**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

**Хід роботи**

**Завдання 1. Створення регресора однієї змінної.**

Лістинг коду 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  
  
# Завантаження даних  
input\_file = 'data\_singlevar\_regr.txt'  
data = np.loadtxt(input\_file, delimiter=',')  
X, Y = data[:, :-1], data[:, -1]  
  
# Розділення даних на навчальні та тестові  
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 лінійної регресії  
linear\_regressor = linear\_model.LinearRegression()  
linear\_regressor.fit(X\_train, Y\_train)  
  
# Прогнозування результатів  
Y\_test\_pred = linear\_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()  
  
# Виведення результатів  
print("Linear regressor performance:")  
print(f"Mean absolute error = {round(sm.mean\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")

print(f"Mean squared error = {round(sm.mean\_squared\_error(Y\_test, Y\_test\_pred), 2)}")  
  
  
print(f"Median absolute error = {round(sm.median\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Explain variance score = {round(sm.explained\_variance\_score(Y\_test, Y\_test\_pred), 2)}")  
print(f"R2 score = {round(sm.r2\_score(Y\_test, Y\_test\_pred), 2)}")  
  
# Збереження моделі  
output\_model\_file = 'model.pkl'  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(linear\_regressor, f)  
  
# Завантаження моделі  
with open(output\_model\_file, 'rb') as f:  
 model\_linregr = pickle.load(f)  
  
Y\_test\_pred\_new = model\_linregr.predict(X\_test)  
print(f"\nNew mean absolute error = {round(sm.mean\_absolute\_error(Y\_test, Y\_test\_pred\_new), 2)}")

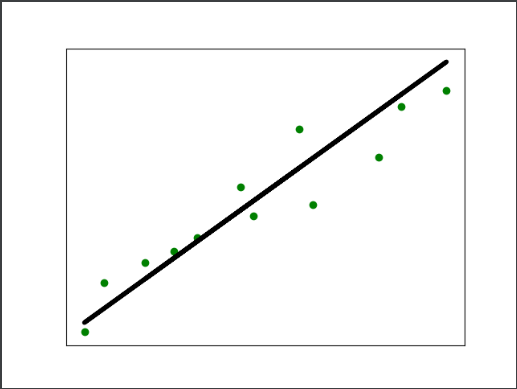


Рис.1 – Результат виконання лінійної регресії.

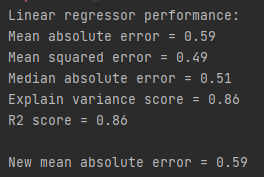


Рис.2 – Аналіз моделі та її роботи в коді та після серіалізації, завантаження зі серіалізованого файлу.

**Завдання 2. Передбачення за допомогою регресії однієї змінної.**

За номером 8 буде використано дані з файлу data\_regr\_3.txt.

Лістинг коду файлу LR\_3\_task\_2.py:

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
# Завантаження даних  
input\_file = 'data\_regr\_3.txt'  
data = np.loadtxt(input\_file, delimiter=',')  
X, Y = data[:, :-1], data[:, -1]  
  
# Розділення даних на навчальні та тестові  
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 лінійної регресії  
linear\_regressor = linear\_model.LinearRegression()  
linear\_regressor.fit(X\_train, Y\_train)  
  
# Прогнозування результатів  
Y\_test\_pred = linear\_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()  
  
# Виведення результатів  
print("Linear regressor performance:")  
print(f"Mean absolute error = {round(sm.mean\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Mean squared error = {round(sm.mean\_squared\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Median absolute error = {round(sm.median\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Explain variance score = {round(sm.explained\_variance\_score(Y\_test, Y\_test\_pred), 2)}")  
print(f"R2 score = {round(sm.r2\_score(Y\_test, Y\_test\_pred), 2)}")

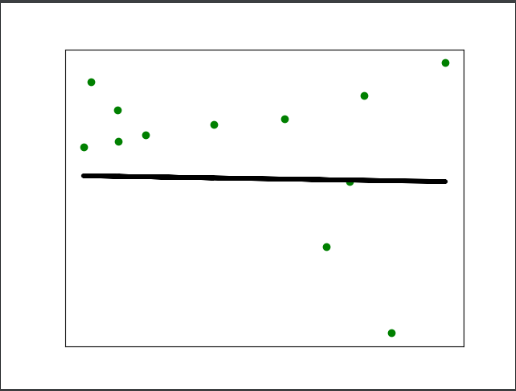


Рис.3 – Результат виконання лінійної регресії за власними даними.

