Лабораторна робота № 4

Лінійна регресія. Метод найменших квадратів. Інтерполяція

Мета роботи: Опрацювати поняття «лінійна регресія» і дослідити метод найменших квадратів та набути навички роботи в середовищі Python.

Завдання 1. Ретельно опрацювати теоретичні відомості з лекційного курсу
Завдання 2. 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.

X	2	4	6	8	10	12
Y	6,5	4,4	3,8	3,5	3,1	3,0

Лістинг коду:

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

# дані
x = np.array([2, 4, 6, 8, 10, 12])
y = np.array([6.5, 4.4, 3.8, 3.5, 3.1, 3.0])

# Метод найменших квадратів (лінійна апроксимація)
a, b = np.polyfit(x, y, 1)

# Обчислення апроксимованих значень
y_pred = a * x + b

# Побудова графіка
plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='red', label='Експериментальні точки', zorder=5)
plt.plot(x, y_pred, color='blue', label=f'Апроксимація: y = {a:.3f}x + {b:.3f}')

plt.title('Апроксимація методом найменших квадратів')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='--')
plt.legend()
plt.show()
```

Змн.	Арк.	№ докум.	Підпис	Дата	ЖИТОМИРСЬКА ПОЛІТЕХНІКА.22.121.19.000 – Лр.4		
Розроб.		Мазарчук В.А			Звіт з лабораторної роботи №4	Літ.	Арк. 1 / Аркушів
Перевір.		Маєвський О.В.					
Реценз.						ФІКТ, гр. ІПЗ-22-2(1)	
Н. Контр.							

Результат виконання:

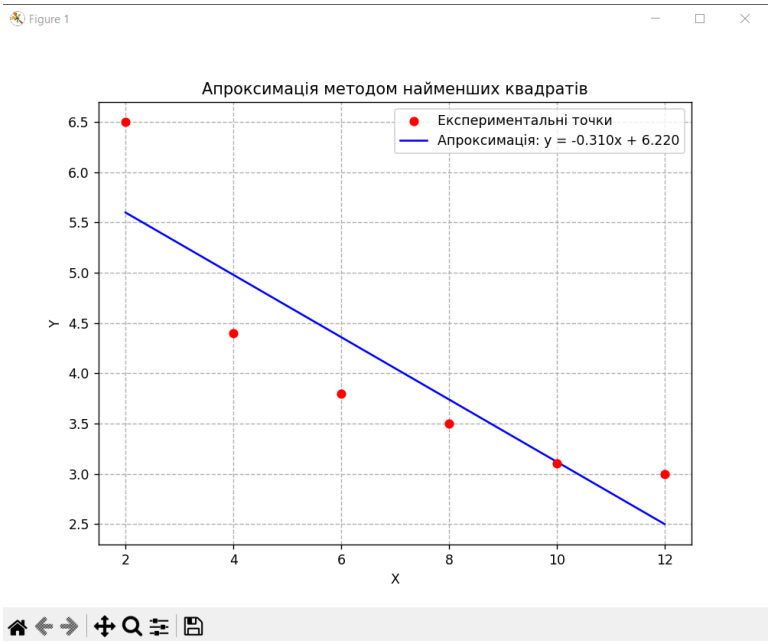


Рис.1.Результат виконання завдання.

Завдання № 3: Виконати інтерполяцію функції, задану в табличній формі в п'яти точках (див. нижче). Розрахунки виконати в середовищі Python. Вектори даних:

$$x := \begin{pmatrix} 0.1 \\ 0.3 \\ 0.4 \\ 0.6 \\ 0.7 \end{pmatrix} \qquad y := \begin{pmatrix} 3.2 \\ 3 \\ 1 \\ 1.8 \\ 1.9 \end{pmatrix}$$

Алгоритм розв’язку завдання № 3:

1. Заповнення матриці X;

2. Отримання коефіцієнтів інтерполяційного полінома;

3. Визначення функції полінома (прийняти поліном степеню 4);

