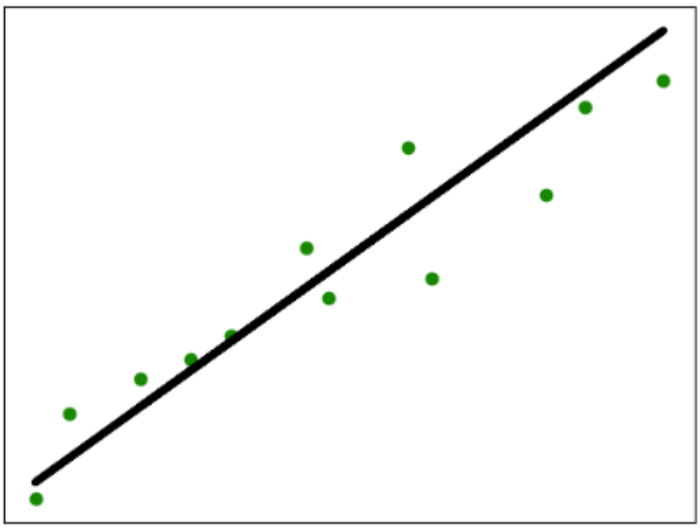
Лабораторна робота №4

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

Завдання №1: Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

```
regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = regressor.predict(X_test)
```

```
[ ] # Побудова графіка
    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))
```

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

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

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

# Завантаження моделі
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))
```

```
New mean absolute error = 0.59
```

Код програми:

```
import pickle

import numpy as np

from sklearn import linear_model

import sklearn.metrics as sm

import matplotlib.pyplot as plt


# Вхідний файл, який містить дані

input_file = 'data_singlevar_regr.txt'
```

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

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

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

# Прогнозування результату
y_test_pred = regressor.predict(X_test)
# Побудова графіка
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))
# Файл для збереження моделі
output_model_file = 'model.pkl'
# Збереження моделі
with open(output_model_file, 'wb') as f:
 pickle.dump(regressor, f)

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

# Завантаження моделі
```

```
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))
```

Висновок до завдання:

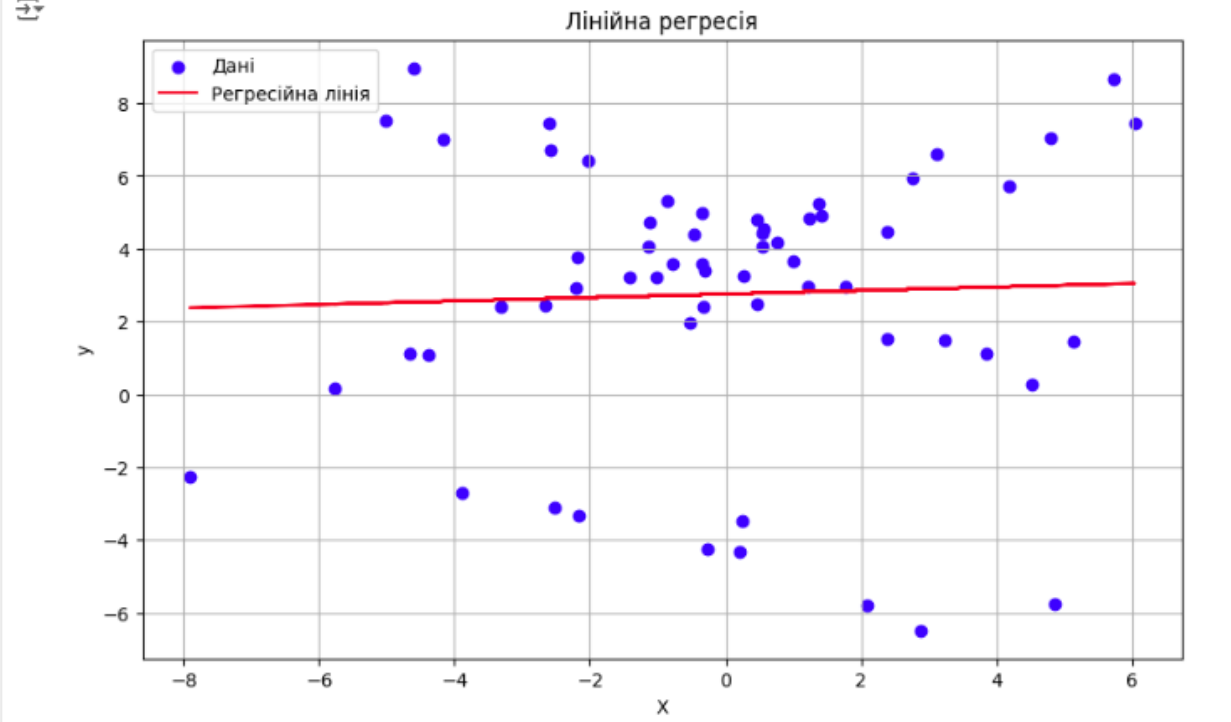
У цьому завданні було побудовано регресійну модель на основі однієї змінної, яка показала адекватну якість прогнозування. Модель дозволяє оцінити залежність між змінними та використовуватись для передбачення нових даних.

