ЛАБОРАТОРНА РОБОТА № 3

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

Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

```python
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 = 'D:\LabsNew\Fourth Course\Second semester\Системи штучного інтелекту\Solutions\Actual\Lab3\Task\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()
```

Рис. 1. Код програми

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

Рис. 2. Код програми

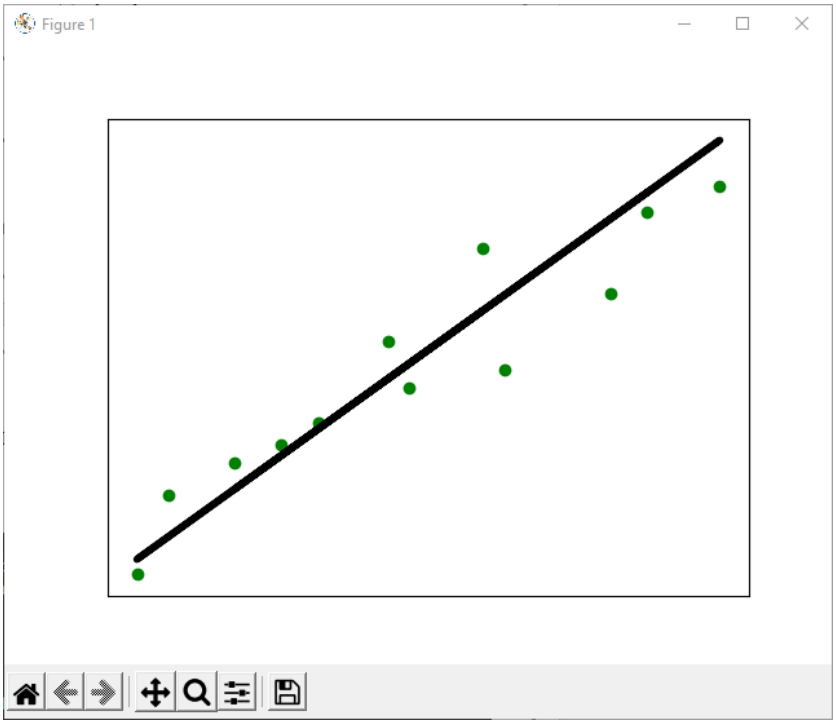

Рис. 3. Результат виконання програми

```
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
```

Рис. 4. Результат виконання програми

Проаналізувавши результати оцінки якості на рис. 4, можемо зробити наступні висновки:

1. Середня абсолютна помилка – 0.59 є оптимальною
2. Середня квадратична помилка – 0.49 показує, що ця модель має середню кількість грубих помилок
3. Середня абсолютна помилка – 0.51 є середнім показником
4. Оцінка регресії – 0.86 є близьким до 1, це означає, що модель добре пояснює дані
5. У збереженій моделі середня абсолютна помилка така ж – 0.59

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.

