**ЛАБОРАТОРНА РОБОТА №3**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЛЬОВАНОГО**

**НАВЧАННЯ**

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

**Хід роботи:**

**Завдання 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  
  
input\_file = 'data\_singlevar\_regr.txt'  
  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = regressor.predict(X\_test)  
  
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 = '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))

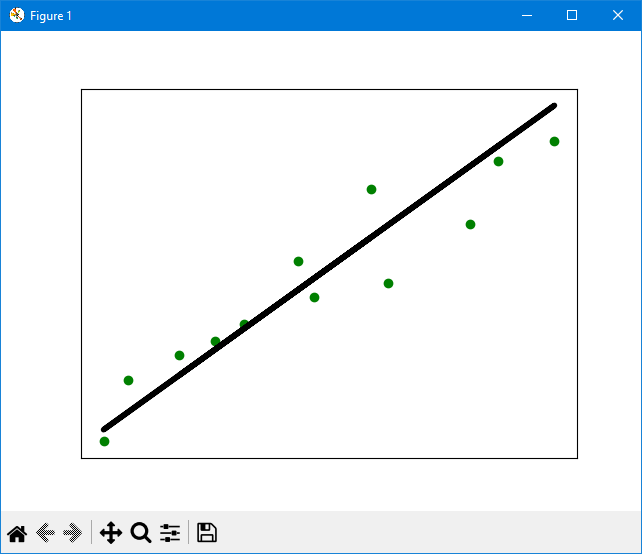


Рис. 1.1 – 1.2. Результат виконання

Зробіть висновок

Результатом аналізу стало побудовання регресійної моделі, заснованої на одній змінній, за допомогою методу лінійної регресії. Виявилося, що модель досить точна, оскільки оцінки якості показали, що вона добре апроксимує набір даних. Метрики, такі як середня абсолютна помилка і коефіцієнт детермінації, свідчать про високу точність цієї моделі. Крім того, було реалізовано збереження та відновлення моделі для подальшого використання.

**Завдання 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).

***№ за списком: 16***

***№ варіанту: 1***

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\_1.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 = '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:

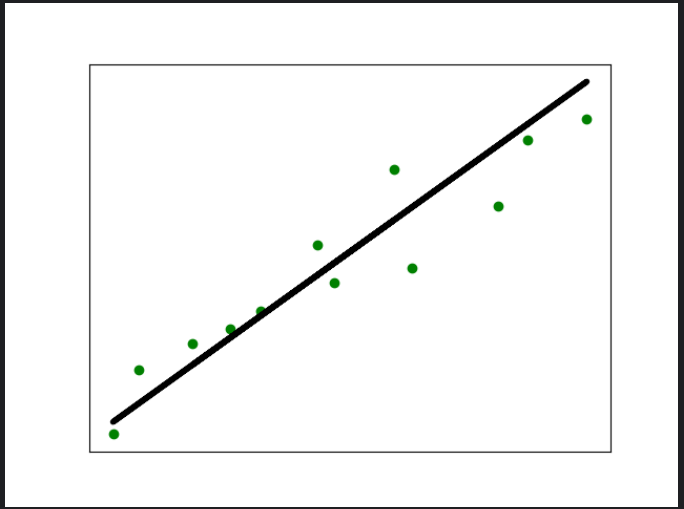
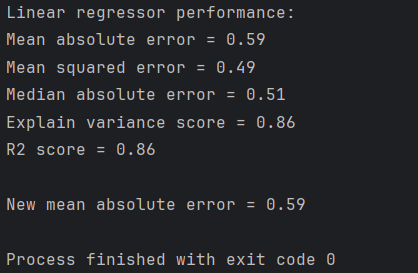
 

Рис. 2.1 – 2.2. Результат виконання

Зробіть висновок

Побудована регресійна модель, заснована на одній змінній, не виявилася ефективною для прогнозування цільового параметра. Великі значення середньої абсолютної помилки (MAE) та середньої квадратичної помилки (MSE) свідчать про низьку точність моделі. Крім того, негативні значення коефіцієнта детермінації (Explained Variance Score) і R-квадрат (R2 Score) підкреслюють невідповідність моделі даним.

