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:

GitHub: https://github.com/ipz201svo/AI

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
# 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))
```

					ДУ «Житомирська політехніка».23.121.17.000 – Лр3		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Скаковський В.О.			Звіт з лабораторної роботи №3	Літ.	Арк. 1 / Аркушів 17
Перевір.		Голенко М.Ю.					
Керівник						ФІКТ Гр. ІПЗ-20-1	
Н. контр.							
Зав. каф.							

```
        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),
        )
```

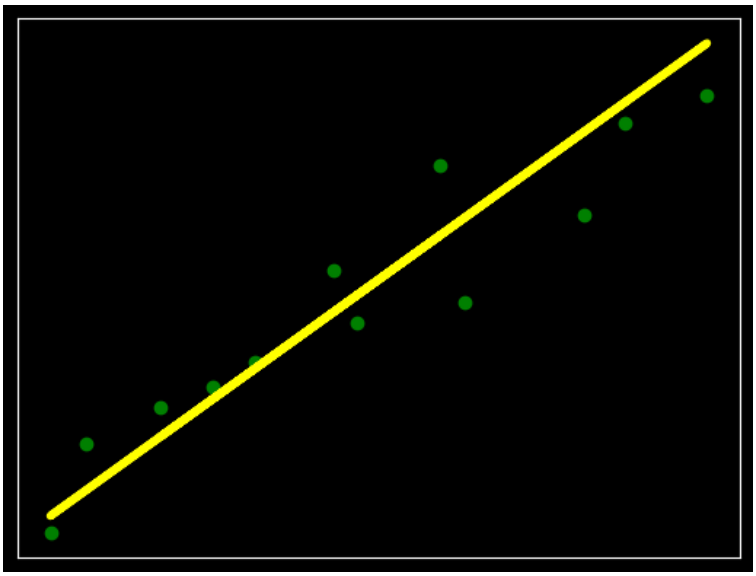

Рис. 1.1 – Результат виконання програми

```
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 – Результат виконання програми

Модель лінійної регресії натренована на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка становить 0.59, середня квадратична різниця дорівнює 0.49,медіана абсолютних помилок дорівнює 0.51, оцінка пояснені дисперсії становить 0.86, нова абсолютна помилка для моделі, відновленої з використанням збереженого файлу також дорівнює 0.59.

Завдання 2: Передбачення за допомогою регресії однієї змінної.

```
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="yellow", 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))
```

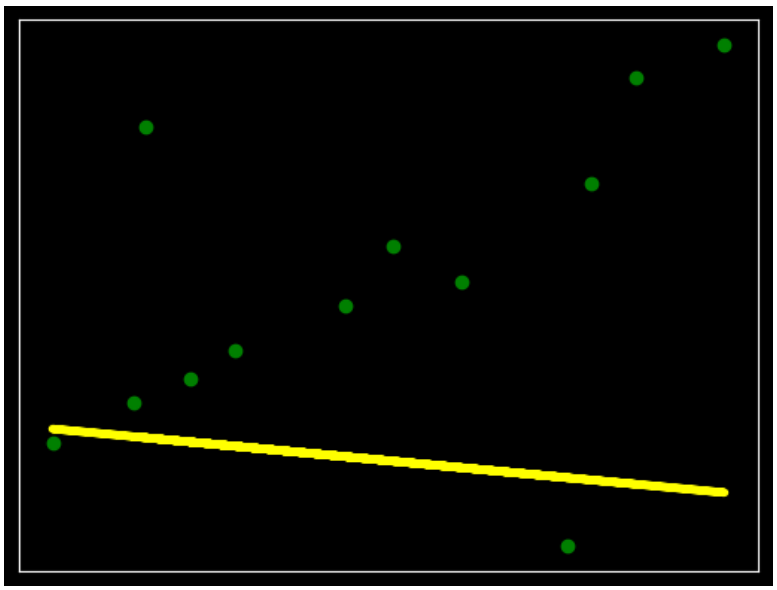

Рис. 2.1 – Результат виконання програми

```
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
```

Рис. 2.2 – Результат виконання програми

Модель лінійної регресії натренована на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка становить 2.42, середня квадратична різниця дорівнює 9.02 (значення вище ніж попереднє, що свідчить про великі відхилення у прогнозах), медіана абсолютних помилок дорівнює 2.14, оцінка поясненої дисперсії становить -0.15 (прогнози надто відмінні від фактичних значень).

