Міністерство освіти і науки України

Державний університет ,,Житомирська політехніка”

Кафедра: Фікт

Група: ІПЗ-20-1

# Лабораторна робота №3

«ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО

НАВЧАННЯ»

Виконав: Денисюк Н. Г.

Прийняв: Голенко М. Ю.

**Мета роботи:** використовуючи спеціалізовані бібліотеки і мову

програмування Python дослідити методи регресії та неконтрольованої

класифікації даних у машинному навчанні.

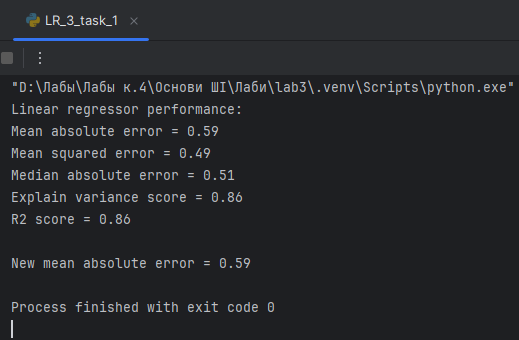
**Виконання роботи:**

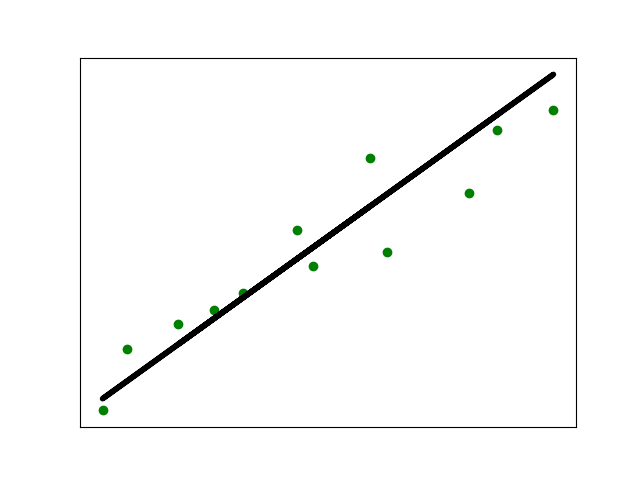
**Завдання №1:** Створення регресора однієї змінної

Код:

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))

**Результат виконння:**





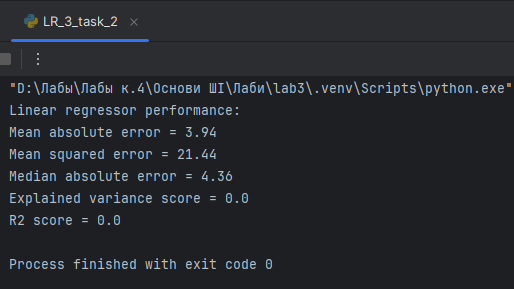
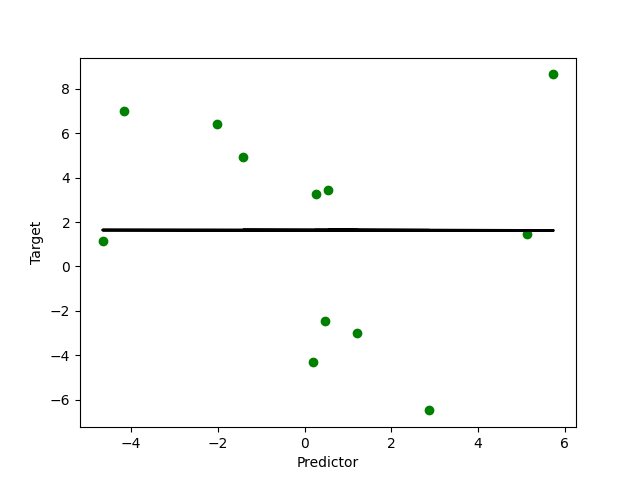
**Завдання №2:** Передбачення за допомогою регресії однієї змінної



