**ЛАБОРАТОРНА РОБОТА №2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

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

**Хід роботи:**

**Завдання 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 та що вони позначають

та вид (числові чи категоріальні):

• age (вік) - числовий

• workclass (робочий клас) - категоріальний

• fnlwgt (ваговий коефіцієнт) - числовий

• education (рівень освіти) - категоріальний

• education-num (числовий еквівалент рівня освіти) - числовий

• marital-status (сімейний стан) - категоріальний

• occupation (заняття або професія) - категоріальний

• relationship (відносини в сім’ї) - категоріальний

• race (раса) – категоріальний

• sex (стать) - категоріальний

• capital-gain (приріст капіталу) - числовий

• capital-loss (втрати капіталу) - числовий

• hours-per-week (кількість годин, які фізична особа працює на тиждень) -

числовий

• native-country (країна-походження) – категоріальний

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

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, 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(', ')  
  
 if data[-1] == '<=50K' and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == '>50K' and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
scaller = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
X = scaller.fit\_transform(X)  
  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0, dual=True))  
  
classifier.fit(X=X, y=y)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,  
 random\_state=5)  
scaller = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
X\_train = scaller.fit\_transform(X\_train)  
classifier.fit(X=X\_train, y=y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
accuracy\_values = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)  
print("Accuracy: " + str(round(100 \* accuracy\_values.mean(), 2)) + "%")  
precision\_values = cross\_val\_score(classifier, X, y, scoring='precision\_weighted',  
 cv=3)  
print("Precision: " + str(round(100 \* precision\_values.mean(), 2)) + "%")  
recall\_values = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)  
print("Recall: " + str(round(100 \* recall\_values.mean(), 2)) + "%")  
f1\_values = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("F1: " + str(round(100 \* f1\_values.mean(), 2)) + "%")  
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 = np.array([-1] \* len(input\_data))  
count = 0  
for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = item  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([item])[0])  
 count += 1  
input\_data\_encoded = input\_data\_encoded.astype(int)  
input\_data\_encoded = [input\_data\_encoded]  
  
predicted\_class = classifier.predict(input\_data\_encoded)  
print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])



Рис. 1. Результат виконання

Зробіть висновок до якого класу належить тестова точка

Тестова точка відноситься до категорії з доходом більше 50K, що вказує на те, що особа заробляє суму, перевищуючу 50 тисяч доларів.

**Завдання 2:** Порівняння якості класифікаторів SVM з нелінійними ядрами.

Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM.

1) З поліноміальним ядром

Лістинг програми (змінений фрагмент коду):



Результат виконання

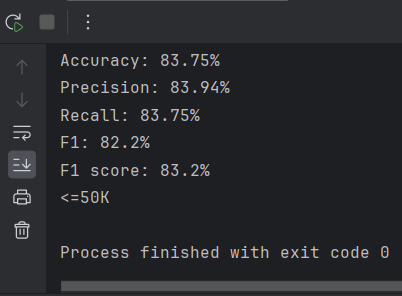


Рис. 2.1. Результат виконання

2) З гаусовим ядром

Лістинг програми (змінений фрагмент коду):



Результат виконання:

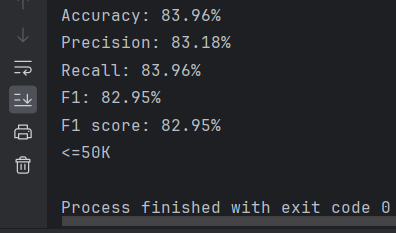


Рис. 2.2. Результат виконання

3) З сигмоїдальним ядром

Лістинг програми (змінений фрагмент коду):



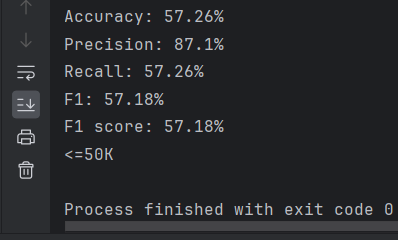
Результат виконання:

Рис. 2.3. Результат виконанняВисновок, який з видів SVM найкраще виконує завдання класифікації за

результатами тренування

SVM з гаусовим ядром (rbf) продемонстрував найкращий результат, володіючи найвищими значеннями точності (accuracy), точності (precision), повноти та F1 Score порівняно з іншими двома моделями. Таким чином, для даної задачі класифікації найбільш оптимальним варіантом є SVM з гаусовим ядром.

