Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

Завдання 2.1. Створення регресора однієї змінної

Лістинг програми:

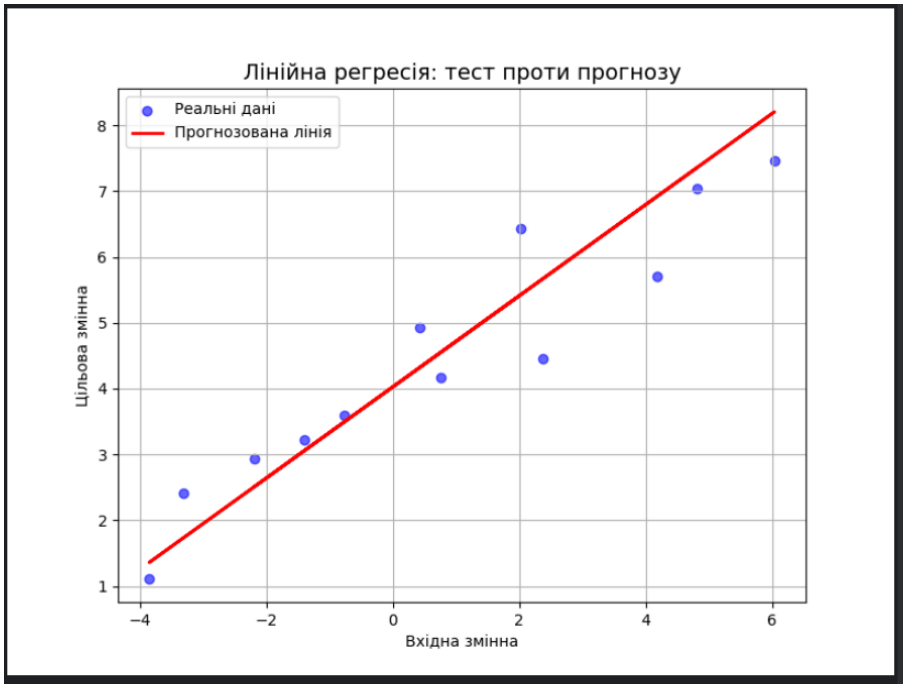
```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt

input_file = 'data_singlevar_regr.txt'

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

# Розбивка даних на навчальну та тестову вибірки
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Навчальні дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

# Створення об'єкта для лінійної регресії
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = regressor.predict(X_test)

# Побудова графіка з іншими стилями та кольорами
plt.figure(figsize=(8, 6))

# Зміна кольору точок (реальних даних) на синій
plt.scatter(X_test, y_test, color='blue', label='Реальні дані', alpha=0.6)  # Сині точки для реальних даних

# Зміна кольору лінії прогнозу на червоний
plt.plot(X_test, y_test_pred, color='red', linewidth=2, label='Прогнозована лінія')  # Червона лінія для прогнозів
```

```
# Додавання заголовка, міток осей, легенди та сітки
plt.title('Лінійна регресія: тест проти прогнозу', fontsize=14)
plt.xlabel('Вхідна змінна')
plt.ylabel('Цільова змінна')
plt.legend()
plt.grid(True)
# Показати графік
plt.show()

# Збереження моделі у файл за допомогою pickle
with open('linear_regressor_model.pkl', 'wb') as f:
    pickle.dump(regressor, f)

# Оцінка моделі за допомогою різних метрик
print("Результати роботи лінійної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Середньоквадратична похибка =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Корінь середньоквадратичної похибки =",
round(np.sqrt(sm.mean_squared_error(y_test, y_test_pred)), 2))  # RMSE
print("Медіанна абсолютна похибка =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Коефіцієнт поясненої дисперсії =",
round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 =", round(sm.r2_score(y_test, y_test_pred), 2))

# Завантаження та перевірка моделі, якщо це потрібно
with open('linear_regressor_model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

# Прогнозування за допомогою завантаженої моделі (перевірка)
y_test_pred_loaded_model = loaded_model.predict(X_test)
print("R2 для завантаженої моделі:", round(sm.r2_score(y_test,
y_test_pred_loaded_model), 2))
```

