ЛАБОРАТОРНА РОБОТА № 2
ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

Хід роботи:

Завдання №1: Класифікація за допомогою машин опорних векторів (SVM).

age (вік) – вік персони

workclass (клас роботи) - Приватний, Федеральний уряд, Місцевий уряд, Уряд штату, Без оплати, Ніколи не працював.

fnlwgt (фінальна вага) - кількість людей.

education (освіта) - Бакалавр, коледж, Випускник, Проф-школа, Асоційоване-акдем, Асоційоване-вок, Магістр, Доктор, Дошкільна.

education-num (номер освіти) - Числове представлення рівня освіти

marital-status (сімейний стан) – Одружений (чоловік/дружина), Розлучений, Ніколи не одружений/заміжня, Розділений, Вдівець/Вдова.

occupation (зайнятість) – Технічна підтримка, Інші-послуги, Продажі, Керівник-менеджер, Проф-спеціальність, Захисна служба, Збройні Сили.

relationship (відносини) - Дружина, Дитина, Чоловік, Не в родині, Родич, Неодружений/незаміжня.

race (раса) - Біла, Азіатська, Чорна.

sex (стать) - Жіноча, Чоловіча.

capital-gain (приріст капіталу) - Зафіксований приріст капіталу (дохід).

capital-loss (втрата капіталу) - Зафіксовані втрати капіталу (збитки).

hours-per-week (годин-на-тиждень) - Кількість робочих годин на тиждень.

native-country (рідна країна) - Країна походження.

Числові Ознаки (6): age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week.

Категоріальні Ознаки (8): workclass, education, marital-status, occupation, relationship, race, sex, native-country.

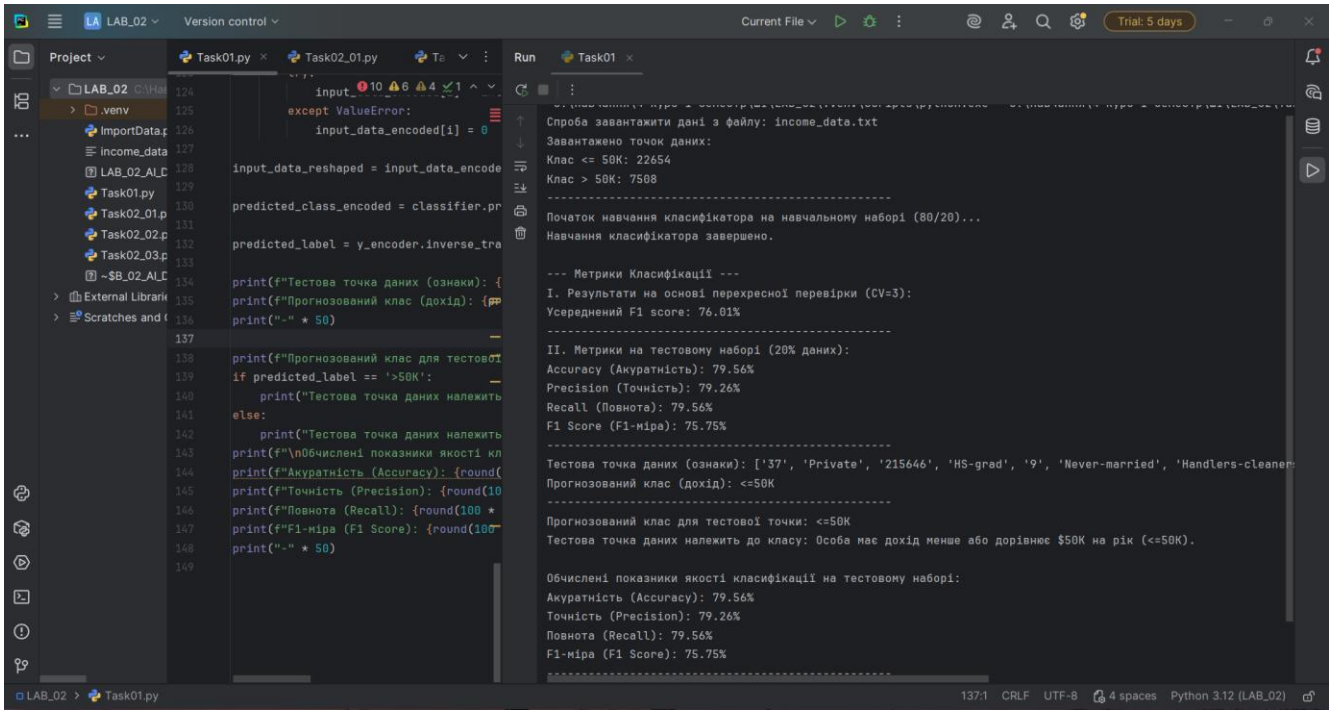


Рис.1.1 Виконання завдання

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
import sys

input_file = 'income_data.txt'

X_raw = []
y_labels = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

print(f"Спроба завантажити дані з файлу: {input_file}")
try:
    with open(input_file, 'r') as f:
        for line in f.readlines():
            if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
                break

            if '?' in line:
                continue


            data = line.strip().split(', ')

            if not data or len(data) < 15:
                continue

            label = data[-1]
```

```
            if label == '<=50K' and count_class1 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class1 += 1

            elif label == '>50K' and count_class2 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class2 += 1

except FileNotFoundError:
    print(f"\nПОМИЛКА: Файл '{input_file}' не знайдено.")
    print("Переконайтеся, що файл 'income_data.txt' існує у кореневому каталозі.")
    sys.exit(1)

print(f"Завантажено точок даних:\nКлас <= 50K: {count_class1}\nКлас > 50K:
{count_class2}")


X = np.array(X_raw)
y_labels = np.array(y_labels)

# workclass (1), education (3), marital-status (5), occupation (6),
# relationship (7), race (8), sex (9), native-country (13)
categorical_feature_indices = [1, 3, 5, 6, 7, 8, 9, 13]

label_encoders = {}
num_features = X.shape[1]

X_encoded = np.empty(X.shape, dtype=float)


for i in range(num_features):
    if i in categorical_feature_indices:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders[i] = le
    else:
        X_encoded[:, i] = X[:, i].astype(float)

X = X_encoded.astype(int)
y_encoder = preprocessing.LabelEncoder()
y = y_encoder.fit_transform(y_labels)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)


classifier = OneVsOneClassifier(LinearSVC(random_state=0, max_iter=10000))
print("-" * 50)
print("Початок навчання класифікатора на навчальному наборі (80/20)...")
classifier.fit(X_train, y_train)
print("Навчання класифікатора завершено.")

y_test_pred = classifier.predict(X_test)

f1_scores_cv = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
f1_mean_cv = f1_scores_cv.mean()

print("\n--- Метрики Класифікації ---")
print("I. Результати на основі перехресної перевірки (CV=3):")
```

```
print(f"Усереднений F1 score: {round(100 * f1_mean_cv, 2)}%")
print("-" * 50)

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
f1_test = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)

print("II. Метрики на тестовому наборі (20% даних):")
print(f"Accuracy (Акуратність): {round(100 * accuracy, 2)}%")
print(f"Precision (Точність): {round(100 * precision, 2)}%")
print(f"Recall (Повнота): {round(100 * recall, 2)}%")
print(f"F1 Score (F1-міра): {round(100 * f1_test, 2)}%")
print("-" * 50)

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White',
             'Male', '0', '0', '40', 'United-States']

input_data_encoded = np.empty(len(input_data), dtype=int)

for i, item in enumerate(input_data):
    if i in categorical_feature_indices:
        try:
            input_data_encoded[i] = label_encoders[i].transform([item])[0]
        except ValueError:
            input_data_encoded[i] = 0
    else:
        try:
            input_data_encoded[i] = int(item)
        except ValueError:
            input_data_encoded[i] = 0

input_data_reshaped = input_data_encoded.reshape(1, -1)

predicted_class_encoded = classifier.predict(input_data_reshaped)[0]

predicted_label = y_encoder.inverse_transform([predicted_class_encoded])[0]

print(f"Тестова точка даних (ознаки): {input_data}")
print(f"Прогнозований клас (дохід): {predicted_label}")
print("-" * 50)

