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)}")
```

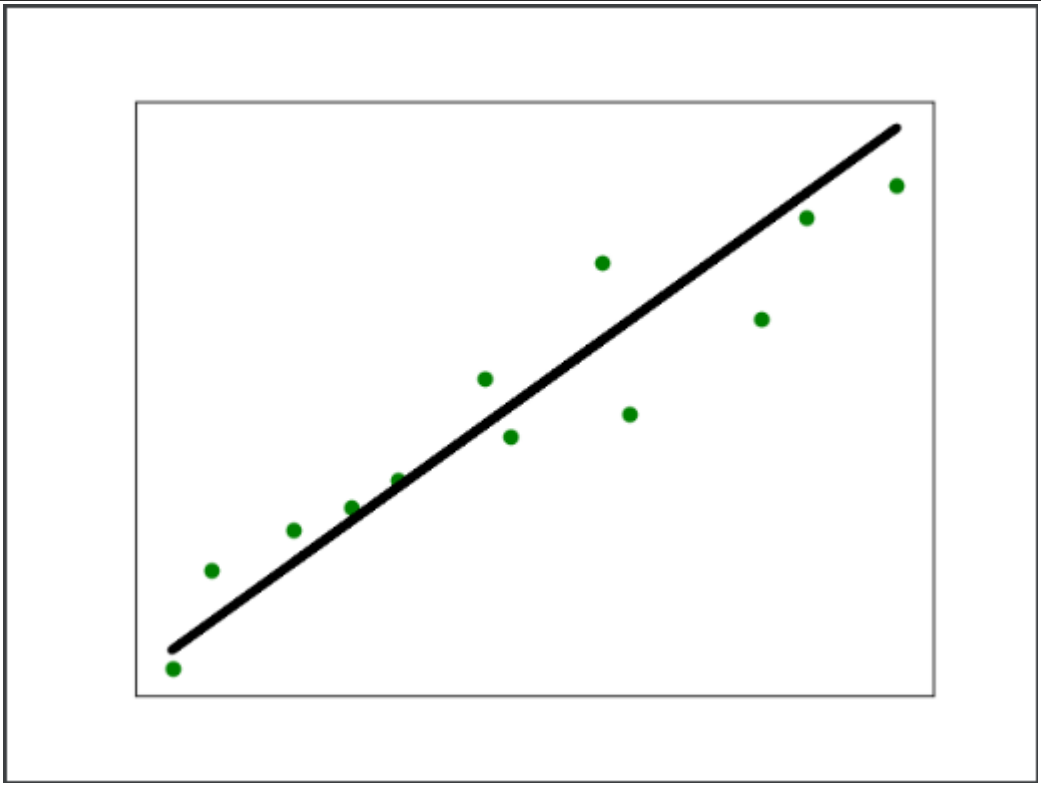

Рис.1 – Результат виконання лінійної регресії.

```
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.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)}")
```

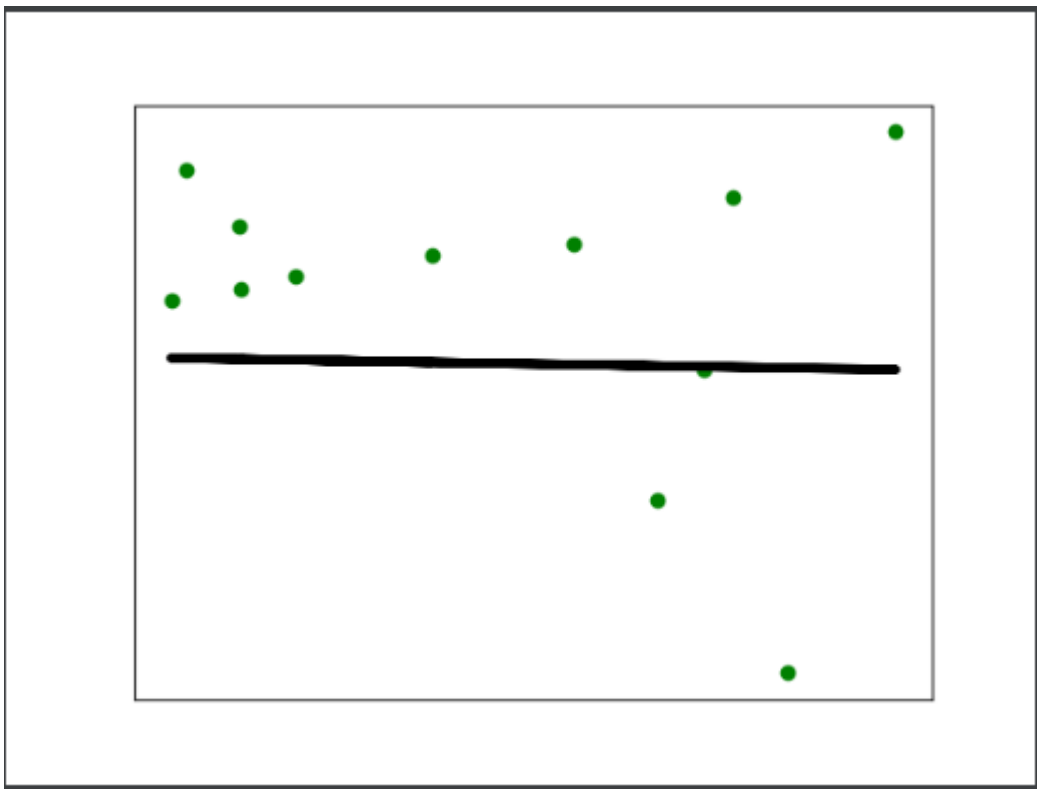

Рис.3 – Результат виконання лінійної регресії за власними даними.

