

# ЗВІТ

## З лабораторної роботи №3

### ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ

Студента КН-20-1 навчальної групи

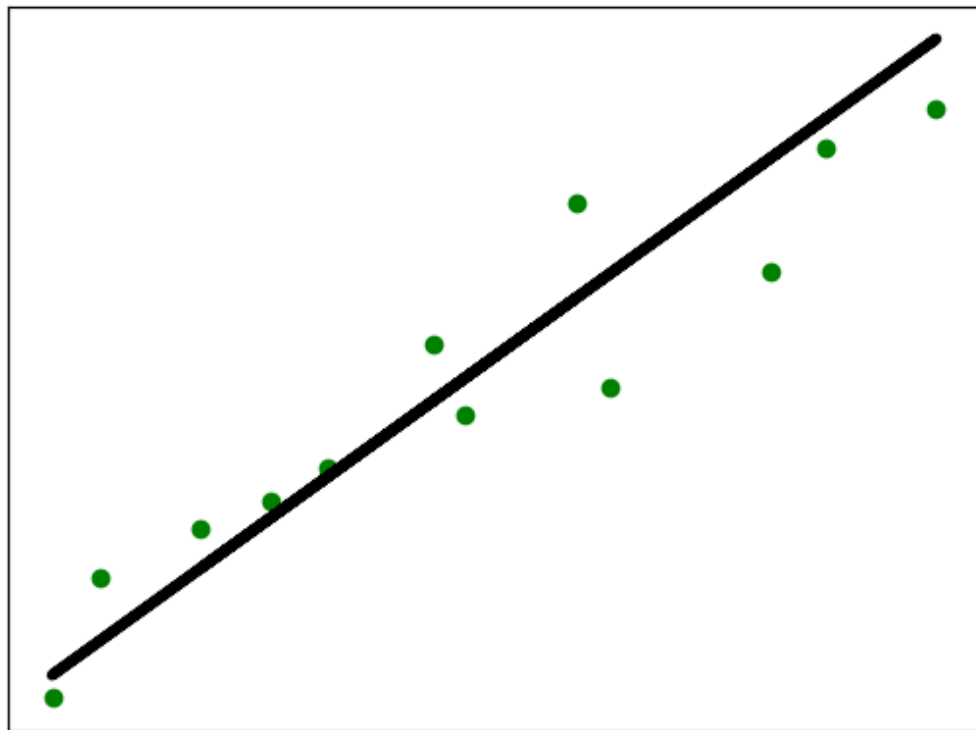
Кірія Даніли Олеговича варіант №6

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

#### Завдання 1. Створення регресора однієї змінної

Figure 1

— □ ×



```
Run: L_3_task_1 ×
"D:\LABS\PRESENT\AI (Python)\LAB3\venv\Scripts\python.exe"
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

Process finished with exit code 0
```

Код програми:

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

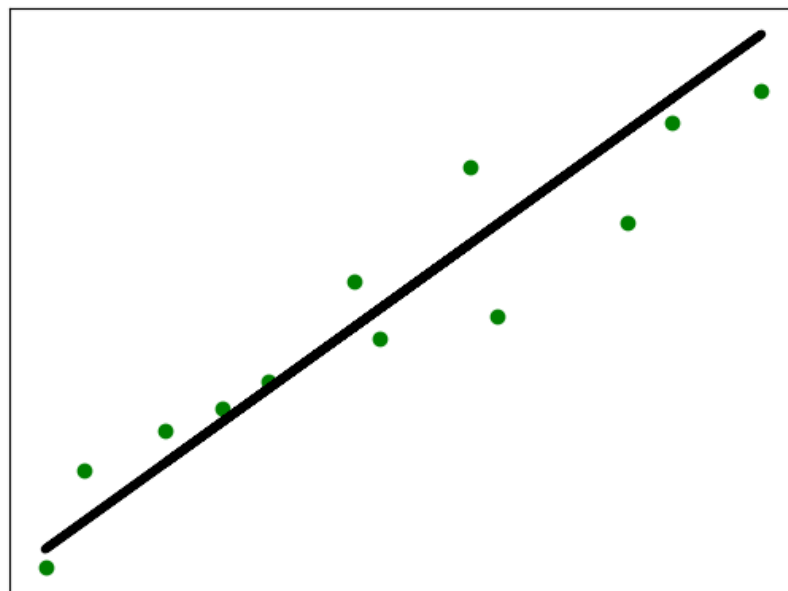
matplotlib.use('TkAgg')
# Вхідний файл, який містить дані
input_file = 'data_singlevar_regr.txt'
# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# 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))
```

## Завдання 2.2. Передбачення за допомогою регресії однієї змінної

### Варіант 1

Figure 1