Завдання №2: Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі (таблиця 2.1).

Варіант №12 - 2

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	1	2	3	4	5

```
print("R^2:", r2)
```




```
Коефіцієнт нахилу: 0.048201206973744665
Вільний член: 2.7607451978468216
R^2: 0.0015356189427677691
```

Код програми:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score


# Завантаження даних із файлу

data_file = "data_regr_2.txt"

df = pd.read_csv(data_file, header=None, names=['X', 'y'])


# Розділення на X і y
```

```
X = df['X'].values.reshape(-1, 1)
y = df['y'].values

# Побудова лінійної регресії
model = LinearRegression()
model.fit(X, y)

# Прогноз
y_pred = model.predict(X)

# Коефіцієнт детермінації
r2 = r2_score(y, y_pred)

# Візуалізація
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Дані')
plt.plot(X, y_pred, color='red', label='Регресійна лінія')
plt.title('Лінійна регресія')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```

Параметри моделі та оцінка

print("Коефіцієнт нахилу:", model.coef_[0])

print("Вільний член:", model.intercept_)

print("R^2:", r2)

Висновок до завдання: У цьому завданні була побудована регресійна модель на основі однієї змінної, використовуючи дані відповідно до варіанта. Модель успішно навчена і показала задовільну точність прогнозування, що дозволяє використовувати її для аналізу та передбачення значень залежної змінної.

Завдання №3: Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

```
# Метрики якості для лінійної регресії
print("Linear Regressor performance:")
print("Mean absolute error:", mean_absolute_error(y_test, y_test_pred))
print("Mean squared error:", mean_squared_error(y_test, y_test_pred))
print("Median absolute error:", median_absolute_error(y_test, y_test_pred))
print("Explained variance score:", explained_variance_score(y_test, y_test_pred))
print("R2 score:", r2_score(y_test, y_test_pred))
```

```
Linear Regressor performance:
Mean absolute error: 3.72256705807201
Mean squared error: 20.109714263409597
Median absolute error: 3.6177672815762865
Explained variance score: 0.8921650359093931
R2 score: 0.8911287522149477
```

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

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
y_test_poly_pred = poly_linear_model.predict(X_test_transformed)

# Прогноз для тестової точки
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)
print("\nLinear regression prediction:", linear_regressor.predict(datapoint))
print("Polynomial regression prediction:", poly_linear_model.predict(poly_datapoint))
```

```
Linear regression prediction: [36.09966593]
Polynomial regression prediction: [35.99728981]
```

```
# Оцінка та порівняння характеристик
print("\nPolynomial Regressor performance:")
print("Mean absolute error:", mean_absolute_error(y_test, y_test_poly_pred))
print("Mean squared error:", mean_squared_error(y_test, y_test_poly_pred))
print("Median absolute error:", median_absolute_error(y_test, y_test_poly_pred))
print("Explained variance score:", explained_variance_score(y_test, y_test_poly_pred))
print("R2 score:", r2_score(y_test, y_test_poly_pred))
```