```
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
```

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)}")
```

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

```
Linear regression:
[36.05286276]
Polynomial regression:
[41.46007151]
```

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, ran-
dom_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()
```

```
Mean squared error = 3736.39
Mean absolute error = 49.51
R2 score = 0.32
Regression coefficient = 1057.06
Regression intercept = 154.13
```

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

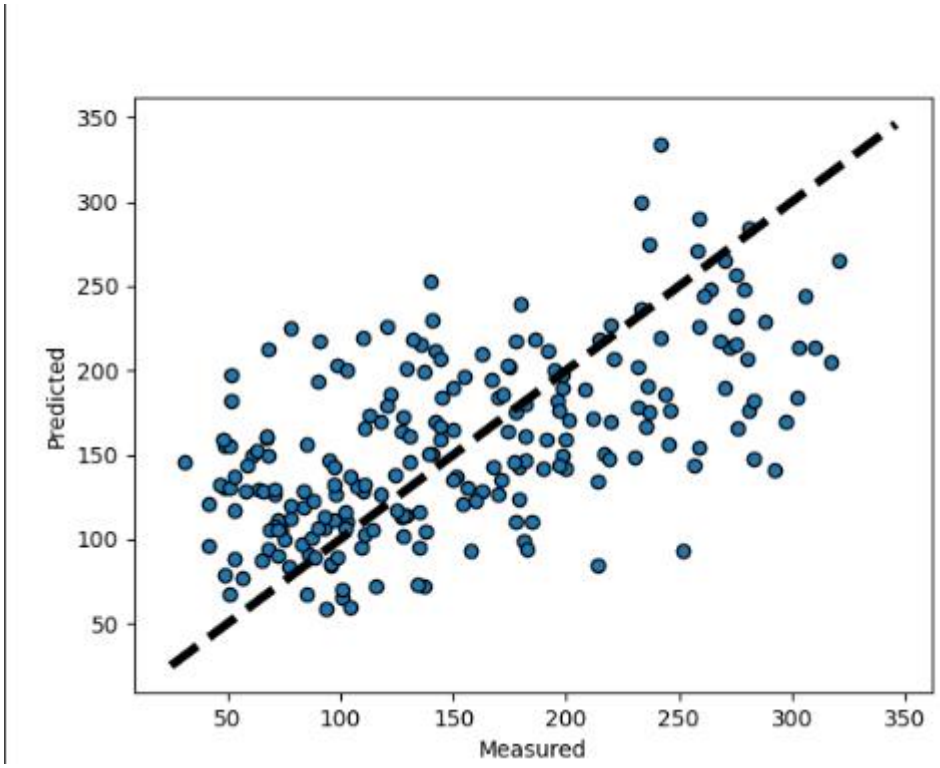

Рис.8 – Графік результату лінійної регресії даних про діабет.

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, ran-
dom_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()
```

```
Mean absolute error = 1.28
Mean squared error = 2.35
Regression coefficient = 0.03
Regression intercept = 3.04
R2 score = -0.01
```

Рис.9 – Характеристика лінійної регресії випадкових даних.

