## ЛАБОРАТОРНА РОБОТА №2

## ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

**Хід роботи**

**GitHub репозиторій:** https://github.com/AlexanderHorielko/SAI_Horielko_PI-59

**Завдання 2.1**. Класифікація за допомогою машин опорних векторів (SVM)

| Назва | Опис | Тип значень |
|---|---|---|
| age | Вік | Числове |
| workclass | Вид працевлаштування | Категоріальне |
| fnlwgt | Кількість осіб з такими ж ознаками | Числове |
| education | Навчання | Категоріальне |
| education-num | Років навчання | Числове |
| marital-status | Сімейне положення | Категоріальне |
| occupation | Професія | Категоріальне |
| relationship | Відносини | Категоріальне |
| race | Раса | Категоріальне |
| sex | Стать | Категоріальне |
| capital-gain | Приріст капіталу | Числове |
| capital-loss | Втрата капіталу | Числове |
| hours-per-week | Кількість робочих годин на тиждень | Числове |
| native-country | Країна походження | Категоріальне |

```
import numpy as np
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

# Вхідний файл, який містить дані
input_file = 'income_data.txt'

# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

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.append(data)
            count_class1 += 1

        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)

# Перетворення рядкових даних на числові
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        current label encoder = preprocessing.LabelEncoder()
        label_encoder.append(current_label_encoder)
        X_encoded[:, i] = current_label_encoder.fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=5)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Accuracy score: " + str(round(100 * accuracy.mean(), 2)) + "%")

precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
print("Precision score: " + str(round(100 * precision.mean(), 2)) + "%")

recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)
print("Recall score: " + str(round(100 * recall.mean(), 2)) + "%")

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
```

```
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Han-
dlers-cleaners', 'Not-in-family', 'White',
              'Male','0', '0', '40', 'United-States']

# Кодування тестової точки даних
input_data_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = int(input_data[i])
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([(in-
put_data[i])])[-1])
        count += 1

input_data_encoded = np.array(input_data_encoded)

# Використання класифікатора для кодованої точки даних
# та виведення результату
predicted_class = classifier.predict([input_data_encoded])
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рисунок 1. Код програми

Accuracy score: 62.64%
Precision score: 69.18%
Recall score: 38.24%
F1 score: 56.15%
Тестова точка - <=50K. Отже тестова точка має дохід менше 50 тисяч в рік.

**Завдання 2.2.** Порівняння якості класифікаторів SVM з нелінійними ядрами

**Поліноміальне ядро**

Accuracy score: 58.41%

Precision score: 41.6%

Recall score: 33.05%

F1 score: 46.5%

>50K

**Гаусове ядро**

Accuracy score: 78.61%

Precision score: 98.72%

Recall score: 14.26%

F1 score: 71.95%

>50K

**Сигмоїдальне ядро**

Accuracy score: 63.89%

Precision score: 27.01%

Recall score: 26.48%

F1 score: 63.77%

<=50K

Найбільш точним виявився SVM класифікатор з гаусовим ядром.

**Завдання 2.3.** Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

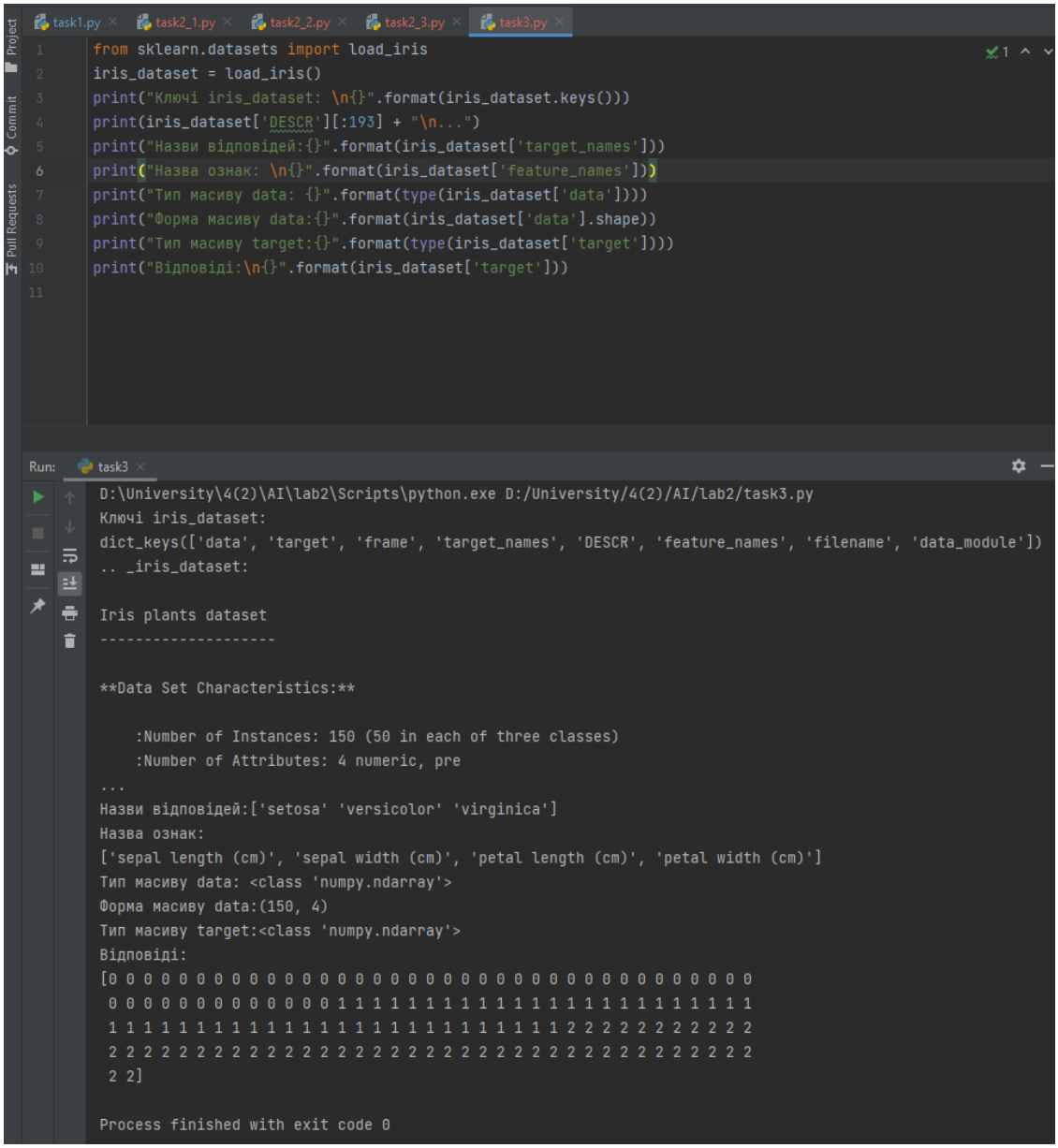


Рисунок 1. Код структури даних

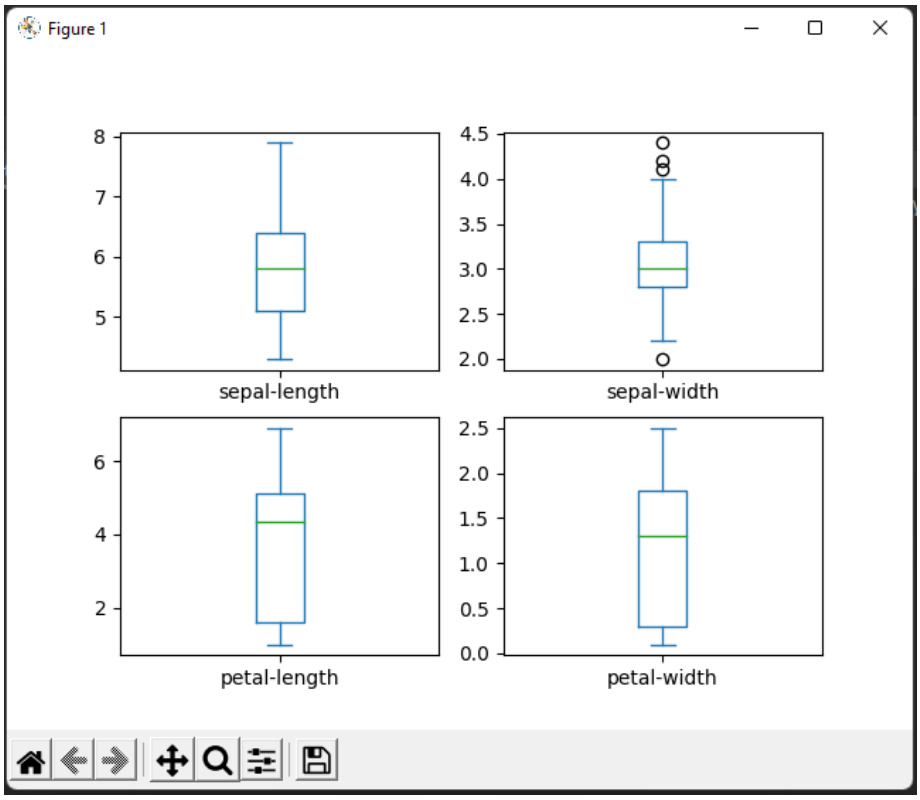


Рисунок 2. Одновимірні графіки

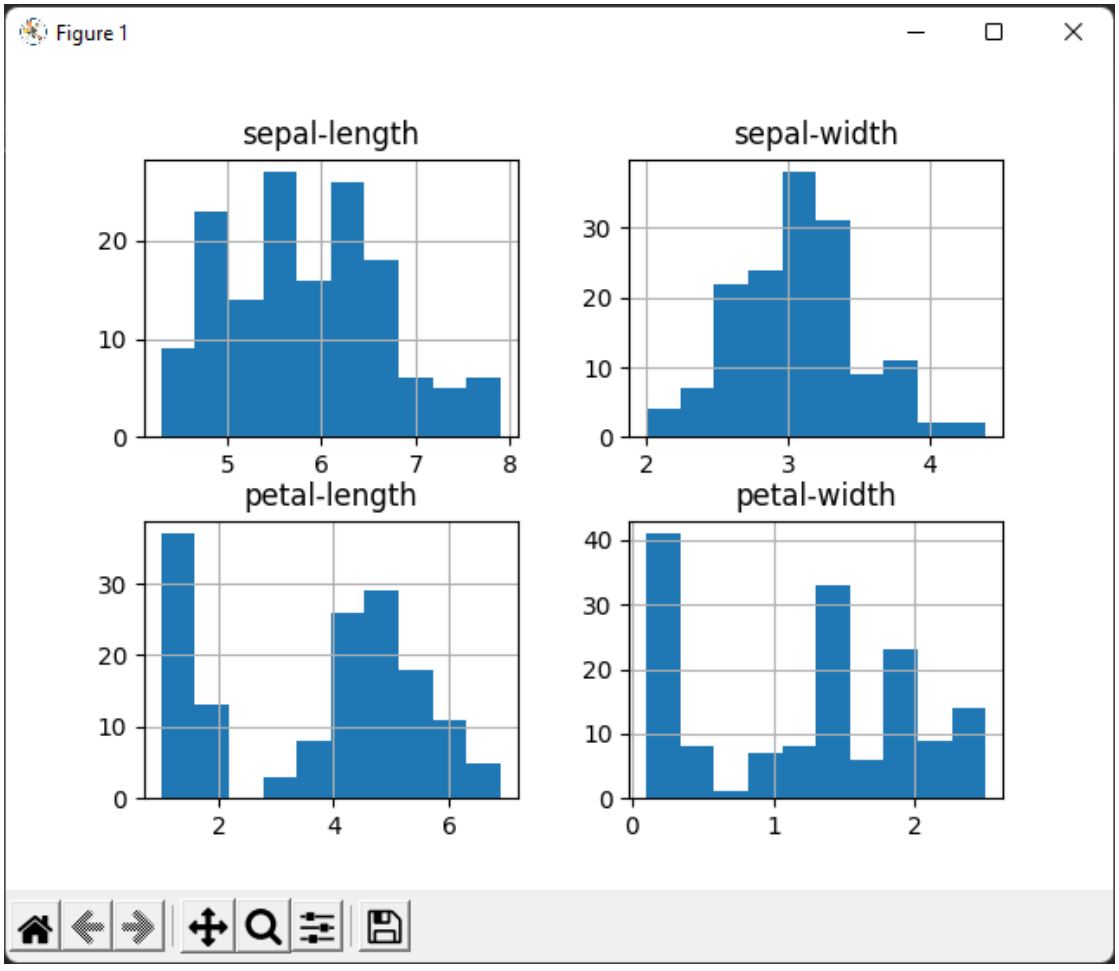


Рисунок 3. Діаграма розмаху атрибутів вхідних даних

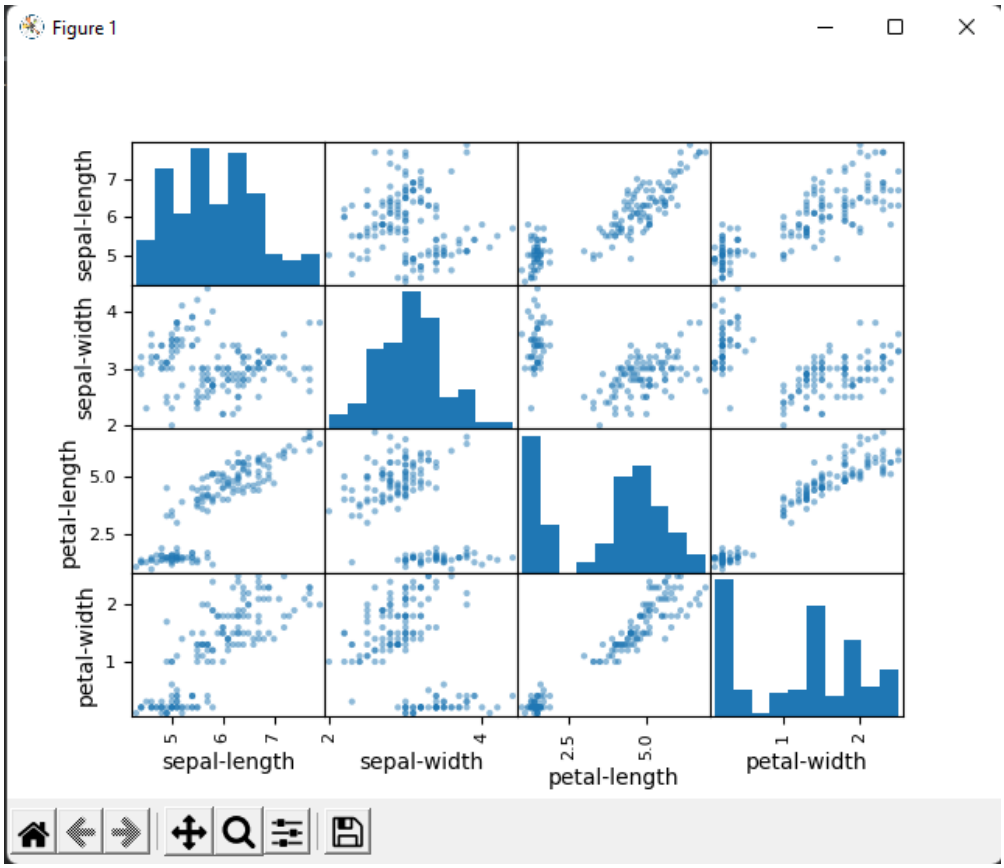


Рисунок 4. Матриця діаграм розсіювання

