Лабораторна робота №3

з дисципліни Основи штучного інтелекту
студента групи ЗІПЗк-22-1
Перехватова Алевтина Олександрівна
дата виконання: 03.12.2023

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

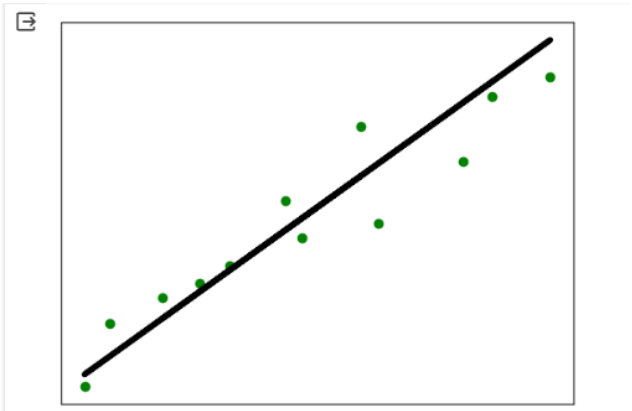
Виконання роботи

Завдання 2.1

Дія1:

```
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
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– Код програми



```
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
```

Рис.1.2 – Реакція програми на дію

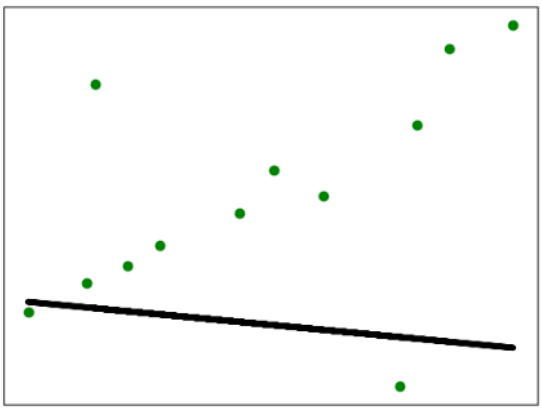
Завдання 2.2

Варіант №2

Дія1:

```
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_regr_2.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
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.1.3– Код програми



```
Linear regressor performance:
Mean absolute error = 2.42
Mean squared error = 9.02
Median absolute error = 2.14
Explain variance score = -0.15
R2 score = -1.61

New mean absolute error = 2.42
```

Рис.1.4 – Реакція програми на дію

Завдання 2.3

Дія1:

```
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
import pickle
import matplotlib.pyplot as plt
# Вхідний файл, який містить дані
input_file = 'data_multivar_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)
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
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))
# 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))
```

Рис.1.5– Код програми

```
Linear regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explain variance score = 0.86
R2 score = 0.86

New mean absolute error = 3.58

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46319764]
```

Рис.1.6 – Реакція програми на дію

Завдання 2.4

Дія1:

```
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
regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)
ypred = regr.predict(Xtest)
print("Coefficients:", regr.coef_)
print("Intercept:", regr.intercept_)
print("R2 Score:", r2_score(ytest, ypred))
print("Mean Absolute Error:", mean_absolute_error(ytest, ypred))
print("Mean Squared Error:", 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()
```

Рис.1.7– Код програми

```
Coefficients: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Intercept: 154.35892852801342
R2 Score: 0.4377497118254099
Mean Absolute Error: 44.800645233553276
Mean Squared Error: 3075.3306886803252
```

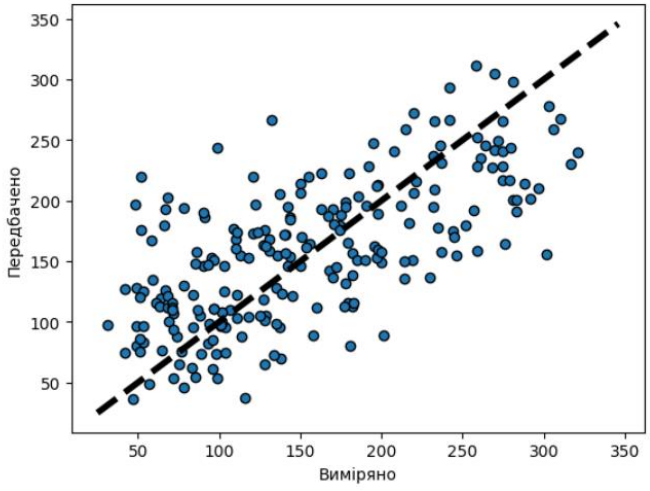


Рис.1.8 – Реакція програми на дію

Завдання 2.5

Дія1:

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Generate random data
np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Reshape X to a column vector
X = X.reshape(-1, 1)