Файл за варіантом: data_regr_4.txt

```
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 = 'D:\LabsNew\Fourth Course\Second semester\Системи штучного інтелекту\Solutions\Actual\Lab3\Task\data_regr_4.txt'

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

Рис. 5. Код програми

Код програми є аналогічним до попереднього завдання, тільки на початку вказується інший сорс-файл.

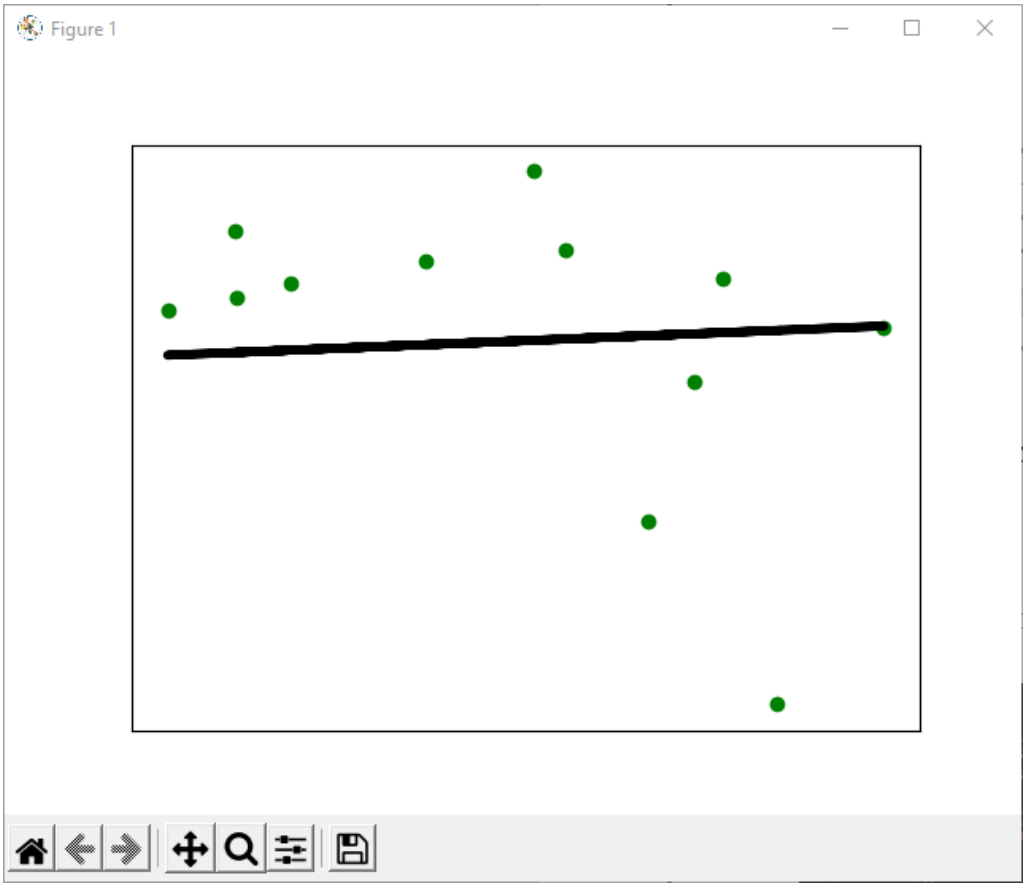

Рис. 6 Результат виконання програми

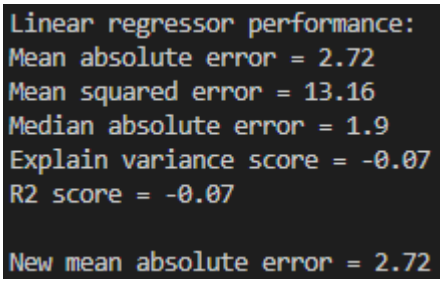

Рис. 7. Результат виконання програми

Проаналізувавши результати оцінки якості на рис. 7, можемо зробити наступні висновки:

1. Середня абсолютна помилка – 2.72 є доволі високою
2. Середня квадратична помилка – 13.16 показує, що ця модель має велику кількість грубих помилок
3. Середня абсолютна помилка – 1.9 є середнім показником
4. Оцінка регресії – -0.07 означає, що якість регресії низька
5. Коефіцієнт детермінації - -0.07 є дуже поганим показником
6. У збереженій моделі середня абсолютна помилка така ж – 2.72

З цих даних можемо зробити висновок, що ця модель не є оптимальною

Завдання 2.3. Створення багатовимірного регресора

Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

```python
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 = 'Task/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:]

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

# Прогнозування результату
y_test_pred = regressor.predict(X_test)

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

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

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

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
print("\nLinear regression: ", regressor.predict(datapoint))
print("Polynomial regression: ", poly_linear_model.predict(poly_datapoint))
```

Рис. 8. Код програми

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

Linear regression:  [36.05286276]
Polynomial regression:  [41.46504705]
```

Рис. 9. Результат виконання програми

За оцінкою якості можна зробити висновок, що регресія є доволі якісною, але має грубі помилки. Поліноміальна регресія показала більш точний результат, аніж лінійна.

Завдання 2.4. Регресія багатьох змінних

Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets.

```python
import matplotlib.pyplot as plt
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 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 тренування
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

# Прогноз
y_pred = regr.predict(X_test)

print("Regression coefficient", regr.coef_)
print("Regression intercept", regr.intercept_)
print("R2 score =", round(r2_score(y_test, y_pred), 2))
print("Mean absolute error =", round(mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error =", round(mean_squared_error(y_test, y_pred), 2))

fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, 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()
```