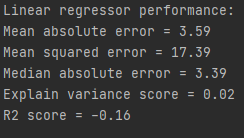


Рис.4 – Аналіз моделі за власними даними.

**Завдання 3. Створення багатовимірного регресора.**

Лістинг коду 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  
import matplotlib.pyplot as plt  
  
# Завантаження даних  
input\_file = 'data\_multivar\_regr.txt'  
data = np.loadtxt(input\_file, delimiter=',')  
X, Y = data[:, :-1], data[:, -1]  
  
# Розділення даних на навчальні та тестові  
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 лінійної регресії  
linear\_regressor = linear\_model.LinearRegression()  
linear\_regressor.fit(X\_train, Y\_train)  
  
# Прогнозування результатів  
Y\_test\_pred = linear\_regressor.predict(X\_test)  
  
# Виведення результатів  
print("Linear regressor performance:")  
print(f"Mean absolute error = {round(sm.mean\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Mean squared error = {round(sm.mean\_squared\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Median absolute error = {round(sm.median\_absolute\_error(Y\_test, Y\_test\_pred), 2)}")  
print(f"Explain variance score = {round(sm.explained\_variance\_score(Y\_test, Y\_test\_pred), 2)}")  
print(f"R2 score = {round(sm.r2\_score(Y\_test, Y\_test\_pred), 2)}")  
  
# Створення поліноміальної регресії  
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(f"Linear regression:\n{linear\_regressor.predict(datapoint)}")  
print(f"Polynomial regression:\n{poly\_linear\_model.predict(poly\_datapoint)}")

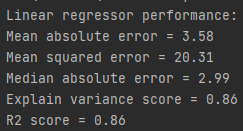


Рис.5 – Характеристика моделі лінійного регресора на даних з багатьма ознаками.

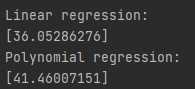


Рис.6 – Порівняння лінійного та поліноміального регресорів.

**Завдання 4. Регресія багатьох змінних.**

Лістинг коду файлу LR\_3\_task\_4.py:

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import linear\_model, datasets  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
  
diabetes = datasets.load\_diabetes()  
X = diabetes.data[:, np.newaxis, 2]  
Y = diabetes.target  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.5, random\_state=0)  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, Y\_train)  
Y\_pred = regressor.predict(X\_test)  
  
print(f"Mean squared error = {round(mean\_squared\_error(Y\_test, Y\_pred), 2)}")  
print(f"Mean absolute error = {round(mean\_absolute\_error(Y\_test, Y\_pred), 2)}")  
print(f"R2 score = {round(r2\_score(Y\_test, Y\_pred), 2)}")  
print(f"Regression coefficient = {round(regressor.coef\_[0], 2)}")  
print(f"Regression intercept = {round(regressor.intercept\_, 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('Measured')  
ax.set\_ylabel('Predicted')  
plt.show()

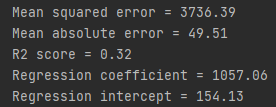


Рис.7 – Характеристика ефективності лінійної регресії на даних про діабет.

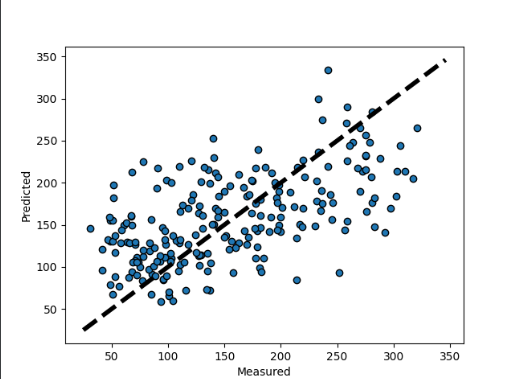


Рис.8 – Графік результату лінійної регресії даних про діабет.

**Завдання 5. Самостійна побудова регресії.**

За номером 8 буде використано спосіб варіанту 8.

