**Практична робота № 3**

**Варіант 13**

**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 3.1:** Створення регресора однієї змінної. Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data\_singlevar\_regr.txt

Лістинг файлу 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)}")

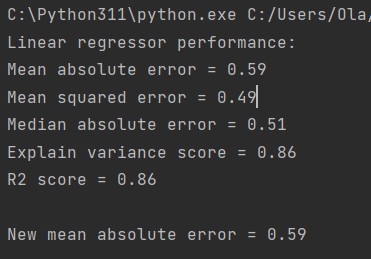
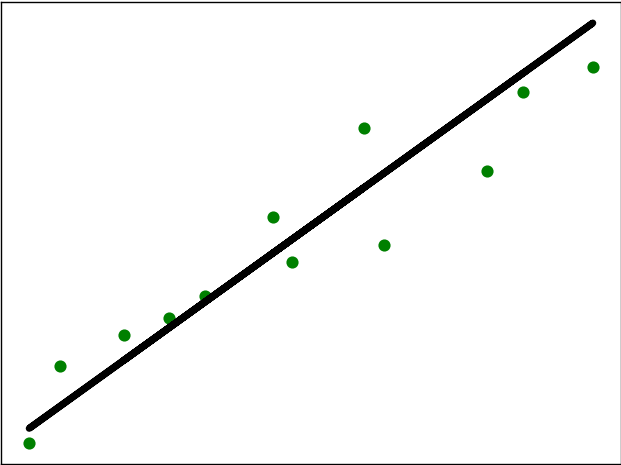
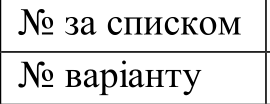


Рис.3.1. task-1.py

**Завдання 3.2:** Передбачення за допомогою регресії однієї змінної. Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту.

**Варіант 3 файл: data\_regr\_3.txt**



Лістинг файлу 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)}")

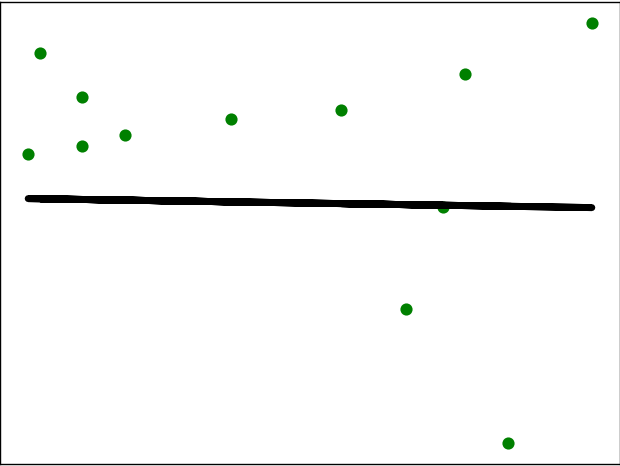
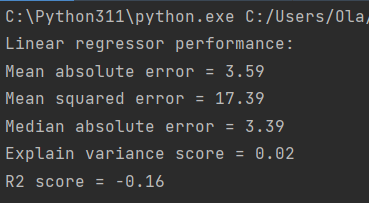
 

Рис.3.2. task-2.py

**Завдання 3.3:** Створення багатовимірного регресора. Використовувати файл вхідних даних: data\_multivar\_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Лістинг файлу 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)}")

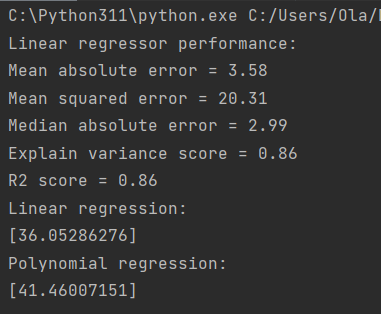


Рис.3.3. task-3.py

**Завдання 3.4:** Регресія багатьох змінних. Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets.

Набір даних містить 10 вихідних змінних — вік, стать, індекс маси тіла, середній артеріальний тиск і шість вимірювань сироватки крові, отриманих у 442 пацієнтів із цукровим діабетом, а також реакцію, що цікавить, — кількісний показник прогресування захворювання через 1 рік після вихідного рівня. Отже, існує 442 екземпляри з 10 атрибутами. Колонка 11 є кількісною мірою прогресування захворювання через 1 рік після вихідного рівня. Кожен з цих 10 атрибутів був відцентрований по середньому та масштабований за часом стандартного відхилення n\_samples (тобто сума квадратів кожного стовпця складає1).

