**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**

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

 **Age** – вік особи (числовий).

 **Workclass** – тип зайнятості (категоріальний).

 **Fnlwgt** – фінальна вага (числовий).

 **Education** – рівень освіти (категоріальний).

 **Education-num** – кількість років навчання (числовий).

 **Marital-status** – сімейний стан (категоріальний).

 **Occupation** – сфера зайнятості (категоріальний).

 **Relationship** – сімейне становище в домогосподарстві (категоріальний).

 **Race** – раса (категоріальний).

 **Sex** – стать (категоріальний).

 **Capital-gain** – дохід від капіталу (числовий).

 **Capital-loss** – втрати від капіталу (числовий).

 **Hours-per-week** – кількість годин роботи на тиждень (числовий).

 **Native-country** – країна народження (категоріальний).

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

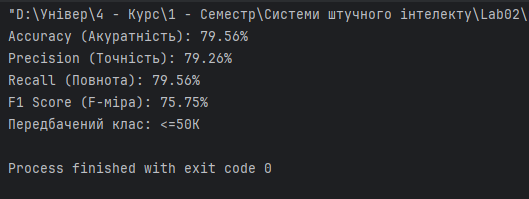
Лістинг програми:  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.metrics import accuracy\_score, precision\_score, recall\_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

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:  
 le = preprocessing.LabelEncoder()  
 X\_encoded[:, i] = le.fit\_transform(X[:, i])  
 label\_encoder.append(le)  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
# Створення SVM-класифікатора  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
  
# Розділення даних на навчальні та тестові вибірки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Навчання класифікатора на навчальних даних  
classifier.fit(X\_train, y\_train)  
  
# Обчислення основних показників якості класифікації  
y\_pred = classifier.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')  
  
# Виведення результатів  
print(f"Accuracy (Акуратність): {accuracy \* 100:.2f}%")  
print(f"Precision (Точність): {precision \* 100:.2f}%")  
print(f"Recall (Повнота): {recall \* 100:.2f}%")  
print(f"F1 Score (F-міра): {f1 \* 100:.2f}%")  
  
# 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 = np.array([-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  
  
# Використання класифікатора для передбачення класу  
input\_data\_encoded = input\_data\_encoded.reshape(1, -1) # Має бути 2D масив для predict  
predicted\_class = classifier.predict(input\_data\_encoded)  
  
# Виведення результату для нової тестової точки даних  
print("Передбачений клас:", label\_encoder[-1].inverse\_transform([predicted\_class[0]])[0])

Виконання програми:



1.3. Зробіть висновок до якого класу належить тестова точка.

З результату ми можемо побачити що наша точка відноситься до класу <=50k

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

з поліноміальним ядром;

з гаусовим ядром;

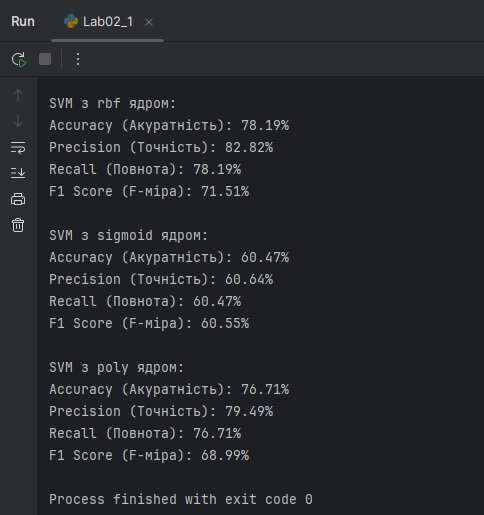
з сигмоїдальним ядром.

Для кожного виду класифікатора отримайте та запишіть у звіт показники якості алгоритму класифікації.

