ЛАБОРАТОРНА РОБОТА №3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ

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

Завдання 1. Створення регресора однієї змінної.

```
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))
```

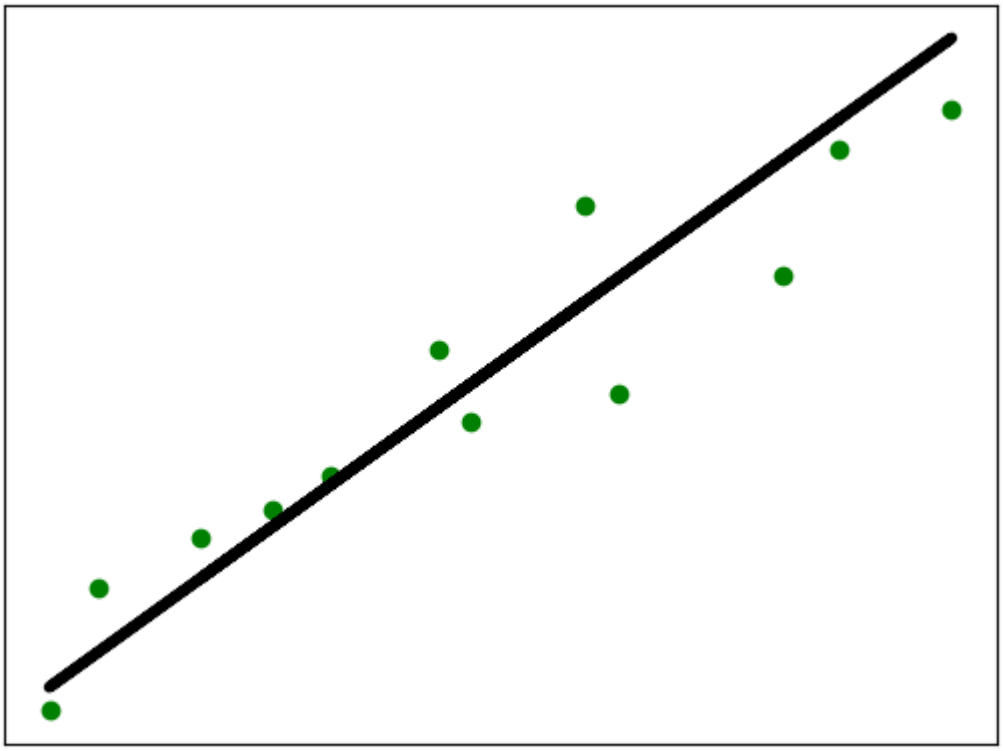


Рис. 1 Результат роботи скрипта LR_3_task_1.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\LR_3_task_1.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
```

Рис. 2 Результат роботи скрипта LR_3_task_1.py

Висновок: модель для вихідних даних побудована валідно. MAE, MSE – середня якість. Показник R2 – добре.

Завдання 2. Передбачення за допомогою регресії однієї змінної.

Номер за списком – 8, варіант – 3

Код скрипта LR_3_task_2.py

```
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 = '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))
```

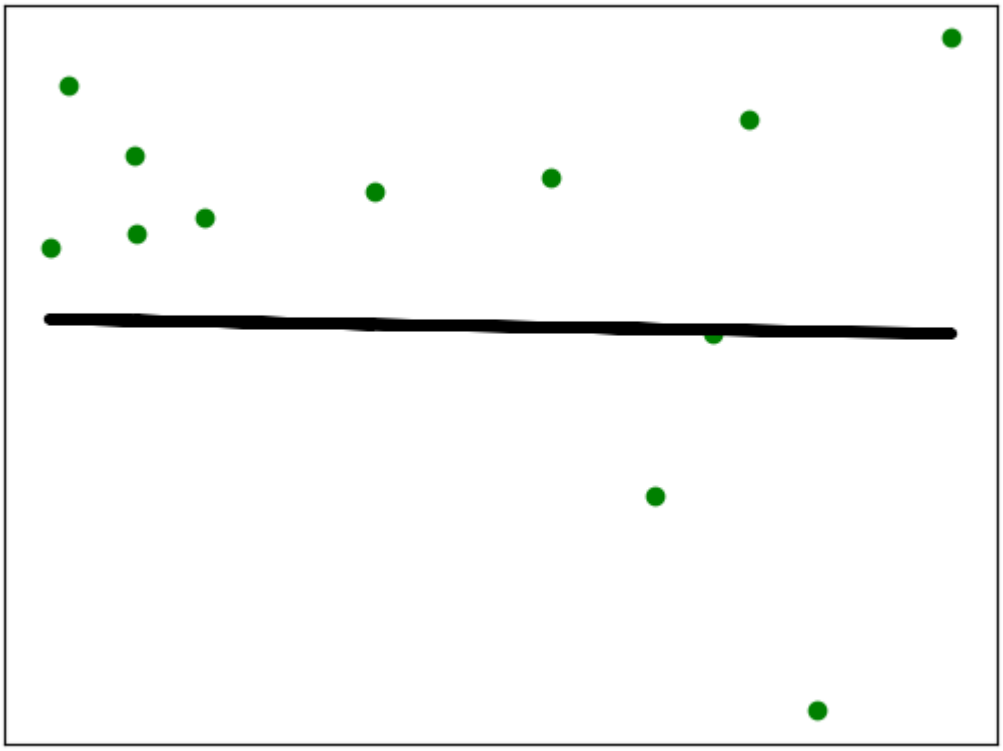


Рис. 3 Результат роботи скрипта LR_3_task_2.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\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
```