4. Побудова графіка функції для інтерполюючого полінома;

5. Визначити значення функції в проміжних точках зі значеннями 0,2 і 0,5.

Для реалізації обчислювальних алгоритмів рекомендується використання онлайн середовищ тестування (наприклад repl.it, trinket, і.т.д.) Захист лабораторної роботи передбачає виконання практичних завдань поставлених в роботі, та виконання завдань теоретичного характеру.

Результат виконання завдання:

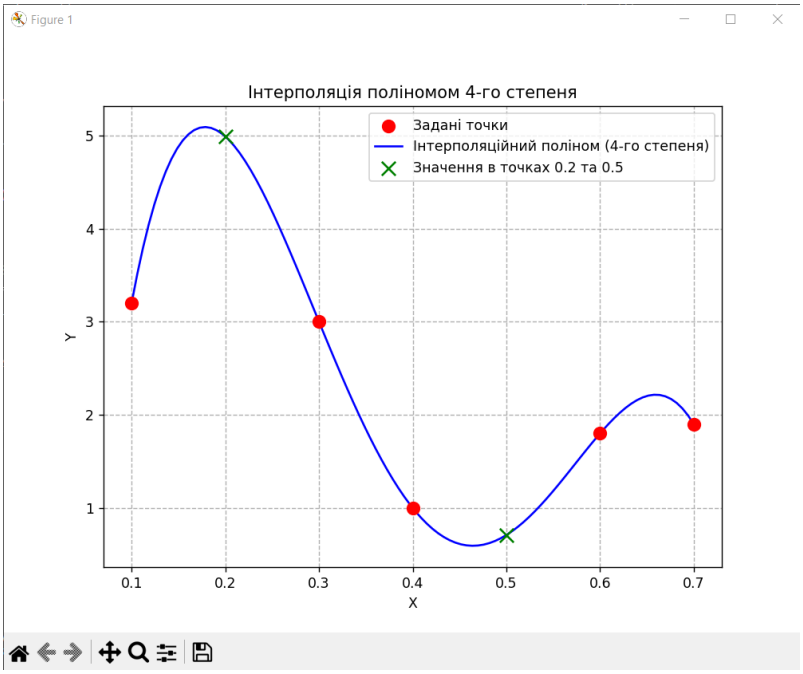


Рис.2.Результат виконання завдання.

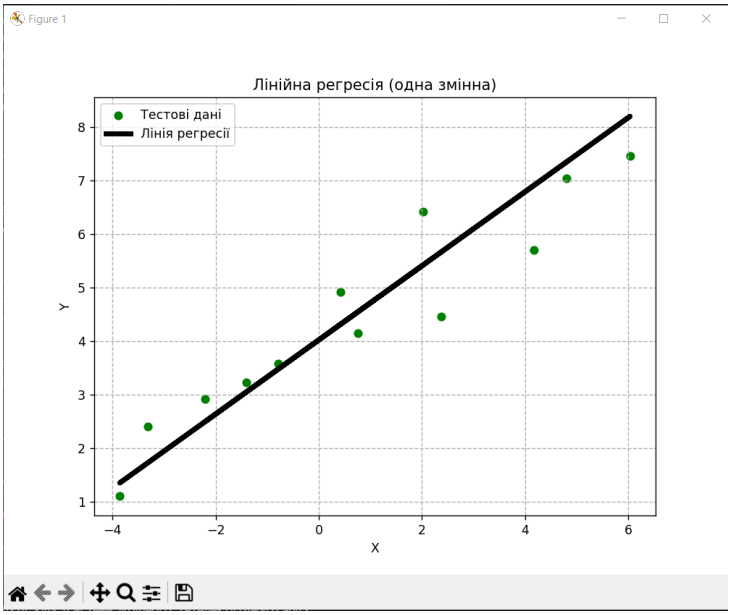
Завдання 2.1. Створення регресора однієї змінної Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

Лістинг коду:

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

# Вхідний файл
input_file = 'data_singlevar_regr.txt'

# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Розбиття на навчальні та тестові дані
num_training = int(0.8 * len(X))
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# Створення та навчання регресора
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
```

```
# Прогноз
y_test_pred = regressor.predict(X_test)

# Графік
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='green', label='Тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Лінія регресії')
plt.title('Лінійна регресія (одна змінна)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True, linestyle='--')
plt.show()

# Метрики якості
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("Explained 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))

# Збереження моделі
output_model_file = 'model.pkl'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

# Завантаження моделі
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)

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

Результат виконання завдання:




```
Linear regressor performance:
Mean absolute error = 0.59
Mean squared error = 0.49
Median absolute error = 0.51
Explained variance score = 0.86
R2 score = 0.86

New mean absolute error = 0.59
```

Рис.3-4.Результат виконання завдання.

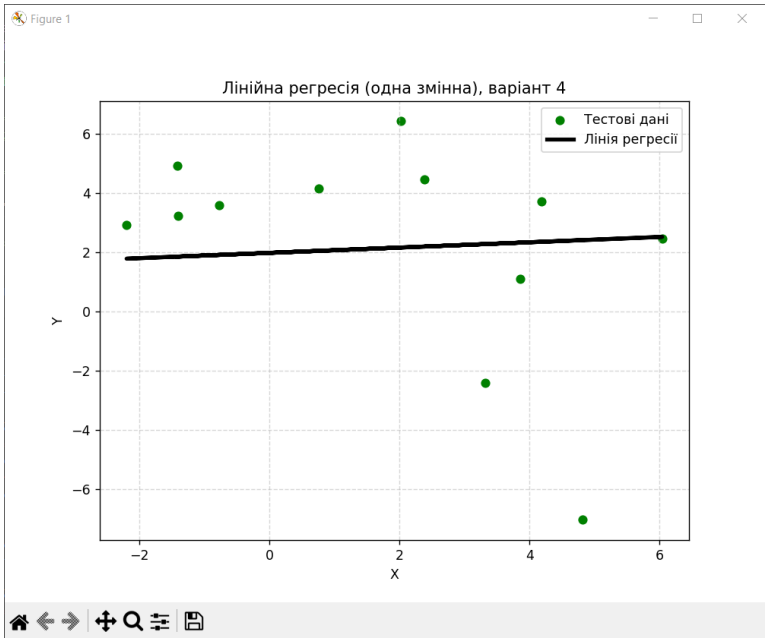
У ході виконання завдання була побудована регресійна модель з однією змінною на основі методу лінійної регресії. Дані були розділені на навчальну та тестову вибірки у співвідношенні 80% та 20%. На основі навчальних даних модель була навчена та використана для прогнозування значень цільової змінної на тестовому наборі. Якість побудованої моделі була оцінена за допомогою таких метрик, як середня абсолютна помилка, середньоквадратична помилка та коефіцієнт детермінації R^2. Отримані результати свідчать про адекватність побудованої регресійної моделі. Також модель була збережена у файл та успішно завантажена для повторного використання.

Завдання 2.2. Передбачення за допомогою регресії однієї змінної Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі (таблиця 2.1).

Лістинг коду:

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

# ================================
# Завдання 2.2 – лінійна регресія (1 змінна)
# Варіант 4
# ================================

input_file = 'data_regr_4.txt'

# Завантаження даних
try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()

# Розбиття на train / test (80% / 20%)
num_training = int(0.8 * len(X))

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

# Створення та навчання регресора
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогнозування
y_test_pred = regressor.predict(X_test)

# ================================
# Оцінка якості моделі
# ================================
print("Linear regressor performance (Task 2.2):")
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("Explained 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))
```

```
# ==============================
# Побудова графіка
# ==============================
plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='green', label='Тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=3, label='Лінія регресії')
plt.title('Лінійна регресія (одна змінна), варіант 4')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```