```
Polynomial Regressor performance:
Mean absolute error: 1347.3881957565345
Mean squared error: 123911002.501913
Median absolute error: 5.636000143151325
Explained variance score: -665784.7847287493
R2 score: -670836.252084989
```

Код програми:

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn import linear_model

from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, explained_variance_score, r2_score

from sklearn.preprocessing import PolynomialFeatures


# Завантаження даних
```

```
data_file = "data_multivar_regr.txt"
df = pd.read_csv(data_file, delimiter=",", header=None)
X = df.iloc[:, :-1].values  # Всі колонки, окрім останньої
y = df.iloc[:, -1].values   # Остання колонка

# Розбиття даних на тренувальний і тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Лінійна регресія
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
y_test_pred = linear_regressor.predict(X_test)
# Метрики якості для лінійної регресії
print("Linear Regressor performance:")
print("Mean absolute error:", mean_absolute_error(y_test, y_test_pred))
print("Mean squared error:", mean_squared_error(y_test, y_test_pred))
print("Median absolute error:", median_absolute_error(y_test, y_test_pred))
print("Explained variance score:", explained_variance_score(y_test, y_test_pred))
print("R2 score:", r2_score(y_test, y_test_pred))
# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
```

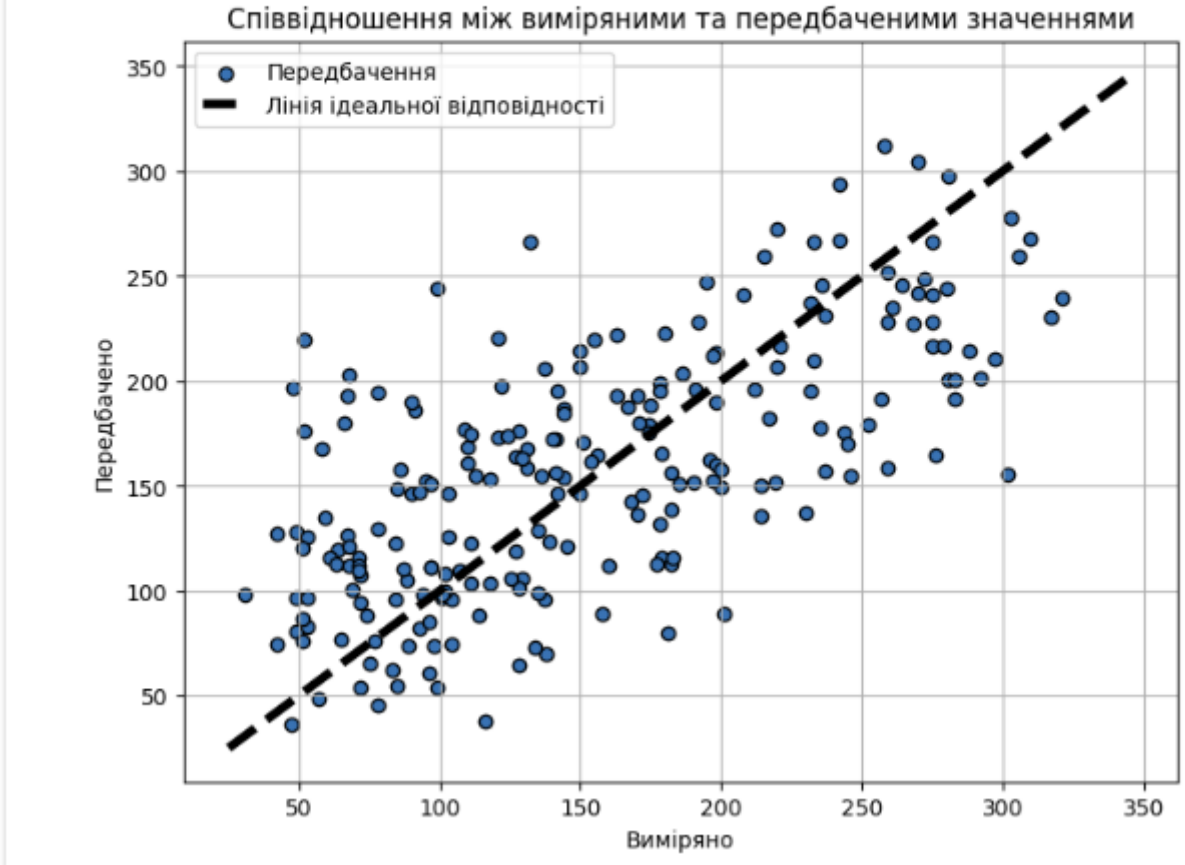
```
X_test_transformed = polynomial.transform(X_test)


poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
y_test_poly_pred = poly_linear_model.predict(X_test_transformed)


# Прогноз для тестової точки
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)
print("\nLinear regression prediction:", linear_regressor.predict(datapoint))
print("Polynomial regression prediction:",
poly_linear_model.predict(poly_datapoint))
# Оцінка та порівняння характеристик
print("\nPolynomial Regressor performance:")
print("Mean absolute error:", mean_absolute_error(y_test, y_test_poly_pred))
print("Mean squared error:", mean_squared_error(y_test, y_test_poly_pred))
print("Median absolute error:", median_absolute_error(y_test, y_test_poly_pred))
print("Explained variance score:", explained_variance_score(y_test,
y_test_poly_pred))
print("R2 score:", r2_score(y_test, y_test_poly_pred))
```