# Plot the original data
plt.scatter(X, y, label='Original Data')

# Linear regression
linear_regressor = LinearRegression()
linear_regressor.fit(X, y)
y_linear_pred = linear_regressor.predict(X)
plt.plot(X, y_linear_pred, label='Linear Regression', color='red')

# Polynomial regression
degree = 3  # You can adjust the degree of the polynomial
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
print("Linear Term (X[0]) coefficient:", poly_regressor.coef_[1])
print("Polynomial Features (X_poly) coefficients:", poly_regressor.coef_[2:])
# Fit LinearRegression to the extended training data
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)

# Print the intercept and coefficients
print("Intercept:", lin_reg.intercept_)
print("Coefficients:", lin_reg.coef_)

y_poly_pred = poly_regressor.predict(X_poly)
plt.plot(X, y_poly_pred, label='Polynomial Regression', color='green')

# Evaluate models
print("Linear Regression - R2 Score:", r2_score(y, y_linear_pred))
print("Polynomial Regression - R2 Score:", r2_score(y, y_poly_pred))

# Display the plot
plt.legend()
plt.title('Linear and Polynomial Regression')
plt.xlabel('X')
```

Рис.1.9– Код програми

```
Linear Term (X[0]) coefficient: -0.0018347718967672328
Polynomial Features (X_poly) coefficients: []
Intercept: -0.024203742634715267
Coefficients: [ 0.34014935 -0.00183477]
Linear Regression - R2 Score: 0.5594486415622437
Polynomial Regression - R2 Score: 0.5594884845764285
```

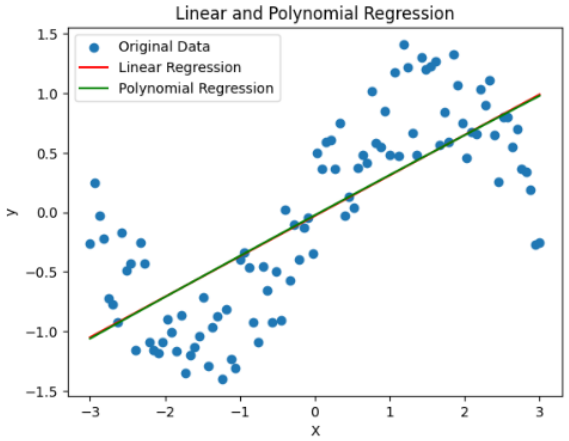


Рис.1.10 – Реакція програми на дію

Завдання 2.6

Дія1:

```python
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
from sklearn.model_selection import train_test_split

np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
X = X.reshape(-1, 1)

degree = 2
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(X)
X_train, X_val, y_train, y_val = train_test_split(X_poly, y, test_size=0.2, random_state=42)

lin_reg = LinearRegression()

train_errors, val_errors = [], []

for m_train in range(1, len(X_train) + 1):
    lin_reg.fit(X_train[:m_train], y_train[:m_train])

    y_train_pred = lin_reg.predict(X_train[:m_train])

    y_val_pred = lin_reg.predict(X_val)

    train_errors.append(mean_squared_error(y_train[:m_train], y_train_pred))
    val_errors.append(mean_squared_error(y_val, y_val_pred))

plt.plot(np.sqrt(train_errors), label='Training Set')
plt.plot(np.sqrt(val_errors), label='Validation Set')
plt.title(f'Learning Curves - Polynomial Degree {degree}')
plt.xlabel('Training Set Size')
plt.ylabel('Root Mean Squared Error')
plt.legend()
plt.show()
```

Рис.1.11– Код програми

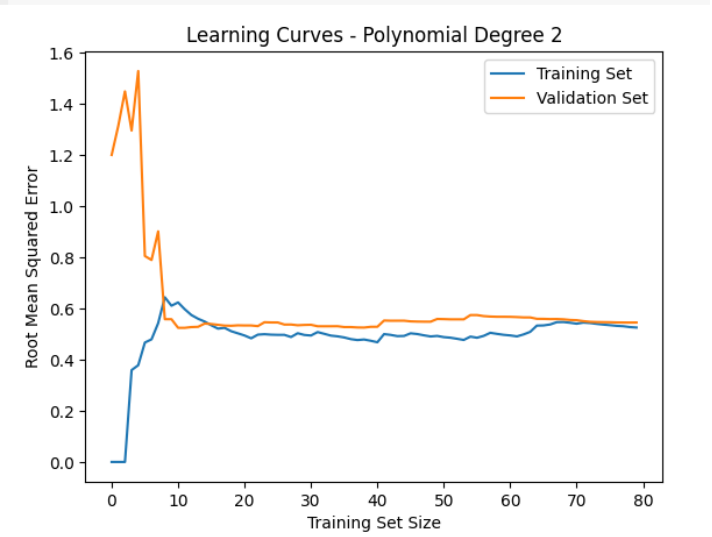


Рис.1.12 – Реакція програми на дію

Завдання 2.7

Дія1:

```
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_clusrets=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('Вхідні данні')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
```

Рис.1.13– Код програми

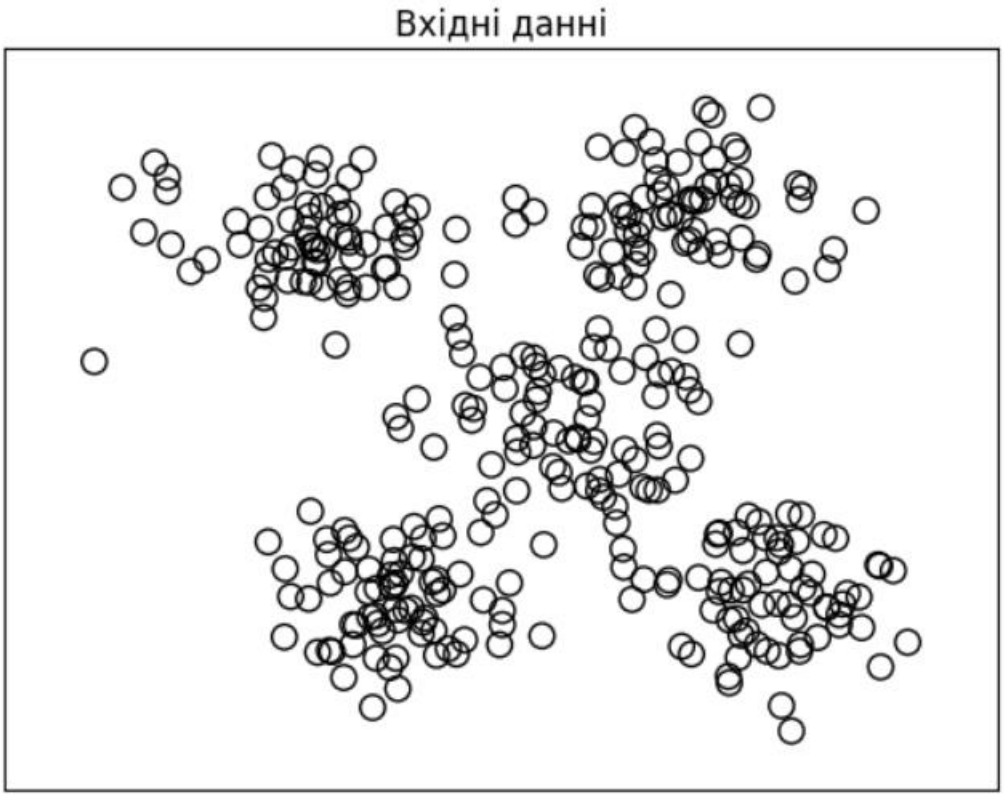


Рис.1.14 – Реакція програми на дію