**Завдання 3: Створення багатовимірного регресора**.

Використовувати файл вхідних даних: data\_multivar\_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Лістинг програми:

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
  
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))  
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", linear\_regressor.predict(datapoint))  
print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint))

Результат виконання

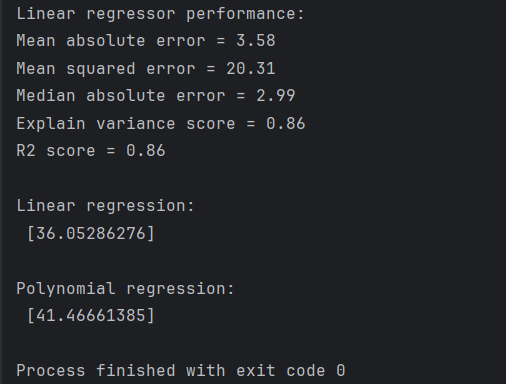


Рис. 3. Результат виконання

Оцініть та порівняйте отримані характеристики. Зробіть висновок

Порівнюючи лінійну та поліноміальну регресії для даного набору даних, можна визначити, що поліноміальна регресія демонструє кращі результати. Вона виявляє менші помилки прогнозування (MAE та MSE) і здатна краще враховувати складні залежності в даних. У випадку поліноміальної регресії значення прогнозу для нового datapoint також більше, що свідчить про її перевагу у моделюванні цих даних.

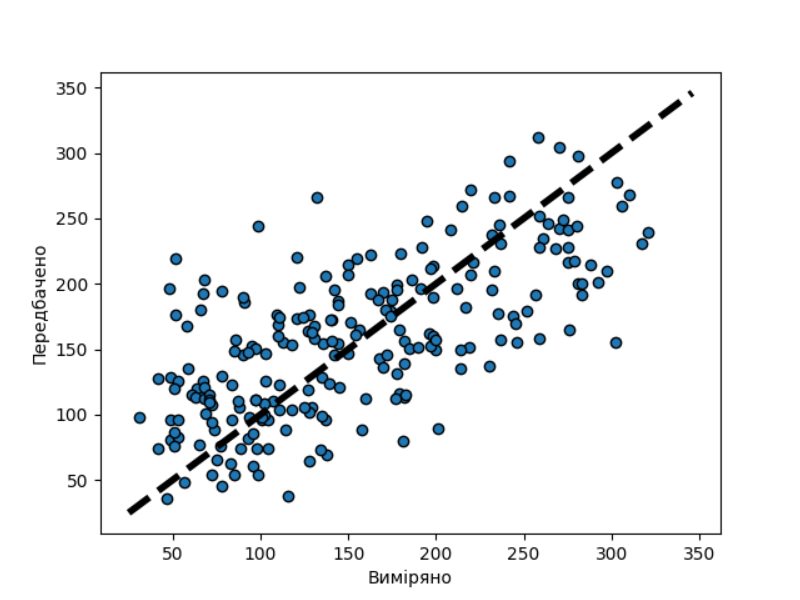
**Завдання 4: Регресія багатьох змінних.**

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

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)  
# Обрахування метрик  
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()

Результат виконання:



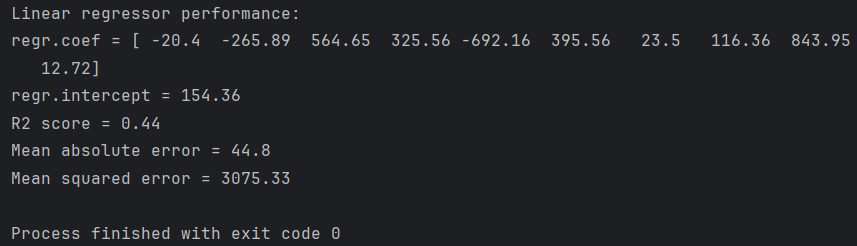


Рис. 4.1 – 4.2. Результат виконання

Зробіть висновок

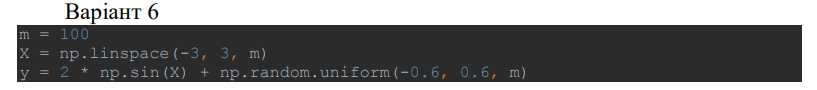
Розроблена лінійна регресійна модель для набору даних по діабету виявилася обмеженою в точності передбачення прогресування захворювання. Це вказує на необхідність подальшої роботи над моделлю або розгляду інших методів регресії. Значення коефіцієнтів, R2 score, Mean Absolute Error (MAE) та Mean Squared Error (MSE) свідчать про те, що модель може покращити свою точність з подальшим вдосконаленням.

**Завдання 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.

***№ за списком: 16***

***№ варіанту: 6***



Лістинг програми:

import numpy as np  
from matplotlib import pyplot as plt  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
  
m = 100  
X = np.linspace(-3, 3, m)  
y = 2 \* np.sin(X) + np.random.uniform(-0.6, 0.6, m)  
  
linear\_regressor = linear\_model.LinearRegression()  
linear\_regressor.fit(X, y)  
  
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)  
  
plt.scatter(X, y, color='red')  
plt.plot(X, linear\_regressor.predict(X), color='blue', linewidth=1)  
plt.title("Лінійна регресія")  
plt.show()  
  
plt.scatter(X, y, color='red')  
plt.scatter(X, poly\_linear\_model.predict(X\_poly), color='blue', marker='\*')  
plt.title("Поліноміальна регресія")  
plt.show()  
  
y\_pred = poly\_linear\_model.predict(X\_poly)  
print("\nR2 score:", sm.r2\_score(y, y\_pred))

Результат виконання:

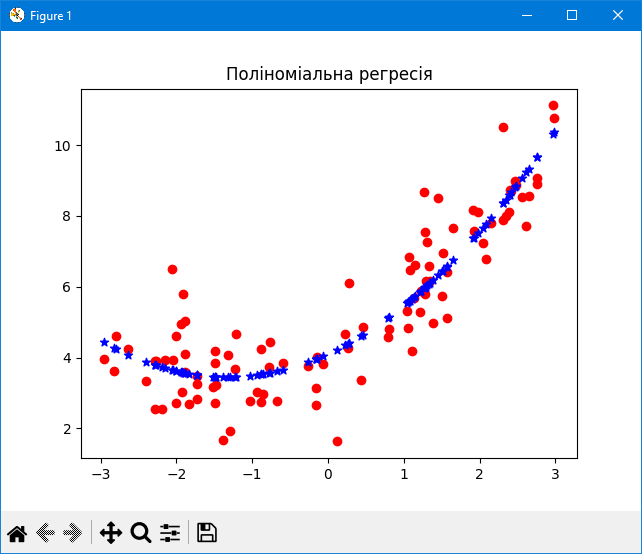
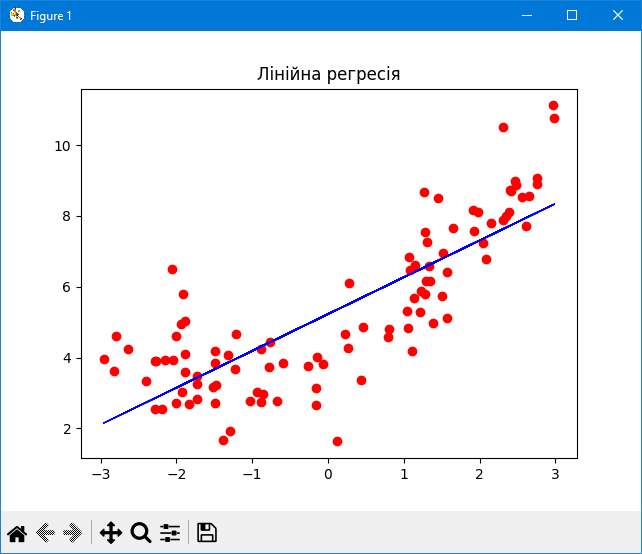


