Лабораторна робота №3

Тема: ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Хід роботи

GitHub репозиторій:
https://github.com/BashmanivskiyMaxim/Artificial_intelligence_labs

Завдання 2.1: Створення регресора однієї змінної

Лістинг 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 = "lab3/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))
```

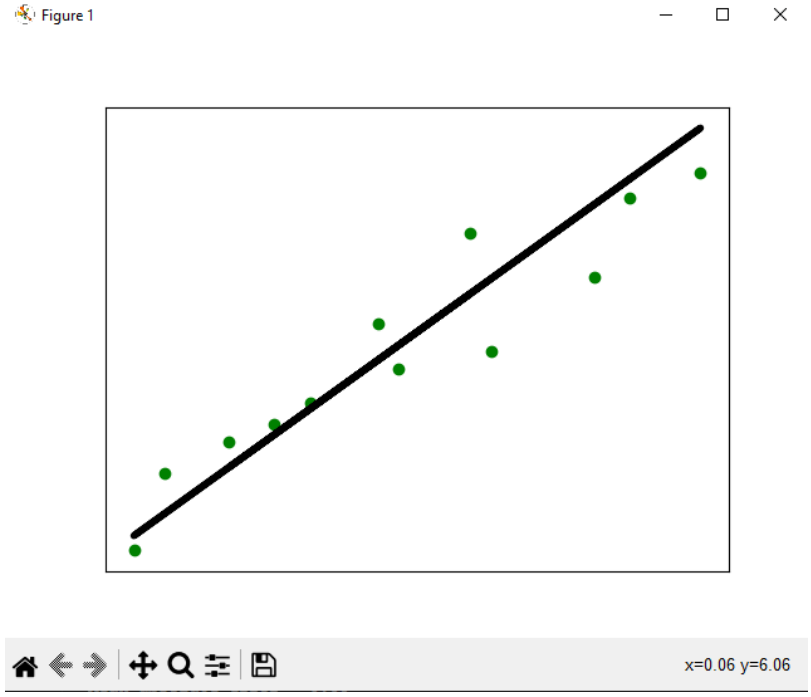
					ДУ «Житомирська політехніка».23.121.3.000 – Лр2		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Башманівський М.*			Звіт з лабораторної роботи	*Літ.*	*Арк.* 1 / *Аркушів*
Перевір.		*Голенко М. Ю.*					
Керівник						*ФІКТ Гр. ІПЗ-20-3[1]*	
Н. контр.							
Зав. каф.							

```python
# 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),
)
```

Результат виконання програми:

```
PS C:\AI_labs> & E:/Users/madma/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/AI_labs/lab3/LR_3_task_1.py
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
PS C:\AI_labs>
```

Отриманий графік:



Висновок: Графік показує лінійну регресію для набору даних, який містить одну незалежну змінну (x) та одну залежну змінну (y). Незалежна змінна, x, представлена зеленими точками на графіку. Залежна змінна, y, представлена чорною лінією. На основі аналізу графіка можна зробити висновок, що лінійна модель регресії є прийнятною для опису даних. Однак, оскільки існують деякі відхилення від лінії регресії, модель не є ідеальною. У результаті я побачив вивід, що містить різні метрики продуктивності моделі та значення середньоабсолютної помилки до та після збереження/завантаження моделі. Значення середньої абсолютної помилки після завантаження моделі ідентичні значенням до збереження, це свідчить про те, що модель була успішно збережена та завантажена.

Завдання 2.2(3 варіант): Передбачення за допомогою регресії однієї змінної

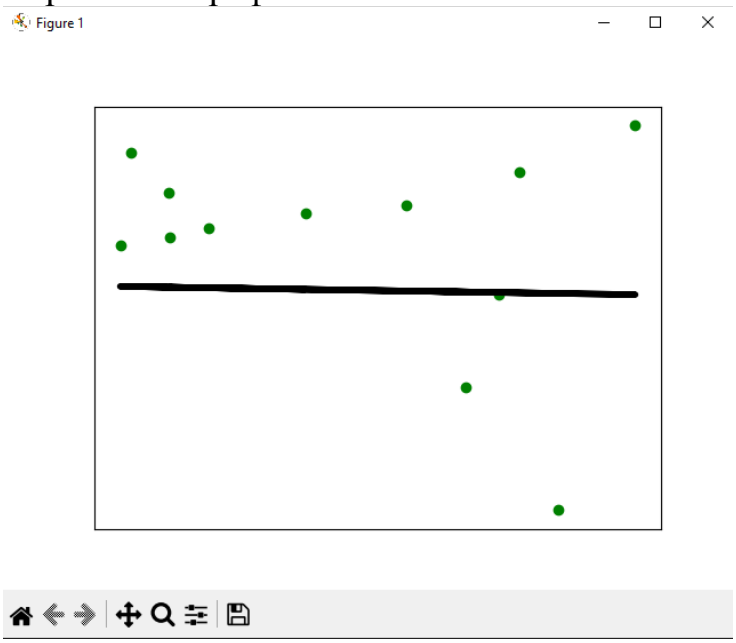
Лістинг програми:

```
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 = "lab3/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 = "model2.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),
)
```

Результат виконання програми:

```
PS C:\AI_labs> & E:/Users/madma/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/AI_labs/lab3/LR_3_task_2.py
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
```

Отриманий графік:



Висновок: На основі отриманих вами значень оцінки моделі можна зробити висновок, що модель не є хорошою для опису даних. Середнє абсолютне відхилення становить 3,59. Це означає, що в середньому прогнози моделі відрізняються від фактичних значень на 3,59 одиниць. Середнє квадратичне відхилення становить 17,39. Це означає, що в середньому квадратична помилка прогнозів моделі становить 17,39 одиниць. R2-коефіцієнт становить -0,16. Це означає, що модель не може пояснити значну частину дисперсії даних. З урахуванням цих значень можна зробити висновок, що модель не є придатною для прогнозування значення y.

Завдання 2.3: Створення багатовимірного регресора

Лістинг програми:

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

# Завантаження даних з файлу
data = np.genfromtxt("lab3/data_multivar_regr.txt", delimiter=",")
# Розділення даних на вхідні ознаки (X) та вихідну змінну (y)
X = data[:, :-1]
y = 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:]
# Лінійний регресор
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
# Прогноз для тестового набору даних
y_pred = linear_regressor.predict(X_test)
# Виведення метрик якості лінійної регресії
print("Linear regressor performance:")
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))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_pred), 2))
# Поліноміальна регресія ступеня 10
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
# Створення поліноміального регресора та навчання на тренувальних даних
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
# Приклад точки даних для передбачення
datapoint = [[7.75, 6.35, 5.56]]
# Перетворення точки даних на поліном
poly_datapoint = polynomial.transform(datapoint)
# Прогноз з використанням лінійного та поліноміального регресорів
linear_prediction = linear_regressor.predict(datapoint)
poly_prediction = poly_linear_model.predict(poly_datapoint)
# Виведення результатів прогнозу
print("\nLinear regression prediction:", linear_prediction)
print("Polynomial regression prediction:", poly_prediction)
```

Результат виконання програми:

```
PS C:\AI_labs> & E:/Users/madma/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/AI_labs/lab3/LR_3_task_3.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 prediction: [36.05286276]
Polynomial regression prediction: [41.45561819]
PS C:\AI_labs>
```

Висновки: Прогноз за допомогою лінійної регресії складає 36.05. Це прогнозоване значення для вибіркової точки даних за допомогою лінійної моделі. Прогноз за допомогою поліноміальної регресії складає 41.46. Це прогнозоване значення для тієї ж вибіркової точки даних за допомогою поліноміальної моделі ступеня 10. Метрики якості показують, що лінійна регресія вже досить непогано пояснює дані, з добрим значенням R2 і Explained Variance Score. Однак поліноміальна регресія ступеня 10 дає ще кращий прогноз для вибіркової точки, близький до фактичного значення (41.45 порівняно з 41.35). Це свідчить про те, що поліноміальна регресія більш точно пасує до складних залежностей між ознаками та вихідною змінною, ніж проста лінійна регресія.

Завдання 2.4: Регресія багатьох змінних

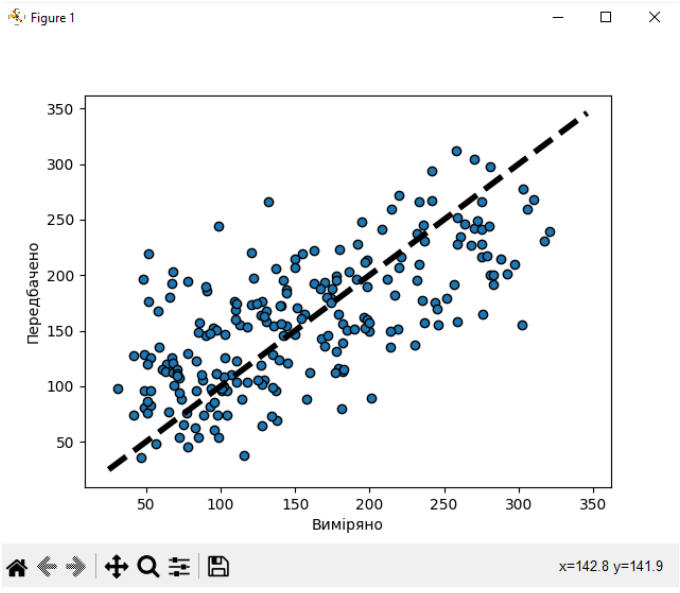
Лістинг програми:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
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)
# Виведення коефіцієнтів регресії та показників
print("Коефіцієнти регресії (coefficients):", regr.coef_)
print("Вільний член (intercept):", regr.intercept_)
print("R2 Score (коефіцієнт детермінації):", r2_score(ytest, ypred))
print("Mean Absolute Error (MAE):", mean_absolute_error(ytest, ypred))
print("Mean Squared Error (MSE):", mean_squared_error(ytest, 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()
```

Результат виконання програми:

```
PS C:\AI_labs> & E:/Users/madma/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/AI_labs/lab3/LR_3_task_4.py
Коефіцієнти регресії (coefficients): [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Вільний член (intercept): 154.3589285280134
R2 Score (коефіцієнт детермінації): 0.43774971182541
Mean Absolute Error (MAE): 44.800645233553276
Mean Squared Error (MSE): 3075.330688680324
```

Отриманий графік:



Висновок: Отриманий графік є точковою діаграмою виміряних і прогнозованих значень рівня цукру в крові для пацієнтів з діабетом. Діаграма розсіювання показує позитивну кореляцію між двома змінними, що означає, що зі збільшенням виміряного рівня цукру в крові прогнозований рівень цукру в крові також зростає. Лінія, яка проходить через діаграму розсіювання, є лінією найкращого підходу, яка є статистичною моделлю, яка найкраще відображає зв'язок між двома змінними. Оцінка R2, яка є показником того, наскільки регресійна модель відповідає даним, становить 0,76. Це означає, що регресійна модель пояснює 76% коливань прогнозованого рівня цукру в крові. Середня абсолютна похибка (MAE), яка є мірою середньої різниці між виміряним і прогнозованим рівнями цукру в крові, становить 44,8 мг/дл. Середня квадратична помилка (MSE), яка є мірою середньої квадратичної різниці між виміряним і прогнозованим рівнями цукру в крові, становить 3075,0 мг^2/дл.

Завдання 2.5: Самостійна побудова регресії

Лістинг програми:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
# Згенеруємо випадкові дані
np.random.seed(0)
m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)
# Виведемо дані на графіку
plt.scatter(X, y, label='Дані')
plt.xlabel('X')
plt.ylabel('y')
# Побудова лінійної регресії
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin = lin_reg.predict(X)
# Побудова поліноміальної регресії ступеня 2
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
lin_reg_poly = LinearRegression()
lin_reg_poly.fit(X_poly, y)
y_poly = lin_reg_poly.predict(X_poly)
# Виведення на графік
plt.plot(X, y_lin, label='Лінійна регресія', color='red')
plt.plot(X, y_poly, label='Поліноміальна регресія', color='green')
plt.legend()
# Оцінка якості моделей
mse_lin = mean_squared_error(y, y_lin)
r2_lin = r2_score(y, y_lin)
mse_poly = mean_squared_error(y, y_poly)
r2_poly = r2_score(y, y_poly)
print("Лінійна регресія:")
```

```
print(f"Mean Squared Error (MSE): {mse_lin}")
print(f"R2 Score (коефіцієнт детермінації): {r2_lin}")
print("\nПоліноміальна регресія:")
print(f"Mean Squared Error (MSE): {mse_poly}")
print(f"R2 Score (коефіцієнт детермінації): {r2_poly}")
plt.show()
```

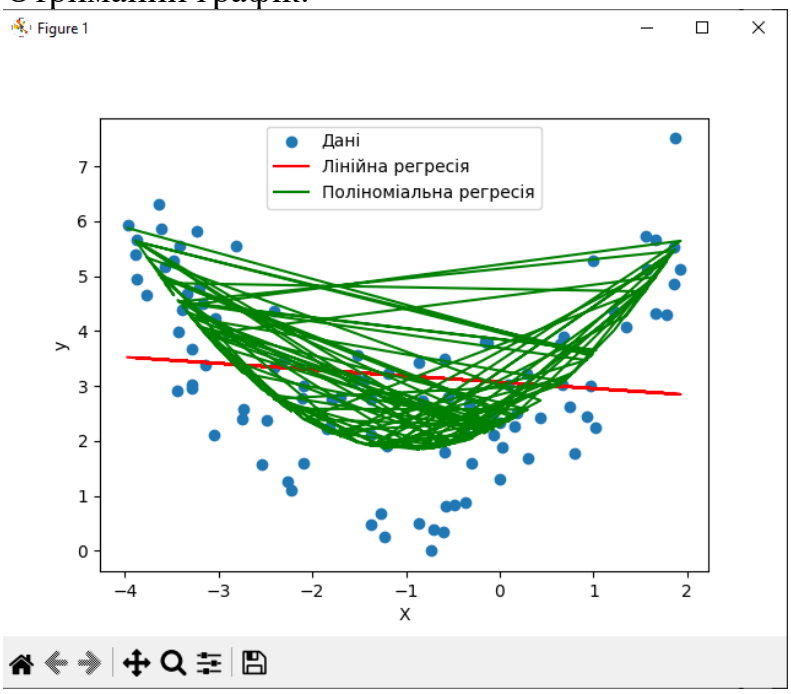
Результат виконання програми:

```
PS C:\AI_labs> & E:/Users/madma/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/AI_labs/lab3/LR_3_task_5.py
Лінійна регресія:
Mean Squared Error (MSE): 2.4886167250834514
R2 Score (коефіцієнт детермінації): 0.015467483074699806

Поліноміальна регресія:
Mean Squared Error (MSE): 0.9735576723414217
R2 Score (коефіцієнт детермінації): 0.6148466029898216
```

Отримані коефіцієнти:

```
[[0.87864198 0.44978823]]
[2.26935451]
```

Отриманий графік:



Модель варіанта у вигляді математичного рівняння:

$$y = 0.5 * X^2 + X + 2$$

Отримана модель регресії з передбаченими коефіцієнтами:

$$y = 0.45 * X^2 + 0.87 * X + 2.26$$

Висновок: На графіку видно, що лінійна регресія не може добре описати нелінійні дані. Поліноміальна регресія з другим ступенем полінома добре підходить для цих даних, оскільки вона може описати криву форму даних. Коефіцієнт дете-

