ЛАБОРАТОРНА РОБОТА № 3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ

Мета роботи: використовуючи спеціалізовані бібліотеки і мову програмування Python дослідити методи регресії та неконтрольованої класифікації даних у машинному навчанні.
GitHub репозиторій: **https://github.com/SMTH666/OAI-Lab**

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

Завдання 2.1. Створення регресора однієї змінної

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

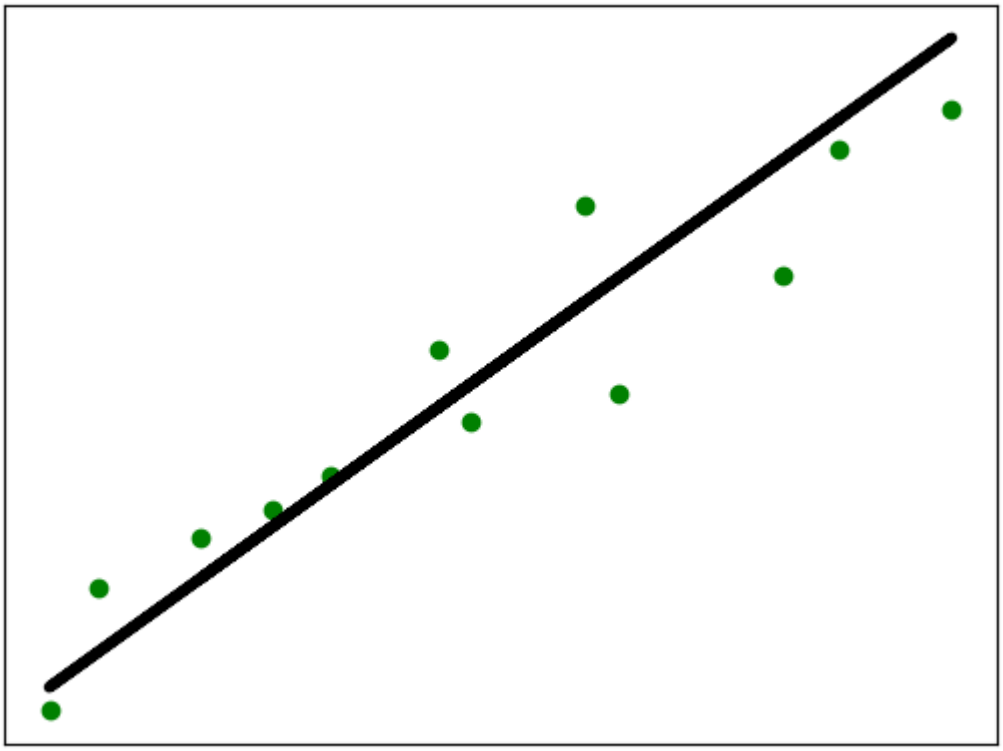
Лістинг програми:

```
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 = 'data_singlevar_regr.txt'

# Завантаження даних
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)

# 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'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
y_test_pred_new = regressor_model.predict(X_test)
print("\nNew mean absolute error =",
      round(sm.mean_absolute_error(y_test, y_test_pred_new), 2))
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\main.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

Process finished with exit code 0
```



Регресійний аналіз за допомогою лінійної регресії

Висновок: Модель лінійної регресії була навчена на 80% даних та протестована на залишкових 20%. У цьому тесті були отримані наступні результати: середня абсолютна помилка - 0.59, середня квадратична різниця - 0.49, медіана абсолютних помилок - 0.51, оцінка поясненої дисперсії - 0.86. Після того, як модель була збережена та відновлена з використанням збереженого файлу, нова абсолютна помилка також склала 0.59. Це свідчить про те, що збережена та відновлена модель надають ті ж самі результати, що й оригінал.

Завдання 2.2. 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 (таблиця 2.1).

