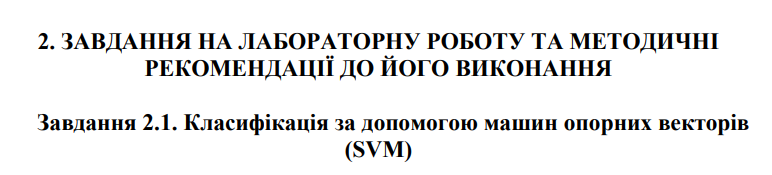
ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

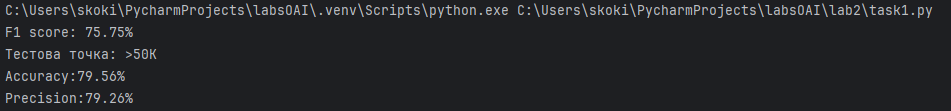
**Посилання на GitHub**: https://github.com/enot1k666/labsOAI.git



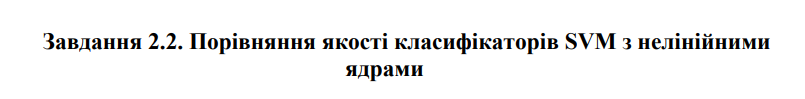
1. age (Вік) - Числова ознака, позначає вік особи.
2. workclass (Клас роботи) - Категоріальна ознака, позначає клас роботи особи.
3. fnlwgt - Числова ознака, характеризує вагу вибірки.
4. education (Освіта) - Категоріальна ознака, позначає рівень освіти особи.
5. education-num (Числовий рівень освіти) - Числова ознака, позначає те сам що і "education", але числом.
6. marital-status (Сімейний стан) - Категоріальна ознака, позначає сімейний стан особи.
7. occupation (Заняття) - Категоріальна ознака, вказує на заняття особи.
8. relationship (Відносини) - Категоріальна ознака, описує відносини особи з іншими.
9. race (Раса) - Категоріальна ознака, позначає расу особи.
10. sex (Стать) - Бінарна ознака, вказує на стать особи (жіноча або чоловіча).
11. capital-gain (Приріст капіталу) - Числова ознака, позначає приріст капіталу особи.
12. capital-loss (Втрати капіталу) - Числова ознака, позначає втрати капіталу особи.
13. hours-per-week (Години на тиждень) - Числова ознака, показує кількість годин, які особа працює на тиждень.
14. native-country (Країна походження) - Категоріальна ознака, вказує на країну походження особи.

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  
from sklearn.metrics import accuracy\_score, precision\_score  
from sklearn.metrics import f1\_score  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
#Відкриємо файл і прочитаємо рядки.  
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  
 elif 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, dual=False, max\_iter=10000))  
  
# Розділення на тренувальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Навчання класифікатора  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
# Обчислення F-міри для SVМ-класифікатора  
f1 = f1\_score(y\_test, y\_test\_pred, average="weighted")  
print("F1 score: " + str(round(100 \* f1, 2)) + "%")  
  
  
input\_data = ["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
  
# Використання класифікатора для кодованої точки даних  
predicted\_class = classifier.predict(input\_data\_encoded)  
predicted\_label = label\_encoder[-1].inverse\_transform(predicted\_class)[0]  
print("Тестова точка:",predicted\_label)  
  
# Обчислення акуратності  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
print("Accuracy:" + str(round(100 \* accuracy, 2)) + "%")  
  
# Обчислення точності  
precision = precision\_score(y\_test, y\_test\_pred, average="weighted")  
print("Precision:" + str(round(100 \* precision, 2)) + "%")

**Результат виконання:**



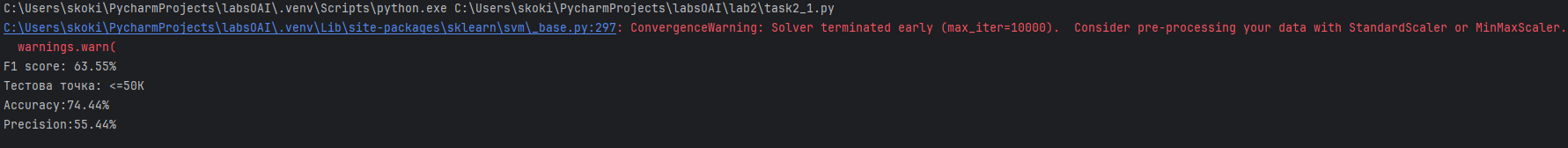
**Відповідь:** Тестова точка належить класу: >50K



1. Поліноміальне ядро.