```
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)

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)
pyplot.show()

dataset.hist()
pyplot.show()

scatter_matrix(dataset)
pyplot.show()
```

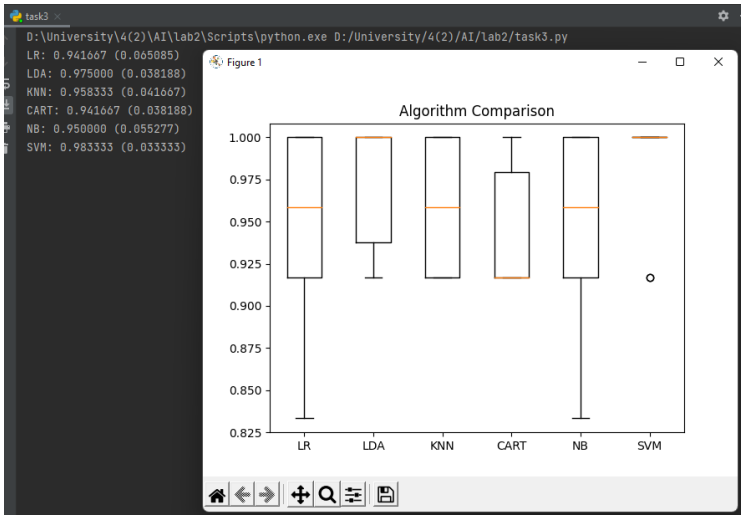


Рисунок 5. Порівняння алгоритмів

Проаналізувавши ортиманий графік, я обрав метод класифікації SVM, тому що він показав найвищу якість.