Таблиця 2.1

№ за списком	21	22	23	24	25	26	27	28	29	30
№ варіанту	1	2	3	4	5	1	2	3	4	5

Варіант 3 файл: data_regr_3.txt

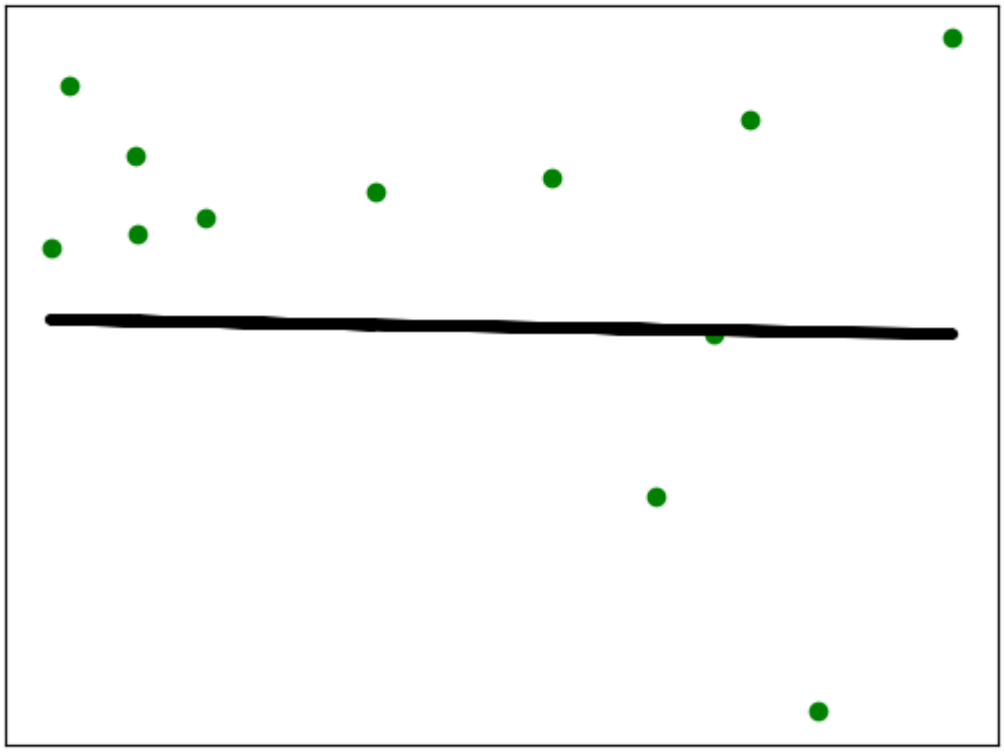
Лістинг програми:

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

input_file = 'data_regr_3.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)

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))

output_model_file = 'model2.pkl'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

with open(output_model_file, 'rb') as f:
```

```
    regressor_model = pickle.load(f)
y_test_pred_new = regressor_model.predict(X_test)
print("\nNew mean absolute error =",
      round(sm.mean_absolute_error(y_test, y_test_pred_new), 2))
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\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

New mean absolute error = 3.59

Process finished with exit code 0
```



Висновок: Модель лінійної регресії була навчена на 80% доступних даних, а потім перевірена на залишкових 20%. У цьому тесті були отримані наступні метрики: середня абсолютна помилка дорівнює 3.59, середня квадратична різниця становить 17.39 (вище, ніж попередня, вказуючи на значні відхилення у прогнозах). Медіана абсолютних помилок складає 3.39, оцінка поясненої дисперсії дорівнює 0.02 (вказуючи на те, що прогнози значно відрізняються від фактичних значень). Після того, як модель була збережена та відновлена з використанням збереженого файлу, нова абсолютна помилка становить 3.59. Це свідчить про те, що збережена і відновлена модель дає ті ж самі результати, що й оригінал.

Завдання 2.3. Створення багатовимірного регресора

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

Лістинг програми:

```
from math import degrees
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
from joblib import dump, load

input_file = '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:]

linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

y_test_pred = linear_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))

# Збереження моделі лінійної регресії
dump(linear_regressor, 'linear_regressor_model.joblib')

# Відновлення моделі лінійної регресії
loaded_linear_regressor = load('linear_regressor_model.joblib')

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)

# Збереження та відновлення моделі поліноміальної регресії
dump(poly_linear_model, 'poly_linear_model.joblib')
loaded_poly_linear_model = load('poly_linear_model.joblib')

print("\nLinear regression:\n", loaded_linear_regressor.predict(datapoint))
```

```
print("\nPolynomial regression:\n", loaded_poly_linear_model.predict(poly_data-
point))
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\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:
 [36.05286276]

Polynomial regression:
 [41.45561819]