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.svm import SVC  
from sklearn.multiclass import OneVsOneClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_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  
  
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:  
 le = preprocessing.LabelEncoder()  
 X\_encoded[:, i] = le.fit\_transform(X[:, i])  
 label\_encoder.append(le)  
  
X = X\_encoded[:, :-1].astype(int)  
y = X\_encoded[:, -1].astype(int)  
  
# Розділення даних на навчальні та тестові вибірки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
# Функція для навчання та оцінки SVM  
def train\_and\_evaluate\_svm(kernel\_type):  
 print(f"\nSVM з {kernel\_type} ядром:")  
 classifier = SVC(kernel=kernel\_type)  
 classifier.fit(X\_train, y\_train)  
 # Передбачення та оцінка  
 y\_pred = classifier.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')  
 # Виведення результатів  
 print(f"Accuracy (Акуратність): {accuracy \* 100:.2f}%")  
 print(f"Precision (Точність): {precision \* 100:.2f}%")  
 print(f"Recall (Повнота): {recall \* 100:.2f}%")  
 print(f"F1 Score (F-міра): {f1 \* 100:.2f}%")  
  
# Гаусове (радіальне) ядро  
train\_and\_evaluate\_svm('rbf')  
  
# Сигмоїдальне ядро  
train\_and\_evaluate\_svm('sigmoid')  
  
# Поліноміальне ядро  
train\_and\_evaluate\_svm('poly')

Виконання програми:



2.2. У висновках опишіть який з видів SVM найкраще виконує завдання класифікації за результатами тренування.

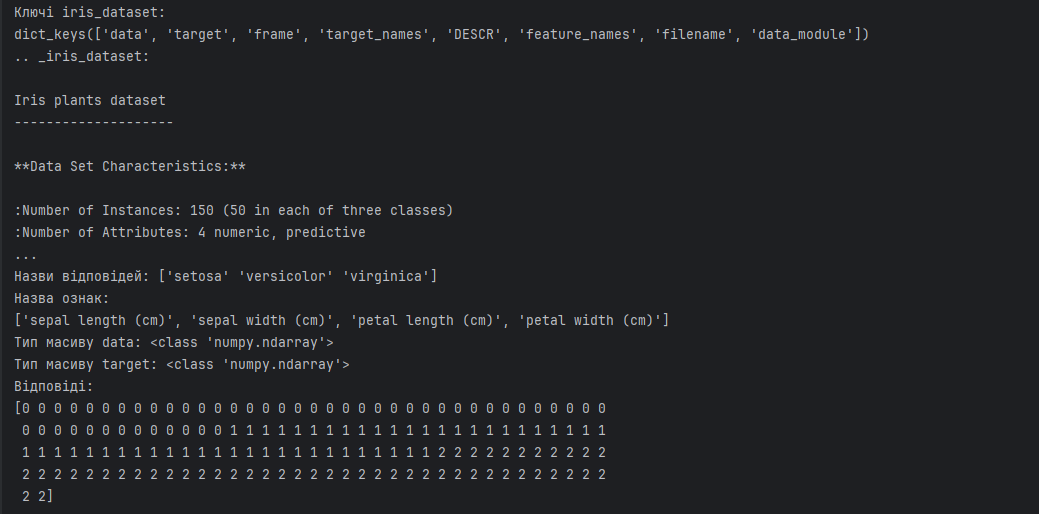
RBF ядро показало найкращі результати за **точністю (precision)** та **повнотою (recall)**, що робить його найбільш ефективним у класифікації в цьому експерименті. Крім того, це ядро виконувалося найшвидше, що є ще однією перевагою.

3.1. Код для ознайомлення зі структурою даних та результати його виконання занесіть у звіт

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("Тип масиву target: {}".format(type(iris\_dataset['target'])))  
  
print("Відповіді:\n{}".format(iris\_dataset['target']))