```
Run: L_3_task_2 x
"D:\LABS\PRESENT\AI (Python)\LAB3\venv\Scripts\python.exe"
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

Process finished with exit code 0
```

Код програми:

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

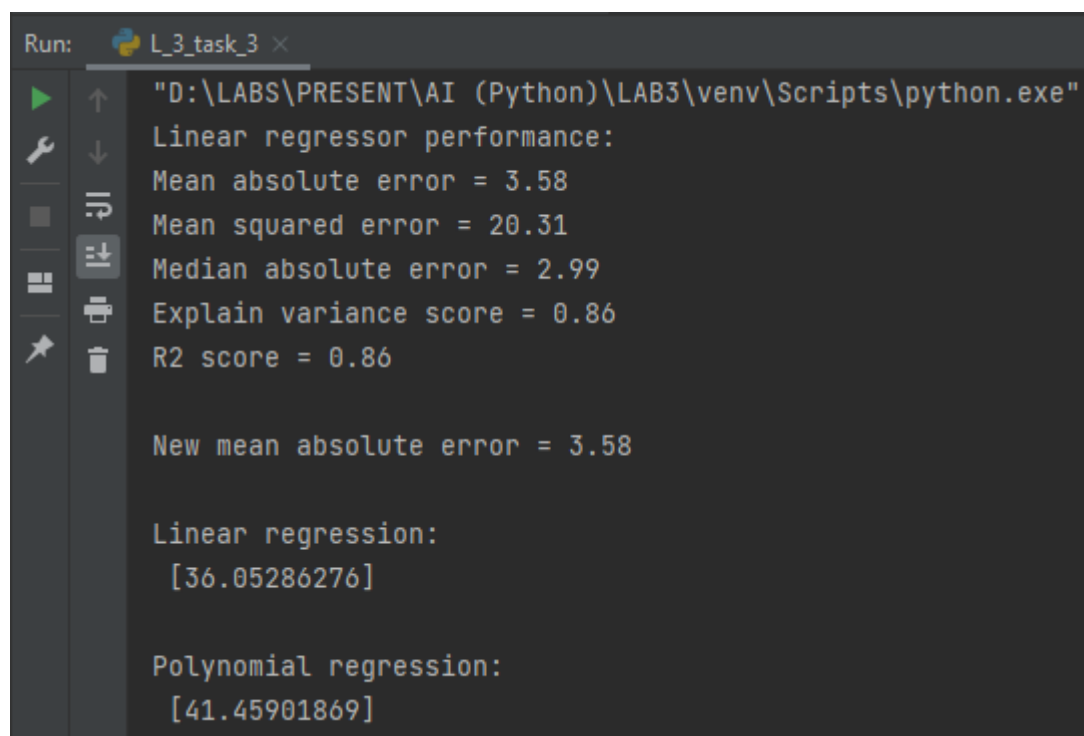
matplotlib.use('TkAgg')
# Вхідний файл, який містить дані
input_file = 'data_regr_1.txt'
# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# 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))

```

### Завдання 2.3. Створення багатовимірного регресора



```

Run: L_3_task_3 x
"D:\LABS\PRESENT\AI (Python)\LAB3\venv\Scripts\python.exe"
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:
[41.45901869]

