ЛАБОРАТОРНА РОБОТА №3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Посилання на проект: https://github.com/ipz202-rev/AI-lab3

Хід роботи:

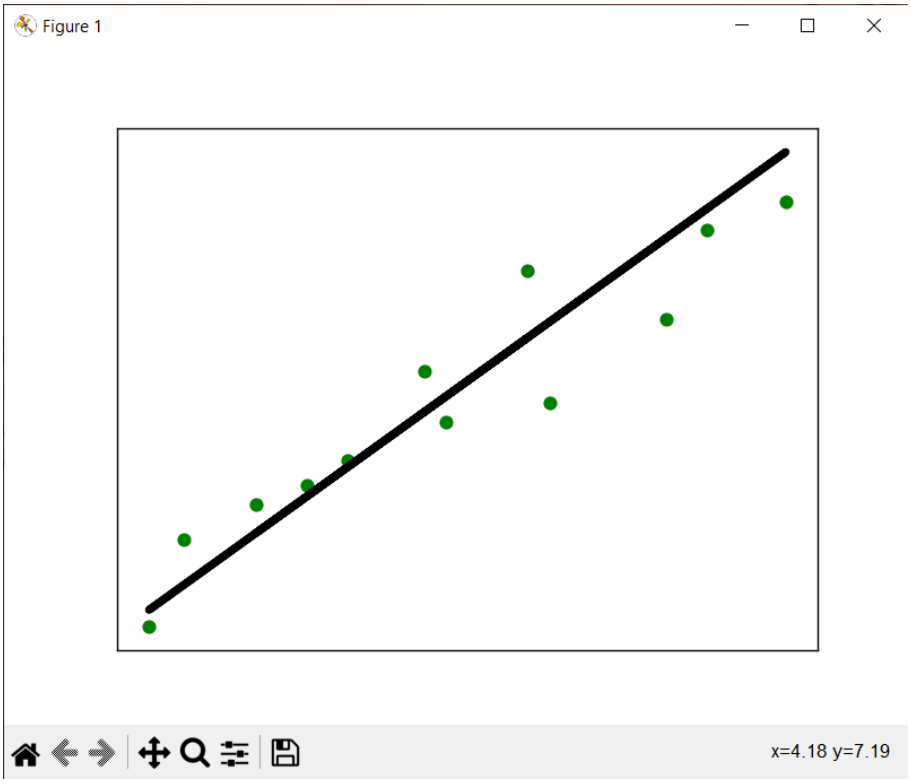
Завдання №3.1. Створення регресора однієї змінної.

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

Лістинг файлу LR_3_task_1.py:

```
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))
```



```
D:\course-4\semester-1\ai\lab3_proj
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
```

Рис.3.1.1 – 3.1.2. Результат виконання коду.

За отриманими результатами регресійна модель на основі однієї змінної має високу точність, яка виражена в низьких значеннях Mean Absolute Error (MAE) і Mean Squared Error (MSE). Модель також демонструє високу здатність пояснювати варіацію в даних, яка виражена в Explained Variance Score і R2 Score. Значення 0.86 для обох цих показників свідчать про те, що модель може пояснити близько 86% варіації в пояснюваній змінній. Значення Median Absolute Error досить низьке (0.51), що означає стабільність моделі.

Завдання №3.2. Передбачення за допомогою регресії однієї змінної.

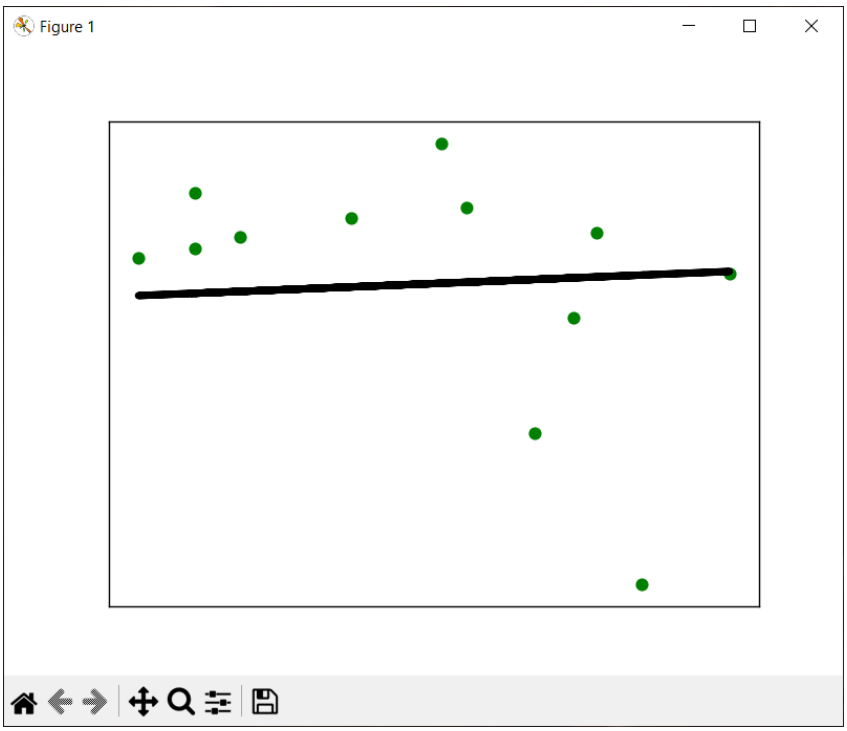
Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі.