Завдання 3: Створення багатовимірного регресора.

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
# 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
        linear = linear_model.LinearRegression()
        linear.fit(X_train, y_train)
        # Прогнозування результату
        y_test_pred = linear.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))
        # 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", linear.predict(datapoint))
        print("\nPolynomial regression:\n", poly_linear_model.predict(poly_datapoint))
```

```
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
```

Рис. 3.1 – Результат виконання програми

```
Linear regression:
 [36.05286276]

Polynomial regression:
 [41.45976677]
```

Рис. 3.2 – Результат виконання програми

Середня абсолютна помилка становить 3.58, середня квадратична різниця дорівнює 20.31(значення вище ніж попереднє, що свідчить про великі відхилення у прогнозах) ,медіана абсолютних помилок дорівнює 2.99, оцінка поясненої дисперсії становить 0.86(прогнози схожі до фактичних значень. Для поліноміальної регресії використовується поліном 10-го ступеня, для вхідного значення [7.75, 6.35, 5.56] , поліноміальна регресійна модель прогнозує значення 41.46, в той час як лінійна регресія прогнозує значення 36.05. Обидві моделі мають схожі значення показників, що свідчить про те, що обидві моделі надають приблизно однакові результати.

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("Linear regressor performance:")
print("Coef = ", regr.coef_)
print("Intercept = ", regr.intercept_)
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()
```

```
Linear regressor performance:
Coef =  [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Intercept =  154.3589285280134
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```