Рис. 4 Результат роботи скрипта LR_3_task_2.py

Завдання 3. Створення багатовимірного регресора.

```
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("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))
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))
```

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\LR_3_task_3.py
Linear Regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explained variance score = 0.86
R2 score = 0.86

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46007151]

Process finished with exit code 0
```

Рис. 5 Результат роботи скрипта LR_3_task_3.py

Висновок: Якщо порівнювати з з лінійним регресором, поліноміальний регресор демонструє кращі результати. На це вказує значення 41.45

Завдання 4. Регресія багатьох змінних.

```
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)
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()
```

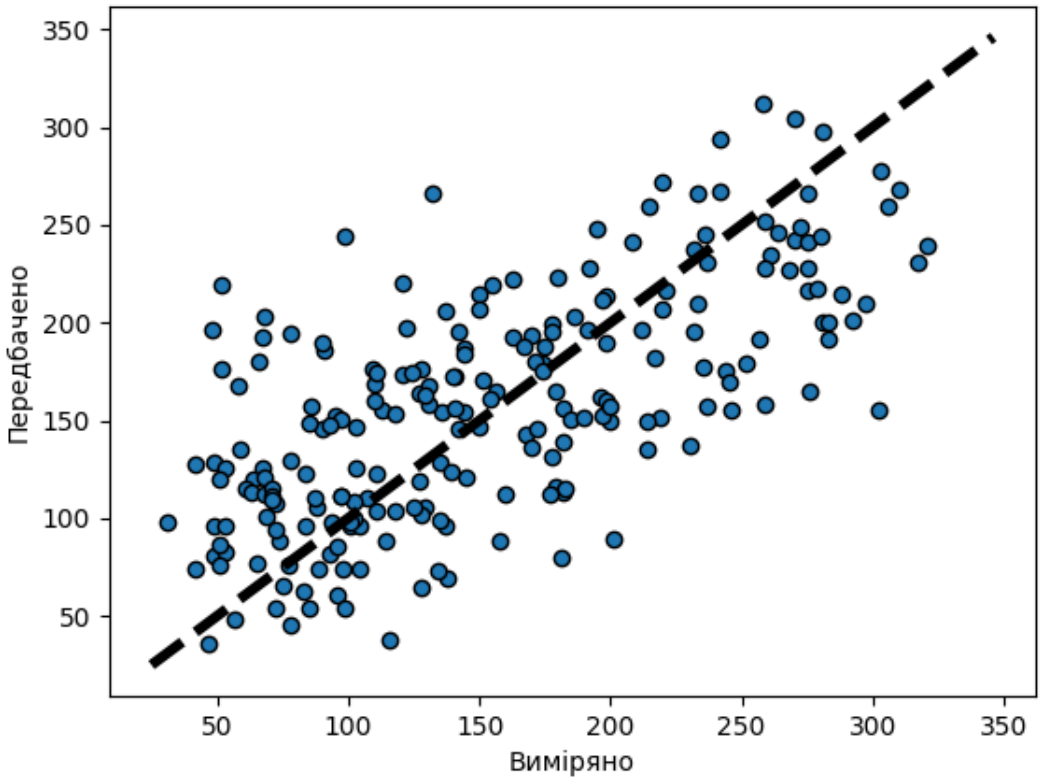


Рис. 6 Результат роботи скрипта LR_3_task_4.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\LR_3_task_4.py
regr.coef = [ -20.4  -265.89  564.65  325.56 -692.16  395.56   23.5   116.36  843.95
   12.72]
regr.intercept = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33

Process finished with exit code 0
```

Рис. 7 Результат роботи скрипта LR_3_task_4.py

Завдання 5. Самостійна побудова регресії.