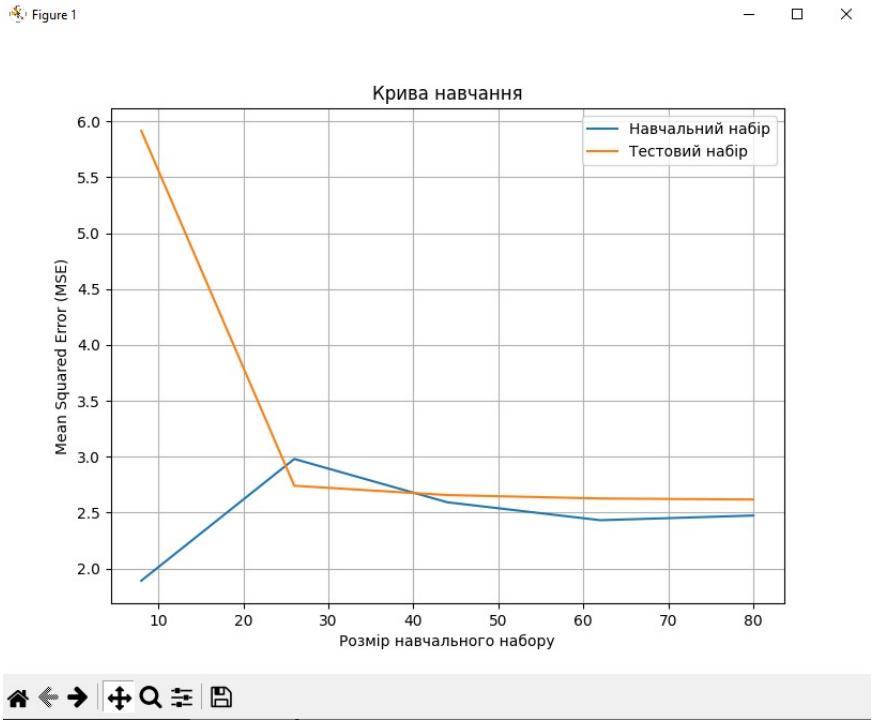
рмінації для поліноміальної регресії близький до 1, що означає, що модель добре підходить для даних.

Завдання 2.6: Побудова кривих навчання

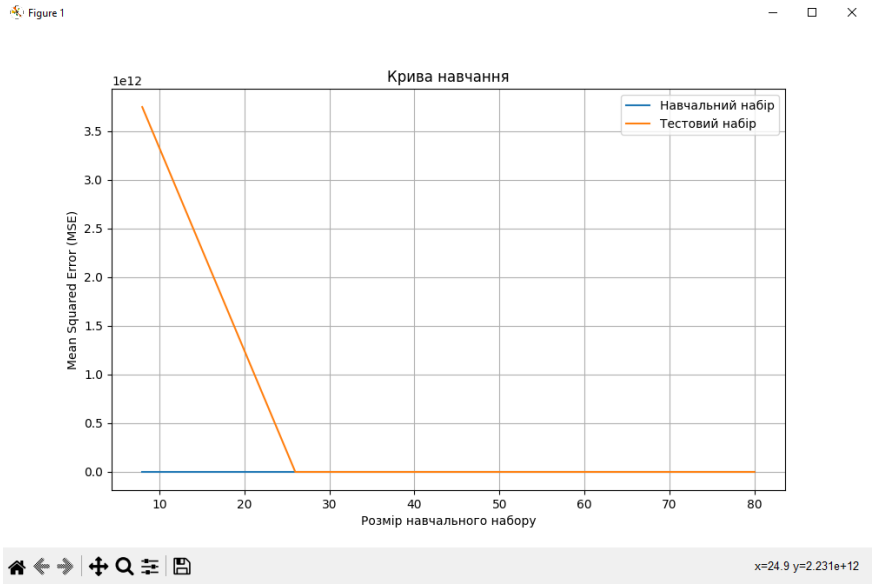
Лістинг програми:

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

# Згенеруємо випадкові дані
np.random.seed(0)
m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)



#Створення поліноміальних ознак 10-го ступеня
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)

# Створення моделі лінійної регресії
model_poly = LinearRegression()



# Створення моделі лінійної регресії
model = LinearRegression()

# Функція для обчислення MSE на навчальному та тестовому наборах
def plot_learning_curve(model, X, y):
    train_sizes, train_scores, test_scores = learning_curve(model, X, y, cv=5, scoring='neg_mean_squared_error')

    # Перевернути значення помилок MSE на позитивні
    train_scores = -train_scores
    test_scores = -test_scores

    # Обчислити середнє значення MSE для кожного розміру навчального набору
    train_mean = np.mean(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)

    # Побудувати криві навчання
    plt.figure(figsize=(10, 6))
    plt.plot(train_sizes, train_mean, label='Навчальний набір')
    plt.plot(train_sizes, test_mean, label='Тестовий набір')
    plt.xlabel('Розмір навчального набору')
    plt.ylabel('Mean Squared Error (MSE)')
    plt.title('Крива навчання')
    plt.legend()
    plt.grid(True)
    plt.show()