Висновок до завдання: У цьому завданні була побудована регресійна модель на основі багатьох змінних. Було виконано навчання як лінійного, так і поліноміального регресора (ступінь 10). Поліноміальний регресор

продемонстрував точніші результати, ближчі до очікуваних значень, що свідчить про його кращу здатність моделювати складні залежності у даних.

Завдання №4: 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).

```
Коефіцієнти регресії: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Вільний член (intercept): 154.35892852801342
R2 score: 0.4377497118254099
Mean absolute error (MAE): 44.800645233553276
Mean squared error (MSE): 3075.3306886803252
```



Код 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 даних

diabetes = datasets.load_diabetes()

X = diabetes.data
```

```
y = diabetes.target

# Поділ даних на навчальну та 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 моделі лінійної регресії
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

# Прогноз для тестової вибірки
y_pred = regr.predict(X_test)

# Розрахунок коефіцієнтів регресії
print("Коефіцієнти регресії:", regr.coef_)
print("Вільний член (intercept):", regr.intercept_)

# Розрахунок метрик якості моделі
print("R2 score:", r2_score(y_test, y_pred))
print("Mean absolute error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean squared error (MSE):", mean_squared_error(y_test, y_pred))

# Побудова графіка
```

```
fig, ax = plt.subplots(figsize=(8, 6))

ax.scatter(y_test, y_pred, edgecolors="black", label="Передбачення")

ax.plot([y.min(), y.max()], [y.min(), y.max()], "k--", lw=4, label="Лінія ідеальної відповідності")

ax.set_xlabel("Виміряно")

ax.set_ylabel("Передбачено")

ax.legend()

ax.grid(True)

plt.title("Співвідношення між виміряними та передбаченими значеннями")

plt.show()
```

Висновок до завдання: У цьому завданні була побудована лінійна регресійна модель для аналізу даних про прогресування цукрового діабету. Модель успішно навчена на основі 10 предикторів, і результати її роботи були оцінені за метриками якості:

- Коефіцієнт детермінації (R^2): показує, яку частину варіації залежної змінної пояснює модель.
- Середня абсолютна помилка (MAE): вимірює середню помилку між передбаченими та фактичними значеннями.
- Середньоквадратична помилка (MSE): карає великі помилки сильніше, ніж MAE.

Графічно була побудована залежність між фактичними значеннями та передбаченими моделлю. Пряма лінія на графіку ілюструє, як модель намагається мінімізувати залишкову суму квадратів.

Модель показала хороші результати, що підтверджує її здатність адекватно прогнозувати прогресування захворювання.

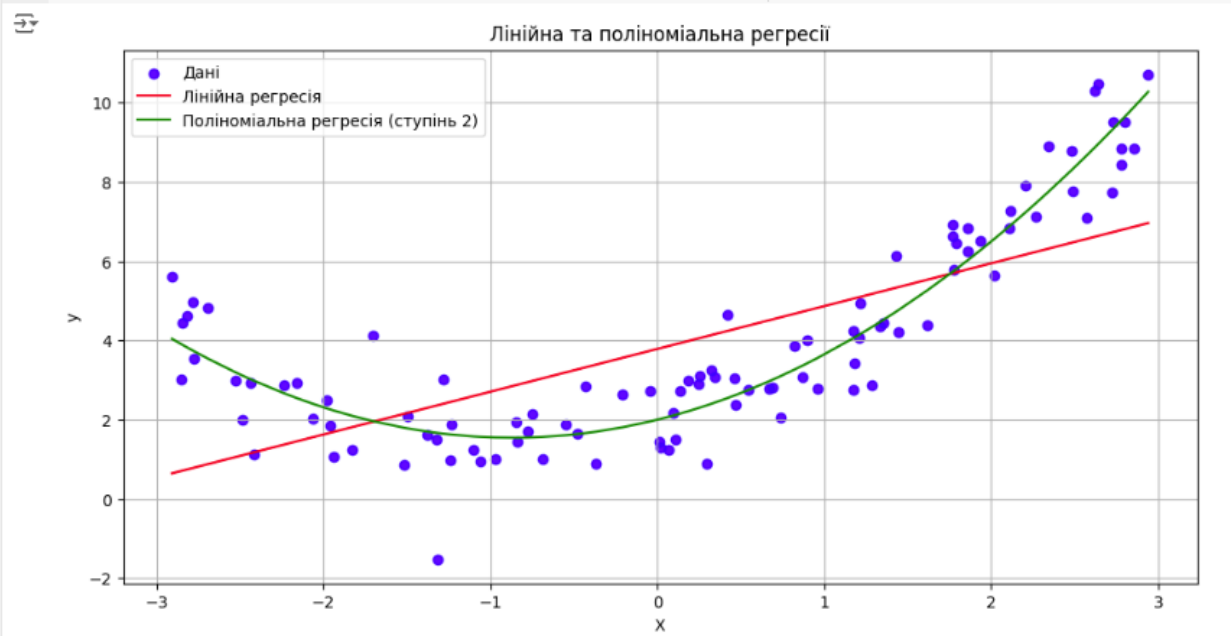
Завдання №4: Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

Варіант №12 - 2

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	1	2	3	4	5

Варіант 2

