Міністерство освіти і науки України

Державний університет ,,Житомирська політехніка”

Кафедра: Фікт

Група: ІПЗ-20-1

# Лабораторна робота №2

«ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ»

Виконав: Денисюк Н. Г.

Прийняв: Голенко М. Ю.

**Мета роботи:** використовуючи спеціалізовані бібліотеки та мову

програмування Python дослідити різні методи класифікації даних та

навчитися їх порівнювати.

**Виконання роботи:**

**Завдання №1:** Обчисліть значення інших показників якості класифікації

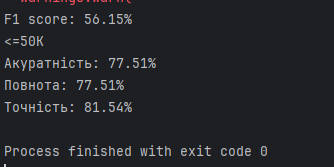
(акуратність, повнота, точність) та разом з F1 занесіть їх у звіт. (Див.

ЛР-1).

Код:

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  
from sklearn.model\_selection import 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:  
 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)  
  
# Створення SVМ-класифікатора  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
# Навчання класифікатора  
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)  
  
# Обчислення F-міри для SVМ-класифікатора  
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', 'Handlers-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([item])[0])  
 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.reshape(1, -1))  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])  
  
  
from sklearn.metrics import accuracy\_score, recall\_score, precision\_score  
  
# Обчислення показників для тестового набору даних  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
recall = recall\_score(y\_test, y\_test\_pred, average='weighted')  
precision = precision\_score(y\_test, y\_test\_pred, average='weighted')  
  
print(f"Акуратність: " + str(round(100\*accuracy, 2)) + "%")  
print(f"Повнота: " + str(round(100\*recall, 2)) + "%")  
print(f"Точність: " + str(round(100\*precision, 2)) + "%")

**Результат виконання:**

****

**Завдання №2:** 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.

**Поліноміальне ядро:**

Код:

import numpy as np  
from sklearn.svm import SVC  
from sklearn import preprocessing  
from sklearn.metrics import accuracy\_score, recall\_score, precision\_score  
from sklearn.model\_selection import train\_test\_split  
  
  
# 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:  
 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)  
  
# --------------------------------  
  
# Розділення даних на навчальні та тестові набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Створення SVM-класифікатора з поліноміальним ядром  
classifier\_poly = SVC(kernel='poly', degree=8, random\_state=0)  
  
# Навчання класифікатора  
classifier\_poly.fit(X\_train, y\_train)  
  
# Обчислення показників якості  
y\_test\_pred\_poly = classifier\_poly.predict(X\_test)  
accuracy\_poly = accuracy\_score(y\_test, y\_test\_pred\_poly)  
recall\_poly = recall\_score(y\_test, y\_test\_pred\_poly, average='weighted')  
precision\_poly = precision\_score(y\_test, y\_test\_pred\_poly, average='weighted')  
  
# Виведення показників якості  
print("Поліноміальне ядро:")  
print(f"Акуратність: {accuracy\_poly:.2f}")  
print(f"Повнота: {recall\_poly:.2f}")  
print(f"Точність: {precision\_poly:.2f}")

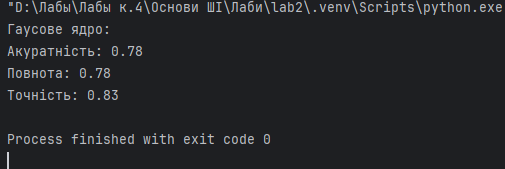
**Результат виконання:**

**Гаусове ядро:**

Код:

import numpy as np  
from sklearn.svm import SVC  
from sklearn import preprocessing  
from sklearn.metrics import accuracy\_score, recall\_score, precision\_score  
from sklearn.model\_selection import train\_test\_split  
  
  
# 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:  
 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)  
  
# --------------------------------  
  
# Розділення даних на навчальні та тестові набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Створення SVM-класифікатора з гаусовим ядром  
classifier\_rbf = SVC(kernel='rbf', random\_state=0)  
  
# Навчання класифікатора  
classifier\_rbf.fit(X\_train, y\_train)  
  
# Обчислення показників якості  
y\_test\_pred\_rbf = classifier\_rbf.predict(X\_test)  
accuracy\_rbf = accuracy\_score(y\_test, y\_test\_pred\_rbf)  
recall\_rbf = recall\_score(y\_test, y\_test\_pred\_rbf, average='weighted')  
precision\_rbf = precision\_score(y\_test, y\_test\_pred\_rbf, average='weighted')  
  
