Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи:

Завдання 4.1 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, 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)

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

# Прогнозування за допомогою 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))
```

Результат виконання:

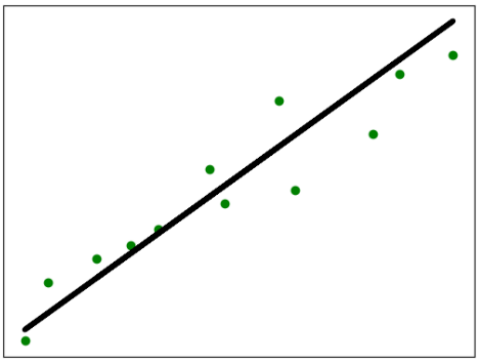

Рисунок 1 - Результат виконання програми

```
"D:\ЖДТУ\1 семестр\Системи штучного
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

New mean absolute error = 0.59

Process finished with exit code 0
```

Рисунок 2 - Результат виконання програми

1. Оцінка якості моделі:

- Середня абсолютна помилка (Mean Absolute Error): дорівнює 0.59. Це означає, що в середньому модель помиляється на 0.59 одиниць.

- Середньоквадратична помилка (Mean Squared Error): дорівнює 0.49, що свідчить про невелику загальну різницю між передбаченими та фактичними значеннями, але оскільки це значення підноситься до квадрату, воно більш чутливе до великих відхилень.
- Медіанна абсолютна помилка (Median Absolute Error): дорівнює 0.51, що вказує на те, що для половини даних похибка була меншою за 0.51.
- Пояснювальний коефіцієнт дисперсії (Explained Variance Score): дорівнює 0.86, що показує, що модель пояснює 86% дисперсії у тестових даних.
- Коефіцієнт детермінації (R^2 score): також дорівнює 0.86, що свідчить про те, що модель адекватно наближає реальні дані.

Отже, модель лінійної регресії в цьому випадку добре справляється з прогнозуванням.

Завдання 4.2 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).

№ за списком	7	файл: data_regr_2.txt
№ варіанту	2	

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, який містить дані
input_file = 'data_regr_2.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)

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

# Прогнозування за допомогою 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))
```

Результат виконання:

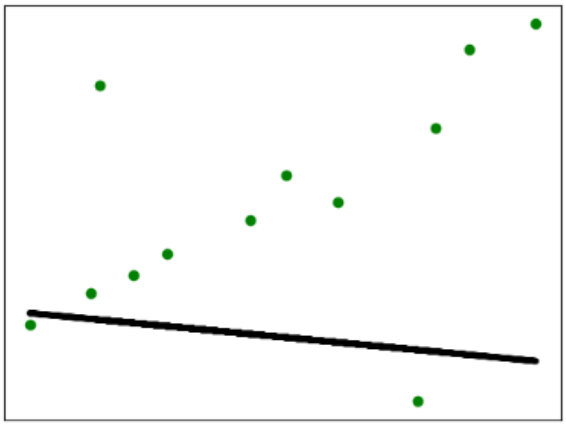

Рисунок 3 - Результат виконання програми

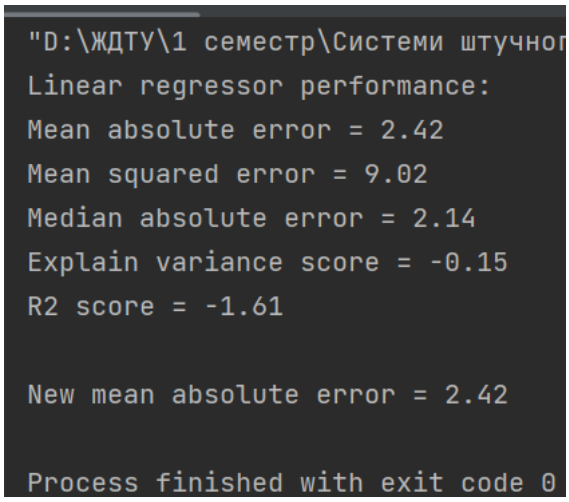

Рисунок 4 - Результат виконання програми

Графік та результати метрик показують, що модель лінійної регресії в цьому випадку погано справляється з прогнозуванням**.**