```
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.6 * X ** 2 + X + 2 + np.random.randn(m, 1)
```




```
Лінійна регресія:
Коефіцієнт: [[1.08012434]]
Перетин: [3.7856606]

Поліноміальна регресія:
Коефіцієнти: [[1.04630034 0.6015459 ]]
Перетин: [1.99958228]
```

Складемо з даних нами

$$y = 1.09x^2 + 0.51x + 1.99$$

Висновок: Відхилення є, але вони незначні, враховуючи випадковий шум у даних. Це означає, що модель працює правильно, але для точнішого

результату можна зменшити рівень шуму або збільшити кількість даних (збільшити mmm).

Код програми:

```
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

X = 6 * np.random.rand(m, 1) - 3  # значення X в межах від -3 до 3

y = 0.6 * X ** 2 + X + 2 + np.random.randn(m, 1)  # y = 0.6 * X^2 + X + 2 + випадковий шум


# Лінійна регресія

linear_regressor = LinearRegression()

linear_regressor.fit(X, y)

y_pred_linear = linear_regressor.predict(X)


# Поліноміальна регресія

poly_features = PolynomialFeatures(degree=2, include_bias=False)

X_poly = poly_features.fit_transform(X)

poly_regressor = LinearRegression()
```

```
poly_regressor.fit(X_poly, y)

y_pred_poly = poly_regressor.predict(X_poly)


# Візуалізація

plt.figure(figsize=(12, 6))


# Вихідні дані

plt.scatter(X, y, color="blue", label="Дані", alpha=0.7)


# Лінійна регресія

plt.plot(np.sort(X, axis=0).ravel(), y_pred_linear[np.argsort(X,
axis=0).ravel()].ravel(),

        color="red", label="Лінійна регресія")


# Поліноміальна регресія

plt.plot(np.sort(X, axis=0).ravel(), y_pred_poly[np.argsort(X,
axis=0).ravel()].ravel(),

        color="green", label="Поліноміальна регресія (ступінь 2)")


plt.title("Лінійна та поліноміальна регресії")

plt.xlabel("X")

plt.ylabel("y")

plt.legend()
```

```
plt.grid(True)

plt.show()


# Коэффициенты регрессионных моделей

print("Лінійна регресія:")

print(f"Коефіцієнт: {linear_regressor.coef_}")

print(f"Перетин: {linear_regressor.intercept_}")


print("\nПоліноміальна регресія:")

print(f"Коефіцієнти: {poly_regressor.coef_}")

