.legend()
plt.grid(True)
plt.show()

print("Значення функції в проміжних точках:")
x_test = [0.2, 0.5]
y_test = polynomial(x_test)
for xt, yt in zip(x_test, y_test):
    print(f"P({xt}) = {yt:.4f}")
```

Результат виконання програми:

```
••• Матриця Вандермонда:
    [[1.000e+00 1.000e-01 1.000e-02 1.000e-03 1.000e-04]
     [1.000e+00 3.000e-01 9.000e-02 2.700e-02 8.100e-03]
     [1.000e+00 4.000e-01 1.600e-01 6.400e-02 2.560e-02]
     [1.000e+00 6.000e-01 3.600e-01 2.160e-01 1.296e-01]
     [1.000e+00 7.000e-01 4.900e-01 3.430e-01 2.401e-01]]
    Коефіцієнти інтерполяційного полінома:
    [  -8.18        186.25       -864.02777778 1480.55555556 -852.77777778]
    Функція інтерполяційного полінома:
            4        3       2
    -852.8 x + 1481 x - 864 x + 186.2 x - 8.18
```




```
    Значення функції в проміжних точках:
    P(0.2) = 4.9889
    P(0.5) = 0.7089
```

Рис. 10

У завданні було виконано інтерполяцію функції за допомогою полінома 4-го степеня. Складено матрицю Вандермонда та розв'язано систему лінійних рівнянь для знаходження коефіцієнтів, на основі яких створено функцію полінома.

На графіку (рис. 10) інтерполяційний поліном проходить через усі задані експериментальні точки.

Значення функції в проміжних точках 0.2 і 0.5 дорівнює 4.9889 і 0.7089 відповідно.

Висновок: в ході виконання лабораторної роботи ми дослідити методи регресії даних у машинному навчанні, використовуючи спеціалізовані бібліотеки та мову програмування Python.

Репозиторій: https://github.com/Vanchik21/AI