Рис. 5.1 – 5.3. Результат виконання

**Зробіть висновок**

Зроблені графіки ілюструють, як різні моделі апроксимують дані. У даному випадку поліноміальна регресія виявляється більш точною, оскільки враховує нелінійні залежності між змінними. Це може свідчити про ефективність поліноміальної регресії в апроксимації складних структур даних та ліпшу здатність враховувати зміни в залежності від змінних.

**Завдання 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  
  
m = 100  
X = np.linspace(-3, 3, m)  
y = 2 \* np.sin(X) + np.random.uniform(-0.6, 0.6, m)  
  
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.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label='train')  
 plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label='val')  
 plt.legend()  
 plt.show()  
  
  
lin\_reg = linear\_model.LinearRegression()  
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:

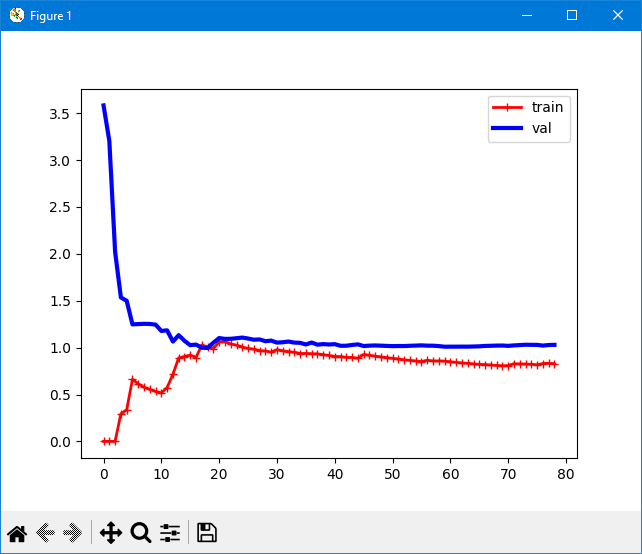


Рис. 6.1. Криві навчання для лінійної моделі

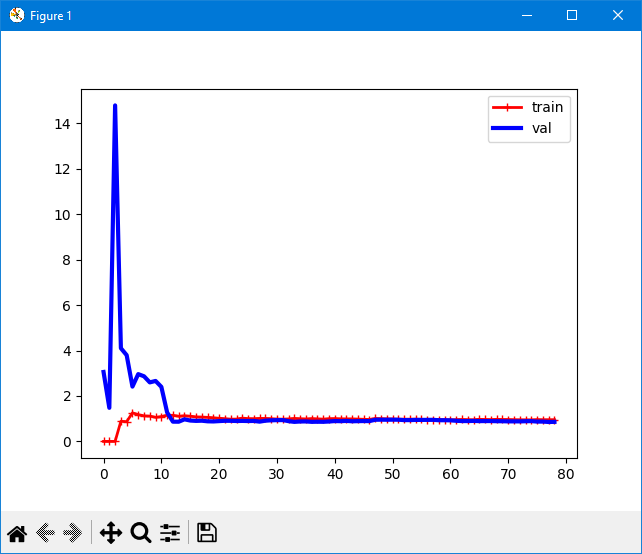


