**ЛАБОРАТОРНА РОБОТА № 2**

***Тема:*** ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

**Хід роботи:**

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

**14 ознак з набору данних(із прикладом):**

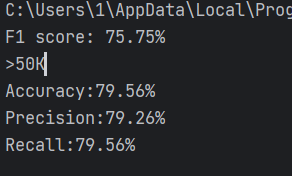
1. 39 - числова ознака: вік.
2. State-gov - категоріальна ознака: тип зайнятості (у цьому випадку, державна служба).
3. 77516 - числова ознака: унікальний ідентифікатор або номер.
4. Bachelors - категоріальна ознака: рівень освіти (у цьому випадку, бакалаврська).
5. 13 - числова ознака: кількість років навчання (як частина рівня освіти).
6. Never-married - категоріальна ознака: сімейний стан (у цьому випадку, ніколи не одружений/незаміжня).
7. Adm-clerical - категоріальна ознака: професія або робоча посада (у цьому випадку, адміністративний клерк).
8. Not-in-family - категоріальна ознака: статус в сім'ї (у цьому випадку, не в родині).
9. White - категоріальна ознака: раса (у цьому випадку, білий).
10. Male - категоріальна ознака: стать (чоловік).
11. 0 - числова ознака: кількість власних дітей.
12. 40 - числова ознака: кількість годин роботи на тиждень.
13. United-States - категоріальна ознака: країна проживання (у цьому випадку, Сполучені Штати Америки).
14. <=50K - категоріальна ознака: цільова змінна, яка може позначати дохід (у цьому випадку, менше або рівно 50 тисяч доларів на рік).
15. Початок форми

**Лістинг програми:**

import numpy as np  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_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  
  
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  
from sklearn.metrics import f1\_score  
f1 = f1\_score(y\_test, y\_test\_pred, average="weighted")  
print("F1 score: " + str(round(100 \* f1, 2)) + "%")  
  
# Передбачення результату для тестової точки даних  
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 та виведення даних  
predicted\_class = classifier.predict(input\_data\_encoded)  
predicted\_label = label\_encoder[-1].inverse\_transform(predicted\_class)[0]  
print(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 програми:**

****

**Рис.1** Результат виконання програми

**Тестова точка належить до класу >50K**

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

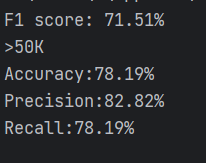
**З поліноміальним ядром:**

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, recall\_score  
from sklearn.svm import SVC  
from sklearn.model\_selection import GridSearchCV  
  
param\_grid = {  
 'C': [0.1, 1, 10],  
 'degree': [2, 3, 4],  
 'coef0': [0, 1, 2]  
}  
  
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  
 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 SVM-класифікатора з поліноміальним ядром  
classifier = OneVsOneClassifier(SVC(kernel = 'poly', C = 1.0, degree = 8, coef0 = 1))  
  
# 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  
from sklearn.metrics import f1\_score  
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(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)) + "%")

**З гаусовим ядром:**

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, 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  
  
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', random\_state = 0))  
  
# 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  
from sklearn.metrics import f1\_score  
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(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 програми:**

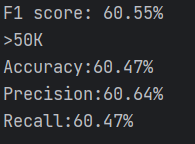
****

**Рис.3** Результат з Гаусовим ядром

**З сигмоїдальним ядром:**

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, 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  
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', random\_state = 0))  
  
# 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  
from sklearn.metrics import f1\_score  
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(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 програми:**

****

**Рис.4** Результат з сигмоїдальним ядром

1. **Поліноміальне ядро**: Використовується поліноміальне ядро, коли вважаємо, що взаємозв'язки між ознаками можуть бути аппроксимовані поліномами
2. **Гаусове ядро (RBF)**: Гаусове ядро зазвичай є добрим вибором за замовчуванням, оскільки воно добре працює з різними типами даних і здатне моделювати складні нелинійні залежності.
3. **Сигмоїдальне ядро**: Сигмоїдальне ядро корисно, коли дані мають нестандартні форми та взаємозв'язки