Process finished with exit code 0
```

Висновок: Середнє абсолютне відхилення складає 3.58, середньоквадратична різниця - 20.31 (це значення вище, ніж попереднє, що вказує на значні розбіжності у прогнозах). Медіана абсолютних помилок становить 2.99, оцінка поясненої дисперсії - 0.86 (прогнози схожі на фактичні значення). Для поліноміальної регресії використовується поліном 10-го ступеня, і для вхідного значення [7.75, 6.35, 5.56] поліноміальна регресійна модель передбачає значення 41.46, в той час як лінійна регресія передбачає значення 36.05. Обидві моделі мають схожі показники, що свідчить про те, що вони надають приблизно однакові результати.

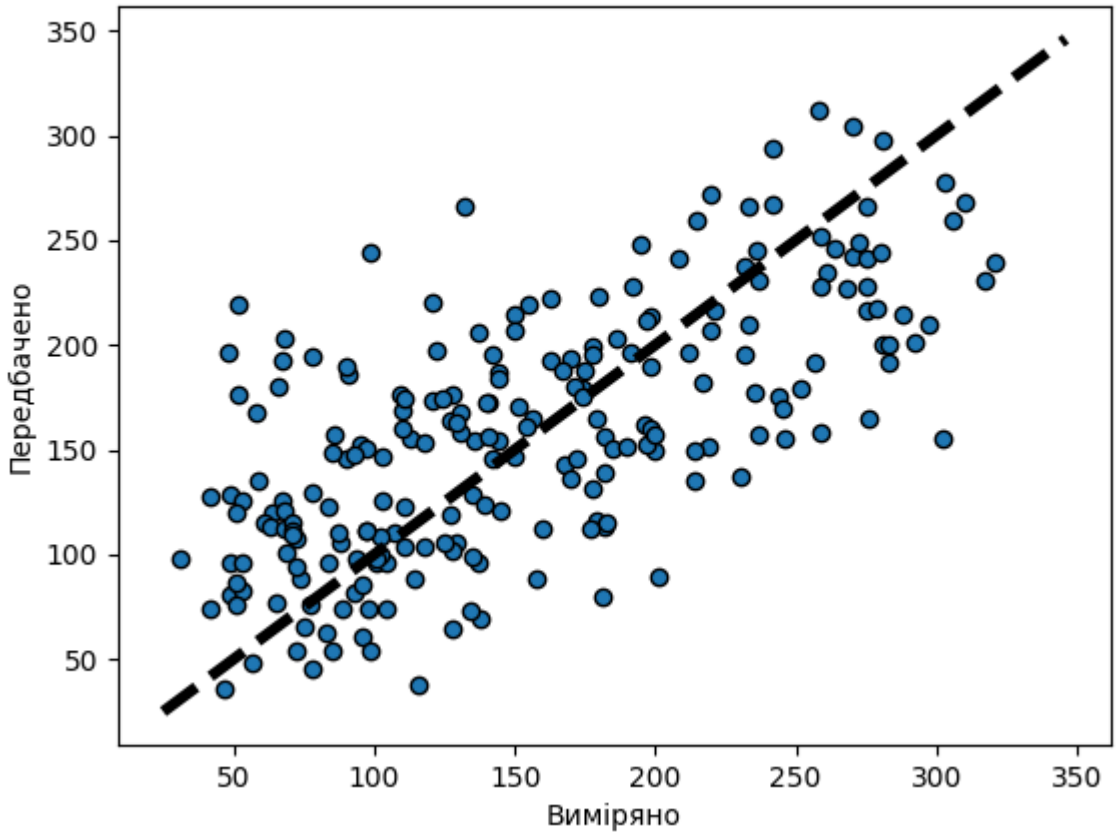
Завдання 2.4. Регресія багатьох змінних

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, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

# Масштабування ознак для поліпшення роботи лінійної регресії
scaler = StandardScaler()
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, ran-
dom_state=0)

regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
y_pred = regr.predict(X_test)

print("Linear regressor performance:")
print("Coefficients:", regr.coef_)
print("Intercept:", regr.intercept_)
print("R2 score:", round(r2_score(y_test, y_pred), 2))
print("Mean absolute error:", round(mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error:", round(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()
```

Результат виконання:

```
un    LR_3_task_4 ×

C:\Users\Worker\PycharmProjects\Lab03\venv\Scripts\python.exe C:\Users\Worker\PycharmProjects\Lab03\LR_3_task_4.py
Linear regressor performance:
regr.coef_ = [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
regr.intercept_ = 154.3589285280134
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33

Process finished with exit code 0
```



Висновок: Було виконано лінійну регресію для набору даних «Diabetes» та було отримано такі результати якості:**Коефіцієнти регресії** представлються як масив чисел,вони вказують на вагу кожної ознаки -0.97055556 -12.64686835 26.85770273 15.48541795 -32.92275106 18.81473774 1.11761899 5.53487216 40.14242421 0.60496027],**Перетин** рівний 154.36 і представляє відсоток,на який зміщується пряма регресії,**Оцінка R2** дорівнює 0.44(модель пояснює близькьо 44% варіації в цільовій змінній),Середня Абсолютна різниця становить 44.8,Середня

Квадратична Оцінка дорівнює 3075.33(показує різницю між прогнозованим і фактичним значеннями).Модель має обмежену ефективність,що підтверджує значення

Завдання 2.5. Самостійна побудова регресії

Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) 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.