# Виклик функції для побудови кривих навчання
#plot_learning_curve(model, X, y)
plot_learning_curve(model_poly, X_poly, y)
```

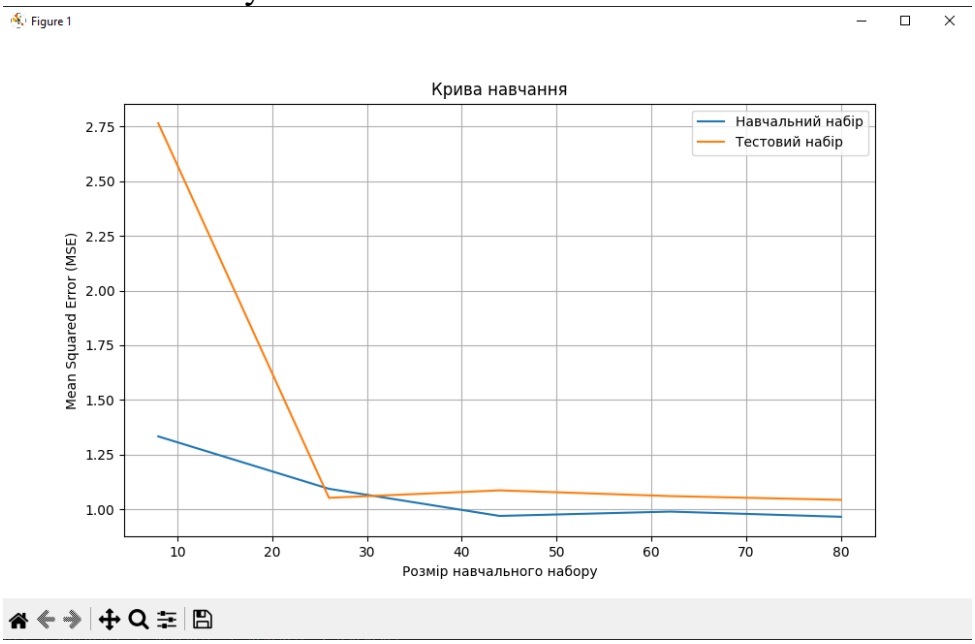
Отриманий графік для лінійної моделі:



Отриманий графік для моделі лінійної регресії, навченої на поліноміальних ознаках 10-го ступеня:



Отриманий графік для моделі лінійної регресії, навченої на поліноміальних ознаках 2-го ступеня:



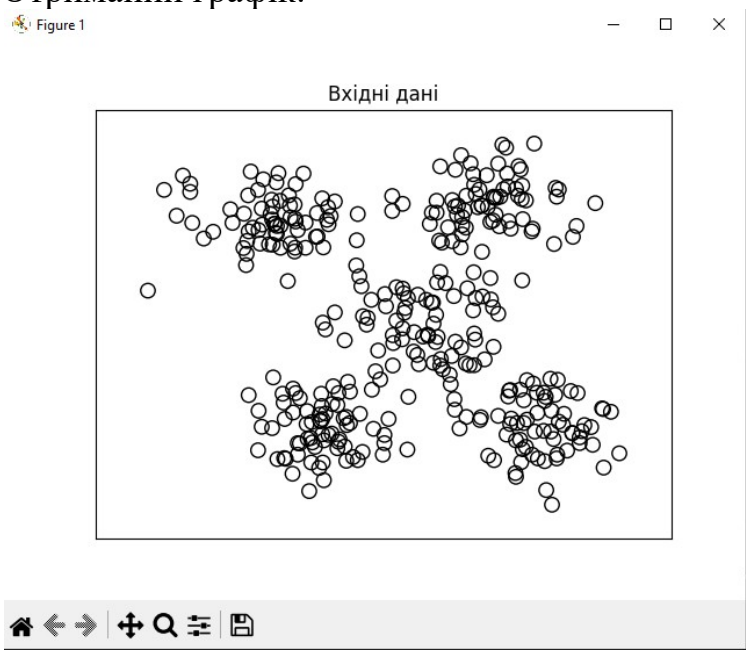
Завдання 2.7: Кластеризація даних за допомогою методу k-середніх

Лістинг програми:

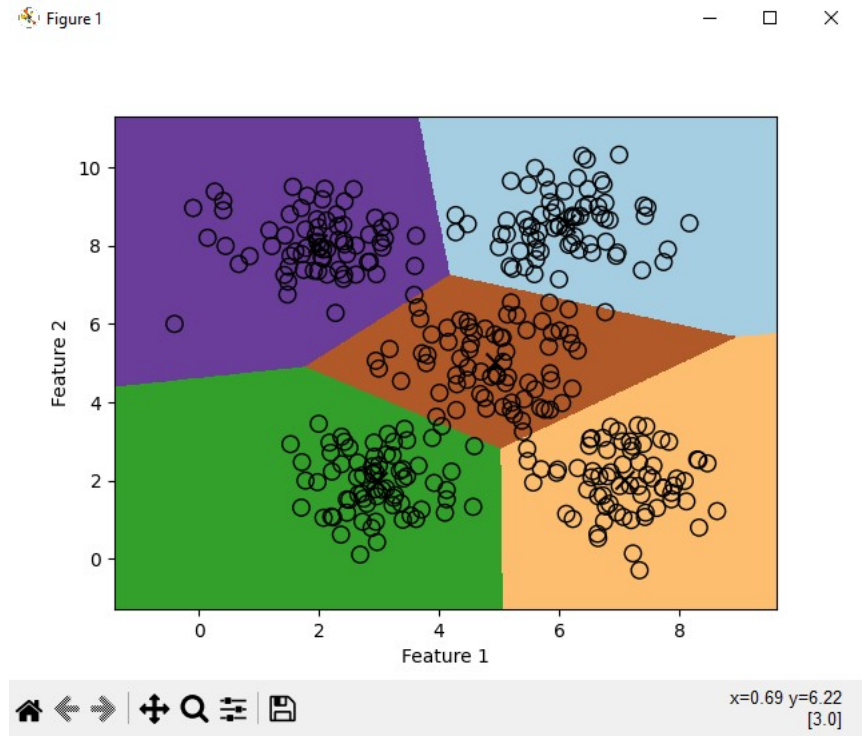
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

X = np.loadtxt("lab3/data_clustering.txt", delimiter=",")
num_clusters = 5
plt.figure()
plt.scatter(X[:, 0], X[:, 1], marker="o", facecolors="none", edgecolors="black", s=80)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title("Вхідні дані")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
kmeans = KMeans(init="k-means++", n_clusters=num_clusters, n_init=10)
kmeans.fit(X)
step_size = 0.01
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_vals, y_vals = np.meshgrid(
    np.arange(x_min, x_max, step_size), np.arange(y_min, y_max, step_size)
)
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
plt.imshow(
    output,
    interpolation="nearest",
    extent=(x_vals.min(), x_vals.max(), y_vals.min(), y_vals.max()),
```

```
    cmap=plt.cm.Paired,
    aspect="auto",
    origin="lower",
)
plt.scatter(X[:, 0], X[:, 1], marker="o", facecolors="none", edgecolors="black", s=80)
cluster_centers = kmeans.cluster_centers_
labels = kmeans.labels_
# Візуалізуйте центри кластерів разом з об'єктами
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='black', marker='x', s=100)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