Код:

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
  
# Завантаження даних  
data = np.loadtxt('data\_regr\_3.txt', delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбивка даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)  
  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
  
# Навчання моделі  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
  
# Візуалізація результатів  
plt.scatter(X\_test, y\_test, color='green')  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=2)  
plt.xlabel('Predictor')  
plt.ylabel('Target')  
plt.show()  
  
# Показники якості  
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("Explained 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))  
  
# Матриця помилок (Confusion Matrix) не застосовується до задач регресії

**Результат виконання:**

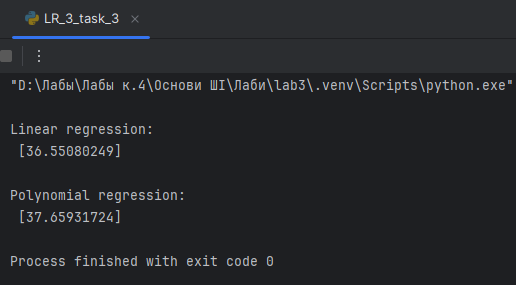
****

**Завдання №3:** Створення багатовимірного регресора

Код:

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  
data = np.loadtxt('data\_multivar\_regr.txt', delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
# Розбивка даних на навчальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)  
  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
  
# Навчання моделі  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
  
  
# 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))

**Результат виконання:**

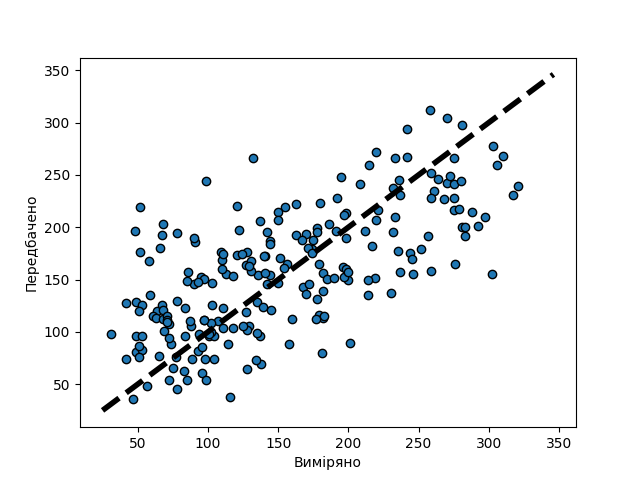


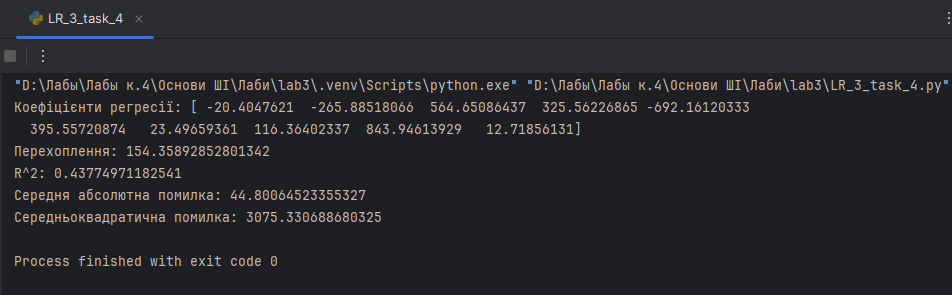
**Завдання №4:** Регресіябагатьох змінних

Код:

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)  
  
# Коефіцієнти регресії  
print("Коефіцієнти регресії:", regr.coef\_)  
# Перехоплення (intercept)  
print("Перехоплення:", regr.intercept\_)  
# R^2 (коефіцієнт детермінації)  
print("R^2:", r2\_score(ytest, ypred))  
# Середня абсолютна помилка  
print("Середня абсолютна помилка:", mean\_absolute\_error(ytest, ypred))  
# Середньоквадратична помилка  
print("Середньоквадратична помилка:", 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:**