```

Код програми:

```

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

```

```

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()

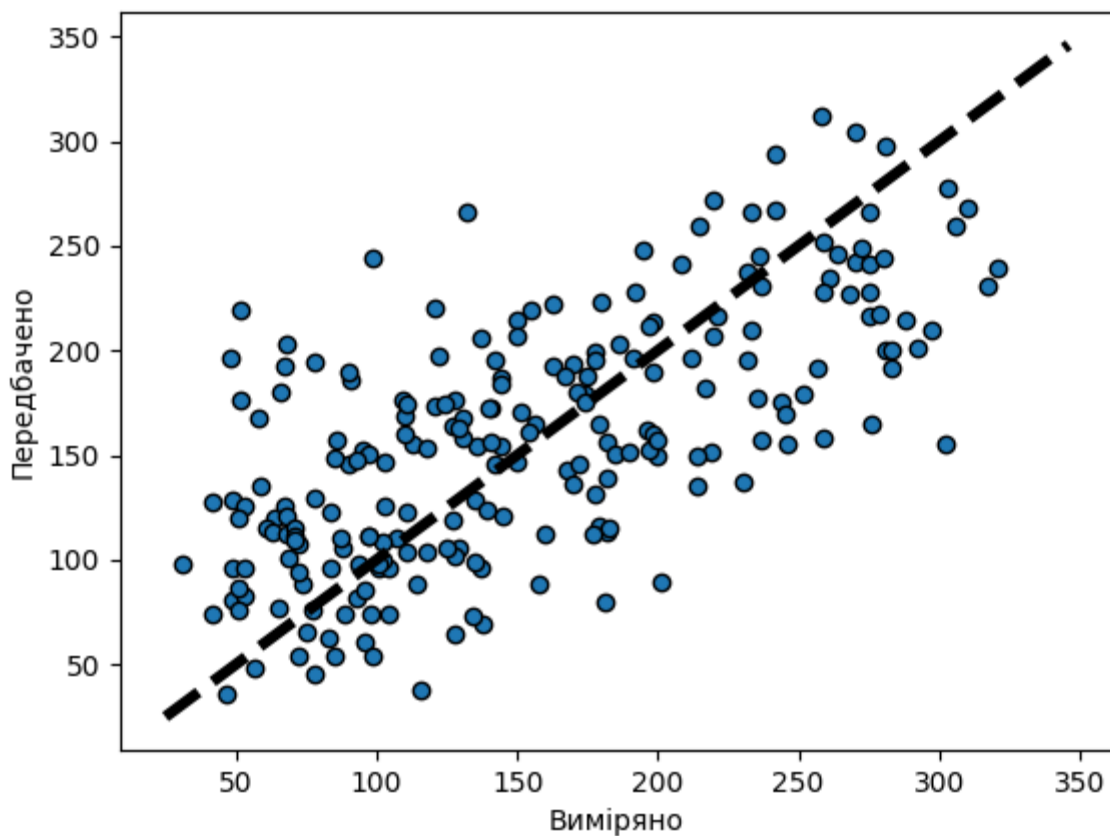
```

```
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))
```

## Завдання 2.4. Регресія багатьох змінних

Figure 1



x=26.0 y=96.1

```
Run: L_3_task_4 x
"D:\LABS\PRESENT\AI (Python)\LAB3\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI (Python)/LAB3/L_3_task_4.py"
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
```

Код програми:

```

import matplotlib
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.model_selection import train_test_split

matplotlib.use('TkAgg')
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))

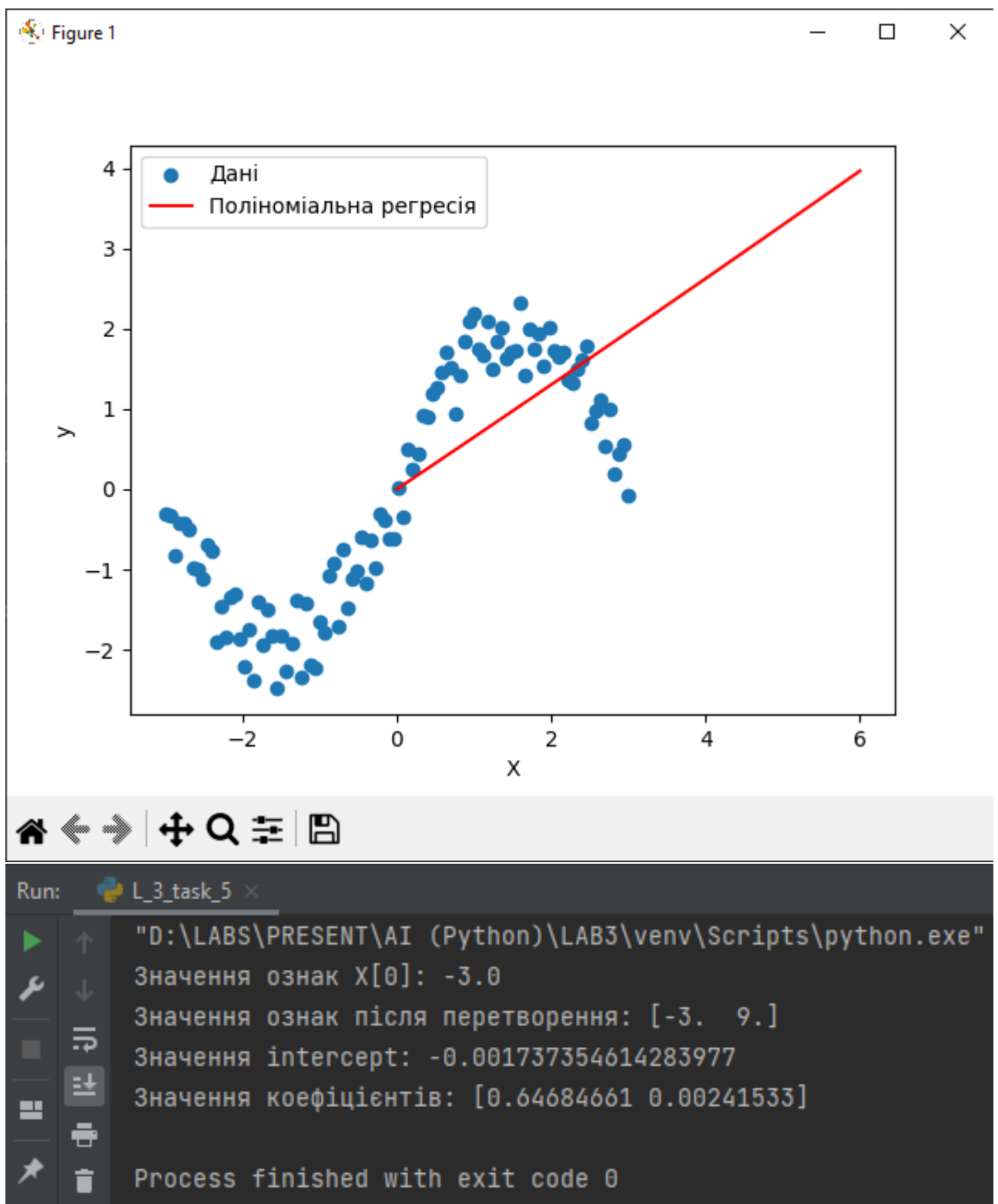
fig, ax = plt.subplots()
ax.scatter(ytest, ypred, edgecolors = (0, 0, 0))
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw = 4)
ax.set_xlabel('Виміряно')
ax.set_ylabel('Передбачено')
plt.show()