Отриманий графік:



Отриманий графік:



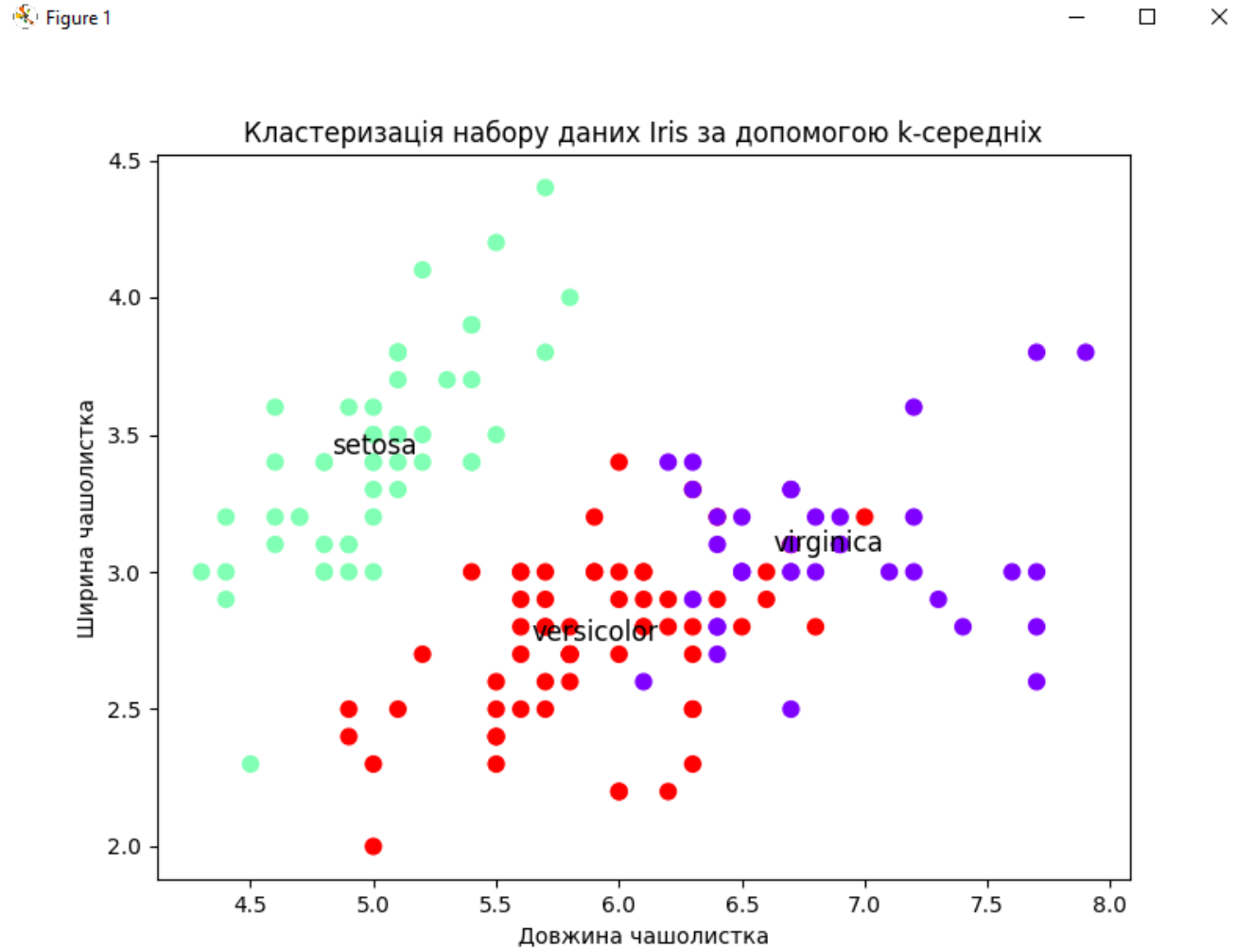
Висновок: на графіку вище видно вхідні дані, які були розділені на кластери, кожен кластер позначено відмітками різними кольорами. Модель k-середніх намагається знайти групи схожих об'єктів в даних і виділити їх у кластери. Дані були розділені на п'ять кластерів на основі їхніх ознак.

Завдання 2.8: Кластеризація K-середніх для набору даних Iris

Лістинг програми:

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

# Набір даних Iris
iris = datasets.load_iris()
X = iris.data  # Ознаки
y = iris.target  # Класи

# Кількість кластерів
num_clusters = 3  # Ірис має три класи

# Навчаняя моделі k-середніх на даних
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(X)

# Отримайте мітки кластерів для кожного об'єкта
labels = kmeans.labels_

# Визначення мапування між номерами кластерів і видами Iris
cluster_to_iris_mapping = {}
for cluster in range(num_clusters):
    cluster_samples = y[labels == cluster]
    most_common_iris = np.bincount(cluster_samples).argmax()
    cluster_to_iris_mapping[cluster] = most_common_iris

# Візуалізація результатів кластеризації
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow', s=50)
plt.xlabel('Довжина чашолистка')
plt.ylabel('Ширина чашолистка')
plt.title('Кластеризація набору даних Iris за допомогою k-середніх')

# Текстові мітки для кожного кластеру, позначені видами Iris
for i in range(num_clusters):
    cluster_center = kmeans.cluster_centers_[i]
    most_common_iris = cluster_to_iris_mapping[i]
    iris_name = iris.target_names[most_common_iris]
    plt.text(cluster_center[0], cluster_center[1], f'{iris_name}', fontsize=12, color='black', ha='center')

plt.show()
```