print(f"Прогнозований клас для тестової точки: {predicted_label}")
if predicted_label == '>50K':
    print("Тестова точка даних належить до класу: Особа має дохід більше 50K на
рік (>50K).")
else:
    print("Тестова точка даних належить до класу: Особа має дохід менше або дорів-
нює 50K на рік (<=50K).")
print(f"\nОбчислені показники якості класифікації на тестовому наборі:")
print(f"Акуратність (Accuracy): {round(100 * accuracy, 2)}%")
print(f"Точність (Precision): {round(100 * precision, 2)}%")
print(f"Повнота (Recall): {round(100 * recall, 2)}%")
print(f"F1-міра (F1 Score): {round(100 * f1_test, 2)}%")
print("-" * 50)
```

Завдання №2: У попередньому завданні ми побачили, як простий алгоритм SVM LinearSVC може бути використаний для знаходження межі рішення для лінійних даних. Однак у разі нелінійно розділених даних, пряма лінія не може бути використана як межа прийняття рішення. Натомість використовується модифікована версія SVM, звана Kernel SVM.

В основному, ядро SVM проектує дані нижніх вимірювань, що нелінійно розділяються, на такі, що лінійно розділяються більш високих вимірювань таким чином, що точки даних, що належать до різних класів, розподіляються за різними вимірами. В цьому є закладена складна математика, але вам не потрібно турбуватися про це, щоб використовувати SVM. Ми можемо просто використовувати бібліотеку Scikit-Learn Python для реалізації та використання SVM ядра.

Реалізація SVM ядра за допомогою Scikit-Learn аналогічна до простого SVM.

Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM. з поліноміальним ядром; з гаусовим ядром; з сигмоїдальним ядром. Для кожного виду класифікатора отримайте та запишіть у звіт показники якості алгоритму класифікації.

Через дуже довгу прогрузку, довелося змешити максимальну к-сть точок до 5000.

Дослідіження нелінійного класифікатора SVM з поліноміальним ядром

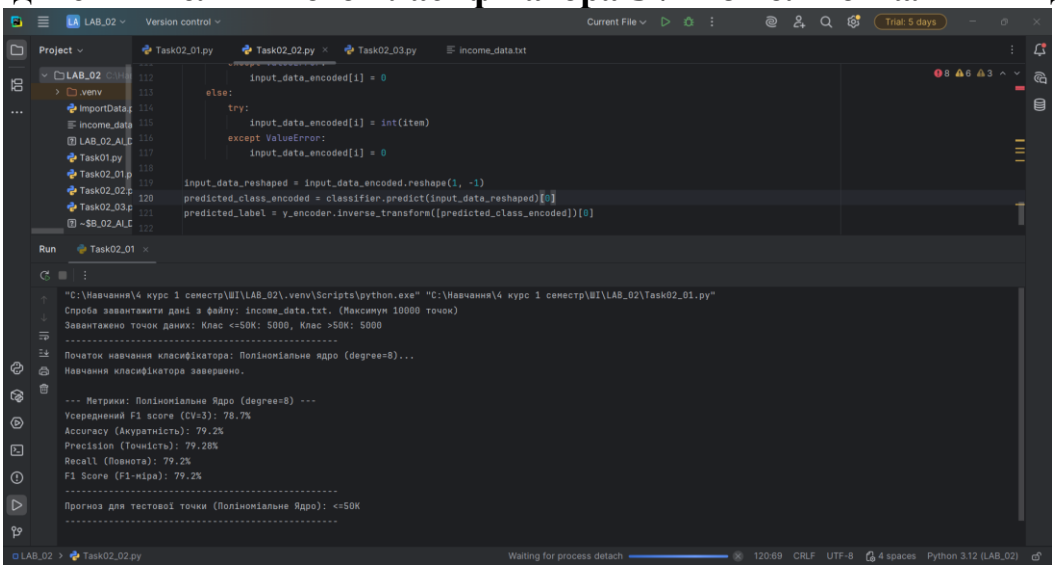


Рис.2.1 Виконання завдання

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
import sys

input_file = 'income_data.txt'

X_raw = []
y_labels = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

print(f"Спроба завантажити дані з файлу: {input_file}")

try:
    with open(input_file, 'r') as f:
        for line in f.readlines():
            if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
                break
            if '?' in line:
                continue
            data = line.strip().split(', ')
            if not data or len(data) < 15:
                continue

            label = data[-1]

            if label == '<=50K' and count_class1 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class1 += 1
            elif label == '>50K' and count_class2 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class2 += 1

except FileNotFoundError:
    print(f"\nПОМИЛКА: Файл '{input_file}' не знайдено.")
    sys.exit(1)

print(f"Завантажено точок даних: Клас <=50K: {count_class1}, Клас >50K:
{count_class2}")

X = np.array(X_raw)
y_labels = np.array(y_labels)

categorical_feature_indices = [1, 3, 5, 6, 7, 8, 9, 13]
label_encoders = {}
num_features = X.shape[1]
X_encoded = np.empty(X.shape, dtype=float)

for i in range(num_features):
    if i in categorical_feature_indices:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders[i] = le
```

```
    else:
        X_encoded[:, i] = X[:, i].astype(float)

X = X_encoded.astype(int)
y_encoder = preprocessing.LabelEncoder()
y = y_encoder.fit_transform(y_labels)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = SVC(kernel='poly', degree=8, random_state=0)
print("-" * 50)
print("Початок навчання класифікатора: Поліноміальне ядро (degree=8)...")
classifier.fit(X_train, y_train)
print("Навчання класифікатора завершено.")

y_test_pred = classifier.predict(X_test)

f1_scores_cv = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
f1_mean_cv = f1_scores_cv.mean()

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
f1_test = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)

print("\n--- Метрики: Поліноміальне Ядро (degree=8) ---")
print(f"Усереднений F1 score (CV=3): {round(100 * f1_mean_cv, 2)}%")
print(f"Accuracy (Акуратність): {round(100 * accuracy, 2)}%")
print(f"Precision (Точність): {round(100 * precision, 2)}%")
print(f"Recall (Повнота): {round(100 * recall, 2)}%")
print(f"F1 Score (F1-міра): {round(100 * f1_test, 2)}%")
print("-" * 50)

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White',
              'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.empty(len(input_data), dtype=int)

for i, item in enumerate(input_data):
    if i in categorical_feature_indices:
        try:
            input_data_encoded[i] = label_encoders[i].transform([item])[0]
        except ValueError:
            input_data_encoded[i] = 0
    else:
        try:
            input_data_encoded[i] = int(item)
        except ValueError:
            input_data_encoded[i] = 0

input_data_reshaped = input_data_encoded.reshape(1, -1)
predicted_class_encoded = classifier.predict(input_data_reshaped)[0]
predicted_label = y_encoder.inverse_transform([predicted_class_encoded])[0]

print(f"Прогноз для тестової точки (Поліноміальне Ядро): {predicted_label}")
print("-" * 50)
```

Дослідіження нелінійного класифікатора SVM з гаусовим ядром

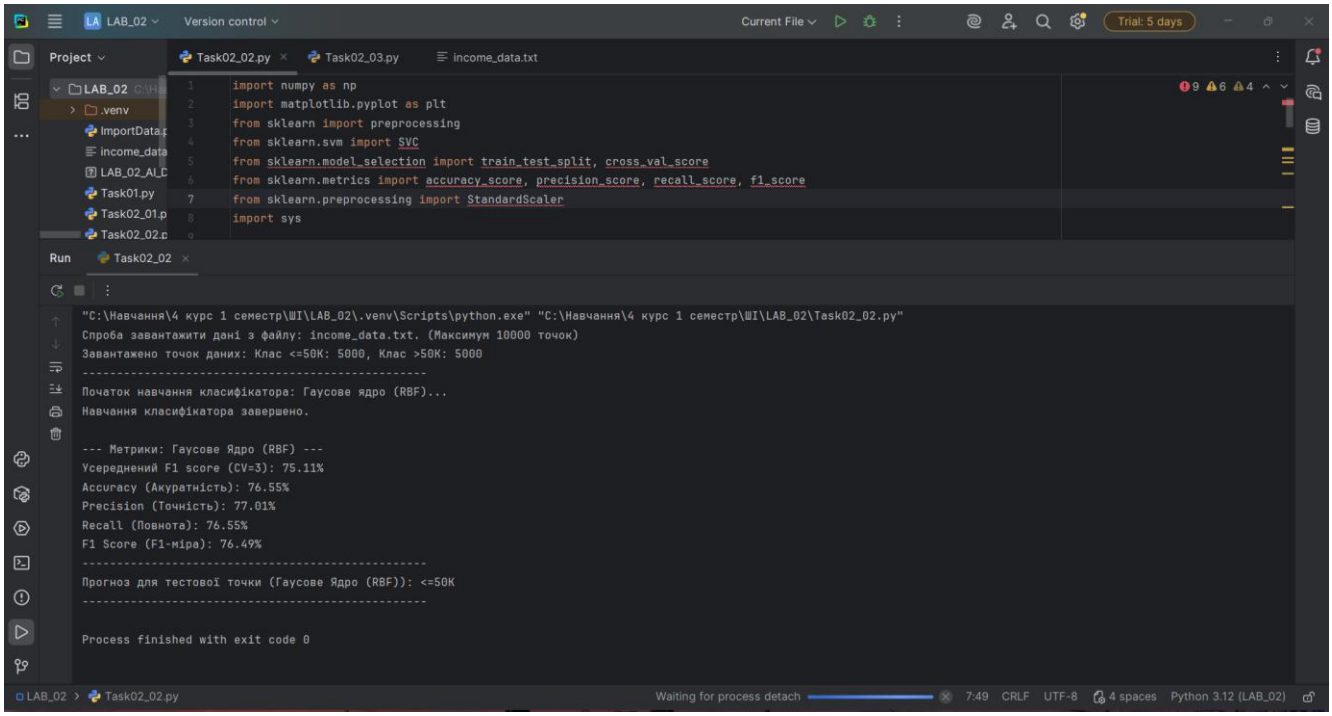


Рис.2.2 Виконання завдання

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.preprocessing import StandardScaler
import sys

input_file = 'income_data.txt'
X_raw = []
y_labels = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

print(f"Спроба завантажити дані з файлу: {input_file}. (Максимум {2 *
max_datapoints} точок)")

try:
    with open(input_file, 'r') as f:
        for line in f.readlines():
            if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
                break
            if '?' in line:
                continue
            data = line.strip().split(', ')
            if not data or len(data) < 15:
                continue
```

```
            label = data[-1]

            if label == '<=50K' and count_class1 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class1 += 1
            elif label == '>50K' and count_class2 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class2 += 1

except FileNotFoundError:
    print(f"\nПОМИЛКА: Файл '{input_file}' не знайдено.")
    sys.exit(1)

print(f"Завантажено точок даних: Клас <=50K: {count_class1}, Клас >50K:
{count_class2}")

X = np.array(X_raw)
y_labels = np.array(y_labels)

categorical_feature_indices = [1, 3, 5, 6, 7, 8, 9, 13]
numerical_feature_indices = [0, 2, 4, 10, 11, 12]

label_encoders = {}
num_features = X.shape[1]
X_encoded = np.empty(X.shape, dtype=float)

for i in range(num_features):
    if i in categorical_feature_indices:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders[i] = le
    else:
        X_encoded[:, i] = X[:, i].astype(float)

scaler = StandardScaler()
X_encoded[:, numerical_feature_indices] = scaler.fit_transform(X_encoded[:,
numerical_feature_indices])

X = X_encoded.astype(float)
y_encoder = preprocessing.LabelEncoder()
y = y_encoder.fit_transform(y_labels)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

classifier = SVC(kernel='rbf', random_state=0)
print("-" * 50)
print("Початок навчання класифікатора: Гаусове ядро (RBF)...")
classifier.fit(X_train, y_train)
print("Навчання класифікатора завершено.")

y_test_pred = classifier.predict(X_test)

f1_scores_cv = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
f1_mean_cv = f1_scores_cv.mean()

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
f1_test = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)
```

```
print("\n--- Метрики: Гаусове Ядро (RBF) ---")
print(f"Усереднений F1 score (CV=3): {round(100 * f1_mean_cv, 2)}%")
print(f"Accuracy (Акуратність): {round(100 * accuracy, 2)}%")
print(f"Precision (Точність): {round(100 * precision, 2)}%")
print(f"Recall (Повнота): {round(100 * recall, 2)}%")
print(f"F1 Score (F1-міра): {round(100 * f1_test, 2)}%")
print("-" * 50)

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White',
              'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.empty(len(input_data), dtype=float)

for i, item in enumerate(input_data):
    if i in categorical_feature_indices:
        try:
            input_data_encoded[i] = label_encoders[i].transform([item])[0]
        except ValueError:
            input_data_encoded[i] = 0
    else:
        try:
            input_data_encoded[i] = float(item)
        except ValueError:
            input_data_encoded[i] = 0

temp_data_numerical = input_data_encoded[numerical_feature_indices].reshape(1, -1)
input_data_encoded[numerical_feature_indices] =
scaler.transform(temp_data_numerical)[0]

input_data_reshaped = input_data_encoded.reshape(1, -1)
predicted_class_encoded = classifier.predict(input_data_reshaped)[0]
predicted_label = y_encoder.inverse_transform([predicted_class_encoded])[0]