```

## Завдання 2.5. Самостійна побудова регресії

Варіант 6





Рівняння отриманої поліноміальної моделі регресії має вигляд:

$$y = 0.64684661 \cdot X^2 + 0.00241533 \cdot X + (-0.001737354614283977)$$

Код програми:

```

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

matplotlib.use('TkAgg')

m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, m)

# 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)

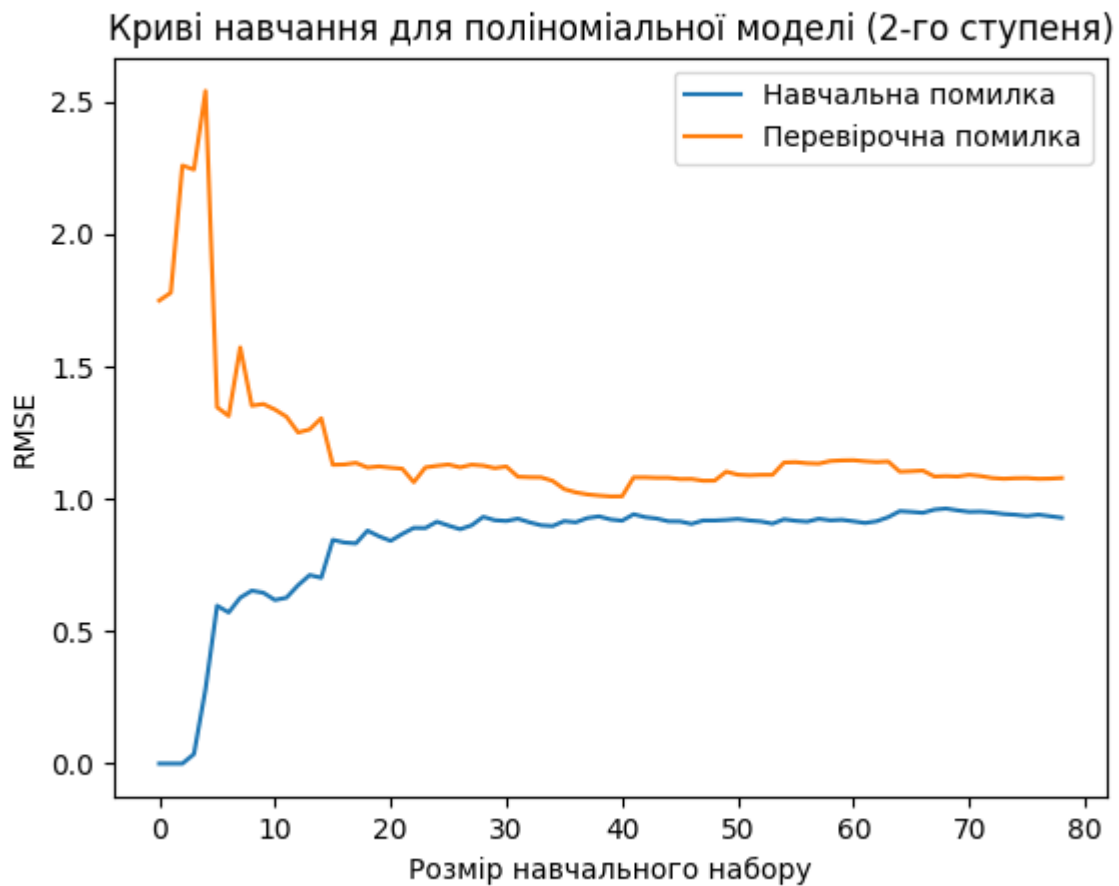
# Генерація значень для побудови графіку
X_plot = np.linspace(0, 6, 100)
y_plot = lin_reg.predict(poly_features.transform(X_plot.reshape(-1, 1)))

# 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.show()