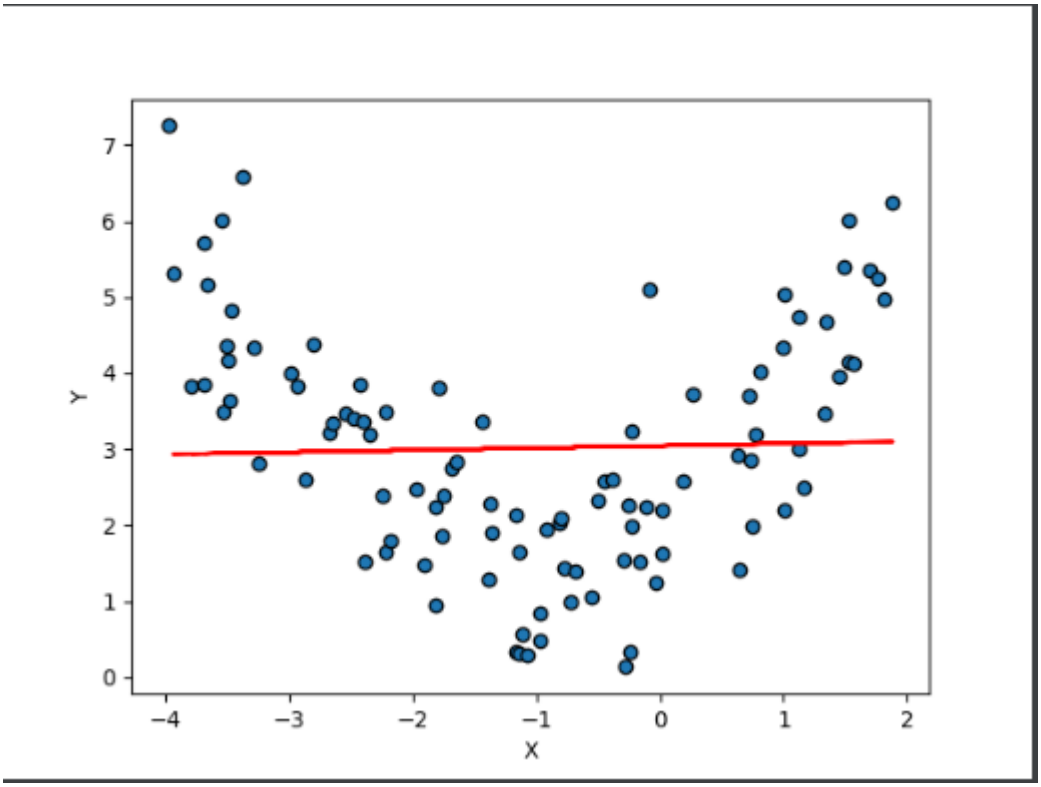

Рис.10 – Лінійна регресія випадкових даних.

```
Mean absolute error = 0.76
Mean squared error = 0.86
Regression coefficient = 0.0
Regression intercept = 2.14
R2 score = 0.65
```

Рис.11 – Характеристика поліноміальної регресії випадкових даних.

