Лабораторна робота №4

Дослідження методів регресії

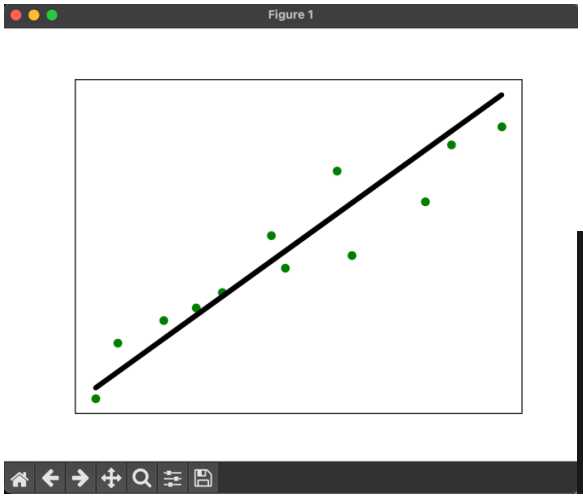
Виконав: ІПЗ-21-3 Осипчук Антон Олексійович

Github: https://github.com/AntonOsypchuk1/ai_lab/tree/main/lab4

Завдання 1. Створення регресора однієї змінної.

```
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
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))
```




```
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
```

Лінійний регресор було успішно створено, збережено та завантажено. Значення показника mean absolute error залишилося незмінним при використанні завантаженої моделі.

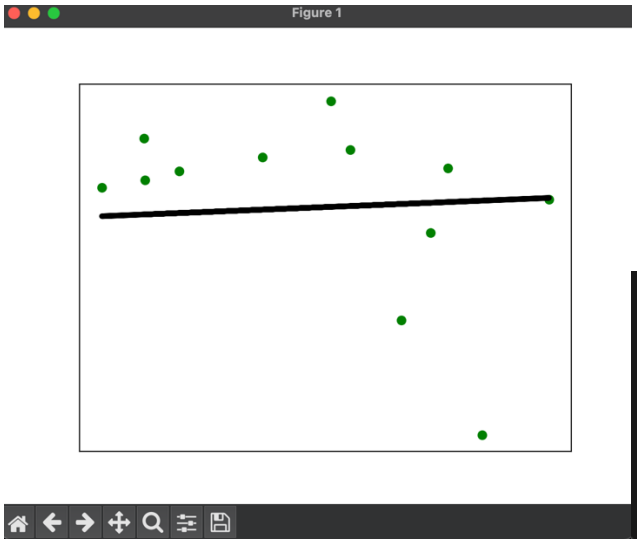
Завдання 2. Передбачення за допомогою регресії однієї змінної.

14 варіант - 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 = 'data/data_regr_4.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
```

```python
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))
```




```
Linear regressor performance:
Mean absolute error = 2.72
Mean squared error = 13.16
Median absolute error = 1.9
Explain variance score = -0.07
R2 score = -0.07

New mean absolute error = 2.72
```

Регресійну модель на основі однієї змінної було побудовано з використанням даних із файлу відповідно до варіанта. Після збереження та завантаження моделі показник mean absolute error залишився незмінним під час тестування.

Завдання 3. Створення багатовимірного регресора.

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

input_file = 'data/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:]

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

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))

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

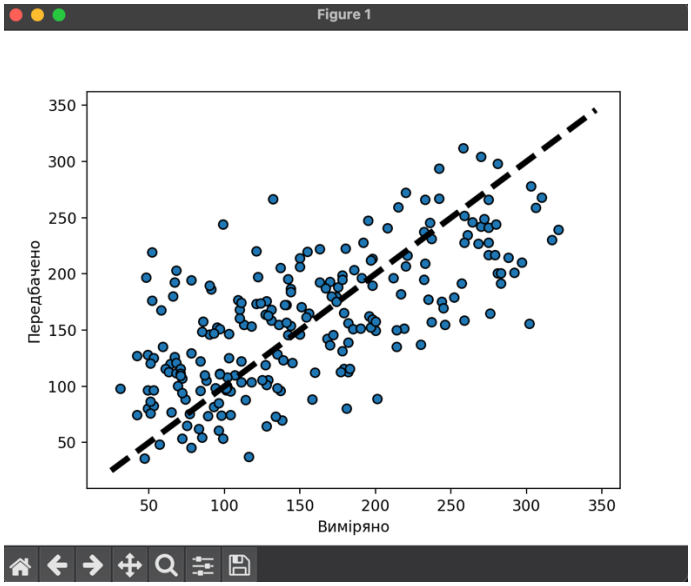
```
(venv) venvantonosypchuk@MacBook-Air-Anton lab4 % python3 task3.py
Linear regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explain variance score = 0.86
R2 score = 0.86

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.08371002]
```

Було побудовано лінійну регресійну модель та поліноміальну модель 10-го ступеня на основі множинних змінних. Для тестування була обрана точка з значенням,

наближеним до того, що міститься в початковому наборі даних. Порівнявши результати передбачень обох моделей з реальним значенням, виявилось, що поліноміальна регресія дала точніший результат, який ближчий до фактичного значення.

Завдання 4. Регресія багатьох змінних.