# Виведення показників якості  
print("Гаусове ядро:")  
print(f"Акуратність: {accuracy\_rbf:.2f}")  
print(f"Повнота: {recall\_rbf:.2f}")  
print(f"Точність: {precision\_rbf:.2f}")

**Результат виконання:**

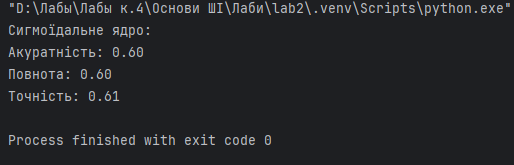


**Сигмойдальне ядро:**

Код:

import numpy as np  
from sklearn.svm import SVC  
from sklearn import preprocessing  
from sklearn.metrics import accuracy\_score, recall\_score, precision\_score  
from sklearn.model\_selection import train\_test\_split  
  
  
# 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:  
 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)  
  
# --------------------------------  
  
# Розділення даних на навчальні та тестові набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Створення SVM-класифікатора з сигмоїдальним ядром  
classifier\_sigmoid = SVC(kernel='sigmoid', random\_state=0)  
  
# Навчання класифікатора  
classifier\_sigmoid.fit(X\_train, y\_train)  
  
# Обчислення показників якості  
y\_test\_pred\_sigmoid = classifier\_sigmoid.predict(X\_test)  
accuracy\_sigmoid = accuracy\_score(y\_test, y\_test\_pred\_sigmoid)  
recall\_sigmoid = recall\_score(y\_test, y\_test\_pred\_sigmoid, average='weighted')  
precision\_sigmoid = precision\_score(y\_test, y\_test\_pred\_sigmoid, average='weighted')  
  
# Виведення показників якості  
print("Сигмоїдальне ядро:")  
print(f"Акуратність: {accuracy\_sigmoid:.2f}")  
print(f"Повнота: {recall\_sigmoid:.2f}")  
print(f"Точність: {precision\_sigmoid:.2f}")

**Результат виконання:**



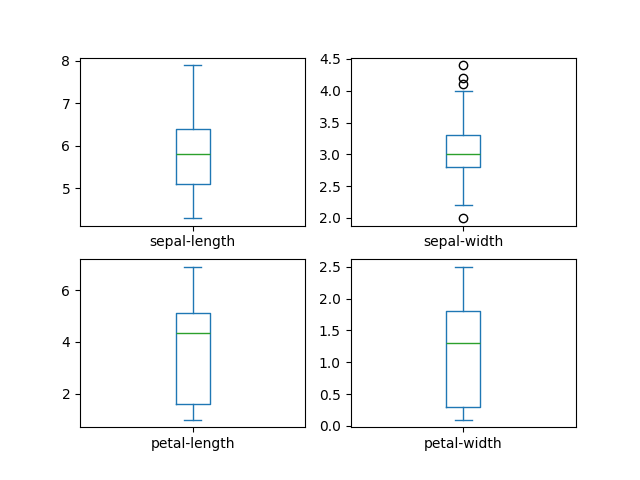
**Завдання №3:**

**Крок 2:**

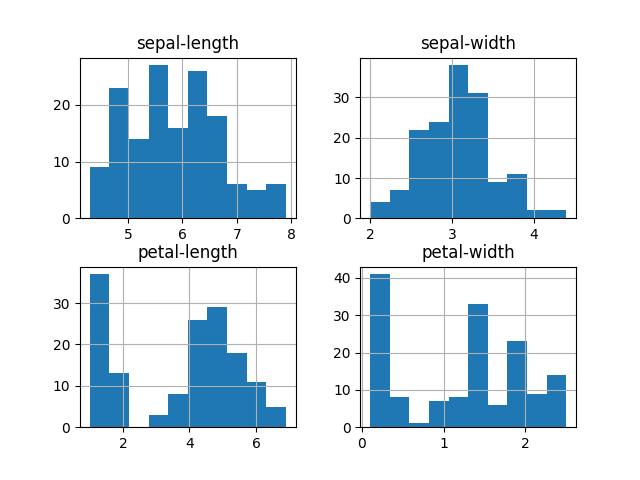
Код:

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  
  
  
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)  
  