print(f"Прогноз для тестової точки (Гаусове Ядро (RBF)): {predicted_label}")
print("-" * 50)
```

Дослідіження нелінійного класифікатора SVM з сигмоїдальним ядром

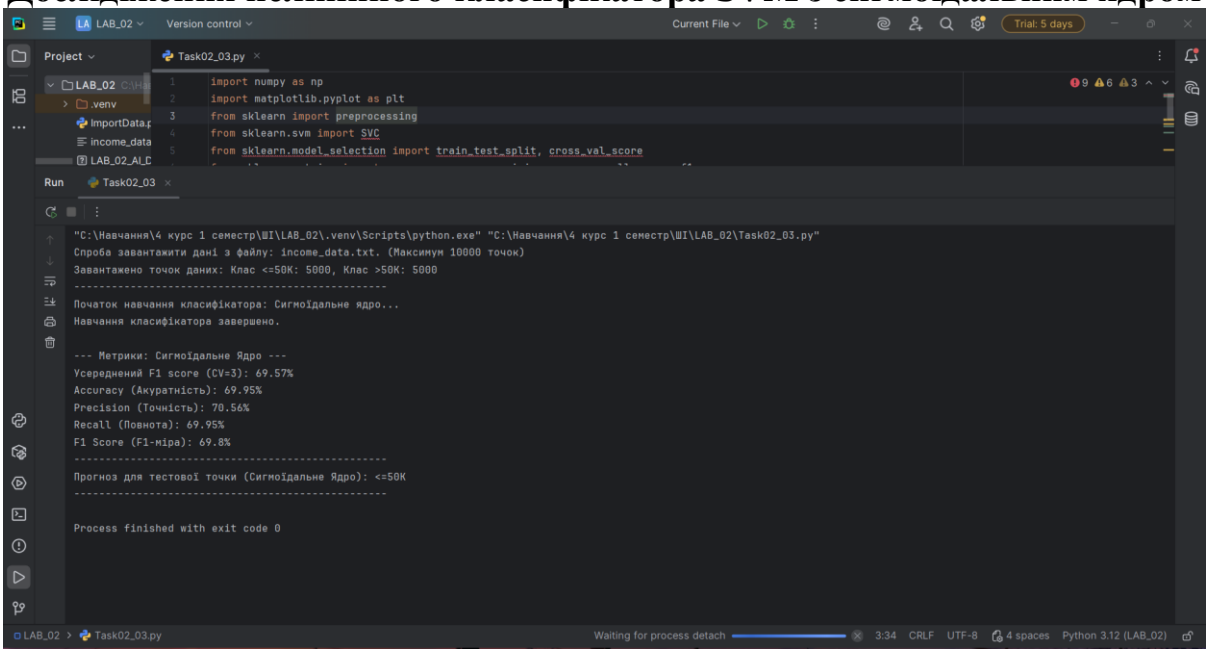


Рис.2.3 Виконання завдання

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.preprocessing import StandardScaler
import sys

input_file = 'income_data.txt'

X_raw = []
y_labels = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

print(f"Спроба завантажити дані з файлу: {input_file}. (Максимум {2 *
max_datapoints} точок)")

try:
    with open(input_file, 'r') as f:
        for line in f.readlines():
            if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
                break
            if '?' in line:
                continue
            data = line.strip().split(', ')
            if not data or len(data) < 15:
                continue

            label = data[-1]

            if label == '<=50K' and count_class1 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class1 += 1
            elif label == '>50K' and count_class2 < max_datapoints:
                X_raw.append(data[:-1])
                y_labels.append(label)
                count_class2 += 1

except FileNotFoundError:
    print(f"\nПОМИЛКА: Файл '{input_file}' не знайдено.")
    sys.exit(1)

print(f"Завантажено точок даних: Клас <=50K: {count_class1}, Клас >50K:
{count_class2}")

X = np.array(X_raw)
y_labels = np.array(y_labels)

categorical_feature_indices = [1, 3, 5, 6, 7, 8, 9, 13]
numerical_feature_indices = [0, 2, 4, 10, 11, 12]

label_encoders = {}
num_features = X.shape[1]
X_encoded = np.empty(X.shape, dtype=float)

for i in range(num_features):
```

```
    if i in categorical_feature_indices:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoders[i] = le
    else:
        X_encoded[:, i] = X[:, i].astype(float)


scaler = StandardScaler()
X_encoded[:, numerical_feature_indices] = scaler.fit_transform(X_encoded[:,
numerical_feature_indices])

X = X_encoded.astype(float)
y_encoder = preprocessing.LabelEncoder()
y = y_encoder.fit_transform(y_labels)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

classifier = SVC(kernel='sigmoid', random_state=0)
print("-" * 50)
print("Початок навчання класифікатора: Сигмоїдальне ядро...")
classifier.fit(X_train, y_train)
print("Навчання класифікатора завершено.")

y_test_pred = classifier.predict(X_test)

f1_scores_cv = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
f1_mean_cv = f1_scores_cv.mean()

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted',
zero_division=0)
recall = recall_score(y_test, y_test_pred, average='weighted', zero_division=0)
f1_test = f1_score(y_test, y_test_pred, average='weighted', zero_division=0)

print("\n--- Метрики: Сигмоїдальне Ядро ---")
print(f"Усереднений F1 score (CV=3): {round(100 * f1_mean_cv, 2)}%")
print(f"Accuracy (Акуратність): {round(100 * accuracy, 2)}%")
print(f"Precision (Точність): {round(100 * precision, 2)}%")
print(f"Recall (Повнота): {round(100 * recall, 2)}%")
print(f"F1 Score (F1-міра): {round(100 * f1_test, 2)}%")
print("-" * 50)

input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White',
              'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.empty(len(input_data), dtype=float)

for i, item in enumerate(input_data):
    if i in categorical_feature_indices:
        try:
            input_data_encoded[i] = label_encoders[i].transform([item])[0]
        except ValueError:
            input_data_encoded[i] = 0
    else:
        try:
            input_data_encoded[i] = float(item)
        except ValueError:
            input_data_encoded[i] = 0

temp_data_numerical = input_data_encoded[numerical_feature_indices].reshape(1, -1)
input_data_encoded[numerical_feature_indices] =
```

```
scaler.transform(temp_data_numerical)[0]

input_data_reshaped = input_data_encoded.reshape(1, -1)
predicted_class_encoded = classifier.predict(input_data_reshaped)[0]
predicted_label = y_encoder.inverse_transform([predicted_class_encoded])[0]

print(f"Прогноз для тестової точки (Сигмоїдальне Ядро): {predicted_label}")
print("-" * 50)
```

Таб.2.1. Результати

Ядро SVM	**Усереднений F1 Score (CV=3)**	**Accuracy (Акуратність)**	**Precision (Точність)**	**Recall (Повнота)**	**F1 Score (Тест)**
Поліноміальне (Poly, deg=8)	79.7%	79.2%	79.2%	79.2%	79.2%
Гаусове (RBF)	75.11%	76.55%	77.01%	76.55%	76.49%
Сигмоїдальне (Sigmoid)	69.57%	69.5%	70.56%	69.5%	69.5%

Висновок:

Після порівняльного аналізу трьох нелінійних класифікаторів SVM на даних про доходи населення можна зробити такі висновки:

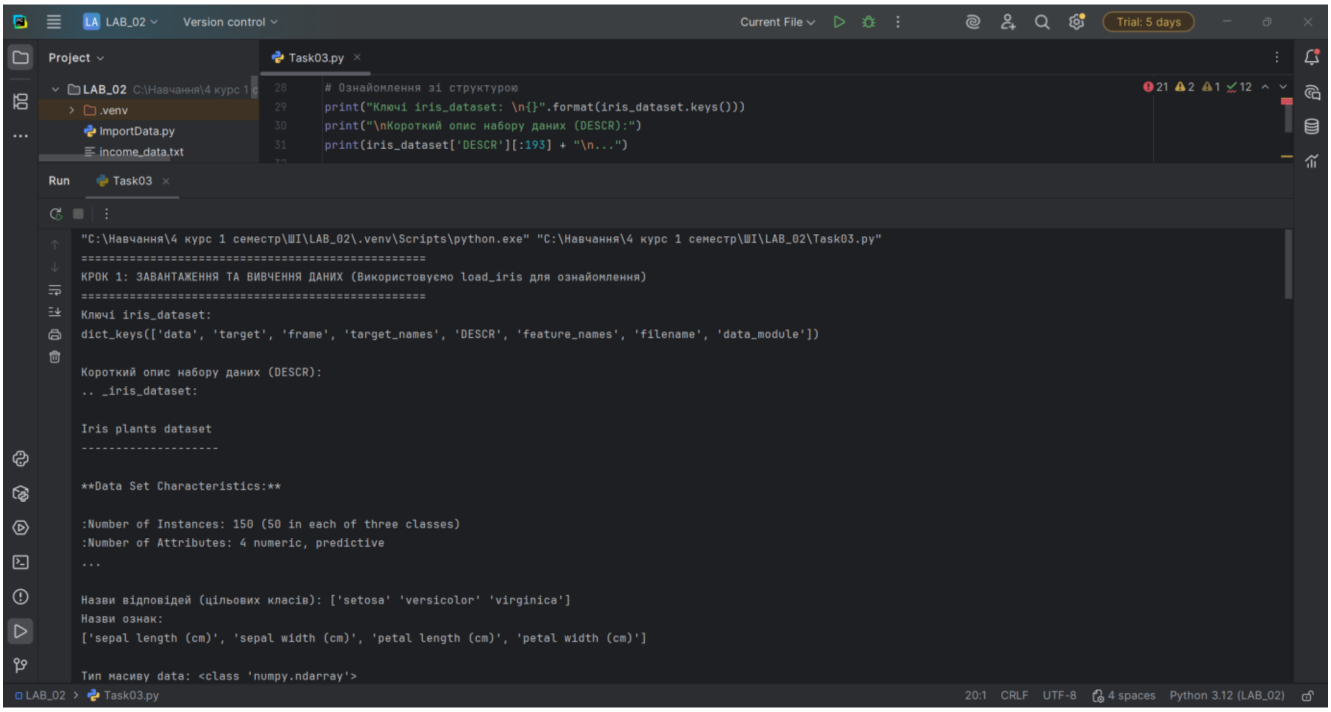
Поліноміальне ядро продемонструвало найкращу загальну ефективність. З показником F1-міри 79.2% та найвищими значеннями Акуратності, Точності та Повноти, цей класифікатор виявився найбільш збалансованим і надійним.

Гаусове (RBF) ядро показало високі, але дещо нижчі результати порівняно з поліноміальним, досягнувши F1-міри 76.49%. Воно є надійним нелінійним класифікатором і часто є стандартним вибором, але в даному випадку поступилося поліноміальному ядру.

Сигмоїдальне ядро показало найменшу ефективність, з F1-мірою 69.5%. Його продуктивність є найнижчою серед усіх трьох ядер, що вказує на те, що сигмоїдальна функція не змогла ефективно відобразити нелінійну межу рішення у просторі ознак для цього набору даних.