Лістинг файлу 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 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], 2)}")  
print(f"Regression intercept = {round(regressor.intercept\_, 2)}")  
print(f"R2 score = {round(r2\_score(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('Measured')  
ax.set\_ylabel('Predicted')  
plt.show()

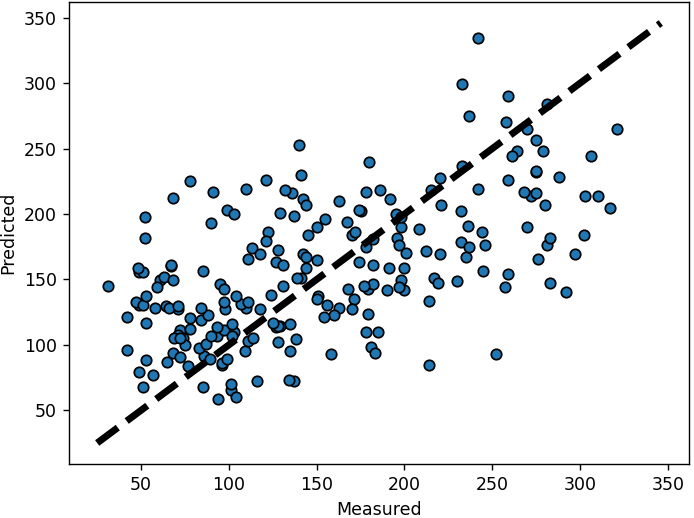
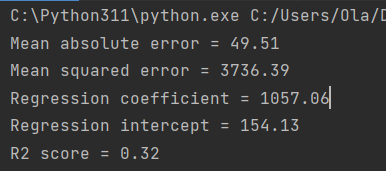
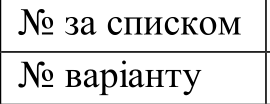
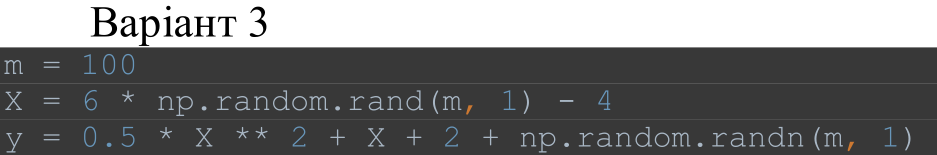
` 

Рис.3.4. task-4.py

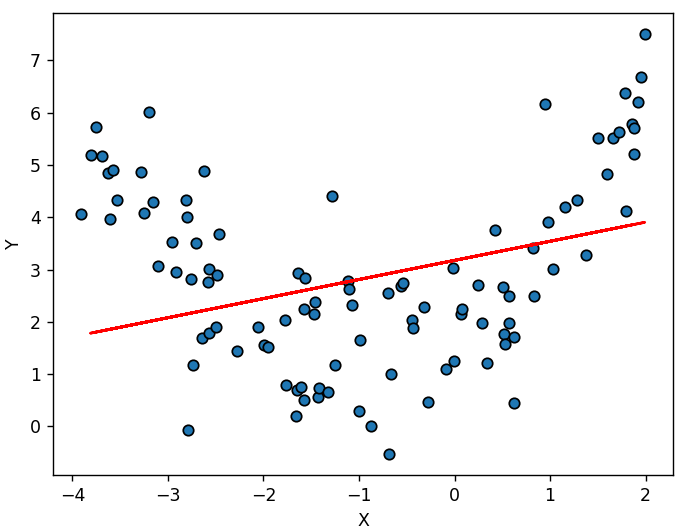
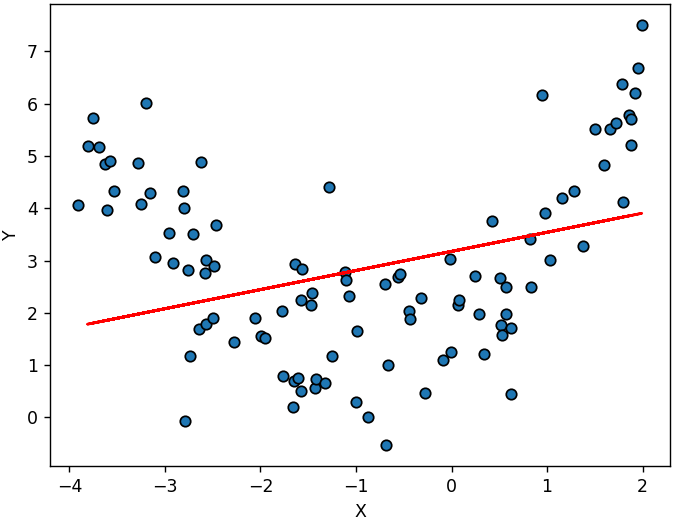
**Завдання 3.5:** Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.