# shape  
print(dataset.shape)  
  
# Зріз даних head  
print(dataset.head(20))  
  
# Стастичні зведення методом describe  
print(dataset.describe())  
  
# Розподіл за атрибутом class  
print(dataset.groupby('class').size())  
  
# Діаграма розмаху  
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)  
pyplot.show()  
  
# Гістограма розподілу атрибутів датасета  
dataset.hist()  
pyplot.show()  
  
#Матриця діаграм розсіювання  
scatter\_matrix(dataset)  
pyplot.show()

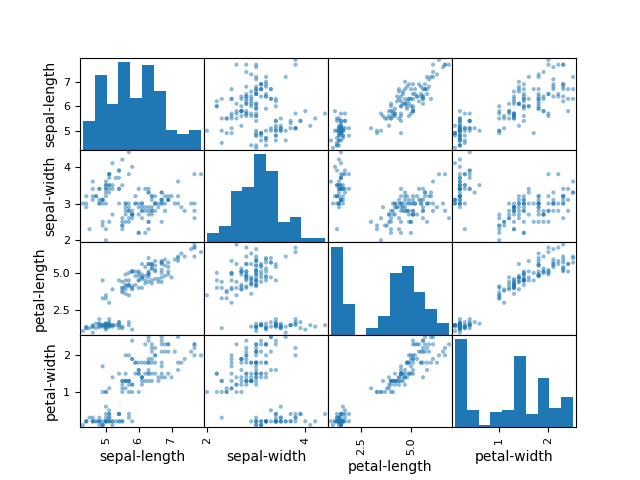
**Результати виконання:**

****

Діаграма розмаху



Гістограма розподілу атрибутів датасета



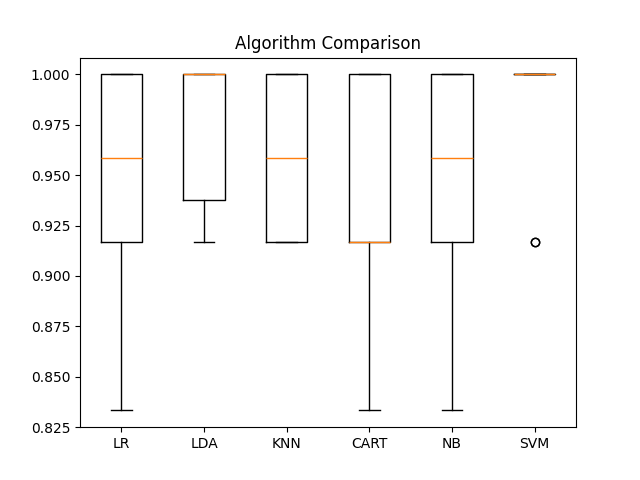
Матриця діаграм розсіювання

**Крок 3,4:**

Код:

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  
  
  
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)  
  