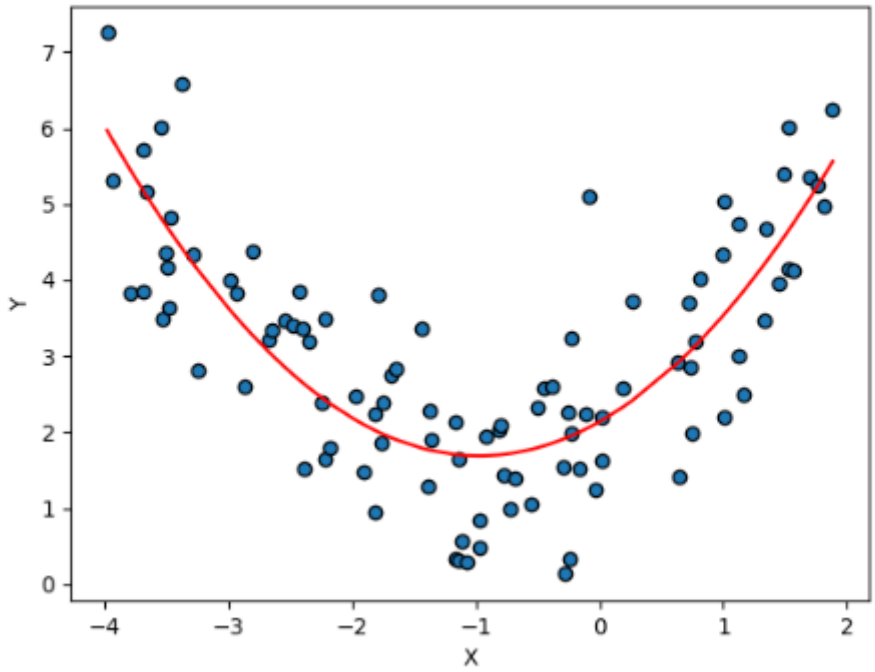

Рис.12 – Поліноміальна регресія випадкових даних.

