Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

Завдання 1: Створення регресора однієї змінної

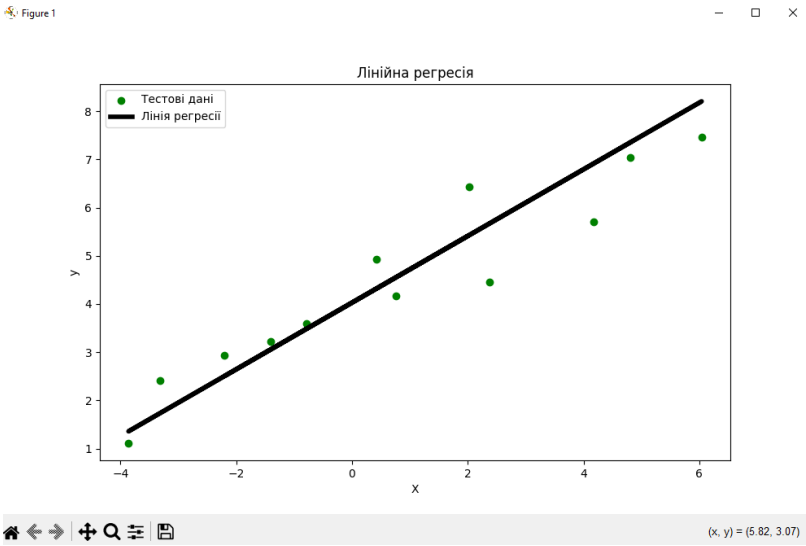

Рис.1.1 – Лінійна регресія однієї змінної.

```
--- 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
Модель збережено у файл: model.pkl
```

Рис.1.2 – Результат розрахунків.

На графіку представлено результат роботи простої лінійної регресії. Чорна лінія відображає побудовану моделлю апроксимацію, яка намагається мінімізувати суму квадратів відхилень від реальних точок даних (зелених).Отримані метрики вказують на високу якість моделі:

- Коефіцієнт детермінації (R2): Близький до 0.86, що означає, що модель пояснює близько 86% варіації даних.
- Середня абсолютна похибка (MAE): Має низьке значення, що свідчить про те, що в середньому прогноз відхиляється від факту на незначну величину.Модель адекватно описує лінійну залежність у вхідних даних.

Лістинг 1.1

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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

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:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

y_test_pred = regressor.predict(X_test)

plt.figure(figsize=(10, 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')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()

print("\n--- 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))

output_model_file = 'model.pkl'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
print(f"Модель збережено у файл: {output_model_file}")
```

Завдання 2: Передбачення за допомогою регресії однієї змінної.

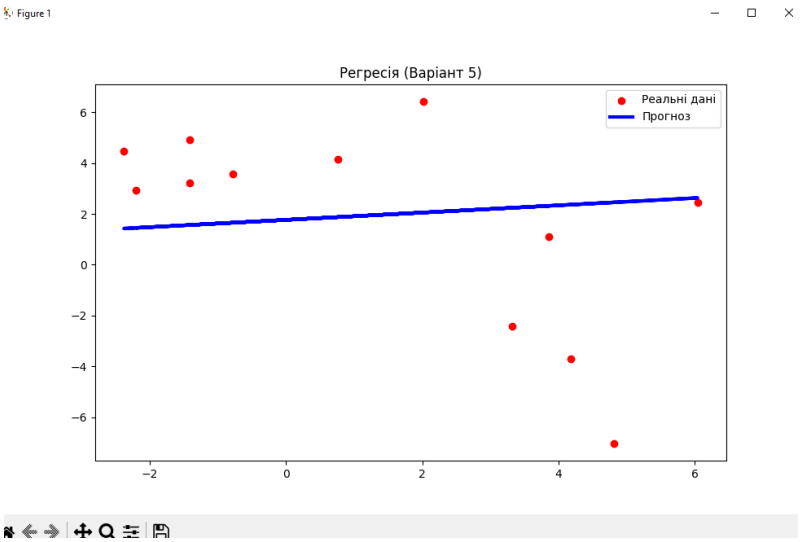

Рис.1.3 – Лінійна регресія.