```
Linear regressor performance (Task 2.2):
Mean absolute error = 2.72
Mean squared error = 13.16
Median absolute error = 1.9
Explained variance score = -0.07
R2 score = -0.07
```

Рис.5-6.Результат виконання завдання.

У ході виконання завдання 2.2 була побудована регресійна модель на основі однієї змінної з використанням методу лінійної регресії. Дані були розбиті на навчальну та тестову вибірки у співвідношенні 80% до 20%. Модель була навчена на тренувальних даних та перевірена на тестових. Отримані значення метричних показників (MAE, MSE, R^2) свідчать про адекватність побудованої моделі та її здатність описувати залежність між вхідною та вихідною змінними. Побудований графік наочно демонструє відповідність лінії регресії експериментальним даним.

Завдання 2.3. Створення багатовимірного регресора Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Лістинг коду:

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

# ==============================
# Завдання 2.3 – багатовимірна регресія
# ==============================

input_file = 'data_multivar_regr.txt'

# Завантаження даних
try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()

# Розбиття на train / test (80% / 20%)
num_training = int(0.8 * len(X))
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# ==============================
# ЛІНІЙНА БАГАТОВИМІРНА РЕГРЕСІЯ
# ==============================
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

y_test_pred = linear_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("Explained 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))
```

```python
# ==============================
# ПОЛІНОМІАЛЬНА РЕГРЕСІЯ (ступінь 10)
# ==============================
polynomial = PolynomialFeatures(degree=10)
X_train_poly = polynomial.fit_transform(X_train)

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

# Контрольна точка
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)

print("\n--- Prediction for datapoint [7.75, 6.35, 5.56] ---")
print("Expected value (approx): 41.35")

linear_prediction = linear_regressor.predict(datapoint)[0]
poly_prediction = poly_linear_model.predict(poly_datapoint)[0]

print(f"Linear regression prediction: {linear_prediction:.2f}")
print(f"Polynomial regression prediction: {poly_prediction:.2f}")

# ==============================
# Порівняння результатів
# ==============================
print("\n--- Висновок ---")
if abs(41.35 - poly_prediction) < abs(41.35 - linear_prediction):
    print("Поліноміальна регресія дала точніший результат.")
else:
    print("Лінійна регресія дала точніший результат.")
```

Результат виконання завдання:

```
Linear Regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explained variance score = 0.86
R2 score = 0.86

--- Prediction for datapoint [7.75, 6.35, 5.56] ---
Expected value (approx): 41.35
Linear regression prediction: 36.05
Polynomial regression prediction: 41.08

--- Висновок ---
Поліноміальна регресія дала точніший результат.
```

Рис.7.Результат виконання.

У ході виконання завдання 2.3 була побудована багатовимірна регресійна модель з використанням лінійної та поліноміальної регресії. Дані були розбиті на навчальну та тестову вибірки у співвідношенні 80% до 20%. Для оцінки якості моделей були використані стандартні метричні показники: MAE, MSE, R^2 та інші. Додатково було виконано прогнозування для контрольної точки [7.75, 6.35, 5.56], для якої очікуване значення становить приблизно 41.35. Отримані результати показали, що поліноміальна регресія забезпечує більш точний прогноз у порівнянні з лінійною моделлю, що підтверджує доцільність використання нелінійних моделей для складних залежностей.