****

****

**Завдання №5:** Самостійна побудова регресії

****

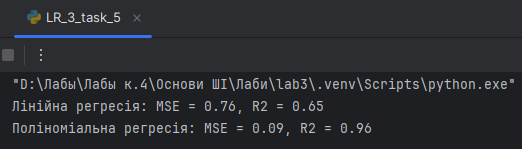
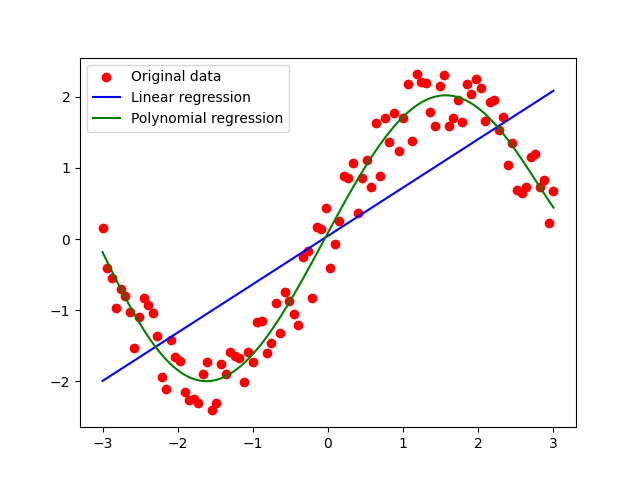
Варіант 8



Код:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.pipeline import make\_pipeline  
  
# Генерація випадкових даних  
m = 100  
X = np.linspace(-3, 3, m).reshape(-1, 1)  
y = 2 \* np.sin(X).flatten() + np.random.uniform(-0.5, 0.5, m)  
  
# Побудова моделі лінійної регресії  
linear\_regressor = LinearRegression()  
linear\_regressor.fit(X, y)  
y\_pred\_linear = linear\_regressor.predict(X)  
  
# Побудова моделі поліноміальної регресії  
degree = 5  
polynomial\_features = PolynomialFeatures(degree=degree)  
polynomial\_regressor = make\_pipeline(polynomial\_features, LinearRegression())  
polynomial\_regressor.fit(X, y)  
y\_pred\_poly = polynomial\_regressor.predict(X)  
  
# Виведення графіків  
plt.scatter(X, y, color='red', label='Original data')  
plt.plot(X, y\_pred\_linear, color='blue', label='Linear regression')  
plt.plot(X, y\_pred\_poly, color='green', label='Polynomial regression')  
plt.legend()  
plt.show()  
  
# Оцінка якості моделей  
mse\_linear = mean\_squared\_error(y, y\_pred\_linear)  
r2\_linear = r2\_score(y, y\_pred\_linear)  
mse\_poly = mean\_squared\_error(y, y\_pred\_poly)  
r2\_poly = r2\_score(y, y\_pred\_poly)  
  
print(f"Лінійна регресія: MSE = {mse\_linear:.2f}, R2 = {r2\_linear:.2f}")  
print(f"Поліноміальна регресія: MSE = {mse\_poly:.2f}, R2 = {r2\_poly:.2f}")