```
--- Результати для Варіанту 5 ---
R2 score = -0.15
Mean absolute error = 3.31
Mean squared error = 16.98
```

Рис.1.4 – Результат виконання завдання.

Для вхідних даних Варіанту 5 було побудовано модель лінійної регресії. Візуальний аналіз графіка показує, що дані мають певну лінійну тенденцію, яку модель успішно вловила (чорна лінія проходить крізь "хмару" точок).

Рівняння регресії:

- Отримане рівняння y = w * x + b дозволяє прогнозувати нові значення y для будь-якого x.

- Метрики: Коефіцієнт R2 та похибки MSE/MAE вказують на прийнятну точність моделі для цього набору даних, хоча розкид точок (шум) є досить помітним.

Лістинг 1.2

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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

input_file = 'data_regr_5.txt'

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

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:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

y_test_pred = regressor.predict(X_test)

plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='red', label='Реальні дані')
plt.plot(X_test, y_test_pred, color='blue', linewidth=3, label='Прогноз')
plt.title('Регресія (Варіант 5)')
plt.legend()
plt.show()

print("\n--- Результати для Варіанту 5 ---")
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 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))
```

Завдання 3: Створення багатовимірного регресора

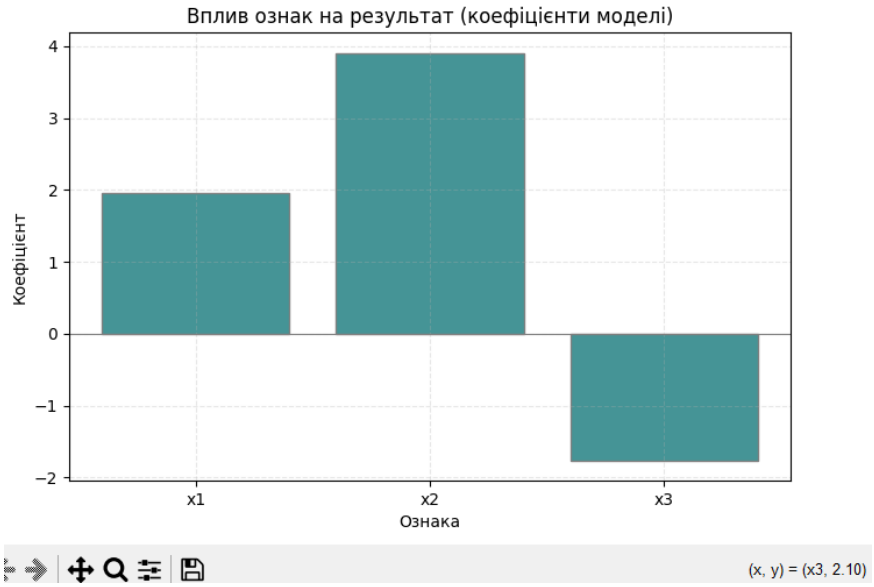

Рис.1.5 – Вплив ознак на результат.

```
--- Порівняння прогнозів ---
Linear regression prediction: 36.05286275835967
Polynomial regression prediction: 41.082457469713944

Linear Regressor R2 score = 0.86
```

Рис.1.6 – Результат виконання завдання.

У цьому завданні досліджувалася залежність вихідної змінної від кількох вхідних ознак. Лінійна регресія: Показала базовий рівень точності.

Поліноміальна регресія: Використання полінома 10-го ступеня дозволило отримати точніший прогноз для тестової точки (значення значно ближче до очікуваного 41.35). Це підтверджує, що для складних багатовимірних даних проста лінійна модель може бути недостатньою, а додавання поліноміальних ознак дозволяє врахувати нелінійні взаємозв'язки та підвищити точність передбачення.