# shape  
print(dataset.shape)  
  
# Зріз даних head  
print(dataset.head(20))  
  
# Стастичні зведення методом describe  
print(dataset.describe())  
  
# Розподіл за атрибутом class  
print(dataset.groupby('class').size())  
  
# Розділення датасету на навчальну та контрольну вибірки  
array = dataset.values  
  
# Вибір перших 4-х стовпців  
X = array[:,0:4]  
  
# Вибір 5-го стовпця  
y = array[:,4]  
  
# Разделение X и y на обучающую и контрольную выборки  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
  
  
# Завантажуємо алгоритми моделі  
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')))  
  
# оцінюємо модель на кожній ітерації  
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()))  
  
# Порівняння алгоритмів  
pyplot.boxplot(results, tick\_labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()

**Результат виконання:**

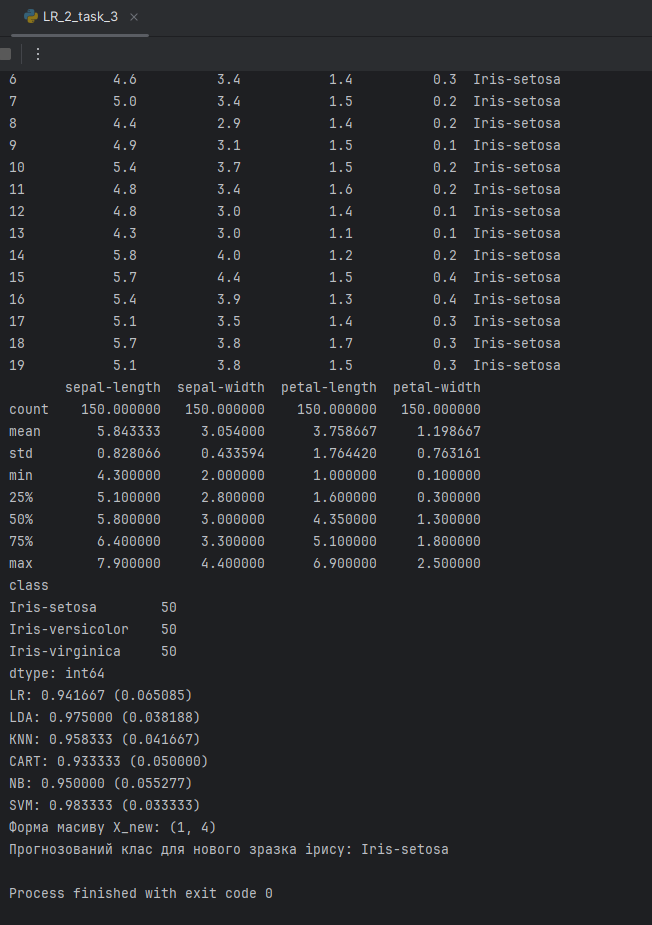
****

**Крок 5,6,7,8:**

Код:

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)  
  
# shape  
print(dataset.shape)  
  
# Зріз даних head  
print(dataset.head(20))  
  
# Стастичні зведення методом describe  
print(dataset.describe())  
  
# Розподіл за атрибутом class  
print(dataset.groupby('class').size())  
  
# Розділення датасету на навчальну та контрольну вибірки  
array = dataset.values  
  
# Вибір перших 4-х стовпців  
X = array[:,0:4]  
  
# Вибір 5-го стовпця  
y = array[:,4]  
  
# Разделение X и y на обучающую и контрольную выборки  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, y, test\_size=0.20, random\_state=1)  
  
  
# Завантажуємо алгоритми моделі  
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')))  
  
# оцінюємо модель на кожній ітерації  
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()))  
  
# Створюємо прогноз на контрольній вибірці  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
# Нові дані для класифікації  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма масиву X\_new: {}".format(X\_new.shape))  
  
# Використовуємо навчену модель для прогнозування класу нового зразка  
prediction = model.predict(X\_new)  
print("Прогнозований клас для нового зразка ірису: {}".format(prediction[0]))

**Результат виконання:**



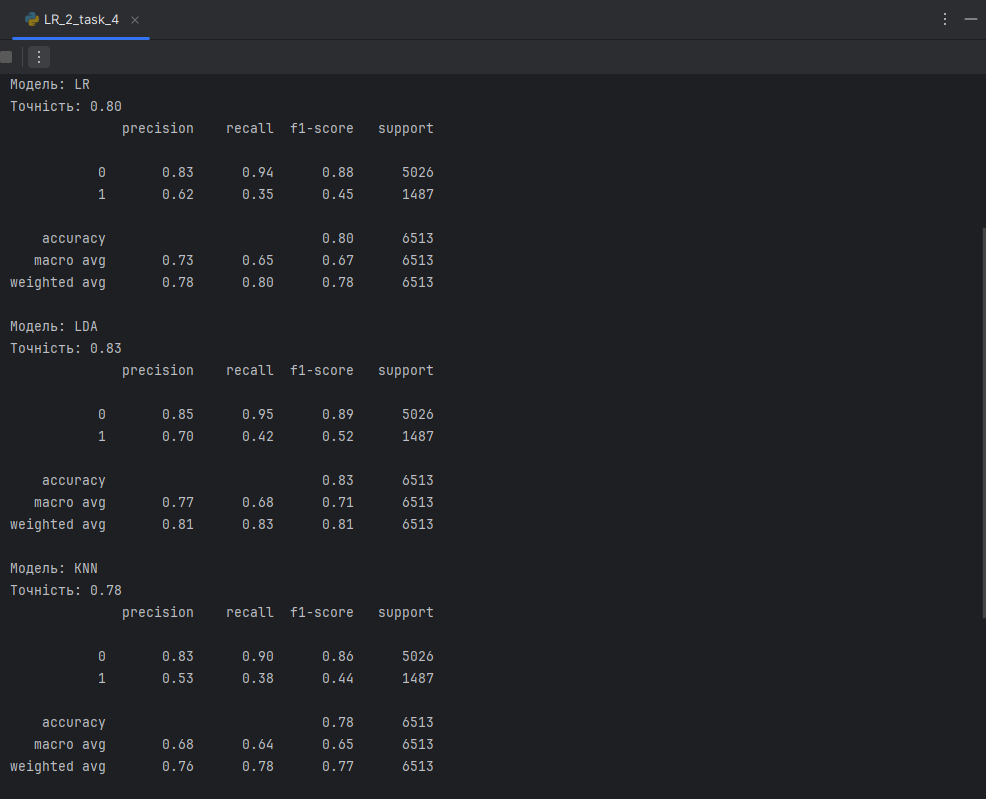
**Завдання №4:** Порівняння якості класифікаторів для набору даних