№ за списком	24
№ варіанту	4

Лістинг файлу LR_3_task_2.py:

```
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_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 = '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))
```




```
D:\course-4\semester-1\ai\lab3_pro
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

Process finished with exit code 0
```

Рис.3.2.1 – 3.2.2. Результат виконання коду.

Створена регресійна модель на основі однієї змінної демонструє дуже низьку точність та нездатність пояснити варіацію в даних. Отримані значення MAE та MSE високі, що означає, що модель допускає значні помилки у прогнозуванні цільової змінної. Значення Explained Variance Score і R2 Score вказують на те, що модель не здатна пояснити варіацію в даних. Модель є непридатною для прогнозування значень цільової змінної і потребує подальшого вдосконалення або зміни підходу до моделювання.

Завдання №3.3. Створення багатовимірного регресора.

Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Лістинг файлу LR_3_task_3.py:

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
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))

# 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))
```

```
D:\course-4\semester-1\ai\lab3_proje
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:
 [41.46197721]

Process finished with exit code 0
```

Рис.3.3.1. Результат виконання коду.

Коефіцієнт поліноміальної регресії більший за коефіцієнт лінійної регресії, що означає, що в поліноміальній регресії більший вплив вхідних змінних на цільову змінну, і така модель більш чутлива до змін в даних, порівняно з лінійною. Враховуючи, що результат, отриманий з використанням поліноміальної регресії, ближче до 41.35, ця модель краще підходить для вхідного набору даних.

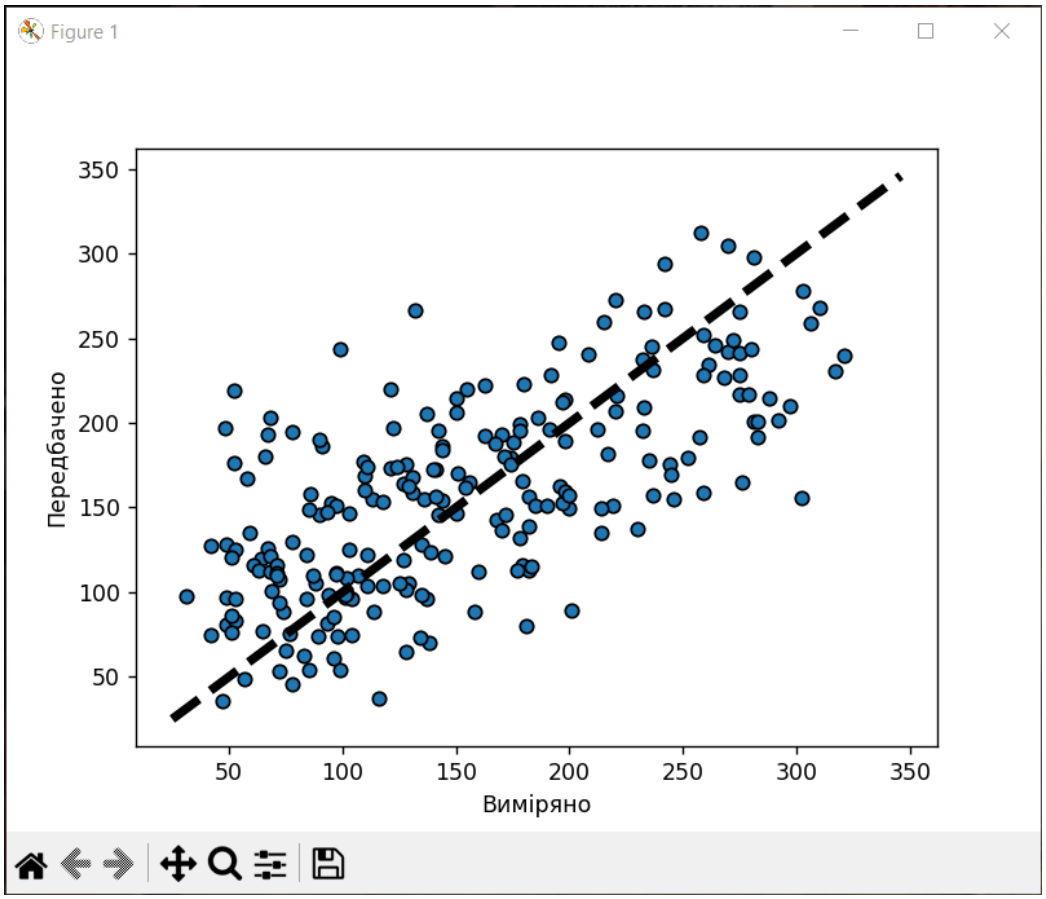
Завдання №3.4. Регресія багатьох змінних.

Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets.

Лістинг файлу LR_3_task_4.py:

```
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)
```

```
print("Linear regressor performance:")
print("regr.coef =", np.round(regr.coef_, 2))
print("regr.intercept =", round(regr.intercept_, 2))
print("R2 score =", round(r2_score(ytest, ypred), 2))
print("Mean absolute error =", round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =", round(mean_squared_error(ytest, ypred), 2))

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()
```




```
D:\course-4\semester-1\ai\lab3_project\venv\Scripts\python.exe D:\course-4\semester-1\a
Linear regressor performance:
regr.coef = [ -20.4  -265.89  564.65  325.56 -692.16  395.56   23.5   116.36  843.95
   12.72]
regr.intercept = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33

Process finished with exit code 0
```

Рис.3.4.1 – 3.4.2. Результат виконання коду.

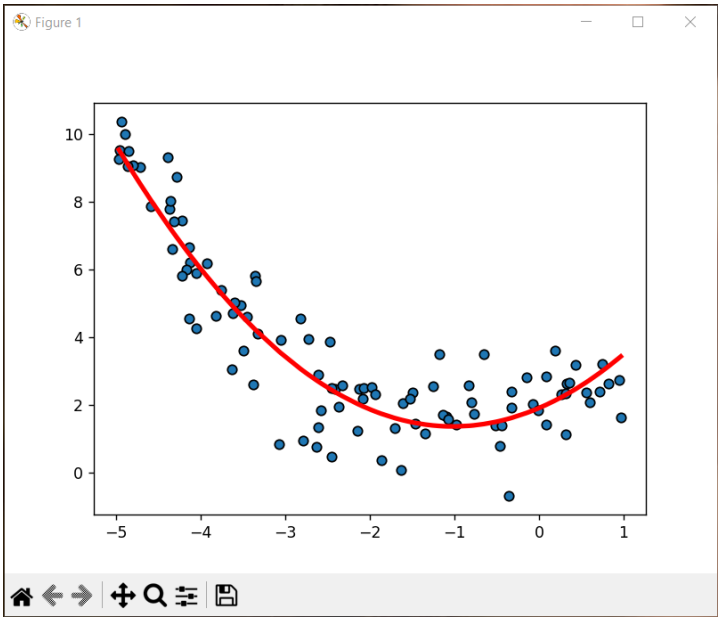
Результати лінійної регресії показують, що модель має великі помилки в прогнозах (високі MAE і MSE), та не може добре пояснити варіацію в даних (низький R2 Score). Коефіцієнти регресії вказують на різний вплив вхідних змінних на цільову змінну. Загалом, модель потребує покращення або розгляду інших підходів для досягнення точніших прогнозів.

Завдання №3.5. Самостійна побудова регресії.

Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

Лістинг файлу LR_3_task_5.py:

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

m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)

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

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_poly, y)
y_predict = poly_linear_model.predict(X_poly)

print('X[0]:', X[0])
print('X_poly:', X_poly[0])
print("Polynomial regressor coefficient:", poly_linear_model.coef_)
print("Polynomial regressor intercept:", poly_linear_model.intercept_)

X_flattened = X.flatten()
y_pred_flattened = y_predict.flatten()
sorted_indices = np.argsort(X_flattened)
X_arr = X_flattened[sorted_indices]
y_pred = y_pred_flattened[sorted_indices]

plt.figure()
plt.scatter(X, y, edgecolors=(0, 0, 0))
plt.plot(X_arr, y_pred, color="red", linewidth=3)
plt.show()
```




```
D:\course-4\semester-1\ai\lab3_project\venv\Scripts\python.ex
X[0]: [-4.59084615]
X_poly: [-4.59084615 21.07586837]
Polynomial regressor coefficient: [[1.06641501 0.52381176]]
Polynomial regressor intercept: [1.91934808]

Process finished with exit code 0
```

Рис.3.5.1. Результат виконання коду.

Модель у вигляді математичного рівняння:

$y = 0.5x_1^2 + x_1 + 2 +$ гауссів шум

Отримана модель регресії з передбаченими коефіцієнтами:

$y = 0.524x_1^2 + 1.066x_1 + 1.919$

Отримані коефіцієнти близькі до модельних, що означає, що модель навчена правильно.

Завдання №3.6. Побудова кривих навчання.

