ЛАБОРАТОРНА РОБОТА № 3

ПОРІВНЯННЯ МЕТОДІВ РЕГРЕСІЇ

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

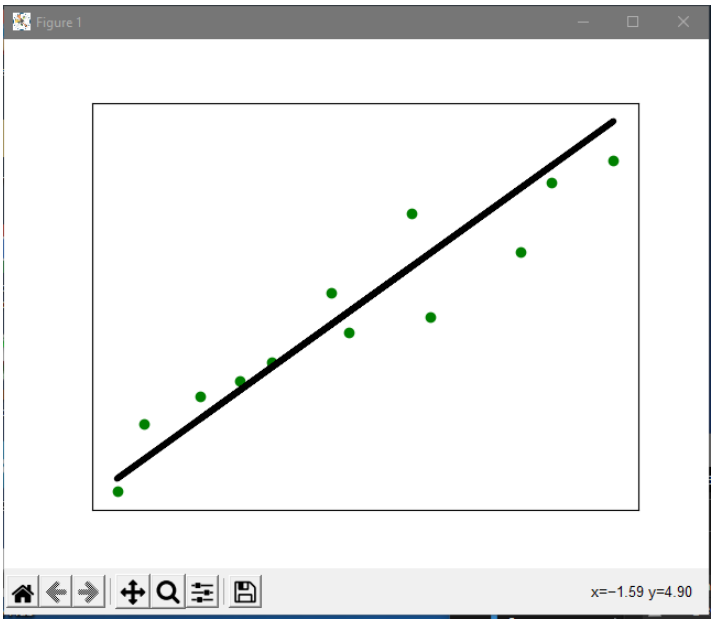
Хід роботи

Завдання 2.1. Створення регресора однієї змінної

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))
```




```
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. 1 Графік функції та оцінки якості

На основі отриманих даних можемо сказати, що модель добре справляється з поставленим завданням.

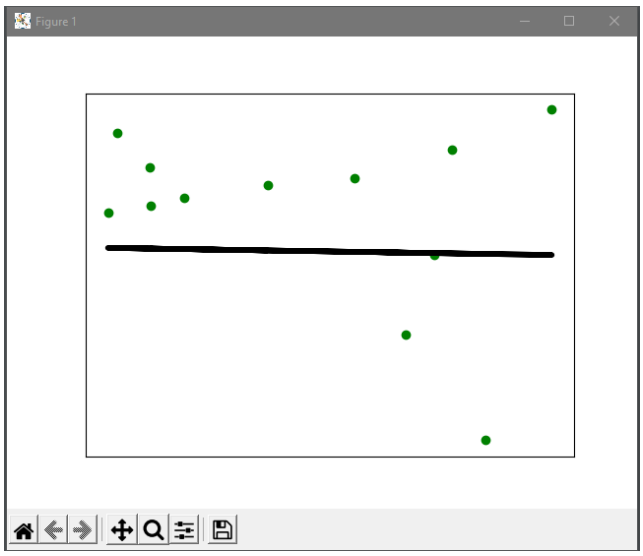
Завдання 2.2. Передбачення за допомогою регресії однієї змінної

№ за списком	3
№ варіанту	3

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_regr_3.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))
```




```
Linear regressor performance:
Mean absolute error = 3.59
Mean squared error = 17.39
Median absolute error = 3.39
Explain variance score = 0.02
R2 score = -0.16

New mean absolute error = 3.59
```

Рис. 2 Графік функції та оцінки якості

Цього разу можемо бачити, що в датасеті є аномальні дані MSE vs MAE, до яких модель не пристосована

Завдання 2.3. Створення багатовимірного регресора

Код програми:

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

input_file = 'data_multivar_regr.txt'
data = np.loadtxt(input_file, delimiter = ',')

X, y = data[:, :-1], data[:, -1]

num_training = int(0.8 * len(X))
num_test = len(X) - num_training
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)

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))

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:\n",
linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n",
poly_linear_model.predict(poly_datapoint))
```

```
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:
 [41.46197042]
```

Рис. 3 Виведення характеристик та порівняння моделей

На основі отриманого результату, можна сказати, що поліноміальний регресор справляється краще за лінійний при регресії з декількома характеристиками.