Лістинг програми:

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score,f1\_score  
  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
#Відкриємо файл і прочитаємо рядки.  
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  
 elif 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(SVC(kernel='poly',degree=8,random\_state=0,C= 0.001,max\_iter = 10000))  
  
# Розділення на тренувальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Навчання класифікатора  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
# Обчислення F-міри для SVМ-класифікатора  
f1 = f1\_score(y\_test, y\_test\_pred, average="weighted")  
print("F1 score: " + str(round(100 \* f1, 2)) + "%")  
  
  
input\_data =["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
  
# Використання класифікатора для кодованої точки даних  
predicted\_class = classifier.predict(input\_data\_encoded)  
predicted\_label = label\_encoder[-1].inverse\_transform(predicted\_class)[0]  
print("Тестова точка:",predicted\_label)  
  
# Обчислення акуратності  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
print("Accuracy:" + str(round(100 \* accuracy, 2)) + "%")  
  
# Обчислення точності  
precision = precision\_score(y\_test, y\_test\_pred, average="weighted")  
print("Precision:" + str(round(100 \* precision, 2)) + "%")

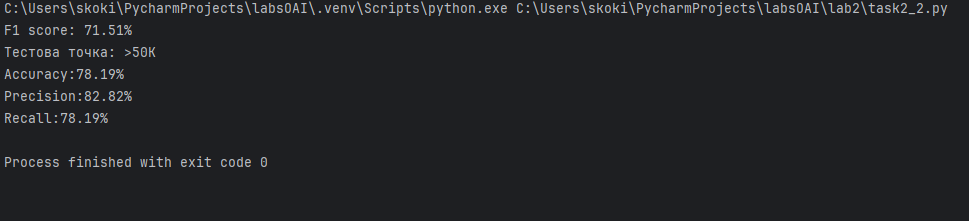
**Результат виконання:**  


2. Гаусове ядро

Лістинг програми:

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score,f1\_score, recall\_score  
  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
#Відкриємо файл і прочитаємо рядки.  
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  
 elif 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(SVC(kernel='rbf',max\_iter = 10000))  
  
# Розділення на тренувальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Навчання класифікатора  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
# Обчислення F-міри для SVМ-класифікатора  
f1 = f1\_score(y\_test, y\_test\_pred, average="weighted")  
print("F1 score: " + str(round(100 \* f1, 2)) + "%")  
  
  
input\_data =["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
  
# Використання класифікатора для кодованої точки даних  
predicted\_class = classifier.predict(input\_data\_encoded)  
predicted\_label = label\_encoder[-1].inverse\_transform(predicted\_class)[0]  
print("Тестова точка:",predicted\_label)  
  
# Обчислення акуратності  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
print("Accuracy:" + str(round(100 \* accuracy, 2)) + "%")  
  
# Обчислення точності  
precision = precision\_score(y\_test, y\_test\_pred, average="weighted")  
print("Precision:" + str(round(100 \* precision, 2)) + "%")  
  
# Обчислення повноти  
recall = recall\_score(y\_test, y\_test\_pred, average="weighted")  
print("Recall:" + str(round(100 \* recall, 2)) + "%")

**Результат виконання:**

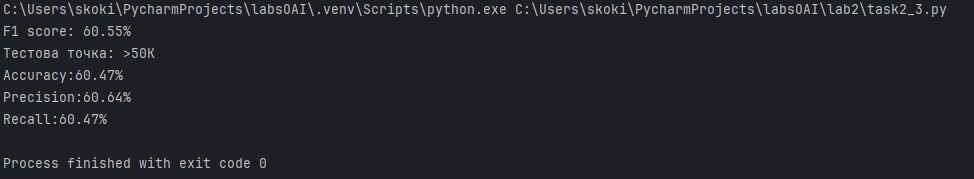


3. Сигмоїдальне ядро.