Побудуйте криві навчання для ваших даних у попередньому завданні.

Лістинг файлу LR_3_task_6.py:

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

```
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)


def plot_learning_curves(model, X_, y_):
    X_train, X_val, y_train, y_val = train_test_split(X_, y_, test_size=0.2)
    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_predict, y_train[:m_]))
        val_errors.append(mean_squared_error(y_val_predict, y_val))
    plt.ylim(0, 3)
    plt.plot(np.sqrt(train_errors), "r-+", lw=2, label="train")
    plt.plot(np.sqrt(val_errors), "b-", lw=3, label="val")
    plt.show()


lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)

polynomial_regression = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)

polynomial_regression = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)
```

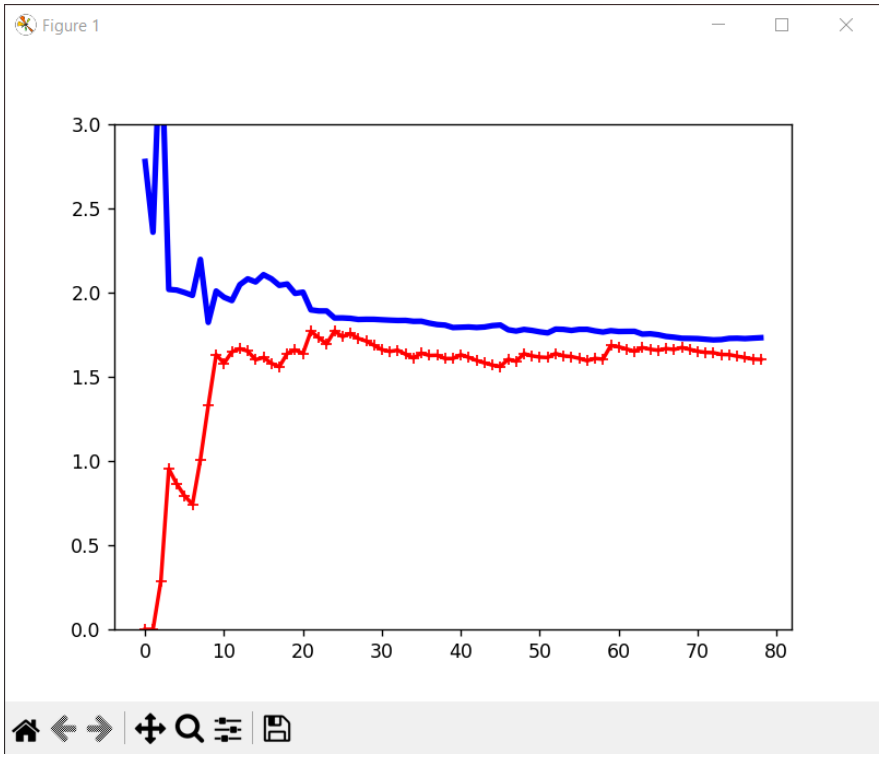


Рис.3.6.1. Криві навчання для лінійної моделі.