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)
```

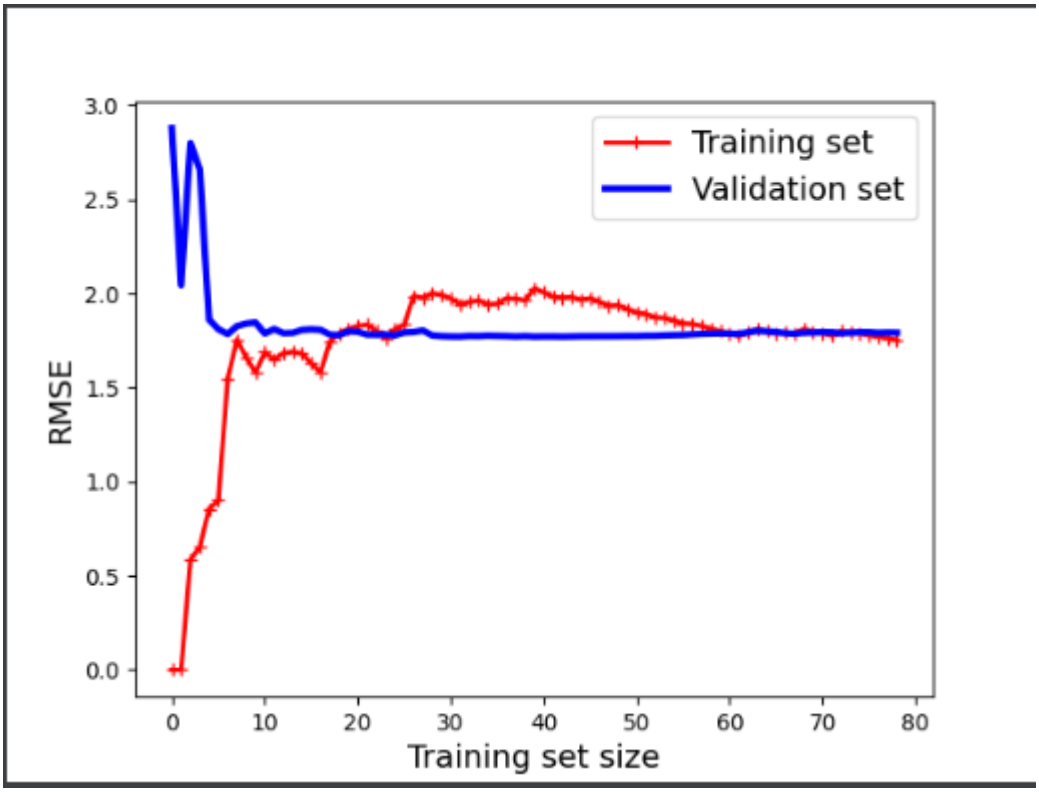

Рис.13 – Криві навчання для лінійної моделі.

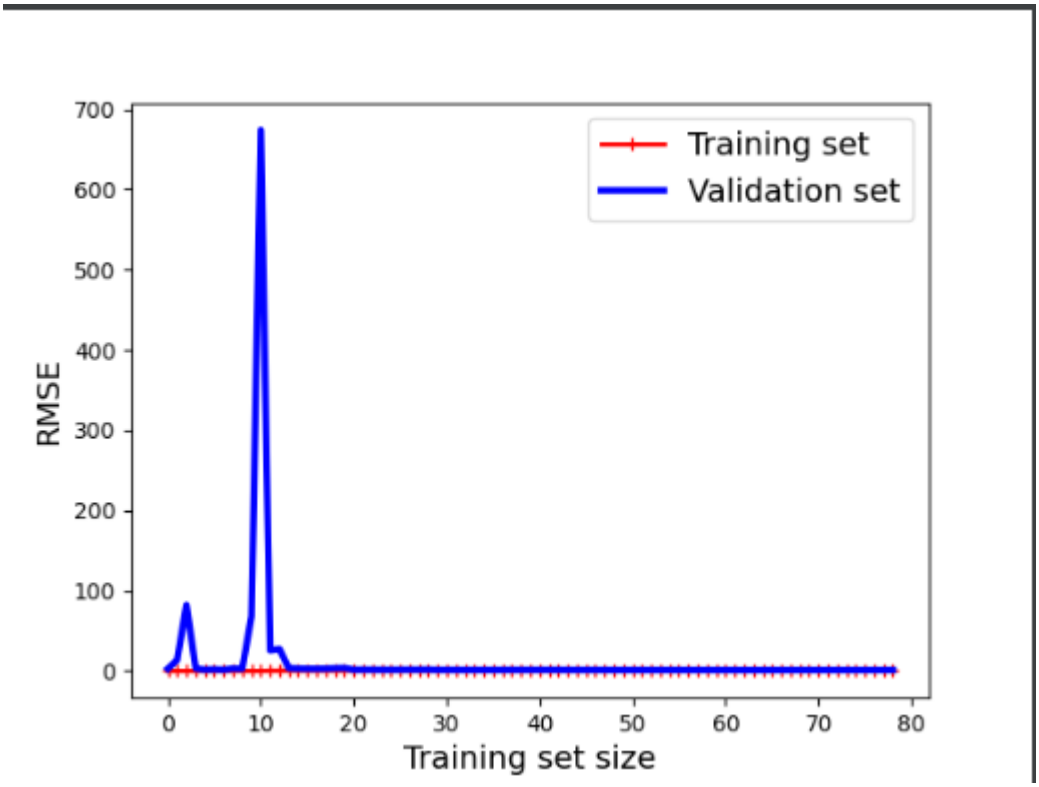

Рис.14 – Криві навчання для поліноміальної моделі 10го ступеня.

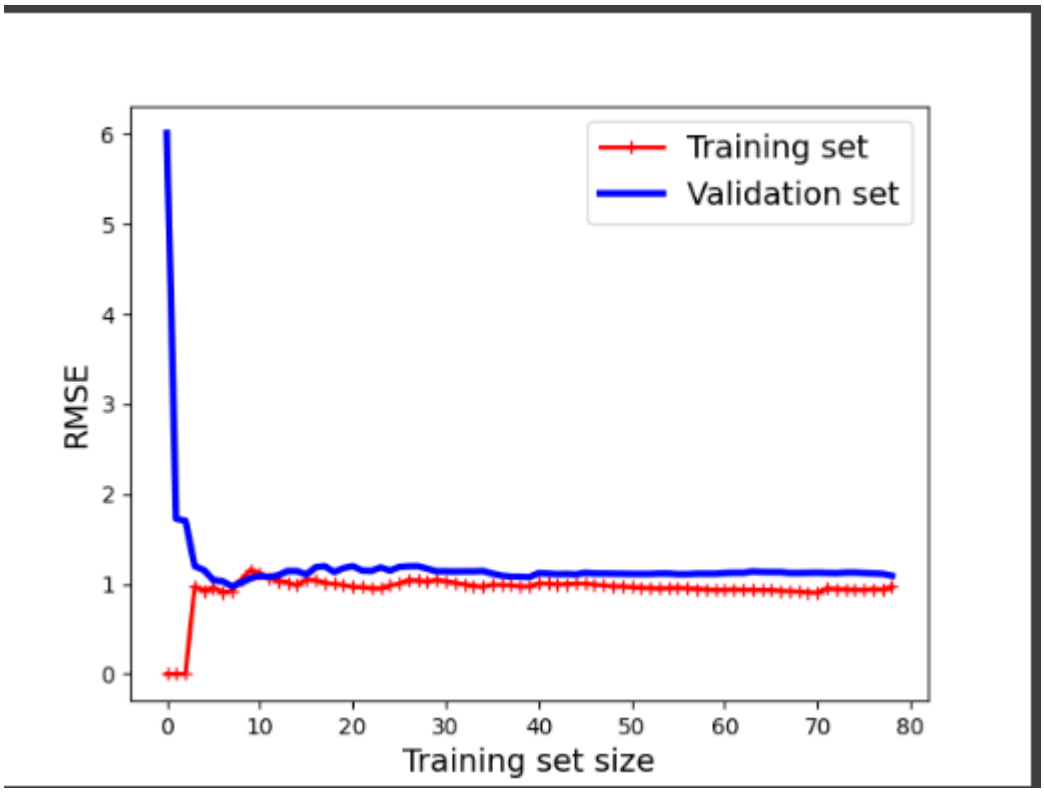

Рис.15 – Криві навчання для поліноміальної моделі 2го ступеня.

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_val-
ues.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()
```