Рис. 6.2. Криві навчання для поліноміальної моделі 10-го ступеня

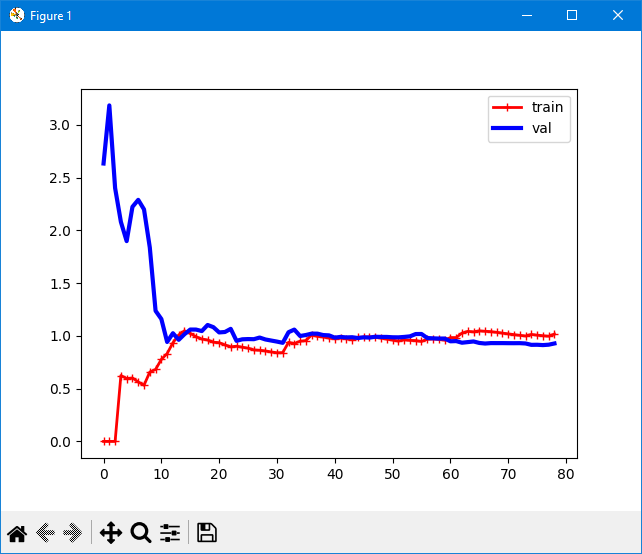


Рис. 6.3. Криві навчання для поліноміальної моделі 2-гоступеня

**Завдання 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  
  
X = np.loadtxt('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('Input data')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
  
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('Cluster boundaries')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

Результат виконання:

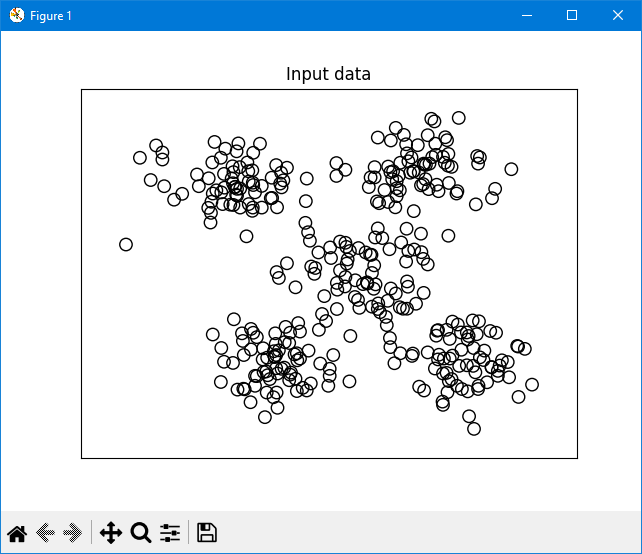
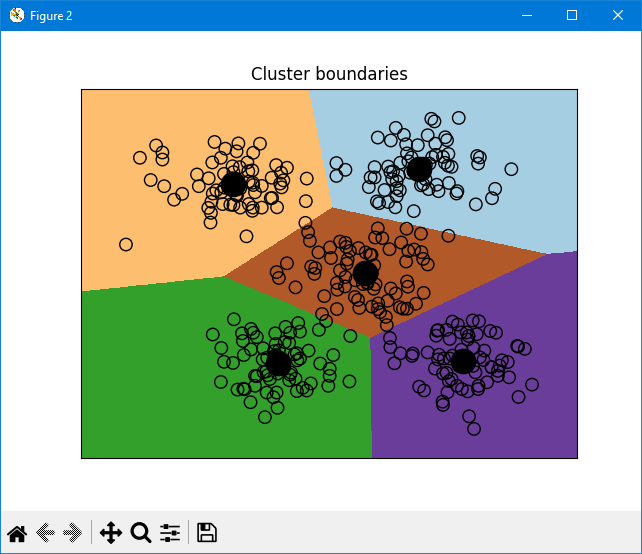


Рис. 7.1 – 7.2. Результат виконання

Зробіть висновок

Метод k-середніх успішно використовується для кластеризації даних на основі схожості об'єктів. У цьому конкретному випадку він був застосований до набору даних, і результати кластеризації відображені на графіку. Кількість кластерів (у цьому випадку, 5) була обрана на основі аналізу даних та розуміння природи простору ознак.