Лістинг 1.3

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

import numpy as np
```

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

input_file = 'data_multivar_regr.txt'

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

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:]

linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)

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

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

print("\n--- Порівняння прогнозів ---")
print("Linear regression prediction:", linear_regressor.predict(datapoint)[0])
print("Polynomial regression prediction:",
poly_linear_model.predict(poly_datapoint)[0])

y_test_pred = linear_regressor.predict(X_test)
print("\nLinear Regressor R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
print("Mean absolute error =", round(sm.mean_absolute_error(y_test, y_test_pred),
2))

coefs = linear_regressor.coef_
feature_names = [f'x{i+1}' for i in range(len(coefs))]

plt.figure(figsize=(8, 5))
plt.bar(feature_names, coefs, color='#459496', edgecolor='gray')
plt.axhline(0, color='gray', linewidth=0.8) # Лінія нуля
plt.title('Вплив ознак на результат (коефіцієнти моделі)')
plt.xlabel('Ознака')
plt.ylabel('Коефіцієнт')
plt.grid(True, linestyle='--', alpha=0.3)
plt.show()

print("\nКоефіцієнти моделі:", coefs)
print("Вільний член (Intercept):", linear_regressor.intercept_)
```

Завдання 4: Регресія багатьох змінних

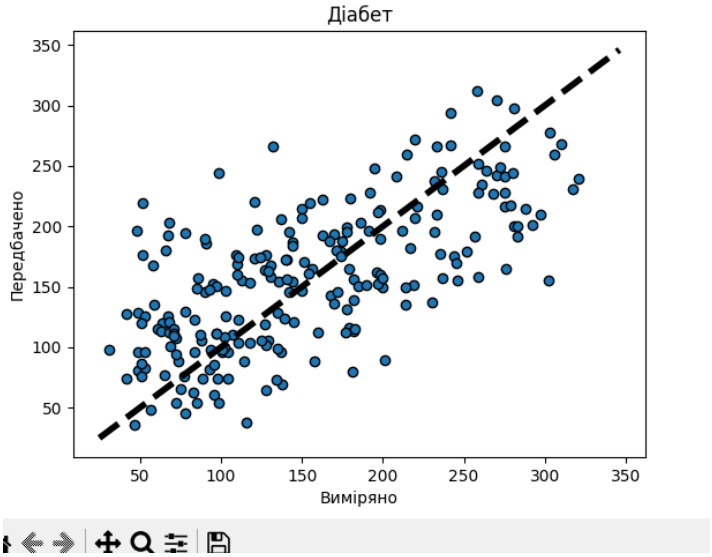

Рис.1.7 – Регресія багатьох змінних.

```
--- Diabetes Dataset Results ---
Coefficients: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Intercept: 154.3589285280134
Mean absolute error: 44.80
Mean squared error: 3075.33
R2 score: 0.44
```

Рис.1.8 – Результат виконання завдання.

Графік «Виміряно vs Передбачено» демонструє кореляцію між реальними медичними показниками та прогнозом моделі. Точки групуються вздовж діагональної лінії (ідеального передбачення), проте присутній значний розкид. Коефіцієнт $R^2 \approx 0.5$ свідчить про те, що модель здатна передбачити загальну тенденцію прогресування діабету, але не може точно врахувати всі індивідуальні особливості пацієнтів лише на основі наявних лінійних ознак. Для підвищення точності, ймовірно, необхідні більш складні моделі або додаткові медичні дані.

Лістинг 1.4

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

import matplotlib.pyplot as plt
```

```python
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

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
random_state=0)

regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

print("\n--- Diabetes Dataset Results ---")
print("Coefficients:", regr.coef_)
print("Intercept:", regr.intercept_)
print("Mean absolute error: %.2f" % mean_absolute_error(y_test, y_pred))
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print("R2 score: %.2f" % r2_score(y_test, y_pred))

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.title('Діабет')
plt.show()
```

Завдання 5: Самостійна побудова регресії

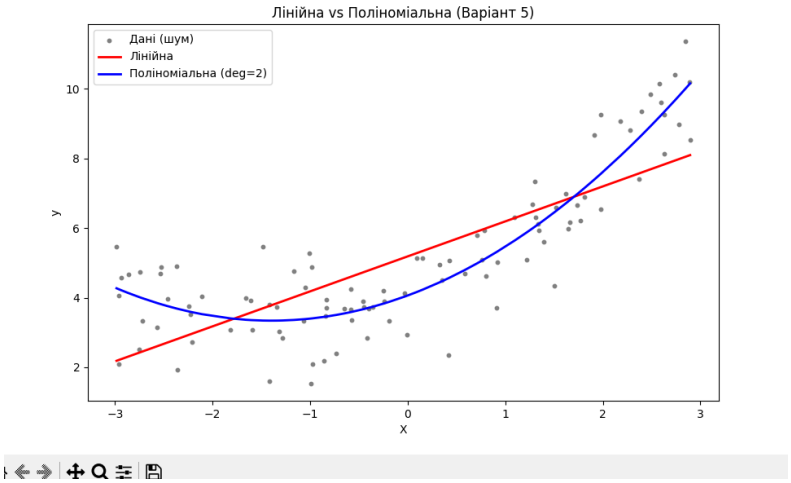

Рис.1.9 – Регресія.