```

## Завдання 2.6. Побудова кривих навчання

Figure 1



x=10.1 y=2.234

Код програми:

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

matplotlib.use('TkAgg')

m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, m)

# 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)
```

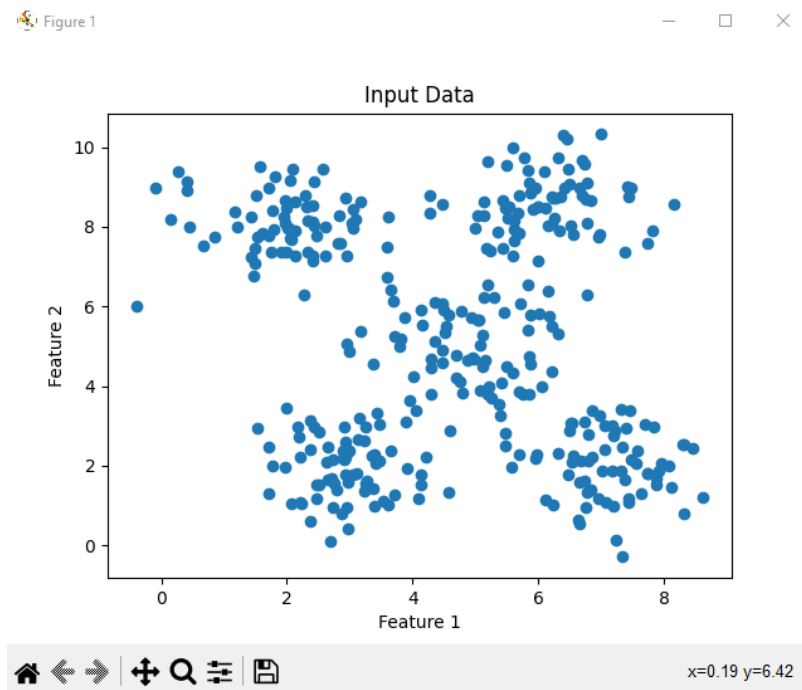
```
# Функція для побудови кривих навчання
def plot_learning_curves(model, X, y, X_val, y_val):
    train_errors, val_errors = [], []
    for m in range(1, len(X)):
        model.fit(X[:m], y[:m])
        y_train_predict = model.predict(X[:m])
        y_val_predict = model.predict(X_val)
        train_errors.append(mean_squared_error(y[:m],
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()

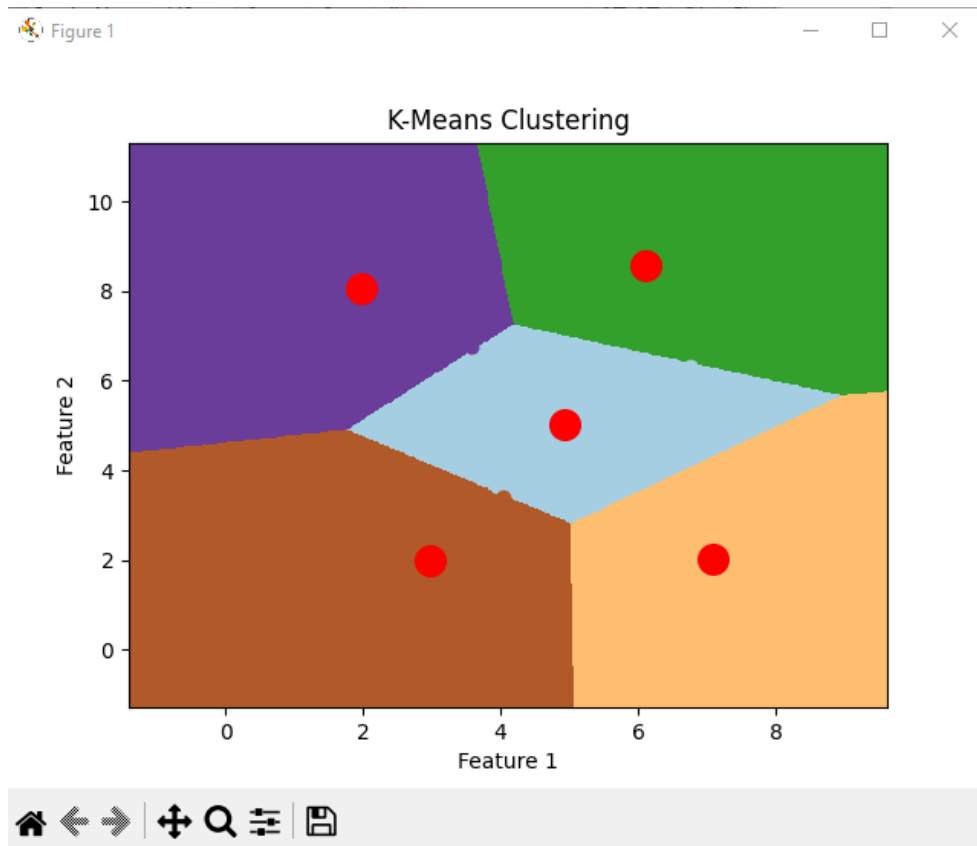
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X_train.reshape(-1, 1))
X_poly_val = poly_features.transform(X_val.reshape(-1, 1))

