**Лабораторна робота № 2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

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

Github: <https://github.com/igogol-hie/myrepoz987/tree/main/SAI-LR-2>

**Хід роботи**

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

Створіть класифікатор у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 ознак (атрибутів). Метою є з'ясування умов, за яких щорічний прибуток людини перевищує $50000 або менше цієї величини за допомогою бінарної класифікації.

Випишіть у звіт всі 14 ознак з набору даних – їх назви, що вони позначають та вид (числові чи категоріальні).

Обчисліть значення інших показників якості класифікації (акуратність, повнота, точність) та разом з F1 занесіть їх у звіт. (Див. ЛР-1).

Зробіть висновок до якого класу належить тестова точка.

14 ознак набору даних:

age – позначає вік, числові;

workclass – позначає робочий клас, категоріальні;

fnlwgt – числові;

education – категоріальні;

education-num – числові;

marital-status – категоріальні;

occupation – категоріальні;

relationship – категоріальні;

race – категоріальні;

sex – категоріальні;

capital-gain – числові;

capital-loss – числові;

hours-per-week – числові;

native-country – категоріальні.

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  
  
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(', ')  
 income\_class = data[-1]  
 if income\_class == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if income\_class == '>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:  
 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, random\_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)) + "%")  
  
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:  
 encoder = label\_encoder[count]  
 input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])  
 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])

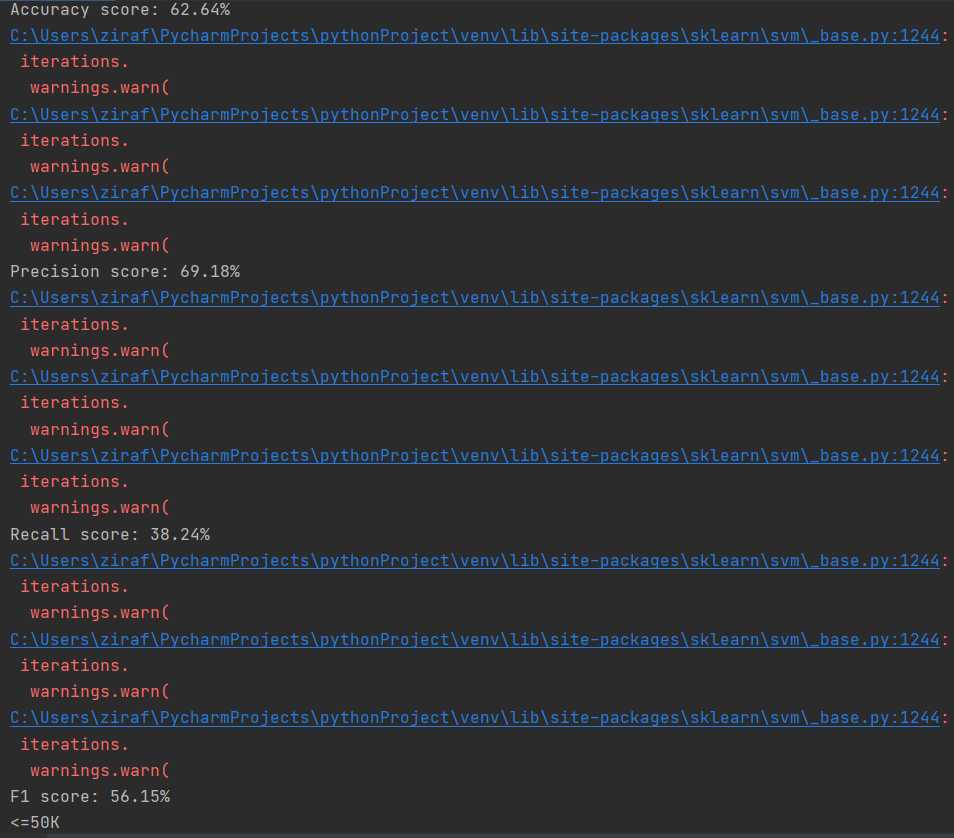


Рис. 1 Результат виконання програми

Висновок: тестова точка належить до класу <=50k, тобто людина заробляє менше або рівно 50к.

