ЛАБОРАТОРНА РОБОТА № 4

Дослідження методів регресії

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

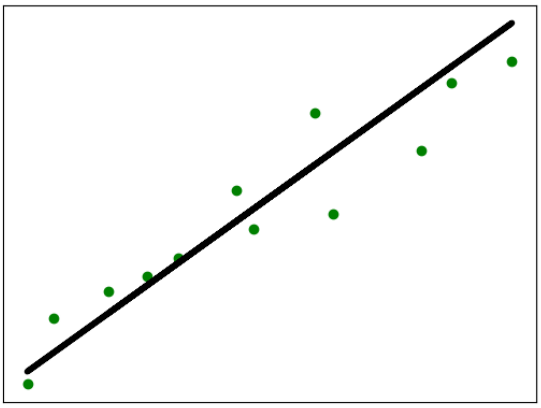
GitHub = https://github.com/Ozzornin/AIS/tree/master

Завдання 1. Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

```
import pickle

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

# 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.savefig("LR_4_task_1.png")
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
```



Побудова регресійної моделі на основі однієї змінної. Модель показала високі результати з $R^2 = 0.86$, що вказує на хороше прогнозування залежності.

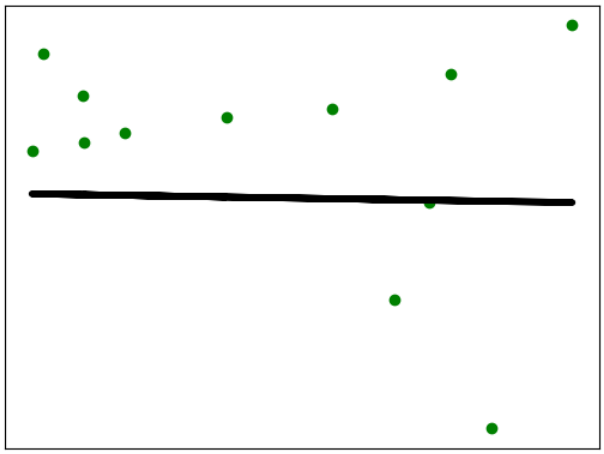
Завдання 2. Передбачення за допомогою регресії однієї змінної

```
import pickle

import matplotlib.pyplot as plt
import numpy as np
import sklearn.metrics as sm
```

```python
from sklearn import linear_model

# 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.savefig("LR_4_task_2.png")
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 = 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
```



Регресія на одній змінній показала нижчу точність ($R^2 = -0.16$). Модель не змогла ефективно передбачити дані.

Завдання 3. Створення багатовимірного регресора

```python
import numpy as np
import sklearn.metrics as sm
from sklearn import linear_model
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:]
# 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))
# 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),
)
# 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:\n", 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

New mean absolute error = 3.58

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.45675521]
```

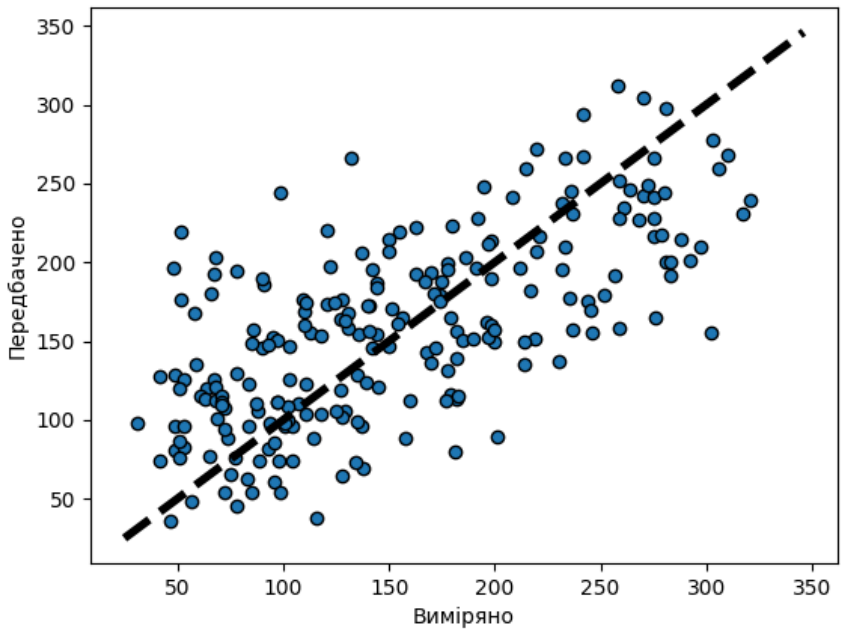
Багатовимірний регресор. Лінійна регресія дала хороші результати ($R^2 = 0.86$), а поліноміальна регресія ще краще підійшла до даних.