**Результат виконання:**

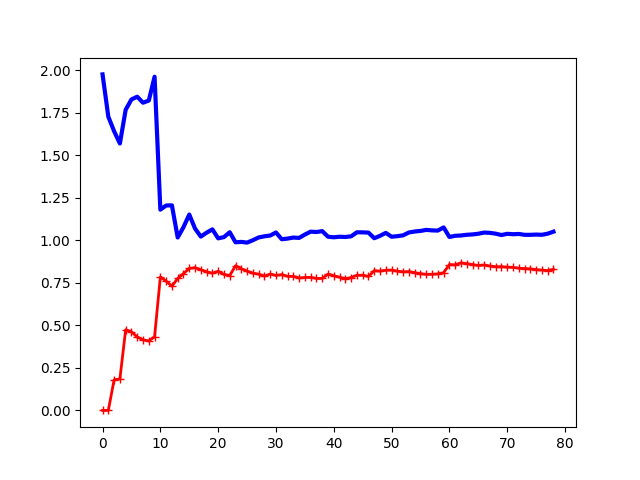
****

**Завдання №6:** Побудова кривих навчання

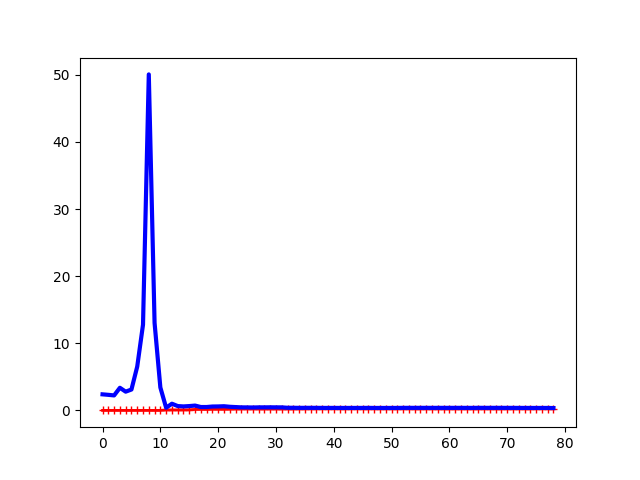
Код:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import learning\_curve  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_squared\_error  
  
# Генерація даних  
m = 100  
X = np.linspace(-3, 3, m).reshape(-1, 1)  
y = 2 \* np.sin(X).flatten() + np.random.uniform(-0.5, 0.5, m)  
  
  
# Функція для побудови кривих навчання  
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")  
  
  
# Побудова моделі поліноміальної регресії  
polynomial\_regressor = Pipeline([("poly\_features",  
 PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", LinearRegression())])  
  
  
polynomial\_regressor\_second = Pipeline([("poly\_features",  
 PolynomialFeatures(degree=2, include\_bias=False)),  
 ("lin\_reg", LinearRegression())])  
  
  
# Візуалізація кривих навчання  
plot\_learning\_curves(LinearRegression(), X, y)  
plt.show()  
plot\_learning\_curves(polynomial\_regressor, X, y)  
plt.show()  
plot\_learning\_curves(polynomial\_regressor\_second, X, y)  
plt.show()