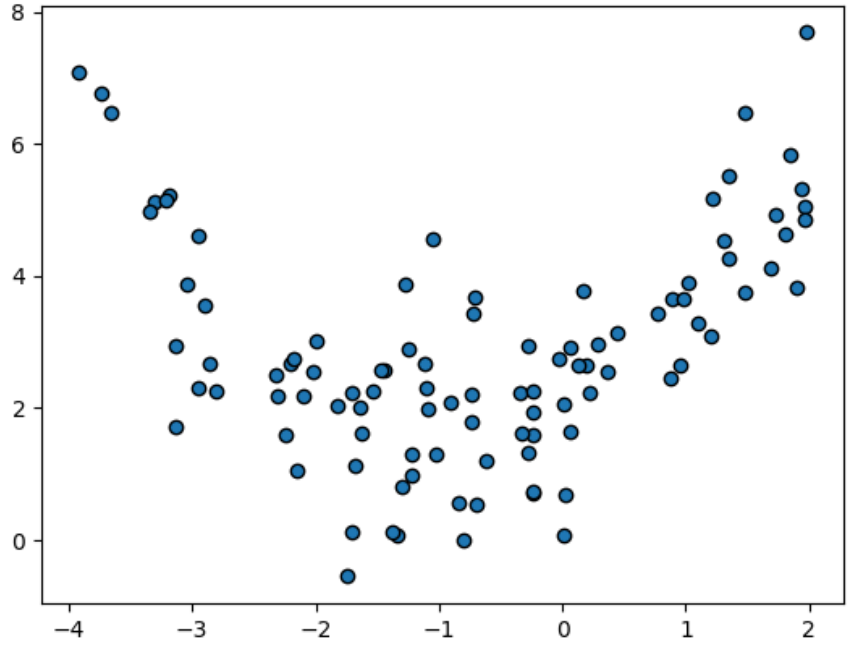
Таблиця 2.2

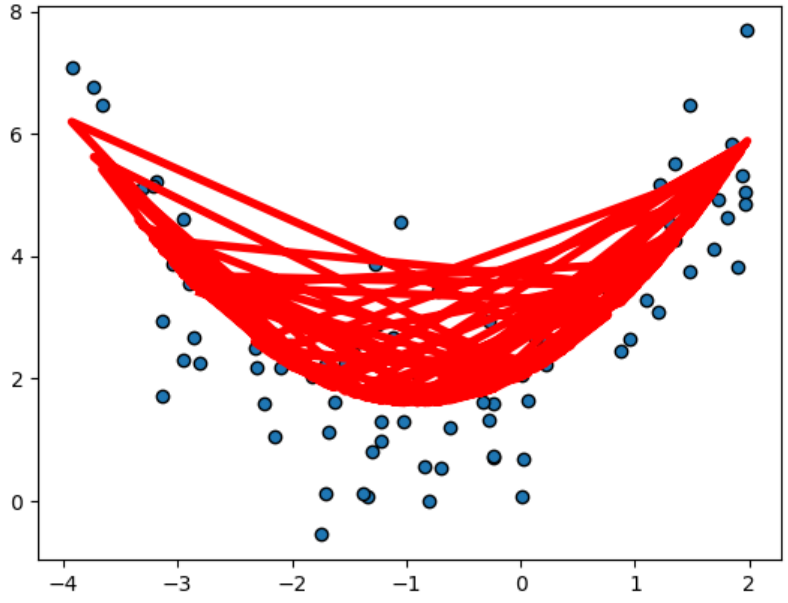
Лістинг програми:

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

m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)

fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors=(0, 0, 0))
plt.show()

# Трансформація ознак за допомогою поліноміальних ознак
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)

print("X[0] =", X[0])
print("X[1] =", X[1])
print("Y[1] =", y[1])

# Навчання лінійної регресії на поліноміальних ознаках
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)

print("Перетин:", lin_reg.intercept_)
print("Коефіцієнти регресії:", lin_reg.coef_)

# Передбачення на основі навченої моделі
y_pred = lin_reg.predict(X_poly)

fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors=(0, 0, 0))
plt.plot(X, y_pred, color='red', linewidth=4)
plt.show()
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\LR_3_task_5.py
X[0] = [-2.80620389]
X[1] = [-0.24139816]
Y[1] = [1.9314117]
Перетин: [2.03076045]
Коефіцієнти регресії: [[0.93474719 0.50841172]]