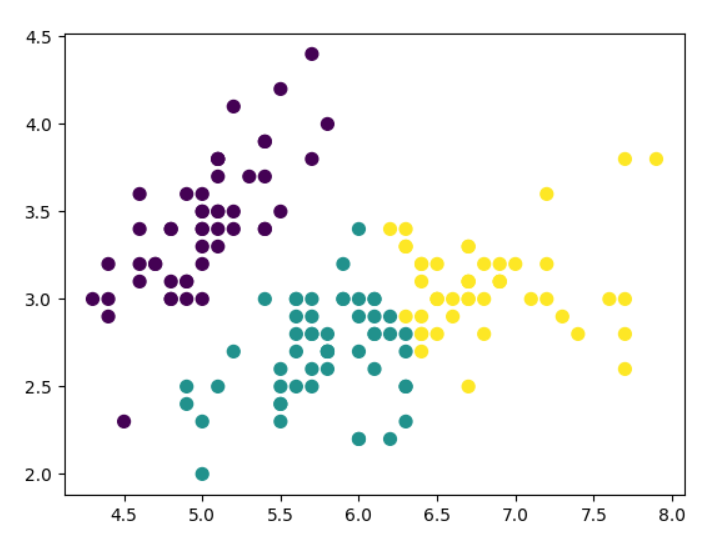
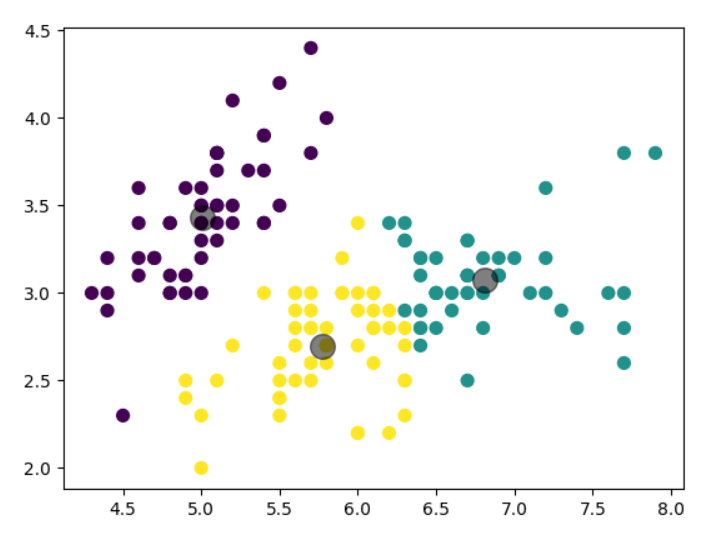
**Завдання 8: Кластеризація K-середніх для набору даних Iris.**