Результат виконання:

```
"C:\Program Files\Python313\python.exe" "D:\
Результати роботи лінійної регресії:
Середня абсолютна похибка = 0.59
Середньоквадратична похибка = 0.49
Корінь середньоквадратичної похибки = 0.7
Медіанна абсолютна похибка = 0.51
Коефіцієнт поясненої дисперсії = 0.86
R2 = 0.86
R2 для завантаженої моделі: 0.86

Process finished with exit code 0
```

Візуалізація:



Завдання 2.2. Передбачення за допомогою регресії однієї змінної

Варіант 1 файл: data_regr_1.txt

Лістинг програми

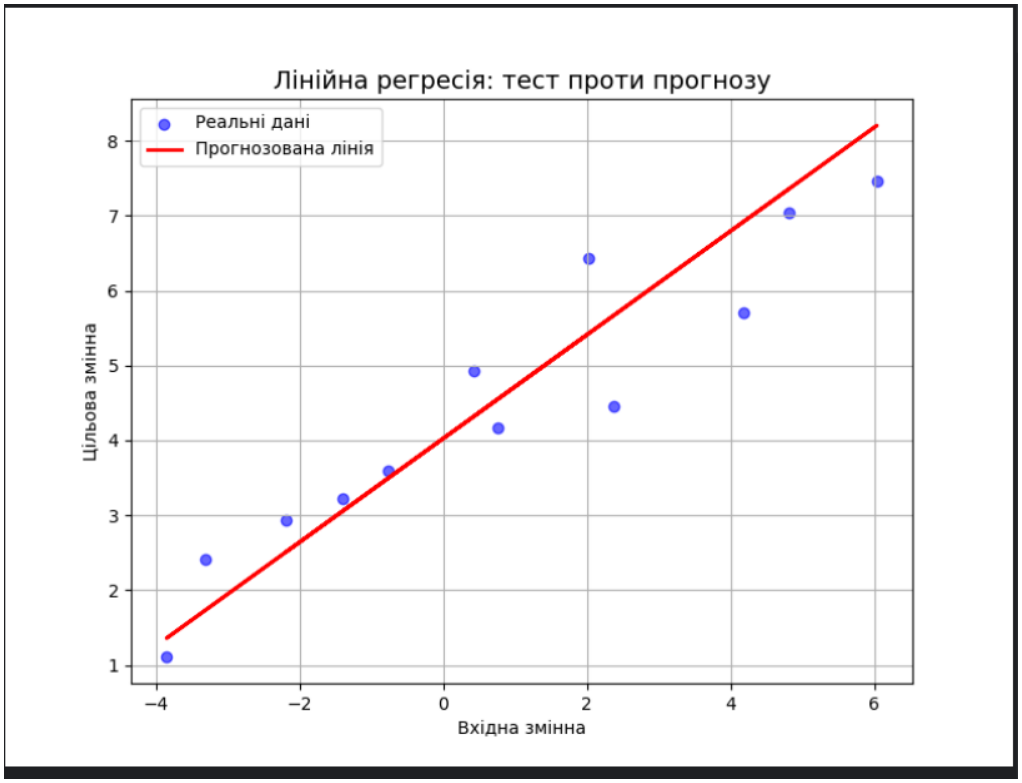
```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt

input_file = 'data_regr_1.txt'

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

# Розбивка даних на навчальну та тестову вибірки
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Навчальні дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

# Створення об'єкта для лінійної регресії
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = regressor.predict(X_test)

# Побудова графіка з іншими стилями та кольорами
```

```
plt.figure(figsize=(8, 6))

# Зміна кольору точок (реальних даних) на блакитний
plt.scatter(X_test, y_test, color='blue', label='Реальні дані', alpha=0.6)

# Зміна кольору лінії прогнозу на червоний
plt.plot(X_test, y_test_pred, color='red', linewidth=2, label='Прогнозована
лінія')

# Додавання заголовка, міток осей, легенди та сітки
plt.title('Лінійна регресія: тест проти прогнозу', fontsize=14)
plt.xlabel('Вхідна змінна')
plt.ylabel('Цільова змінна')
plt.legend()
plt.grid(True)

# Показати графік
plt.show()

# Збереження моделі у файл за допомогою pickle
with open('linear_regressor_model.pkl', 'wb') as f:
    pickle.dump(regressor, f)

# Оцінка моделі за допомогою різних метрик
print("Результати роботи лінійної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Середньоквадратична похибка =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Корінь середньоквадратичної похибки =",
round(np.sqrt(sm.mean_squared_error(y_test, y_test_pred)), 2))  # RMSE
print("Медіанна абсолютна похибка =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Коефіцієнт поясненої дисперсії =",
round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 =", round(sm.r2_score(y_test, y_test_pred), 2))

# Прогнозування за допомогою завантаженої моделі (перевірка)
y_test_pred_loaded_model = loaded_model.predict(X_test)
print("R2 для завантаженої моделі:", round(sm.r2_score(y_test,
y_test_pred_loaded_model), 2))
```