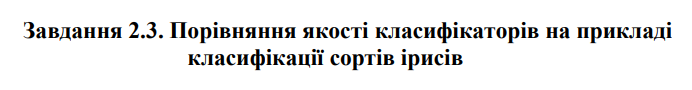
Лістинг програми:

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score,f1\_score, recall\_score  
  
# Вхідний файл, який містить дані  
input\_file = 'income\_data.txt'  
  
# Читання даних  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
#Відкриємо файл і прочитаємо рядки.  
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  
 elif 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(SVC(kernel='sigmoid',max\_iter = 10000))  
  
# Розділення на тренувальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Навчання класифікатора  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
# Обчислення F-міри для SVМ-класифікатора  
f1 = f1\_score(y\_test, y\_test\_pred, average="weighted")  
print("F1 score: " + str(round(100 \* f1, 2)) + "%")  
  
  
input\_data =["52", "Self-emp-inc", "287927", "HS-grad", "9", "Married-civ-spouse", "Exec-managerial", "Wife", "White", "Female", "15024", "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([input\_data[i]])[0])  
 count += 1  
input\_data\_encoded = np.array(input\_data\_encoded).reshape(1, -1)  
  
# Використання класифікатора для кодованої точки даних  
predicted\_class = classifier.predict(input\_data\_encoded)  
predicted\_label = label\_encoder[-1].inverse\_transform(predicted\_class)[0]  
print("Тестова точка:",predicted\_label)  
  
# Обчислення акуратності  
accuracy = accuracy\_score(y\_test, y\_test\_pred)  
print("Accuracy:" + str(round(100 \* accuracy, 2)) + "%")  
  
# Обчислення точності  
precision = precision\_score(y\_test, y\_test\_pred, average="weighted")  
print("Precision:" + str(round(100 \* precision, 2)) + "%")  
  
# Обчислення повноти  
recall = recall\_score(y\_test, y\_test\_pred, average="weighted")  
print("Recall:" + str(round(100 \* recall, 2)) + "%")

Результат виконання:



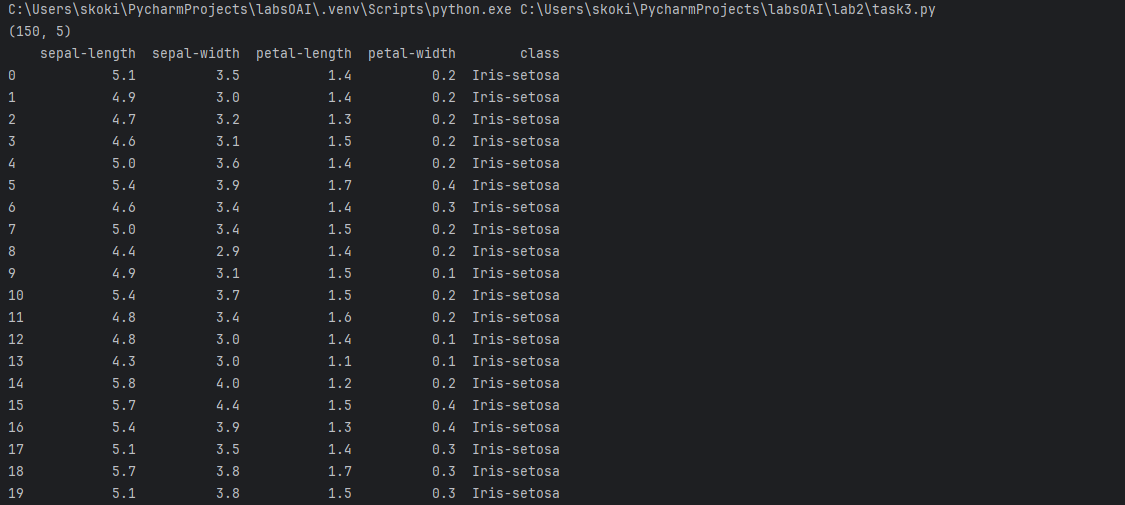
**Висновок:** Спосіб класифікації з нелінійним ядром SVM, зокрема з поліноміальним ядром, визнається як найефективніший з точки зору повноти. Однак, щодо точності і акуратності, найкращим виявляється нелінійний класифікатор SVM із гаусовим ядром. Загалом можна стверджувати, що класифікатор із гаусовим ядром видає найкращі результати у виконанні поставлених завдань.