№ за списком – 10, № варіанту – 8

```
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(-0.5, 0.5, m)
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X, y)
polynomial = PolynomialFeatures(degree=2, include_bias=False)
X_poly = polynomial.fit_transform(X)
polynomial.fit(X_poly, y)
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_poly, y)
y_pred = poly_linear_model.predict(X_poly)
print("\nr2: ", sm.r2_score(y, y_pred))
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.plot(X, y_pred, "+", color='blue', linewidth=2)
plt.title("Поліноміальна регресія")
plt.show()
```

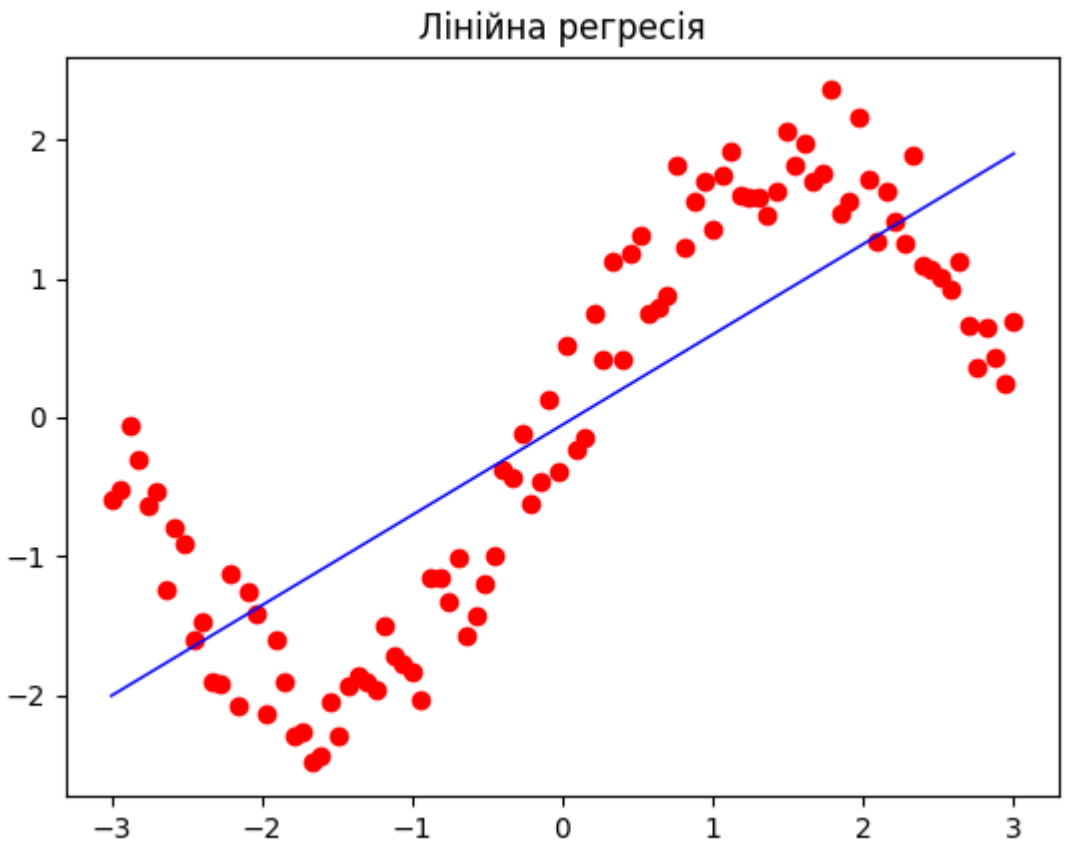


Рис. 8 Результат роботи скрипта LR_3_task_5.py

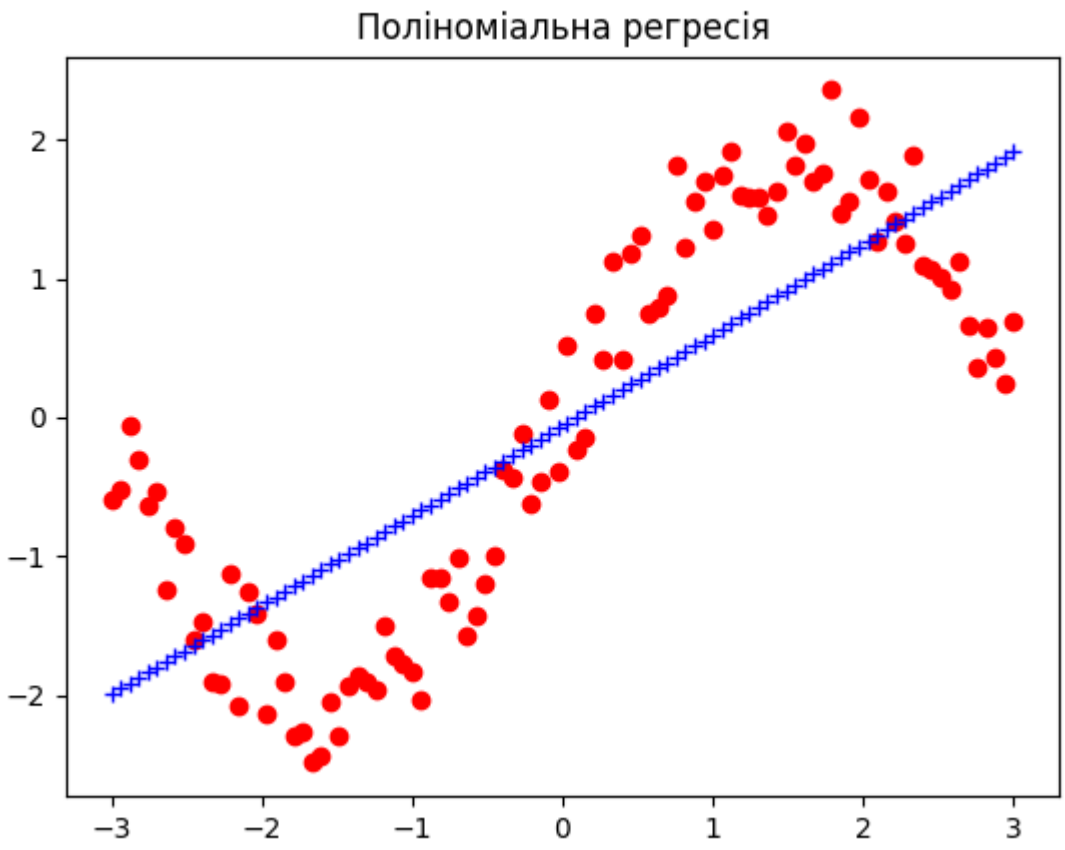


Рис. 9 Результат роботи скрипта LR_3_task_5.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\LR_3_task_5.py