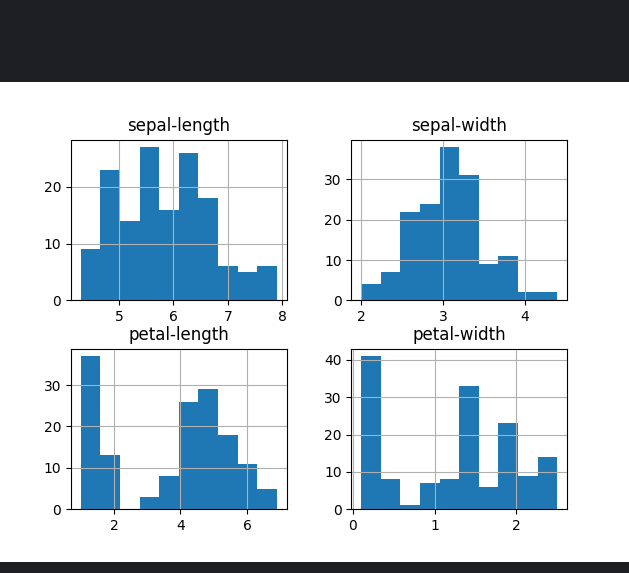
Виконання програми:



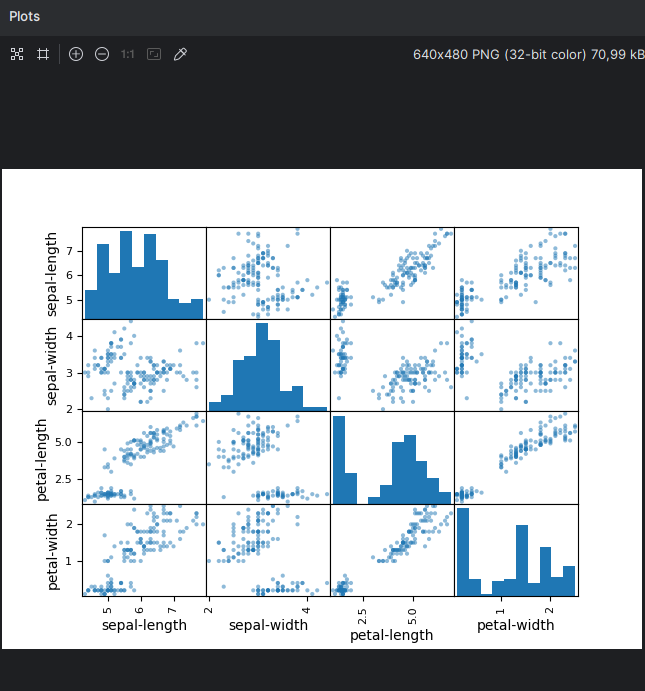
3.2. Графіки функції занесіть у звіт!



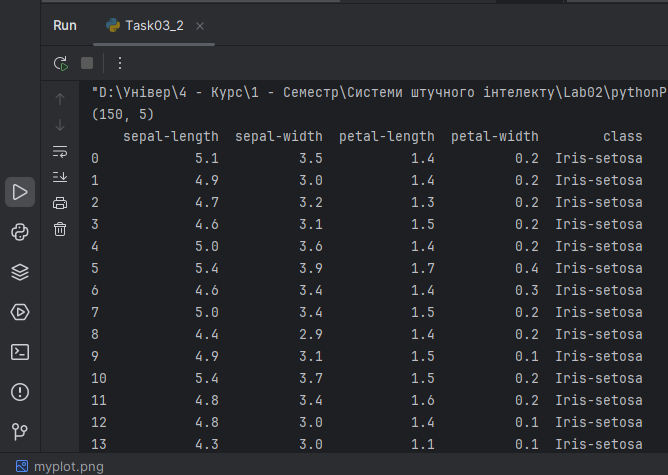
3.3. Графіки функції занесіть у звіт!