Завдання 2.4. Регресія багатьох змінних

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

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, ran-
dom_state = 0)
```

```
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
y_pred = regr.predict(X_test)

print("Linear regressor performance:")
print("regr.coef_ =", regr.coef_)
print("regr.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()
```

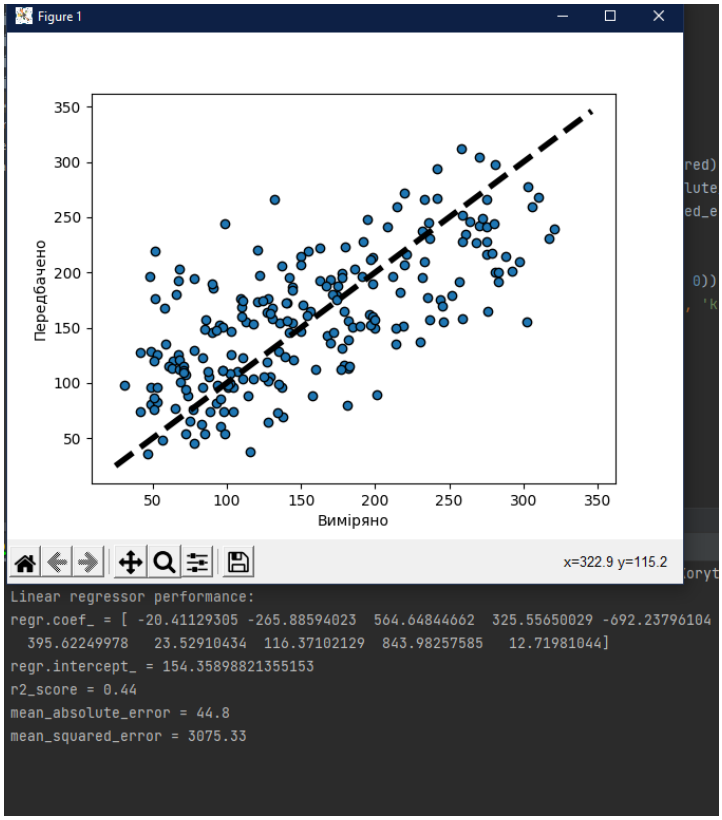


Рис. 4 Результати лінійої регресії

На основі отриманих результатів, можна побачити, що похибка є великою, але обрана регресія працює краще ніж звичайна регресія з використанням середніх значень.

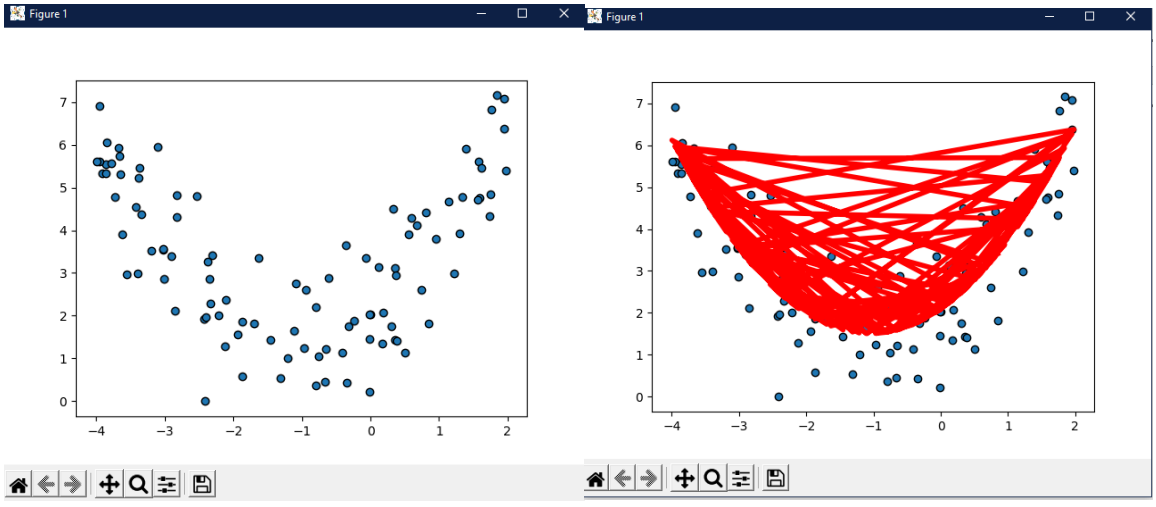
Завдання 2.5. Самостійна побудова регресії

Варіант 3

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
from sklearn.preprocessing import PolynomialFeatures
```

```
m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)

fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors = (0, 0, 0))
plt.show()

print(X[1], y[1])

poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(np.array(X).reshape(-1, 1))

linear_regression = linear_model.LinearRegression()
linear_regression.fit(X_poly, y)
print(linear_regression.intercept_, linear_regression.coef_)
y_pred = linear_regression.predict(X_poly)

fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors = (0, 0, 0))
plt.plot(X, y_pred, color='red', linewidth=4)
plt.show()
```




```
[-1.63315098] [3.35232832]
[2.08716493] [[1.11368304 0.53093351]]
```

Рис. 5 Результати регресії

Модель рівняння: y = 0.5x2 + 1x + 2 + гаусовий шум. Отримана модель регресії з передбаченими коефіцієнтами: y = 0.53x2 + 1.1x + 2

Можна зробити висновок, що поліноміальна регресія надає змогу будувати моделі для нелінійних даних.