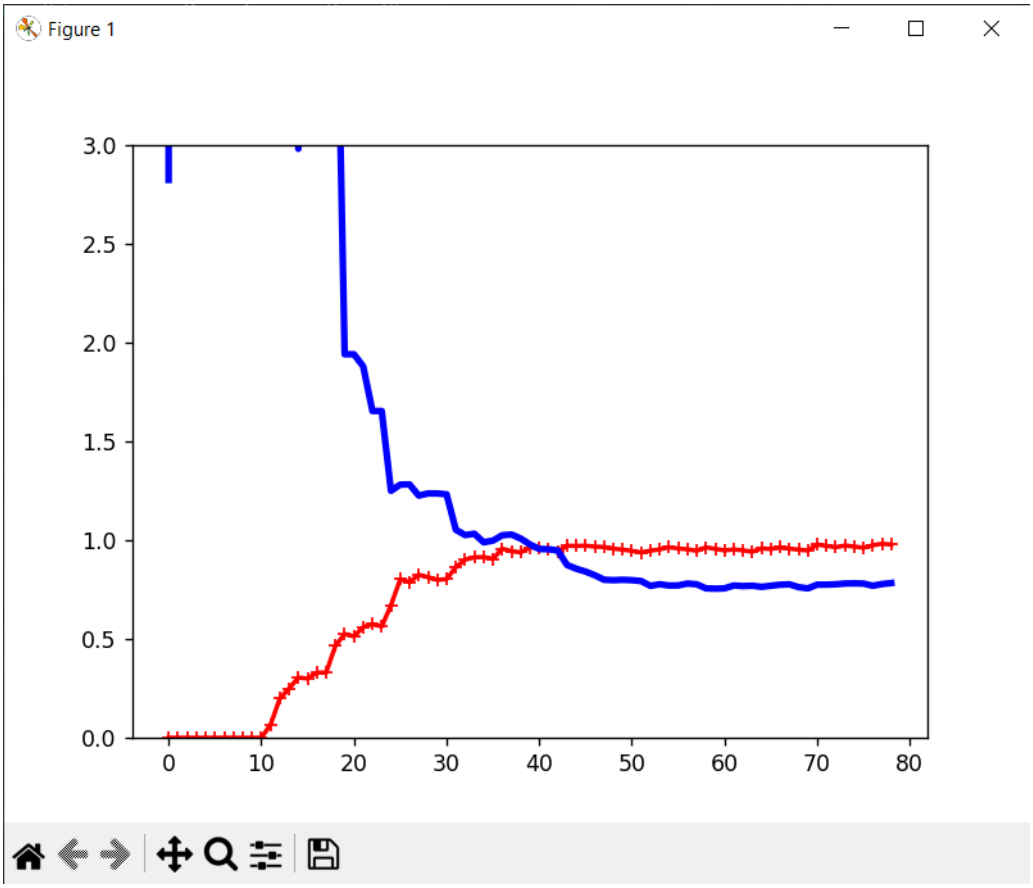


Рис.3.6.2. Криві навчання для поліноміальної моделі 10-го ступеня.

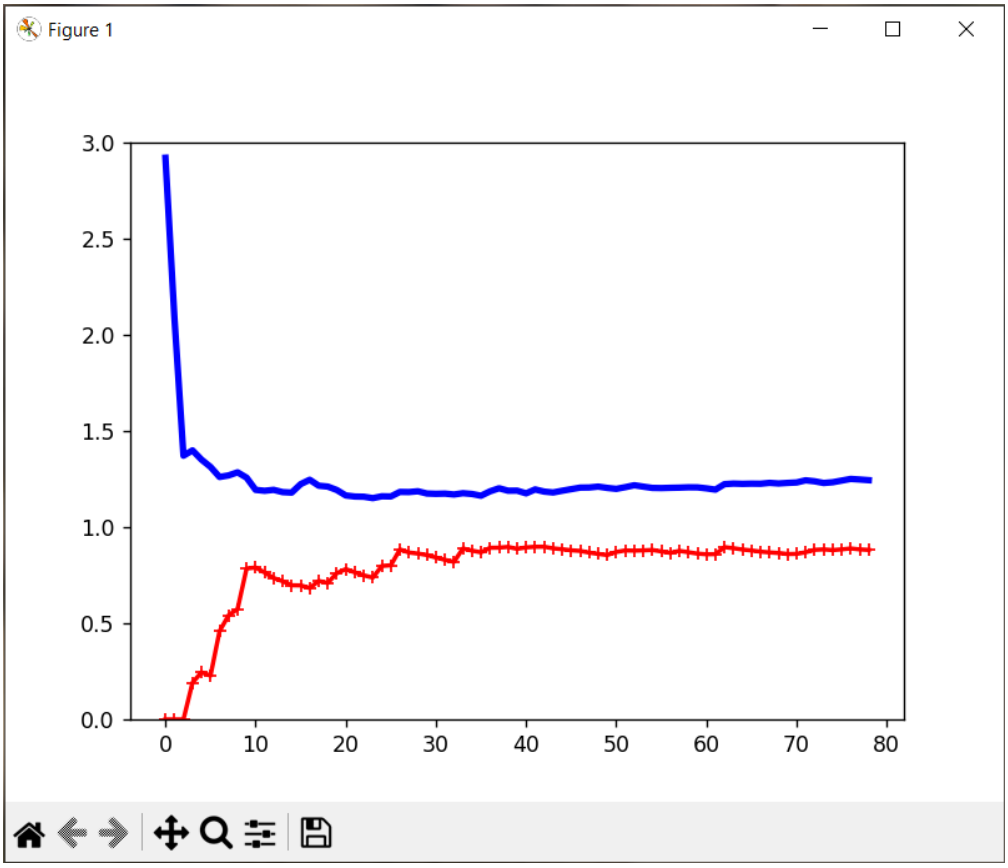


Рис.3.6.3. Криві навчання для поліноміальної моделі 2-го ступеня.

Завдання №3.7. Кластеризація даних за допомогою методу k-середніх.

Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data_clustering.txt.

