**ЗВІТ**

**З лабораторної роботи №2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

(навчальна дисципліна «СУЧАСНИЙ ШТУЧНИЙ ІНТЕЛЕКЕТ»)

Студента КН-20-1 навчальної групи

Кірія Даніли Олеговича варіант №6

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

**Завдання 1.**

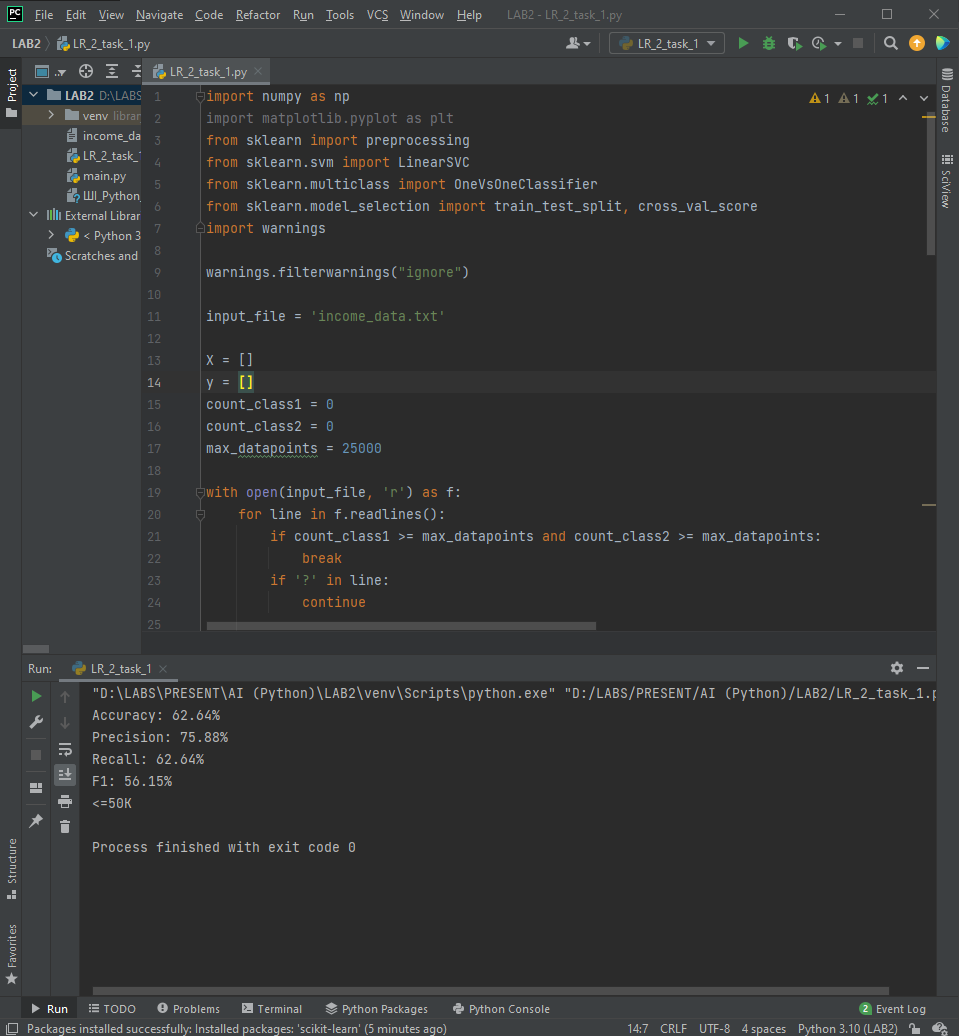
Ознаки з набору даних:

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Type** | **Description** |
| age | Integer | Вік |
| workclass | Categorical | Private, Self-emp-not-inc, Selfemp-inc, Federal-gov, Localgov, State-gov, Without-pay, Never-worked |
| education | Categorical | Bachelors, Some-college, 11th, HS-grad, Prof-school, Assocacdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool |
| education-num | Integer | Рівень освіти |
| marital-status | Categorical | Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouseabsent, Married-AF-spouse |
| race | Categorical | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black |
| relationship | Categorical | Wife, Own-child, Husband, Notin-family, Other-relative, Unmarried |

|  |  |  |
| --- | --- | --- |
| occupation | Categorical | Tech-support, Craft-repair, Other-service, Sales, Execmanagerial, Prof-specialty, Handlers-cleaners, Machine-opinspct, Adm-clerical, Farmingfishing, Transport-moving, Privhouse-serv, Protective-serv, Armed-Forces |
| sex | Binary | Female, Male |
| capital-gain | Integer | Прибуток |
| capital-loss | Integer | Витрати |
| hours-per-week | Integer | Години роботи |
| native-country | Categorical | United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(GuamUSVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands |
| income | Binary | >50K, <=50K |

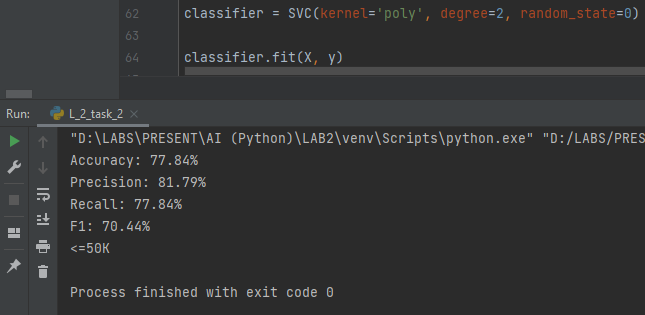
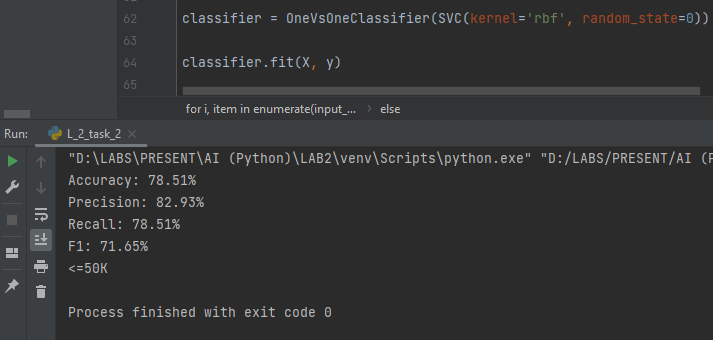
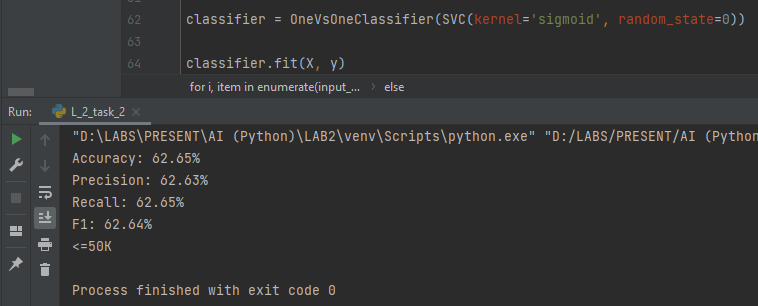
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
import warnings  
  
warnings.filterwarnings("ignore")  
  
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[:-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  
  
X = np.array(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
classifier.fit(X, y)  
  
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\_test\_pred = classifier.predict(X\_test)  
  
num\_folds = 3  
accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=num\_folds)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=num\_folds)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=num\_folds)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")  
f1\_values = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=num\_folds)  
print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")  
  
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 = [-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([input\_data[i]])[0])  
 count += 1  
  
input\_data\_encoded = np.array(input\_data\_encoded)  
  
predicted\_class = classifier.predict([input\_data\_encoded])  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Код програми