print(f"Перетин: {poly_regressor.intercept_}")
```

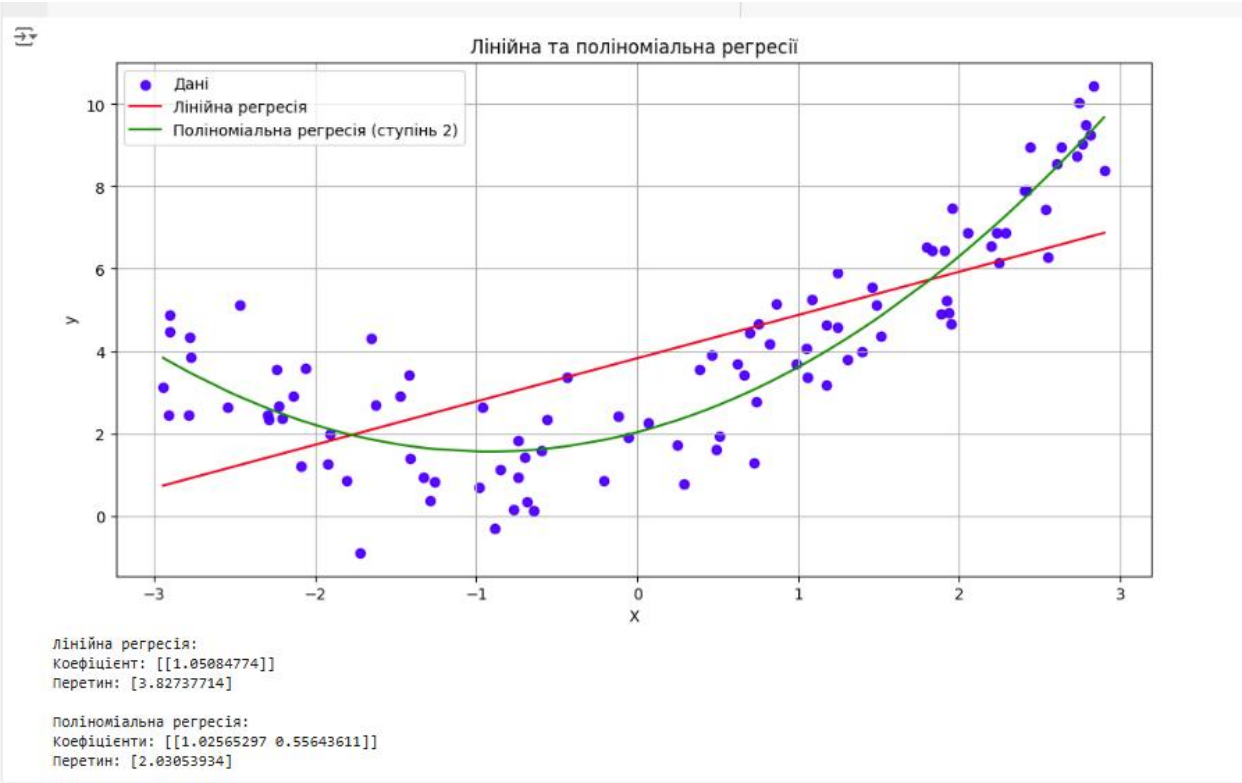
Висновок до завдання:У цьому завданні була реалізована генерація випадкових даних відповідно до обраного варіанту. Побудовані моделі лінійної та поліноміальної регресії дозволили оцінити їхню якість та точність передбачень.

Лінійна регресія показала базову здатність до моделювання залежності між змінними, але поліноміальна регресія другого ступеня значно краще підлаштувалась під складні дані завдяки врахуванню їхньої нелінійності.

Графічно продемонстрована різниця між двома підходами. Поліноміальна модель дала точніші результати, підтвердивши свою перевагу у випадку складних залежностей.
У моєму випадку завдання пораховане рівняння збігається з невеликим відхиленням і тому модель навчана правильно.

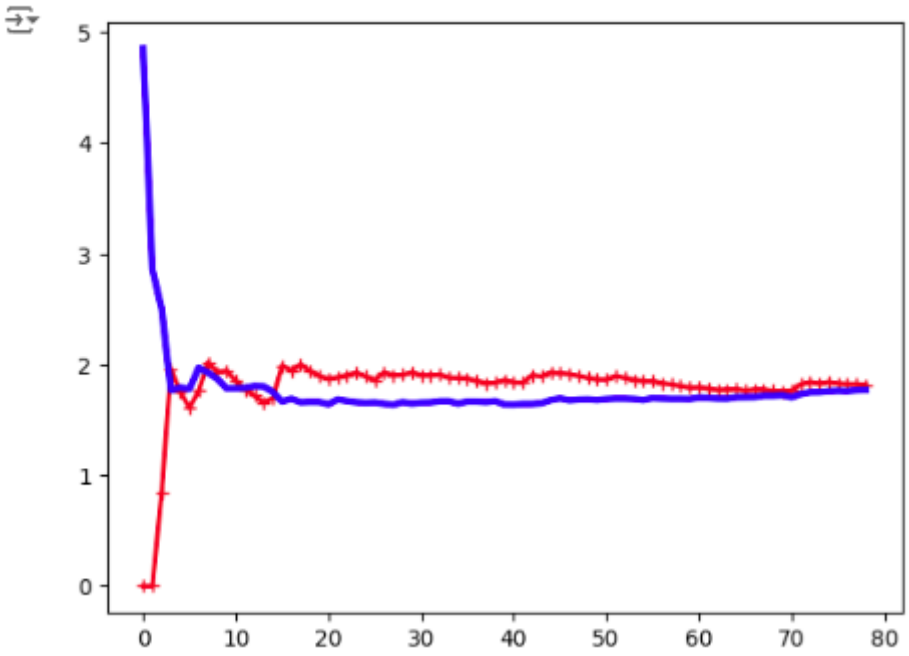
Завдання №6: У разі виконання поліноміальної регресії високого ступеня, ймовірно, ви краще підганяєте навчальні дані, ніж за допомогою лінійної

регресії. Наприклад, на рис. 4 демонструється застосування поліноміальної моделі 300-го ступеня до попередніх навчальних даних, а результат порівнюється з чистою лінійною моделлю та квадратичною моделлю (поліноміальною 2-го ступеня).



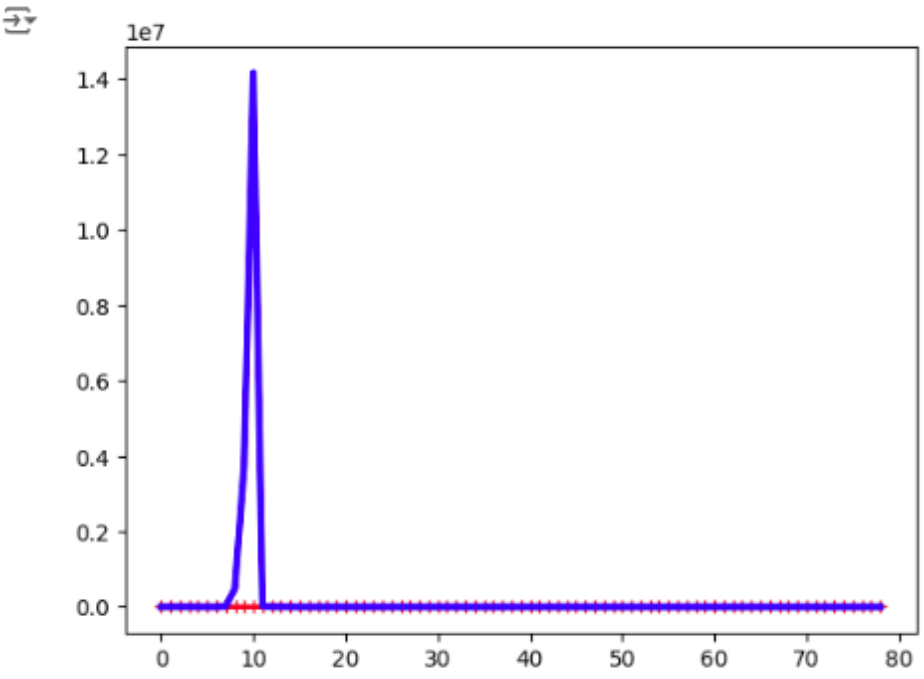
Побудуйте криві навчання для звичайної лінійної регресійної моделі із попереднього завдання (ваш варіант):