Результат виконання:

```
Результати роботи лінійної регресії:
Середня абсолютна похибка = 0.59
Середньоквадратична похибка = 0.49
Корінь середньоквадратичної похибки = 0.7
Медіанна абсолютна похибка = 0.51
Коефіцієнт поясненої дисперсії = 0.86
R2 = 0.86
R2 для завантаженої моделі: 0.86

Process finished with exit code 0
```

Візуалізація:



Завдання 2.3. Створення багатовимірного регресора

Лістинг програми

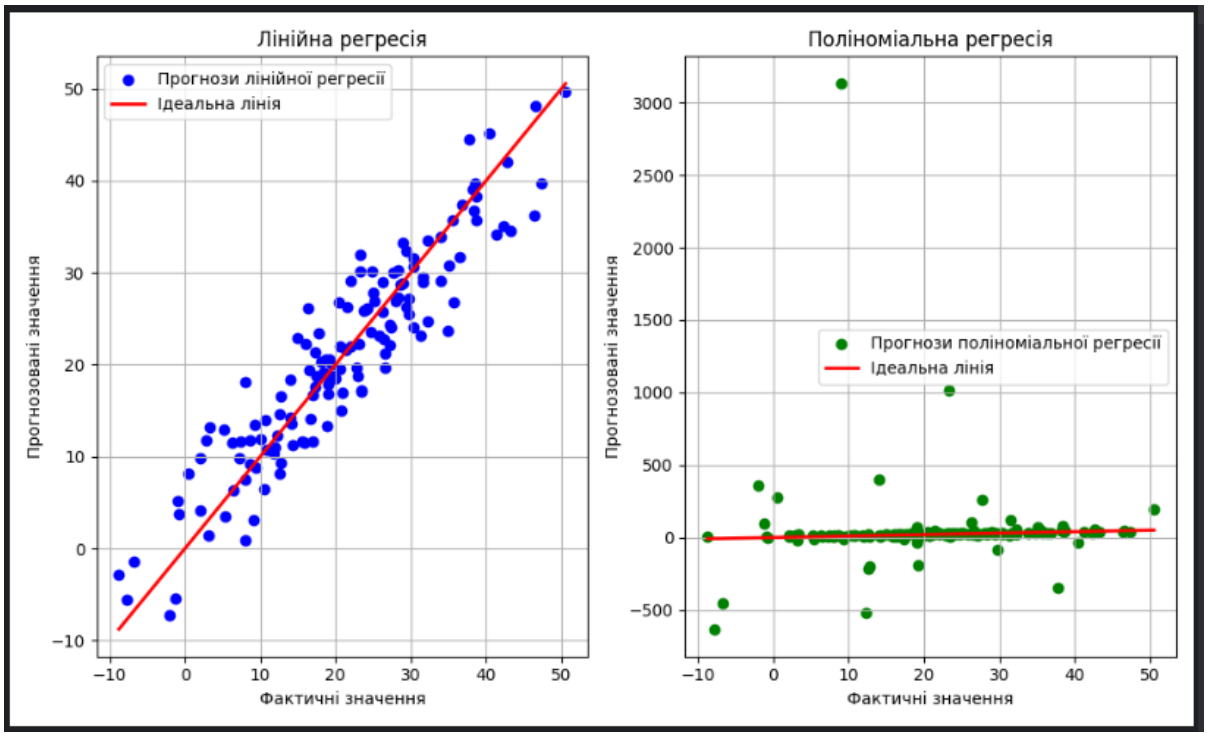
```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

input_file = 'data_multivar_regr.txt'

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

# Розбивка даних на навчальну та тестову вибірки
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Навчальні дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

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

# Прогнозування результату для лінійної регресії
y_test_pred = regressor.predict(X_test)
# Поліноміальна регресія
```

```
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
X_test_transformed = polynomial.transform(X_test)

# Створення моделі для поліноміальної регресії
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)

# Прогнозування для поліноміальної регресії
y_test_pred_poly = poly_linear_model.predict(X_test_transformed)

# Виведення результатів для лінійної та поліноміальної регресії
print("\nПрогнозування для лінійної регресії:\n", regressor.predict([[7.75, 6.35,
5.56]]))
print("\nПрогнозування для поліноміальної регресії:\n",
poly_linear_model.predict(polynomial.fit_transform([[7.75, 6.35, 5.56]])))

# Оцінка моделей
print("\nМетрики ефективності лінійної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test,
y_test_pred), 2))
print("Середньоквадратична похибка =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("R2 =", round(sm.r2_score(y_test, y_test_pred), 2))

print("\nМетрики ефективності поліноміальної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test,
y_test_pred_poly), 2))
print("Середньоквадратична похибка =", round(sm.mean_squared_error(y_test,
y_test_pred_poly), 2))
print("R2 =", round(sm.r2_score(y_test, y_test_pred_poly), 2))

# Візуалізація результатів для порівняння моделей
plt.figure(figsize=(10, 6))

# Графік для лінійної регресії
plt.subplot(1, 2, 1)
plt.scatter(y_test, y_test_pred, color='blue', label='Прогнози лінійної регресії')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linewidth=2, label='Ідеальна лінія')
plt.title('Лінійна регресія')
plt.xlabel('Фактичні значення')
plt.ylabel('Прогнозовані значення')
plt.legend()
plt.grid(True)

# Графік для поліноміальної регресії
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_test_pred_poly, color='green', label='Прогнози
поліноміальної регресії')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linewidth=2, label='Ідеальна лінія')
plt.title('Поліноміальна регресія')
plt.xlabel('Фактичні значення')
plt.ylabel('Прогнозовані значення')
plt.legend()
plt.grid(True)

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

Результат виконання:

```
Прогнозування для лінійної регресії:
 [36.05286276]

Прогнозування для поліноміальної регресії:
 [41.08248978]

Метрики ефективності лінійної регресії:
Середня абсолютна похибка = 3.58
Середньоквадратична похибка = 20.31
R2 = 0.86

Метрики ефективності поліноміальної регресії:
Середня абсолютна похибка = 67.99
Середньоквадратична похибка = 88448.73
R2 = -587.73

Process finished with exit code 0
```

Візуалізація:



Завдання 2.4. Регресія багатьох змінних

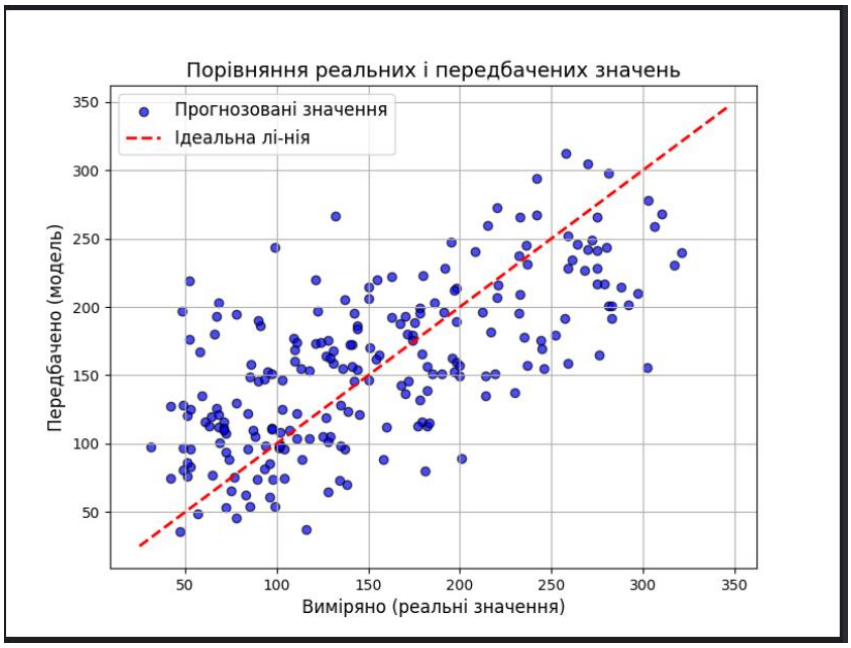
Лістинг програми:

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

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

# Розбиття на тренувальні та тестові набори
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size=0.5,
random_state=0)

# Створення лінійної моделі регресії
regr = linear_model.LinearRegression()
regr.fit(Xtrain, Ytrain)

# Прогнозування значень на тестових даних
ypred = regr.predict(Xtest)

# Виведення коефіцієнтів та інтерсепту моделі
print(f"Коефіцієнт регресії (regr.coef_): {regr.coef_}")
print(f"Інтерсепт (regr.intercept_): {regr.intercept_}")

# Оцінка моделі
print(f"R^2 (r2_score): {r2_score(Ytest, ypred)}")
print(f"Середня абсолютна похибка (mean_absolute_error):
{mean_absolute_error(Ytest, ypred)}")
print(f"Середньоквадратична похибка (mean_squared_error):
{mean_squared_error(Ytest, ypred)}")

# Візуалізація результатів
fig, ax = plt.subplots(figsize=(8, 6))  # Збільшений розмір графіка

# Графік порівняння реальних і передбачених значень
ax.scatter(Ytest, ypred, color='b', edgecolors='black', alpha=0.7,
label='Прогнозовані значення')

# Додавання ідеальної лінії (y = x) для порівняння
ax.plot([y.min(), y.max()], [y.min(), y.max()], "r--", lw=2, label='Ідеальна лі-
нія')

# Налаштування графіка
ax.set_xlabel('Виміряно (реальні значення)', fontsize=12)
ax.set_ylabel('Передбачено (модель)', fontsize=12)
ax.set_title('Порівняння реальних і передбачених значень', fontsize=14)
ax.legend(loc='best', fontsize=12)
ax.grid(True)

# Показ графіка
plt.show()
```