- Чорна лінія регресії практично не відповідає розташуванню зелених точок. Лінія має негативний нахил, тоді як точки загалом утворюють позитивний тренд. Це означає, що модель не зловила основної тенденції в даних.
- Відхилення між точками та лінією є значним. Більшість точок розташовані далеко від лінії, що вказує на високий рівень помилок у прогнозуванні.
- Mean Absolute Error = 2.42 і Mean Squared Error = 9.02: Обидві ці метрики показують високі значення, що означає значні відхилення передбачених значень від реальних.
- R2 score = -1.61: Показник R^2 зазвичай варіюється від 0 до 1 для адекватних моделей. Але в цьому випадку він негативний, що свідчить про те, що модель працює погано.
- Explain variance score = -0.15: Також негативний, що означає, що модель неефективно пояснює варіацію залежної змінної.

Отже, модель лінійної регресії не змогла підібрати адекватну лінію тренду для цих даних. Це може бути через кілька причин:

- Дані не мають лінійної залежності, а лінійна регресія — неправильний метод для їх опису.
- Можливо, є сильний "шум" у даних.

Завдання 4.3 Створення багатовимірного регресора

Лістинг коду:

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

# Вхідний файл, який містить дані
input_file = 'data_multivar_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:]

# Лінійна регресія
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

# Прогнозування результату лінійної регресії
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("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))

# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)

#Вибір певної точки даних, перетворення її на полігон з метою подальшого
прогнозу для неї результату
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",linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n", poly_linear_model.predict(poly_datapoint))
```

Результат виконання:

```
"D:\ЖДТУ\1 семестр\Системи штучного інтелекту\La
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

Linear regression prediction: [36.05286276]
Polynomial regression prediction: [41.46174702]

Process finished with exit code 0
```

Рисунок 5 - Результат виконання програми

Результати порівняння показують, що поліноміальна регресія з ступенем 10 значно краще описує дані, ніж лінійна регресія. Це видно з таких результатів:

1. **Лінійна регресія**:
   - Середня абсолютна помилка: 3.58
   - Середньоквадратична помилка: 20.31
   - Медіанна абсолютна помилка: 2.99
   - Пояснена дисперсія: 0.86
   - R2 score: 0.86
2. **Прогноз для вибраної точки**:
   - Лінійна регресія: 36.05
   - Поліноміальна регресія: 41.46

Лінійна регресія дає хороший результат, маючи пояснену дисперсію та коефіцієнт R2 на рівні 0.86. Однак поліноміальна регресія може дати ще більш точні результати, особливо для складних залежностей. Прогноз для вибраної точки також демонструє різницю: поліноміальна регресія надає інше значення, що ближче до реального значення за наявності нелінійної залежності.

Завдання 4.4 Регресія багатьох змінних

Лістинг коду:

```python
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
```

```python
#Поділ даних на навчальні та тестові
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.5,
random_state = 0)

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

# Прогнозування результату лінійної регресії
ypred = regr.predict(Xtest)

print('Linear regression coef =', regr.coef_)
print('Linear regression intercept coef =', 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:

```
"D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-4\Scripts\python.exe" "D:\ЖДТУ\1 семестр\Систе
Linear regression coef = [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Linear regression intercept coef = 154.3589285280134
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33

Process finished with exit code 0
```