r2:  0.6341579010784149

Process finished with exit code 0
```

Рис. 10 Результат роботи скрипта LR_3_task_5.py

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

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

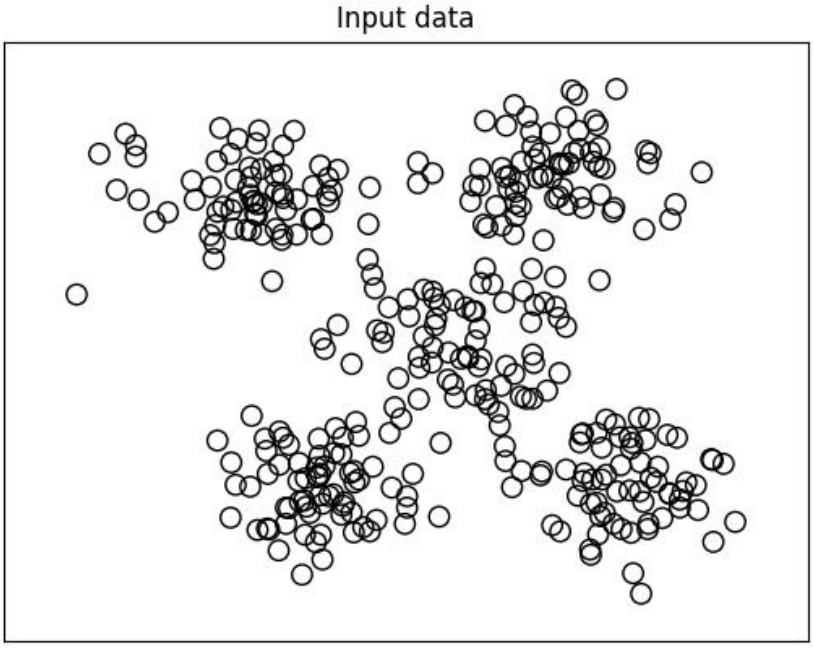


Рис. 11 Результат роботи скрипта LR_3_task_7.py

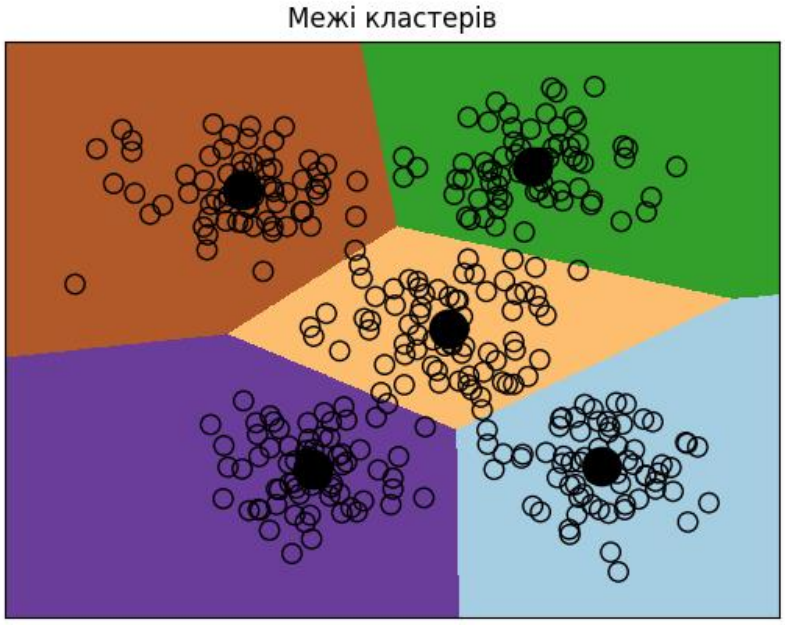


Рис. 12 Результат роботи скрипта LR_3_task_7.py

Висновок: метод k-середніх валідно працює, але за умови, відомої кількісті кластерів.

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

Код скрипту:

```
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, ran-
dom_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()
```

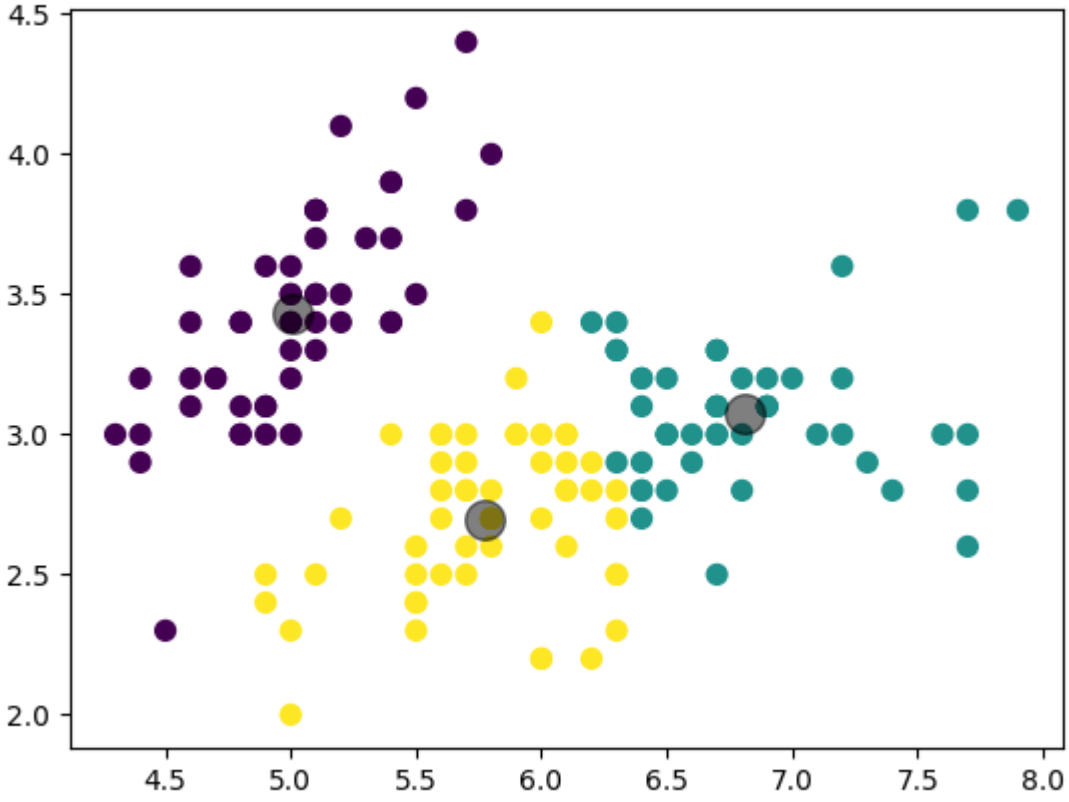


Рис. 13 Результат роботи скрипта LR_3_task_8.py

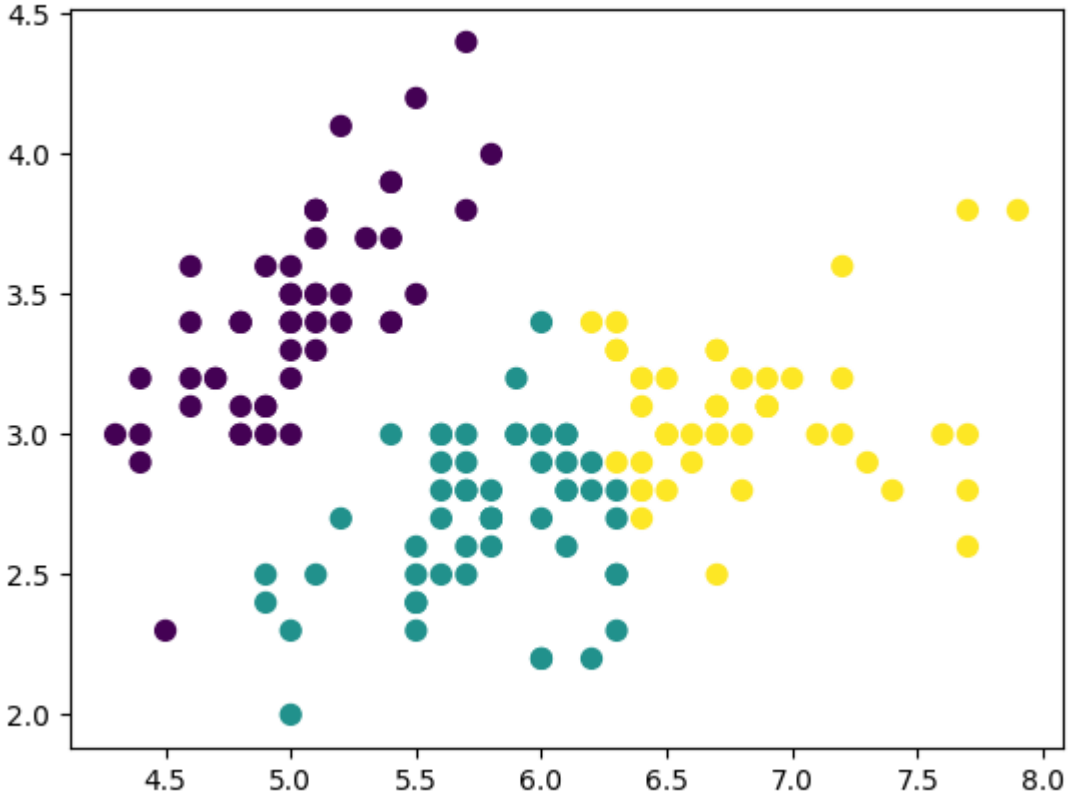
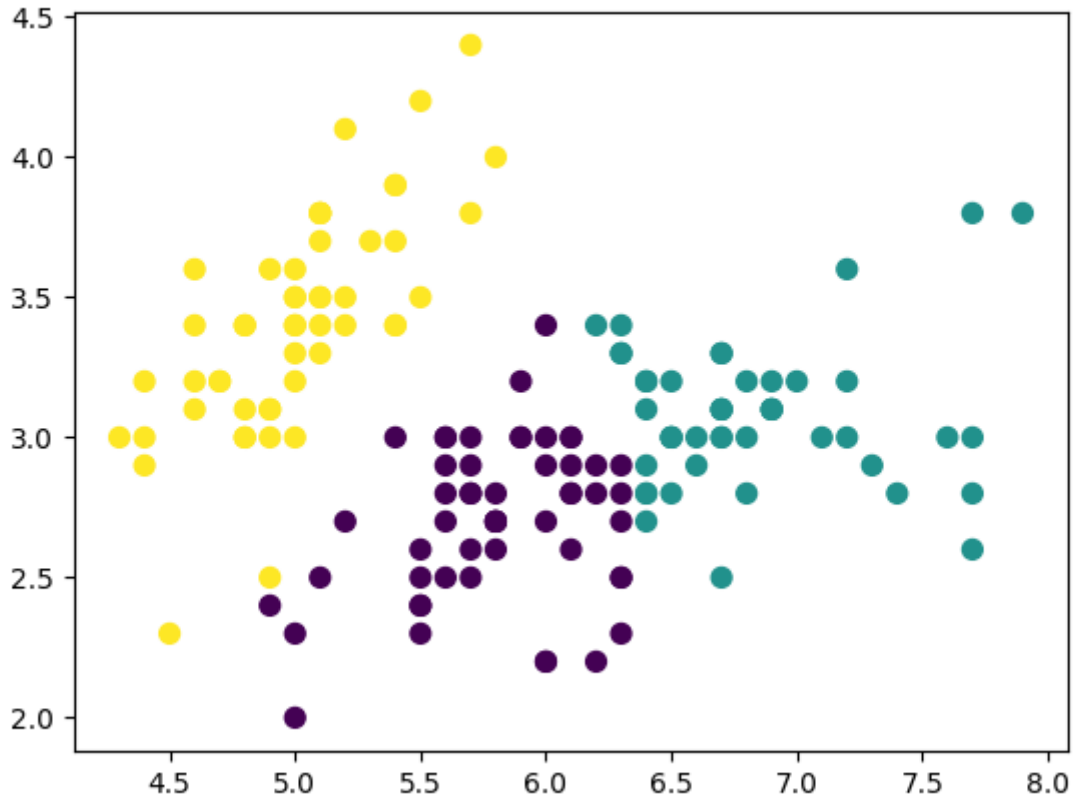