Рис. 4.1 – Результат виконання програми

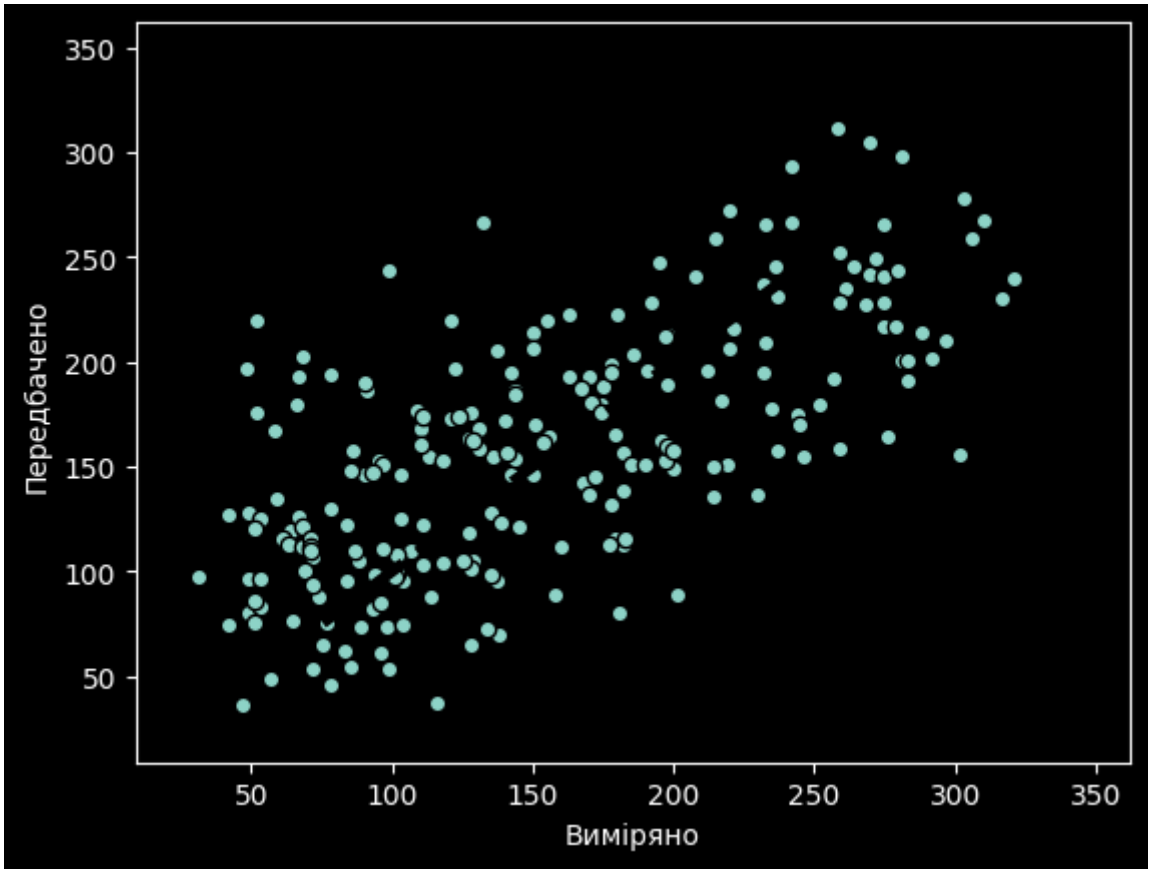

Рис. 4.1 – Результат виконання програми

Було виконано лінійну регресію для набору даних «Diabetes» та було отримано такі результати якості:

Коефіцієнти регресії представляються як масив чисел, вони вказують на вагу кожної ознаки [-20.4047621, -265.88518, 564.65086437, 325.56226865, 692.16120333, 395.55720874, 23.49659361, 116.36402337 843.94613929, 12.71856131].

Перетин рівний 154.36 і представляє відсоток, на який зміщується пряма регресії.

Оцінка R2 дорівнює 0.44 (модель пояснює близько 44% варіації в цільовій змінній).

Середня Абсолютна різниця становить 44.8.

Середня Квадратична Оцінка дорівнює 3075.33 (показує різницю між прогнозованим і фактичним значеннями).

Модель має обмежену ефективність, що підтверджує значення R2.

Завдання 5: Самостійна побудова регресії.

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
# Розбивка даних на навчальний та тестовий набори

X_train, X_test, y_train, ytest = train_test_split(X.reshape((-1, 1)), y,
test_size
= 0.5, random_state = 0)

lin_reg = linear_model.LinearRegression()
lin_reg.fit(X.reshape((-1, 1)), y)

poly_featueres = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_featueres.fit_transform(X.reshape((-1, 1)))
poly_reg = linear_model.LinearRegression()
poly_reg.fit(X_poly, y)

print('Linear regression:')
print('Intercept: ', lin_reg.intercept_)
print('Coefficients: ', lin_reg.coef_)

print('\nPolynomial regression:')
print('Intercept: ', poly_reg.intercept_)
print('Coefficients: ', poly_reg.coef_)

plt.scatter(X, y)
plt.plot(X, lin_reg.predict(X.reshape(-1, 1)), color='red', linewidth=2,
label='linear')
plt.plot(X, poly_reg.predict(X_poly), color='blue', linewidth=2,
label='polynomial')
plt.show()
```