Результат виконання програми

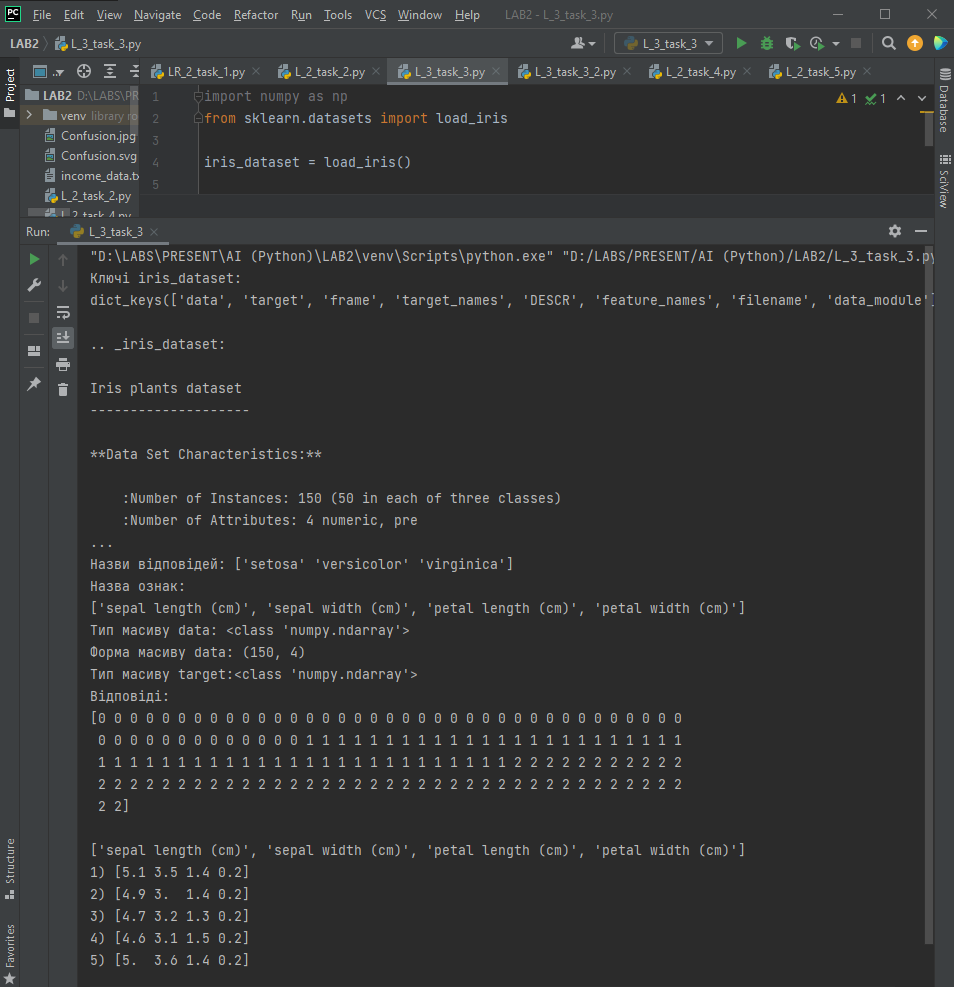
**Завдання 2.**

****  

**Завдання 3.**

import numpy as np  
from sklearn.datasets import load\_iris  
  
iris\_dataset = load\_iris()  
  
print("Ключі iris\_dataset: \n{}\n".format(iris\_dataset.keys()))  
print(iris\_dataset['DESCR'][:193] + "\n...")  
print("Назви відповідей: {}".format(iris\_dataset['target\_names']))  
print("Назва ознак: \n{}".format(iris\_dataset['feature\_names']))  
print("Тип масиву data: {}".format(type(iris\_dataset['data'])))  
print("Форма масиву data: {}".format(iris\_dataset['data'].shape))  
print("Тип масиву target:{}".format(type(iris\_dataset['target'])))  
print("Відповіді:\n{}\n".format(iris\_dataset['target']))  
  
print(format(iris\_dataset['feature\_names']))  
for i in range(5):  
 print('{}'.format(i+1) + ") " + "{}".format(iris\_dataset['data'][i]))