Рис. 14 Результат роботи скрипта LR_3_task_8.py



\

Рис. 15 Результат роботи скрипта LR_3_task_8.py

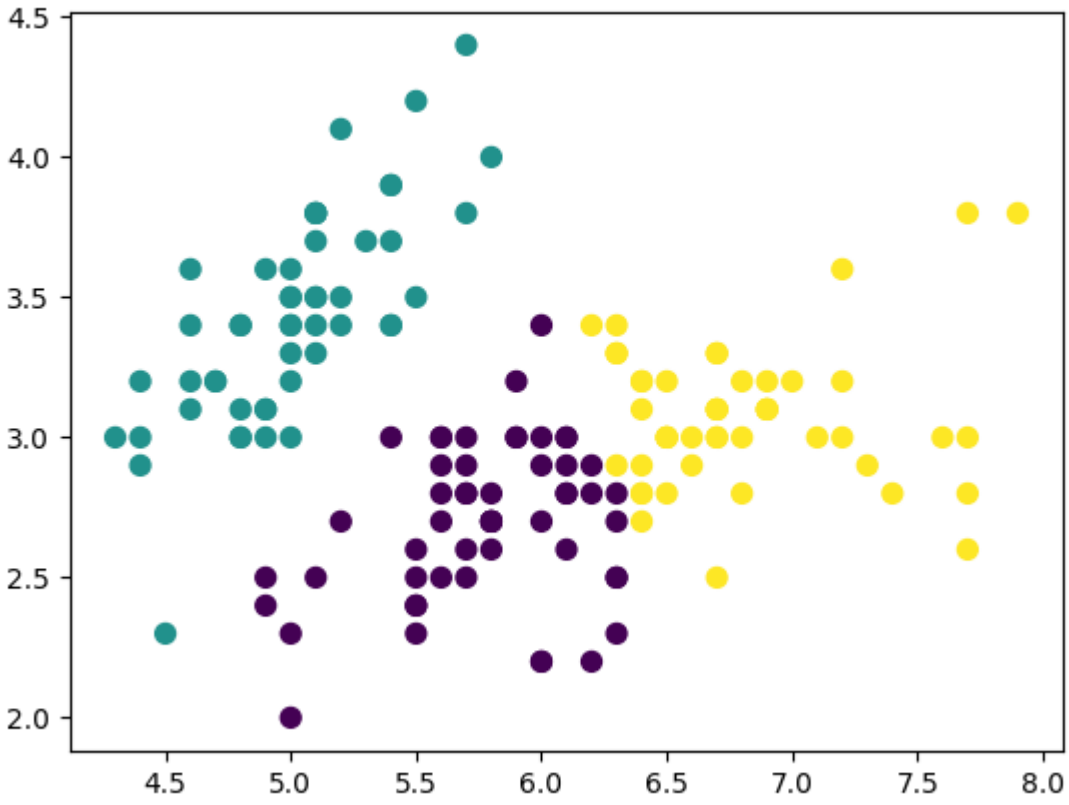


Рис. 16 Результат роботи скрипта LR_3_task_8.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\LR_3_task_8.py
n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, ran-dom_state: None, copy_x: True
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 1 2 1 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2
 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
using find_clusters():
n_clusters: 3, rseed: 2
n_clusters: 3, rseed: 0
n_clusters: 3, rseed: 0

Process finished with exit code 0
```