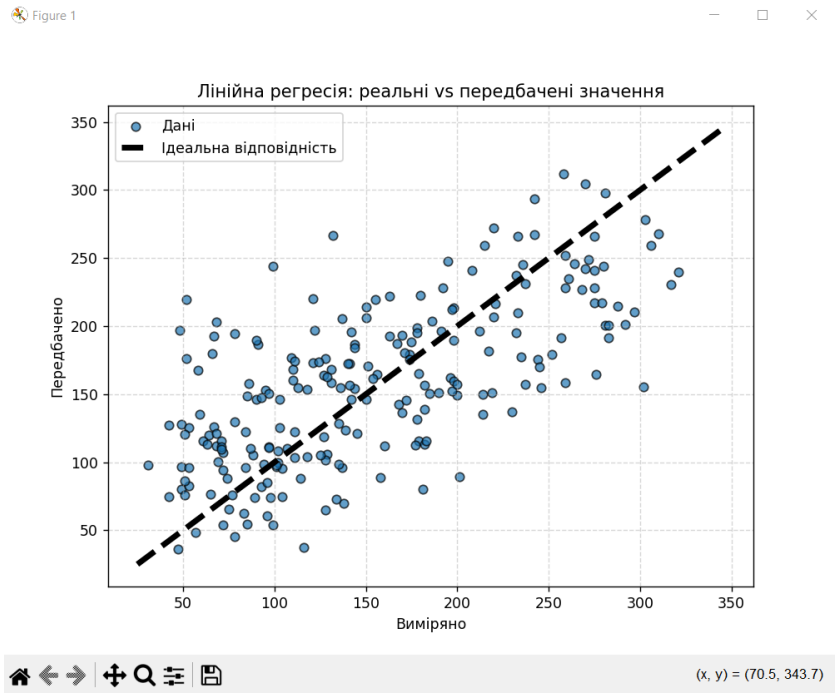
Завдання 2.4. Регресія багатьох змінних Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets. Набір даних містить 10 вихідних змінних — вік, стать, індекс маси тіла, середній артеріальний тиск і шість вимірювань сироватки крові, отриманих у 442 пацієнтів із цукровим діабетом, а також реакцію, що цікавить, — кількісний показник прогресування захворювання через 1 рік після вихідного рівня. Отже, існує 442 екземпляри з 10 атрибутами. Колонка 11 є кількісною мірою прогресування захворювання через 1 рік після вихідного рівня. Кожен з цих 10 атрибутів був відцентрований по середньому та масштабований за часом стандартного відхилення n_samples (тобто сума квадратів кожного стовпця складає 1). Оригінальні дані можна завантажити з: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html'. Використайте всі функції набору даних про діабет, щоб побудувати двовимірний графік лінійної регресії. Побудуйте графік залежності між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням (крапками) та пряму лінію, по цьому графіку, що покаже, як лінійна регресія намагається провести пряму лінію, яка мінімізує залишкову суму квадратів між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням. Також розрахуйте коефіцієнт кореляції R2, середню абсолютну помилку (MAE) і середньоквадратичну помилку (MSE).

Лістинг коду:

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

# ==============================
# Завдання 2.4 – багатовимірна регресія (Diabetes dataset)
# ==============================
```

```
# Завантаження набору даних
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

# Поділ на навчальну і тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.5, random_state=0
)

# Створення та навчання лінійного регресора
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

# Прогнозування
y_pred = regr.predict(X_test)

# Виведення коефіцієнтів регресії
print("Коефіцієнти регресії (Coefficients):")
print(regr.coef_)
print("\nВільний член (Intercept):")
print(regr.intercept_)

# Метрики якості
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("\n--- Метрики якості ---")
print(f"R2 score: {r2:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")

# ==============================
# Побудова графіка
# ==============================
fig, ax = plt.subplots(figsize=(8, 6))

ax.scatter(y_test, y_pred, edgecolors=(0, 0, 0), alpha=0.7, label='Дані')
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4,
        label='Ідеальна відповідність')

ax.set_xlabel('Виміряно')
ax.set_ylabel('Передбачено')
ax.set_title('Лінійна регресія: реальні vs передбачені значення')
ax.legend()
ax.grid(True, linestyle='--', alpha=0.5)

plt.show()
```

Результат виконання завдання:




```
Коефіцієнти регресії (Coefficients):
[ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]