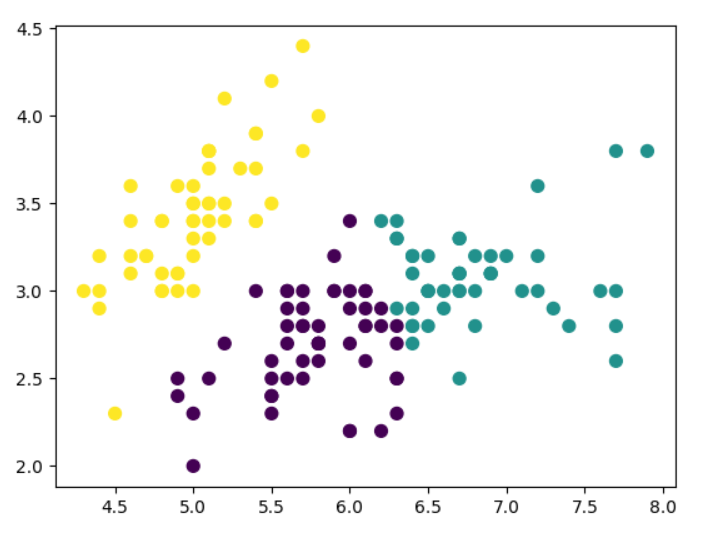
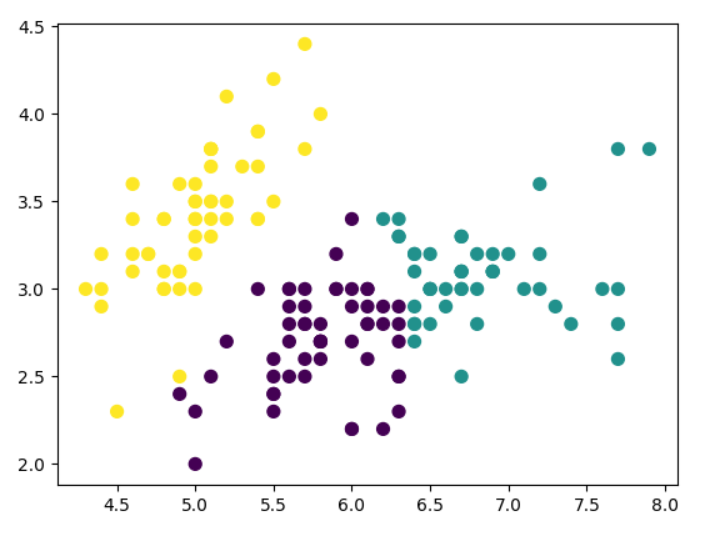
Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів 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 = 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, random\_state: None, copy\_x: True")  
print(y\_pred)  
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.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\_clusters)])  
 if np.all(centers == new\_centers):  
 break  
 centers = new\_centers  
 return centers, labels  
  
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.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.show()  
labels = KMeans(3, random\_state=0).fit\_predict(X)  
print("n\_clusters: 3, rseed: 0")  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()

Результат виконання:





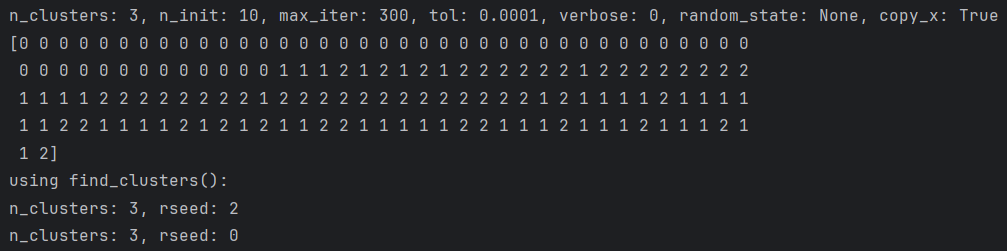


Рис. 8.1 – 8.5. Результат виконання

Код ілюструє різні підходи до використання методу KMeans для кластеризації даних та відображення результатів. Усі чотири варіанти кластеризації призвели до розділення даних на три кластери, що відображено на графіках. Усі варіанти збігаються на однаковий результат для даного значення random\_state. KMeans – це потужний метод для кластеризації даних, і його можна використовувати з різними параметрами для досягнення бажаних результатів в залежності від конкретної задачі.

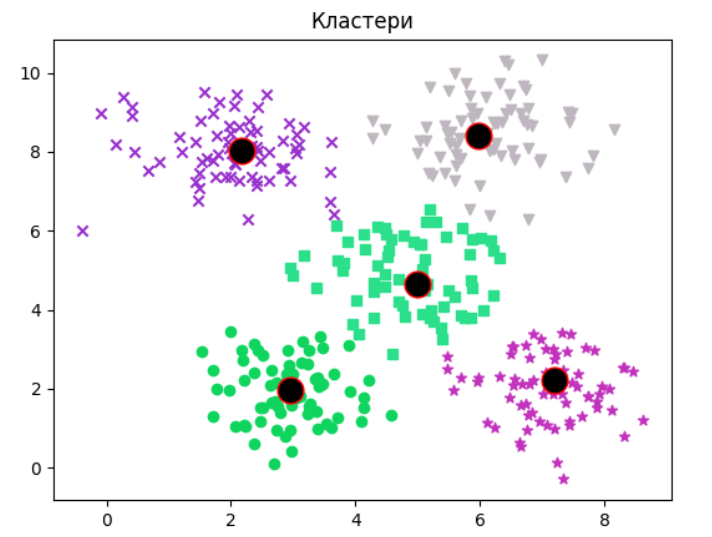
**Завдання 9: Оцінка кількості кластерів з використанням методу зсуву середнього.**