Рис. 10. Код програми

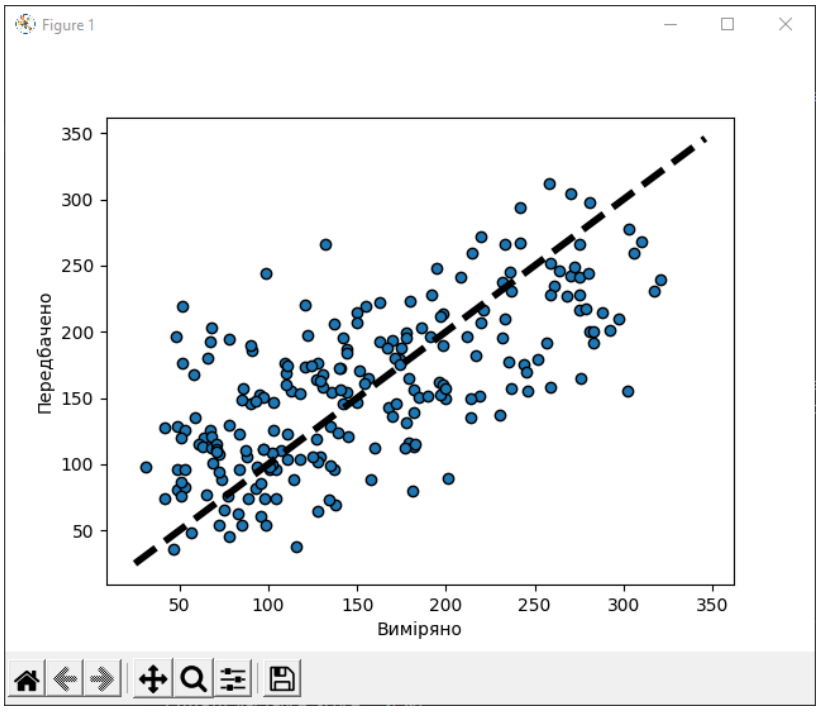

Рис. 11. Результат виконання програми

```
Regression coefficient [ -20.41129305 -265.88594023  564.64844662  325.55650029 -692.23796104
  395.62249978   23.52910434  116.37102129  843.98257585   12.71981044]
Regression intercept 154.35898821355153
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```

Рис. 12. Результат виконання програми

За отриманими показниками можна зробити висновок, що є досить сильний взаємозв'язок між незалежною та залежною змінними (високі показники регресії); середнє значення змінної – 154.36; коефіцієнт детермінації – 0.44 є не дуже високим показником, а отже модель пояснює дані на есредньому рівні; високий показник грубих помилок.

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.

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

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

X = X.reshape((m, 1))

# Створення об'єкта лінійного регресора
linear_regression = LinearRegression()
linear_regression.fit(X, y)


# Побудова графіка лінійної регресії
plt.scatter(X, y, color='green')
plt.plot(X, linear_regression.predict(X), color='black', linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()

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

poly_regression = LinearRegression()
poly_regression.fit(X_poly, y)

print("X[0]", X[0])
print("X_poly", X_poly)
print("Coefficients", poly_regression.coef_)
print("Intercept", poly_regression.intercept_)

# Впорядковуємо точки по осі X
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape(len(X_grid), 1)

# Побудова графіка поліноміальної регресії
plt.scatter(X, y, color='green')
plt.plot(X_grid, poly_regression.predict(poly_features.fit_transform(X_grid)), color='black', linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()
```