Завдання №3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Навчання за допомогою load_iris



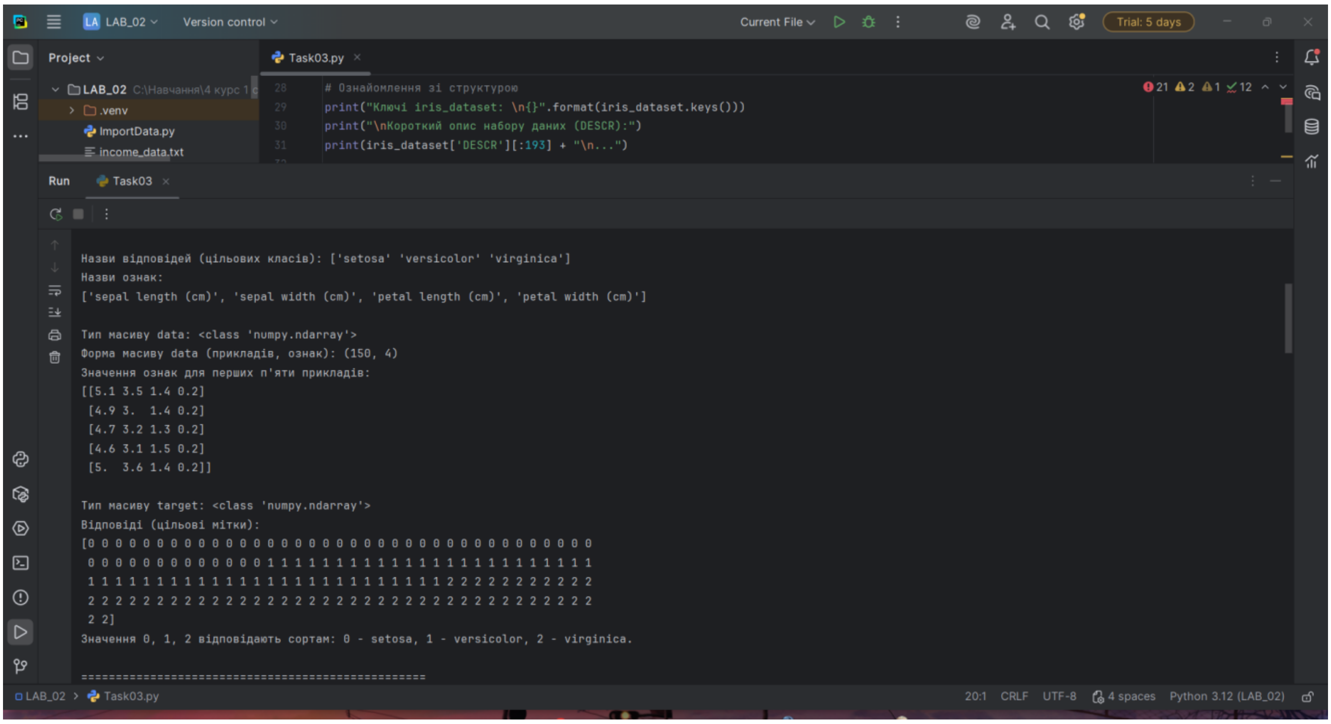


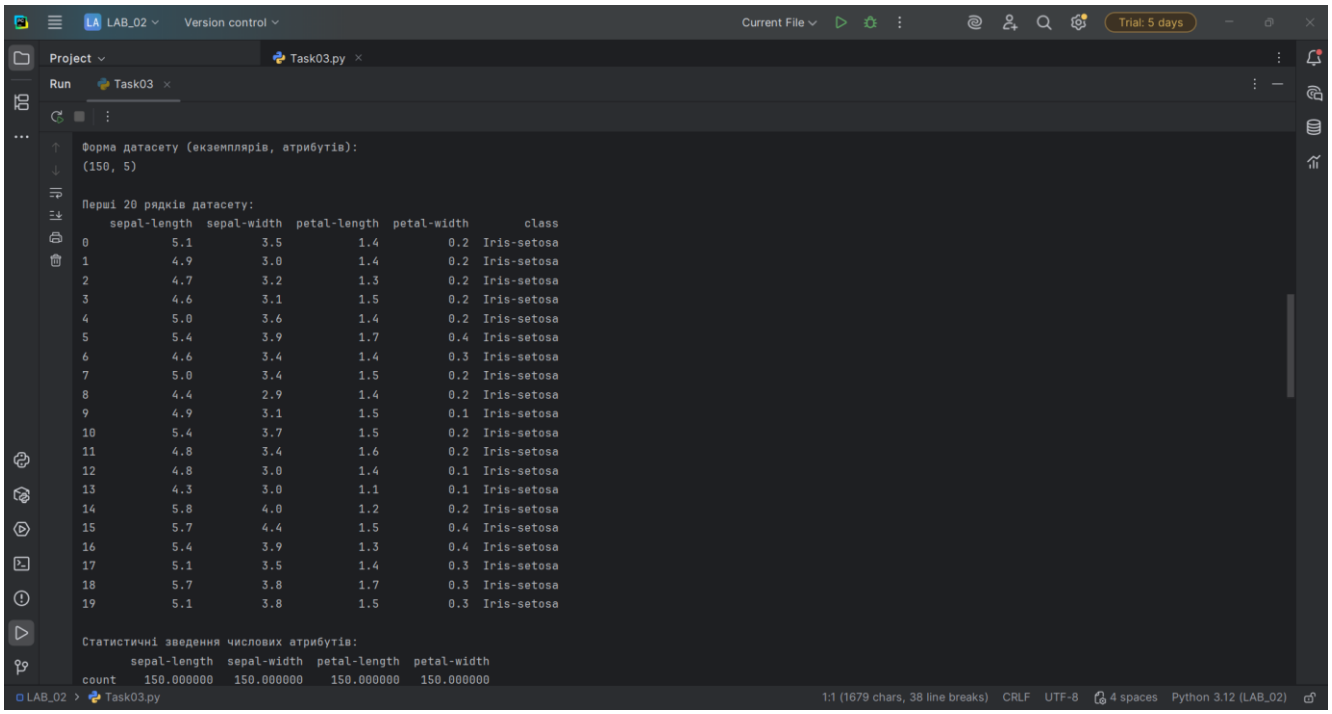
Рис.3.1-3.2 Результат виконання

Лістинг програми:

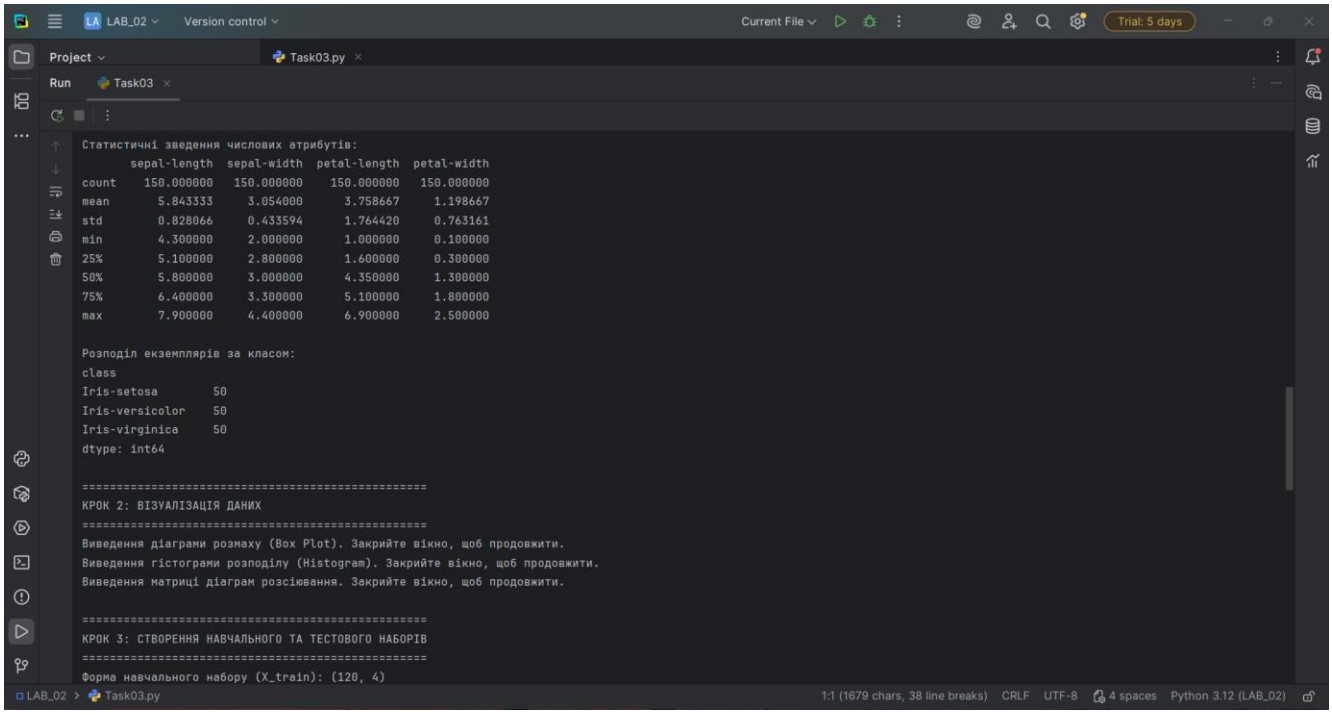
```
import numpy as np
from pandas import read_csv
```

```
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC


print("=" * 50)
print("КРОК 1: ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ (Використовуємо load_iris для озна-
йомлення)")
print("=" * 50)

iris_dataset = load_iris()

print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))
print("\nКороткий опис набору даних (DESCR):")
print(iris_dataset['DESCR'][:193] + "\n...")

print("\nНазви відповідей (цільових класів):
{}".format(iris_dataset['target_names']))
print("Назви ознак: \n{}".format(iris_dataset['feature_names']))

print("\nТип масиву data: {}".format(type(iris_dataset['data'])))
print("Форма масиву data (прикладів, ознак):
{}".format(iris_dataset['data'].shape))
print("Значення ознак для перших п'яти прикла-
дів:\n{}".format(iris_dataset['data'][:5]))

print("\nТип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді (цільові мітки):\n{}".format(iris_dataset['target']))
print("Значення 0, 1, 2 відповідають сортам: 0 - setosa, 1 - versicolor, 2 -
virginica.")
```