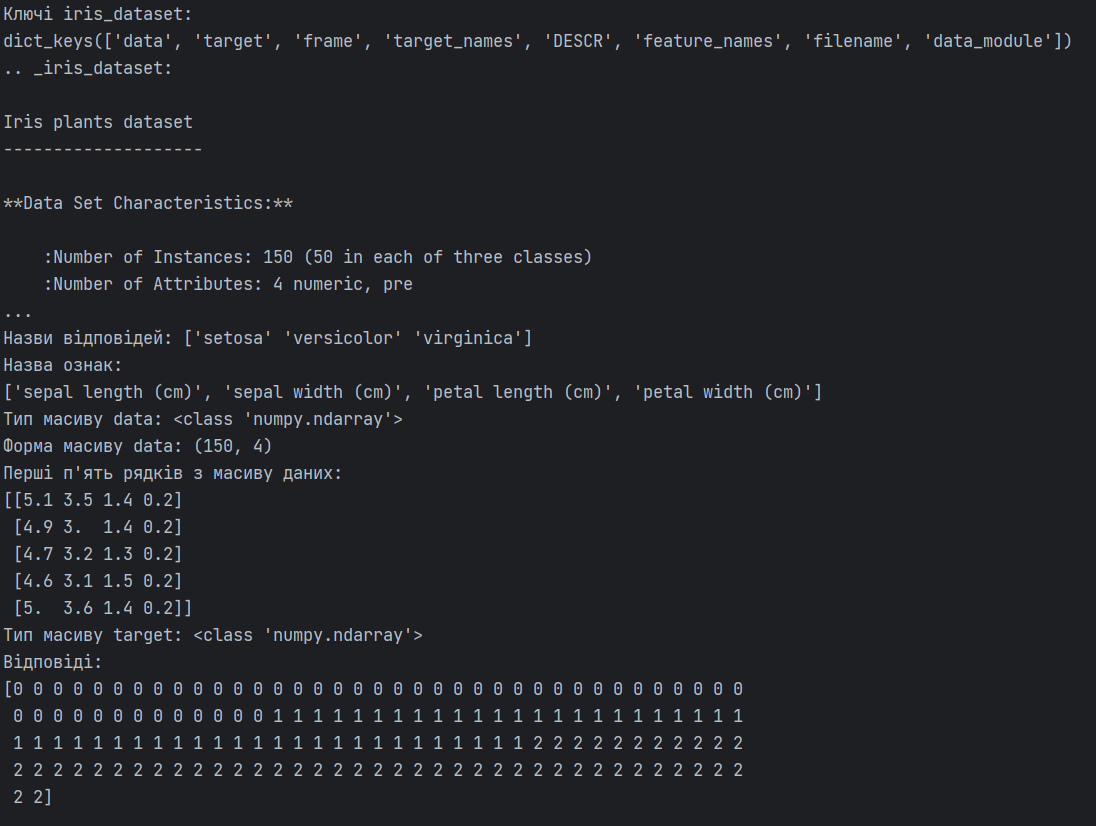
**Як ми бачимо, що кожен із ядер має власні переваги й мінуси, саме в нашому випадку найкраще себе показав метод із Гаусовим ядром.**

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

**Лістинг програми:**

from sklearn.datasets import load\_iris  
  
iris\_dataset = load\_iris()  
print("Ключі iris\_dataset: \n{}".format(iris\_dataset.keys()))  
print(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("Перші п'ять рядків з масиву даних:")  
print(iris\_dataset["data"][:5])  
  
print("Тип масиву target: {}".format(type(iris\_dataset["target"])))  
print("Відповіді:\n{}".format(iris\_dataset["target"]))

**Результат виконання програми:**

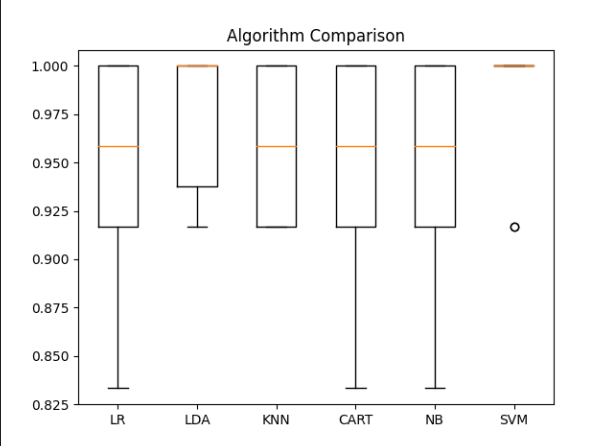
****

**Рис.5** Результат виконання програми

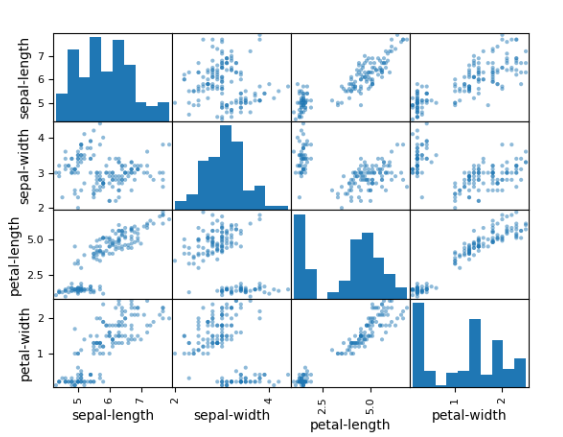
**Лістинг програми:**

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  
from sklearn.datasets import load\_iris  
  