Результат виконання:

```
C:\Users\vanwa\PycharmProjects\pythonProject3\.venv\Scripts\python.exe C:\Users\vanwa\PycharmProjects\pythonProject3\lab4\4.4.py
Коефіцієнт регресії (regr.coef_): [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Інтерсепт (regr.intercept_): 154.3589285280134
R^2 (r2_score): 0.43774971182541
Середня абсолютна похибка (mean_absolute_error): 44.800645233553276
Середньоквадратична похибка (mean_squared_error): 3075.330688680324

Process finished with exit code 0
```

Візуалізація:



Завдання 2.5. Самостійна побудова регресії

Варіант 6

```
m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, m)
```

Лістинг коду:

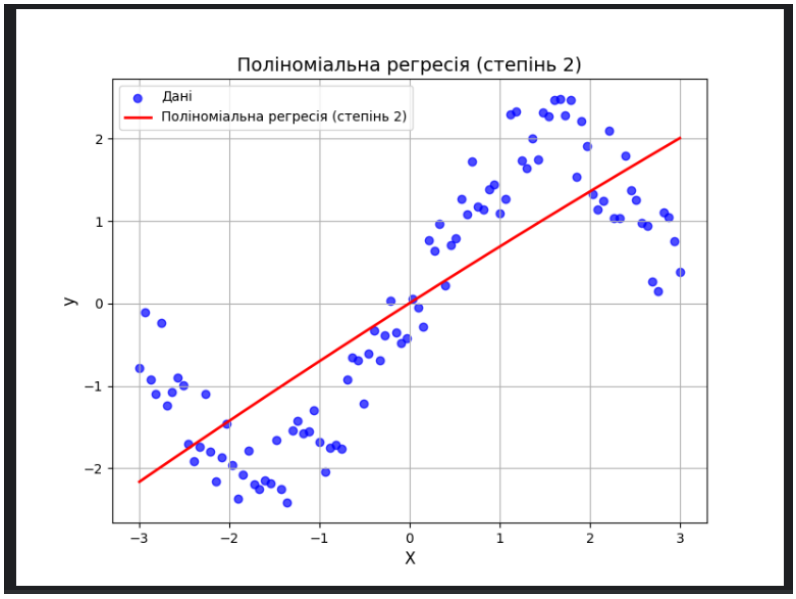
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Генерація випадкових даних
m = 100   # Кількість точок (зразків)
X = np.linspace(-3, 3, m).reshape(-1, 1)   # Перетворюємо X у 2D масив (очікуваний формат для sklearn)
y = 2 * np.sin(X).ravel() + np.random.uniform(-0.6, 0.6, m)   # y залишається як одномірний масив