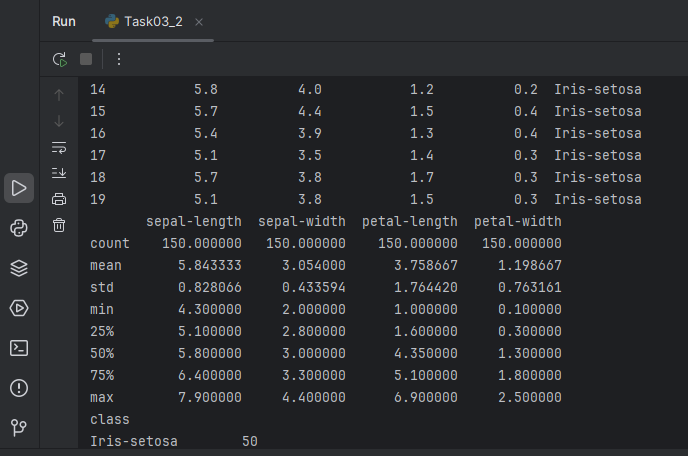


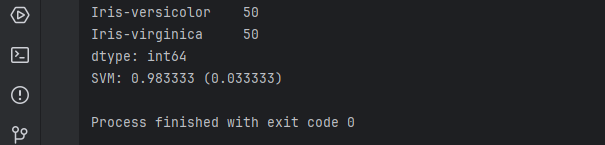
3.4. Код для візуалізації та отримані графіки занесіть у звіт

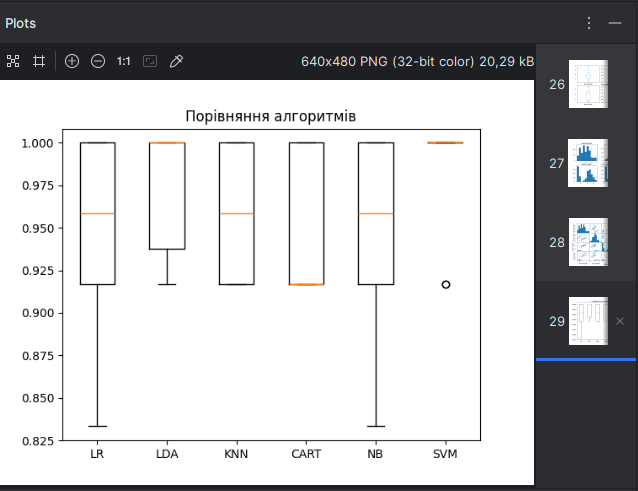


3.5 Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим







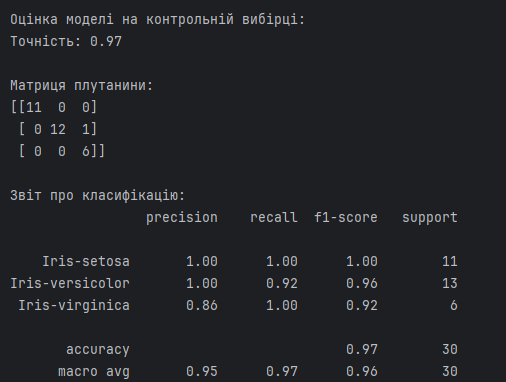


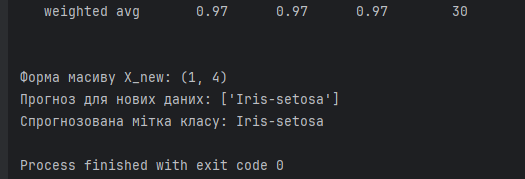
Серед усіх методів класифікації, я б обрав **Support Vector Machine (SVM)** з нелінійним ядром як найкращий варіант для задачі класифікації, де дані мають складну структуру, і ми шукаємо найкращу точність. Завдяки своїй гнучкості, стабільності та здатності працювати з нелінійними межами між класами, цей метод перевершує інші класичні алгоритми класифікації в багатьох задачах.

3.6. Коди та результати занесіть у звіт. У висновках опишіть яку якість класифікації за результатами тренування вдалося досягти та до якого класу належить квітка з кроку 8.

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  
from sklearn.multiclass import OneVsRestClassifier  
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("Розмірність датасету: {}".format(dataset.shape))  
  
# Зріз даних head  
print("\nПерші 20 рядків даних:")  
print(dataset.head(20))  
  
# Стастичні зведення методом describe  
print("\nСтатистичні зведення даних:")  
print(dataset.describe())  
  
# Розподіл за атрибутом class  
print("\nРозподіл класів:")  
print(dataset.groupby('class').size())  
  
# Діаграма розмаху  
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)  
pyplot.title('Діаграма розмаху для атрибутів')  
pyplot.show()  
  