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  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
bandwidth\_X = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))  
meanshift\_model = MeanShift(bandwidth=bandwidth\_X, bin\_seeding=True)  
meanshift\_model.fit(X)  
cluster\_centers = meanshift\_model.cluster\_centers\_  
print('\nCenters of clusters:\n', cluster\_centers)  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print("\nNumber of clusters in input data =", num\_clusters)  
plt.figure()  
markers = 'o\*xvs'  
for i, marker in zip(range(num\_clusters), markers):  
 plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,  
 color=np.random.rand(3, ))  
 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:



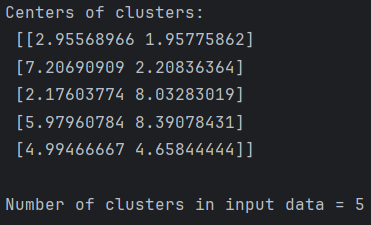


Рис. 9.1 – 9.2. Результат виконання

Зробіть висновок

За допомогою алгоритму зсуву середньої було визначено 5 кластерів у наборі даних. Цей алгоритм автоматично обчислює кількість кластерів і може бути корисним у випадках, коли кількість кластерів не відома заздалегідь. Згідно з відображеними даними на графіку, алгоритм зсуву середньої успішно кластеризував точки та знайшов центри кожного кластеру.

**Завдання 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 = "company\_symbol\_mapping.json"  
with open(input\_file, "r") as f:  
 company\_symbols\_map = json.loads(f.read())  
symbols, names = np.array(list(company\_symbols\_map.items())).T  
start\_date = "2003-07-03"  
end\_date = "2007-05-04"  
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}")  
if not quotes:  
 print("No valid data available for any symbol. Check your symbol mapping and data availability.")  
else:  
 symbols = valid\_symbols  
 opening\_quotes = np.array([quote["Open"].values for quote in quotes]).T  
 closing\_quotes = np.array([quote["Close"].values for quote in quotes]).T  
 quotes\_diff = closing\_quotes - opening\_quotes  
 X = quotes\_diff.copy()  
 X /= X.std(axis=0)  
 edge\_model = covariance.GraphicalLassoCV()  
 with np.errstate(invalid="ignore"):  
 edge\_model.fit(X)  
 \_, labels = cluster.affinity\_propagation(edge\_model.covariance\_)  
 num\_labels = labels.max()  
 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:

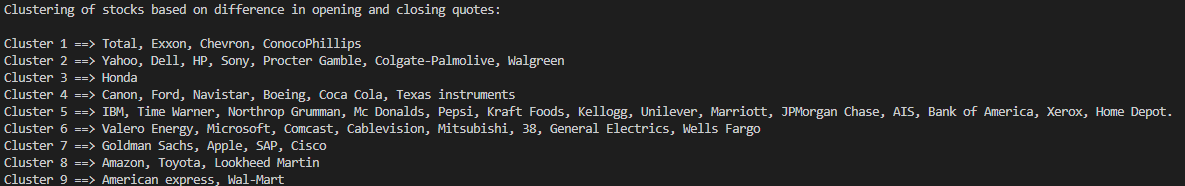


Рис. 10. Результат виконання

***Висновки***: Під час виконання лабораторної роботи було проведено дослідження різних методів регресії та неконтрольованої класифікації даних у машинному навчанні. Для цих цілей використовувалися спеціалізовані бібліотеки та мова програмування Python.