Рис. 17 Результат роботи скрипта LR_3_task_8.py

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

Код скрипту:

```
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('Clusters')
plt.show()
```

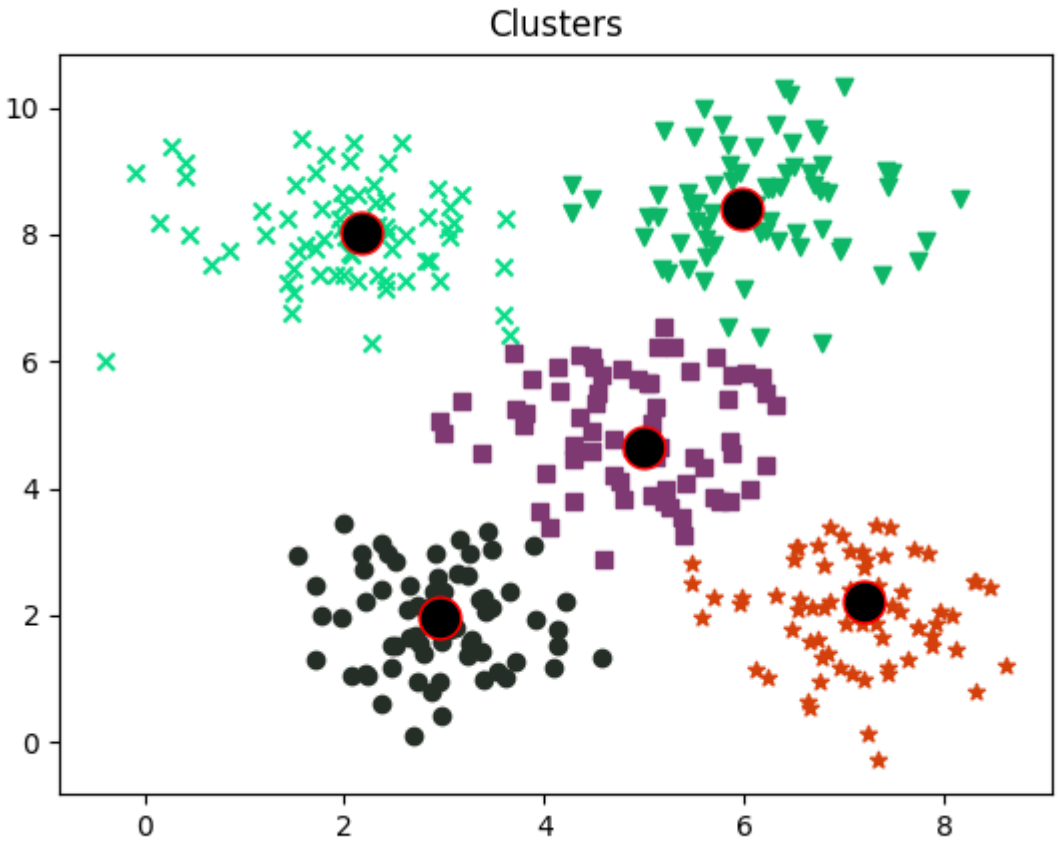


Рис. 18 Результат роботи скрипта LR_3_task_9.py

```
C:\Python310\python.exe C:\Users\yuri\Desktop\learn\shi\lab3\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
```

Рис. 19 Результат роботи скрипта LR_3_task_9.py

Код лабораторної роботи знаходиться на репозиторії **shi_labs_koida**, доступ відкритий на пошту ***valeriifilippovzt@gmail.com***.

Висновок по лабораторній роботі:

Під час виконання завдань лабораторної роботи було використано спеціалізовані бібліотеки мови програмування Python для дослідження методів регресії та неконтрольованої класифікації даних у машинному навчанні.