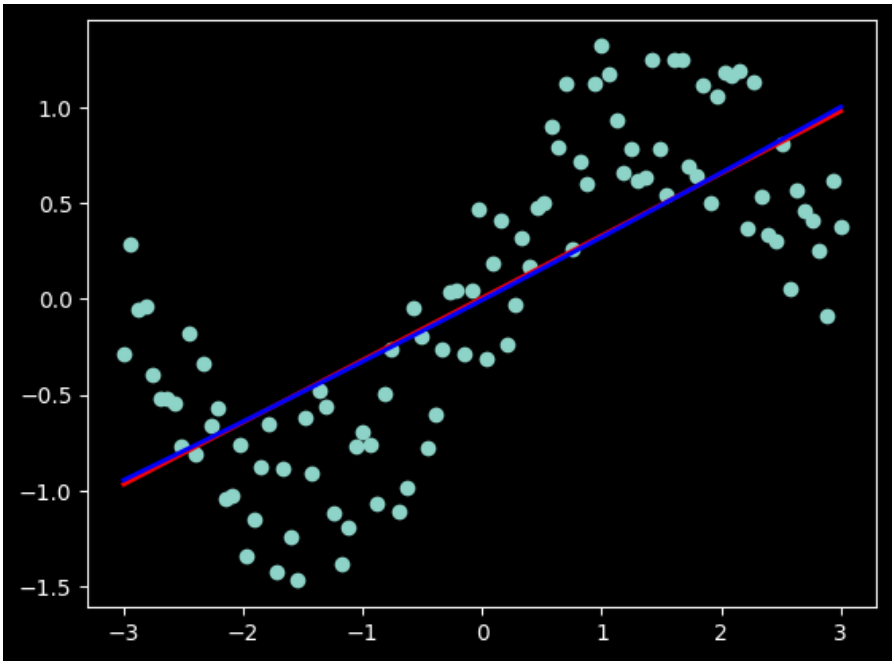

Рис. 5.1 – Результат виконання програми

```
Linear regression:
Intercept:  0.0064398786531946785
Coefficients:  [0.32460724]

Polynomial regression:
Intercept:  -0.005036954665643778
Coefficients:  [0.32460724 0.00374986]
```

Рис. 5.2 – Результат виконання програми

Модель у вигляді математичного рівняння

$$y = \sin(x) + \text{гаусів шум}$$

$$y = 0.32x^2 + 0.0037x$$

Завдання 6: Побудова кривих навчання.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

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

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')
    plt.show()

lin_reg = linear_model.LinearRegression()
plot_learning_curves(lin_reg, np.array(X).reshape(-1, 1), y)

polynomial_regression = Pipeline([
 ('poly_features', PolynomialFeatures(degree=2, include_bias=False)),
 ('lin_reg', linear_model.LinearRegression()),
])

plot_learning_curves(polynomial_regression, np.array(X).reshape(-1, 1), y)
```

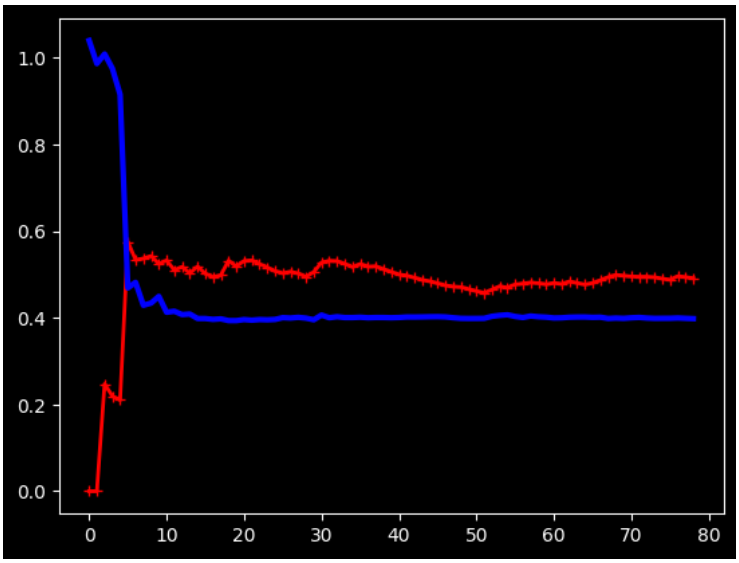

Рис. 6.1 – Результат виконання програми

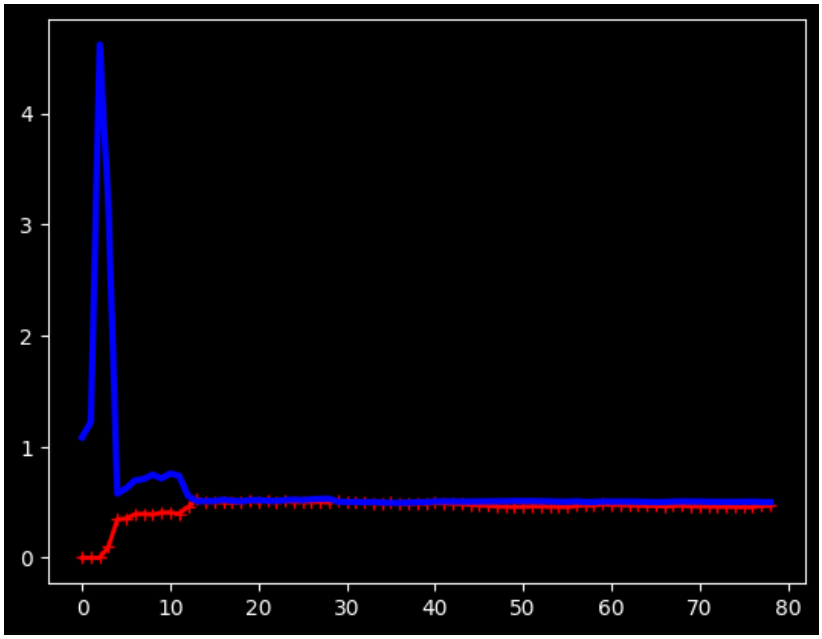

Рис. 6.2 – Результат виконання програми

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="yellow",
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 дані")
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")
        plt.title("Границі кластерів та центри")
        plt.xlim(x_min, x_max)
        plt.ylim(y_min, y_max)
        plt.xticks(())
        plt.yticks(())
        plt.show()
```