Лістинг файлу LR_3_task_7.py:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')
num_clusters = 5

# Включення вхідних даних до графіка
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('Input data')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

# Створення об'єкту KMeans
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
# Навчання моделі кластеризації KMeans
kmeans.fit(X)
# Визначення кроку сітки
step_size = 0.01

# Відображення точок сітки
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))

# Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

# Графічне відображення областей та виділення їх кольором
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')

# Графічне відображення областей та виділення їх кольором
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none',
            edgecolors='black', s=80)

# Відображення центрів кластерів
cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1],
            marker='o', s=210, linewidths=4, color='black',
            zorder=12, facecolors='black')
```

```
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('Cluster boundaries')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

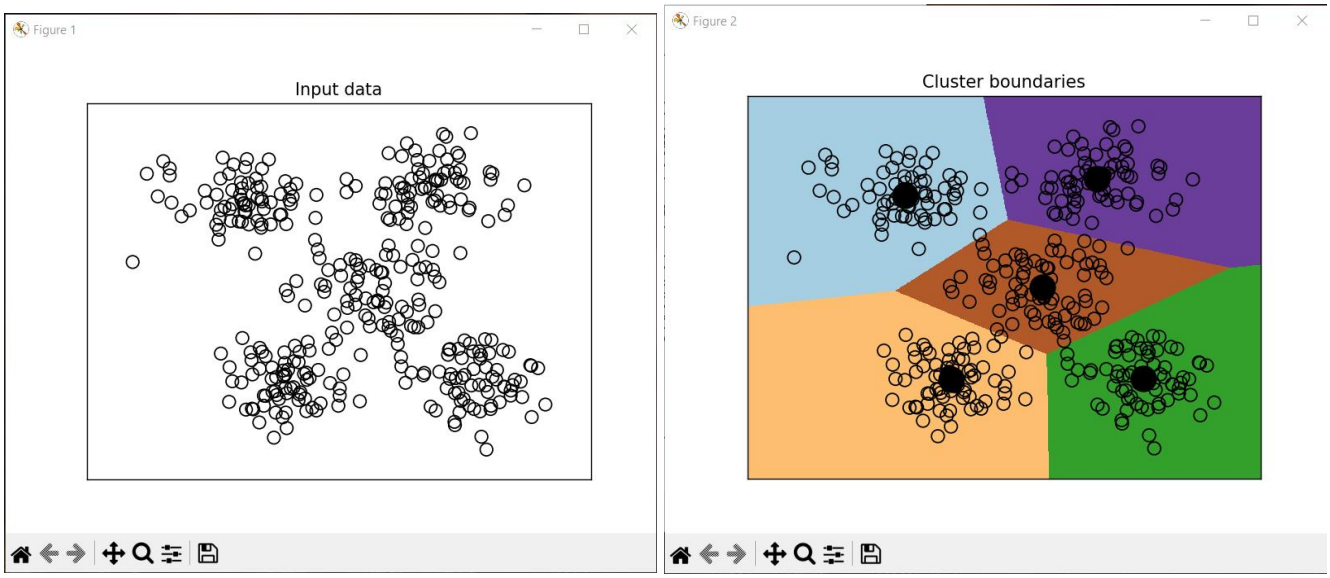


Рис.3.7.1 – 3.7.2. Результат виконання коду.

Метод k-середніх використовується для кластеризації даних на основі схожості об’єктів, коли кількість кластерів заздалегідь відома.

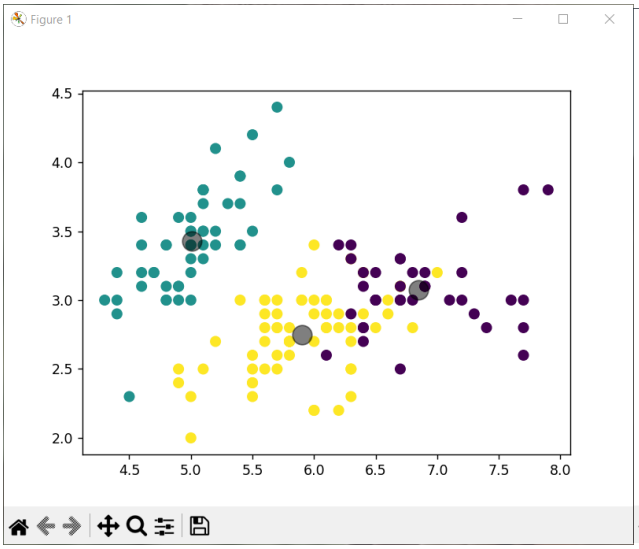
Завдання №3.8. Кластеризація K-середніх для набору даних Iris.

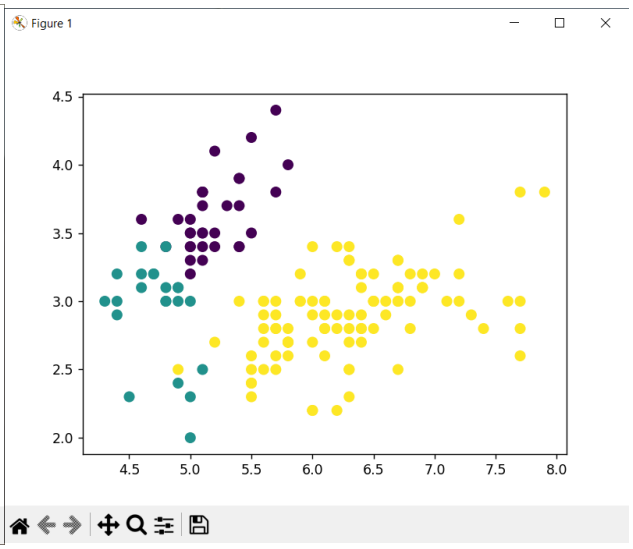
Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів набору даних Iris.