Завдання 4. Регресія багатьох змінних

```
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

```
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.5,
random_state=0)
regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)
ypred = regr.predict(Xtest)
print("Estimated coefficients for the linear regression ", regr.coef_)
print(str(regr.intercept_))
print("R2: ", r2_score(ytrain, ypred))
print("Mean absolute error regression loss: ", mean_absolute_error(ytrain,
ypred))
print("Mean squared error regression loss: ", mean_squared_error(ytrain, ypred))
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.savefig("LR_4_task_4.png")
```

```
Estimated coefficients for the linear regression  [ -20.4047621  -265.88518066
564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
154.3589285280134
R2:  -0.49114093907045775
Mean absolute error regression loss:  81.37295843615576
Mean squared error regression loss:  9521.146930399502
```



Багатозмінна регресія на основі набору даних про діабет. Модель мала низьку точність (R^2 = -0.49), що свідчить про проблеми з передбаченням.

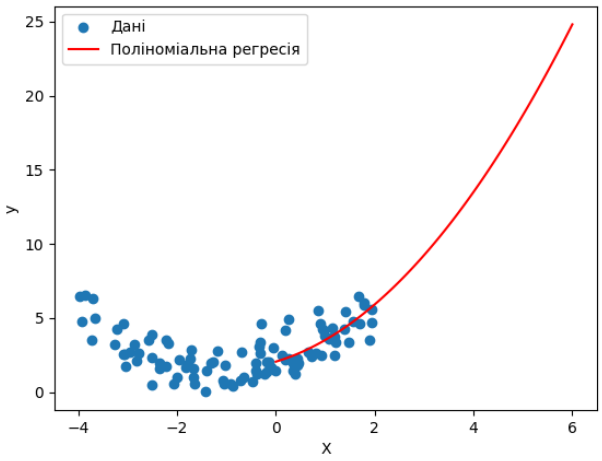
Завдання 5. Самостійна побудова регресії

Мій варіант 13

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	6	7	8	9	10

```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) - 4
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
# Побудова поліноміальних ознак (квадратичних) для X
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X.reshape(-1, 1))
# Виведення значень ознак X[0] та X_poly на екран
print("Значення ознак X[0]:", X[0])
print("Значення ознак після перетворення:", X_poly[0])
# Підгонка лінійної моделі до розширених даних
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)
# Виведення значень коефіцієнтів полінома
intercept = lin_reg.intercept_
coefficients = lin_reg.coef_
print("Значення intercept:", intercept)
print("Значення коефіцієнтів:", coefficients)
# Генерація значень для побудови графіку
X_plot = np.linspace(0, 6, 100)
y_plot = lin_reg.predict(poly_features.transform(X_plot.reshape(-1, 1)))
# Побудова графіку
plt.scatter(X, y, label="Дані")
plt.plot(X_plot, y_plot, label="Поліноміальна регресія", color="red")
plt.xlabel("X")
plt.ylabel("y")
plt.legend()
plt.savefig("LR_4_task_5.png")
```

```
Значення ознак X[0]: [-1.32432308]
Значення ознак після перетворення: [-1.32432308  1.75383163]
Значення intercept: [2.0579947]
Значення коефіцієнтів: [[0.99901533 0.4657998 ]]
```



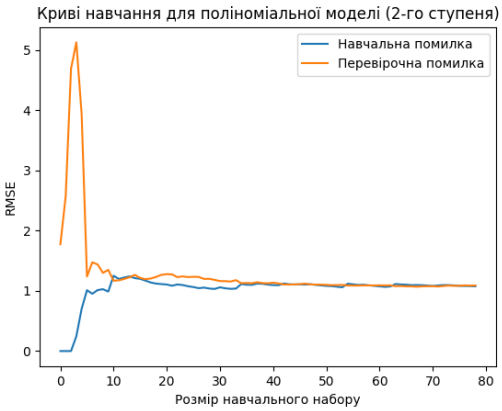
Рівняння отриманої поліноміальної моделі регресії має вигляд:

$y=0.99901533 \cdot X2 +0.4657998 \cdot X+2.0579947$

Завдання 6. Побудова кривих навчання

```
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.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)
# Розділимо дані на навчальний та перевірочний набори
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)


# Функція для побудови кривих навчання
def plot_learning_curves(model, X, y, X_val, y_val):
    train_errors, val_errors = [], []
    for m in range(1, len(X)):
        model.fit(X[:m], y[:m])
        y_train_predict = model.predict(X[:m])
        y_val_predict = model.predict(X_val)
        train_errors.append(mean_squared_error(y[:m], y_train_predict))
        val_errors.append(mean_squared_error(y_val, y_val_predict))
    plt.plot(np.sqrt(train_errors), label="Навчальна помилка")
    plt.plot(np.sqrt(val_errors), label="Перевірочна помилка")
    plt.legend()


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

```
X_poly_val = poly_features.transform(X_val)
lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X_poly, y_train, X_poly_val, y_val)
plt.xlabel("Розмір навчального набору")
plt.ylabel("RMSE")
plt.title("Криві навчання для поліноміальної моделі (2-го ступеня)")
plt.savefig("LR_4_tasK_6.png")
```



Побудова кривих навчання для поліноміальної регресії. Криві показали, як змінюються навчальні і перевірочні помилки залежно від кількості даних, і допомогли проаналізувати якість моделі.

Висновок:

У цій лабораторній роботі було досліджено різні методи регресії для аналізу даних. Використання лінійної регресії на простих наборах даних показало високу точність моделі, однак із збільшенням кількості змінних точність прогнозування значно знижувалася. Поліноміальна регресія виявилася ефективнішою у випадках з нелінійними даними. Завдяки побудові кривих навчання було проаналізовано поведінку моделей залежно від кількості даних, що дало змогу оцінити ефективність навчання.