**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  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_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[:-1].split(', ')  
 income\_class = data[-1]  
 if income\_class == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if income\_class == '>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:  
 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(SVC(kernel='poly', degree=8, max\_iter=5000))  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_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)) + "%")  
  
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:  
 encoder = label\_encoder[count]  
 input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])  
 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])

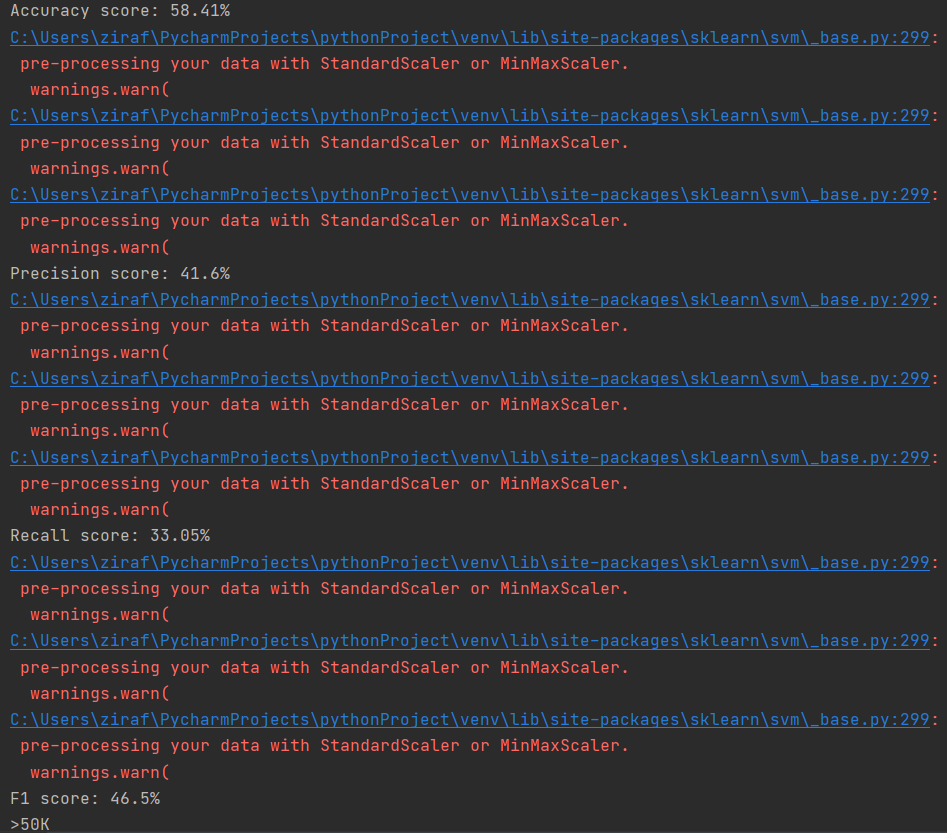


Рис. 2 Результат виконання програми

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_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[:-1].split(', ')  
 income\_class = data[-1]  
 if income\_class == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if income\_class == '>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:  
 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(SVC(kernel='rbf', max\_iter=10000))  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_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)) + "%")  
  
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:  
 encoder = label\_encoder[count]  
 input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])  
 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])

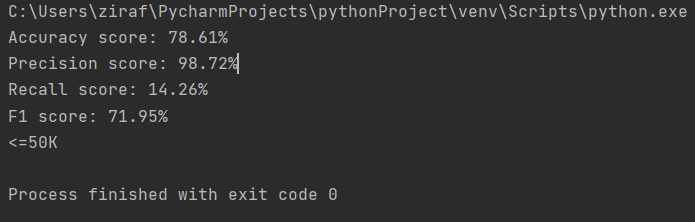


Рис. 3 Результат виконання програми

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_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[:-1].split(', ')  
 income\_class = data[-1]  
 if income\_class == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if income\_class == '>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:  
 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(SVC(kernel='sigmoid', max\_iter=10000))  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_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)) + "%")  
  