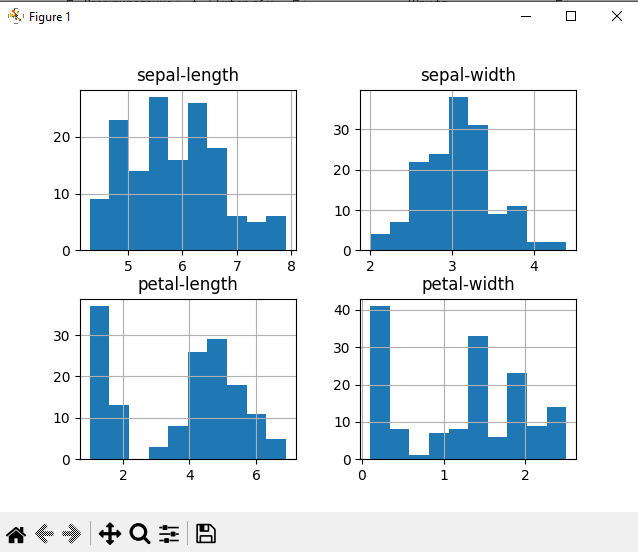
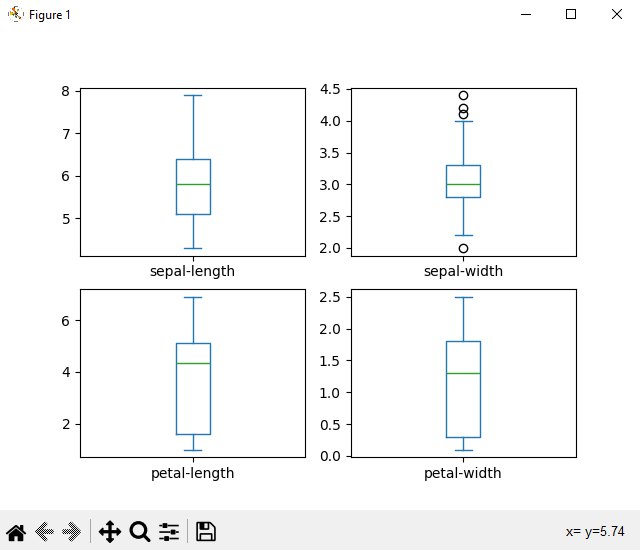
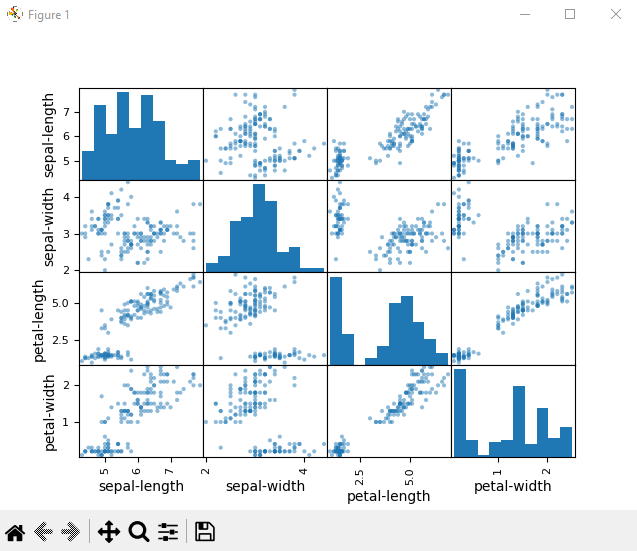
Код програми



