**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:

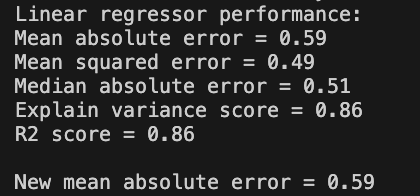
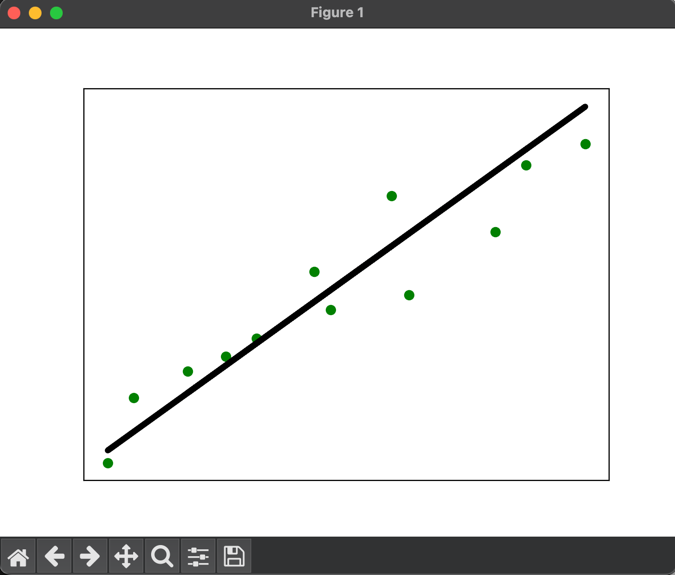
pickle.dump(regressor, f)

# 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))



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

output\_model\_file = 'model.pkl'

# Збереження моделі

with open(output\_model\_file, 'wb') as f:

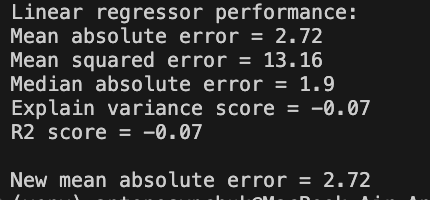
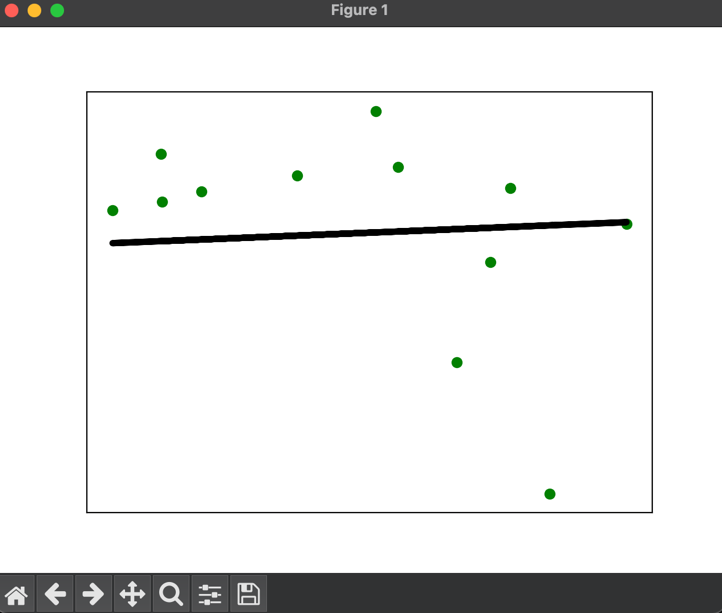
pickle.dump(regressor, f)

# 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))



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.

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]]

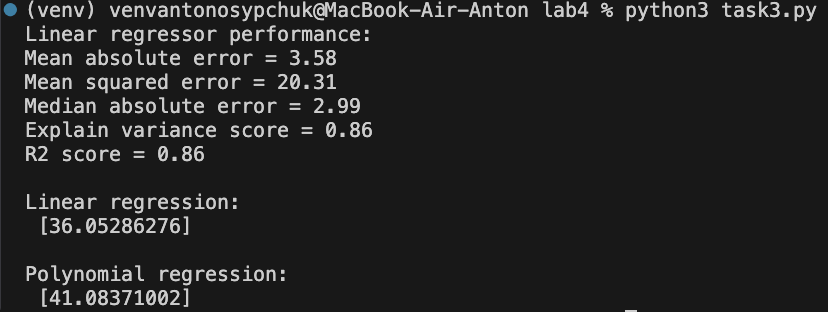
poly\_datapoint = polynomial.fit\_transform(datapoint)

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))



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()