Лістинг файлу 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()

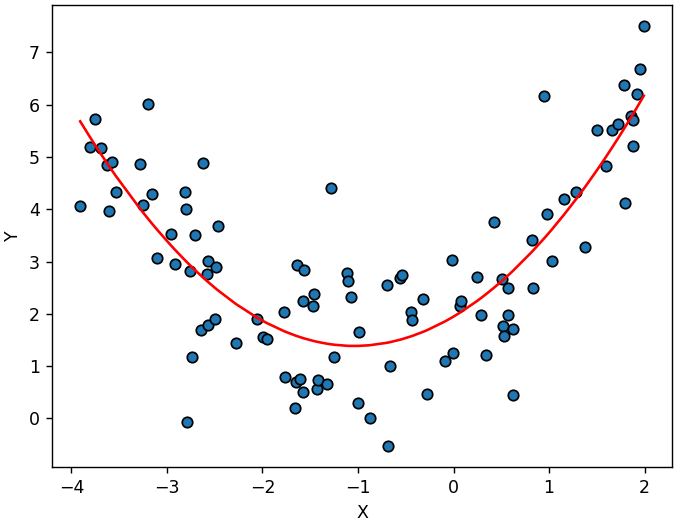
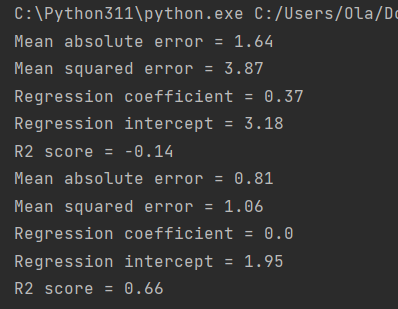
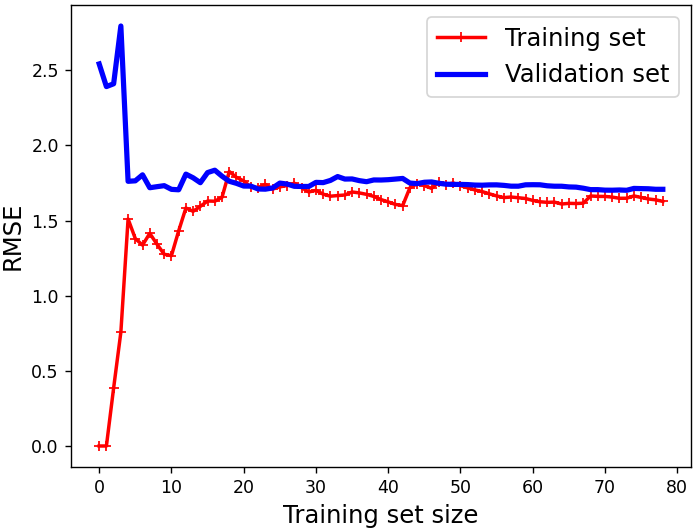
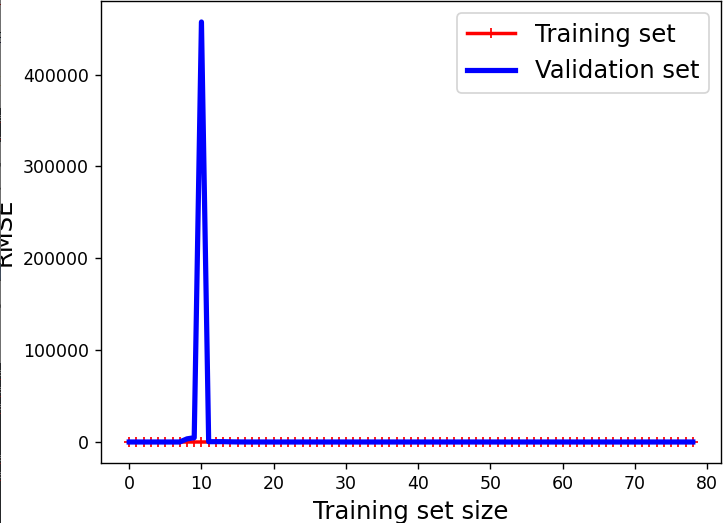
 