Process finished with exit code 0
```





Графіки моделі

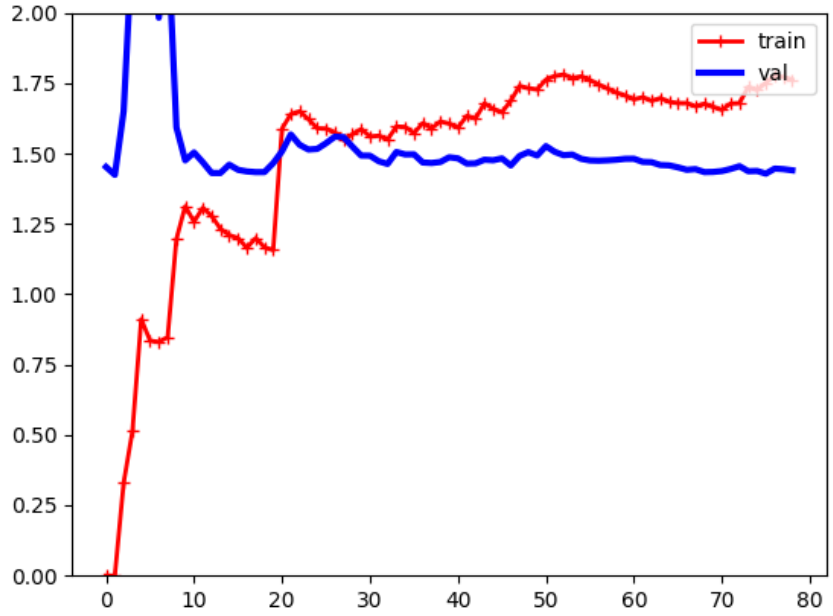
Завдання 2.6. Побудова кривих навчання

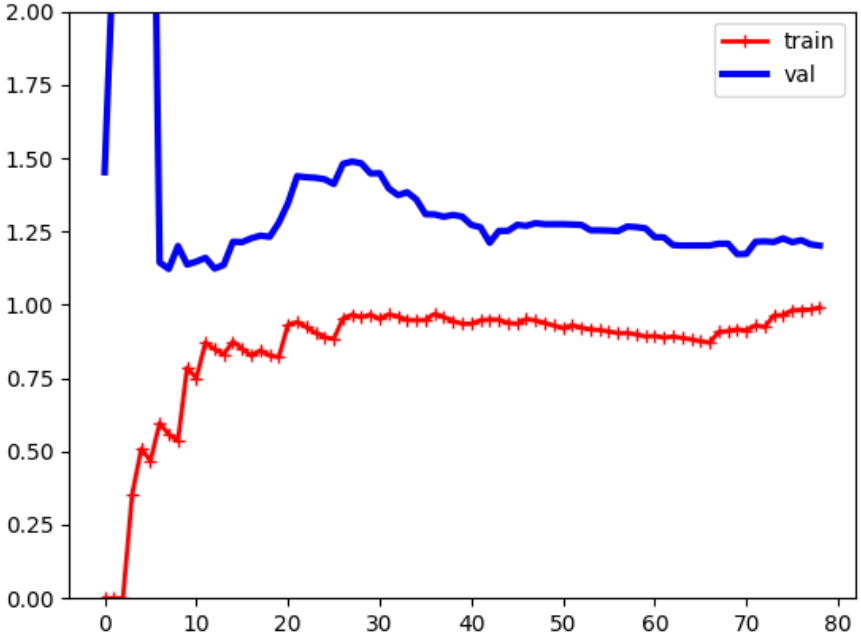
Побудуйте криві навчання для ваших даних у попередньому завданні.

Лістинг програми:

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

def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, ran-
dom_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_predict, y_train[:m]))
        val_errors.append(mean_squared_error(y_val_predict, y_val))
    fig, ax = plt.subplots()
    ax.plot(np.sqrt(train_errors), "r-+", linewidth=2, label='train')
    ax.plot(np.sqrt(val_errors), "b-", linewidth=3, label='val')
    plt.ylim(0, 2)
    plt.legend(loc='upper right')
    plt.show()

m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)


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

polynomial_regression = Pipeline([
    ('poly_features', PolynomialFeatures(degree=2, include_bias=False)),
    ('lin_reg', linear_model.LinearRegression()),
])

plot_learning_curves(polynomial_regression, X, y)
```

Результат виконання:


train
val
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
60
70
80


train
val
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40
50
60
70
80

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

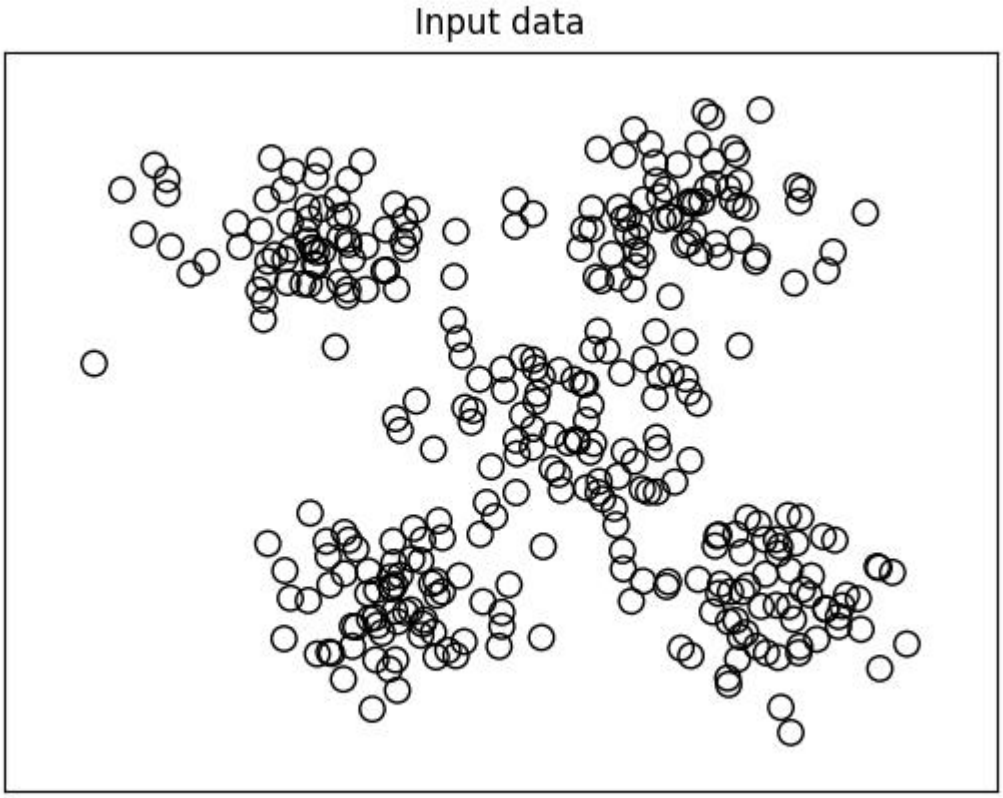
Завдання 2.7. Кластеризація даних за допомогою методу k-середніх