Лістинг коду LR\_3\_task\_5.py:

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import linear\_model  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 4  
y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
indices = np.argsort(X, axis=0)  
X = X[indices].reshape(-1, 1)  
Y = y[indices].reshape(-1, 1)  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.5, random\_state=0)  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, Y\_train)  
Y\_pred = regressor.predict(X\_test)  
  
print(f"Mean absolute error = {round(mean\_absolute\_error(Y\_test, Y\_pred), 2)}")  
print(f"Mean squared error = {round(mean\_squared\_error(Y\_test, Y\_pred), 2)}")  
print(f"Regression coefficient = {round(regressor.coef\_[0][0], 2)}")  
print(f"Regression intercept = {round(regressor.intercept\_[0], 2)}")  
print(f"R2 score = {round(r2\_score(Y\_test, Y\_pred), 2)}")  
  
plt.scatter(X, Y, edgecolors=(0, 0, 0))  
plt.plot(X\_test, Y\_pred, color="red")  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.show()  
  
poly = PolynomialFeatures(degree=2)  
X\_poly = poly.fit\_transform(X)  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_poly, Y)  
Y\_pred = regressor.predict(X\_poly)  
  
print(f"Mean absolute error = {round(mean\_absolute\_error(Y, Y\_pred), 2)}")  
print(f"Mean squared error = {round(mean\_squared\_error(Y, Y\_pred), 2)}")  
print(f"Regression coefficient = {round(regressor.coef\_[0][0], 2)}")  
print(f"Regression intercept = {round(regressor.intercept\_[0], 2)}")  
print(f"R2 score = {round(r2\_score(Y, Y\_pred), 2)}")  
  
plt.scatter(X, Y, edgecolors=(0, 0, 0))  
plt.plot(X, Y\_pred, color="red")  
plt.xlabel('X')  
plt.ylabel('Y')  
plt.show()

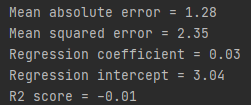


Рис.9 – Характеристика лінійної регресії випадкових даних.

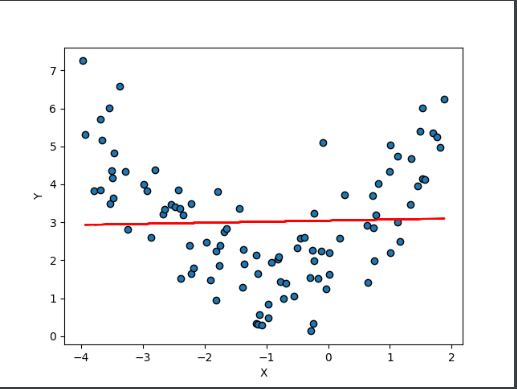


Рис.10 – Лінійна регресія випадкових даних.

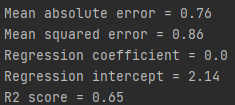


Рис.11 – Характеристика поліномінальної регресії випадкових даних.

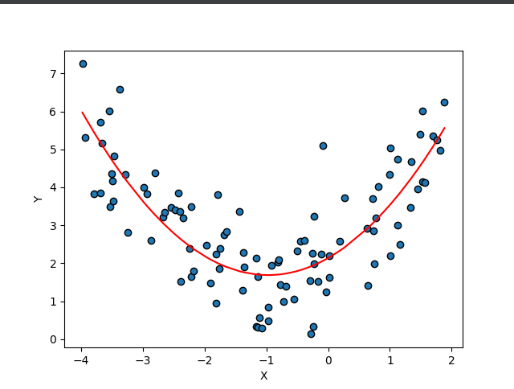


Рис.12 – Поліномінальна регресія випадкових даних.

**Завдання 6. Побудова кривих навчання.**

Лістинг коду 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.model\_selection import train\_test\_split  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures  
  
def plot\_learning\_curves(model, X, Y, m):  
 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="Training set")  
 plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label="Validation set")  
 plt.legend(loc="upper right", fontsize=14)  
 plt.xlabel("Training set size", fontsize=14)  
 plt.ylabel("RMSE", fontsize=14)  
 plt.show()  
  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 5  
Y = 0.5 \* X\*\*2 + X + 2 + np.random.randn(m, 1)  
  
indices = np.argsort(X, axis=0)  
X = X[indices].reshape(-1, 1)  
Y = Y[indices].reshape(-1, 1)  
linear\_reg = LinearRegression()  
plot\_learning\_curves(linear\_reg, X, Y, m)  
  
polynomial\_regression = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
])  
plot\_learning\_curves(polynomial\_regression, X, Y, m)  
  
polynomial\_regression = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=2, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
])  
plot\_learning\_curves(polynomial\_regression, X, Y, m)

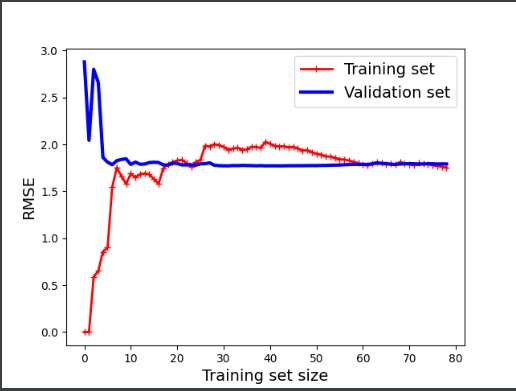


Рис.13 – Криві навчання для лінійної моделі.