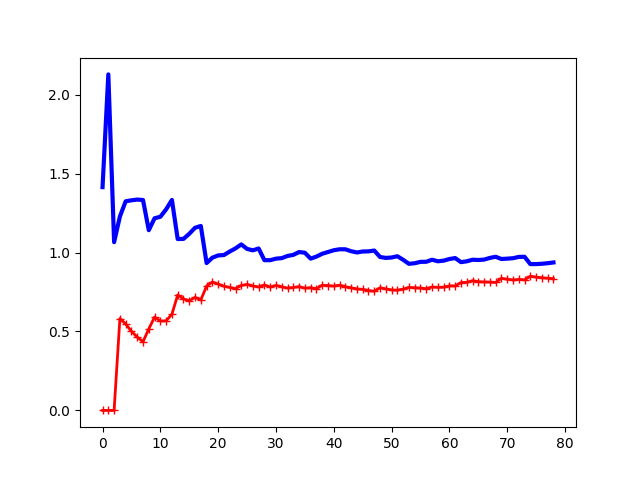
**Результат виконання:**



LinearRegression



polynomial\_regressor



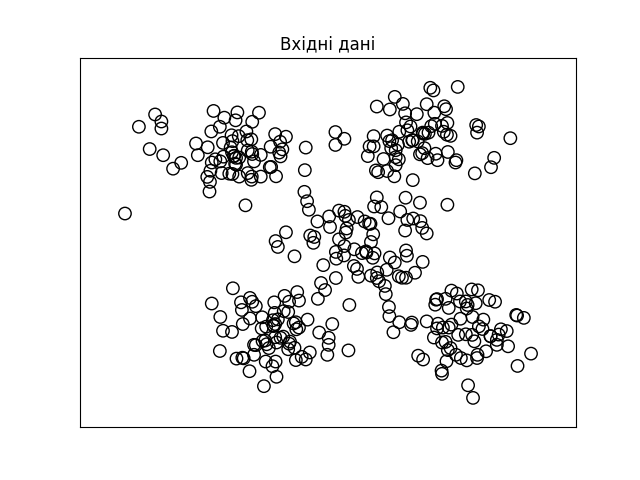
polynomial\_regressor\_second

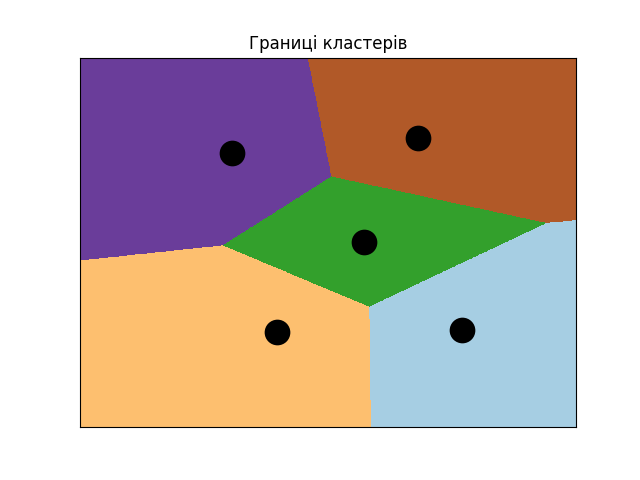
**Завдання №7:** Кластеризація даних за допомогою методу k-середніх

Код:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
  
# Завантаження взідні дані із файлу  
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)  
  
# 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  
  
plt.title('Вхідні дані')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
  
plt.show()  
  
  
# Створення об'єкту KMeans  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
  
# Навчання моделі кластеризації КМеаns  
kmeans.fit(X)  
  
# Визначення кроку сітки  
step\_size = 0.01  
  
# Відображення точок сітки  
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')  
  
# 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')  
  
plt.title('Границі кластерів')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
  
plt.show()

**Результат виконання:**



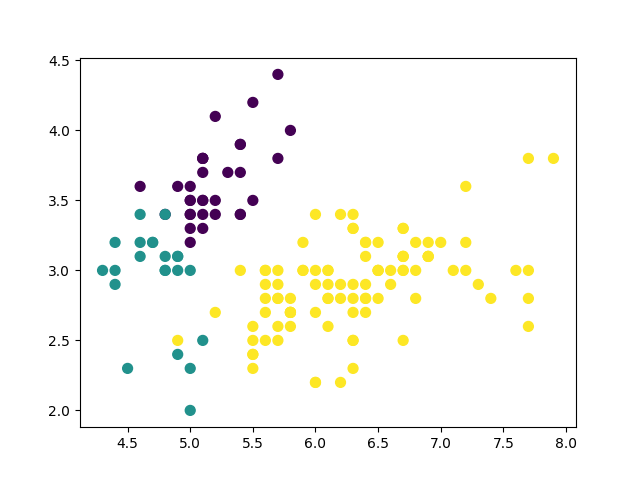
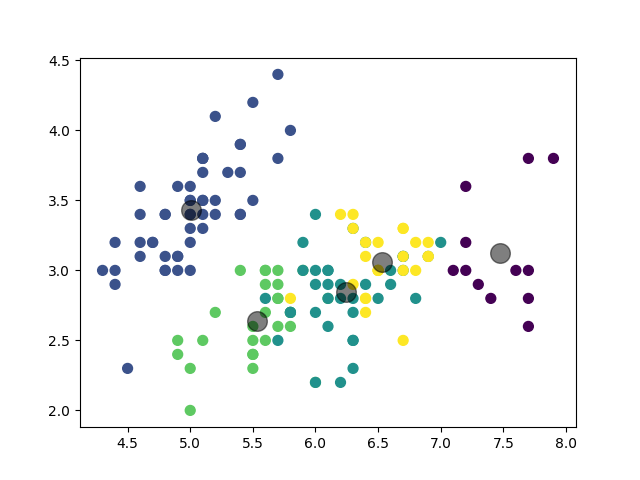