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

Лістинг програми:

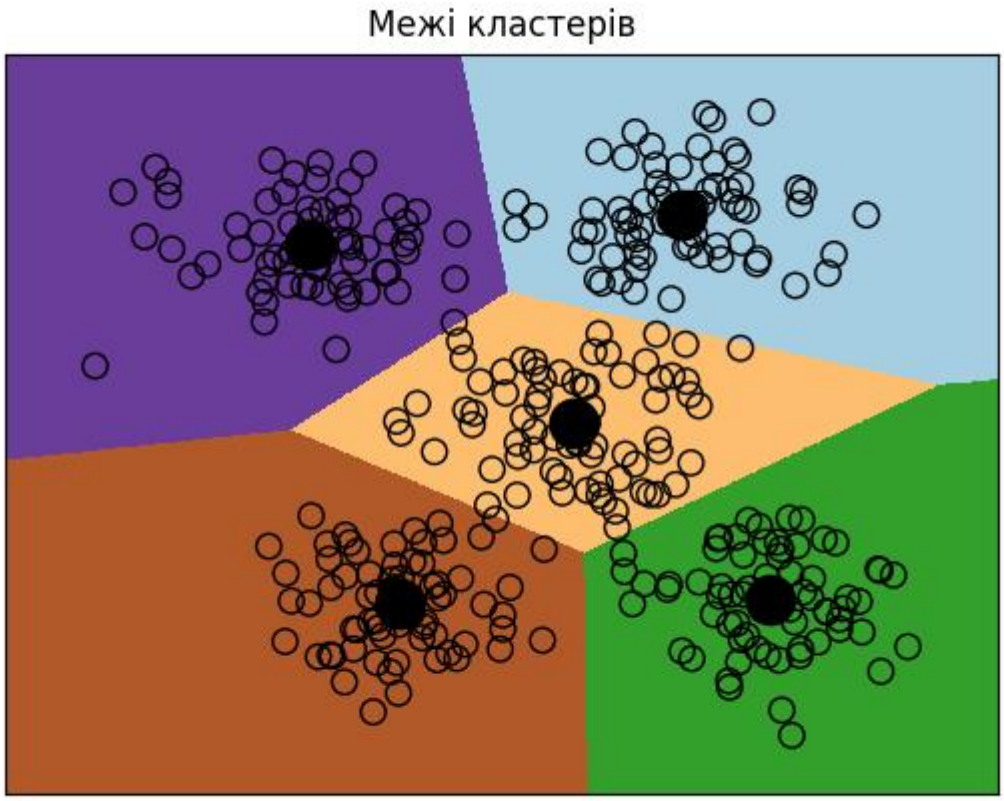
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics

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

# Кількість кластерів
num_clusters = 5

# Візуалізація вхідних даних
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(())
plt.show()
```

Результат виконання:




```
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_
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('Межі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
```

```
plt.yticks(())
plt.show()
```



Висновок: Я використав агоритм K-Means для кластеризації даних, було використано 5 кластерів, на графіку показано розділення простору на кластери та центри кластерів.

Завдання 2.8. 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.

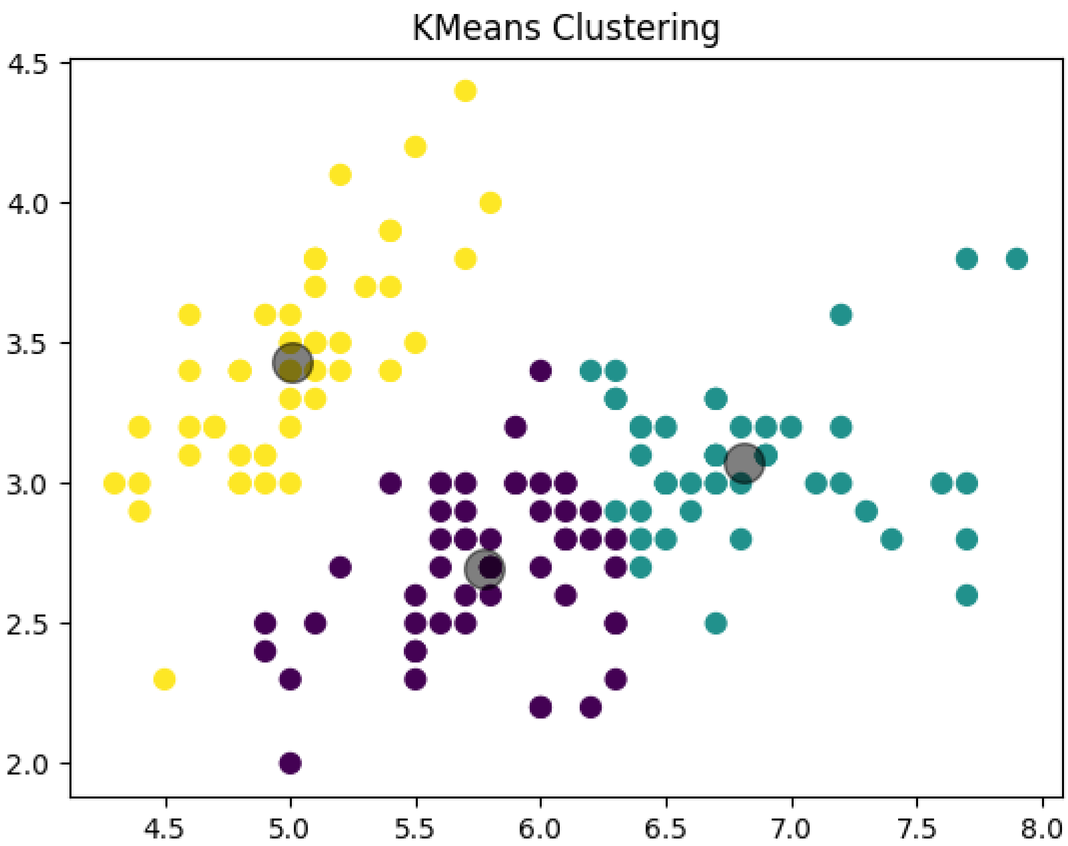
Лістинг програми:

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

iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target

# KMeans з параметрами
kmeans = KMeans(n_clusters=Y.max() + 1, init='k-means++', n_init=10, max_iter=300,
                tol=0.0001, verbose=0, random_state=None, copy_x=True)
kmeans.fit(X)
y_pred = kmeans.predict(X)

print("n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, ran-
dom_state: None, copy_x: True")
print(y_pred)

# Візуалізація результатів KMeans
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.title('KMeans Clustering')
plt.show()


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_clus-
ters)])
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels

# Застосування власної функції для кластеризації
print("using find_clusters():")
centers, labels = find_clusters(X, 3)
print("n_clusters: 3, rseed: 2")
```