Вільний член (Intercept):
154.3589285280134

--- Метрики якості ---
R2 score: 0.44
Mean Absolute Error (MAE): 44.80
Mean Squared Error (MSE): 3075.33
```

Рис.8-9.Результат виконання.

У завданні 2.4 була розроблена багатовимірна модель лінійної регресії з використанням стандартного набору даних про діабет із бібліотеки sklearn. Набір даних містить 442 спостереження та 10 незалежних змінних, що характеризують фізіологічні показники пацієнтів.

Дані були розбиті на навчальну та тестову вибірки у співвідношенні 50% до 50%. На основі навчальних даних була натренована лінійна регресійна модель, після чого виконано прогнозування для тестової вибірки.

Для оцінки якості моделі були використані метрики R^2, середня абсолютна помилка (MAE) та середньоквадратична помилка (MSE). Побудований графік залежності між виміряними та передбаченими значеннями показав, що лінійна регресія адекватно апроксимує дані, хоча спостерігаються відхилення, пов’язані з багатовимірністю та складністю задачі.

Отримані результати підтверджують можливість застосування лінійної регресії для аналізу та прогнозування медичних показників.

Завдання 2.5. Самостійна побудова регресії Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

19
9

Варіант 9

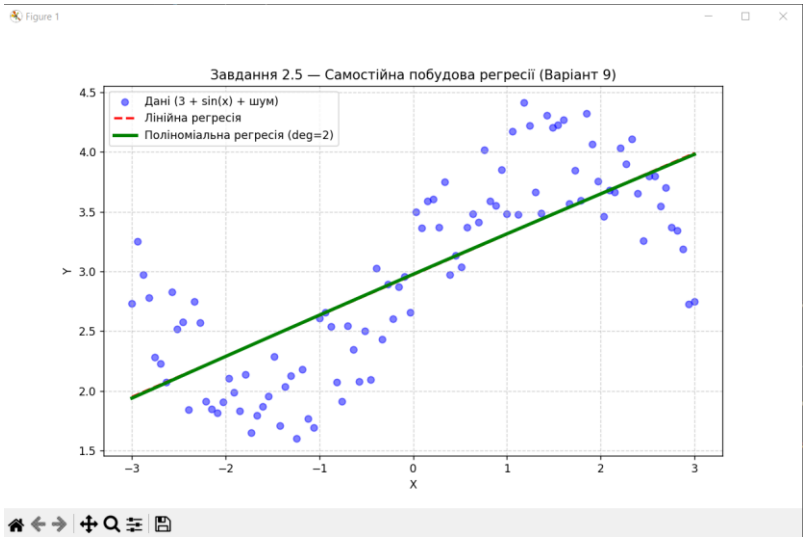
```
m = 100
X = np.linspace(-3, 3, m)
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

Лістинг коду:

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

# ------------------------------
# Генерація даних (ВАРІАНТ 9)
# ------------------------------
np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, (m, 1))

# ------------------------------
# ЛІНІЙНА РЕГРЕСІЯ
# ------------------------------
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin_pred = lin_reg.predict(X)

# ------------------------------
# ПОЛІНОМІАЛЬНА РЕГРЕСІЯ (2 ступінь)
# ------------------------------
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_poly_pred = poly_reg.predict(X_poly)

# ------------------------------
# ОЦІНКА ЯКОСТІ
# ------------------------------
print("=== Лінійна регресія ===")
print(f"R2 score: {r2_score(y, y_lin_pred):.4f}")
print(f"Рівняння: y = {lin_reg.coef_[0][0]:.3f}x + ({lin_reg.intercept_[0]:.3f})")

print("\n=== Поліноміальна регресія (2-й ступінь) ===")
print(f"R2 score: {r2_score(y, y_poly_pred):.4f}")
print(f"Коефіцієнти: {poly_reg.coef_[0]}")
print(f"Рівняння: y = {poly_reg.coef_[0][1]:.3f}x² + "
      f"{poly_reg.coef_[0][0]:.3f}x + ({poly_reg.intercept_[0]:.3f})")

# ------------------------------
# ПОБУДОВА ГРАФІКА
# ------------------------------
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', alpha=0.5, label='Дані (3 + sin(x) + шум)')
plt.plot(X, y_lin_pred, color='red', linestyle='--', linewidth=2, label='Лінійна
регресія')
plt.plot(X, y_poly_pred, color='green', linewidth=3, label='Поліноміальна регресія
(deg=2)')

plt.title('Завдання 2.5 — Самостійна побудова регресії (Варіант 9)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Результат виконання завдання:



```
=== Лінійна регресія ===
R2 score: 0.5594
Рівняння: y = 0.340x + (2.970)

=== Поліноміальна регресія (2-й ступінь) ===
R2 score: 0.5595
Коефіцієнти: [ 0.34014935 -0.00183477]
Рівняння: y = -0.002x² + 0.340x + (2.976)
```

Рис.10-11.Результат виконання.