Лістинг файлу LR_3_task_8.py:

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

iris = load_iris()
X = iris['data']
y = iris['target']
```

```
# Створення об'єкту KMeans, навчання і передбачення вихідних міток
kmeans = KMeans(init='k-means++', n_clusters=y.max() + 1, n_init=10, max_iter=300,
tol=0.0001,
                verbose=0, random_state=None, copy_x=True)
y_kmeans = kmeans.fit_predict(X)

# Графічне відображення вхідних точок і центрів кластеризації
plt.figure()
plt.scatter(X[:, 0], X[:, 1], s=50, c=y_kmeans, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()


# Функція для знаходження кластерів
def find_clusters(X_, n_clusters, rseed=2):
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X_.shape[0])[:n_clusters]
    centers_ = X_[i]

    while True:
        labels_ = pairwise_distances_argmin(X_, centers_)
        new_centers = np.array([X_[labels_ == i].mean(0) for i in
range(n_clusters)])
        if np.all(centers_ == new_centers):
            break
        centers_ = new_centers

    return centers_, labels_


centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

labels = KMeans(3, random_state=0, n_init=10).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```





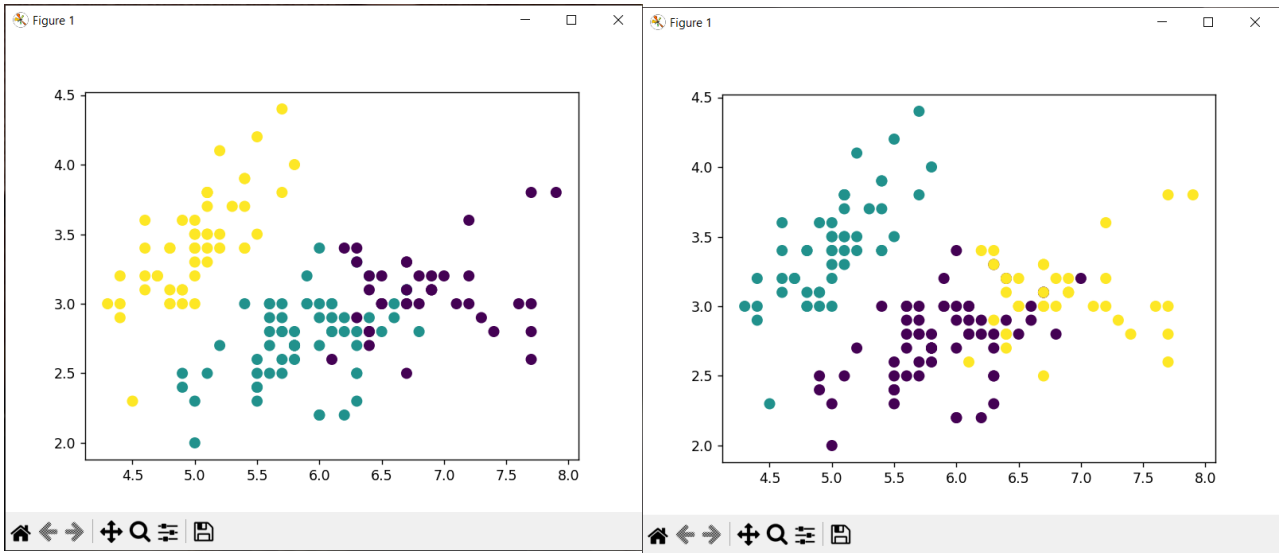


Рис.3.8.1 – 3.8.4. Результат виконання коду.

Було продемонстровано різні способи використання методу KMeans для кластеризації даних. Кожен з варіантів кластеризації показав у результаті поділ на 3 кластери.