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:  
 encoder = label\_encoder[count]  
 input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])  
 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])

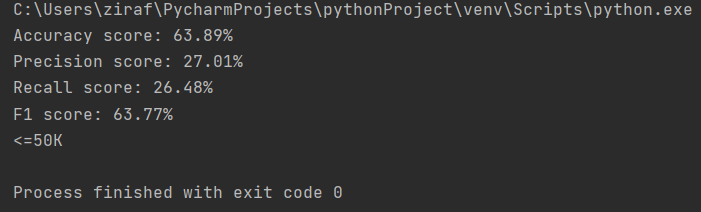


Рис. 4 Результат виконання програми

Висновок: за акуратністю і точністю найкращий вид класифікатора є нелінійній класифікатор SVM з гаусовим ядром, але за повнотою найкраще справився нелінійній класифікатор SVM з поліноміальним ядром. Загалом, найкраще виконує завдання класифікатор з гаусовим ядром.

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

Необхідно класифікувати сорти ірисів за деякими їх характеристиками: довжина та ширина пелюсток, а також довжина та ширина чашолистків.

Також, в наявності є вимірювання цих же характеристик ірисів, які раніше дозволили досвідченому експерту віднести їх до сортів: setosa, versicolor і virginica.

Використовувати класичний набір даних у машинному навчанні та статистиці - Iris. Він включений у модуль datasets бібліотеки scikit-learn.

Лістинг програми:

from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
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  
import numpy as np  
  
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()  
  
scatter\_matrix(dataset)  
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=0.2, random\_state=1)  
  
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()  
  
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))  
  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма массива X\_new: {}".format(X\_new.shape))  
  
prediction = model.predict(X\_new)  
print("Прогноз: {}".format(prediction))  
print("Спрогнозована мітка: {}".format(prediction[0]))