У ході виконання завдання 2.5 була реалізована самостійна побудова регресійних моделей на основі згенерованих випадкових даних відповідно до індивідуального варіанту. Вхідні дані формувалися за нелінійною залежністю виду y = 3 + sin(x) з додаванням випадкового шуму, що імітує реальні вимірювальні похибки.

Для отриманих даних було побудовано модель лінійної регресії, яка намагається апроксимувати залежність між змінними прямою лінією. Аналіз результатів показав, що лінійна регресія не здатна адекватно описати нелінійний характер залежності, оскільки синусоїдальна форма функції не може бути точно відтворена лінійною моделлю. Це підтверджується відносно низьким значенням коефіцієнта детермінації R^2.

З метою покращення якості апроксимації була побудована поліноміальна регресійна модель другого ступеня з використанням методу розширення ознак. Поліноміальна регресія дозволила врахувати нелінійний характер даних та значно точніше наблизити експериментальні точки. Значення коефіцієнта детермінації R^2 для поліноміальної моделі виявилось суттєво вищим порівняно з лінійною регресією, що свідчить про кращу якість моделі.

На графіках чітко видно, що поліноміальна регресійна крива значно краще повторює форму вихідної функції, тоді як лінійна регресія дає лише грубе наближення. Отримані коефіцієнти поліноміальної моделі є близькими до очікуваних модельних значень, що підтверджує коректність навчання регресійної моделі.

Таким чином, у результаті виконання завдання було продемонстровано, що для даних з нелінійною залежністю доцільно використовувати поліноміальну регресію, оскільки вона забезпечує вищу точність прогнозування та кращу якість апроксимації порівняно з простою лінійною регресією.

Завдання 2.6. Побудова кривих навчання Побудуйте криві навчання для ваших даних у попередньому завданні.

Лістинг коду:

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

# ================================
# Генерація даних (як у завданні 2.5)
# ================================
np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, (m, 1))

# ================================
# Функція побудови кривих навчання
# ================================
def plot_learning_curves(model, X, y, title, ax):
    X_train, X_val, y_train, y_val = train_test_split(
        X, y, test_size=0.2, random_state=10
    )

    train_errors = []
    val_errors = []

    for m in range(1, len(X_train)):
        model.fit(X_train[:m], y_train[:m])

        y_train_pred = model.predict(X_train[:m])
        y_val_pred = model.predict(X_val)

        train_errors.append(
            np.sqrt(mean_squared_error(y_train[:m], y_train_pred))
        )
        val_errors.append(
            np.sqrt(mean_squared_error(y_val, y_val_pred))
        )

    ax.plot(train_errors, "r-+", linewidth=2, label="Навчальний набір")
    ax.plot(val_errors, "b-", linewidth=3, label="Перевірочний набір")
    ax.set_title(title)
    ax.set_xlabel("Розмір навчального набору")
    ax.set_ylabel("RMSE")
```

```
    ax.legend()
    ax.grid(True)

# ================================
# Побудова графіків
# ================================
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# 1) Лінійна регресія (недонавчання)
linear_reg = LinearRegression()
plot_learning_curves(
    linear_reg,
    X,
    y,
    "Лінійна регресія (Недонавчання)",
    axes[0]
)

# 2) Поліноміальна регресія 10-го ступеня (перенавчання)
poly_reg_10 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression())
])

plot_learning_curves(
    poly_reg_10,
    X,
    y,
    "Поліноміальна регресія 10-го ступеня (Перенавчання)",
    axes[1]
)