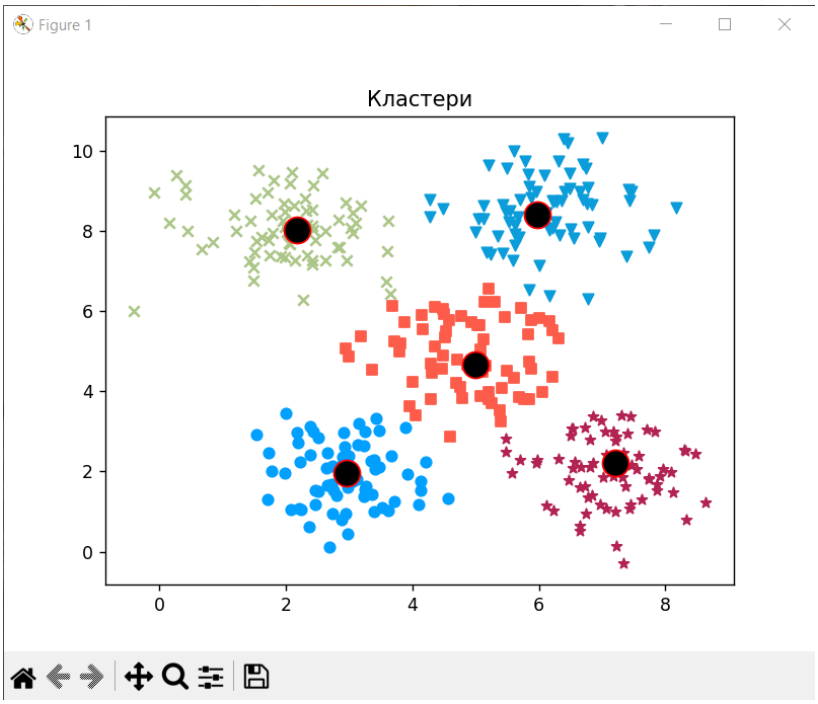
Завдання №3.9. Оцінка кількості кластерів з використанням методу зсуву середнього.

Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

Лістинг файлу LR_3_task_9.py:

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

# Завантаження
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
```

```
print("\nNumber of clusters in input data =", num_clusters)

# Відображення на графіку точок та центрів кластерів
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    # Відображення на графіку точок, що належать поточному кластеру
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,
color=np.random.rand(3,))

    # Відображення на графіку центру кластера
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
markerfacecolor='black', markeredgecolor='red',
             markersize=15)

plt.title('Кластери')
plt.show()
```




```
D:\course-4\semester-1\ai\lab3_project

Centers of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Number of clusters in input data = 5

Process finished with exit code 0
```

Рис.3.9.1 – 3.9.2. Результат виконання коду.

У результаті використання методу зсуву середнього вхідні дані було поділено на 5 кластерів та було знайдено їхні центри.

Завдання №3.10. Оцінка кількості кластерів з використанням методу зсуву середнього.

Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

Лістинг файлу LR_3_task_10.py:

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

# Вхідний файл із символічними позначеннями компаній
input_file = 'company_symbol_mapping.json'

# Завантаження прив'язок символів компаній до їх повних назв
with open(input_file, 'r') as f:
    company_symbols_map = json.loads(f.read())

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

# Завантаження архівних даних котирувань
start_date = datetime.datetime(2003, 7, 3)
end_date = datetime.datetime(2007, 5, 4)
quotes = []
for symbol in symbols:
    quote = yf.Ticker(symbol).history(start=start_date, end=end_date)
    quotes.append(quote)

# Вилучення котирувань, що відповідають
# відкриттю та закриттю біржі
opening_quotes = np.array([quote['Open'].values for quote in
quotes]).astype(float)
closing_quotes = np.array([quote['Close'].values for quote in
quotes]).astype(float)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes

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

# Обчислення різниці між двома видами котирувань
edge_model = covariance.GraphicalLassoCV()

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

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

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

```
D:\course-4\semester-1\ai\lab3_project\venv\Scripts\python.exe D:\course-4\semester-1\ai\lab3_project\LR_3_task_10.py
Cluster 1 ==> Exxon, Chevron, ConocoPhillips, Valero Energy
Cluster 2 ==> Toyota, Ford, Honda, Boeing, Mc Donalds, Apple, SAP, Caterpillar
Cluster 3 ==> Kraft Foods
Cluster 4 ==> Coca Cola, Pepsi, Kellogg, Procter Gamble, Colgate-Palmolive, Kimberly-Clark
Cluster 5 ==> Time Warner, Comcast, Marriott, Wells Fargo, JPMorgan Chase, AIG, American express, Bank of America, Goldman Sachs, Xerox, Wal-Mart, Home Depot, Ryder, DuPont de Nemours
Cluster 6 ==> Microsoft, IBM, HP, Amazon, 3M, General Electrics, Cisco, Texas instruments
Cluster 7 ==> Northrop Grumman, Lookheed Martin, General Dynamics
Cluster 8 ==> GlaxoSmithKline, Pfizer, Sanofi-Aventis, Novartis
Cluster 9 ==> Walgreen, CVS

Process finished with exit code 0
```

Рис.3.10.1. Результат виконання коду.

Висновки: в ході виконання лабораторної роботи було досліджено методи регресії та неконтрольованої класифікації даних у машинному навчанні, використовуючи спеціалізовані бібліотеки та мову програмування Python.