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.show()
```

## Завдання 2.7. Кластеризація даних за допомогою методу k-середніх





Код програми:

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

matplotlib.use('TkAgg')

# Завантаження вхідних даних
data = np.loadtxt('data_clustering.txt', delimiter=',')

# Включення вхідних даних до графіка
plt.scatter(data[:, 0], data[:, 1])
plt.title('Input Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()

# Створення об'єкту KMeans
kmeans = KMeans(n_clusters=5, init='k-means++', n_init=10,
random_state=0)

# Навчання моделі кластеризації KMeans
kmeans.fit(data)
```

```

# Визначення кроку сітки
h = 0.02

# Визначення меж для сітки
x_min, x_max = data[:, 0].min() - 1, data[:, 0].max() + 1
y_min, y_max = data[:, 1].min() - 1, data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))

# Передбачення вихідних міток для всіх точок сітки
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Графічне відображення областей та виділення їх кольором
plt.figure(1)
plt.imshow(Z, interpolation='nearest', extent=(xx.min(), xx.max(),
yy.min(), yy.max()), cmap=plt.cm.Paired, aspect='auto',
origin='lower')

# Відображення вхідних точок
plt.scatter(data[:, 0], data[:, 1], c=kmeans.labels_,
cmap=plt.cm.Paired)
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

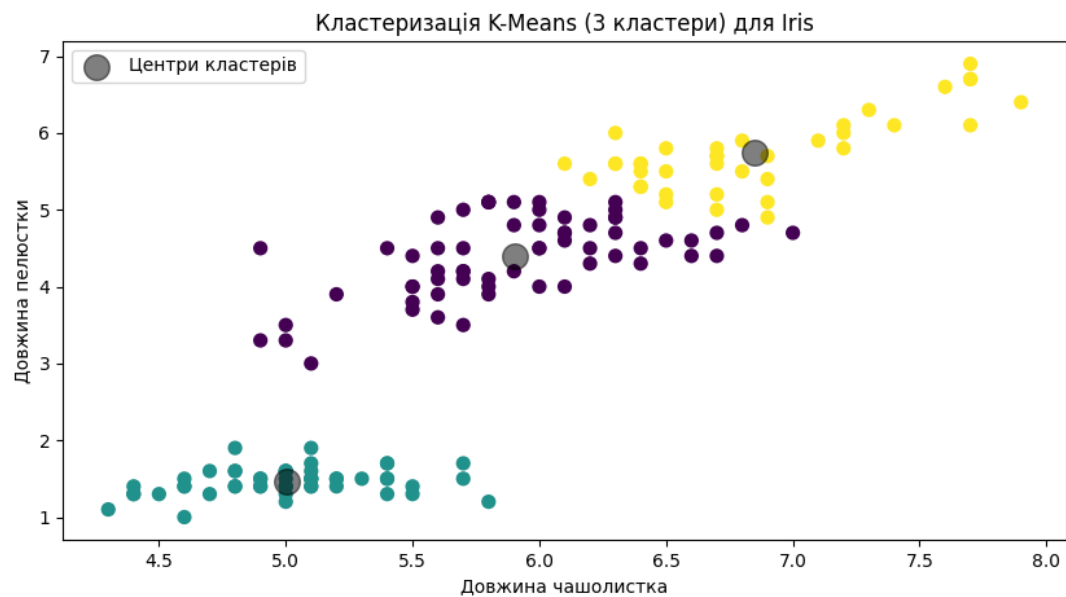
# Відображення центрів кластерів
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:, 1], s=200, c='red', label='Cluster
Centers')

plt.show()

```

Метод кластеризації k-середніх успішно розділив вхідні дані на п'ять кластерів і візуалізував їх межі. Ця процедура дозволила нам чітко розуміти, як дані розподілені між цими п'ятьма групами. Кластери були чітко виділені, і такий підхід може бути корисним для подальшого аналізу або використання в реальних завданнях.