Рис.3.5. task-5.py

**Завдання 3.6:** Побудова кривих навчання. Побудуйте криві навчання для ваших даних у попередньому завданні.

Лістинг файлу 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):  
 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)  
  
polynomial\_regression = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
])  
plot\_learning\_curves(polynomial\_regression, X, Y)  
  
polynomial\_regression = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=2, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
])  
plot\_learning\_curves(polynomial\_regression, X, Y)

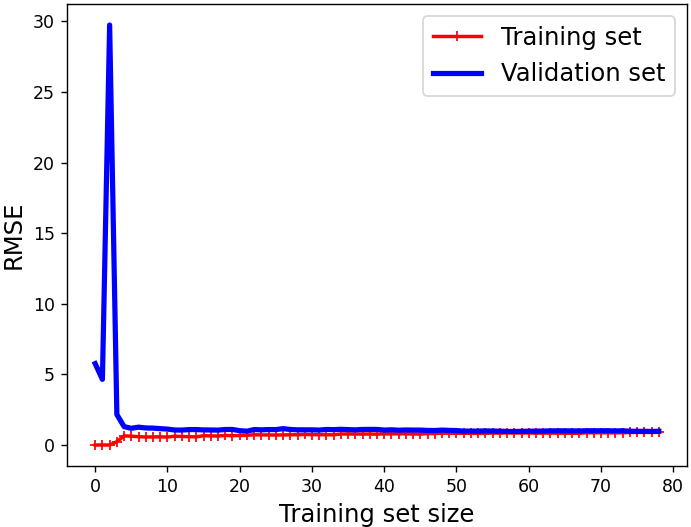


Рис.3.6. task-6.py

**Завдання 3.7:** Кластеризація даних за допомогою методу k-середніх. Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data\_clustering.txt.

Лістинг файлу task-7.py

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='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()

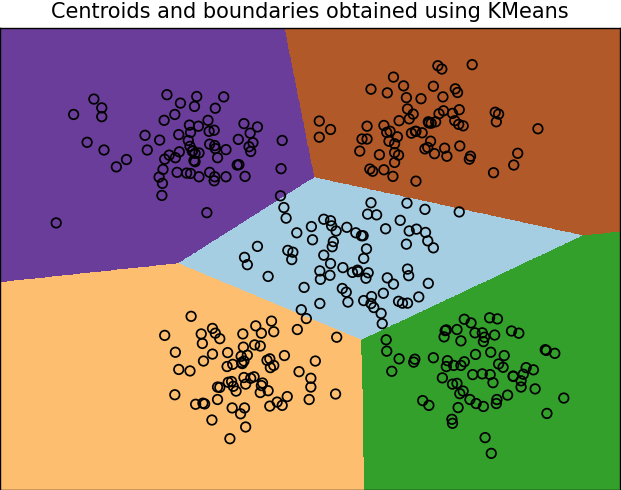
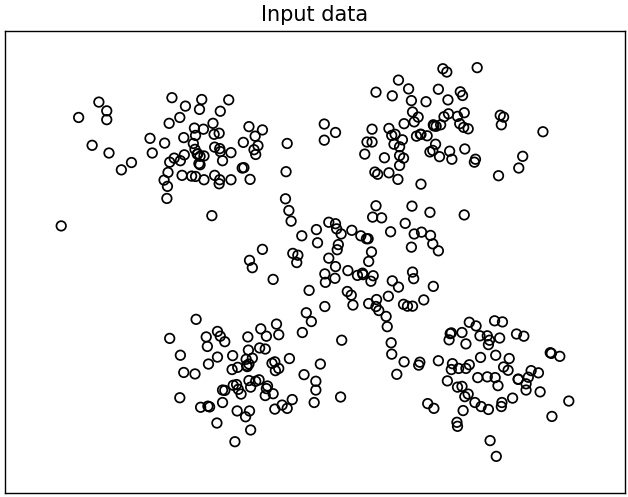
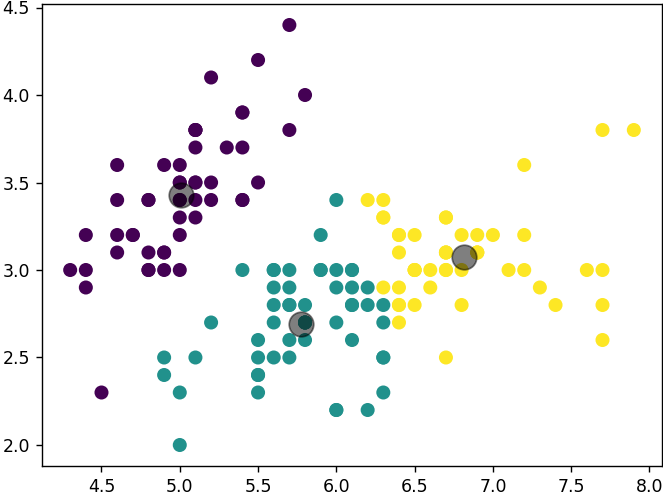
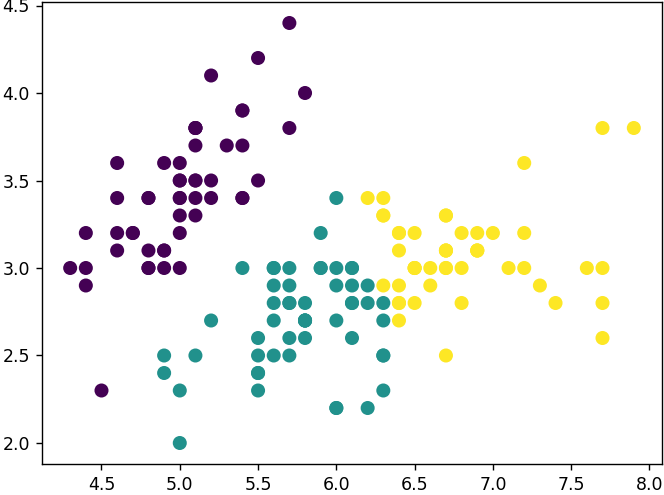