# Гістограма розподілу атрибутів датасета  
dataset.hist()  
pyplot.title('Гістограма розподілу атрибутів')  
pyplot.show()  
  
# Матриця діаграм розсіювання  
scatter\_matrix(dataset)  
pyplot.title('Матриця діаграм розсіювання')  
pyplot.show()  
  
# Розділення датасету на навчальну та контрольну вибірки  
array = dataset.values  
X = array[:, 0:4] # Вибір перших 4-х стовпців  
Y = array[:, 4] # Вибір 5-го стовпця  
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', OneVsRestClassifier(LogisticRegression(solver='liblinear'))))  
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('{}: {:.2f} ± {:.2f}'.format(name, cv\_results.mean(), cv\_results.std()))  
  
# Порівняння алгоритмів  
pyplot.boxplot(results, tick\_labels=names) # Зміна labels на tick\_labels  
pyplot.title('Порівняння алгоритмів')  
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("\nОцінка моделі на контрольній вибірці:")  
print("Точність: {:.2f}".format(accuracy\_score(Y\_validation, predictions)))  
print("\nМатриця плутанини:")  
print(confusion\_matrix(Y\_validation, predictions))  
print("\nЗвіт про класифікацію:")  
print(classification\_report(Y\_validation, predictions))  
  
# Прогноз для нових даних  
X\_new = np.array([[5, 2.9, 1, 0.2]])  
print("\nФорма масиву X\_new: {}".format(X\_new.shape))  
prediction = model.predict(X\_new)  
  
# Виводимо результати прогнозу  
print("Прогноз для нових даних: {}".format(prediction))  
print("Спрогнозована метка класу: {}".format(prediction[0]))





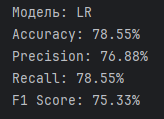
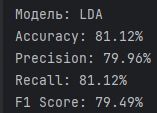
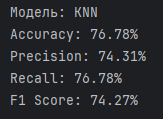
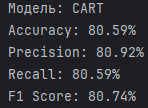
Модель передбачила, що ця квітка належить до класу **Iris-setosa**. Це підтверджує правильність класифікації, оскільки виміри цієї квітки узгоджуються з типовими характеристиками класу Iris-setosa.

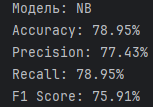
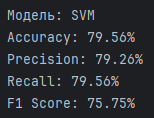
4.1. Порівняння якості класифікаторів для набору даних income\_data.txt

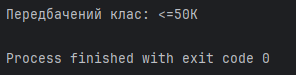
Лістинг програми:

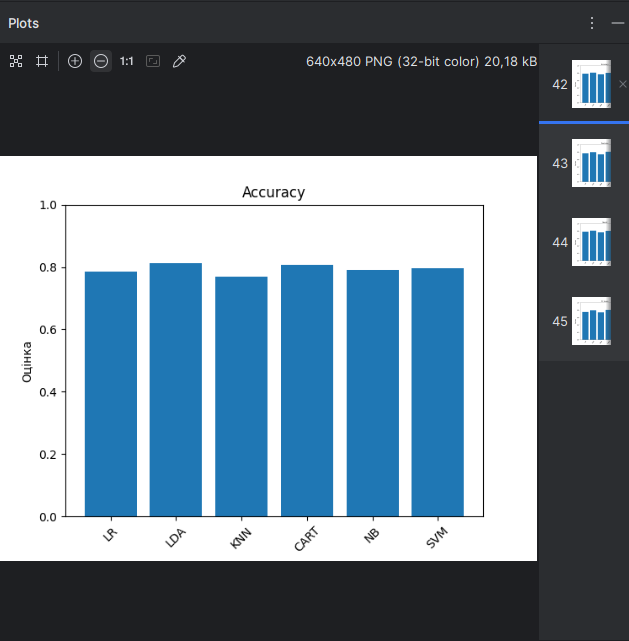
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import preprocessing  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  
from sklearn.svm import LinearSVC  
from sklearn.multiclass import OneVsOneClassifier  
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  
import pandas as pd  
  