Завдання 2.6. Побудова кривих навчання

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
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)

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))
  fig, ax = plt.subplots()
  plt.ylim(0, 2)
  ax.plot(np.sqrt(train_errors), "r-+", linewidth = 2, label = 'train')
  ax.plot(np.sqrt(val_errors), "b-", linewidth = 3, label = 'val')
  plt.show()

linear_regression = linear_model.LinearRegression()
plot_learning_curves(linear_regression, np.array(X).reshape(-1, 1), y)

polynomial_regression = Pipeline([
  ('poly_features', PolynomialFeatures(degree=2, include_bias=False)),
  ('lin_reg', linear_model.LinearRegression()),
])

plot_learning_curves(polynomial_regression, np.array(X).reshape(-1, 1), y)
```

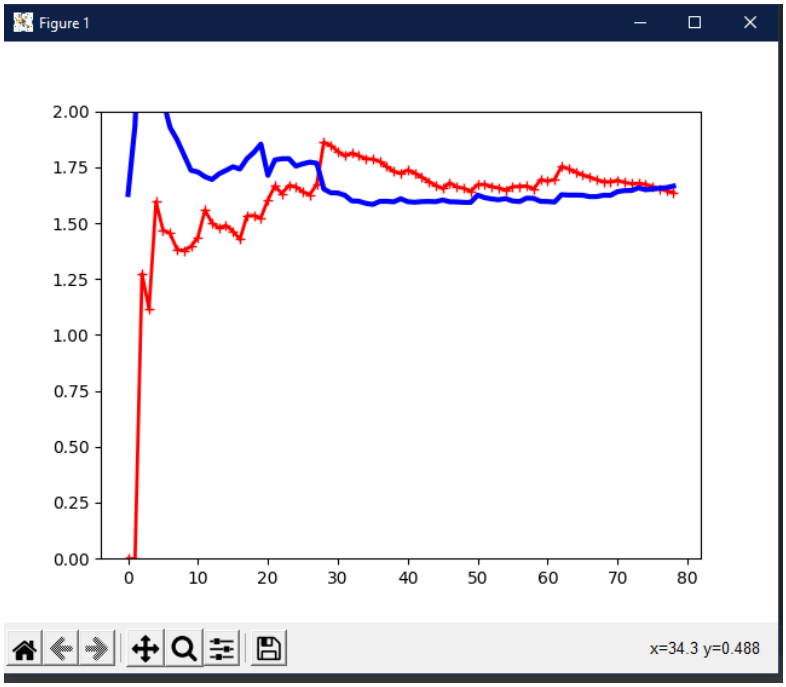


Рис. 6 Криві навчання для лінійної моделі

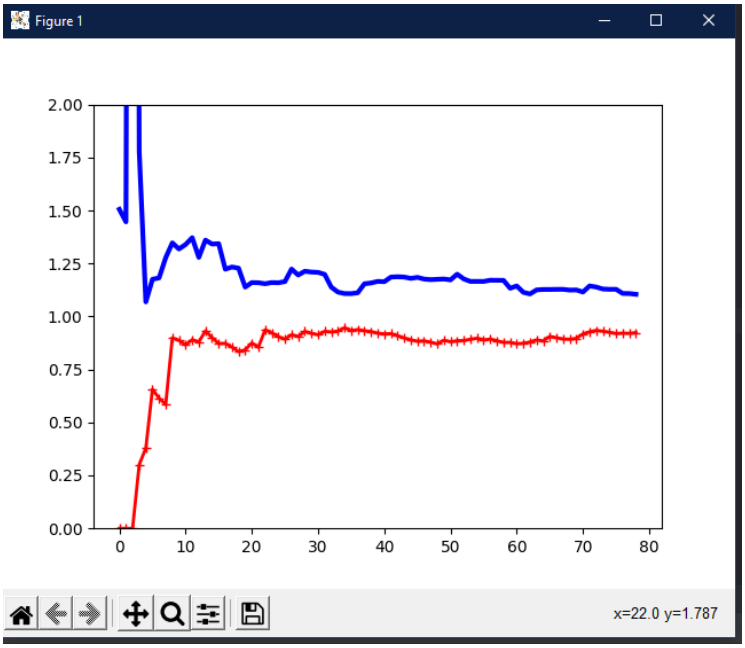


Рис. 7 Криві навчання для поліноміальної моделі

Висновки: на даній лабораторній роботі я дослідив методи регресії даних, використовуючи спеціалізовані бібліотеки та мову програмування Python.