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)  
print(dataset.head(20))  
print(dataset.describe())  
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))  
  
# Нові дані для класифікації (за вашими значеннями)  
X\_new = np.array([[5.0, 2.9, 1.0, 0.2]])  
  
# Зробіть прогноз для нових даних  
prediction = model.predict(X\_new)  
  
# Виведіть результат прогнозу  
print("Прогноз: {}".format(prediction))

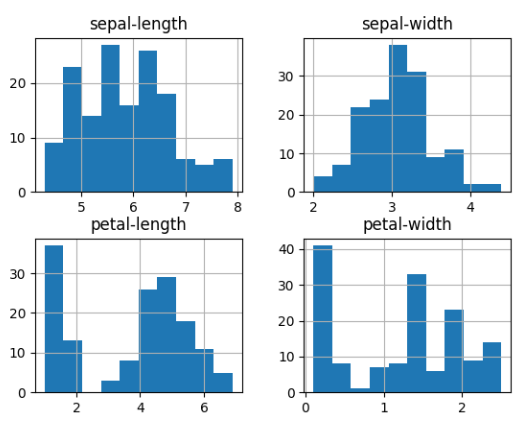
**Результат виконання програми:**

****

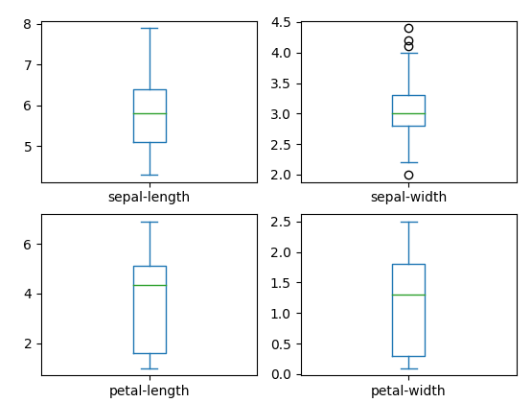
**Рис.5** Візуалізація

****

**Рис.6** Візуалізація

****

**Рис.7** Візуалізація

****

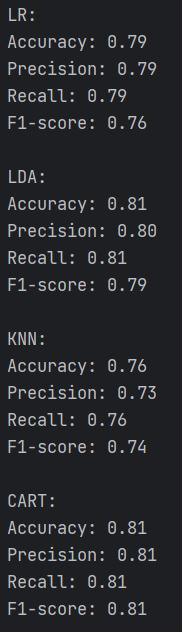
**Рис.8** Візуалізація

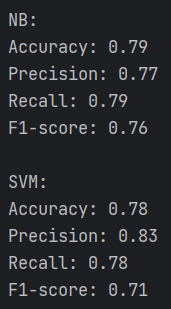
**Завдання 2.4.** Порівняння якості класифікаторів для набору даних завдання 2.1

**Лістинг програми:**

from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
from sklearn.linear\_model import LogisticRegression  
from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.naive\_bayes import GaussianNB  
from sklearn.svm import SVC  
from sklearn import preprocessing  
import numpy as np  
  
  
# 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  
 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)  
  
# Розділення даних на тренувальний та тестовий набори  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Ініціалізація та навчання моделей  
models = [  
 ("LR", LogisticRegression()),  
 ("LDA", LinearDiscriminantAnalysis()),  
 ("KNN", KNeighborsClassifier()),  
 ("CART", DecisionTreeClassifier()),  
 ("NB", GaussianNB()),  
 ("SVM", SVC())  
]  
  
results = []  
  
for name, model in models:  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
   
 accuracy = accuracy\_score(y\_test, y\_pred)  
 precision = precision\_score(y\_test, y\_pred, average='weighted')  
 recall = recall\_score(y\_test, y\_pred, average='weighted')  
 f1 = f1\_score(y\_test, y\_pred, average='weighted')  
   
 results.append((name, accuracy, precision, recall, f1))  
  
# Виведення результатів для порівняння  
for name, accuracy, precision, recall, f1 in results:  
 print(f"{name}:")  
 print(f"Accuracy: {accuracy:.2f}")  
 print(f"Precision: {precision:.2f}")  
 print(f"Recall: {recall:.2f}")  
 print(f"F1-score: {f1:.2f}")  
 print()