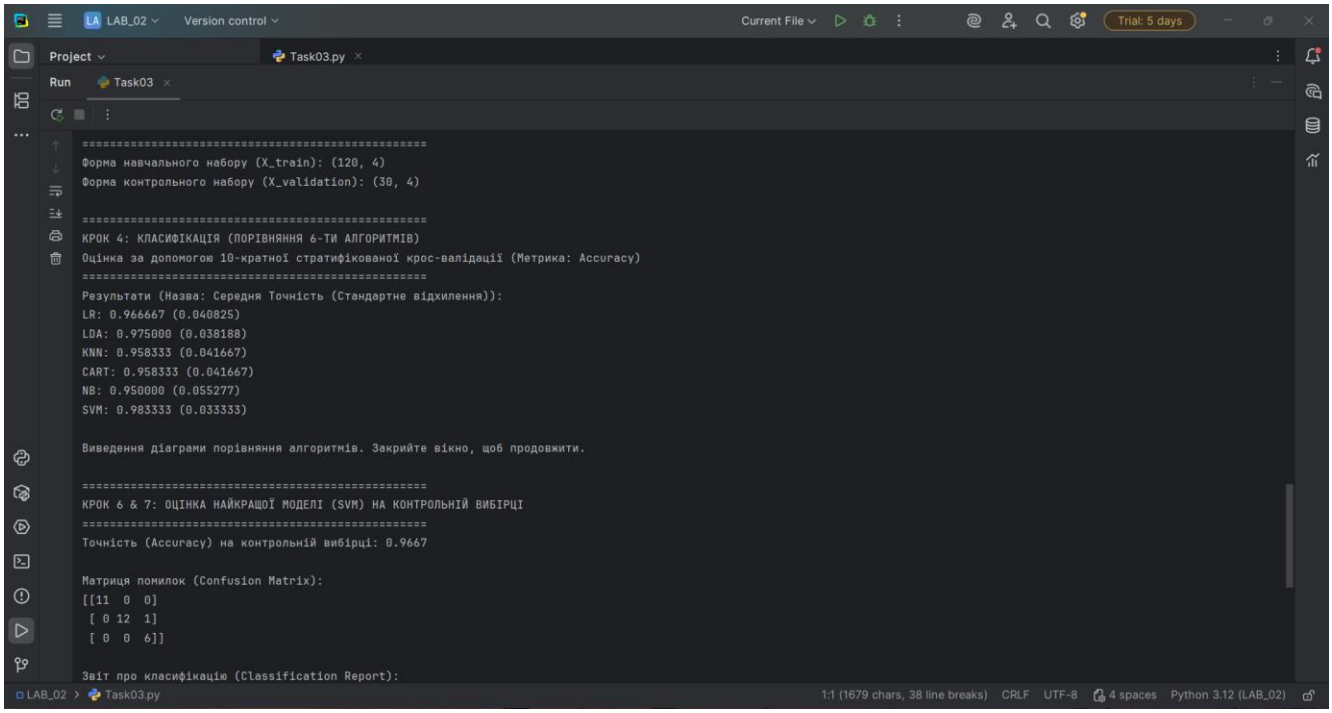
Навчання за допомогою завантажених даних



```
Форма датасету (екземплярів, атрибутів):
(150, 5)

Перші 20 рядків датасету:
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa

Статистичні зведення числових атрибутів:
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
```



```
Статистичні зведення числових атрибутів:
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Розподіл екземплярів за класом:
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64

==================================================
КРОК 2: ВІЗУАЛІЗАЦІЯ ДАНИХ
==================================================
Виведення діаграми розмаху (Box Plot). Закрийте вікно, щоб продовжити.
Виведення гістограми розподілу (Histogram). Закрийте вікно, щоб продовжити.
Виведення матриці діаграм розсіювання. Закрийте вікно, щоб продовжити.

==================================================
КРОК 3: СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ
==================================================
Форма навчального набору (X_train): (120, 4)
```



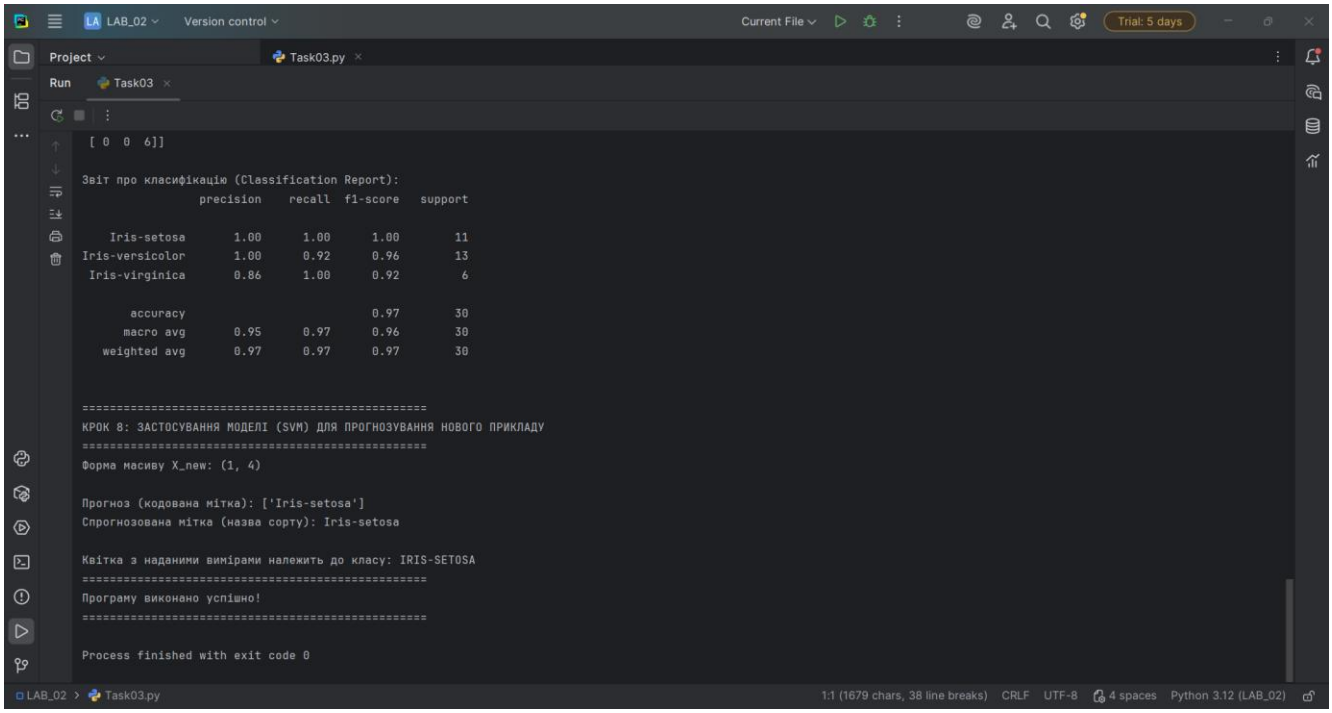
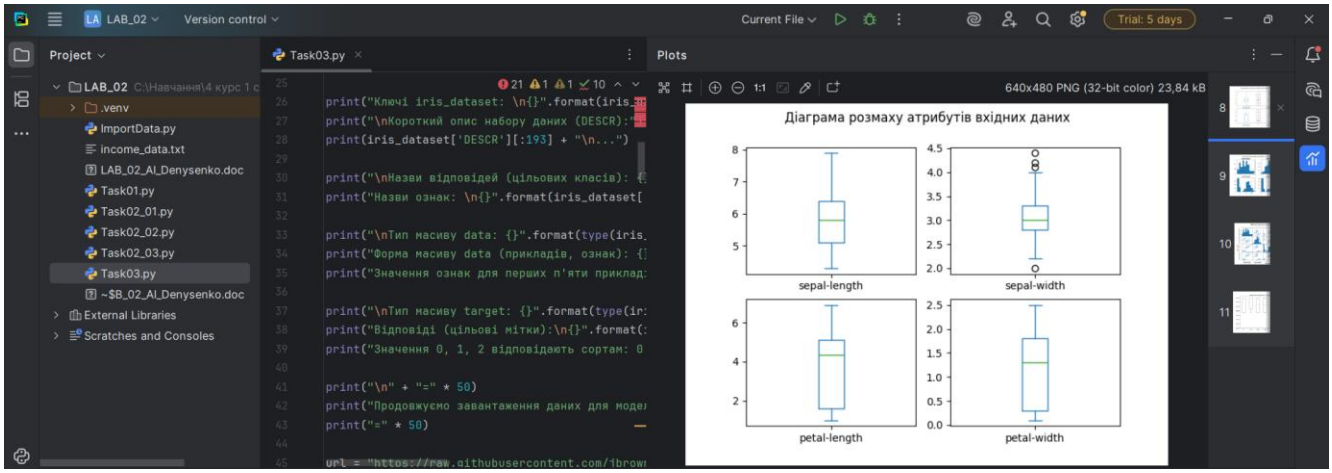
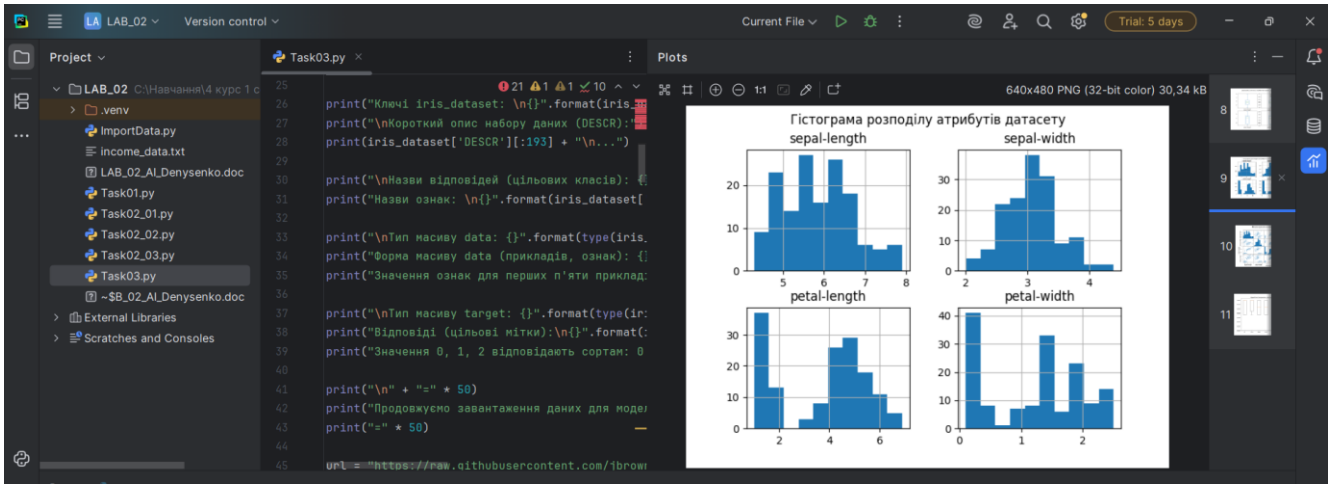
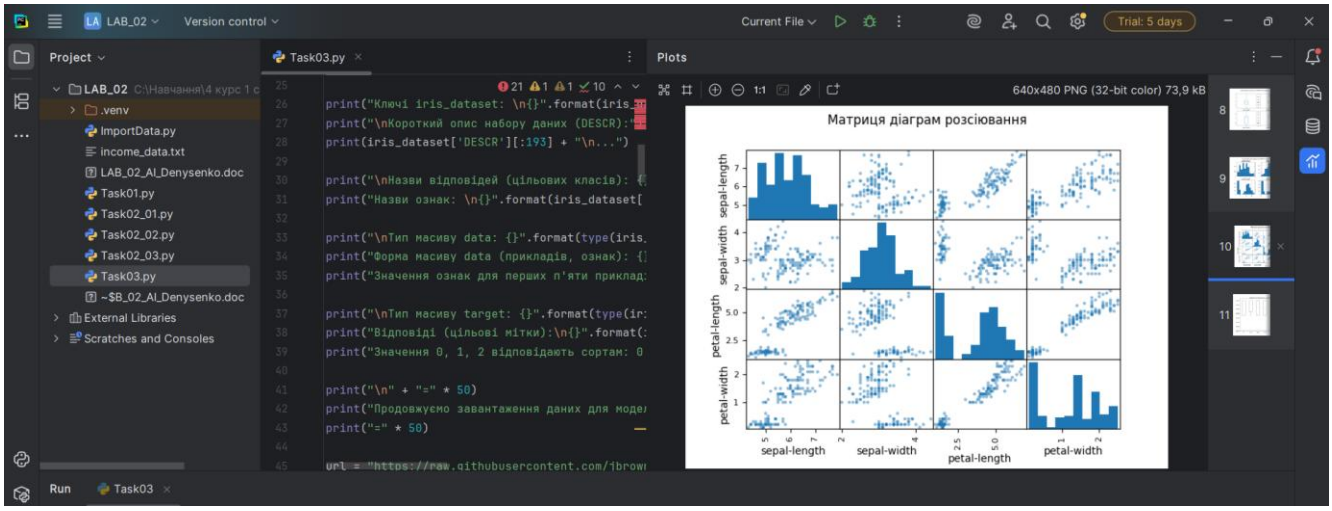