**Завдання №8:** Кластеризація K-середніх для набору даних Iris

Код:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import datasets, cluster  
from sklearn.cluster import KMeans  
from sklearn.metrics import pairwise\_distances\_argmin  
  
# Завантаження даних Iris  
iris = datasets.load\_iris()  
X = iris['data']  
y = iris['target']  
  
# Ініціалізація та навчання KMeans  
cluster.KMeans(n\_clusters=8, init='k-means + +', n\_init=10, max\_iter=300, tol=0.0001,  
 verbose=0, random\_state=None, copy\_x=True, algorithm='auto')  
kmeans = KMeans(n\_clusters=5)  
kmeans.fit(X)  
  
# Прогнозування кластерів  
y\_kmeans = kmeans.predict(X)  
  
# Візуалізація результатів  
plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  
centers = kmeans.cluster\_centers\_  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
plt.show()  
  
# Функція для пошуку 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  
  
# Використання функції для пошуку кластерів  
centers, labels = find\_clusters(X, 3)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()

**Результат виконання:**



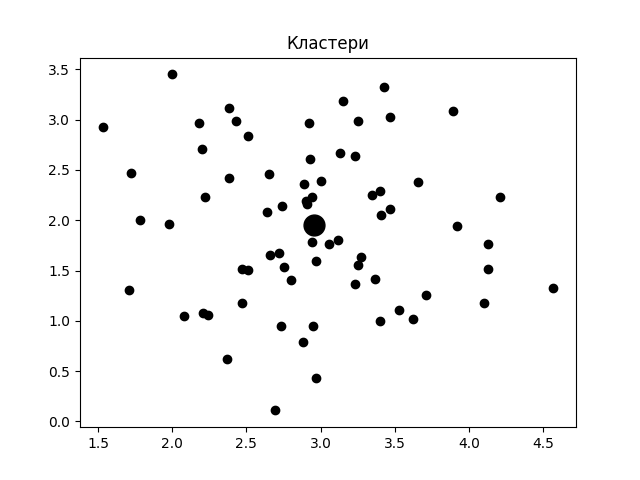
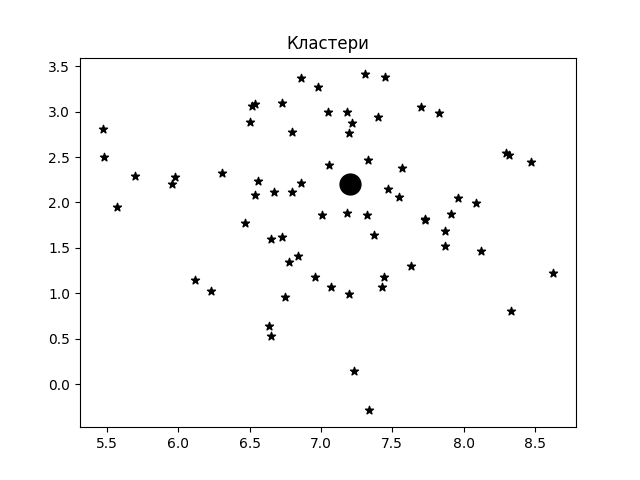
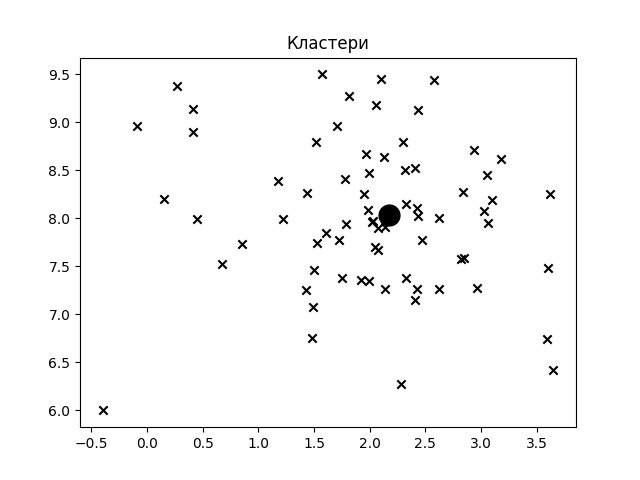
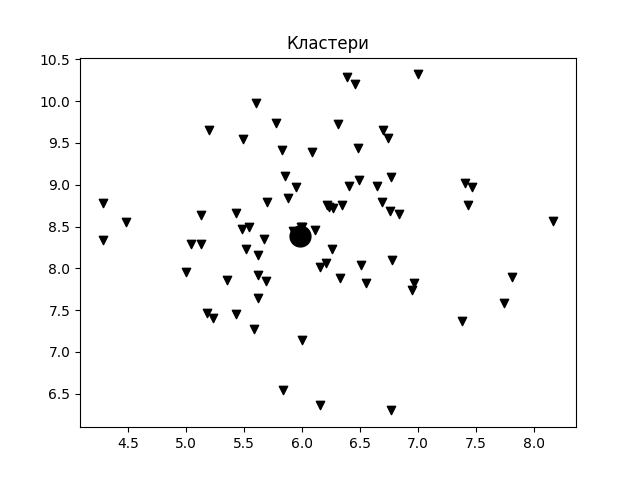
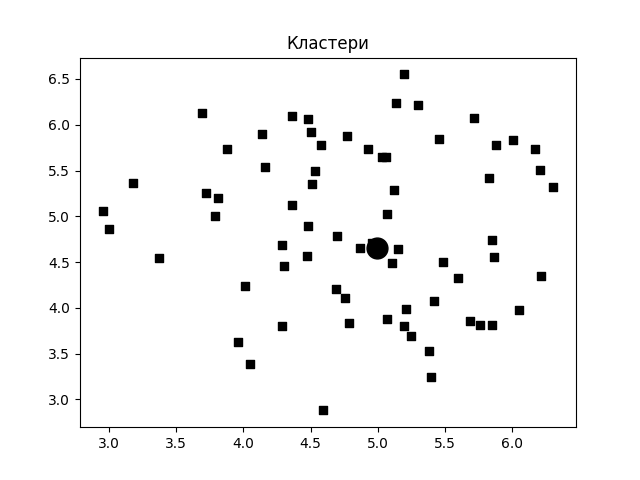
**Завдання №9:** Оцінка кількості кластерів з використанням методу

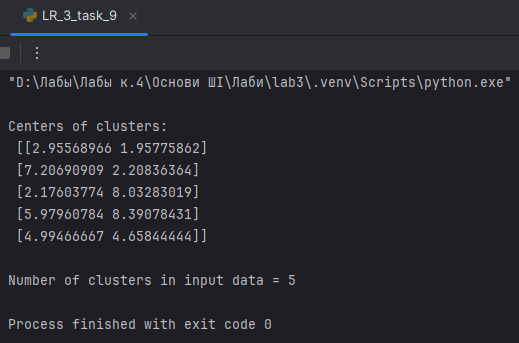
зсуву середнього

Код:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from itertools import cycle  
  
  
# Завантаження  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
# Оцінка ширини вікна для Х  
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 точок котрі належать поточному кластеру  
 plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color='black')  
  