**Результат виконання програми:**

****

****

**Рис.9** Результат виконання програми

**Перед тим, як робити висновки, давайте трішки опишемо кожен із методів:**

1. **Логістична регресія (LR)**: Цей метод використовує логістичну функцію для прогнозування ймовірності належності до класу. Він часто використовується для бінарної та багатокласової класифікації і добре працює для лінійно роздільних даних.
2. **Лінійний дискримінантний аналіз (LDA)**: LDA також працює з лінійно роздільними даними і намагається знайти лінійну комбінацію ознак, яка максимізує відстань між класами.
3. **Метод k-найближчих сусідів (KNN)**: KNN використовує найближчих сусідів об'єкта для класифікації. Він добре працює для даних зі складною структурою та нелинійними залежностями.
4. **Класифікація та регресія за допомогою дерев (CART)**: CART використовує рішальні дерева для розділення даних на різні класи. Вони можуть бути лінійними або нелинійними, залежно від структури дерева.
5. **Наївний баєсовський класифікатор (NB)**: Цей метод ґрунтується на теорії ймовірності і передбачає клас на основі наївного припущення про незалежність між ознаками.
6. **Метод опорних векторів (SVM)**: SVM намагається знайти гіперплощину, яка найкращим чином розділяє дані на класи. Він може використовуватися як для лінійно роздільних, так і для нелинійно роздільних даних за допомогою ядер.

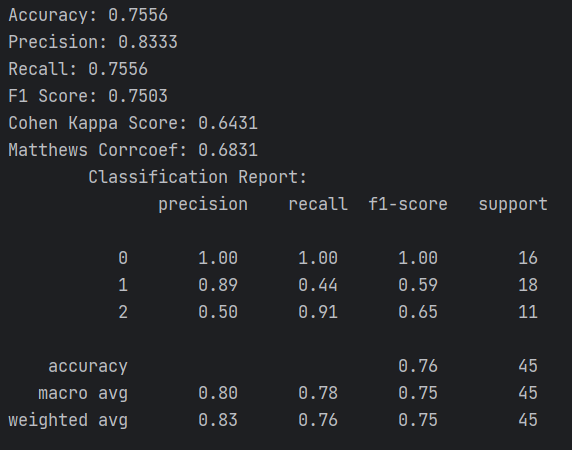
**Як ми бачимо, кожен з них використовується у своїх власних ситуаціях й працює різними методами. Саме в нашому випадку найкраще себе показав метод класифікації та регресії за допомогою дерев.**

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

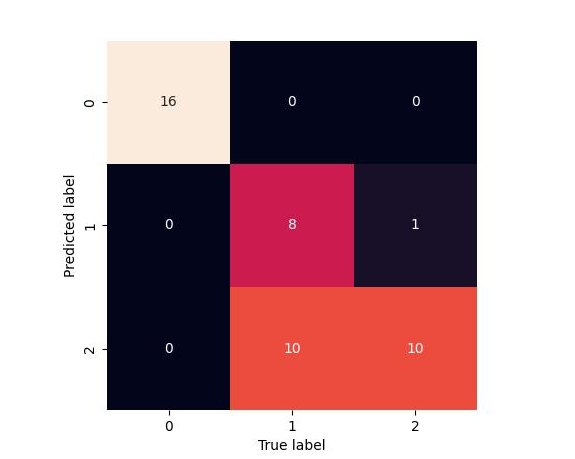
**Лістинг програми:**

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  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
  
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)  
  
accuracy = np.round(metrics.accuracy\_score(ytest, ypred), 4)  
precision = np.round(metrics.precision\_score(ytest, ypred, average='weighted'), 4)  
recall = np.round(metrics.recall\_score(ytest, ypred, average='weighted'), 4)  
f1\_score = np.round(metrics.f1\_score(ytest, ypred, average='weighted'), 4)  
cohen\_kappa = np.round(metrics.cohen\_kappa\_score(ytest, ypred), 4)  
matthews\_corrcoef = np.round(metrics.matthews\_corrcoef(ytest, ypred), 4)  
  
print('Accuracy:', accuracy)  
print('Precision:', precision)  
print('Recall:', recall)  
print('F1 Score:', f1\_score)  
print('Cohen Kappa Score:', cohen\_kappa)  
print('Matthews Corrcoef:', matthews\_corrcoef)  
  
classification\_report = metrics.classification\_report(ytest, ypred)  
print('\t\tClassification Report:\n', classification\_report)  
  
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 програми:**

****

**Рис.10** Результат виконання програми

****

**Рис.11** Візуалізація

Коефіцієнт Коена Каппа (Cohen's Kappa) і коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) є метриками, які використовуються для оцінки якості класифікаційних моделей, особливо в задачах бінарної класифікації. Вони дозволяють враховувати не тільки правильні класифікації, але й помилкові класифікації та дисбаланс класів.

***Висновок:*** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.

***Посилання на GitHub:*** https://github.com/BOYYYKO/ai