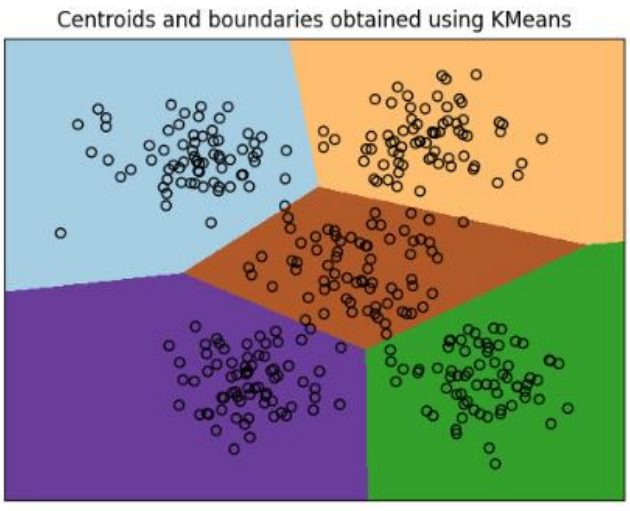

Рис.16 – Відображення кластеризованих даних методом К-середніх.

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

Лістинг коду файлу LR_3_task_8.py:

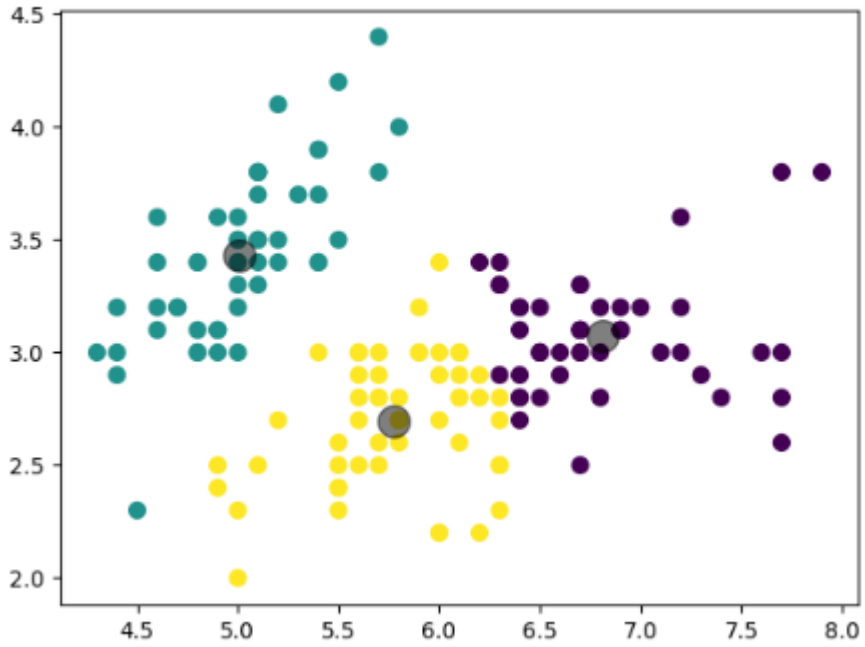

Рис.17 – Ручна кластеризація даних по ірисам.

