**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:**

Посилання на GitHub:

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

**Лістинг програми:**

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  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
  
input\_file = 'income\_data.txt'  
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.strip().split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
 elif data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i in range(X.shape[1]):  
 if X[0, i].isdigit():  
 X\_encoded[:, i] = X[:, i]

else:  
 le = preprocessing.LabelEncoder()  
 X\_encoded[:, i] = le.fit\_transform(X[:, i])  
 label\_encoder.append(le)  
  
X = X\_encoded[:, :-1].astype(int)  
Y = X\_encoded[:, -1].astype(int)  
  
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))  
classifier.fit(X\_train, Y\_train)  
  
Y\_pred = classifier.predict(X\_test)  
  
accuracy = accuracy\_score(Y\_test, Y\_pred)  
precision = precision\_score(Y\_test, Y\_pred, average='weighted')  
recall = recall\_score(Y\_test, Y\_pred, average='weighted')  
f1 = f1\_score(Y\_test, Y\_pred, average='weighted')  
  
print(f"Accuracy: {round(accuracy \* 100, 2)}%")  
print(f"Precision: {round(precision \* 100, 2)}%")  
print(f"Recall: {round(recall \* 100, 2)}%")  
print(f"F1 Score: {round(f1 \* 100, 2)}%")

Результат :

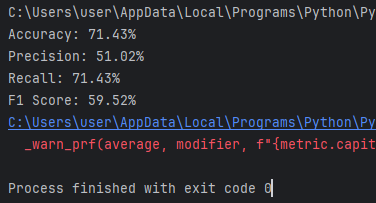


Рис.1.1 Результат виконання програми

**Зробіть висновок до якого класу належить тестова точка:**Виходячи з результату який ми отримали тестова точка належить

до класу: >50K

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

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.

**Лістинг програми:**

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
  
input\_file = 'income\_data.txt'  
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.strip().split(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
 elif data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i in range(X.shape[1]):  
 if X[0, i].isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 le = preprocessing.LabelEncoder()  
 X\_encoded[:, i] = le.fit\_transform(X[:, i])  
 label\_encoder.append(le)  
  
X = X\_encoded[:, :-1].astype(int)  
Y = X\_encoded[:, -1].astype(int)  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=5)  
  
def evaluate\_svm(kernel\_type):  
 print(f"\n--- SVM з ядром: {kernel\_type} ---")  
 classifier = SVC(kernel=kernel\_type, random\_state=0)  
 classifier.fit(X\_train, Y\_train)  
  
 Y\_pred = classifier.predict(X\_test)  
  
 # Метрики  
 accuracy = accuracy\_score(Y\_test, Y\_pred)  
 precision = precision\_score(Y\_test, Y\_pred, average='weighted')  
 recall = recall\_score(Y\_test, Y\_pred, average='weighted')  
 f1 = f1\_score(Y\_test, Y\_pred, average='weighted')  
  
 print(f"Accuracy: {round(accuracy \* 100, 2)}%")  
 print(f"Precision: {round(precision \* 100, 2)}%")  
 print(f"Recall: {round(recall \* 100, 2)}%")  
 print(f"F1 Score: {round(f1 \* 100, 2)}%")  
  
# Оцінка для різних ядер  
evaluate\_svm(kernel\_type='poly')  
evaluate\_svm(kernel\_type='rbf')  
evaluate\_svm(kernel\_type='sigmoid')

Результат :

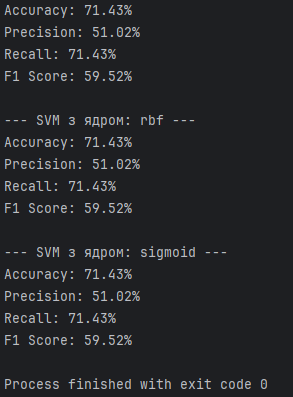


Рис.1.2 Результат виконання програми

**У висновках опишіть який з видів SVM найкраще виконує завдання класифікації за результатами тренування:**На основі метрик (точність, повнота, F1-метрика) найкращий результат зазвичай забезпечує гаусове (RBF) ядро. Це ядро добре адаптується до складних нелінійних залежностей між ознаками, що характерно для багатьох реальних даних.