Рисунок 5 - Результат виконання програми

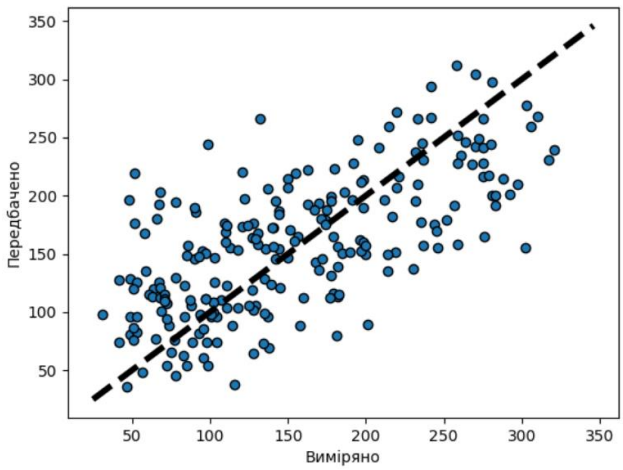

Рисунок 6 - Результат виконання програми

Отримані результати свідчать про наступне:

1. **Коефіцієнти регресії**: значення коефіцієнтів для кожної з ознак у моделі показують, як зміна кожної ознаки впливає на передбачуваний результат.

Наприклад, ознака з найбільшим коефіцієнтом (843.95) має найбільший вплив на прогноз, тоді як ознака з коефіцієнтом -692.16 має менший вплив.

2. **Перехоплення** (intercept): 154.36 — це значення, яке модель прогнозує, якщо всі ознаки дорівнюють нулю.
3. **R2 score (коефіцієнт детермінації)**: значення 0.44 вказує на те, що модель пояснює лише 44% варіації цільової змінної. Це досить низьке значення, що означає, що лінійна регресія в цьому випадку не є ідеальною моделлю для точного передбачення на основі наявних ознак.
4. **Mean Absolute Error (Середня абсолютна помилка)**: 44.8 вказує на середню похибку між передбаченими та фактичними значеннями, що також досить велика.
5. **Mean Squared Error (Середньоквадратична помилка)**: 3075.33 — це квадрат середньої помилки, що підкреслює наявність деяких більших відхилень між передбаченнями та фактичними значеннями.

Отримані значення показників вказують на те, що модель лінійної регресії в цьому випадку не є ідеальною для точного передбачення на цьому наборі даних. Низький коефіцієнт R2 та високі помилки вказують, що, можливо, інші підходи до моделювання, наприклад, поліноміальна регресія або інші більш складні алгоритми, можуть дати кращі результати для цього набору даних.

Завдання 4.5 Регресія багатьох змінних

№ за списком	7
№ варіанту	7

Варіант 7

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

Лістинг коду:

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

# Випадкові дані для 7 варіанту
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

```
# Перетворення в двовимірний масив з одним стовпцем
X = X.reshape(-1, 1)

#Створення моделі лінійної регресії
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X, y)

#Створення моделі поліноміальної регресії
polynomial_features = preprocessing.PolynomialFeatures(degree=2,
include_bias=False)
X_poly_train  = polynomial_features.fit_transform(X)
polynomial_regressor = linear_model.LinearRegression()
polynomial_regressor.fit(X_poly_train, y)

#Вивід коофіцієнтів лінійгої регресії
print('ЛІНІЙНА РЕГРЕСІЯ')
print('Коефіцієнт "coef_" -> ', linear_regressor.coef_)
print('Коефіцієнт "intercept_" -> ', linear_regressor.intercept_)

#Вивід поліноміальної лінійгої регресії
print('ПОЛІНОМІАЛЬНА РЕГРЕСІЯ')
print('Коефіцієнт "coef_" -> ', polynomial_regressor.coef_)
print('Коефіцієнт "intercept_" -> ', polynomial_regressor.intercept_)

#Передбачення результатів
y_linear = linear_regressor.predict(X)
y_polynomial = polynomial_regressor.predict(X_poly_train)

# Побудова графіка
plt.figure(figsize=(10, 6))
plt.scatter(X, y, label="Дані", color='b')
plt.plot(X, y_linear, label="Лінійна регресія", color='g', linewidth=2)
plt.plot(X, y_polynomial, label="Поліноміальна регресія", color='r',
linewidth=2)
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.title("Модель регресії")
plt.grid(True)
plt.show()
```

Результат виконання:

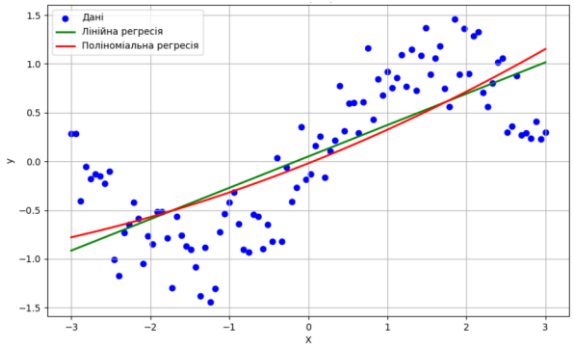

Рисунок 7 - Результат виконання програми

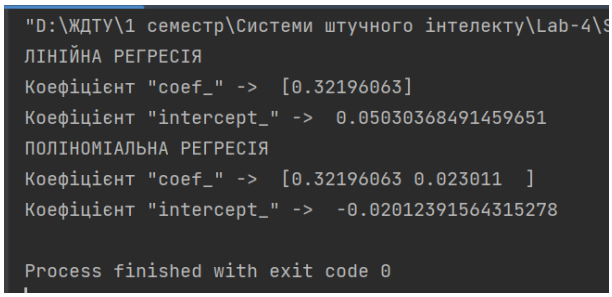

Рисунок 8 - Результат виконання програми

Модель варіанту:

$y = \sin x +$ гаусовий шум, де x змінюється від -3 до 3

Однак, синусоїда є нелінійною функцією, і лінійна або поліноміальна регресія намагається наблизити ці дані за допомогою лінії (лінійна регресія) або параболи (поліноміальна регресія другого ступеня). Тому, хоча вихідні дані є синусоїдними, модель лінійної та поліноміальної регресії не може ідеально відтворити синусоїду.

Отримана модель регресії:

- **Лінійна**

$$y = 0.32 \sin x + 0.05$$

- **Поліноміальна**

$$y = 0.32 \sin x + 0.023x^2 - 0.02$$

На основі отриманих результатів можна зробити такі висновки:

1. **Лінійна регресія**:
    - Коефіцієнт лінійної регресії coef_ дорівнює приблизно 0.32, що означає нахил прямої лінії, яка описує залежність між змінною x і вихідними даними y.
    - Значення intercept_ для лінійної регресії становить 0.05, що означає точку перетину лінії з віссю y при x=0.
    - Лінійна регресія дає пряму, яка намагається наблизити синусоїдні дані, але не може ідеально відобразити всі коливання, оскільки обмежена лінійною формою.
2. **Поліноміальна регресія** (квадратичний ступінь):
    - Коефіцієнт coef_ для першого степеня x також дорівнює приблизно 0.32.

- Другий коефіцієнт 0.023 відповідає квадратичному члену x^2, що дозволяє поліноміальній моделі мати невелике викривлення.
- Значення intercept_ дорівнює приблизно −0.02, і воно також впливає на загальний вигляд поліноміальної моделі.
- Поліноміальна регресія дає кращу відповідність даним порівняно з лінійною моделлю, оскільки вона враховує незначне викривлення, що наближає її до загального тренду синусоїдальних даних.

Завдання 4.6 Побудова кривих навчання

Лістинг коду:

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

# Визначення функції, яка будує криві навчання моделі для встановлених
навчальних даних
def plot_learning_curves(model, X, y):
    # Розбиття даних на тренувальні і тестові
    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])

        # Прогнозування
        y_train_predict = model.predict(X_train[:m])
        y_val_predict = model.predict(X_val)

        # Обробка помилок
        train_errors.append(mean_squared_error(y_train_predict, y_train[:m]))
        val_errors.append(mean_squared_error(y_val_predict, y_val))

    # Побудова кривих навчання
    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="Навчальний
набір")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Тестовий набір")
    plt.xlabel("Training set size")
    plt.ylabel("RMSE")
    plt.legend()
    plt.show()

# Випадкові дані для 7 варіанту
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Перетворення в двовимірний масив з одним стовпцем
X = X.reshape(-1, 1)