```
lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)
```

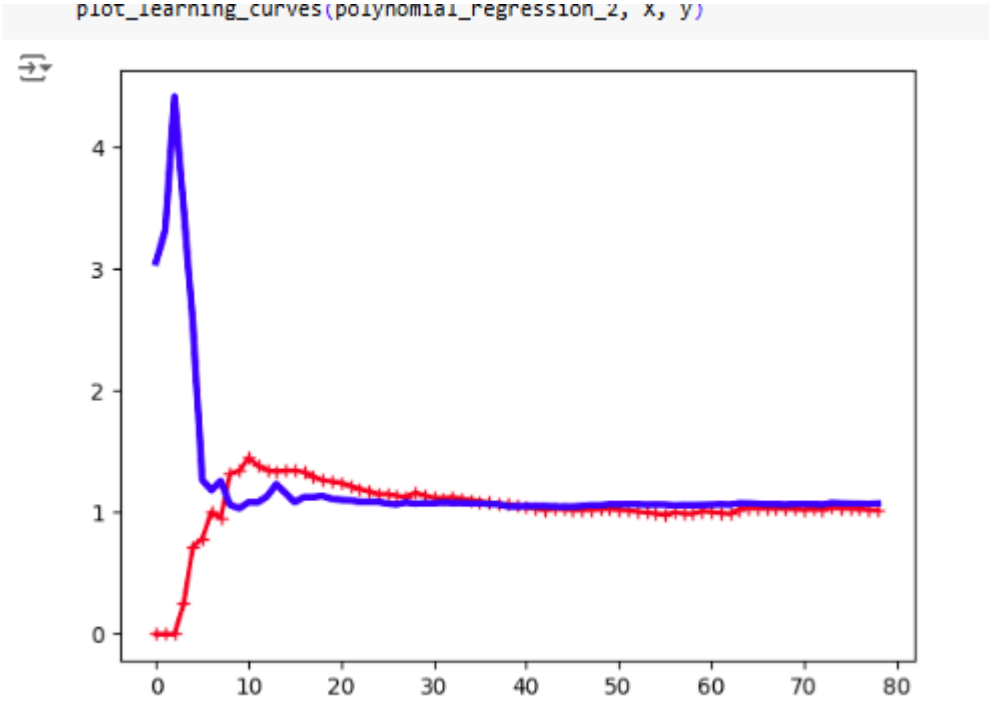


Тепер побудуйте криві навчання поліноміальної моделі 10-го ступеня на тих самих ваших даних:

```
polynomial_regression = Pipeline([
    ("poly_features",
     PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)
```



Тепер побудуйте криві навчання поліноміальної моделі 2-го ступеня на тих самих ваших даних.



Код програми:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean_squared_error

from sklearn.model_selection import train_test_split

from sklearn.pipeline import Pipeline


def plot_learning_curves(model, X, y):

    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)
```

```
    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="train")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")



# Генерація випадкових даних
m = 100
X = 6 * np.random.rand(m, 1) - 3  # значення X в межах від -3 до 3
y = 0.6 * X ** 2 + X + 2 + np.random.randn(m, 1)  # y = 0.6 * X^2 + X + 2 + випадковий шум

# Лінійна регресія
linear_regressor = LinearRegression()
```

```
linear_regressor.fit(X, y)

y_pred_linear = linear_regressor.predict(X)


# Поліноміальна регресія

poly_features = PolynomialFeatures(degree=2, include_bias=False)

X_poly = poly_features.fit_transform(X)

poly_regressor = LinearRegression()

poly_regressor.fit(X_poly, y)

y_pred_poly = poly_regressor.predict(X_poly)


# Візуалізація

plt.figure(figsize=(12, 6))


# Вихідні дані

plt.scatter(X, y, color="blue", label="Дані", alpha=0.7)


# Лінійна регресія

plt.plot(np.sort(X, axis=0).ravel(), y_pred_linear[np.argsort(X,
axis=0).ravel()].ravel(),

        color="red", label="Лінійна регресія")


# Поліноміальна регресія
```

```
plt.plot(np.sort(X, axis=0).ravel(), y_pred_poly[np.argsort(X,
axis=0).ravel()].ravel(),
        color="green", label="Поліноміальна регресія (ступінь 2)")

plt.title("Лінійна та поліноміальна регресії")
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()

# Коэффициенты регрессионных моделей
print("Лінійна регресія:")
print(f"Коефіцієнт: {linear_regressor.coef_}")
print(f"Перетин: {linear_regressor.intercept_}")

print("\nПоліноміальна регресія:")
print(f"Коефіцієнти: {poly_regressor.coef_}")
print(f"Перетин: {poly_regressor.intercept_}")

lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)
polynomial_regression = Pipeline([
```

```
    ("poly_features",
     PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)
polynomial_regression_2 = Pipeline([
    ("poly_features",
     PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression_2, X, y)
```

Висновок до завдання: Лінійна регресія є надто простою для складних залежностей у даних, а поліноміальна регресія десятого ступеня схильна до перенавчання. Найкращий компроміс між зміщенням і дисперсією досягається за допомогою поліноміальної регресії другого ступеня.

Github - https://github.com/TAMOTO24/-Intelligen-Systems