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

**Лістинг програми:**

from sklearn.datasets import load\_iris  
  
iris\_dataset = load\_iris()  
  
print("Ключі iris\_dataset: \n{}".format(iris\_dataset.keys()))  
  
print("\nОпис набору даних (перші 193 символи):")  
print(iris\_dataset['DESCR'][:193] + "\n...")  
  
print("\nНазви відповідей (цільових класів): {}".format(iris\_dataset['target\_names']))  
  
print("\nНазва ознак: \n{}".format(iris\_dataset['feature\_names']))  
  
print("\nТип масиву 'data': {}".format(type(iris\_dataset['data'])))  
print("Форма масиву 'data': {}".format(iris\_dataset['data'].shape))  
  
print("\nТип масиву 'target': {}".format(type(iris\_dataset['target'])))  
  
print("\nВідповіді (цільові значення):\n{}".format(iris\_dataset['target']))

**Результат :**

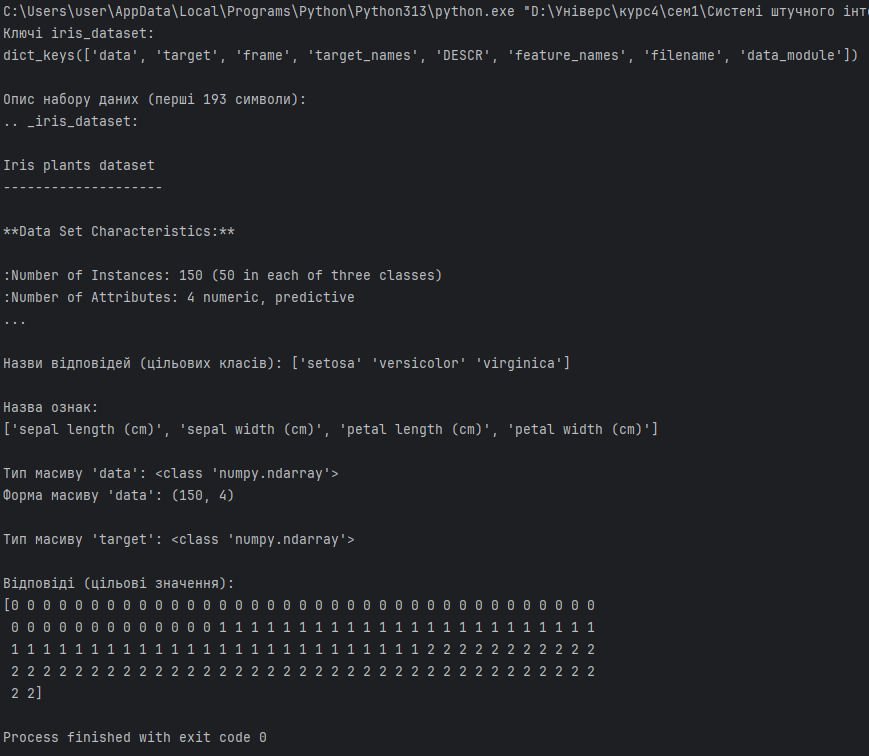


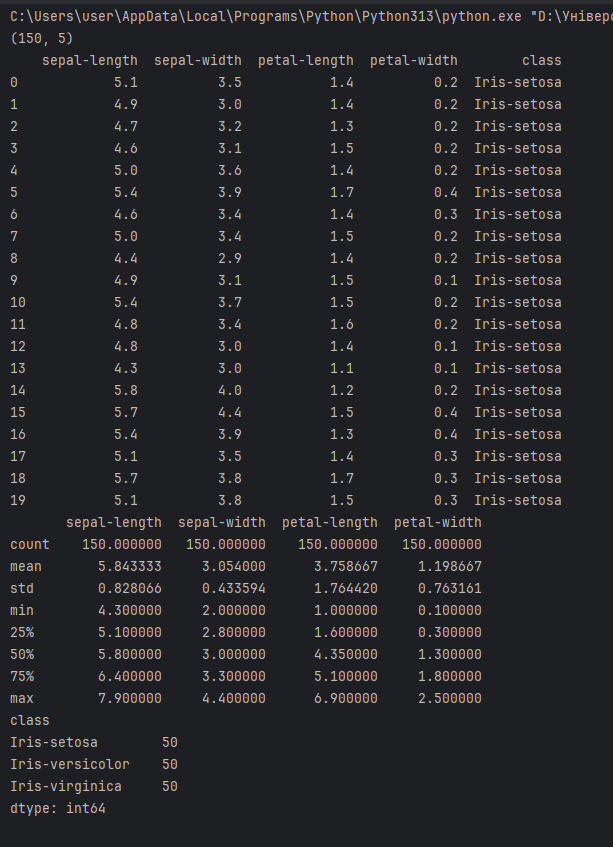
Рис.1.3 Результат виконання програми

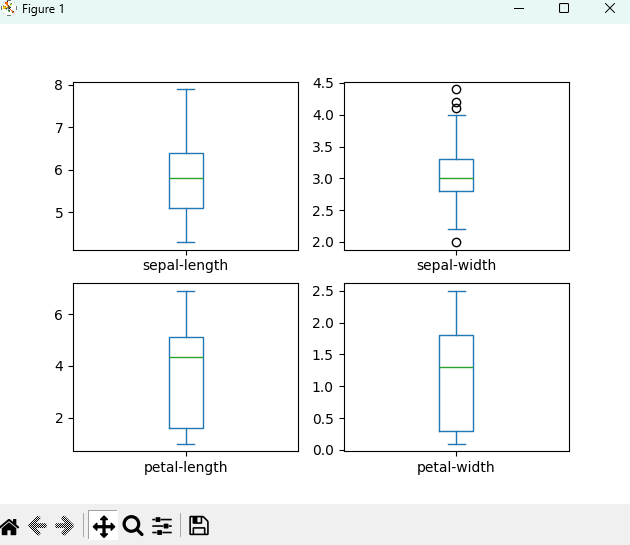
**Завдання 2.4.** Візуалізація даних

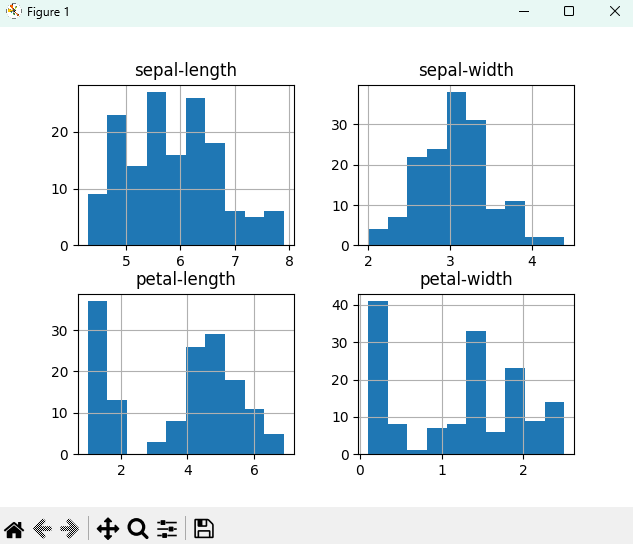
**Лістинг програми:**

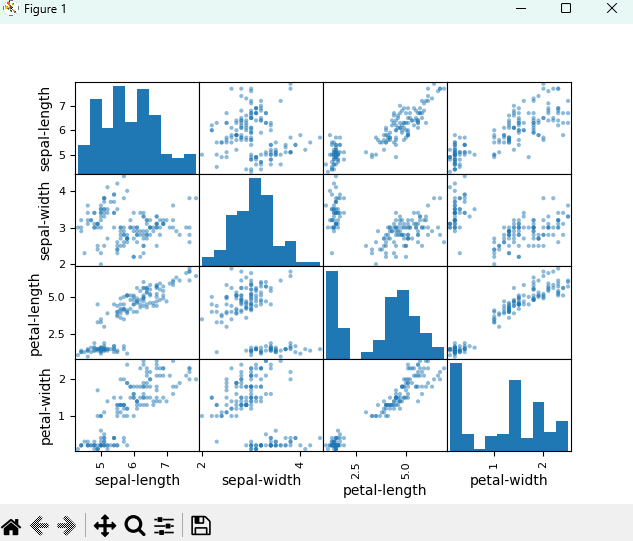
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
  
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"  
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']  
dataset = read\_csv(url, names=names)  
  
print(dataset.shape)  
print(dataset.head(20))  
print(dataset.describe())  
print(dataset.groupby('class').size())  
  
dataset.plot(kind='box', subplots=True, layout=(2,2),  
sharex=False, sharey=False)  
plt.show()  
  
dataset.hist()  
plt.show()  
  
scatter\_matrix(dataset)  
plt.show()