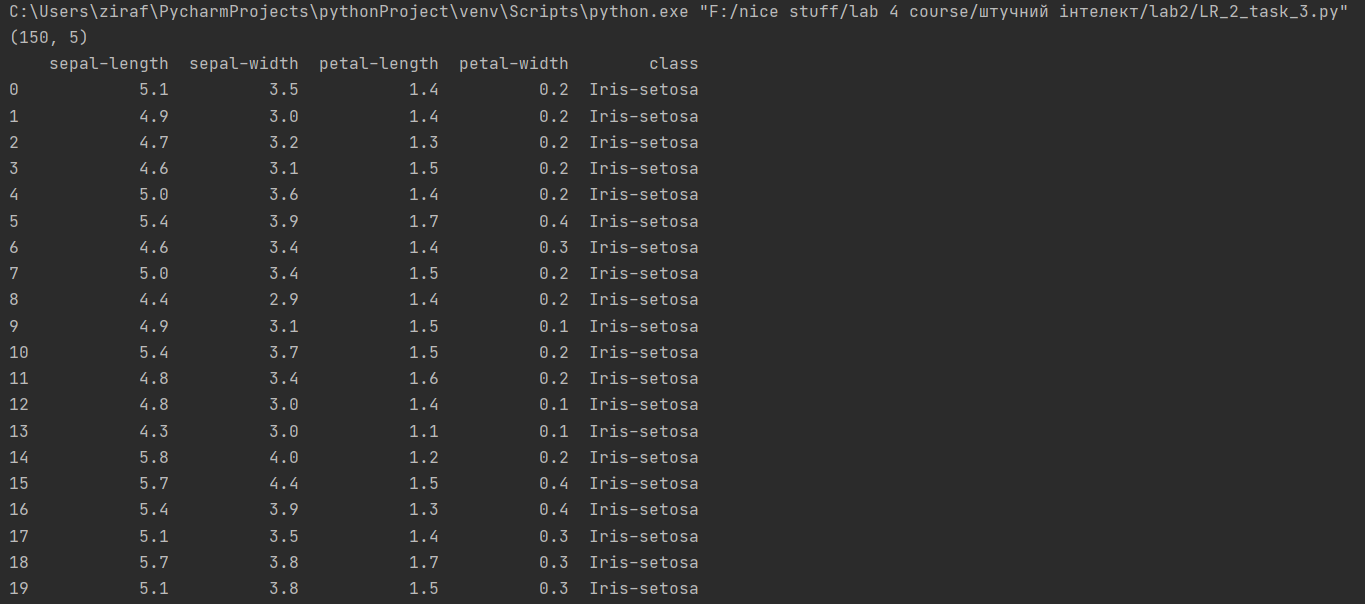


Рис. 5 Результат виконання програми

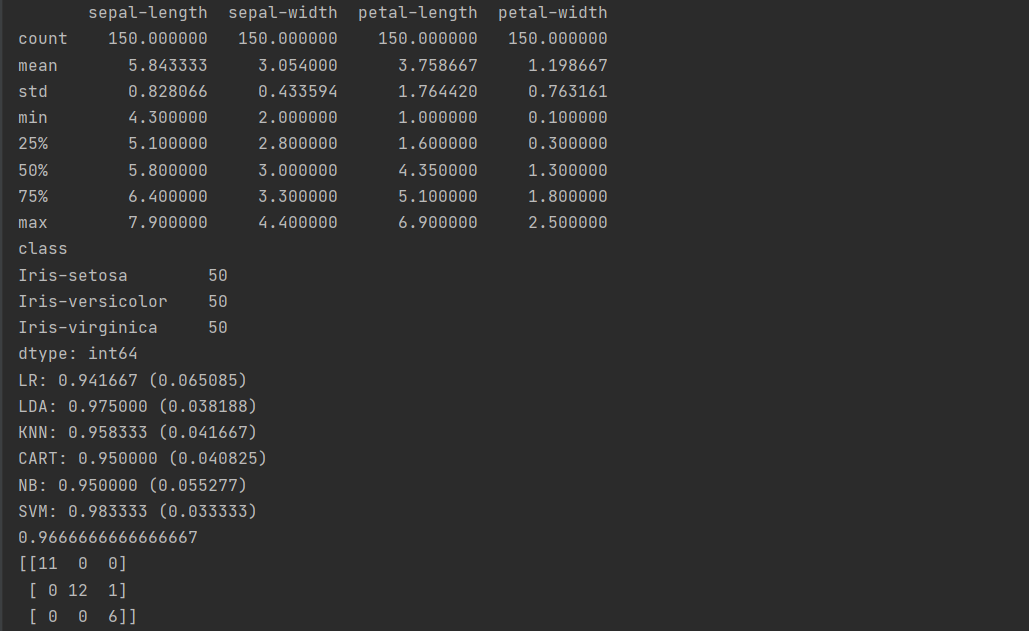


Рис. 6 Результат виконання програми

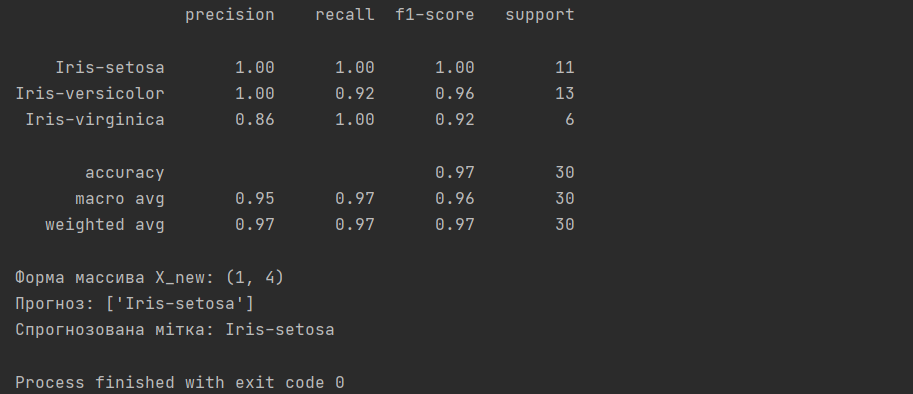


Рис. 7 Результат виконання програми

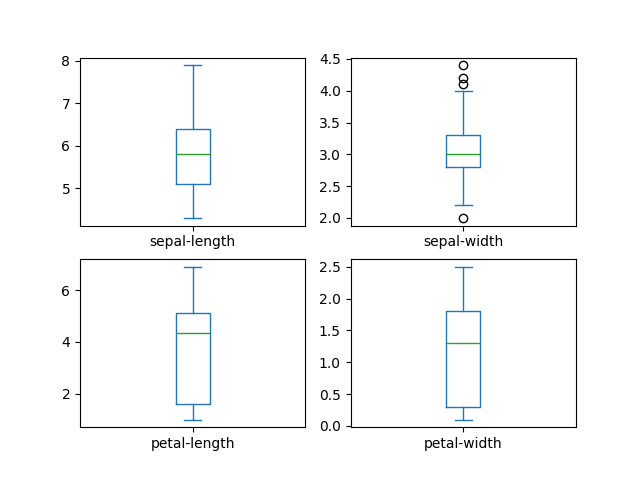


Рис. 8 Діаграма розмаху

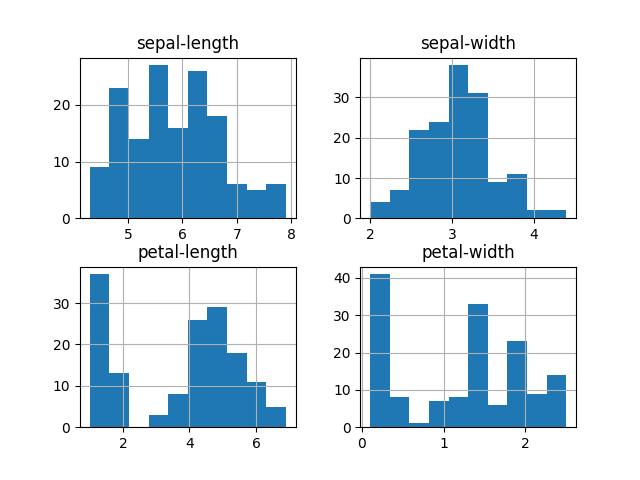


Рис. 9 Гістограма розподілу атрибутів датасета

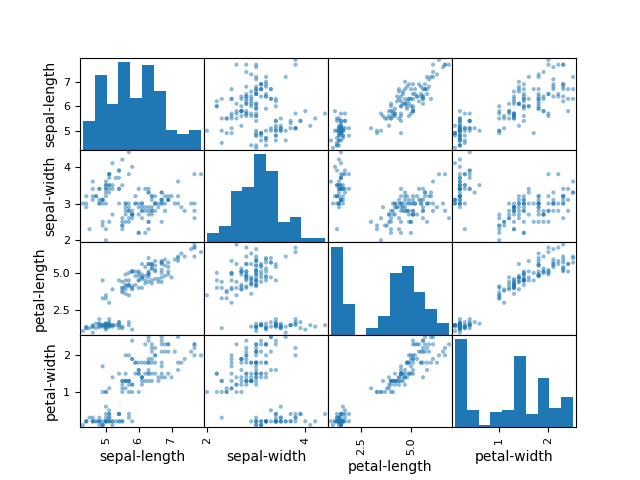


Рис. 10 Матриця діаграм розсіювання

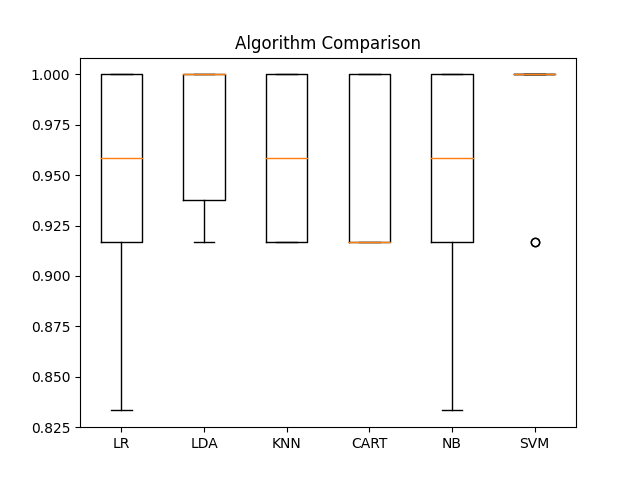


Рис. 11 Алгоритм порівняння

Висновок: Квітка належить до виду Iris-setosa. Було вибрано метод опорних векторів (SVM). Вдалося досягти 0.97 показника якості.

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