# Створення поліноміальних ознак для X
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
```

```
# Виведення деяких значень для перевірки
print("X[0] =", X[0])
print("X_poly[0] =", X_poly[0])

# Лінійна регресія на поліноміальних ознаках
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)  # Навчаємо модель

# Отримуємо коефіцієнти та інтерсепт
intercept = lin_reg.intercept_
coef = lin_reg.coef_
print("Інтерсепт (Intercept):", intercept)
print("Коефіцієнти (coef):", coef)

# Прогнозування значень на нових даних
x_new = np.linspace(min(X), max(X), 100).reshape(-1, 1)  # Нові дані для прогнозу
x_new_poly = poly_features.transform(x_new)
y_new = lin_reg.predict(x_new_poly)

# Побудова графіка
plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='blue', label='Дані', alpha=0.7)
plt.plot(x_new, y_new, color='red', label='Поліноміальна регресія (степінь 2)',
linewidth=2)
plt.xlabel('X', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title('Поліноміальна регресія (степінь 2)', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

Результат виконання:

```
X[0] = [-3.]
X_poly[0] = [-3.  9.]
Інтерсепт (Intercept): -0.0014284343085682188
Коефіцієнти (coef): [0.65472565 0.00583984]

Process finished with exit code 0
```

Візуалізація:



Завдання 2. Експериментально отримані N-значень величини Y при значеннях величини X. Відшукати параметри функції за методом найменших квадратів.

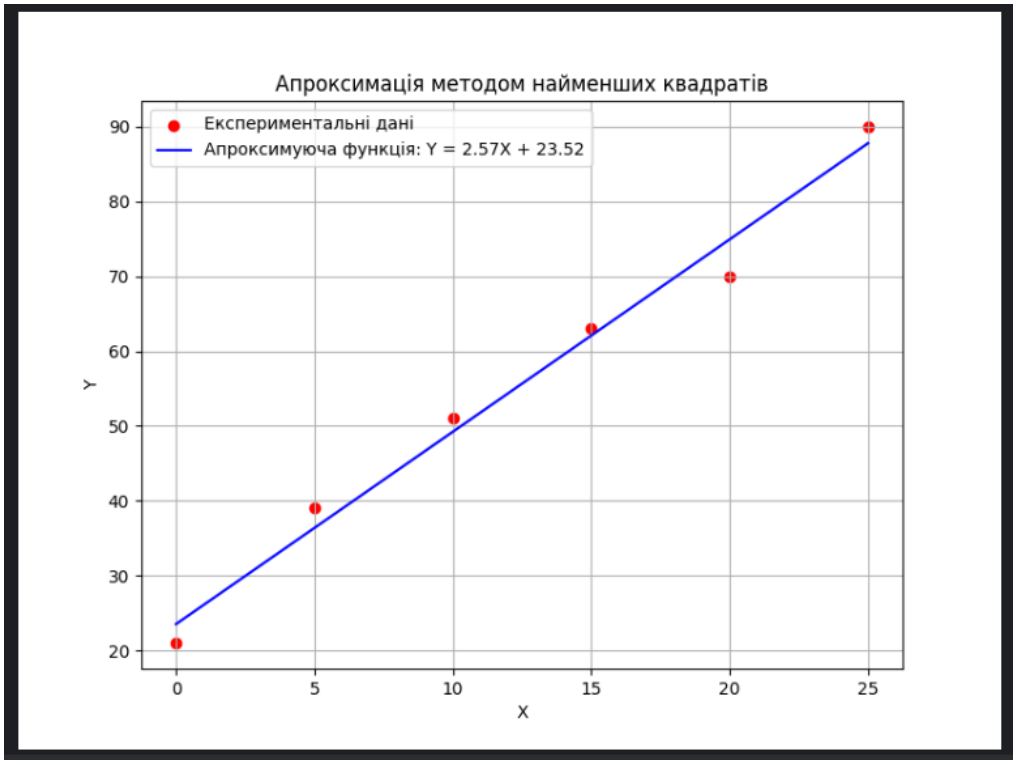
Побудувати графіки, де в декартовій системі координат нанести експериментальні точки і графік апроксимуючої функції.