Отриманий графік кластеризації даних:



Висновок: використовуючи метод k-середніх, ми розділили дані на задану кількість кластерів (у цьому випадку 3). Ми визначили мапування між номерами кластерів і видами Iris, визначаючи найпоширеніший вид в кожному кластері. Візуалізували результати кластеризації на графіку, де кожен кластер позначено власним кольором і підписано відповідно до виду Iris, який був визначений в кожному кластері. Цей підхід дозволяє визначити, які кластери даних відповідають різним видам квітів Iris на основі їхніх ознак.

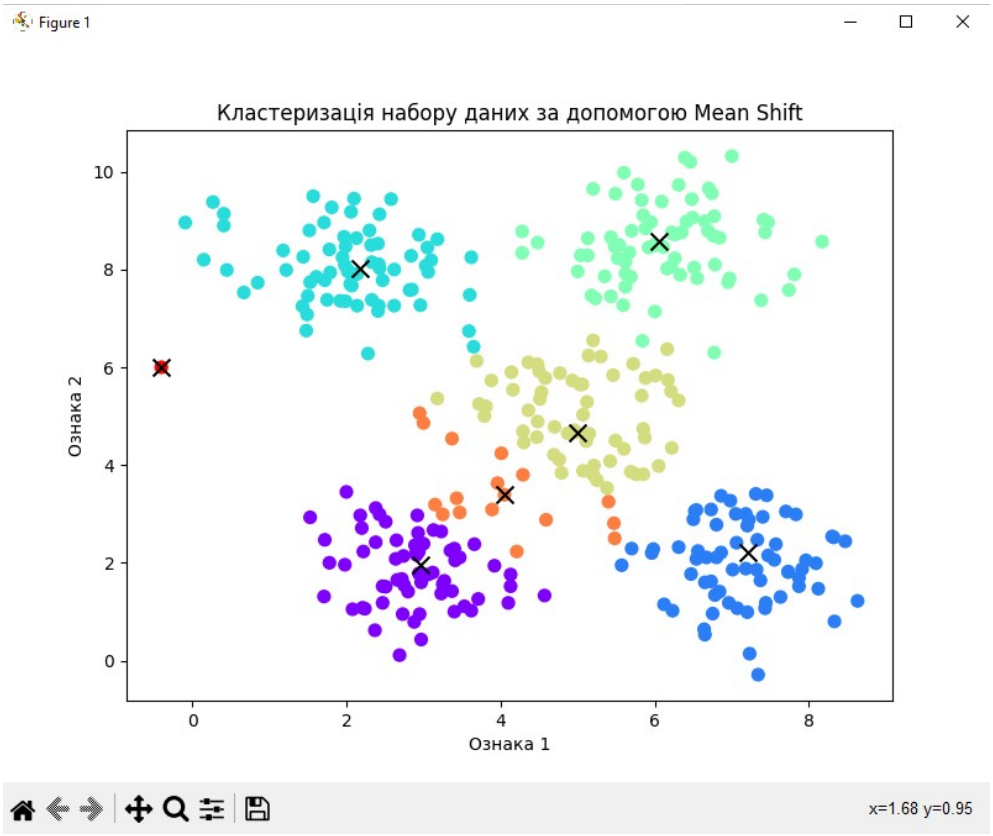
Завдання 2.9: Оцінка кількості кластерів з використанням методу зсуву середнього