# 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.strip().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, dtype=object)  
for i, item in enumerate(X[0]):  
 if item.isdigit():  
 X\_encoded[:, i] = X[:, i]  
 else:  
 le = preprocessing.LabelEncoder()  
 X\_encoded[:, i] = le.fit\_transform(X[:, i])  
 label\_encoder.append(le)  
  
X = X\_encoded[:, :-1].astype(int) # Вхідні ознаки  
y = X\_encoded[:, -1].astype(int) # Мітки класу  
  
# Завантажуємо алгоритми моделі  
models = []  
models.append(('LR', LogisticRegression(solver='liblinear')))  
models.append(('LDA', LinearDiscriminantAnalysis()))  
models.append(('KNN', KNeighborsClassifier()))  
models.append(('CART', DecisionTreeClassifier()))  
models.append(('NB', GaussianNB()))  
models.append(('SVM', OneVsOneClassifier(LinearSVC(random\_state=0))))  
  
# Оцінка кожної моделі  
results = []  
names = []  
  
for name, model in models:  
 # Розділення даних на навчальні та тестові вибірки  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
 # Навчання моделі  
 model.fit(X\_train, y\_train)  
  
 # Передбачення на тестових даних  
 y\_pred = model.predict(X\_test)  
  
 # Обчислення показників якості  
 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')  
  
 # Додавання результатів  
 results.append((accuracy, precision, recall, f1))  
 names.append(name)  
  
 print(f"Модель: {name}")  
 print(f"Accuracy: {accuracy \* 100:.2f}%")  
 print(f"Precision: {precision \* 100:.2f}%")  
 print(f"Recall: {recall \* 100:.2f}%")  
 print(f"F1 Score: {f1 \* 100:.2f}%\n")  
  
# Порівняння алгоритмів  
labels = ['Accuracy', 'Precision', 'Recall', 'F1 Score']  
metrics = np.array(results)   
  
for i, label in enumerate(labels):  
 plt.figure(i)  
 plt.bar(names, metrics[:, i])  
 plt.title(label)  
 plt.ylabel('Оцінка')  
 plt.xticks(rotation=45)  
 plt.ylim(0, 1)  
 plt.show()  
  
# 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 = np.array([-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  
  
# Використання класифікатора для передбачення класу  
input\_data\_encoded = input\_data\_encoded.reshape(1, -1) # Має бути 2D масив для predict  
predicted\_class = models[-1][1].predict(input\_data\_encoded) # Остання модель - SVM  
  
# Виведення результату для нової тестової точки даних  
print("Передбачений клас:", label\_encoder[-1].inverse\_transform([predicted\_class[0]])[0])

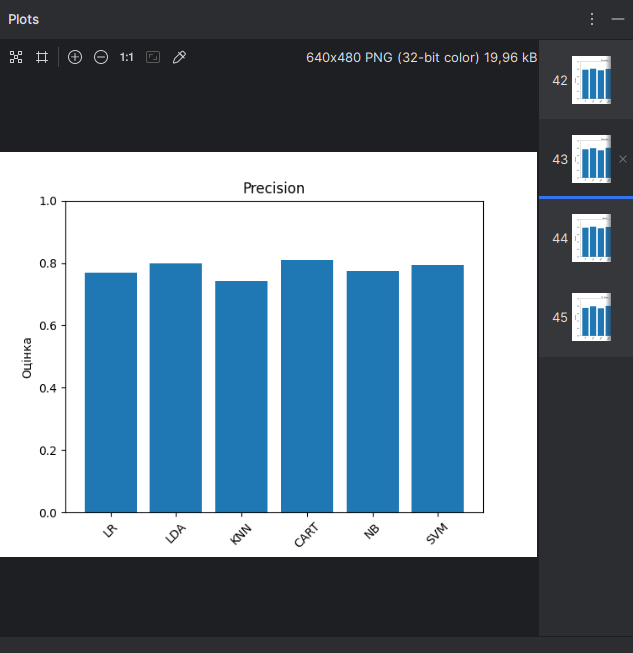
Виконання програми:

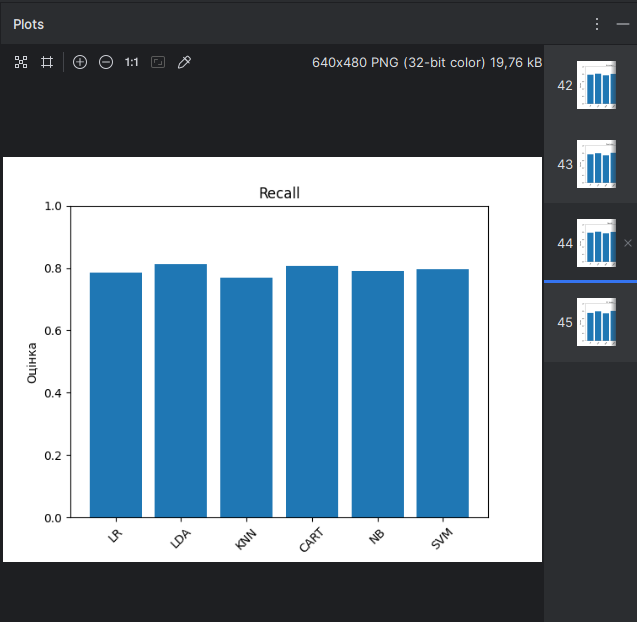
   

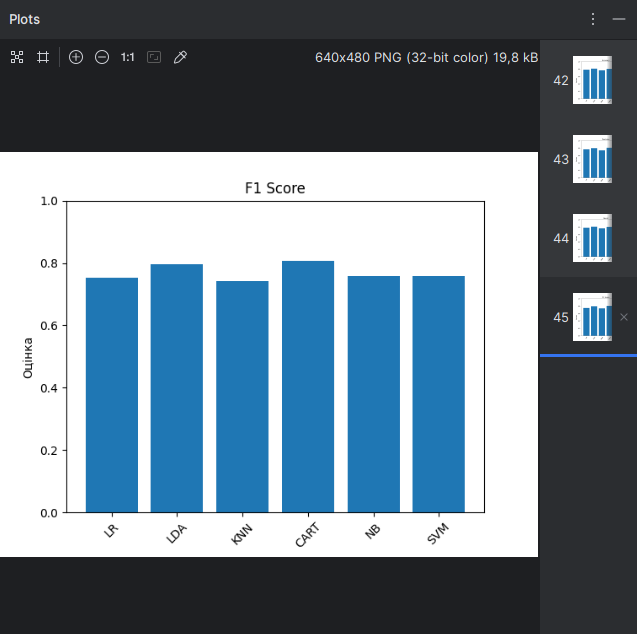
 









Класифікація даних лінійним класифікатором Ridge

5.1. Виправте код та виконайте класифікацію.

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 sklearn.model\_selection import train\_test\_split  
from io import BytesIO # needed for plot  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Завантаження даних  
iris = load\_iris()  
X, y = iris.data, iris.target  
  
# Розбиття даних на навчальну і тестову вибірки  
Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
  
# Ініціалізація та навчання класифікатора Ridge  
clf = RidgeClassifier(tol=1e-2, solver="sag")  
clf.fit(Xtrain, ytrain)  
  
# Прогнозування на тестових даних  
ypred = clf.predict(Xtest)  
  
# Розрахунок показників якості  
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(ytest, ypred))  
  
# Побудова матриці плутанини  
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')  
  
# Збереження матриці плутанини у файл  
plt.savefig("Confusion.jpg")  
  
# Збереження у SVG формат  
f = BytesIO()  
plt.savefig(f, format="svg")

5.2. Опишіть які налаштування класифікатора Ridge тут використані та що вони позначають.

1. **tol=1e-2** — Це параметр толерантності (tolerance), який визначає точність збіжності алгоритму. Якщо різниця в значеннях функції втрат між ітераціями менша за це значення, процес навчання зупиняється.
2. **solver="sag"** — Це вибір алгоритму оптимізації. SAG (Stochastic Average Gradient) — це варіант градієнтного спуску, який обчислює середній градієнт на кожному кроці, що дозволяє працювати з великими наборами даних ефективніше.