```
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
import sklearn.metrics as sm

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




```
Regression coefficient = [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Regression interception = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
2024-12-15 13:50:48.121 Python[53105:83571861] +[IMKClient subclass]: chose IMKClient_Modern
2024-12-15 13:50:48.121 Python[53105:83571861] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

Було розроблено лінійний регресор, використовуючи набір даних по діабету з бібліотеки sklearn.datasets. На основі цього було побудовано графік, який відображає залежність між спостережуваними відповідями в наборі даних та відповідями, передбаченими лінійним наближенням (відображеними у вигляді крапок), а також пряму лінію, що відповідає лінійній регресії, яка мінімізує залишкову суму квадратів між початковими даними та передбаченими значеннями.

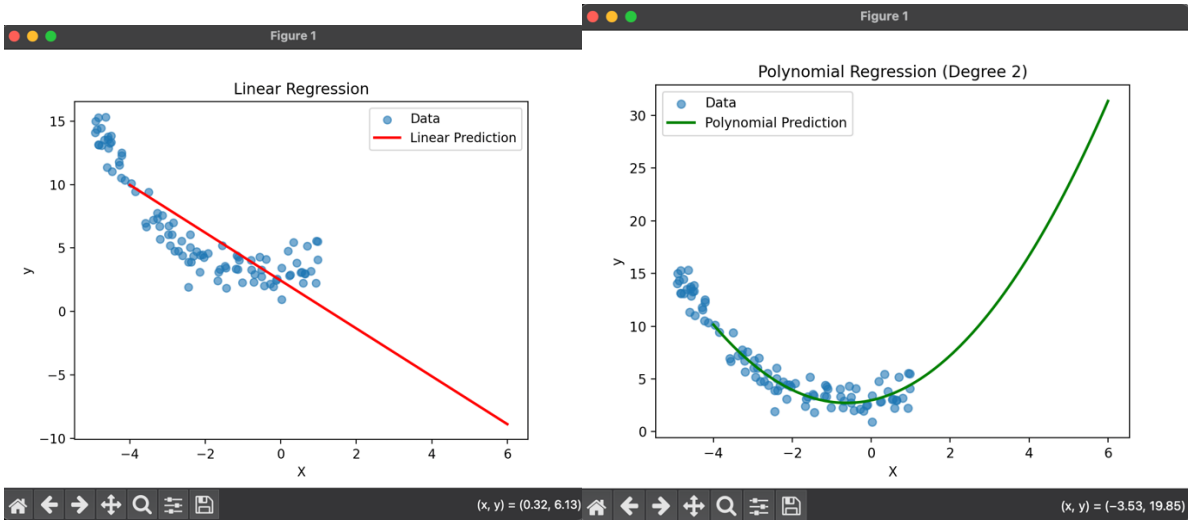
Значення $R^2 = 0.44$ вказує на те, що модель пояснює 44% варіацій залежної змінної. Показник MAE вказує на середнє відхилення прогнозів від реальних значень на 44.8 одиниць. Значення MSE = 3075.33 є досить високим, що свідчить про наявність значних помилок у прогнозах моделі.

Завдання 5. Самостійна побудова регресії.

14 варіант - 4 варіант

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

m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)

linear_reg = LinearRegression()
```

```
linear_reg.fit(X, y)

X_plot = np.linspace(-4, 6, 100).reshape(-1, 1)
y_linear_plot = linear_reg.predict(X_plot)
plt.scatter(X, y, label="Data", alpha=0.6)
plt.plot(X_plot, y_linear_plot, label="Linear Prediction", color="red",
linewidth=2)
plt.title("Linear Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.show()

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

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

print("First data point X[0]:", X[0])
print("Transformed features X_poly[0]:", X_poly[0])
print("Polynomial Regression coefficients:", poly_reg.coef_)
print("Polynomial Regression intercept:", poly_reg.intercept_)

y_poly_plot = poly_reg.predict(poly_features.transform(X_plot))
plt.scatter(X, y, label="Data", alpha=0.6)
plt.plot(X_plot, y_poly_plot, label="Polynomial Prediction",
color="green", linewidth=2)
plt.title("Polynomial Regression (Degree 2)")
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.show()
```




```
venvantonosypchuk@MacBook-Air-Anton lab4 % python3 task5.py
2024-12-15 14:29:20.556 Python[61315:83630185] +[IMKClient subclass]: chose IMKClient_Modern
2024-12-15 14:29:20.556 Python[61315:83630185] +[IMKInputSession subclass]: chose IMKInputSession_Modern
First data point X[0]: [0.22268554]
Transformed features X_poly[0]: [0.22268554 0.04958885]
Polynomial Regression coefficients: [[0.81688192 0.65200066]]
Polynomial Regression intercept: [2.9834279]
```