 # Відображення на графіку центру кластера  
 cluster\_center = cluster\_centers[i]  
 plt.plot(cluster\_center[0], cluster\_center[1], marker='o',  
 markerfacecolor='black', markeredgecolor='black', markersize=15)  
 plt.title('Кластери')  
 plt.show()

**Результат виконання:**

**** ****   

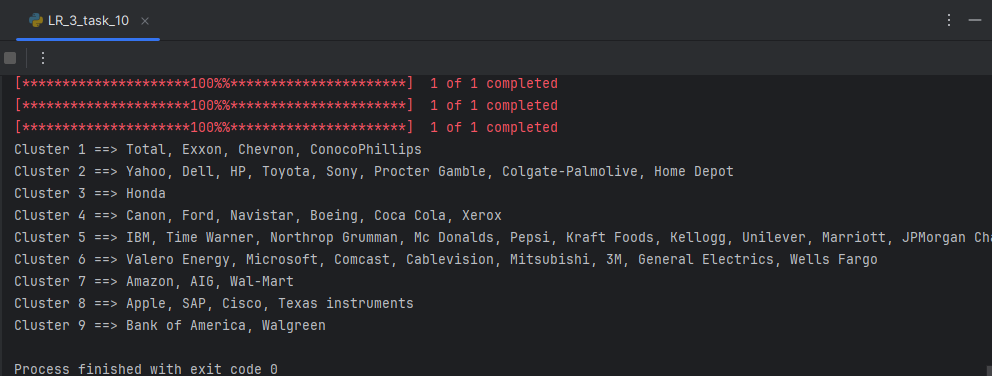


**Завдання №10:** Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

Код:

import datetime  
import json  
import numpy as np  
import matplotlib as plt  
from sklearn import covariance, cluster  
import yfinance as yf  
  
  
# Завантаження прив'язок символів компаній до їх повних назв  
input\_file = 'company\_symbol\_mapping.json'  
with open (input\_file, 'r') as f:  
 company\_symbols\_map = json.load(f)  
  
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 = [yf.download(symbol, start=start\_date, end=end\_date,  
 ignore\_tz=True) for symbol in symbols]  
  
# Вилучення котирувань, що відповідають  
# відкриттю та закриттю біржі  
opening\_quotes\_temp = []  
closing\_quotes\_temp = []  
  
for quote in quotes:  
 if 'Open' in quote and 'Close' in quote:  
 opening\_quotes\_temp.append(quote['Open'].dropna().tolist())  
 closing\_quotes\_temp.append(quote['Close'].dropna().tolist())  
  
opening\_quotes = []  
for items in opening\_quotes\_temp:  
 if len(items) != 0:  
 opening\_quotes.append(items)  
  
closing\_quotes = []  
for items in closing\_quotes\_temp:  
 if len(items) != 0:  
 closing\_quotes.append(items)  
  
opening\_quotes = np.array(opening\_quotes).astype(np.float64)  
closing\_quotes = np.array(closing\_quotes).astype(np.float64)  
  
# Обчислення різниці між двома видами котирувань  
quotes\_diff = closing\_quotes - opening\_quotes  
  
X = quotes\_diff.copy().T  
X /= X.std(axis=0)  
  
# Створення моделі графа  
edge\_model = covariance.GraphicalLassoCV()  
  
# Навчання моделі  
with np.errstate(invalid='ignore'):  
 edge\_model.fit(X)  
  
# 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\_indices = np.where(labels == i)[0]  
 cluster\_names = [names[index] for index in cluster\_indices if index < len(names)]  
 print("Cluster", i + 1, "==>", ', '.join(cluster\_names))

**Результат виконання:**

****

**Висновок:** використовуючи спеціалізовані бібліотеки і мову

програмування Python дослідив методи регресії та неконтрольованої

класифікації даних у машинному навчанні.