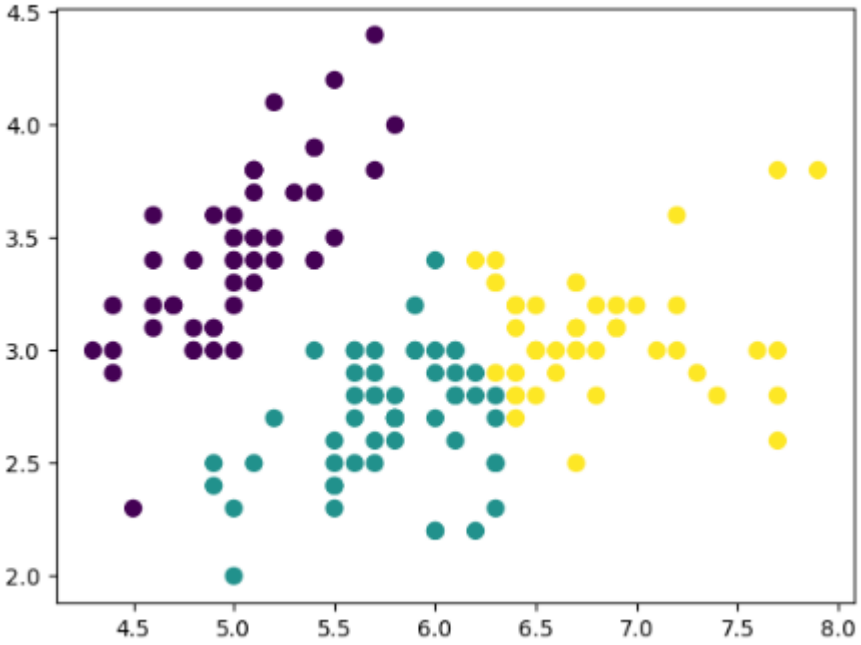

Рис.18 – Кластеризація з використанням створеної функції, випадкове зерно – 2.

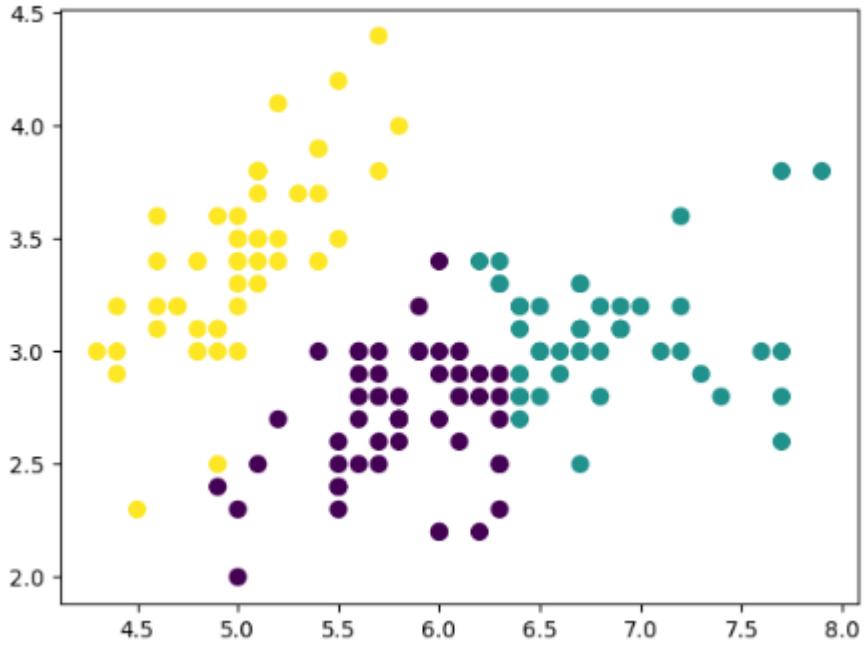

Рис.19 – Кластеризація з використанням створеної функції, випадкове зерно – 0.