**Завдання 3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.**

КРОК 1. ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ

Код для ознайомлення зі структурою даних, 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("Тип масиву target: {}".format(type(iris\_dataset['target'])))  
print("\nЗначення ознак для перших п'яти прикладів:\n{}".format(iris\_dataset['data'][:5]))  
print("\nВідповіді:\n{}".format(iris\_dataset['target']))

Результат виконання:

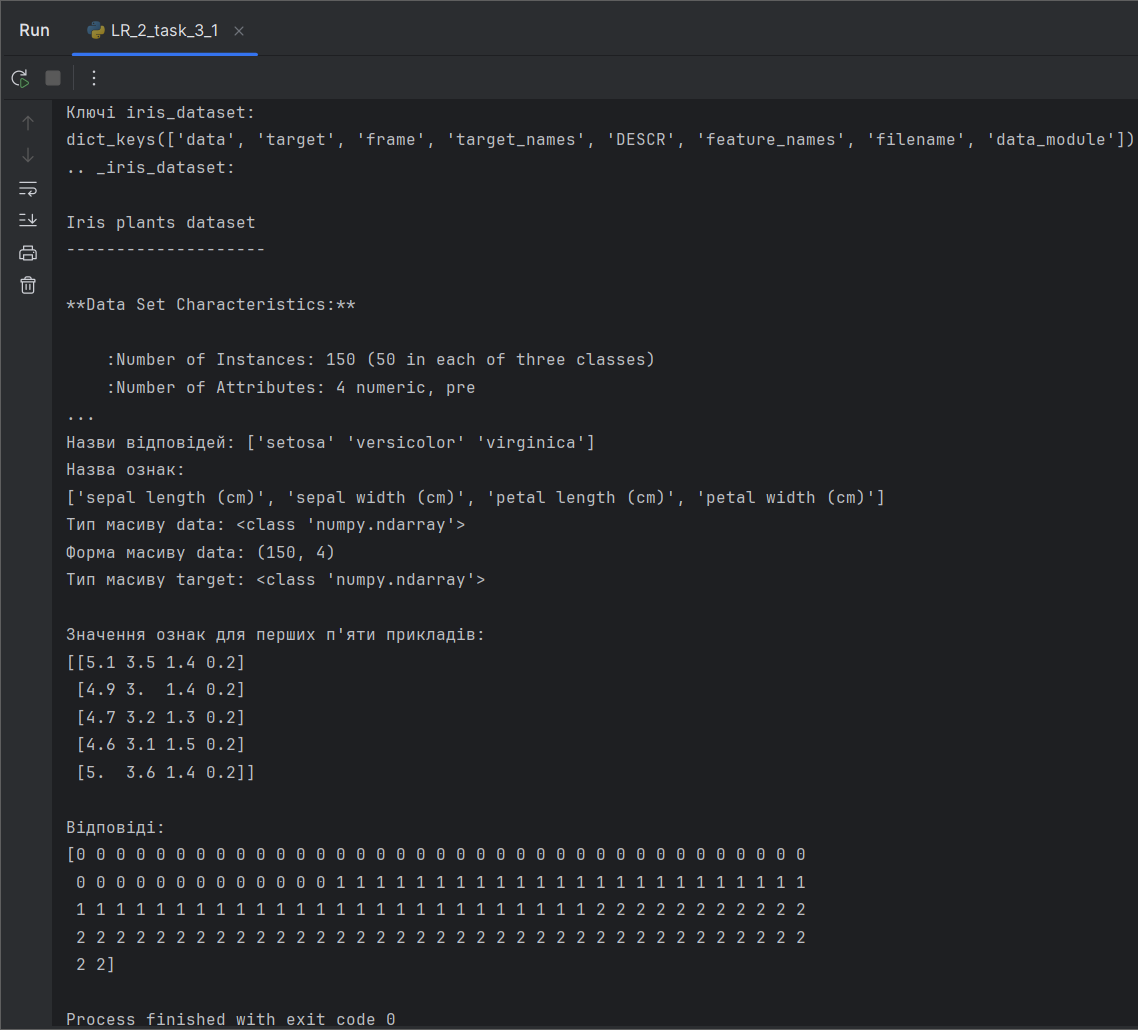


Рис. 3.1 – 3.2. Результат виконання

Завантаження датасету, лістинг програми (змінений):

# Завантаження бібліотек  
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  
# Завантаження датасету  
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())

Результат виконання:

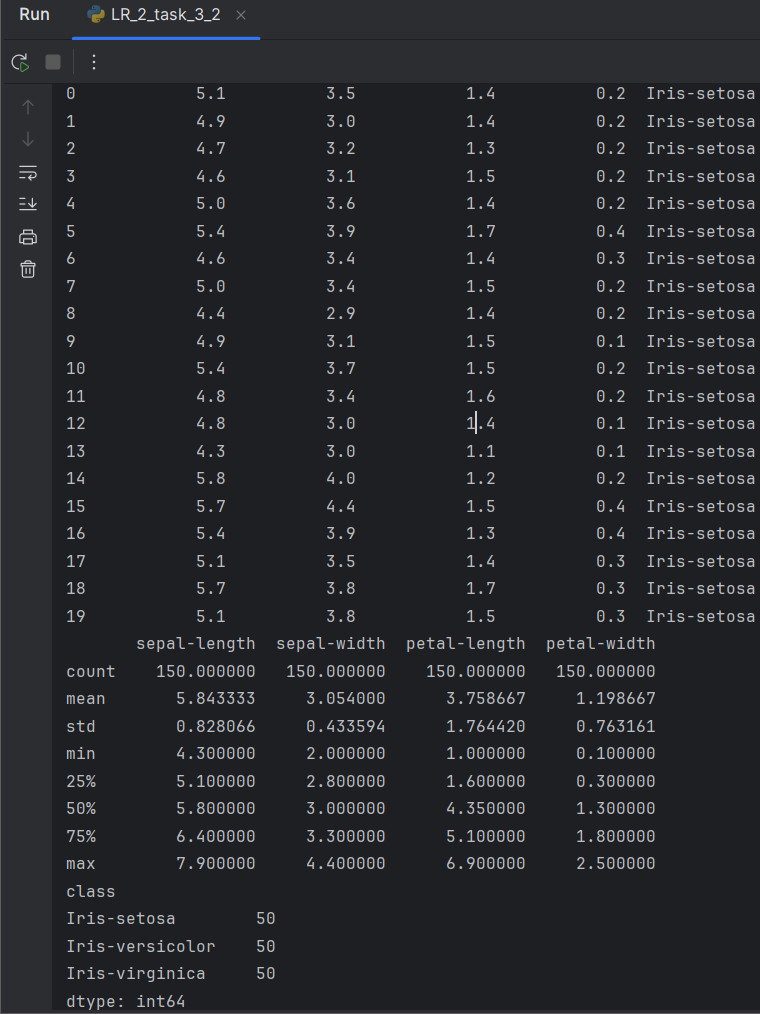


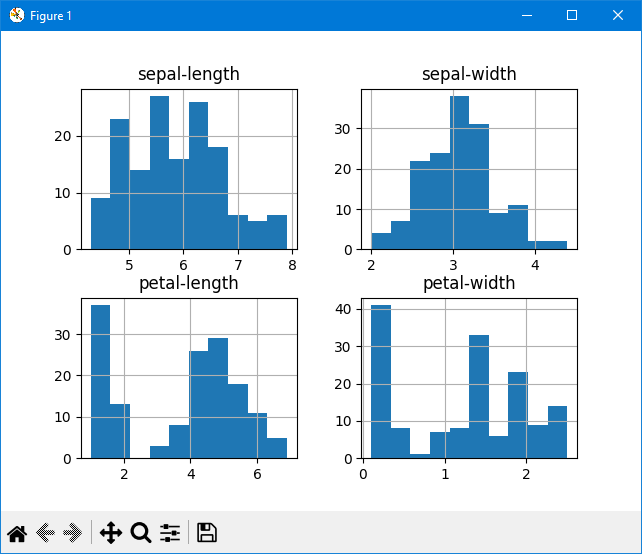
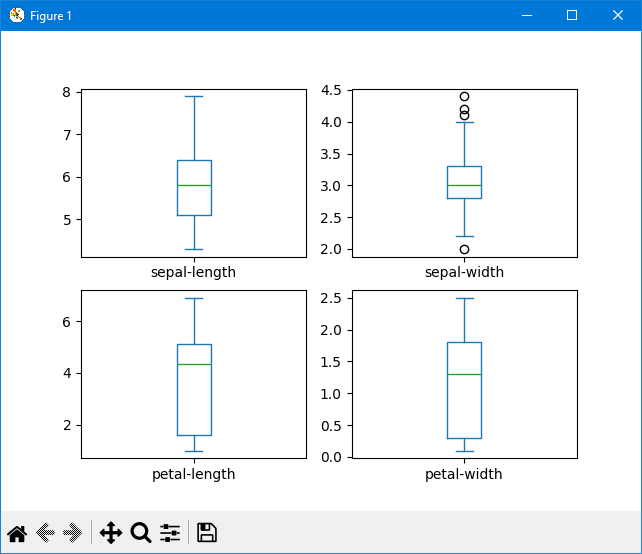
Рис. 3.3 - 3.4. Результат виконання

КРОК 2. ВІЗУАЛІЗАЦІЯ ДАНИХ

Код для візуалізації, лістинг програми (доповнення):

# Завантаження бібліотек  
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  