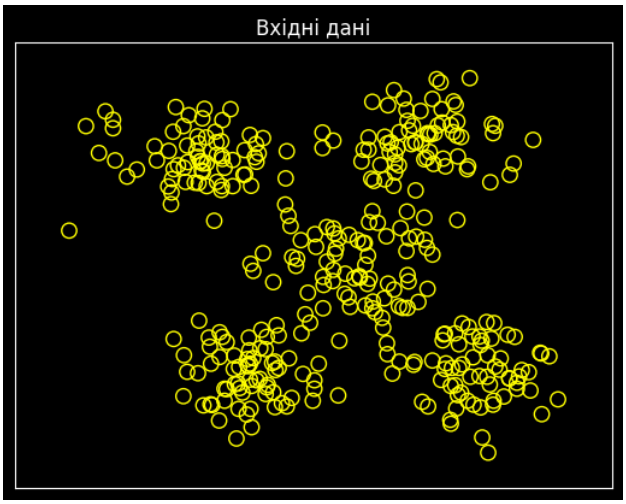

Рис. 7.1 – Результат виконання програми

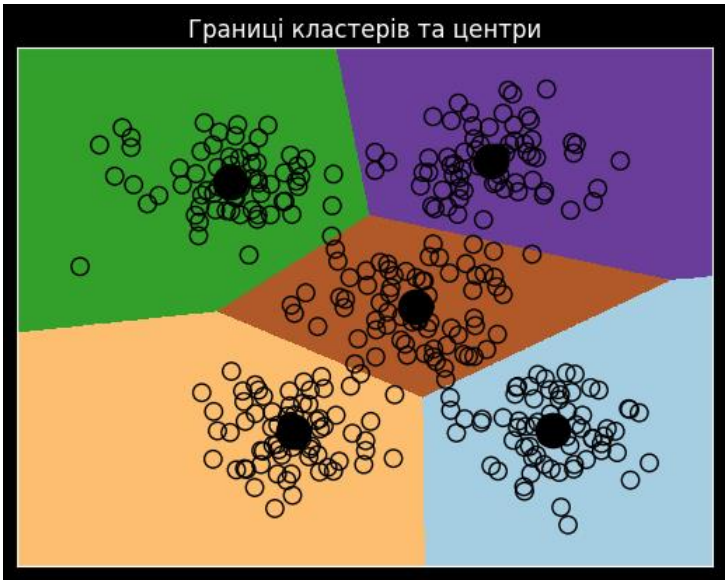

Рис. 7.2 – Результат виконання програми

Ми використали агоритм K-Means для кластеризації даних, було використано 5 кластерів.

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

```
from sklearn.svm import SVC
from sklearn.metrics import pairwise_distances_argmin
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import numpy as np
iris = load_iris()
X = iris["data"]
y = iris["target"]

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",
)

# Створення моделі
kmeans = KMeans(n_clusters = 5)
# Тренування моделі
kmeans.fit(X)
# Створення прогнозу
y_kmeans = kmeans.predict(X)
# Відображення значень
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)
# Функція для знаходження 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


centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")
centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")
```