По аналогії із завданням 2.3 створіть код для порівняння якості класифікації набору даних income\_data.txt (із завдання 2.1) різними алгоритмами.

Використати такі алгоритми класифікації:

Логістична регресія або логіт-модель (LR)

Лінійний дискримінантний аналіз (LDA)

Метод k-найближчих сусідів (KNN)

Класифікація та регресія за допомогою дерев (CART)

Наївний баєсовський класифікатор (NB)

Метод опорних векторів (SVM)

Розрахуйте показники якості класифікації для кожного алгоритму. Порівняйте їх між собою. Оберіть найкращий для рішення задачі. Поясніть чому ви так вирішили у висновках до 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 pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
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  
  
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(', ')  
 income\_class = data[-1]  
 if income\_class == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if income\_class == '>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:  
 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)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
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', max\_iter=10000)))  
  
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()

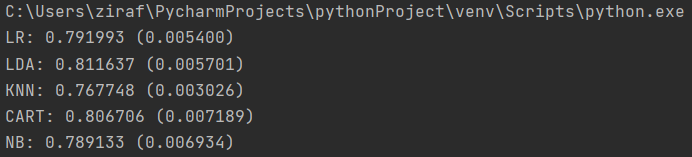




Рис. 12 Результат виконання програми

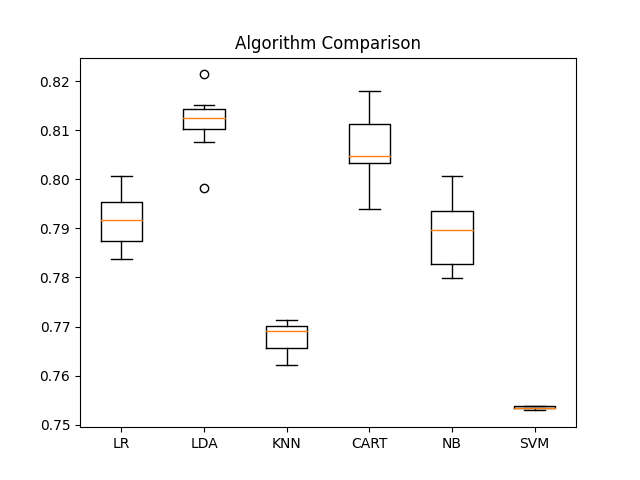


Рис. 13 Результат виконання програми

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

Виправте код та виконайте класифікацію. Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають. Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg. Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.