## Завдання 2.8. Кластеризація K-середніх для набору даних Iris



Код програми:

```
import matplotlib
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris

matplotlib.use('TkAgg')

# Завантаження даних Iris
iris = load_iris()
X = iris.data # Ознаки (довжина і ширина чашолистка і пелюстки)
y = iris.target # Мітки класів

# Створення об'єкту KMeans для кластеризації на 3 кластери
kmeans = KMeans(n_clusters=3, random_state=0)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Отримання міток для кожного прикладу даних
y_kmeans = kmeans.predict(X)

# Візуалізація результатів
plt.figure(figsize=(10, 5))

# Відобразимо довжину чашолистка проти довжини пелюстки з кольорами
# на основі міток кластерів
```

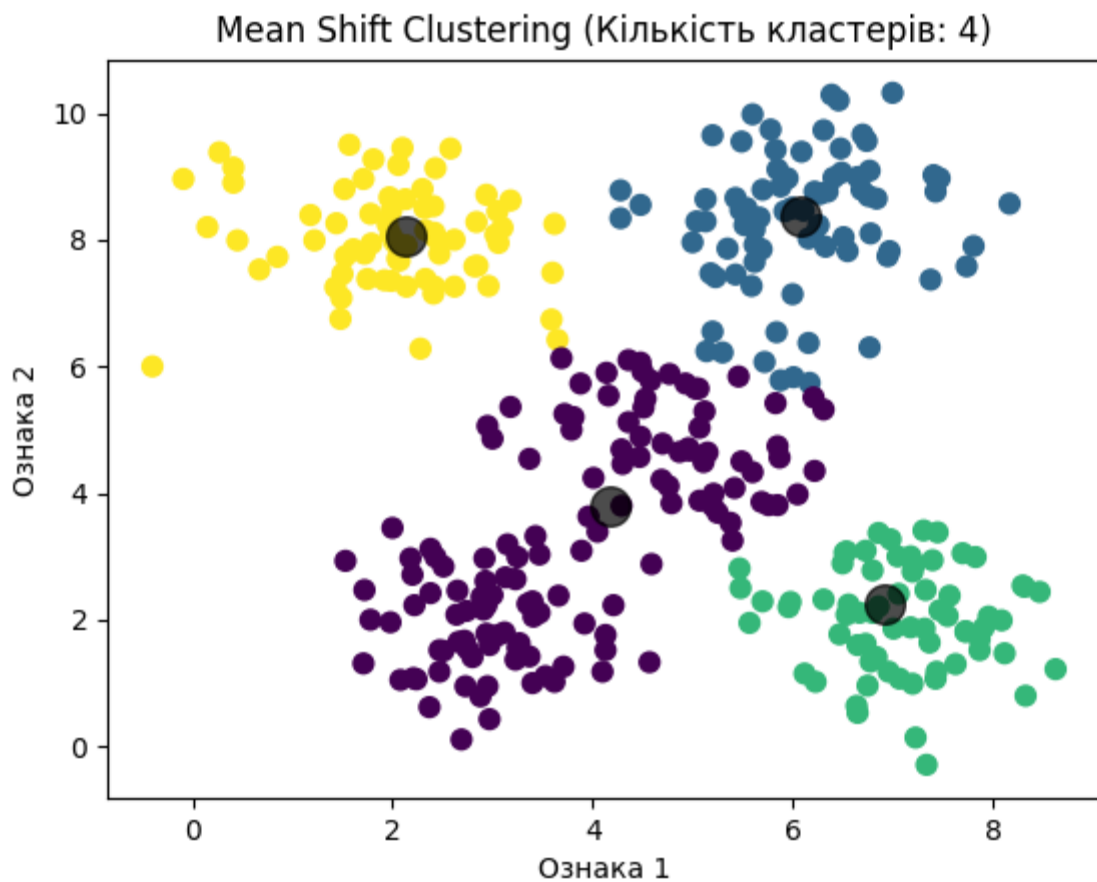
```
plt.scatter(X[:, 0], X[:, 2], c=y_kmeans, cmap='viridis', s=50)
plt.xlabel('Довжина чашолистка')
plt.ylabel('Довжина пелюстки')
plt.title('Кластеризація K-Means (3 кластери) для Iris')

# Відобразимо центри кластерів
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 2], c='black', s=200,
            alpha=0.5, label='Центри кластерів')
plt.legend()

plt.show()
```