5.3. Опишіть які показники якості використовуються та їх отримані результати.

 **Точність (Accuracy)** — це частка правильних передбачень серед усіх передбачень. Точність показує, наскільки добре модель класифікує дані в цілому.

 **Точність (Precision)** — це частка правильних позитивних передбачень серед усіх передбачених позитивних результатів. Показує, наскільки модель точна в прогнозах позитивних класів.

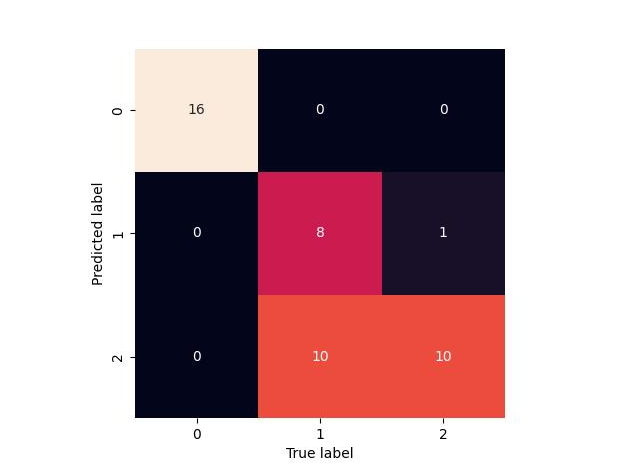
 **Повнота (Recall)** — це частка правильних позитивних передбачень серед усіх реальних позитивних результатів. Вона показує, наскільки добре модель захоплює всі позитивні приклади.

 **F1-міра (F1 Score)** — це гармонічне середнє між точністю і повнотою, що дозволяє отримати збалансовану метрику якості класифікації.

 **Коефіцієнт Коена Каппа (Cohen Kappa Score)** — це показник, який вимірює ступінь узгодженості між передбаченнями і реальними значеннями, коригуючи це на випадкові збіги. Він допомагає оцінити ефективність моделі в порівнянні з випадковими передбаченнями.

 **Кореляція Метьюза (Matthews Corrcoef)** — це показник, який оцінює якість бінарної класифікації. Він враховує усі чотири показники матриці плутанини (TP, FP, TN, FN) і є гармонічним середнім для більш коректної оцінки класифікації, особливо для незбалансованих даних.

5.4. Вставте у звіт та поясніть зображення Confusion.jpg



матриця плутанини показує результат роботи вашого класифікатора Ridge на наборі даних Iris. Візуально відображені три класи (0, 1, 2), які представляють різні види квітки Iris. Ось що означають дані у цій матриці:

* **На діагоналі**: елементи (0,0), (1,1), (2,2) — це правильно передбачені класи. Тобто, модель правильно передбачила 16 квіток класу 0, 8 квіток класу 1, і 10 квіток класу 2.
* **Позадіагональні елементи**: це неправильно передбачені класи.
  + Елемент (2,1) означає, що модель передбачила 10 квіток як клас 2, тоді як вони належали до класу 1.
  + Елемент (1,2) означає, що одна квітка з класу 2 була неправильно класифікована як клас 1.

5.5. Опишіть, що таке коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза. Що вони тут розраховують та що показують.

**Коефіцієнт Коена Каппа**:

* Використовується для оцінки узгодженості між двома спостерігачами або передбаченнями та реальними значеннями, виправляючи очікувану узгодженість випадкових передбачень.
* Значення від -1 (повна незгода) до 1 (повна узгодженість).
* У нашому випадку значення близьке до 1, що означає помірну узгодженість моделі з реальними даними.

**Коефіцієнт кореляції Метьюза**:

* Це більш збалансований показник якості класифікації, особливо корисний для незбалансованих наборів даних, оскільки враховує всі чотири компоненти матриці плутанини.
* Значення від -1 (погана модель) до 1 (ідеальна модель).
* У нашому випадку значення близьке до 1, що свідчить про помірну якість класифікації.

Посилання на ГітХаб:

https://github.com/KaidanovychBohdan/SystemOfAI

**Висновок**: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчитися їх порівнювати.