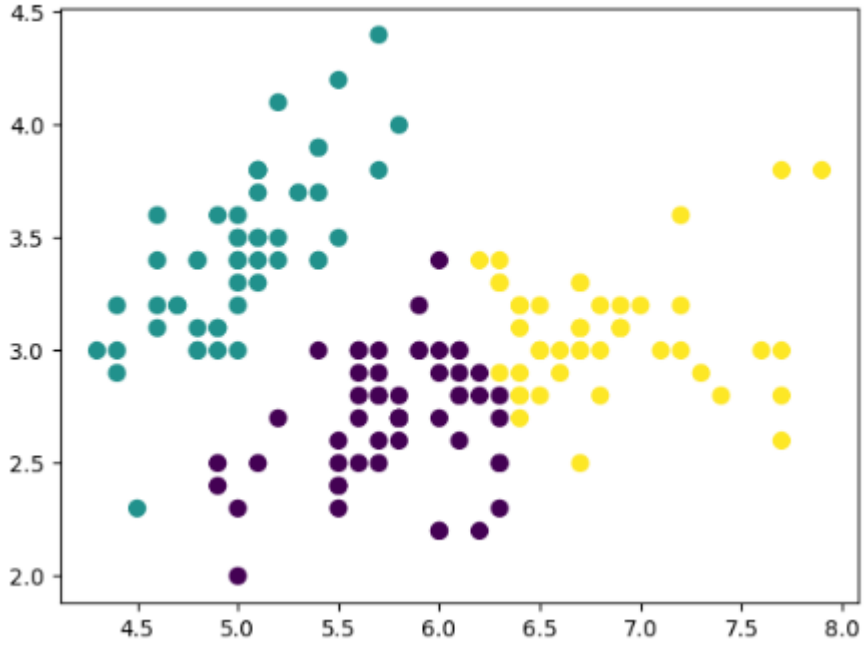

Рис.20 – Кластеризація швидким викликом кластеризатора.

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', markerface-
color='k', markeredgecolor='k', markersize=15)
plt.title(f'Estimated number of clusters: {len(cluster_centers)}')
plt.show()
```

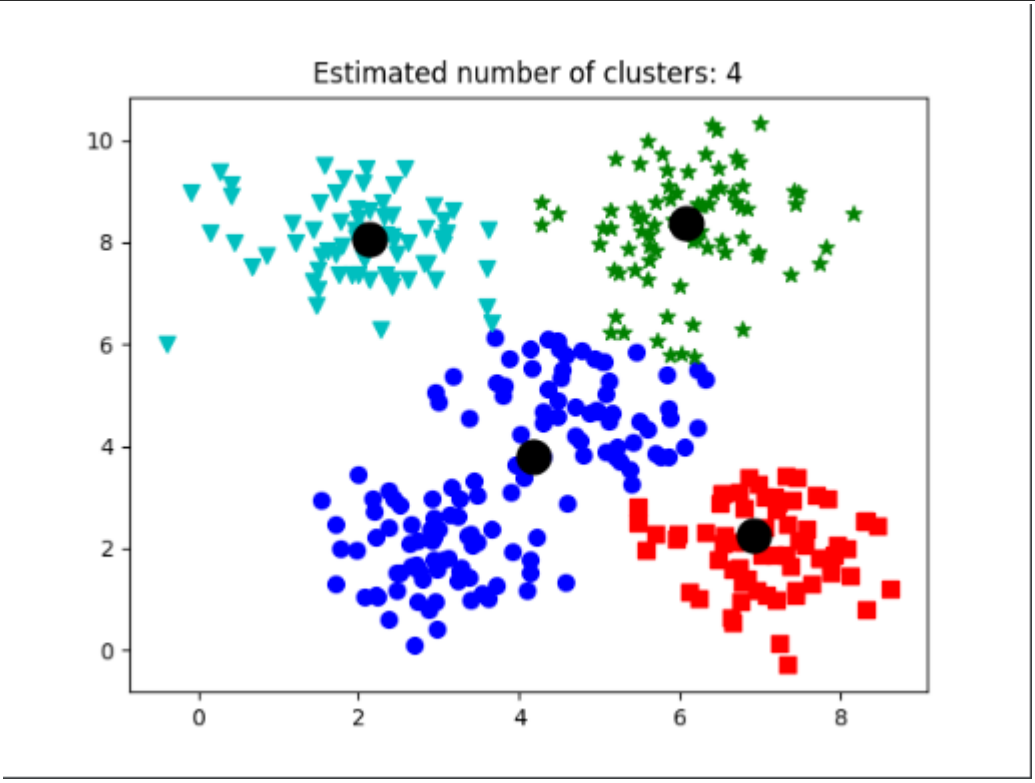

Рис.21 – Відображення кластеризованих даних методом зсуву середнього

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/Xatiko17/AI_Labs