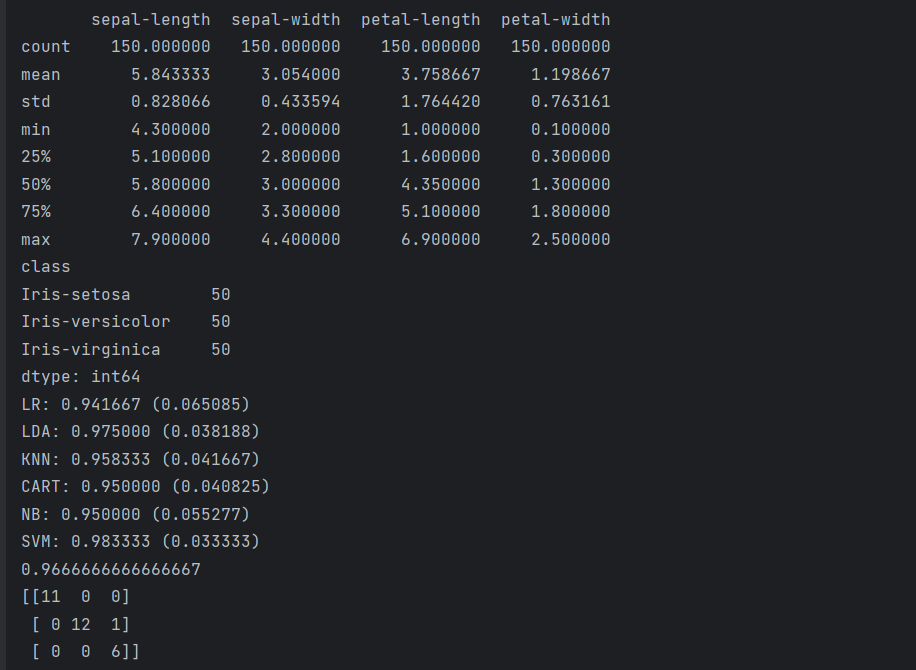


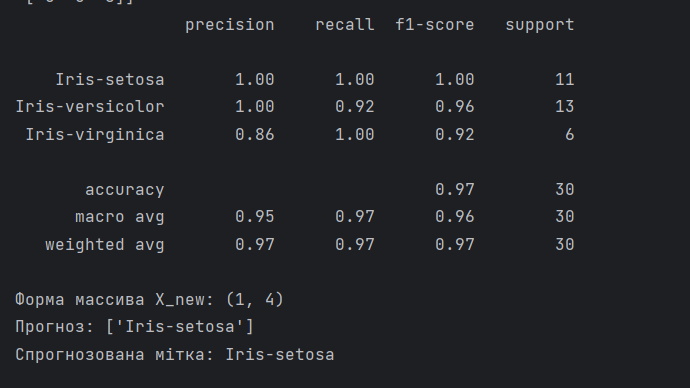
Лістинг програми:

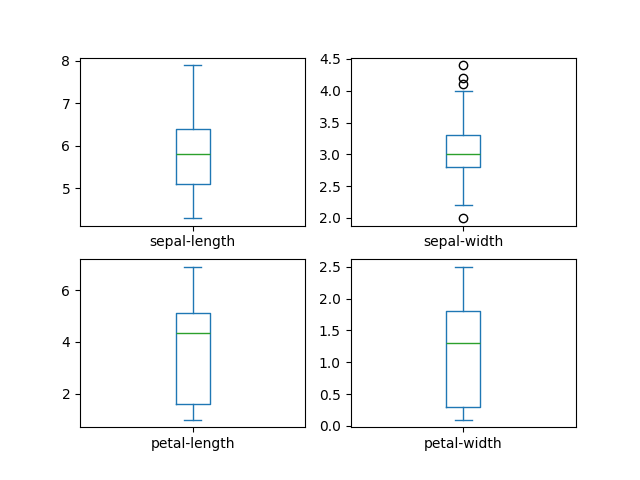
import numpy as np  
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import StratifiedKFold  
from 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()  
  
# 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]  
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, labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()  
  
# Створюємо прогноз на контрольній вибірці  
model = SVC(gamma="auto")  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
# Оцінюємо прогноз  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))  
  
#Отримуємо прогноз  
X\_new = np.array([[5.0, 2.9, 1.0, 0.2]])  
  
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]))