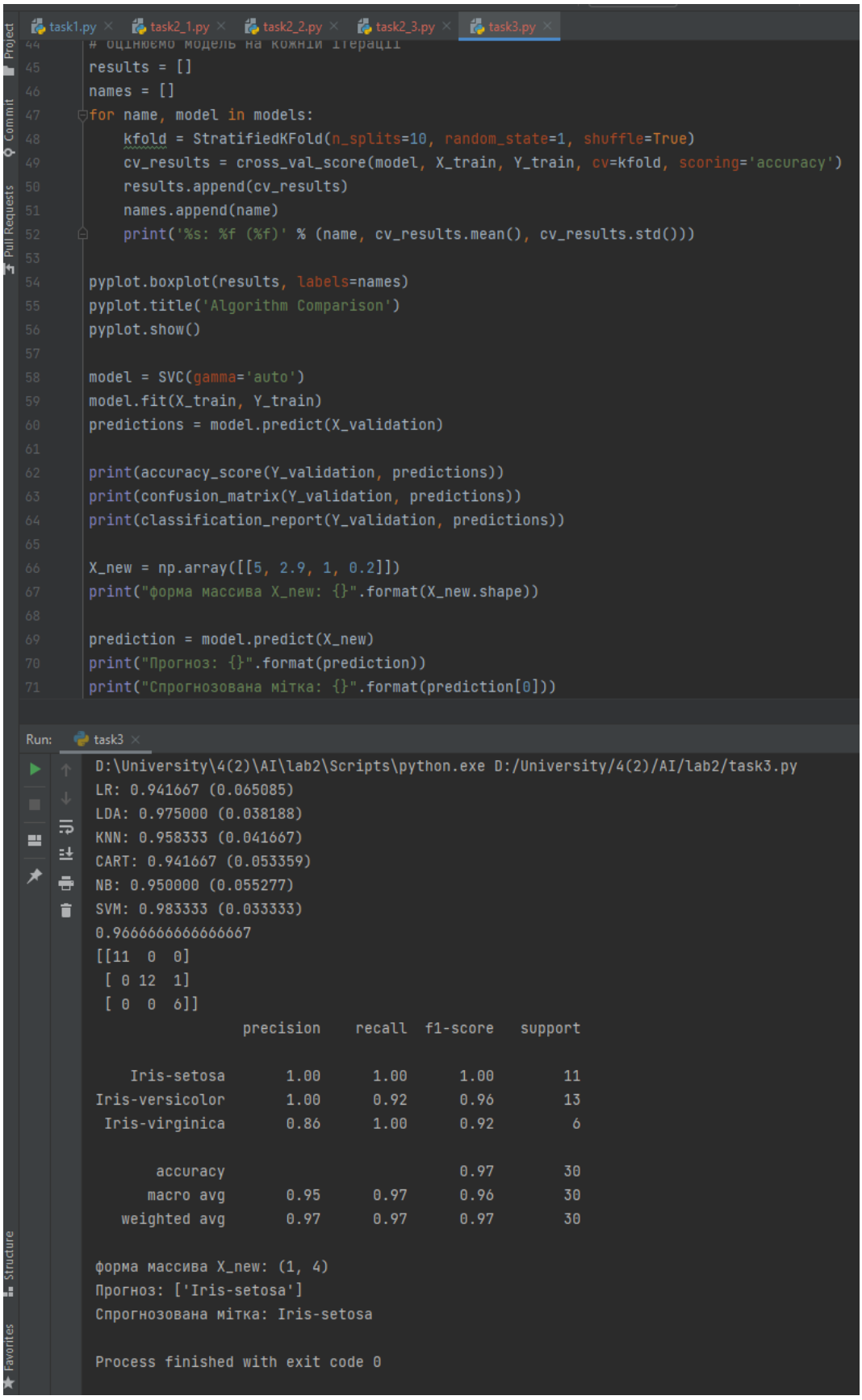


Рисунок 6. Результат

**Завдання 2.4.** Порівняння якості класифікаторів для набору даних завдання 2.1

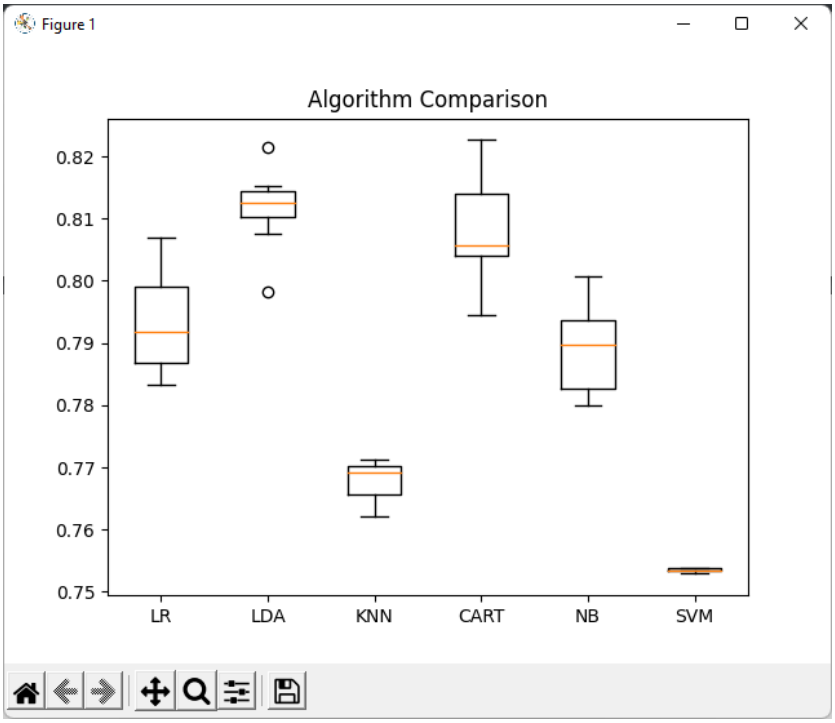


Рисунок 7. Порівняння алгоритмів

**Завдання 2.5.** Класифікація даних лінійним класифікатором Ridge