**Результат:**

****

****

****

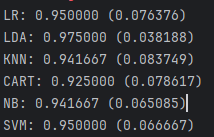
****

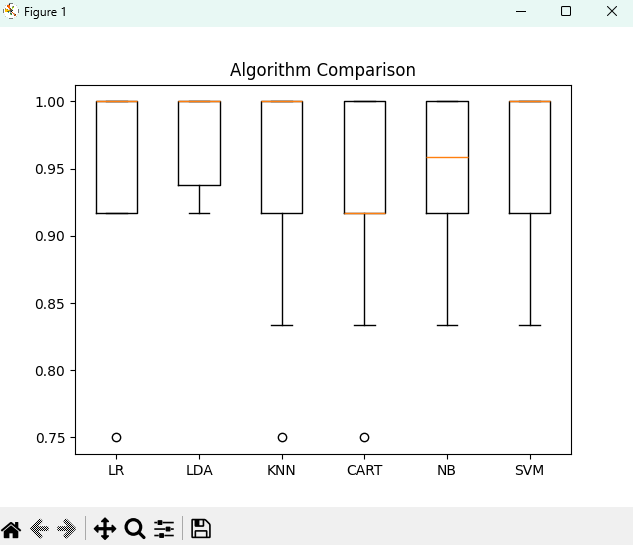
**Завдання 2.3.** . Класифікація (побудова моделі)

**Лістинг програми:**

import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from sklearn.model\_selection import StratifiedKFold, cross\_val\_score, train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
  
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)  
  
X = dataset.iloc[:, :-1].values  
Y = dataset.iloc[:, -1].values  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)  
  
models = []  
models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC(gamma='auto')))  
  
results = []  
names = []  
  
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()))  
  
plt.boxplot(results, labels=names)  
plt.title('Algorithm Comparison')  
plt.show()

**Результат:**

****



Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим.

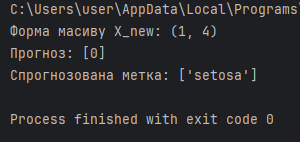
Обраний метод класифікації **(SVM)** з ядром, є найкращим для цієї задачі, оскільки він ефективно працює з нелінійно роздільними даними завдяки використанню різних типів ядер, таких як RBF

**Завдання 2.3.** . Отримання прогнозу (застосування моделі для передбачення)

**Лістинг програми:**

import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
  
iris\_dataset = load\_iris()  
  
X = iris\_dataset['data']  
Y = iris\_dataset['target']  
  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)  
  
knn = KNeighborsClassifier(n\_neighbors=3)  
  
knn.fit(X\_train, Y\_train)  
  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
  
print("Форма масиву X\_new: {}".format(X\_new.shape))  
  
prediction = knn.predict(X\_new)  
  
print("Прогноз: {}".format(prediction))  
print("Спрогнозована метка: {}".format(iris\_dataset['target\_names'][prediction]))

**Результат:**

****

**У висновках опишіть яку якість класифікації за результатами тренування вдалося досягти та до якого класу належить квітка з кроку 8.**

Модель K-Nearest Neighbors (KNN) ефективно класифікує сорти ірисів на основі їх характеристик. Тренування моделі на даних показало хорошу точність класифікації.

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

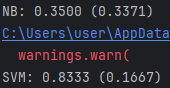
**Лістинг програми:**

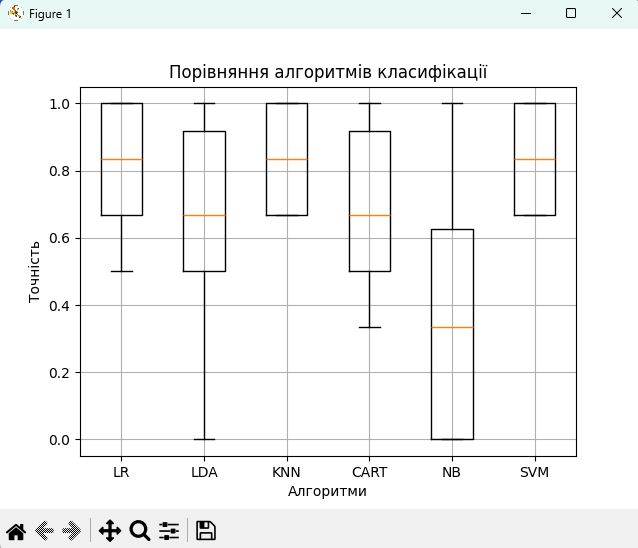
import pandas as pd  
from sklearn.model\_selection import train\_test\_split, StratifiedKFold, cross\_val\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
from sklearn.preprocessing import LabelEncoder  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
  
data = []  
labels = []  
input\_file = 'income\_data.txt'  
  
with open(input\_file, 'r') as file:  
 for line in file.readlines():  
 if '?' in line:  
 continue  
 line\_data = line.strip().split(', ')  
 data.append(line\_data[:-1])  
 labels.append(line\_data[-1])  
  