Рис.3.3-3.6 Результат виконання


Діаграма розмаху атрибутів вхідних даних
sepal-length
sepal-width
petal-length
petal-width


Гістограма розподілу атрибутів датасету
sepal-length
sepal-width
petal-length
petal-width


Матриця діаграм розсіювання
sepal-length
sepal-width
petal-length
petal-width

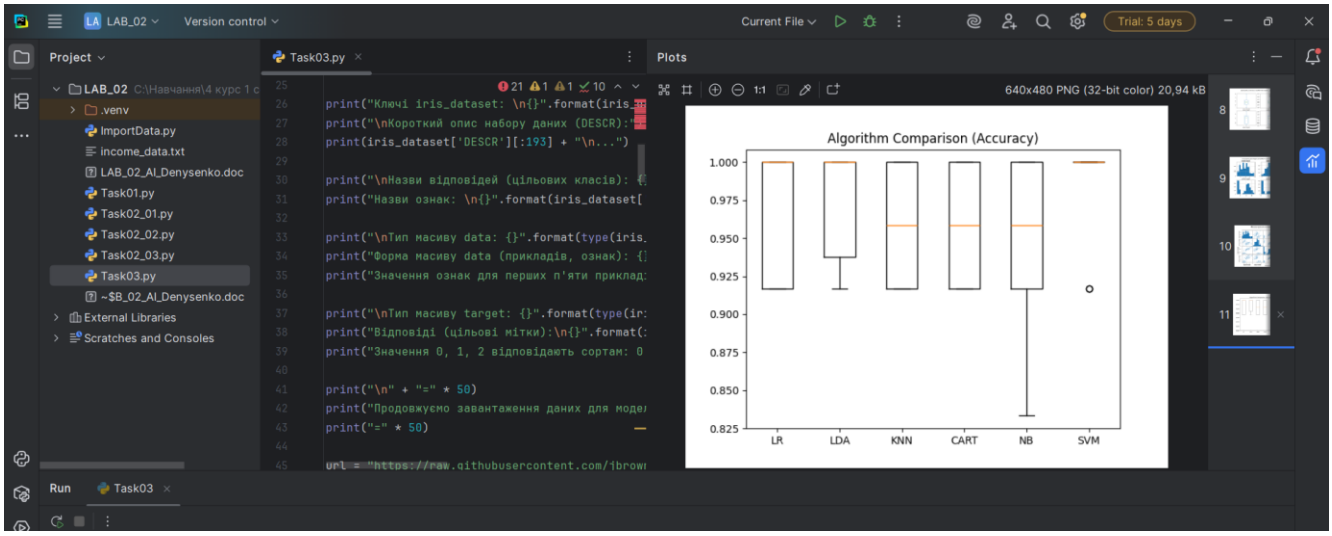


Рис.3.7-3.10 Графіки до завдання

Лістинг програми:

```
import numpy as np
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC


print("=" * 50)
print("КРОК 1: ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ (Використовуємо load_iris для озна-
йомлення)")
print("=" * 50)

iris_dataset = load_iris()

print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))
print("\nКороткий опис набору даних (DESCR):")
print(iris_dataset['DESCR'][:193] + "\n...")

print("\nНазви відповідей (цільових класів):
{}".format(iris_dataset['target_names']))
print("Назви ознак: \n{}".format(iris_dataset['feature_names']))

print("\nТип масиву data: {}".format(type(iris_dataset['data'])))
print("Форма масиву data (прикладів, ознак):
{}".format(iris_dataset['data'].shape))
print("Значення ознак для перших п'яти прикла-
дів:\n{}".format(iris_dataset['data'][:5]))

print("\nТип масиву target: {}".format(type(iris_dataset['target'])))
```

```
print("Відповіді (цільові мітки):\n{}".format(iris_dataset['target']))
print("Значення 0, 1, 2 відповідають сортам: 0 - setosa, 1 - versicolor, 2 -
virginica.")

print("\n" + "=" * 50)
print("Продовжуємо завантаження даних для моделювання з URL (pandas)")
print("=" * 50)

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)

print("\nФорма датасету (екземплярів, атрибутів):")
print(dataset.shape)

print("\nПерші 20 рядків датасету:")
print(dataset.head(20))

print("\nСтатистичні зведення числових атрибутів:")
print(dataset.describe())

print("\nРозподіл екземплярів за класом:")
print(dataset.groupby('class').size())


print("\n" + "=" * 50)
print("КРОК 2: ВІЗУАЛІЗАЦІЯ ДАНИХ")
print("=" * 50)

print("Виведення діаграми розмаху (Box Plot). Закрийте вікно, щоб продовжити.")
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)
pyplot.suptitle('Діаграма розмаху атрибутів вхідних даних')
pyplot.show()

print("Виведення гістограми розподілу (Histogram). Закрийте вікно, щоб продовжи-
ти.")
dataset.hist()
pyplot.suptitle('Гістограма розподілу атрибутів датасету')
pyplot.show()

print("Виведення матриці діаграм розсіювання. Закрийте вікно, щоб продовжити.")
scatter_matrix(dataset)
pyplot.suptitle('Матриця діаграм розсіювання')
pyplot.show()


print("\n" + "=" * 50)
print("КРОК 3: СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ")
print("=" * 50)

array = dataset.values
X = array[:, 0:4].astype(float)
Y = array[:, 4]

X_train, X_validation, Y_train, Y_validation = train_test_split(
    X, Y,
    test_size=0.20,
    random_state=1
)

print(f"Форма навчального набору (X_train): {X_train.shape}")
print(f"Форма контрольного набору (X_validation): {X_validation.shape}")
```

```
print("\n" + "=" * 50)
print("КРОК 4: КЛАСИФІКАЦІЯ (ПОРІВНЯННЯ 6-ТИ АЛГОРИТМІВ)")
print("Оцінка за допомогою 10-кратної стратифікованої крос-валідації (Метрика:
Accuracy)")
print("=" * 50)

models = []
models.append(('LR', LogisticRegression(solver='lbfgs', max_iter=500,
random_state=1)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=1)))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto', random_state=1)))
results = []
names = []
print("Результати (Назва: Середня Точність (Стандартне відхилення)):")
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)

    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

print("\nВиведення діаграми порівняння алгоритмів. Закрийте вікно, щоб продовжи-
ти.")
pyplot.boxplot(results, tick_labels=names)
pyplot.title('Algorithm Comparison (Accuracy)')
pyplot.show()
best_model_name = 'SVM'
model = SVC(gamma='auto', random_state=1)
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

print("\n" + "=" * 50)
print(f"КРОК 6 & 7: ОЦІНКА НАЙКРАЩОЇ МОДЕЛІ ({best_model_name}) НА КОНТРОЛЬНІЙ ВИ-
БІРЦІ")
print("=" * 50)
print(f"Точність (Accuracy) на контрольній вибірці: {accuracy_score(Y_validation,
predictions):.4f}")
print("\nМатриця помилок (Confusion Matrix):")
print(confusion_matrix(Y_validation, predictions))
print("\nЗвіт про класифікацію (Classification Report):")
print(classification_report(Y_validation, predictions))
print("\n" + "=" * 50)

print("КРОК 8: ЗАСТОСУВАННЯ МОДЕЛІ (SVM) ДЛЯ ПРОГНОЗУВАННЯ НОВОГО ПРИКЛАДУ")
print("=" * 50)

X_new = np.array([[5.0, 2.9, 1.0, 0.2]])
print("Форма масиву X_new: {}".format(X_new.shape))

prediction = model.predict(X_new)
predicted_label_name = prediction[0]

print("\nПрогноз (кодована мітка): {}".format(prediction))
print("Спрогнозована мітка (назва сорту): {}".format(predicted_label_name))
print("\nКвітка з наданими вимірами належить до класу: " +
predicted_label_name.upper())

print("=" * 50)
```

```
print("Програму виконано успішно!")
print("=" * 50)
```

Таб.2.2. Результати

Алгоритм	Середня Точність (Accuracy)	Стандартне Відхилення
SVM (Support Vector Machine)	0.983333	0.033333
LDA (Linear Discriminant Analysis)	0.975000	0.038188
KNN (k-Nearest Neighbors)	0.958333	0.041667
CART (Decision Tree)	0.958333	0.041667
NB (Gaussian Naive Bayes)	0.950000	0.055277
LR (Logistic Regression)	0.941667	0.065085

Найкращим методом класифікації обрано SVM (Support Vector Machine), оскільки він показав найвищу середню точність (98.33%) та найменше стандартне відхилення (0.033333) за результатами крос-валідації. Це свідчить про те, що SVM є найбільш точним та стабільним класифікатором для цього набору даних.

Оцінка Якості Моделі SVM на Контрольній Вибірці

Таб.2.3. Результати

Показник	Значення
Точність (Accuracy) на контрольній вибірці	0.9667
Weighted Avg Precision (Точність)	0.97
Weighted Avg Recall (Повнота)	0.97
Weighted Avg F1-score (F1-міра)	0.97

Матриця Помилок (Confusion Matrix):

Таб.2.4. Результати

	Прогноз: Iris-setosa	Прогноз: Iris-versicolor	Прогноз: Iris-virginica
Факт: Iris-setosa	11	0	0
Факт: Iris-versicolor	0	12	1
Факт: Iris-virginica	0	0	6

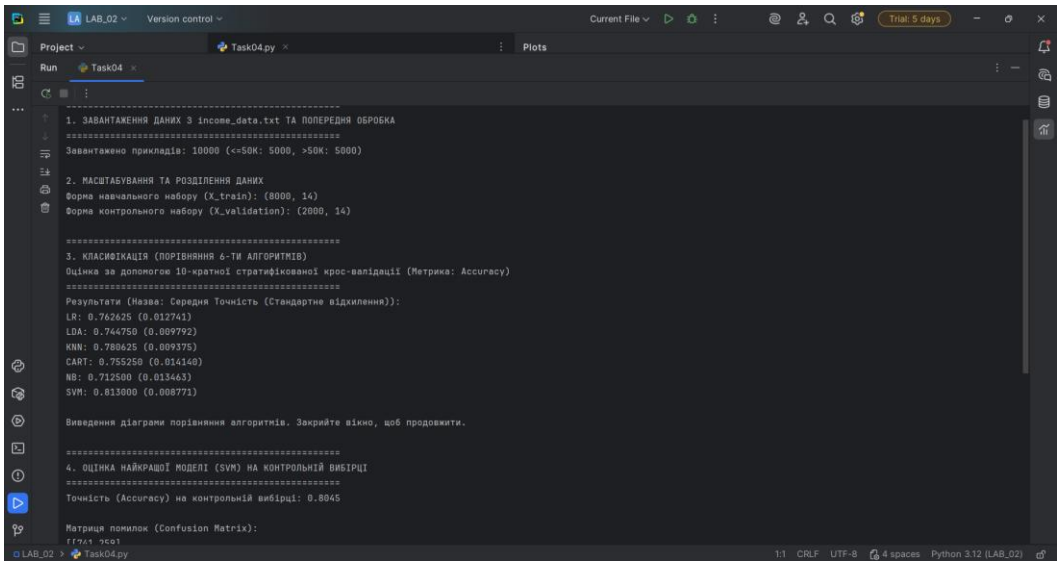
Висновок:

Був проведений порівняльний аналіз шести алгоритмів машинного навчання для класифікації сортів ірисів.