**Результат виконання:**

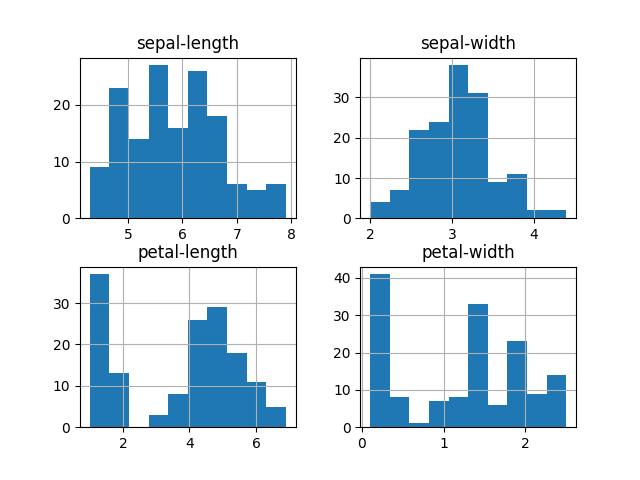




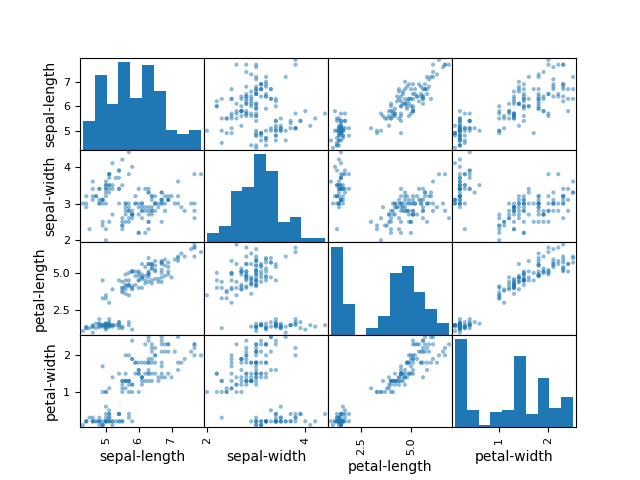




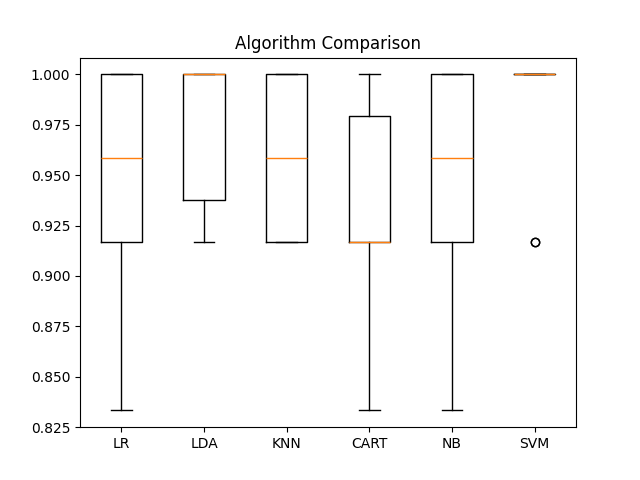
Діаграма розмаху



Гістограма розподілу атрибутів датасета

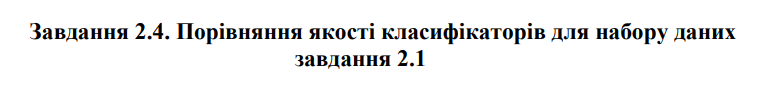


Матриця діаграм розсіювання



Алгоритм порівняння

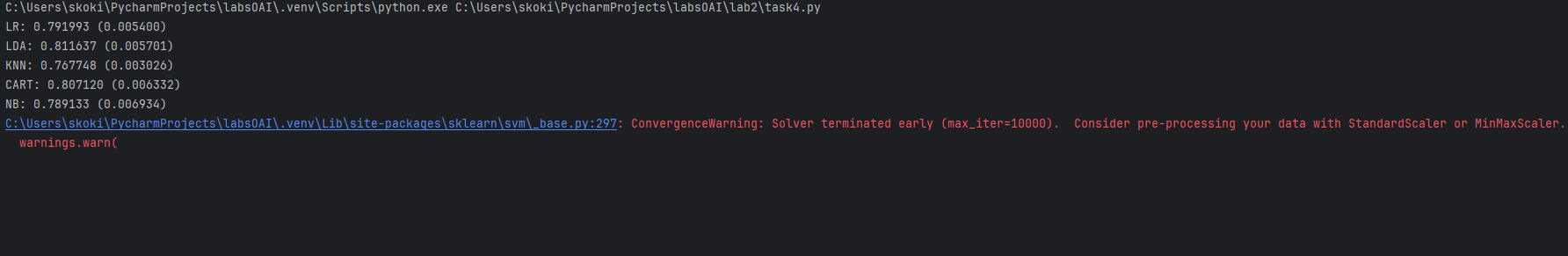
**Висновок:** Було вибрано метод опорних векторів (SVM). Вдалося досягти показника якості 0.97. Квітка належить до виду Iris-setosa.