Рис. 13. Код програми

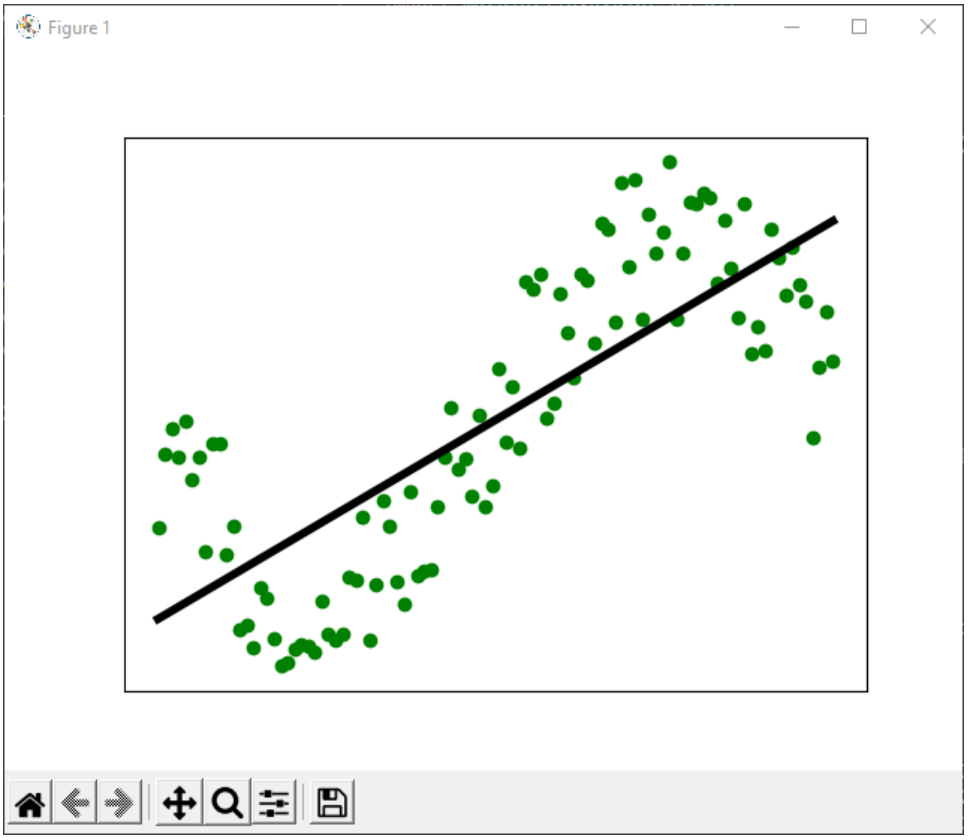

Рис. 14. Графік лінійної регресії

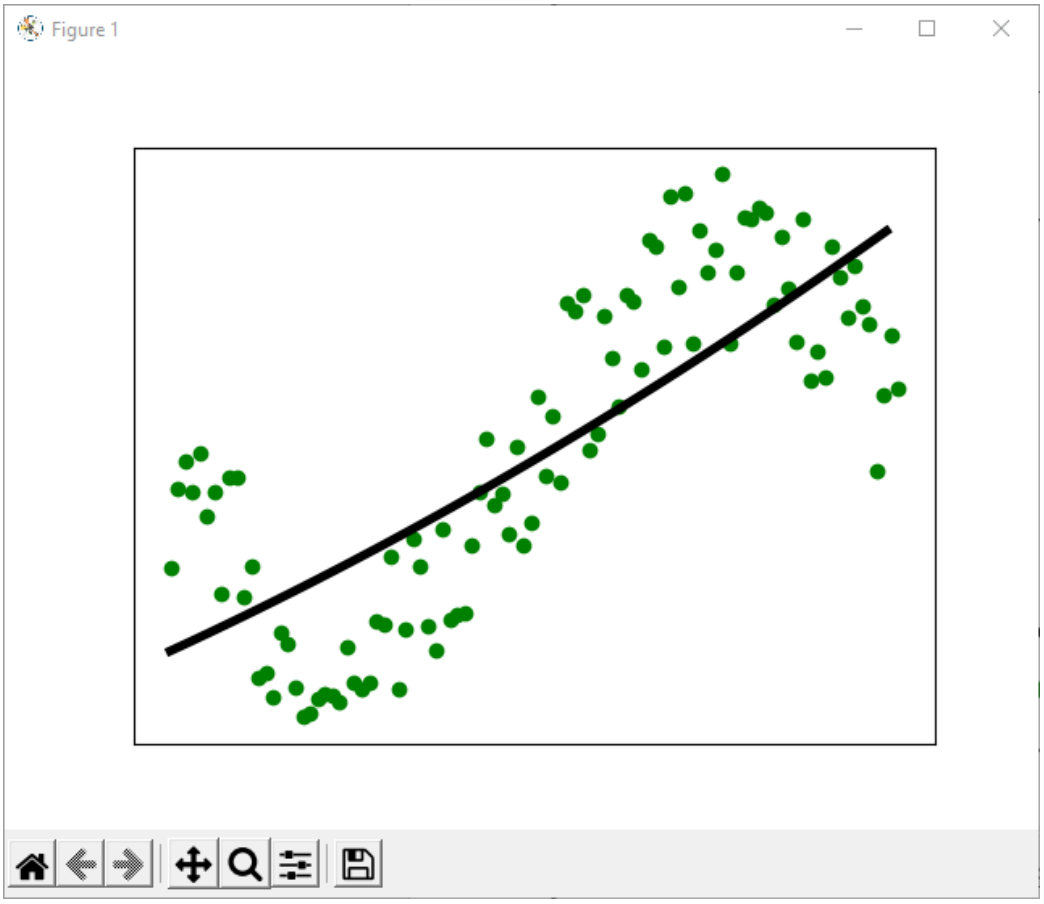

Рис. 15. Графік поліноміальної регресії

По графіках видно, що поліноміальна регресія показує трохи точніші, проте не набагато, результати.