```
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()
```

Рис.1.15– Код програми

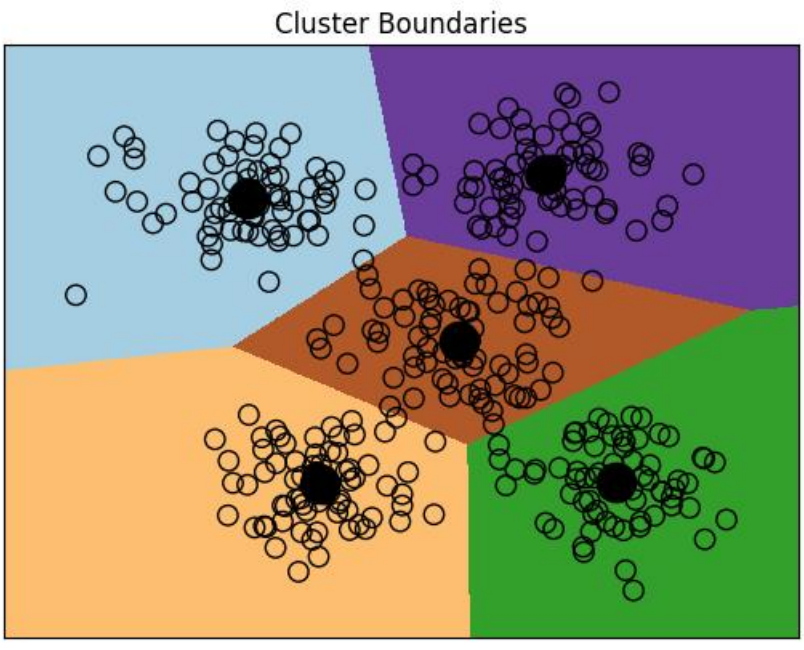


Рис.1.16 – Реакція програми на дію

Завдання 2.8

Дія1:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

iris = datasets.load_iris()
X = iris.data
y = iris.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
kmeans.fit(X_scaled)

cluster_labels = kmeans.labels_

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=cluster_labels, cmap='viridis', edgecolors='k', s=60)
plt.title('K-Means Clustering on Iris Dataset')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

Рис.1.17 – Код програми

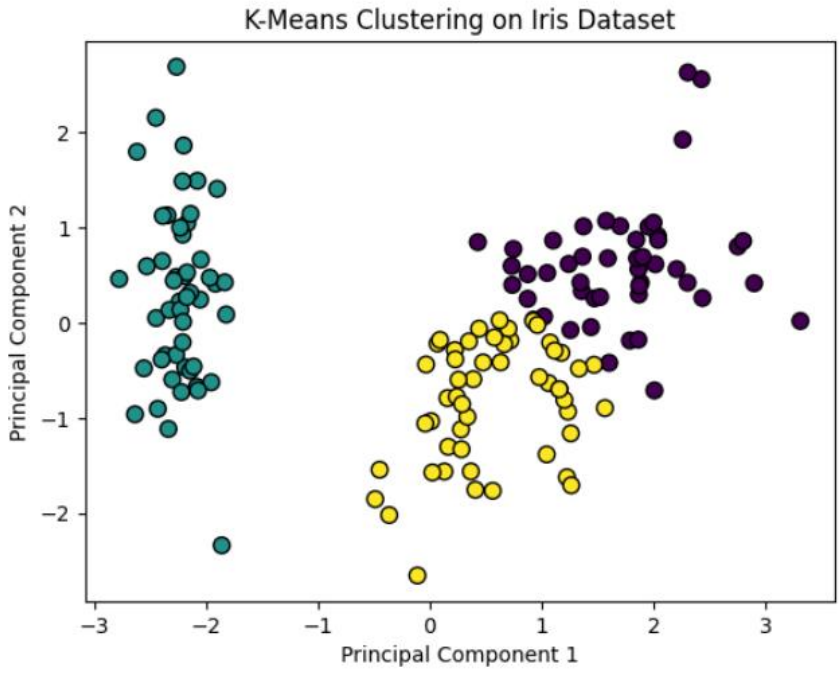


Рис.1.18 – Реакція програми на дію

Завдання 2.9

Дія1:

```
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_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("Number 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='black')
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o', markerfacecolor='black', markeredgecolor='black', markersize=15

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

Рис.1.19 – Код програми

```
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
```

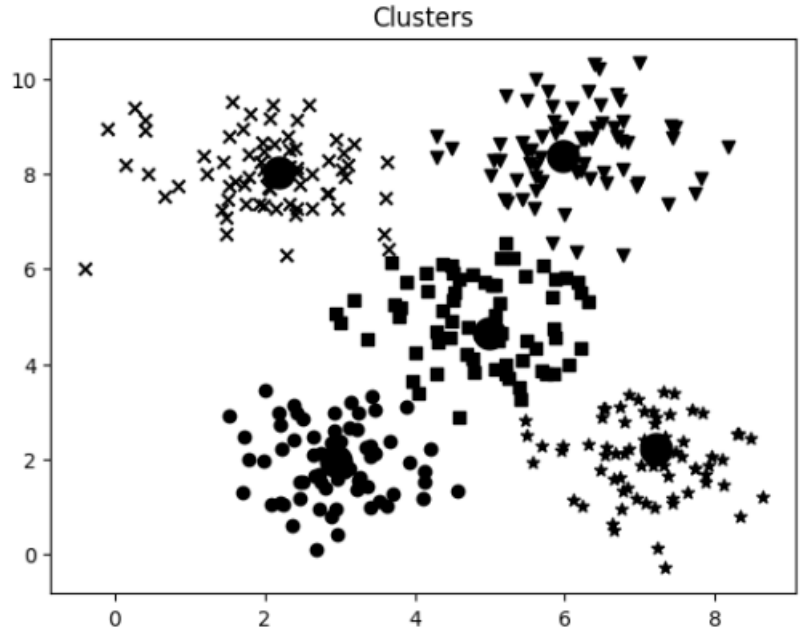


Рис.1.20 – Реакція програми на дію