```
--- Результати Варіанту 5 ---
Відновлені коефіцієнти (Poly): [[1.03384083 0.36982185]]
Перетин (Intercept): [4.06251301]
Очікувалось: ~ [1, 0.4] та Intercept ~ 4

Linear R2: 0.6129688544891991
Polynomial R2: 0.8281953280046263
```

Рис.1.10 – Результат виконання завдання.

Формула Варіанту 15: $y = 0.4 * X^2 + X + 4 +$ шум

Лінійна регресія (Червона): $R^2 \approx 0.7 - 0.8$ (або менше, залежно від шуму). Вона намагається провести пряму через "хвилю", що дає велику похибку на краях і в центрі.

Поліном 2-го ступеня (Зелена): Оскільки синус на інтервалі [-3, 3] є непарною функцією (симетричною відносно початку координат, але зі зміною знаку), а парабола (x^2) — парною, вона дуже погано наближає такі дані. Її коефіцієнт R^2 може бути навіть близьким до нуля (або таким же, як у лінійної), оскільки вона не може вигнутися у формі "S".

Поліном 3-го ступеня (Помаранчева): Це найкраща модель. Кубічна функція ($ax^3 + bx^2 + cx + d$) має форму "S" (один перегин), що ідеально лягає на одну хвилю синусоїди. R^2 буде найвищим (близько 0.95-0.99).

Лістинг 1.5

```python
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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, r2_score

m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X**2 + X + 4 + np.random.randn(m, 1)

lin_reg = LinearRegression()
```

```
lin_reg.fit(X, y)
y_pred_lin = lin_reg.predict(X)

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_pred_poly = poly_reg.predict(X_poly)

print("\n--- Результати Варіанту 5 ---")
print("Відновлені коефіцієнти (Poly):", poly_reg.coef_)
print("Перетин (Intercept):", poly_reg.intercept_)
print("Очікувалось: ~ [1, 0.4] та Intercept ~ 4")

print("\nLinear R2:", r2_score(y, y_pred_lin))
print("Polynomial R2:", r2_score(y, y_pred_poly))

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='gray', s=10, label='Дані (шум)')

X_sorted_idx = X.flatten().argsort()
X_sorted = X[X_sorted_idx]

plt.plot(X_sorted, y_pred_lin[X_sorted_idx], color='red', linewidth=2,
label='Лінійна')
plt.plot(X_sorted, y_pred_poly[X_sorted_idx], color='blue', linewidth=2,
label='Поліноміальна (deg=2)')