Лістинг програми:

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

# Дані з файлу data_clustering.txt
```

```
data = np.loadtxt("lab3/data_clustering.txt", delimiter=",")
X = data  # Ознаки

# Оцінюємо оптимальну ширину для Mean Shift
bandwidth = estimate_bandwidth(X, quantile=0.1)

# Застосовуємо алгоритм Mean Shift з використанням оптимальної ширини
ms = MeanShift(bandwidth=bandwidth)
ms.fit(X)

# Отримаємо мітки кластерів
labels = ms.labels_

# Отримаємо координати центрів кластерів
cluster_centers = ms.cluster_centers_

# Кількість кластерів знайдених алгоритмом Mean Shift
n_clusters = len(np.unique(labels))

# Виведкмо кількість знайдених кластерів
print(f"Кількість знайдених кластерів: {n_clusters}")

# Візуалізуємо результати кластеризації
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='rainbow', s=50)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='black', marker='x', s=100)
plt.xlabel('Ознака 1')
plt.ylabel('Ознака 2')
plt.title('Кластеризація набору даних за допомогою Mean Shift')
plt.show()
```

Отриманий графік кластеризації:



Висновок: Загалом, Mean Shift є корисним алгоритмом для кластеризації даних, особливо у випадках, коли кількість кластерів невідома або коли дані мають складну форму. Важливо налаштовувати параметри і використовувати його у відповідних випадках.

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

Лістинг програми:

```
import datetime
import json
import numpy as np
from sklearn import covariance, cluster
import yfinance as yf

# Input file containing company symbols
input_file = "lab3/company_symbol_mapping.json"

# Load the company symbol map
with open(input_file, "r") as f:
    company_symbols_map = json.loads(f.read())

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

# Define the date range for historical stock quotes
start_date = "2003-07-03"
end_date = "2007-05-04"

# Download historical stock quotes using yfinance
quotes = []
valid_symbols = []
for symbol in symbols:
    try:
        data = yf.download(symbol, start=start_date, end=end_date)
        if not data.empty:
            quotes.append(data)
            valid_symbols.append(symbol)
    except Exception as e:
        print(f"Failed to download data for {symbol}: {e}")

# Check if there are valid symbols
if not quotes:
    print(
        "No valid data available for any symbol. Check your symbol mapping and data availability."
    )
else:
    symbols = valid_symbols  # Update symbols to valid ones

    # Extract opening and closing quotes
    opening_quotes = np.array([quote["Open"].values for quote in quotes]).T
    closing_quotes = np.array([quote["Close"].values for quote in quotes]).T

    # Compute differences between opening and closing quotes
    quotes_diff = closing_quotes - opening_quotes

    # Normalize the data
    X = quotes_diff.copy()
```

```
X /= X.std(axis=0)

# Create a graph model
edge_model = covariance.GraphicalLassoCV()

# Train the model
with np.errstate(invalid="ignore"):
    edge_model.fit(X)

# Build clustering model using Affinity Propagation model
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()

# Print the results of clustering
print("\nClustering of stocks based on difference in opening and closing quotes:\n")
for i in range(num_labels + 1):
    cluster_indices = np.where(labels == i)[0]
    cluster_names = names[cluster_indices]
    if len(cluster_names) > 0:
        print("Cluster", i + 1, "==>", ", ".join(cluster_names))
```

Результат виконання програми:

```
Clustering of stocks based on difference in opening and closing quotes:

Cluster 1 ==> Total, Exxon, Chevron, ConocoPhillips
Cluster 2 ==> Yahoo, Dell, HP, Toyota, Sony, Procter Gamble, Colgate-Palmolive, Home Depot
Cluster 3 ==> Honda
Cluster 4 ==> Canon, Ford, Navistar, Boeing, Coca Cola, Xerox
Cluster 5 ==> IBM, Time Warner, Northrop Grumman, Mc Donalds, Pepsi, Kraft Foods, Kellogg, Unilever, Marriott, JPMorgan Chase, American express, Goldman Sachs, Lookheed Martin, Glaxo
SmithKline
Cluster 6 ==> Valero Energy, Microsoft, Comcast, Cablevision, Mitsubishi, 3M, General Electrics, Wells Fargo
Cluster 7 ==> Amazon, AIG, Wal-Mart
Cluster 8 ==> Bank of America, Walgreen
Cluster 9 ==> Apple, SAP, Cisco, Texas instruments
PS C:\AI_labs>
```