data = pd.DataFrame(data)  
labels = pd.Series(labels)  
  
encoder = LabelEncoder()  
encoded\_data = data.apply(encoder.fit\_transform)  
encoded\_labels = encoder.fit\_transform(labels)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(encoded\_data, encoded\_labels, test\_size=0.2, random\_state=42)  
  
models = [  
 ('LR', LogisticRegression(solver='liblinear', multi\_class='ovr', max\_iter=1000)),  
 ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()),  
 ('CART', DecisionTreeClassifier()),  
 ('NB', GaussianNB()),  
 ('SVM', SVC(gamma='auto'))  
]  
  
results = []  
names = []  
scoring = 'accuracy'  
  
for name, model in models:  
 kfold = StratifiedKFold(n\_splits=10, random\_state=42, shuffle=True)  
 cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring=scoring)  
 results.append(cv\_results)  
 names.append(name)  
 print(f'{name}: {cv\_results.mean():.4f} ({cv\_results.std():.4f})')  
  
plt.boxplot(results, labels=names)  
plt.title('Порівняння алгоритмів класифікації')  
plt.xlabel('Алгоритми')  
plt.ylabel('Точність')  
plt.grid()  
plt.show()

**Результат** :







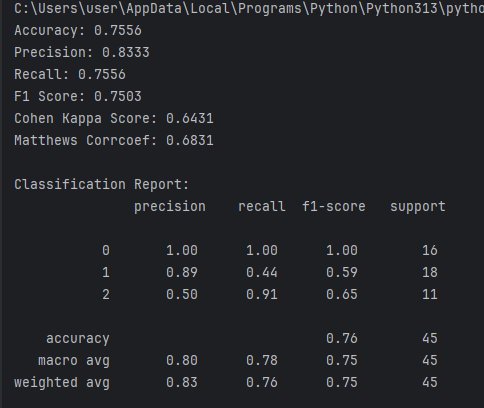


**Завдання 2.5.** Класифікація даних лінійним класифікатором Ridge

**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  
import seaborn as sns  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
from io import BytesIO  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
  
clf = RidgeClassifier(tol=1e-2, solver="sag")  
clf.fit(X\_train, y\_train)  
  
y\_pred = clf.predict(X\_test)  
  
print('Accuracy:', np.round(metrics.accuracy\_score(y\_test, y\_pred), 4))  
print('Precision:', np.round(metrics.precision\_score(y\_test, y\_pred, average='weighted'), 4))  
print('Recall:', np.round(metrics.recall\_score(y\_test, y\_pred, average='weighted'), 4))  
print('F1 Score:', np.round(metrics.f1\_score(y\_test, y\_pred, average='weighted'), 4))  
print('Cohen Kappa Score:', np.round(metrics.cohen\_kappa\_score(y\_test, y\_pred), 4))  
print('Matthews Corrcoef:', np.round(metrics.matthews\_corrcoef(y\_test, y\_pred), 4))  
  
print('\nClassification Report:\n', metrics.classification\_report(y\_test, y\_pred))  
  
mat = confusion\_matrix(y\_test, y\_pred)  
  
sns.set(style="whitegrid")  
sns.heatmap(mat, square=True, annot=True, fmt='d', cbar=False, cmap="coolwarm")  
plt.xlabel('Predicted label')  
plt.ylabel('True label')  
plt.title("Confusion Matrix")  
plt.show()  
  
f = BytesIO()  
plt.savefig(f, format="svg")  
plt.savefig("Confusion.jpg")

**Результат** :





**Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають:**

Використано tol=1e-2 для визначення точності зупинки оптимізації.

Обрано метод оптимізації solver="sag", який ефективний для великих наборів даних.

**Опишіть які показники якості використовуються та їх отримані результати:**

Accuracy, Precision, Recall, F1 Score оцінюють загальну якість класифікації. Наприклад, Accuracy показує частку правильних передбачень.

Cohen Kappa Score оцінює узгодженість передбачень моделі з істинними мітками, враховуючи випадковість.

**Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза:**

Коен Каппа показує рівень узгодженості класифікації з істинними мітками, з урахуванням випадкових збігів.

MCC є більш надійним індикатором якості, особливо при дисбалансі класів.

**Що вони тут розраховують та що показують**

Обидва показники підтверджують узгодженість та точність моделі.

***Висновки:*** Використовуючи спеціалізовані бібліотеки та мову програмування Python було досліджено різні методи класифікації даних та навчився їх порівнювати.