Результат виконання програми

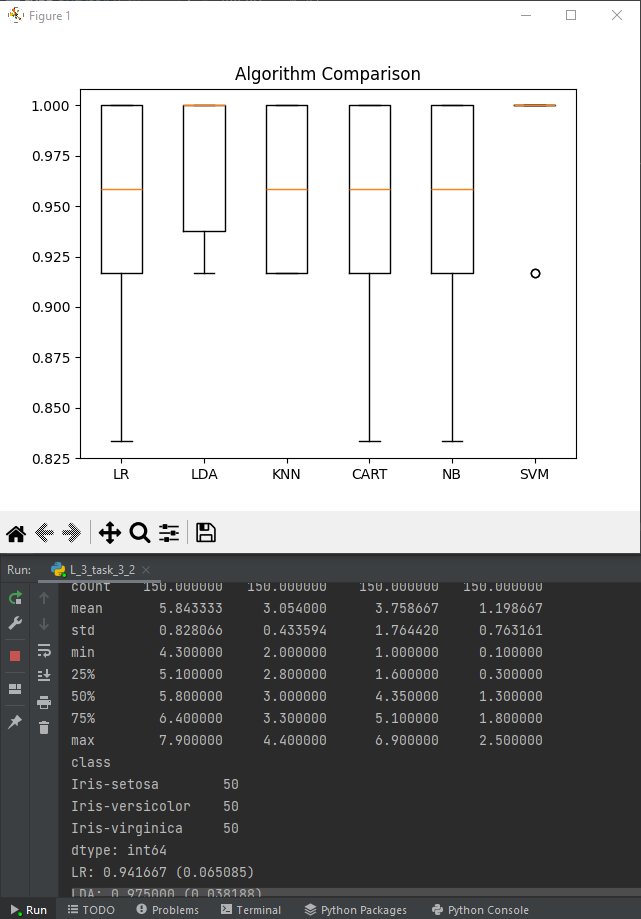
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
import matplotlib  
import matplotlib.pyplot as pyplot  
from sklearn.model\_selection import train\_test\_split  
  
matplotlib.use('TkAgg')  
  
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"  
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)  
pyplot.show()  
dataset.hist()  
pyplot.show()  
  
array = dataset.values  
X = array[:, 0:4]  
y = array[:, 4]  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
  
scatter\_matrix(dataset)  
pyplot.show()

Код програми(1 – 3)

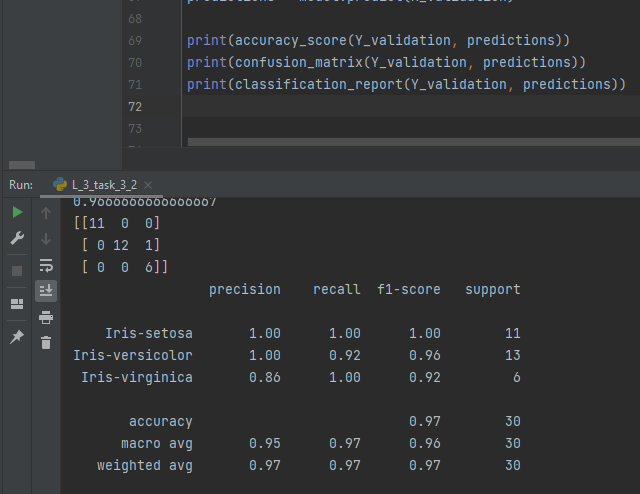
models = []  
models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', 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()))  
  
pyplot.boxplot(results, labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()

Код програми(4)

 Серед методів класифікації, розглянутих у нашому дослідженні, метод опорних векторів (SVM) виявився найбільш оптимальним. Цей метод продемонстрував найвищу середню точність та мінімальне стандартне відхилення в порівнянні з іншими методами. Крім того, слід відзначити, що метод опорних векторів відомий своєю здатністю ефективно працювати з різними типами даних і успішно справлятися з складними завданнями класифікації.