# Завантаження датасету  
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()

Результат виконання:



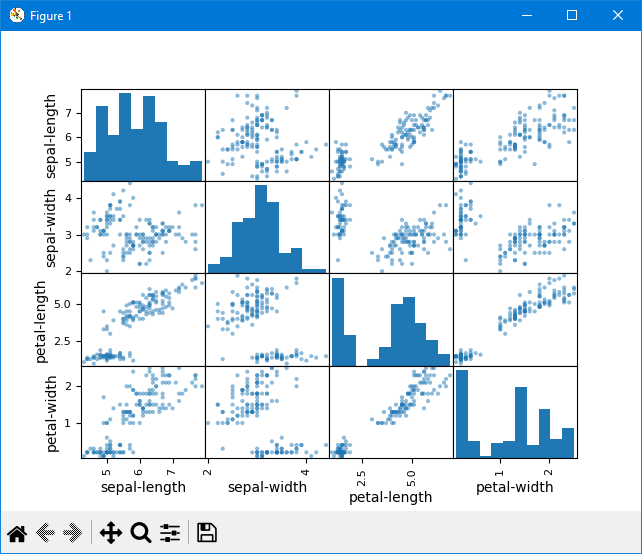
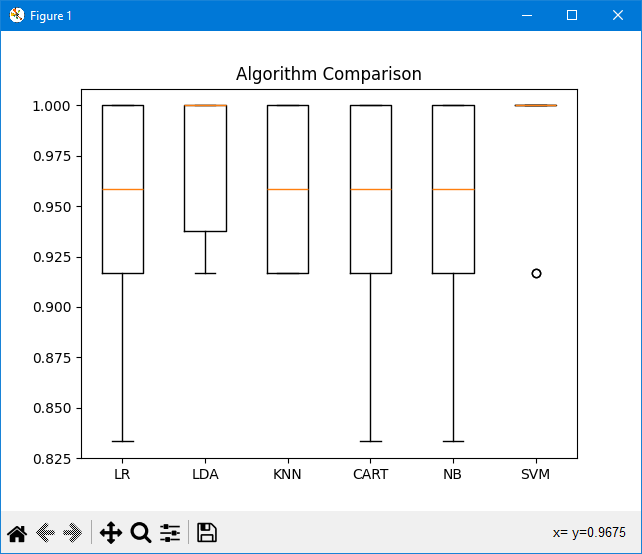


Рис. 3.5 – 3.7. Результат виконання (1 – Діаграма розмаху, 2 – Гістограма розподілу атрибутів датасета, 3 – Матриця діаграм розсіювання)

КРОК 3-4. СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ.

КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ)Будуємо і оцінюємо моделі, лістинг програми (доповнення):

# 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, labels=names)  
pyplot.title('Algorithm Comparison')  
pyplot.show()  
  
# Оцінюємо прогноз  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))



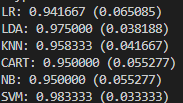


Рис. 3.8 – 3.9. Результат виконання

Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте

найкращим

Виходячи з цих результатів, можна визначити, що найефективнішим методом класифікації для даного набору даних Iris є метод опорних векторів (SVM). Його середня точність становить 0.983333, що перевищує середні показники інших розглянутих методів.

КРОК 5. ОПТИМІЗАЦІЯ ПАРАМЕТРІВ МОДЕЛІ

Частково ми опробовували цей крок у попередньому завданні, коли спробували міняти ядро алгоритму SVM.

КРОК 6. ОТРИМАННЯ ПРОГНОЗУ (ПЕРЕДБАЧЕННЯ НА ТРЕНУВАЛЬ-НОМУ НАБОРІ)

Щоб зробити прогноз, ми викликаємо метод predict Ми можемо протестувати модель на всій вибірці даних, що навчаються, і зробити прогноз та перевірку на контрольній вибірці.

Лістинг програми (доповнення):

# Створюємо прогноз на контрольній вибірці  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)

КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ

Ми можемо оцінити прогноз, порівнявши його з очікуваним результатом контрольної вибірки, а потім обчислити точність класифікації, а також матрицю помилок та звіт про класифікацію.

Лістинг програми (доповнення):

# Оцінюємо прогноз  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))

КРОК 8. ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ

ПЕРЕДБАЧЕННЯ)

Тепер ми можемо отримати прогнози, застосувавши цю модель до нових даних, за якими ми не знаємо правильні мітки.

Повний лістинг програми:

# Завантаження бібліотек  
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  
# Завантаження датасету  
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]  
# 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, 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))  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("Форма масива X\_new: {}".format(X\_new.shape))  
prediction = model.predict(X\_new)  
print("Прогноз: {}".format(prediction))  
print("Спрогнозована мітка: {}".format(prediction[0]))

Результат виконання (отримання прогнозу):

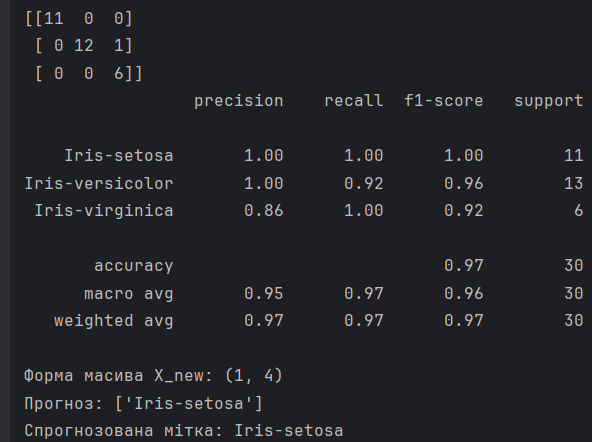


Рис. 3.10. Результат виконання

У висновках опишіть яку якість класифікації за результатами тренування

вдалося досягти та до якого класу належить квітка з кроку 8.

Модель класифікації, побудована на основі даних набору Iris, продемонструвала дуже високу точність, приблизно 96.67%. Квітка зі значеннями ознак (5.0, 2.9, 1.0, 0.2) була визнана як «Iris-setosa» згідно з результатами класифікації.