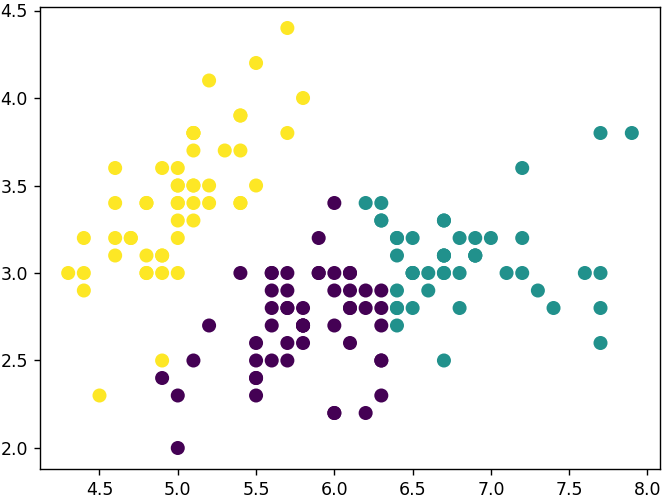
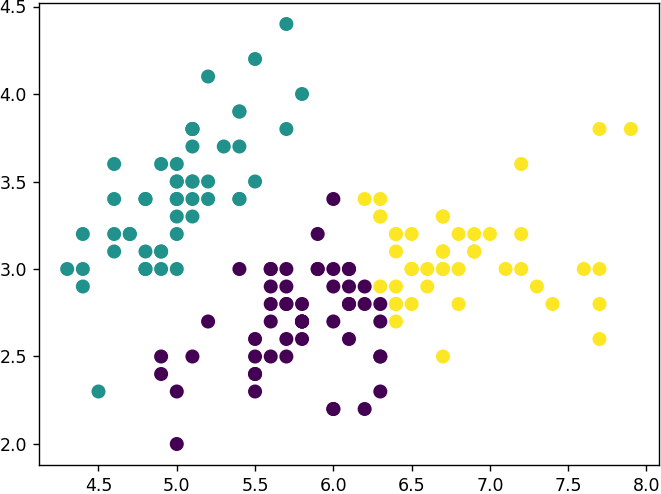
Рис.3.7. task-7.py

**Завдання 3.8:** Кластеризація K-середніх для набору даних Iris. Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки.