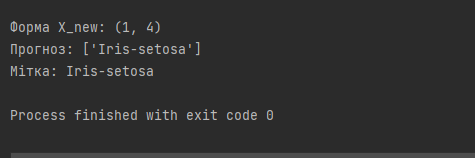
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))

Код програми(6-7)



import numpy as np  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма X\_new:", X\_new.shape)  
prediction = model.predict(X\_new)  
print("Прогноз: {}".format(prediction))  
predicted\_class = prediction[0]  
print("Мітка: {}".format(predicted\_class))

Код програми(8)

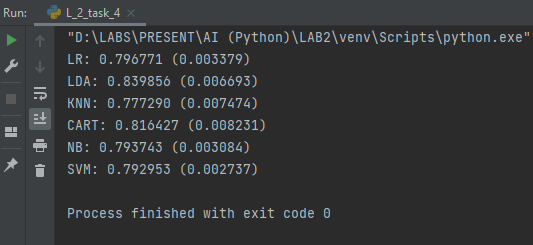


На підставі аналізу точності, матриці помилок та звіту про класифікацію можна зробити припущення, що досягнута висока ефективність класифікації на наборі даних Iris. За результатами аналізу можна стверджувати, що квітка, представлена восьмим прикладом, відноситься до виду Iris-setosa.

**Завдання 4.**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
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'  
dataset = pd.read\_csv(input\_file, sep=',', header=None, names=[  
 'Age', 'Workclass', 'fnlwgt', 'Education', 'Education\_Num', 'Marital\_Status',  
 'Occupation', 'Relationship', 'Race', 'Sex', 'Capital\_Gain', 'Capital\_Loss',  
 'Hours\_Per\_Week', 'Native\_Country', 'Income'  
])  
  
dataset\_encoded = pd.get\_dummies(dataset, columns=[  
 'Workclass', 'Education', 'Marital\_Status',  
 'Occupation', 'Relationship', 'Race', 'Sex', 'Native\_Country'  
])  
  
X = dataset\_encoded.drop('Income', axis=1)  
y = dataset\_encoded['Income']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
  
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='scale')))  
  
results = []  
names = []  
for name, model in models:  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 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()))

Код програми

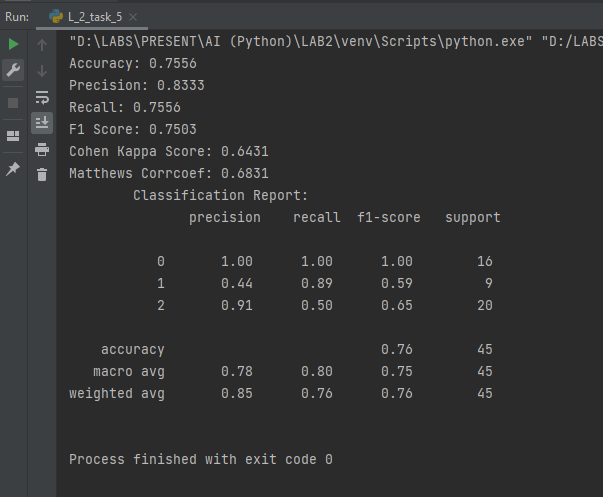


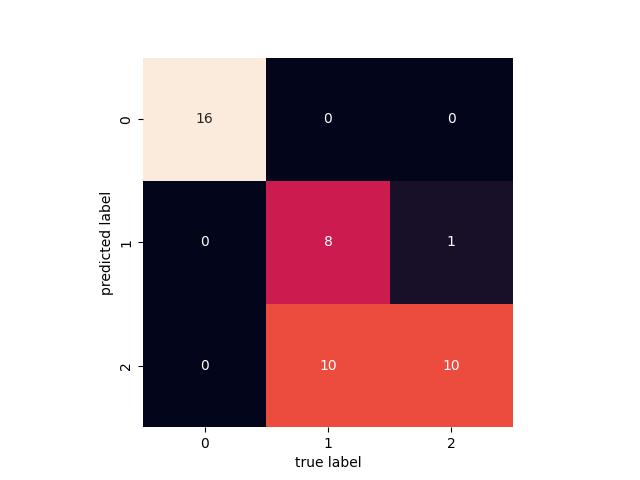
За отриманими результатами можна визначити, що в даному контексті найкращою опцією є метод аналізу лінійних дискримінантів (LDA). Ця модель ефективно враховує взаємозв'язок між ознаками та цільовою змінною. Крім того, стандартне відхилення для LDA є низьким, що свідчить про стабільність цієї моделі, і, важливо відзначити, що її розрахунок не потребує значних обчислювальних ресурсів.