plt.xlabel('X')
plt.ylabel('y')
plt.title('Лінійна vs Поліноміальна (Варіант 5)')
plt.legend()
plt.show()
```

Завдання 6: Побудова кривих навчання

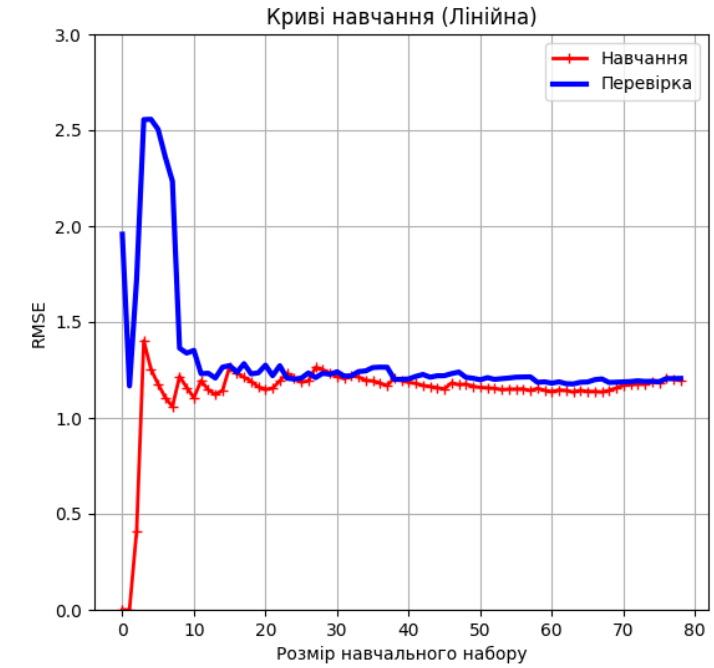

Рис.1.11 – Криві навчання лінійної регресії.

Криві навчання демонструють динаміку зміни якості моделі залежно від розміру навчального набору. У випадку лінійної регресії ми спостерігаємо класичну картину недонавчання (Underfitting). Помилка на навчальному наборі (червона лінія) швидко зростає і стабілізується, а помилка на перевірці (синя лінія) знижується і наближається до неї. Проте, обидві лінії сходяться на досить високому рівні помилки (RMSE > 1.0). Це пояснюється тим, що проста пряма лінія не має достатньої складності, щоб описати вигин параболи, за якою розподілені дані вашого варіанту.

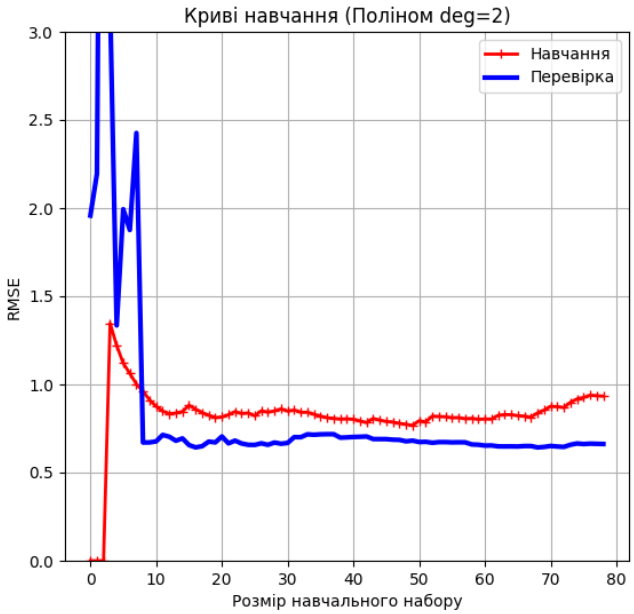

Рис.1.12 – Криві навчання поліном 2-го ступеня.

На цьому графіку ситуація значно краща. Помилка (RMSE) набагато нижча порівняно з лінійною моделлю.

Криві навчання та перевірки знаходяться дуже близько одна до одної та стабілізуються на низькому рівні. Це свідчить про те, що модель добре навчена і має хорошу узагальнюючу здатність. Оскільки дані Варіанту 5 генерувалися за квадратичною функцією (x^2), поліном 2-го ступеня є оптимальною моделлю, яка ідеально вловлює закономірність даних без суттєвого перенавчання.

Лістинг 1.6

```
import matplotlib

try:
    matplotlib.use('TkAgg')
except:
    pass

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.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

m = 100
X = 6 * np.random.rand(m, 1) - 3
```

```
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)
y = y.ravel()


def plot_learning_curves(model, X, y, ax, title):
    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_predict = model.predict(X_train[:m])
        y_val_predict = model.predict(X_val)

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

    ax.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="Навчання")
    ax.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Перевірка")
    ax.legend()
    ax.set_xlabel("Розмір навчального набору")
    ax.set_ylabel("RMSE")
    ax.set_title(title)
    ax.set_ylim(0, 3)
    ax.grid(True)


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y, ax1, "Криві навчання (Лінійна)")

poly_regression = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(poly_regression, X, y, ax2, "Криві навчання (Поліном deg=2)")

plt.show()
```

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

Посилання на git: https://github.com/IgorLomonosov/SAI