**Завдання 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

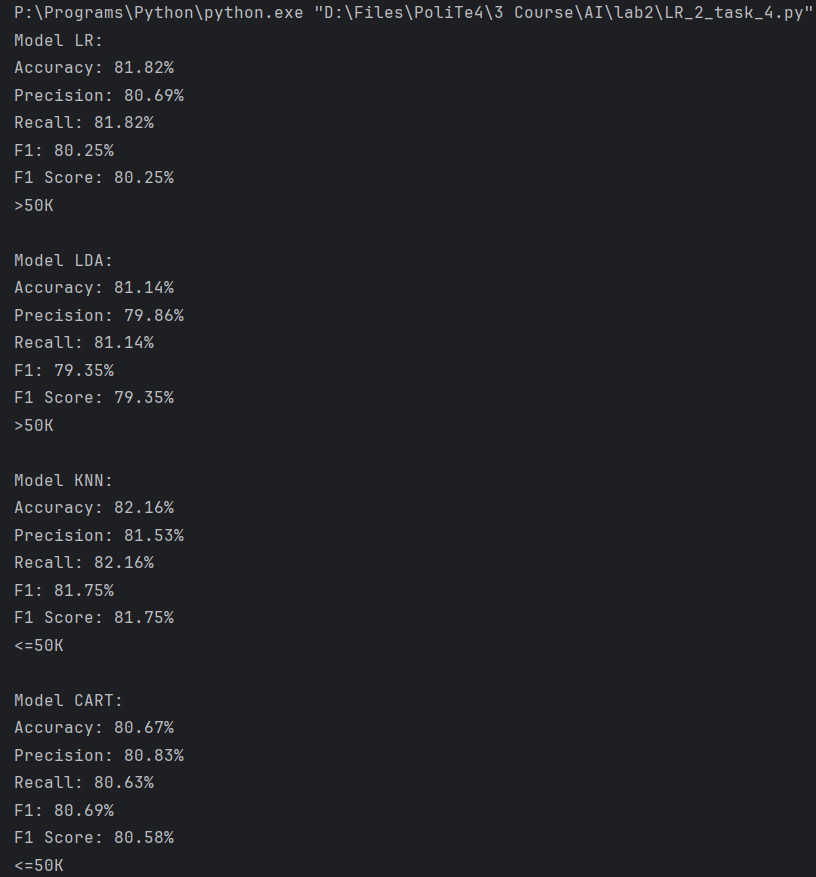
Порівняйте їх між собою. Оберіть найкращий для рішення задачі. Поясніть чому

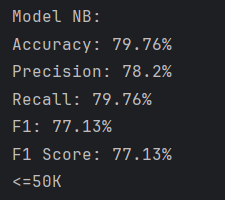
ви так вирішили у висновках до завдання.

Лістинг програми:

import numpy as np  
from matplotlib import pyplot  
from sklearn import preprocessing  
from sklearn.svm import SVC  
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.model\_selection import train\_test\_split, cross\_val\_score, StratifiedKFold  
  
input\_file = "income\_data.txt"  
X = []  
y = []  
count\_class1 = 0  
count\_class2 = 0  
max\_datapoints = 25000  
  
with open(input\_file, "r") as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:  
 break  
  
 if "?" in line:  
 continue  
  
 data = line[:-1].split(", ")  
 if data[-1] == "<=50K" and count\_class1 < max\_datapoints:  
 X.append(data)  
 count\_class1 += 1  
  
 if data[-1] == ">50K" and count\_class2 < max\_datapoints:  
 X.append(data)  
 count\_class2 += 1  
  
X = np.array(X)  
  
label\_encoder = []  
X\_encoded = np.empty(X.shape)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 label\_encoder.append(preprocessing.LabelEncoder())  
 X\_encoded[:, i] = label\_encoder[-1].fit\_transform(X[:, i])  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
scaller = preprocessing.MinMaxScaler(feature\_range=(0, 1))  
X = scaller.fit\_transform(X)  
  