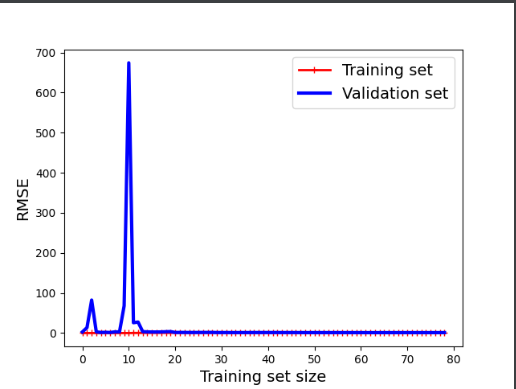


Рис.14 – Криві навчання для поліномінальної моделі 10го ступеня.

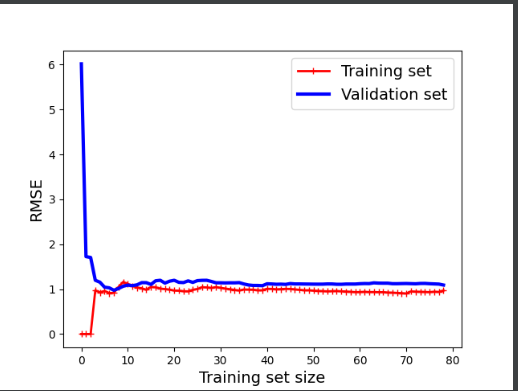


Рис.15 – Криві навчання для поліномінальної моделі 2го ступеня.

**Завдання 7. Кластеризація даних за допомогою методу k-середніх.**

Лістинг коду файлу LR\_3\_task\_7.py:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
num\_clusters = 5  
  
plt.figure()  
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='k', s=30)  
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()  
  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
kmeans.fit(X)  
  
step\_size = 0.01  
x\_values, y\_values = np.meshgrid(np.arange(x\_min, x\_max, step\_size), np.arange(y\_min, y\_max, step\_size))  
output = kmeans.predict(np.c\_[x\_values.ravel(), y\_values.ravel()])  
  
output = output.reshape(x\_values.shape)  
plt.figure()  
plt.clf()  
plt.imshow(output, interpolation='nearest',  
 extent=(x\_values.min(), x\_values.max(), y\_values.min(), y\_values.max()),  
 cmap=plt.cm.Paired, aspect='auto', origin='lower')  
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='k', s=30)  
plt.title('Centroids and boundaries obtained using KMeans')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

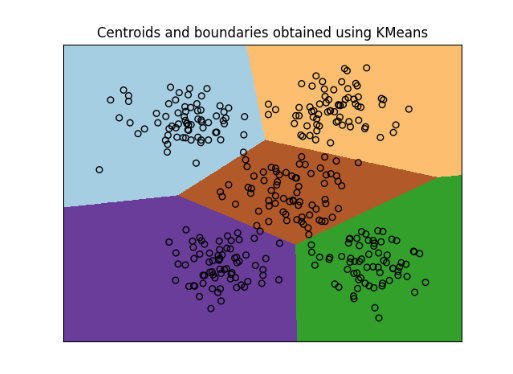


Рис.16 – Відображення кластеризованих даних методом К-середніх.

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

Лістинг коду файлу LR\_3\_task\_8.py:

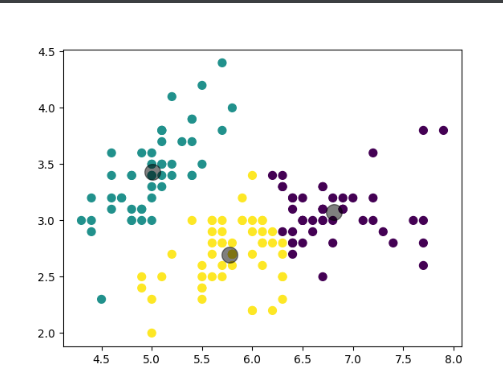


Рис.17 – Ручна кластеризація даних по ірисам.