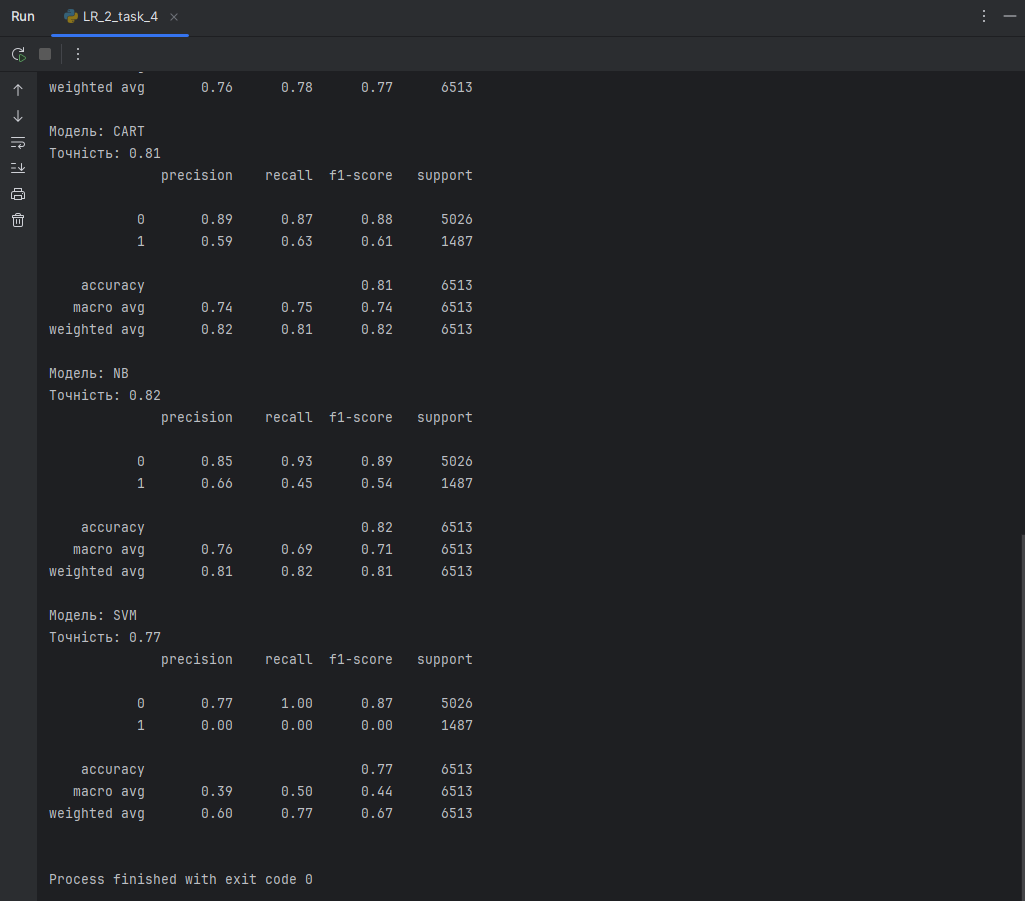
завдання 2.1

Код:

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import accuracy\_score, classification\_report  
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  
  
# Завантаження даних  
url = "income\_data.txt"  
names = ['Age', 'Workclass', 'fnlwgt', 'Education', 'Education-Num', 'Marital-Status', 'Occupation', 'Relationship', 'Race', 'Sex', 'Capital-Gain', 'Capital-Loss', 'Hours-per-week', 'Country', 'Income']  
data = pd.read\_csv(url, names=names)  
  
# Перетворення категорійних даних у числові  
labelEncoder = LabelEncoder()  
for column in data.columns:  
 data[column] = labelEncoder.fit\_transform(data[column])  
  