Модель у вигляді рівняння:

Y = 3 + sin(x) + шум

Отримані коефіцієнти:

```
Coefficients [0.32314122 0.00613376]
Intercept 2.9416625140602286
```

Рис. 16. Отримані коефіцієнти

Модель регресії:

Y = 2.94 + sin(0.32x)

Отримані коефіцієнти не близькі до модельних, що означає, що модель навчена неправильно.

Завдання 2.6. Побудова кривих навчання.

Побудуйте криві навчання для ваших даних у попередньому завданні

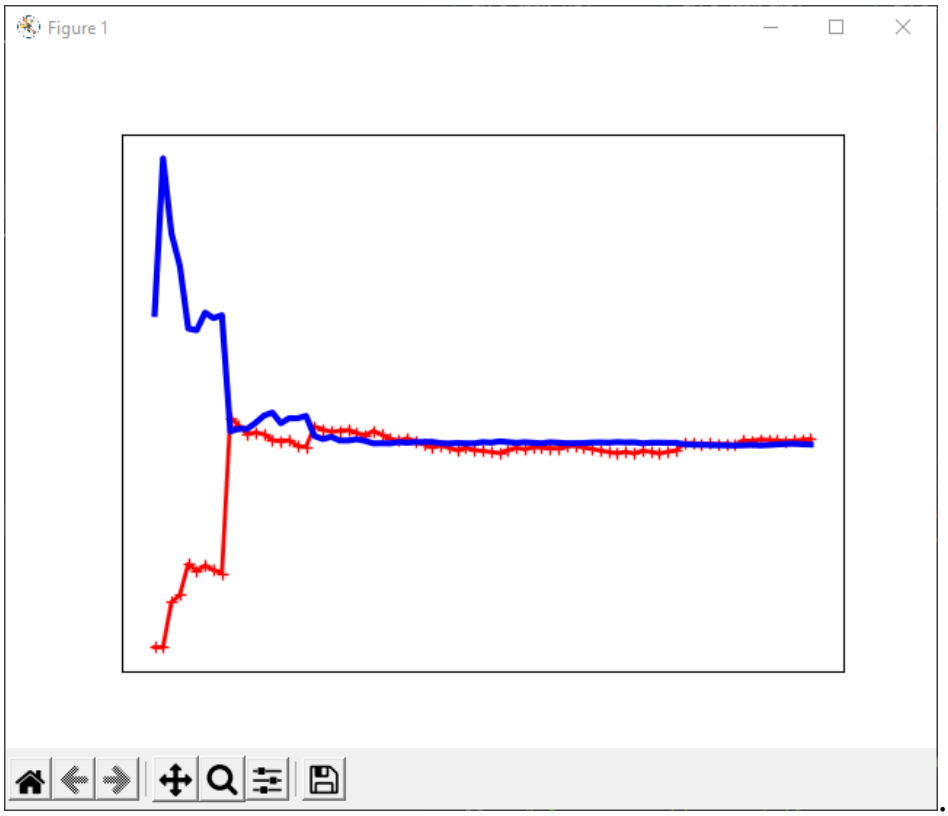

Рис. 17. Криві навчання для лінійної моделі

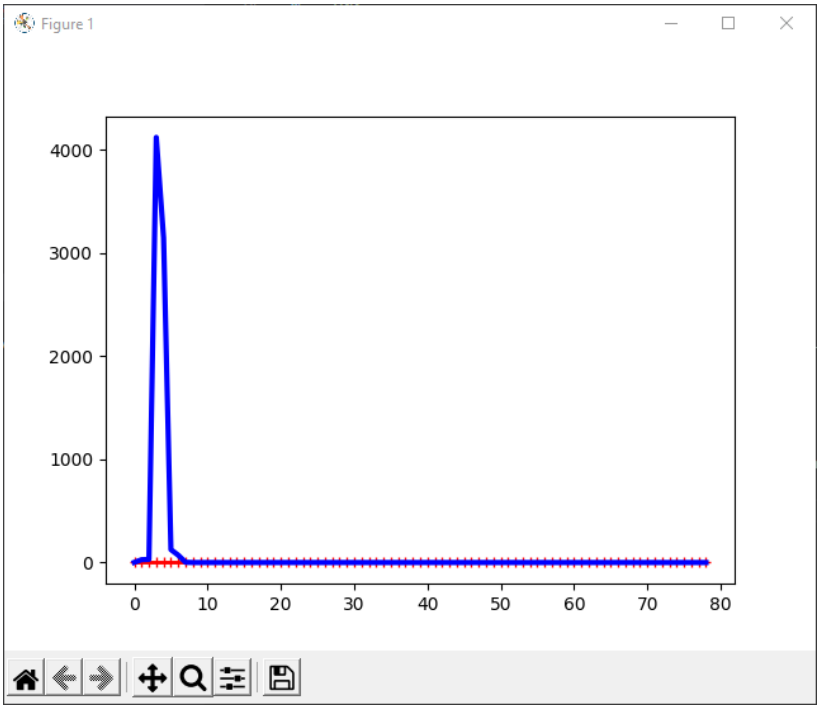

Рис. 18. Криві навчання для поліноміальної моделі 10-го ступеня

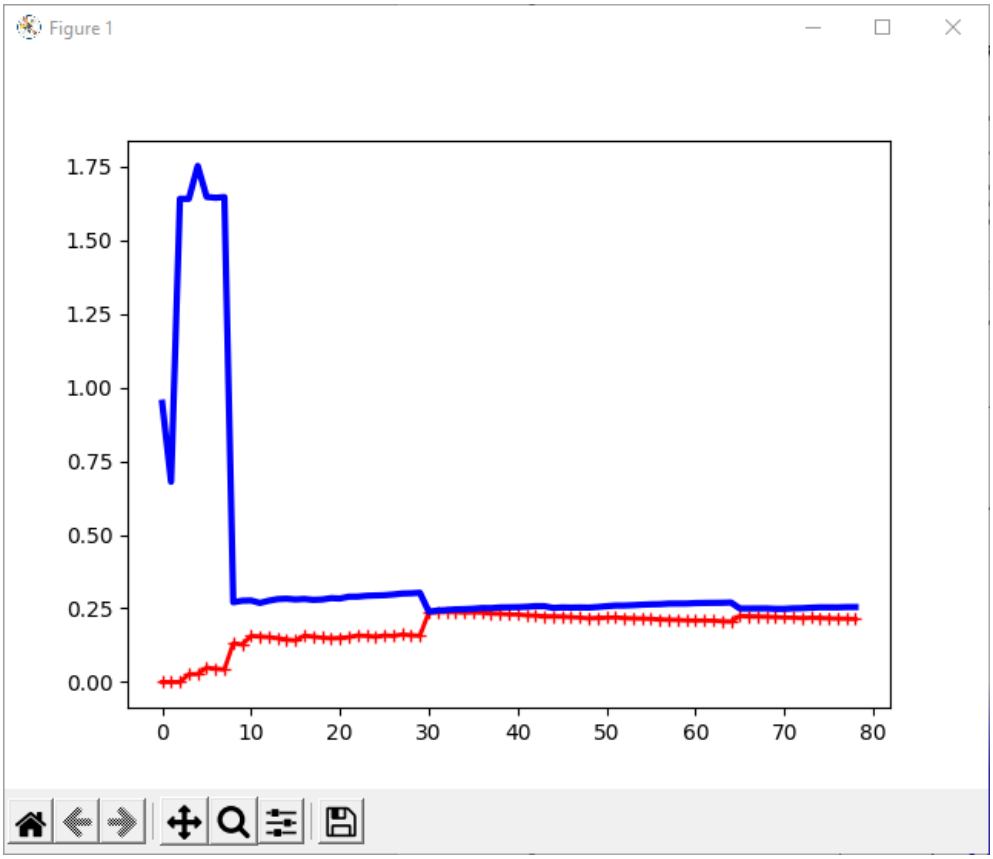

Рис. 19. Криві навчання для поліноміальної моделі 2-го ступеня

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


def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)
    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")


# Вхідні дані
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Створення об'єкта лінійного регресора
linear_regression = LinearRegression()
linear_regression.fit(X, y)
# Побудова графіка лінійної регресії
plot_learning_curves(linear_regression, X, y)
plt.xticks(())
plt.yticks(())
plt.show()

polynomial_regression = Pipeline(
    [("poly_features", PolynomialFeatures(degree=10, include_bias=False)), ("linear_regression", LinearRegression())])
plot_learning_curves(polynomial_regression, X, y)
plt.show()

polynomial_regression = Pipeline(
    [("poly_features", PolynomialFeatures(degree=2, include_bias=False)), ("linear_regression", LinearRegression())])
plot_learning_curves(polynomial_regression, X, y)
plt.show()
```

Рис. 20. Код програми.

Висновок: в ході виконання лабораторної роботи було досліджено методи регресії даних у машинному навчанні за допомогою мови програмування Python.

GitHub