Лістинг програми:

import numpy as np  
from sklearn.datasets import load\_iris  
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.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
  
sns.set()  
  
  
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('\t\tClassification Report:\n', metrics.classification\_report(y\_pred, y\_test))  
  
mat = confusion\_matrix(y\_test, y\_pred)  
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")

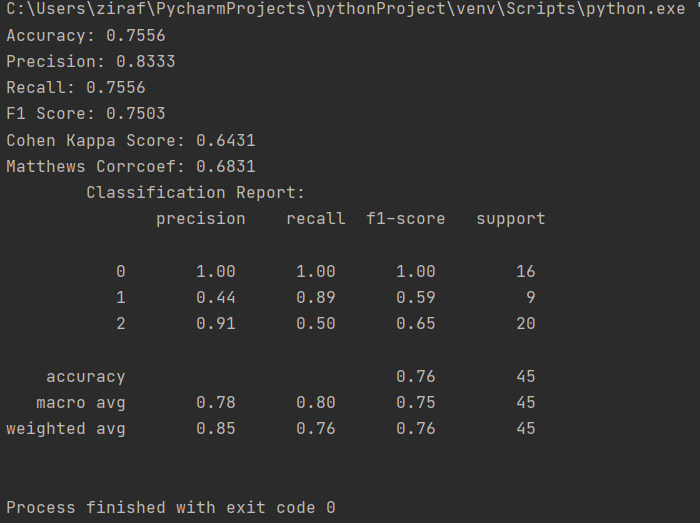


Рис. 14 Результат виконання програми

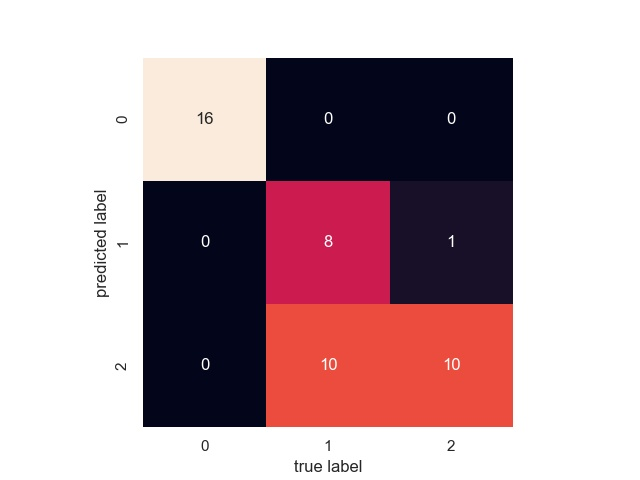


Рис. 15 Confusion.jpg

В класифікаторі Ridge були використані налаштування точності (tol=1e-2) та розв’язник Stochastic Average Gradient descent (solver="sag").  
 Показники якості, які використовувались – акуратність, точність, повнота, коефіцієнт Коена Каппа, коефіцієнт кореляції Метьюза.

На рис. 15 (Confusion.jpg) показана матриця confusion, як skicit-learn може навчатися класифікувати.

Коефіцієнт Коена Каппа – це статистика, яка вимірює міжрегіональну згоду якісних (категоріальних) предметів. Зазвичай вважається, що це надійніший захід, ніж простий розрахунок угоди про відсотки, оскільки k враховує випадкову угоду. Каппа Коена вимірює угоду між двома оцінювачами, кожен із яких класифікує N предметів на C взаємовиключних категорій.

В даному випадку коефіцієнт Коена Каппа (0.6431) показує істотну згоду.

Коефіцієнт кореляції Метьюза або коефіцієнт phi використовується в машинному навчанні як міра якості бінарних (двокласних) класифікацій, запроваджених біохіміком Браяном У. Метьюзом у 1975 році.

Не дивлячись на те, що акуратність точність і повнота в нас доволі високі, коефіцієнт кореляції Метьюза – 0.6831, тому що його оцінка висока в тих випадках, коли класифікатор справляється і з негативними, і з позитивними значеннями.

***Висновки:*** на даній лабораторній ми, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідили різні методи класифікації даних та навчилися їх порівнювати, порівняли різні види класифікатора SVM, проаналізували значення коефіцієнтів Коена Каппа і кореляції Метьюза, порівняли якості класифікаторів на основі класифікації ірисів.