Було побудовано лінійну та поліноміальну регресійні моделі на основі однакового набору даних, де цільові значення (залежні змінні) визначались за формулою:

y = 0.7 * x^2 + x + 3 + C

Після того, як було здійснено прогноз кожної з регресій на рандомному наборі даних в межах (0; 6), стало очевидно, що поліноміальна модель дає прогноз, який краще описує початковий набір даних порівняно з лінійною.

З отриманих значень коефіцієнтів регресії та перехоплення було складено рівняння, яке є наближеним до початкового математичного виразу.

Завдання 6. Побудова кривих навчання.

14 варіант - 4 варіант

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

m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X**2 + X + 3 + np.random.randn(m, 1)

def plot_learning_curves(model, X, y):
    """
```

```
    Plots learning curves for a given model, showing training and
validation errors
    as the training set size increases.

    Parameters:
        model: The machine learning model to evaluate.
        X: Features.
        y: Target values.
    """
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

    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))

    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="Training
error")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Validation
error")
    plt.title("Learning Curves")
    plt.xlabel("Training Set Size")
    plt.ylabel("RMSE")
    plt.legend()
    plt.show()

linear_reg = LinearRegression()
print("Learning curves for Linear Regression:")
plot_learning_curves(linear_reg, X, y)

polynomial_reg = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
print("Learning curves for Polynomial Regression (Degree 10):")
plot_learning_curves(polynomial_reg, X, y)
```

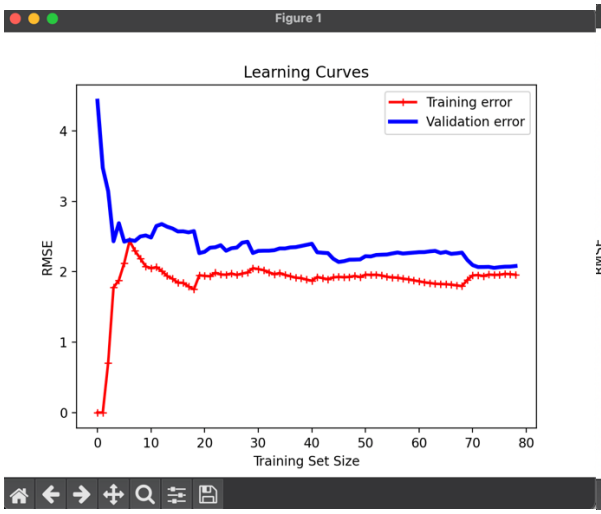


Figure 1
Learning Curves
Training error
Validation error
RMSE
Training Set Size
0
1
2
3
4
0
10
20
30
40
50
60
70
80

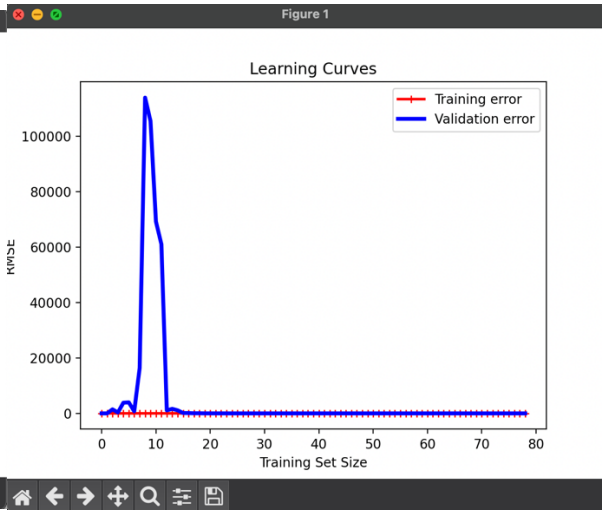


Figure 1
Learning Curves
Training error
Validation error
RMSE
Training Set Size
0
20000
40000
60000
80000
100000
0
10
20
30
40
50
60
70
80