```
import matplotlib.pyplot as plt

iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_st
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)
ypred = clf.predict(Xtest)
from sklearn import metrics

print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(ytest, ypred, average=
'weighted'), 4))
print('Recall:', np.round(metrics.recall_score(ytest, ypred, average=
'weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average=
'weighted'), 4))
print('Cohen Kappa Score:',
      np.round(metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrcoef:',
      np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n',
      metrics.classification_report(ypred, ytest))

sns.set()
mat = confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label');
plt.savefig("Confusion.jpg")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format="svg")
```

```
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831
        Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рисунок 8. Класифікатор Ridge

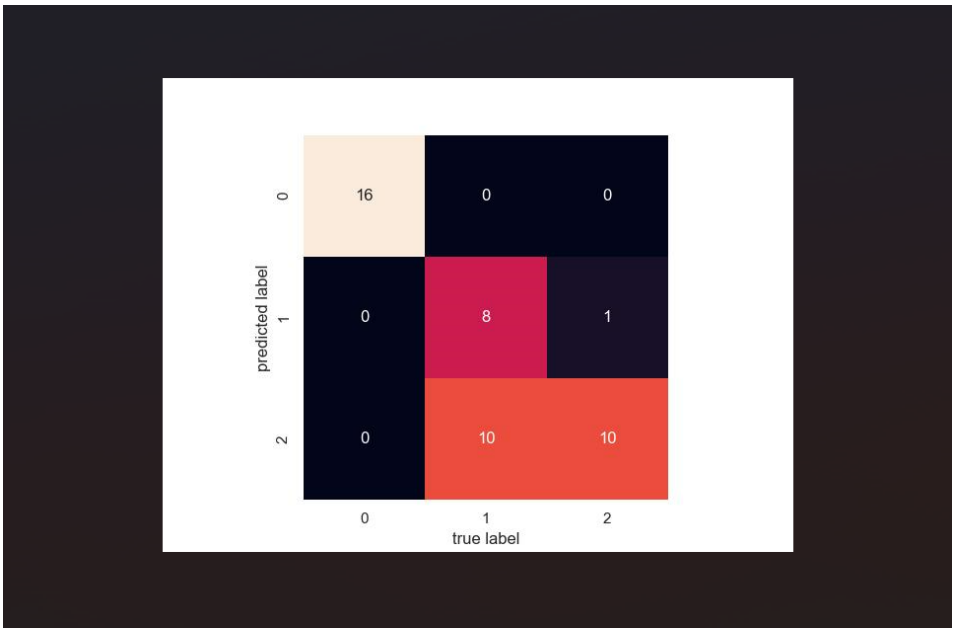


Рисунок 9. Confusion.jpg

Класифікатор має наступні параметри:

• tol – точність класифікації

• solver – алгоритм, який виконує класифікацію

На зображені Confusion.jpg наведені результати класифікації. На вертикальній шкалі відкладені наявні класи ірису в числовій репрезентації, а на горизонтальній передбаченя класи ірису. Цифра на перетині – кількість результатів системи при справжньому і передбаченому класі.

Коефіцієнт кореляції Метьюза – коефіцієнт, який на основі матриці помилок вираховує коефіцієнт від -1 до 1, де 1 – є результатом ідеальної класифікації, а 0 – рівень випадкового вибору.

Коефіцієнт Коена Каппа – коефіцієнт, який також за основу бере матрицю помилок, але замість загальної якості, звертає увагу на нерівноцінне розподілення класів.

**Висновок:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.