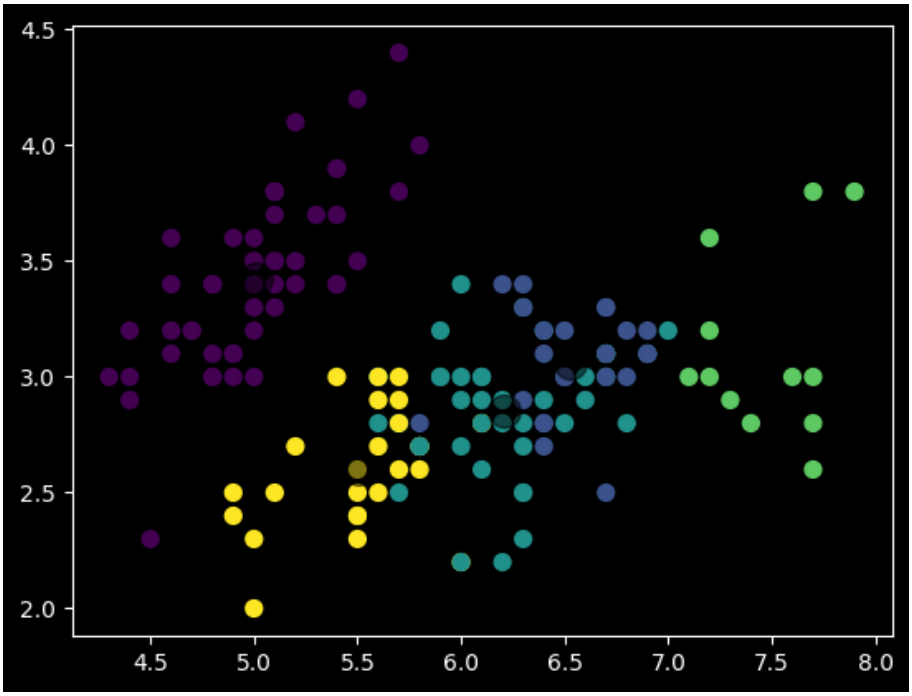

Рис. 8.1 – Результат виконання програми

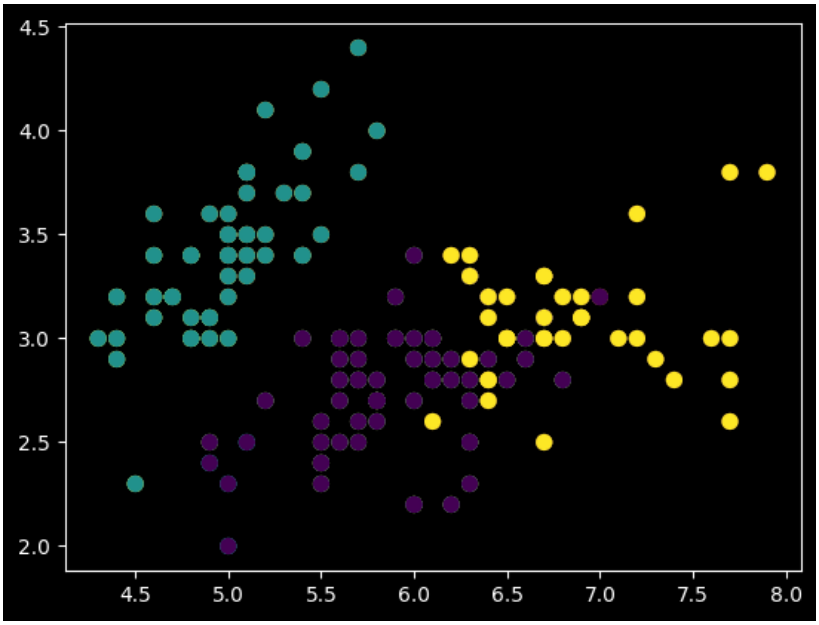

Рис. 8.2 – Результат виконання програми

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

```
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=0.1, n_samples=len(X))
meanshift_model = MeanShift(bandwidth=bandwidth, 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="black")
            cluster_center = cluster_centers[i]
            plt.plot(
                cluster_center[0],
                cluster_center[1],
                marker="o",
                markerfacecolor="white",
                markeredgecolor="yellow",
                markersize=15,
            )

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

```
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. 9.1 – Результат виконання програми