**Завдання 2.9.** Оцінка кількості кластерів з використанням методу зсуву середнього

Figure 1



x=-0.49 y=7.43

Код програми:



```

import matplotlib
import numpy as np
from sklearn.cluster import MeanShift, estimate_bandwidth
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

matplotlib.use('TkAgg')

# Завантаження даних з файлу
data = np.loadtxt('data_clustering.txt', delimiter=',') #
Завантаження даних з файлу

# Оцінка оптимальної ширини вікна
bandwidth = estimate_bandwidth(data, quantile=0.2,
n_samples=len(data))

# Створення об'єкта MeanShift для кластеризації з обраною шириною
вікна
ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)

# Навчання моделі кластеризації Mean Shift
ms.fit(data)

# Визначення центрів кластерів
cluster_centers = ms.cluster_centers_

# Оцінка кількості кластерів
num_clusters = len(cluster_centers)

# Візуалізація результатів
plt.scatter(data[:, 0], data[:, 1], c=ms.labels_, cmap='viridis',
s=50)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1],
c='black', s=200, alpha=0.7)

plt.title(f'Mean Shift Clustering (Кількість кластерів:
{num_clusters})')
plt.xlabel('Ознака 1')
plt.ylabel('Ознака 2')
plt.show()

print(f'Оцінена кількість кластерів: {num_clusters}')

```

**Завдання 2.10.** Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

```

"D:\LABS\PRESENT\AI (Python)\LAB3\venv\Scripts\python.exe" "D:/LABS/PRESENT/AI
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
[*****100%*****] 1 of 1 completed
Cluster 1 ==> Alphabet Inc.
Cluster 2 ==> Apple Inc., Amazon.com Inc., Microsoft Corporation
Cluster 3 ==> Tesla, Inc.

Process finished with exit code 0

```

```

{} company_symbol_mapping.json X
D: > LABS > PRESENT > AI (Python) > LAB3 > {} company_symbol_mapping.json > ...
1  {
2      "AAPL": "Apple Inc.",
3      "GOOGL": "Alphabet Inc.",
4      "AMZN": "Amazon.com Inc.",
5      "MSFT": "Microsoft Corporation",
6      "TSLA": "Tesla, Inc."
7  }

```

Код програми:

```

import datetime
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn.covariance import GraphicalLassoCV
from sklearn.cluster import affinity_propagation
import yfinance as yf # Використовуємо yfinance для отримання
фінансових даних

# Ваш шлях до файлу з прив'язками символічних позначень компаній
with open('company_symbol_mapping.json', 'r') as f:
    company_symbols_map = json.load(f)

symbols, names = np.array(list(company_symbols_map.items())).T

# ...
# Завантаження архівних даних котирувань за допомогою yfinance
start_date = datetime.datetime(2010, 1, 1)
end_date = datetime.datetime(2021, 12, 31)

quotes = []

```

```

# Отримання котирувань для всіх компаній
for symbol in symbols:
    try:
        quote = yf.download(symbol, start=start_date,
end=end_date)['Adj Close']
        if not quote.empty:
            quotes.append(quote)
        else:
            print(f"No data available for [{symbol}]")
    except Exception as e:
        print(f"Failed download: [{symbol}]: {e}")

# ...
# Обчислення середнього значення котирувань для всіх компаній
valid_quotes = [quote.values for quote in quotes if not
quote.empty]
min_length = min(len(quote) for quote in valid_quotes)
closing_quotes = np.array([quote[:min_length] for quote in
valid_quotes], dtype=float)

# Обчислення різниці між відкриттям та закриттям
quotes_diff = closing_quotes - closing_quotes[:, 0][:, None]

X = quotes_diff.copy().T
X /= X.std(axis=0)

# Створення моделі графа
edge_model = GraphicalLassoCV()

# Навчання моделі
with np.errstate(invalid='ignore'):
    edge_model.fit(X)

# Створення моделі кластеризації на основі поширення подібності
_, labels = affinity_propagation(edge_model.covariance_)
num_labels = labels.max()

# ...
for i in range(num_labels + 1):
    cluster_names = names[np.where(labels == i)]
    print("Cluster", i + 1, "=>", ', '.join(cluster_names))

```

**Висновки:** використовуючи спеціалізовані бібліотеки і мову програмування Python дослідив методи регресії та неконтрольованої класифікації даних у машинному навчанні.

GitHub: <https://github.com/invincibleee/Artificial-intelligence.git>