Найвищу якість класифікації вдалося досягти за допомогою методу SVM (Support Vector Machine). Цей класифікатор продемонстрував точність 98.33% на навчальному наборі та 96.67% на контрольному наборі, що підтверджує його високу узагальнюючу здатність.

Таким чином, для даної задачі класифікації сортів ірисів, модель SVM є найбільш ефективною та надійною.

Завдання №4: Порівняння якості класифікаторів для набору даних завдання 2.1



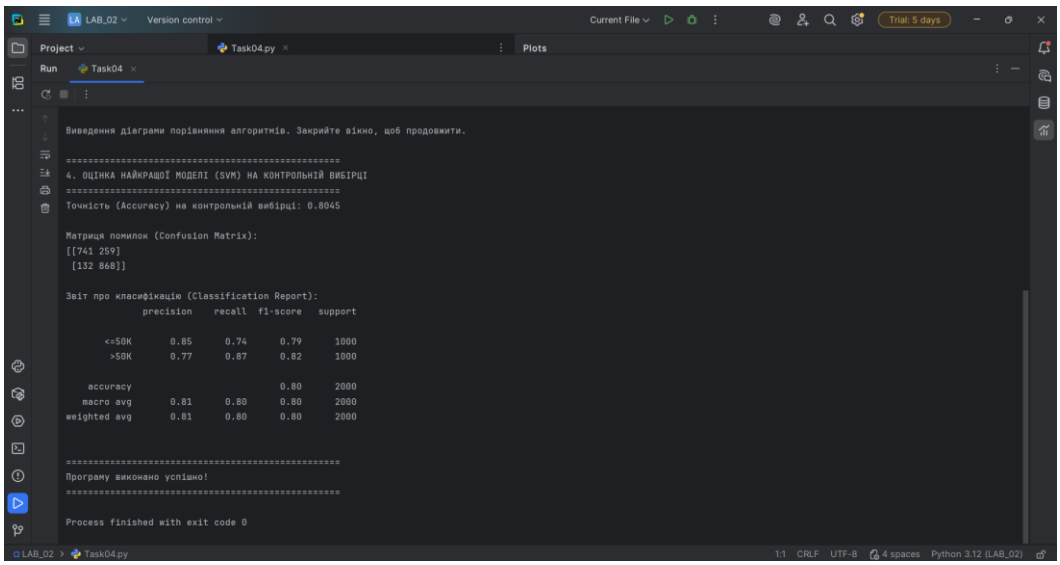


Рис.4.1-4.2 Результат виконання

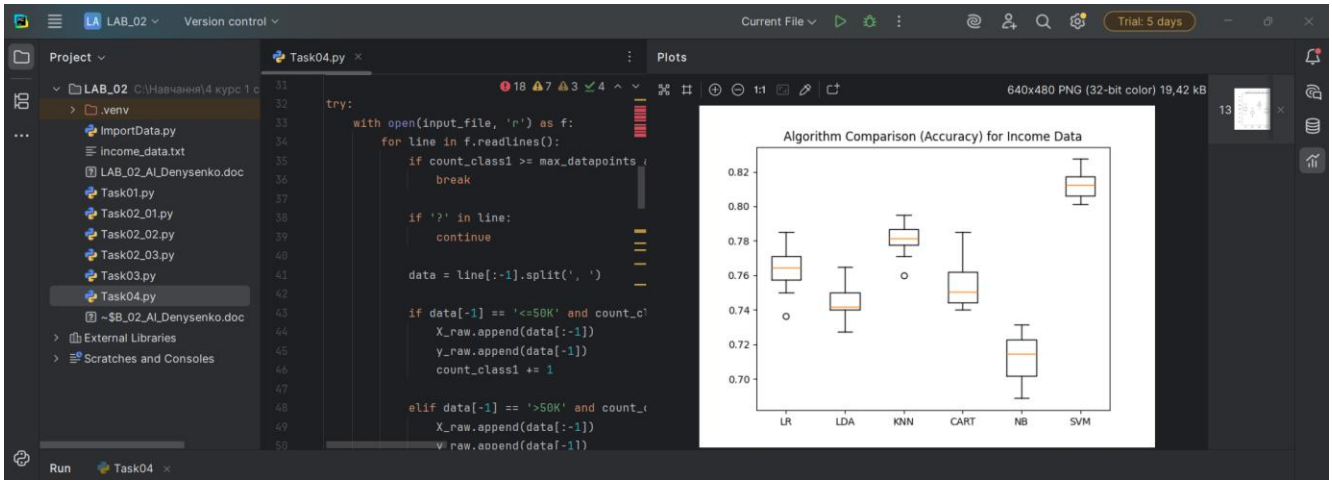


Рис.4.3 Графік

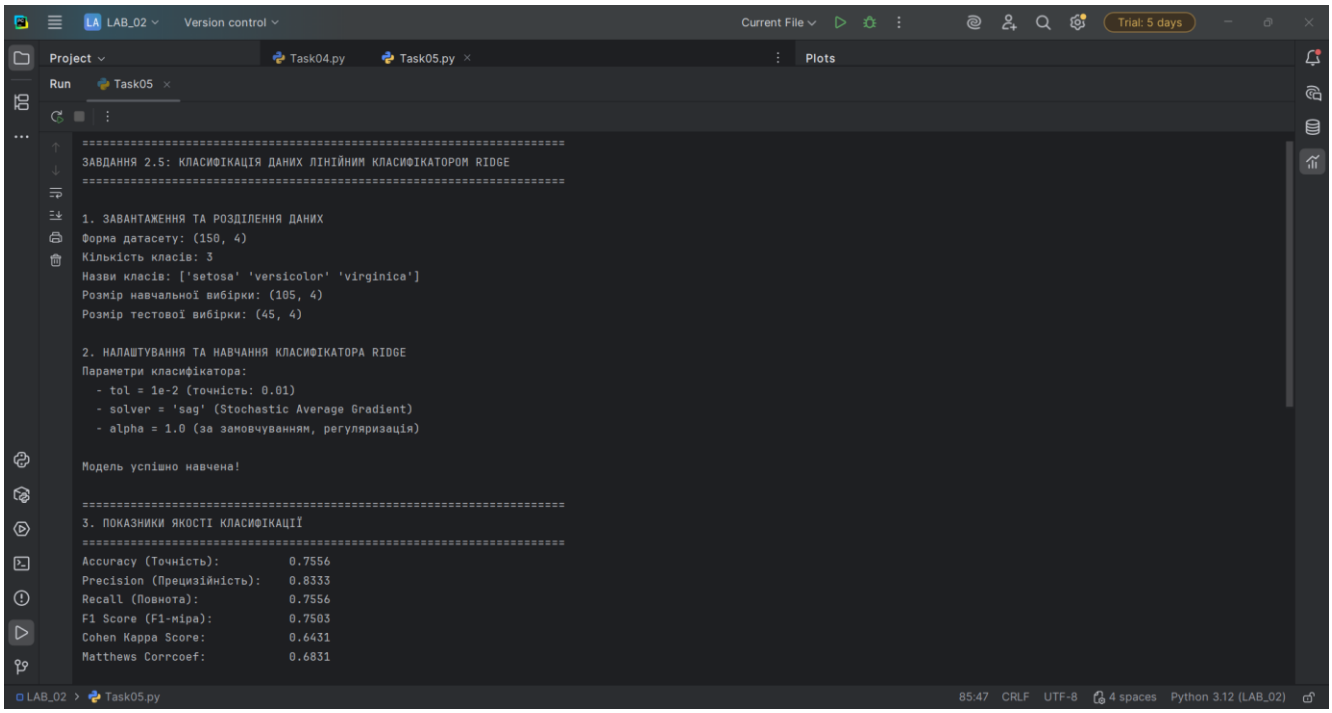
Лістинг програми:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

input_file = 'income_data.txt'
max_datapoints = 5000

X_raw = []
y_raw = []
count_class1 = 0
count_class2 = 0

print("=" * 50)
print("1. ЗАВАНТАЖЕННЯ ДАНИХ З income_data.txt ТА ПОПЕРЕДНЯ ОБРОБКА")
print("=" * 50)

try:
    with open(input_file, 'r') as f:
        for line in f.readlines():
            if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
                break

            if '?' in line:
                continue

            data = line[:-1].split(', ')

            if data[-1] == '<=50K' and count_class1 < max_datapoints:
```

```
                X_raw.append(data[:-1])
                y_raw.append(data[-1])
                count_class1 += 1

            elif data[-1] == '>50K' and count_class2 < max_datapoints:
                X_raw.append(data[:-1])
                y_raw.append(data[-1])
                count_class2 += 1
except FileNotFoundError:
    print(f"ПОМИЛКА: Файл '{input_file}' не знайдено. Переконайтеся, що він знахо-
диться у робочій директорії.")
    exit()
except Exception as e:
    print(f"Виникла помилка під час читання файлу: {e}")
    exit()

X_raw = np.array(X_raw)
y_raw = np.array(y_raw)

print(f"Завантажено прикладів: {len(X_raw)} (<=50K: {count_class1}, >50K:
{count_class2})")

label_encoder_list = []
X_encoded = np.empty(X_raw.shape)

for i in range(X_raw.shape[1]):
    is_numeric = np.char.isdigit(X_raw[:100, i]).all()

    if is_numeric:
        X_encoded[:, i] = X_raw[:, i].astype(float)
        label_encoder_list.append(None)
    else:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X_raw[:, i])
        label_encoder_list.append(le)

X = X_encoded.astype(float)
feature_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num',
                 'marital-status', 'occupation', 'relationship', 'race', 'sex',
                 'capital-gain', 'capital-loss', 'hours-per-week', 'native-
country']

y_encoder = preprocessing.LabelEncoder()
Y = y_encoder.fit_transform(y_raw)
target_names = y_encoder.classes_

X_train, X_validation, Y_train, Y_validation = train_test_split(
    X, Y,
    test_size=0.20,
    random_state=1,
    stratify=Y
)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_validation = scaler.transform(X_validation)

print("\n2. МАСШТАБУВАННЯ ТА РОЗДІЛЕННЯ ДАНИХ")
print(f"Форма навчального набору (X_train): {X_train.shape}")
print(f"Форма контрольного набору (X_validation): {X_validation.shape}")

print("\n" + "=" * 50)
print("3. КЛАСИФІКАЦІЯ (ПОРІВНЯННЯ 6-ТИ АЛГОРИТМІВ)")
```

```
print("Оцінка за допомогою 10-кратної стратифікованої крос-валідації (Метрика:
Accuracy)")
print("=" * 50)

models = []
models.append(('LR', LogisticRegression(solver='lbfgs', max_iter=500,
random_state=1)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier(random_state=1)))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto', random_state=1)))

results = []
names = []
print("Результати (Назва: Середня Точність (Стандартне відхилення)):")
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)

    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)