1	X	0	5	10	15	20	25
	Y	21	39	51	63	70	90

Лістинг коду:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Експериментальні дані
X = np.array([0, 5, 10, 15, 20, 25])
Y = np.array([21, 39, 51, 63, 70, 90])

# Визначення лінійної функції для апроксимації
def linear_func(x, a, b):
    return a * x + b

# Знаходження параметрів функції методом найменших квадратів
params, _ = curve_fit(linear_func, X, Y)
a, b = params

# Генерація значень для побудови апроксимуючої функції
X_fit = np.linspace(min(X), max(X), 500)
Y_fit = linear_func(X_fit, a, b)

# Побудова графіку
plt.figure(figsize=(8, 6))
plt.scatter(X, Y, color='red', label='Експериментальні дані')
plt.plot(X_fit, Y_fit, color='blue', label=f'Апроксимуюча функція: Y = {a:.2f}X +
{b:.2f}')
plt.title('Апроксимація методом найменших квадратів')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid()
plt.show()
```

Результат виконання:



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.

Лістинг коду:

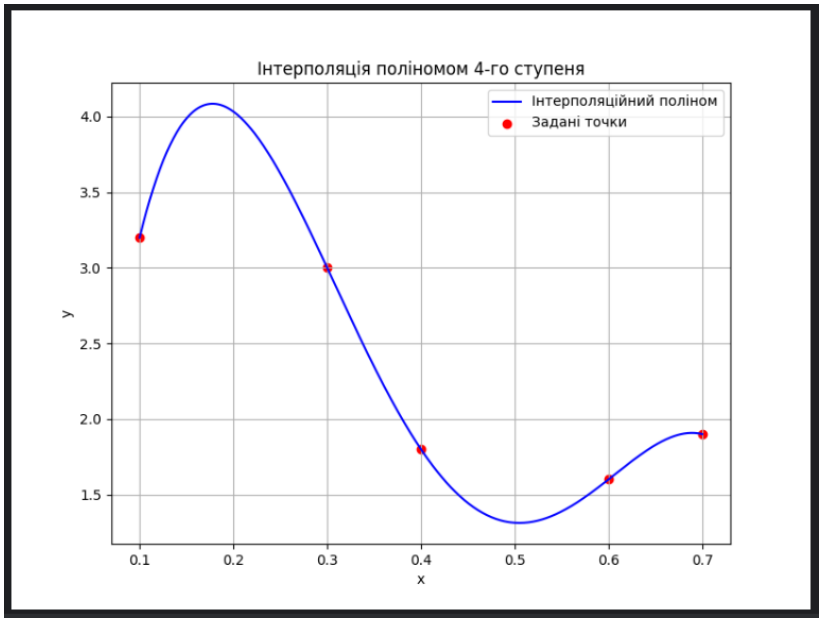
```
import numpy as np
import matplotlib.pyplot as plt

# Вхідні дані
x = np.array([0.1, 0.3, 0.4, 0.6, 0.7])
y = np.array([3.2, 3.0, 1.8, 1.6, 1.9])

# Побудова матриці Вандермонда для полінома 4-го ступеня
```

```
X = np.vander(x, increasing=True)

# Розрахунок коефіцієнтів полінома
coefficients = np.linalg.solve(X, y)

# Функція полінома на основі знайдених коефіцієнтів
def polynomial(x_value):
    return sum(c * x_value**i for i, c in enumerate(coefficients))

# Генерація точок для графіка
x_vals = np.linspace(0.1, 0.7, 500)
y_vals = [polynomial(val) for val in x_vals]

# Побудова графіка
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label="Інтерполяційний поліном", color="blue")
plt.scatter(x, y, color="red", label="Задані точки")
plt.title("Інтерполяція поліномом 4-го ступеня")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid()
plt.show()

# Обчислення значень у точках 0.2 і 0.5
y_at_0_2 = polynomial(0.2)
y_at_0_5 = polynomial(0.5)

# Виведення результатів
print(f"Значення функції у точці x=0.2: {y_at_0_2:.3f}")
print(f"Значення функції у точці x=0.5: {y_at_0_5:.3f}")
```

Результат виконання коду:

```
"C:\Program Files\Python313\python.exe"
Значення функції у точці x=0.2: 4.033
Значення функції у точці x=0.5: 1.313

Process finished with exit code 0
```

Візуалізація:



Висновок: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи регресії даних у машинному навчанні.