Лістинг файлу task-8.py

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

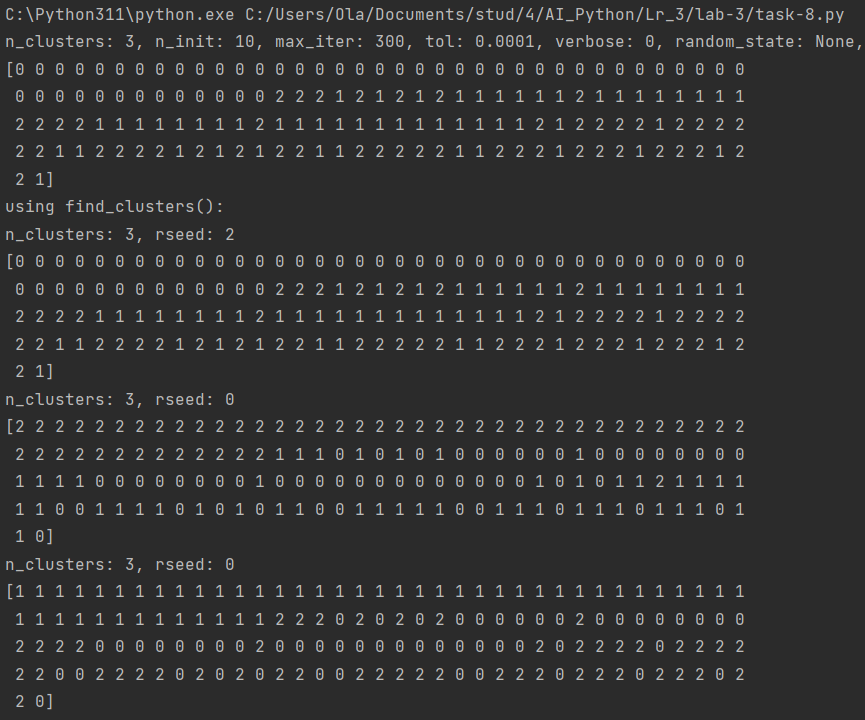
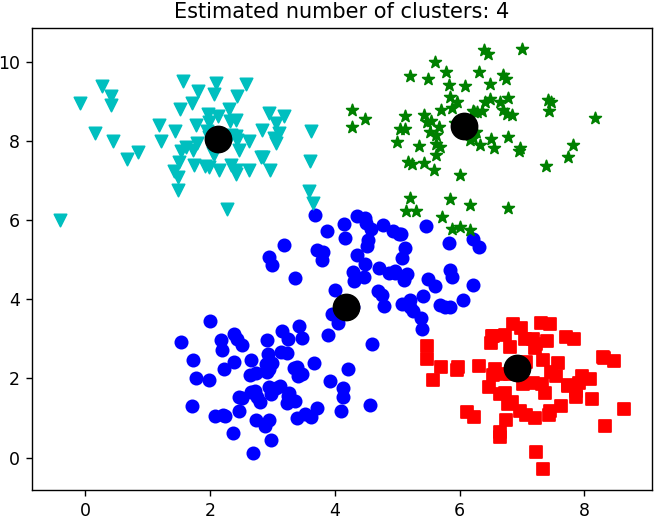


Рис.3.8. task-8.pyQ

**Завдання 3.9:** Оцінка кількості кластерів з використанням методу зсуву середнього. Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data\_clustering.txt.

Лістинг файлу 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()



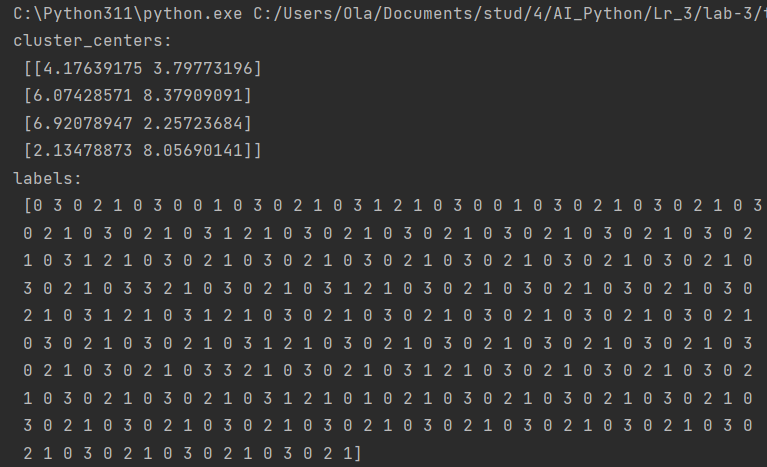


Рис.3.9. task-9.py

[**https://github.com/avrorilka/AI\_Python**](https://github.com/avrorilka/AI_Python)

**Висновки:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python ми дослідити методи методи регресії та неконтрольованої класифікації даних у машинному навчанні. Також ми розглянули поняття регресії, розглянули правила побудови регресійних моделей.