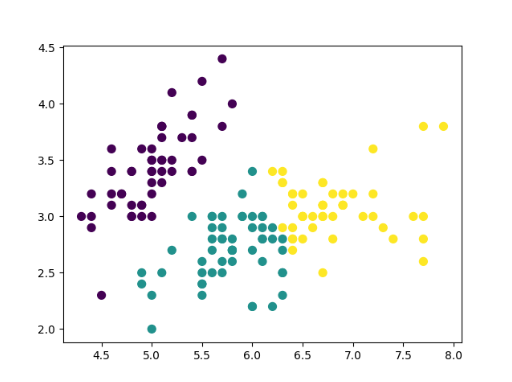


Рис.18 – Кластеризація з використанням створеної функції, випадкове зерно – 2.

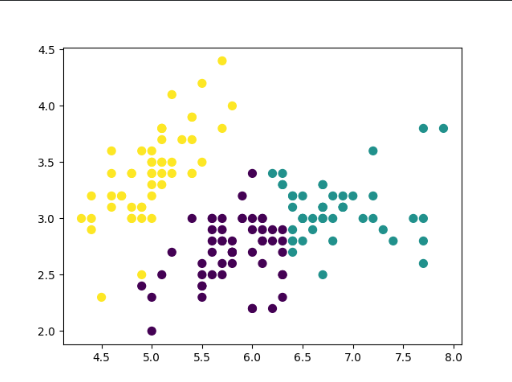


Рис.19 – Кластеризація з використанням створеної функції, випадкове зерно – 0.

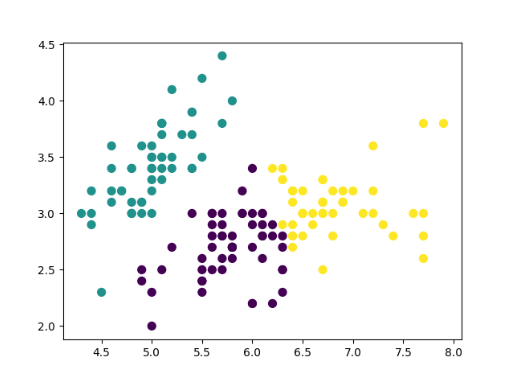


Рис.20 – Кластеризація швидким викликом кластеризатора.

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

Лістинг коду файлу LR\_3\_task\_9.py:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from itertools import cycle  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
bandwidth = estimate\_bandwidth(X, quantile=0.2, n\_samples=500)  
ms = MeanShift(bandwidth=bandwidth, bin\_seeding=True)  
ms.fit(X)  
  
cluster\_centers = ms.cluster\_centers\_  
labels = ms.labels\_  
  
print("cluster\_centers:\n", cluster\_centers)  
print("labels:\n", labels)  
  
plt.figure()  
markers = cycle('o\*sv')  
colors = cycle('bgrcmyk')  
for i, marker in zip(range(len(cluster\_centers)), markers):  
 plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color=next(colors), s=50, label='cluster ' + str(i))  
 cluster\_center = cluster\_centers[i]  
 plt.plot(cluster\_center[0], cluster\_center[1], marker='o', markerfacecolor='k', markeredgecolor='k', markersize=15)  
plt.title(f'Estimated number of clusters: {len(cluster\_centers)}')  
plt.show()

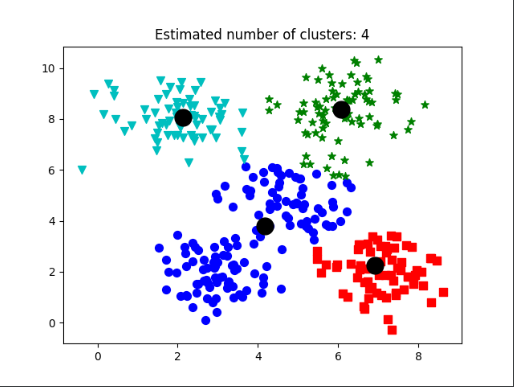


Рис.21 – Відображення кластеризованих даних методом зсуву середнього

***Висновок:*** під час виконання завдань лабораторної роботи було досліджено методи регресії та неконтрольованої класифікації даних у машинному навчанні використовуючи спеціалізовані бібліотеки і мову програмування Python.

Проект до лабораторної роботи можна переглянути за посиланням: <https://github.com/krava02/system_LR>