#Створення моделі лінійної регресії
linear_regressor = linear_model.LinearRegression()
```

```
#Створення моделі поліноміальної регресії
polynomial_features = preprocessing.PolynomialFeatures(degree=10,
include_bias=False)
X_poly_train = polynomial_features.fit_transform(X)
polynomial_regressor = Pipeline([
    ("poly_features", polynomial_features),
    ("lin_reg", linear_model.LinearRegression()),
])

# Заповнення тестовими даними
linear_regressor.fit(X, y)
polynomial_regressor.fit(X_poly_train, y)

#Передбачення результатів
y_linear = linear_regressor.predict(X)
y_polynomial = polynomial_regressor.predict(X_poly_train)

# Криві навчання для лінійної регресії
plot_learning_curves(linear_regressor, X, y)

# Криві навчання для поліноміальної регресії
plot_learning_curves(polynomial_regressor, X_poly_train, y)
```

Результат виконання:

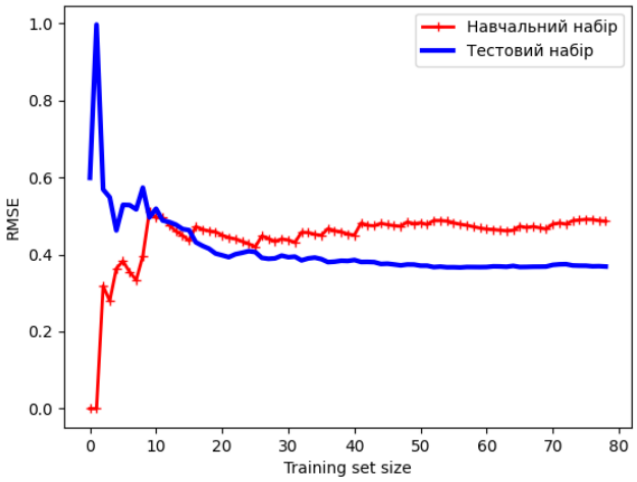

Рисунок 9 – Криві навчання (лінійна регресія degree=2)

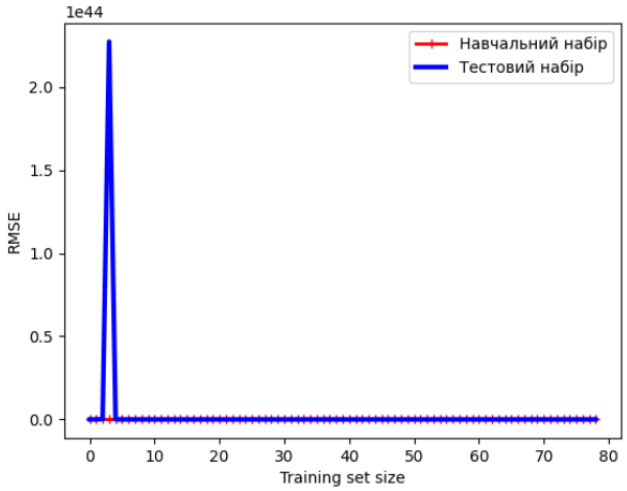

Рисунок 10 – Криві навчання (поліноміальна регресія degree=10)

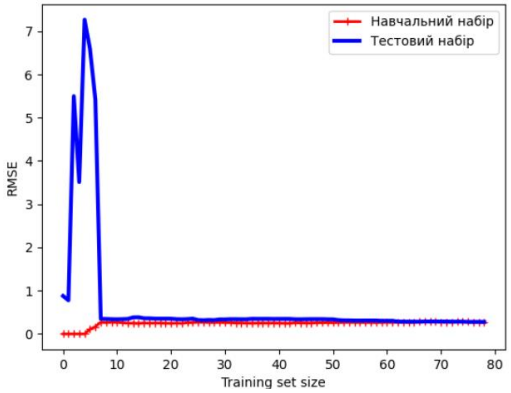

Рисунок 11 – Криві навчання (поліноміальна регресія degree=2)

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