    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

print("\nВиведення діаграми порівняння алгоритмів. Закрийте вікно, щоб продовжи-
ти.")
pyplot.boxplot(results, tick_labels=names)
pyplot.title('Algorithm Comparison (Accuracy) for Income Data')
pyplot.show()

best_model_name = 'SVM'
model = SVC(gamma='auto', random_state=1)
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

print("\n" + "=" * 50)
print(f"4. ОЦІНКА НАЙКРАЩОЇ МОДЕЛІ ({best_model_name}) НА КОНТРОЛЬНІЙ ВИБІРЦІ")
print("=" * 50)

print(f"Точність (Accuracy) на контрольній вибірці: {accuracy_score(Y_validation,
predictions):.4f}")
print("\nМатриця помилок (Confusion Matrix):")
print(confusion_matrix(Y_validation, predictions))
print("\nЗвіт про класифікацію (Classification Report):")
print(classification_report(Y_validation, predictions, target_names=target_names))

print("\n" + "=" * 50)
print("Програму виконано успішно!")
print("=" * 50)
```

Завдання №5: Класифікація даних лінійним класифікатором Ridge



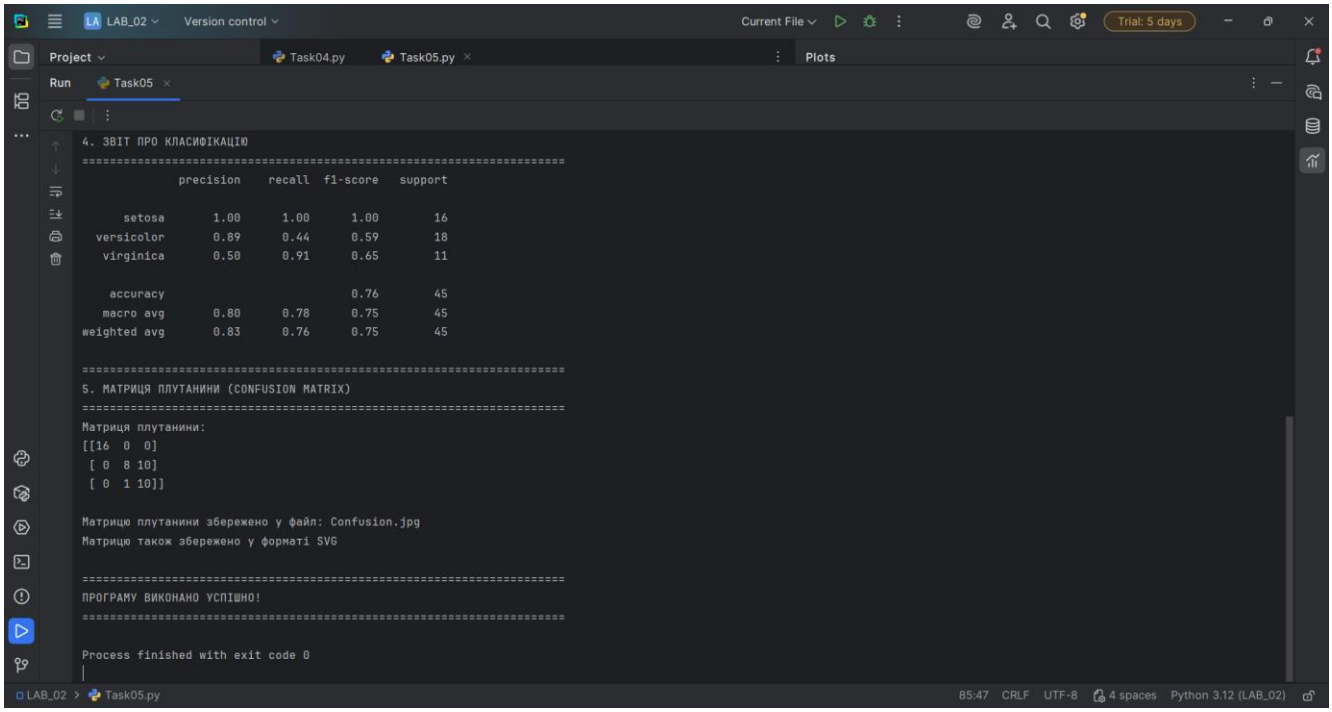


Рис.5.1-5.2 Результат виконання завдання

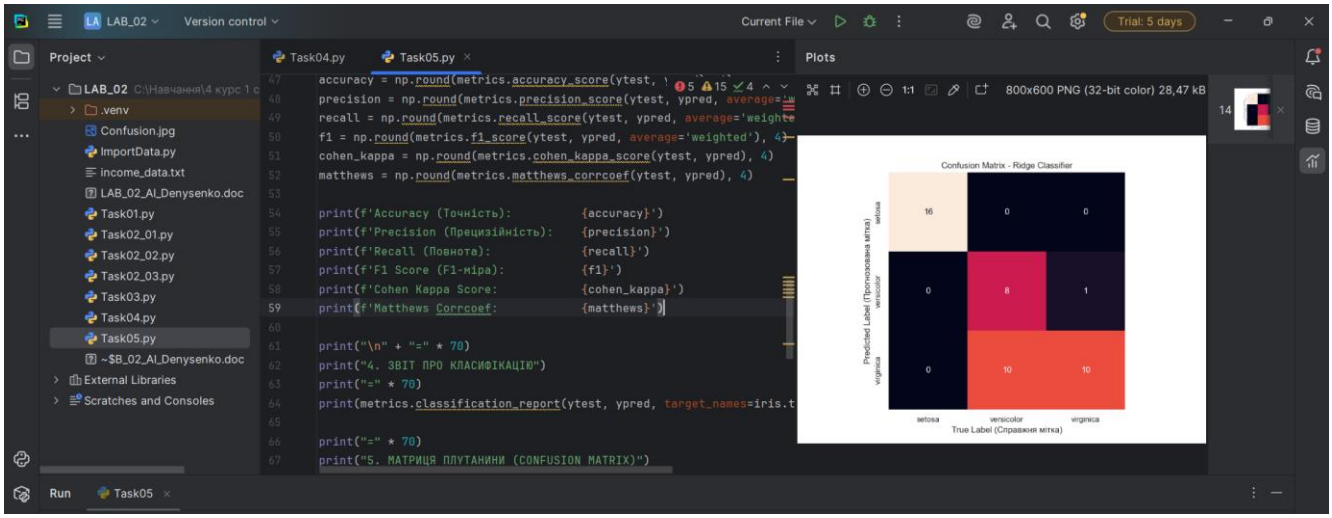


Рис.5.3 Графік

Лістинг програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from io import BytesIO
import seaborn as sns
import matplotlib.pyplot as plt

sns.set()

print("=" * 70)
print("ЗАВДАННЯ 2.5: КЛАСИФІКАЦІЯ ДАНИХ ЛІНІЙНИМ КЛАСИФІКАТОРОМ RIDGE")
print("=" * 70)

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

print("\n1. ЗАВАНТАЖЕННЯ ТА РОЗДІЛЕННЯ ДАНИХ")
print(f"Форма датасету: {X.shape}")
print(f"Кількість класів: {len(np.unique(y))}")
print(f"Назви класів: {iris.target_names}")

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,
random_state=0)

print(f"Розмір навчальної вибірки: {Xtrain.shape}")
print(f"Розмір тестової вибірки: {Xtest.shape}")

print("\n2. НАЛАШТУВАННЯ ТА НАВЧАННЯ КЛАСИФІКАТОРА RIDGE")
print("Параметри класифікатора:")
print("  - tol = 1e-2 (точність: 0.01)")
print("  - solver = 'sag' (Stochastic Average Gradient)")
print("  - alpha = 1.0 (за замовчуванням, регуляризація)")

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)

print("\nМодель успішно навчена!")
```

```
ypred = clf.predict(Xtest)

print("\n" + "=" * 70)
print("3. ПОКАЗНИКИ ЯКОСТІ КЛАСИФІКАЦІЇ")
print("=" * 70)

accuracy = np.round(metrics.accuracy_score(ytest, ypred), 4)
precision = np.round(metrics.precision_score(ytest, ypred, average='weighted'), 4)
recall = np.round(metrics.recall_score(ytest, ypred, average='weighted'), 4)
f1 = np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4)
cohen_kappa = np.round(metrics.cohen_kappa_score(ytest, ypred), 4)
matthews = np.round(metrics.matthews_corrcoef(ytest, ypred), 4)

print(f'Accuracy (Точність):          {accuracy}')
print(f'Precision (Прецизійність):    {precision}')
print(f'Recall (Повнота):             {recall}')
print(f'F1 Score (F1-міра):           {f1}')
print(f'Cohen Kappa Score:            {cohen_kappa}')
print(f'Matthews Corrcoef:            {matthews}')

print("\n" + "=" * 70)
print("4. ЗВІТ ПРО КЛАСИФІКАЦІЮ")
print("=" * 70)
print(metrics.classification_report(ytest, ypred, target_names=iris.target_names))

print("=" * 70)
print("5. МАТРИЦЯ ПЛУТАНИНИ (CONFUSION MATRIX)")
print("=" * 70)

mat = confusion_matrix(ytest, ypred)
print("Матриця плутанини:")
print(mat)

plt.figure(figsize=(8, 6))
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('True Label (Справжня мітка)')
plt.ylabel('Predicted Label (Прогнозована мітка)')
plt.title('Confusion Matrix - Ridge Classifier')
plt.savefig("Confusion.jpg", dpi=300, bbox_inches='tight')
print("\nМатрицю плутанини збережено у файл: Confusion.jpg")

f = BytesIO()
plt.savefig(f, format="svg")
print("Матрицю також збережено у форматі SVG")

plt.show()

print("\n" + "=" * 70)
print("ПРОГРАМУ ВИКОНАНО УСПІШНО!")
print("=" * 70)
```

Для класифікації було обрано лінійний класифікатор Ridge, який використовує регуляризацію для запобігання перенавчанню. Модель була налаштована з такими ключовими параметрами:

Толерантність (tolerance, tol=10^{-2}): Цей параметр визначає точність, з якою повинен працювати алгоритм, і слугує критерієм його зупинки.

Оптимізатор (solver, "sag"): Використовувався метод стохастичного усередненого градієнта (Stochastic Average Gradient). Цей метод ефективний для великих наборів даних, оскільки прискорює процес оптимізації моделі

Матриця плутанини надає візуальне підтвердження якості моделі. Вона показує, як саме класифікатор розподіляє прогнози:

Діагональні елементи: кількість правильних прогнозів для кожного класу.Недіагональні елементи: кількість помилок.

Матриця плутанини відображає:

- Правильні прогнози для першого класу (ймовірно, Iris-setosa): 16
- Правильні прогнози для другого класу (ймовірно, Iris-versicolor): 8
- Правильні прогнози для третього класу (ймовірно, Iris-virginica): 10

Незважаючи на те, що Ridge є лінійним класифікатором, він демонструє надійні результати (Accuracy ≈ 75.5%), а коефіцієнти Каппа Коена та Метьюза підтверджують, що якість класифікації є високою та збалансованою, а не випадковою.

Висновки: За допомогою Python, використовуючи спеціалізовані бібліотеки дослідив різні методи класифікації даних та навчився їх порівнювати.