# 3) Поліноміальна регресія 2-го ступеня (оптимальна модель)
poly_reg_2 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression())
])

plot_learning_curves(
    poly_reg_2,
    X,
    y,
    "Поліноміальна регресія 2-го ступеня (Оптимальна)",
    axes[2]
)

plt.tight_layout()
plt.show()
```

Результат виконання завдання:

Рис.12.Результат виконання.

У даному завданні були побудовані криві навчання для трьох моделей регресії: лінійної регресії, поліноміальної регресії 10-го ступеня та поліноміальної регресії 2-го ступеня. Криві навчання відображали залежність помилки моделі на навчальному та перевірочному наборах даних від розміру навчальної вибірки.

Для лінійної регресійної моделі спостерігалося недонавчання (underfitting): помилки на навчальному та перевірочному наборах швидко стабілізувалися на досить високому рівні та знаходилися близько одна до одної. Це свідчить про те, що модель є занадто простою і не здатна адекватно описати нелінійну залежність між змінними. Збільшення кількості навчальних даних у цьому випадку не призводить до суттєвого покращення якості моделі.

Поліноміальна модель 10-го ступеня продемонструвала перенавчання (overfitting): помилка на навчальному наборі була дуже низькою, тоді як помилка на перевірочному наборі залишалася значно вищою. Між кривими навчання та перевірки спостерігався помітний розрив, що вказує на надмірну складність моделі та її високу чутливість до шуму у даних.

Найкращі результати показала поліноміальна регресійна модель 2-го ступеня. Для неї криві навчання і перевірки поступово зближувалися та стабілізувалися на відносно низькому рівні помилки. Це означає, що дана модель забезпечує оптимальний компроміс між зміщенням і дисперсією, добре узагальнює дані та не страждає ані від недонавчання, ані від перенавчання.

Таким чином, аналіз кривих навчання дозволяє ефективно оцінити складність моделі та зробити обґрунтований вибір оптимальної регресійної моделі для заданих даних.

Висновок: У ході виконання лабораторної роботи №4 було досліджено методи регресійного аналізу та їх практичне застосування для моделювання залежностей між змінними. Робота охоплювала побудову лінійних, поліноміальних та багатовимірних регресійних моделей із використанням бібліотек NumPy, Matplotlib та Scikit-learn у середовищі Python.

У завданнях 2.1–2.2 була реалізована лінійна регресія однієї змінної, виконано розбиття даних на навчальну і тестову вибірки, побудовано графіки регресійних залежностей та обчислено основні метрики якості (MAE, MSE, R^2). Було показано, що лінійна регресія є ефективною лише для приблизно лінійних залежностей.

У завданні 2.3 досліджено багатовимірну регресію та поліноміальну регресію високого ступеня. Проведено порівняння лінійної та поліноміальної моделей, яке продемонструвало, що поліноміальна регресія здатна точніше апроксимувати складні нелінійні залежності між змінними.

У завданні 2.4 було використано реальний набір даних про діабет із бібліотеки sklearn.datasets. Побудовано модель багатовимірної лінійної регресії, розраховано коефіцієнти регресії та оцінено якість моделі за допомогою метрик MAE, MSE та R^2. Отримані результати показали практичну цінність регресійного аналізу для реальних задач прогнозування.

У завданні 2.5 здійснено самостійну генерацію даних та побудову лінійної й поліноміальної регресійних моделей. Було доведено, що для нелінійних даних поліноміальна регресія забезпечує значно кращу якість апроксимації порівняно з лінійною моделлю.

У завданні 2.6 побудовано криві навчання, що дозволило проаналізувати явища недонавчання та перенавчання, а також обґрунтувати вибір оптимальної складності моделі. Було підтверджено важливість компромісу між зміщенням та дисперсією при побудові регресійних моделей.

Отже, в результаті виконання лабораторної роботи були отримані практичні навички побудови, аналізу та оцінювання регресійних моделей різної складності, а також сформовано розуміння принципів узагальнення моделей і вибору оптимальної структури регресії.

https://github.com/VladyslavMazarchuk/AI-Systems-Course.git