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")))  
  
for name, model in models:  
 print(f"Model {name}:")  
  
 model.fit(X=X, y=y)  
  
 f1 = cross\_val\_score(model, X, y, scoring="f1\_weighted", cv=3)  
 accuracy\_values = cross\_val\_score(model, X, y, scoring="accuracy", cv=3)  
 print(f"Accuracy: {round(100 \* accuracy\_values.mean(), 2)}%")  
 precision\_values = cross\_val\_score(model, X, y, scoring="precision\_weighted", cv=3)  
 print(f"Precision: {round(100 \* precision\_values.mean(), 2)}%")  
 recall\_values = cross\_val\_score(model, X, y, scoring="recall\_weighted", cv=3)  
 print(f"Recall: {round(100 \* recall\_values.mean(), 2)}%")  
 f1\_values = cross\_val\_score(model, X, y, scoring="f1\_weighted", cv=3)  
 print(f"F1: {round(100 \* f1\_values.mean(), 2)}%")  
 print(f"F1 Score: {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 = np.array([-1] \* len(input\_data))  
 count = 0  
 for i, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[i] = item  
 else:  
 input\_data\_encoded[i] = int(label\_encoder[count].transform([item])[0])  
 count += 1  
 input\_data\_encoded = input\_data\_encoded.astype(int)  
 input\_data\_encoded = [input\_data\_encoded]  
 predicted\_class = model.predict(input\_data\_encoded)  
 print(f"{label\_encoder[-1].inverse\_transform(predicted\_class)[0]}\n")  
  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(  
 X, y, test\_size=0.20, random\_state=1  
)  
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()

Результат виконання:





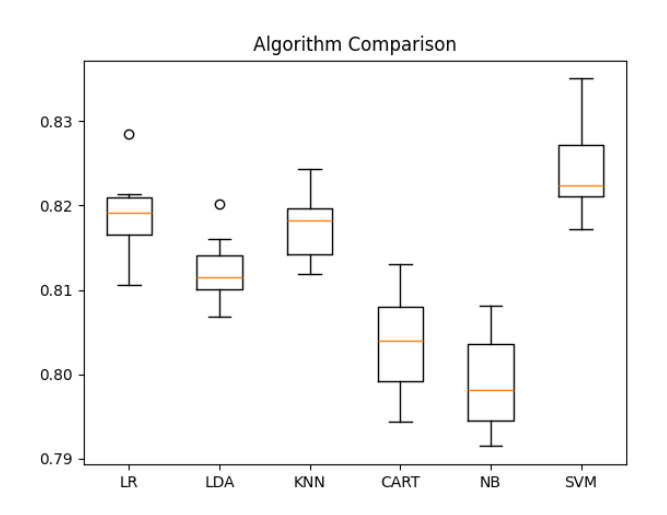


Рис. 4.1 – 4.3. Результат виконання

Оберіть найкращий для рішення задачі. Поясніть чому ви так вирішили у

висновках до завдання.

Після порівняння різних алгоритмів класифікації на наборі даних виявилось, що найкращим є метод опорних векторів (SVM) з точністю приблизно 82.38%. Цей алгоритм також показав найвищий середній F1-счет та найвищу точність. Логістична регресія, лінійний дискримінантний аналіз і метод k-найближчих сусідів також дали добрі результати, але трошки менше, ніж у випадку з SVM. Модель дерева рішень і наївний баєсовський класифікатор показали меншу точність. Отже, для цього завдання оптимальним вибором є метод опорних векторів (SVM) завдяки його високій точності та здатності до збалансованої класифікації об'єктів обох класів.