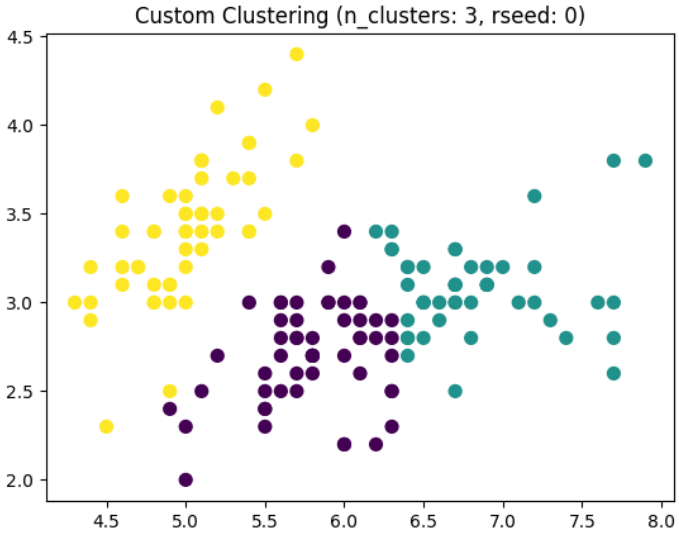
```
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title('Custom Clustering (n_clusters: 3, rseed: 2)')
plt.show()

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

# Використання KMeans з вказаною кількістю кластерів та random_state
labels = KMeans(3, random_state=0).fit_predict(X)
print("using KMeans():")
print("n_clusters: 3, random_state: 0")
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title('KMeans Clustering (n_clusters: 3, random_state: 0)')
plt.show()
```

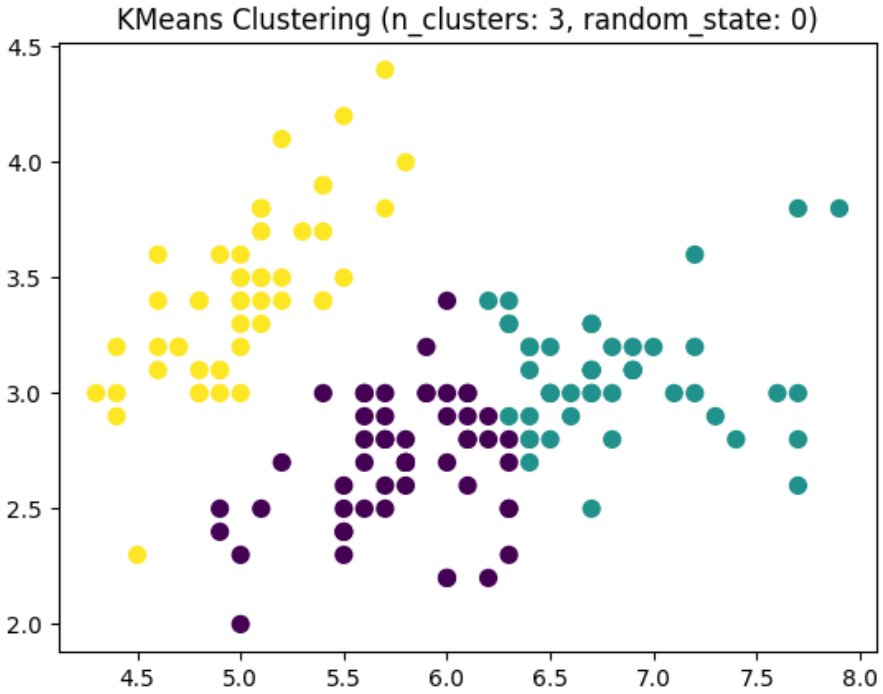
Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\LR_3_task_8.py
n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, random_state: None, copy_x: True
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1
 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1
 1 0]
using find_clusters():
n_clusters: 3, rseed: 2
n_clusters: 3, rseed: 0
C:\Users\toxa1\PycharmProjects\lab03\venv\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppres
  super()._check_params_vs_input(X, default_n_init=10)
using KMeans():
n_clusters: 3, random_state: 0