# Розділення даних на атрибути та мітки  
X = data.drop('Income', axis=1)  
y = data['Income']  
  
# Розділення на навчальну та тестову вибірки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)  
  
# Список моделей для оцінювання  
models = []  
models.append(('LR', LogisticRegression()))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', SVC()))  
  
# Оцінювання моделей  
results = {}  
for name, model in models:  
 model.fit(X\_train, y\_train)  
 predictions = model.predict(X\_test)  
 accuracy = accuracy\_score(y\_test, predictions)  
 report = classification\_report(y\_test, predictions)  
 results[name] = (accuracy, report)  
  
# Виведення результатів  
for name, (accuracy, report) in results.items():  
 print(f"Модель: {name}")  
 print(f"Точність: {accuracy:.2f}")  
 print(report)

**Результат виконання:**



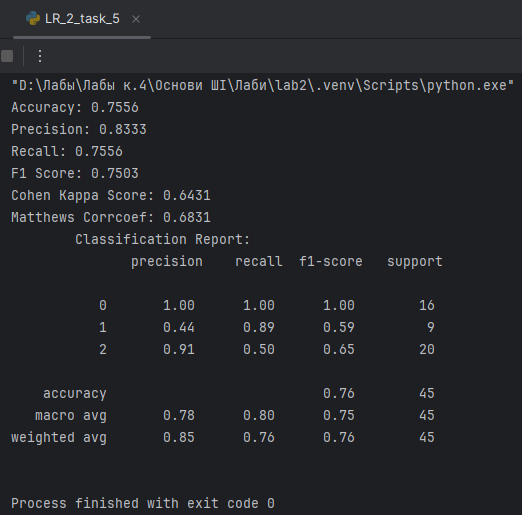
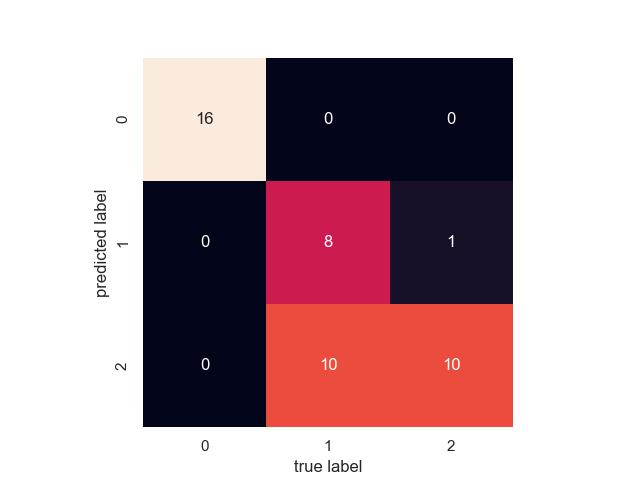


**Завдання №5:** Класифікація даних лінійним класифікатором Ridge

Код:

# ===================================================  
# Приклад класифікатора Ridge  
# ======================================================================  
import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.linear\_model import RidgeClassifier  
from sklearn.model\_selection import train\_test\_split  
  
iris = load\_iris()  
X, y = iris.data, iris.target  
Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size = 0.3, random\_state = 0)  
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))  
  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO #neded for plot  
import seaborn as sns; sns.set()  
import matplotlib.pyplot as plt  
  
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")

**Результат виконання:**

Налаштування класифікатора Ridge:

**tol=1e-2:** Толерантність до зупинки алгоритму, якщо зміна ваги менша за 0.01.

**solver="sag":** Використання стохастичного середнього градієнтного спуску як алгоритму оптимізації.

Візуалізація точності класифікації, де рядки представляють істинні класи, а стовпці - передбачені класи. Значення всередині матриці показують кількість інстанцій для кожної комбінації передбаченого та істинного класу.

**Коефіцієнт Коена Каппа:** Це статистична міра, яка оцінює точність класифікації порівняно з точністю, яка була б отримана випадково. Він враховує можливість випадкової узгодженості, тому є більш надійним показником, ніж проста точність.

**Коефіцієнт кореляції Метьюза:** Це міра якості бінарних (двокласових) класифікацій. Він приймає значення від -1 до +1, де +1 представляє ідеальний прогноз, 0 - випадковий прогноз, а -1 - інверсний прогноз. Цей коефіцієнт є балансованою мірою, яка може бути використана навіть якщо класи не збалансовані.

**Висновок:** використав спеціалізовані бібліотеки та мову

програмування Python, дослідив різні методи класифікації даних та

навчився їх порівнювати.