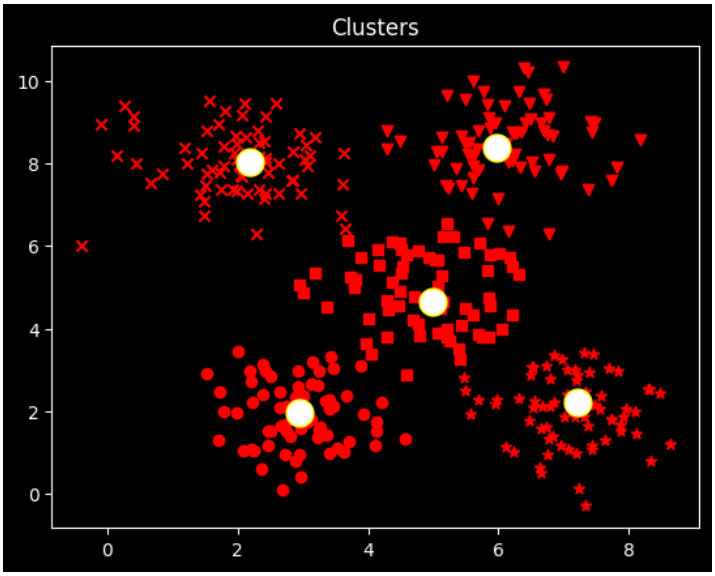

Рис. 9.2 – Результат виконання програми

Алгоритм Mean Shift успішно використовується для кластеризації даних. В результаті було отримано п'ять кластерів, і їх центри були виведені на екран та відображені на графіку.

Завдання 10: Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.

```
import datetime
import json
import numpy as np
from sklearn import covariance, cluster
import yfinance as yf

# Вхідний файл із символічними позначеннями компаній
input_file = "company_symbol_mapping.json"

# Завантаження прив'язок символів компаній до їх повних назв
with open(input_file, "r") as f:
    company_symbols_map = json.loads(f.read())

symbols, names = np.array(list(company_symbols_map.items())).T

# Визначення архівних даних котирувань
start_date = "2003-07-03"
end_date = "2007-05-04"

# Завантаження архівних даних котирувань
quotes = []
valid_symbols = []
for symbol in symbols:
    try:
        data = yf.download(symbol, start=start_date, end=end_date)
        if not data.empty:
            quotes.append(data)
            valid_symbols.append(symbol)
    except Exception as e:
        print(f"Failed to download data for {symbol}: {e}")

# Перевірка чи є валідні дані
if not quotes:
    print(
        "No valid data available for any symbol. Check your symbol mapping and
data availability."
    )
else:
    # Оновлення символів на дійсні
    symbols = valid_symbols

    # Вилучення котирувань, що відповідають відкриттю та закриттю біржі
    opening_quotes = np.array([quote["Open"].values for quote in quotes]).T
    closing_quotes = np.array([quote["Close"].values for quote in quotes]).T

    # Обчислення різниці між двома видами котирувань
    quotes_diff = closing_quotes - opening_quotes

    # Нормалізація даних
    X = quotes_diff.copy()
```

```
            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()

            # Виведення результатів
            print("\nClustering of stocks based on difference in opening and closing
quotes:\n")
            for i in range(num_labels + 1):
                cluster_indices = np.where(labels == i)[0]
                cluster_names = names[cluster_indices]
                if len(cluster_names) > 0:
                    print("Cluster", i + 1, "==>", ", ".join(cluster_names))
```

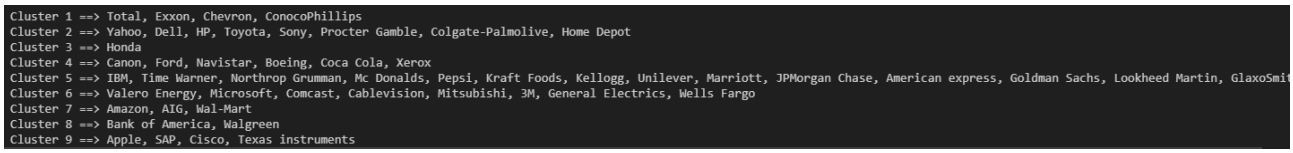

Рис. 10.1 – Результат виконання програми

Висновок: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та навчився їх порівнювати