Process finished with exit code 0
```




Custom Clustering (n_clusters: 3, rseed: 2)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Custom Clustering (n_clusters: 3, rseed: 0)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0



Висновок: K-Means алгоритм може бути використаний для кластеризації даних, і ви можете вибирати різні параметри та початкові умови для отримання різних результатів кластеризації. Точна кількість кластерів та початкові умови можуть бути важливими для правильного розділення даних на кластери

Завдання 2.9. 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.

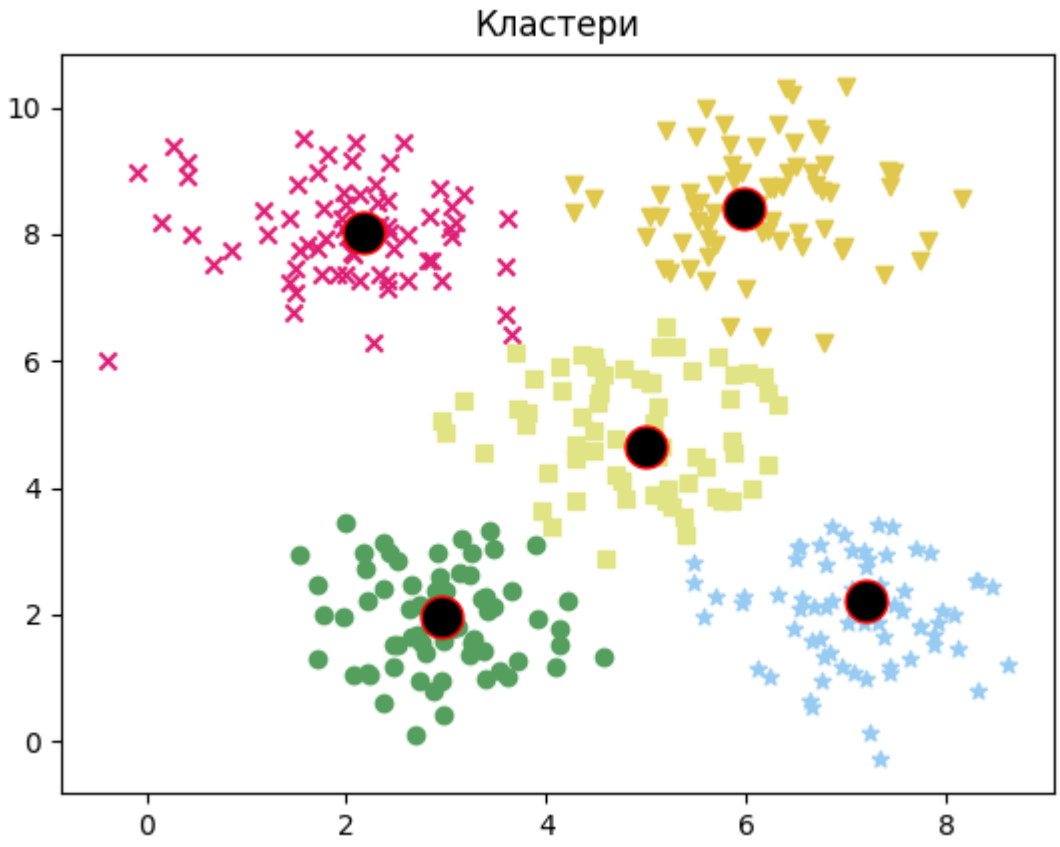
Лістинг програми:

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

# Завантаження даних
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), cycle(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()
```

Результат виконання:

```
C:\Users\toxa1\PycharmProjects\lab03\venv\Scripts\python.exe C:\Users\toxa1\PycharmProjects\lab03\LR_3_task_9.py

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
```



Висновок: Алгоритм Mean Shift успішно використовується для кластеризації даних. В результаті було отримано п'ять кластерів, і їх центри були виведені на екран та відображені на графіку. Кластери виділені різними кольорами для легкості візуалізації та аналізу структури даних.

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

Неможливо виконати завдання через відсутність файлу company_symbol_mapping.json.

Висновок до лабораторної роботи: використовуючи спеціалізовані бібліотеки і мову програмування Python дослідив методи регресії та неконтрольованої класифікації даних у машинному навчанні.