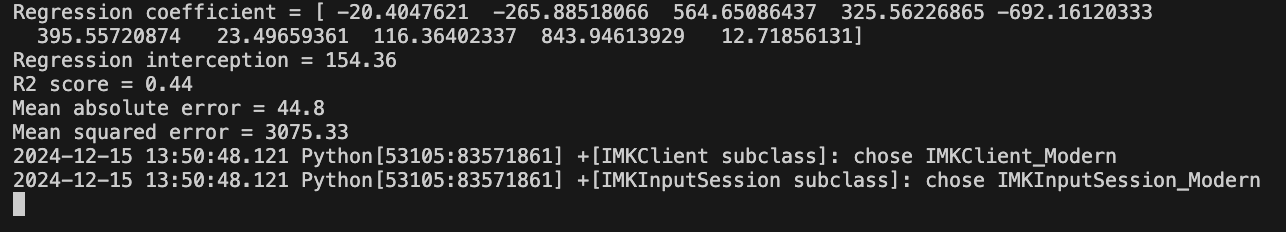
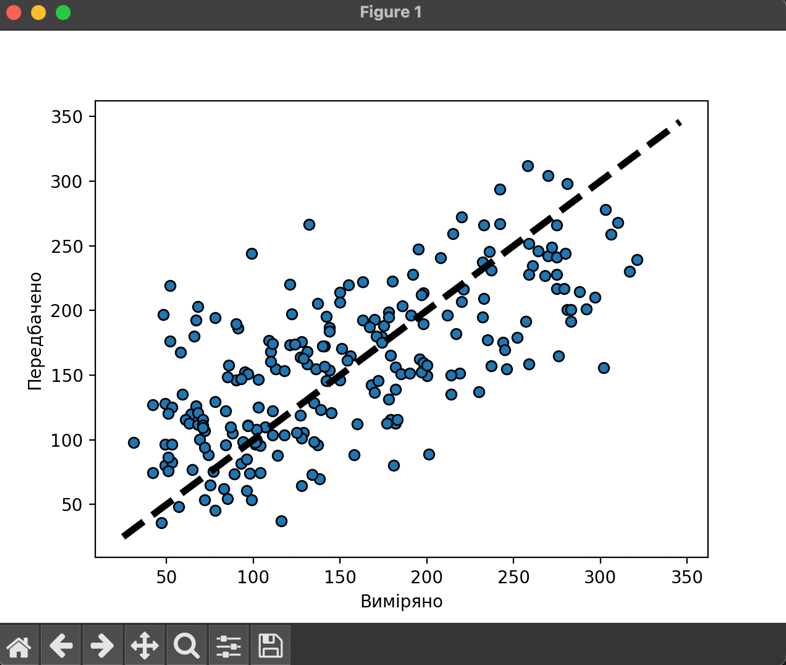
ax.scatter(y\_test, y\_pred, edgecolors = (0, 0, 0))

ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw = 4)

ax.set\_xlabel('Виміряно')

ax.set\_ylabel('Передбачено')

plt.show()



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 R² = 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

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)

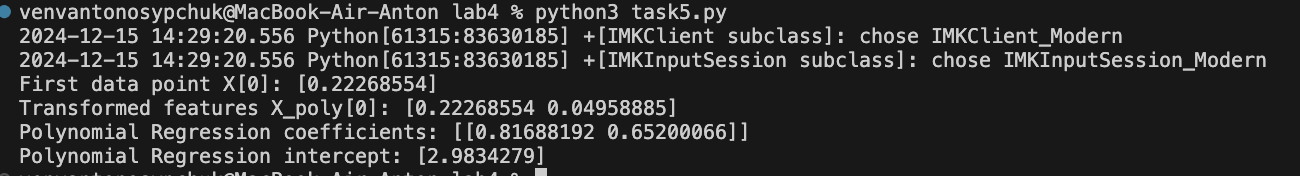
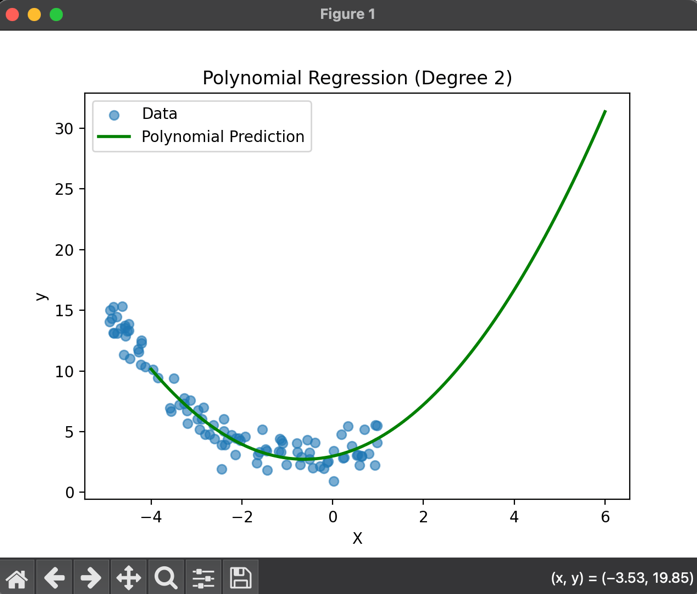
plt.title("Polynomial Regression (Degree 2)")

plt.xlabel("X")

plt.ylabel("y")

plt.legend()

plt.show()



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

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)