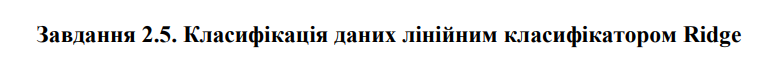


Лістинг програми:

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  
#Відкриємо файл і прочитаємо рядки.  
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  
 elif 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)  
  
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()))

Результат виконання:

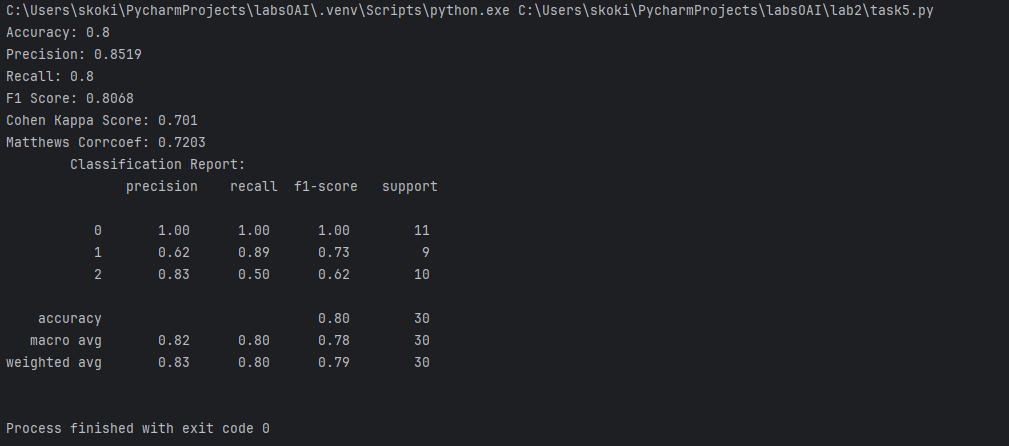


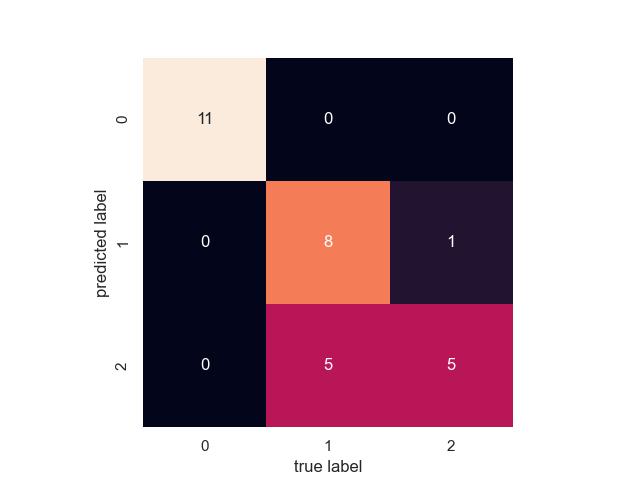


Лістинг програми:

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")

**Результат виконання:**





Файл Confusion

1. **Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.**  
   В класифікаторі Ridge були використані налаштування точності (tol=1e-2) та розв’язник(solver="sag").
2. **Опишіть які показники якості використовуються та їх отримані результати**Показники якості– акуратність, точність, повнота, коефіцієнт Коена Каппа, коефіцієнт кореляції Метьюза.
3. **Вставте у звіт та поясніть зображення Confusion.jpg**На зображені показана матриця confusion, як skicit-learn може навчатися класифікувати.
4. **Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.**

**Коефіцієнт Коена Каппа –** це статистичний показник, який використовується для вимірювання згоди або узгодженості між двома оцінювачами або системами оцінювання. Зазвичай використовується у контексті оцінки точності класифікаційних моделей, особливо в задачах класифікації, де важлива не тільки точність, але і узгодженість між прогнозами. В даному випадку він показує істотну згоду.

**Коефіцієнт кореляції Метьюза -** це інший статистичний показник, який використовується для оцінки якості класифікаційних моделей, особливо в задачах бінарної класифікації (тобто, коли є два класи - позитивний і негативний). Незважаючи на високу точність, акуратність і повноту в нашому випадку, коефіцієнт кореляції Метьюза становить 0.6831. Це вказує на високу оцінку цього коефіцієнта в ситуаціях, коли класифікатор успішно розпізнає як негативні, так і позитивні значення.

**Висновки:** на даній лабораторній ми, використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та

навчились їх порівнювати.