**Завдання 5.**

import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import RidgeClassifier  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
import seaborn as sns  
import matplotlib  
import matplotlib.pyplot as plt  
  
matplotlib.use('TkAgg')  
  
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)  
  
ypred = clf.predict(X\_test)  
  
print('Accuracy:', np.round(metrics.accuracy\_score(y\_test, ypred), 4))  
print('Precision:', np.round(metrics.precision\_score(y\_test, ypred, average='weighted'), 4))  
print('Recall:', np.round(metrics.recall\_score(y\_test, ypred, average='weighted'), 4))  
print('F1 Score:', np.round(metrics.f1\_score(y\_test, ypred, average='weighted'), 4))  
print('Cohen Kappa Score:', np.round(metrics.cohen\_kappa\_score(y\_test, ypred), 4))  
print('Matthews Corrcoef:', np.round(metrics.matthews\_corrcoef(y\_test, ypred), 4))  
print('\t\tClassification Report:\n', metrics.classification\_report(ypred, y\_test))  
  
  
mat = confusion\_matrix(y\_test, 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")  
  
f = BytesIO()  
plt.savefig(f, format="svg")  
  
plt.savefig("Confusion.svg")

Код програми





Метод Ridge використовується для класифікації даних і має параметри, які впливають на його роботу. Параметр "tol=1e-2" визначає точність обчислення і використовується у випадках, коли оптимізаційний алгоритм повинен завершити свою роботу. Параметр "solver="sag"" вказує на метод оптимізації для класифікатора Ridge, де "sag" означає Stochastic Average Gradient Descent.

Щодо показників якості:

* Accuracy визначає відсоток правильних передбачень у порівнянні з загальною кількістю прикладів, і в цьому випадку вона становить 75%.
* Precision вимірює точність передбачень позитивних класів і складає 83%.
* Recall вимірює здатність моделі виявляти всі позитивні приклади, і його значення також становить 75%.
* F1 Score представляє гармонічне середнє між точністю і повнотою, і в даному випадку він становить 75%.

Матриця заплутаності "Confusion.jpg" вказує на кількість правильних і неправильних класифікацій для кожного класу. На головній діагоналі знаходиться кількість правильних класифікацій для кожного класу, і поза головною діагоналлю показана кількість неправильних класифікацій для кожного класу.

Коефіцієнт Коена Каппа (Cohen Kappa Score) вимірює ступінь узгодженості між реальними та передбаченими мітками, враховуючи можливість випадкового вибору. Значення Каппа у цьому випадку становить 0.6431, що вказує на помірний рівень узгодженості між фактичними та передбаченими класами.

Коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) подібний до Коефіцієнта Коена Каппа, але більше враховує дисбаланс класів у вибірці. Зазвичай використовується для бінарної класифікації, і в даному випадку його значення становить 0.6831, що свідчить про добрий рівень узгодженості між фактичними та передбаченими класами.

***Висновки:*** в ході виконання лабораторної роботи було, використовуючи спеціалізовані бібліотеки та мову програмування Python, досліджено різні методи класифікації даних та отримано практичні навички в їх порівнянні.

GitHub: https://github.com/invicibleee/Artificial-intelligence.git