**Завдання 5:** Класифікація даних лінійним класифікатором Ridge.

Лістинг програми:

import numpy as np  
import seaborn as sns  
from sklearn.datasets import load\_iris  
from sklearn.linear\_model import RidgeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import confusion\_matrix  
from io import BytesIO  
import matplotlib.pyplot as plt  
from sklearn import metrics  
sns.set()  
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)  
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))  
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:

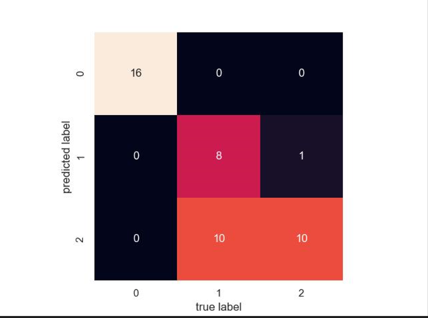
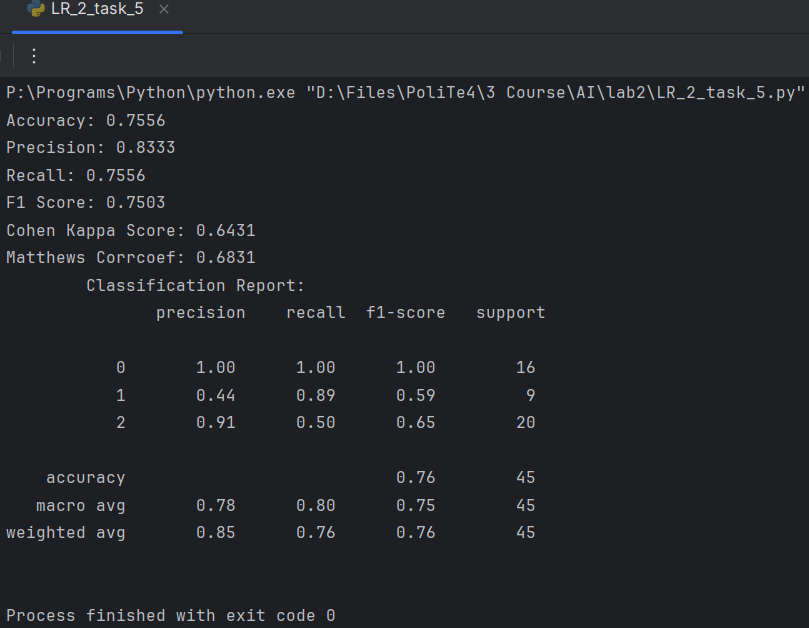


Рис. 5.1 – 5.2. Результат виконання, зображення Confusion.jpg

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.

Налаштування класифікатора Ridge включають параметри:

- tol=1e-2, що вказує, що оптимізатор завершить оптимізацію, якщо зміни параметрів моделі між ітераціями будуть менше за 1e-2.

- solver="sag", який обирає алгоритм стохастичного середнього градієнта для оптимізації класифікатора.

Показники якості для оцінки класифікатора включають:

- Accuracy (точність): визначає частку правильно класифікованих прикладів.

- Precision (точність): вимірює частку правильних позитивних прогнозів серед всіх позитивних прогнозів.

- Recall (повнота): визначає частку правильно виявлених позитивних класів серед усіх істинних позитивних класів.

- F1 Score: гармонічне середнє між точністю та повнотою, представляє компроміс між цими двома метриками.

- Cohen Kappa Score: вимірює узгодженість між спостерігачами у класифікації та коригується на випадковий шанс узгодження.

- Matthews Correlation Coefficient: вимірює ступінь взаємозв'язку між фактичними та прогнозованими класами, враховуючи розподіл всіх чотирьох класифікаційних випадків.

"Confusion.jpg" є матрицею плутанини, де на діагоналі відображені правильно класифіковані елементи, а поза діагоналлю - помилки класифікації для кожного класу.

Коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза використовуються для визначення узгодженості та кореляції між прогнозами та дійсними значеннями, враховуючи випадковість та усі можливі випадки класифікації. Ці метрики допомагають оцінити якість моделі і її здатність до правильної класифікації.

***Висновок***: Під час виконання лабораторної роботи було проведено дослідження різних методів класифікації даних, а також виконано порівняння їх ефективності. Для цих цілей використовувалися спеціалізовані бібліотеки та мова програмування Python.