**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

Серед вхідних ознак можливо точно охарактеризувати наступні ознаки:

* Вік – чисельна
* Форма працевлаштування (державне, приватне тощо) – категоріальна
* Рівень освіти – категоріальна
* Досвід роботи – чисельна
* Сімейний стан – категоріальна
* Тип працевлаштування (тип роботи) – категоріальна
* Поточна роль у сім’ї – категоріальна
* Колір шкіри – категоріальна
* Стать – категоріальна
* Країна проживання - категоріальна

Решта ознак є чисельними, охарактеризувати їх роль та назвати неможливо через відсутність назв колонок у текстовому файлі з даними та можливих підказок у самих даних.

Лістинг коду файлу Task\_1.py:

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

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)  
  
classifier = OneVsOneClassifier(LinearSVC(random\_state=0))  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
classifier.fit(X\_train, y\_train)  
y\_test\_pred = classifier.predict(X\_test)  
  
accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)  
precision = cross\_val\_score(classifier, X, y, scoring='precision\_weighted', cv=3)  
recall = cross\_val\_score(classifier, X, y, scoring='recall\_weighted', cv=3)  
f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)  
print("Accuracy: " + str(round(100\*accuracy.mean(), 2)) + "%")  
print("Precision: " + str(round(100\*precision.mean(), 2)) + "%")  
print("Recall: " + str(round(100\*recall.mean(), 2)) + "%")  
print("F1 score: " + str(round(100\*f1.mean(), 2)) + "%")  
  
  
def predict(input\_data):  
 input\_data\_encoded = [-1] \* len(input\_data)  
 count = 0  
 for index, item in enumerate(input\_data):  
 if item.isdigit():  
 input\_data\_encoded[index] = int(input\_data[index])  
 else:  
 input\_data\_encoded[index] = label\_encoder[count].transform([input\_data[index]])[0]  
 count += 1  
  
 input\_data\_encoded = np.array(input\_data\_encoded)  
 predicted\_class = classifier.predict([input\_data\_encoded])  
 print("Input data:", input\_data)  
 print("Predicted class:", label\_encoder[-1].inverse\_transform(predicted\_class)[0])  
  
  
input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',  
 'Male', '0', '0', '40', 'United-States']  
predict(input\_data)

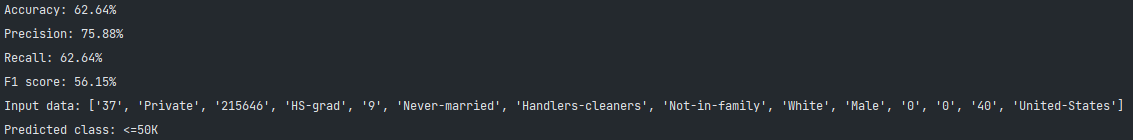


Рис.2.1 – Проаналізовані якість, точність та повнота алгоритму, прогнозований клас вхідних даних

Завдання 2. Порівняння якості класифікаторів SVM з нелінійними ядрами

До порівняння буде взято поліномінальне, гаусове, сигмоїдальне ядра, для цього буде скопійовано попередній код та змінено рядок створення класифікатора, а саме змінено спосіб ініціалізації ядра.

Через велику кількість даних, час навчання для поліномінального ядра займає дуже значну кількість часу, а саме має квадратичну залежність від кількості даних. Для можливого отримання швидкого результату під час виконання роботи, було встановлено максимальний ліміт на кількість даних у 100 рядків. Для інших ядер було встановлено ліміт в 15000 рядків.

Лістинг змін для поліномінального ядра, файл Task\_2\_1.py:

max\_datapoints = 100  
  
with open(input\_file, 'r') as f:  
 for line in f.readlines():  
 if count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints or len(X) >= max\_datapoints:  
 break

\* code skipped \*

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, random\_state=0))

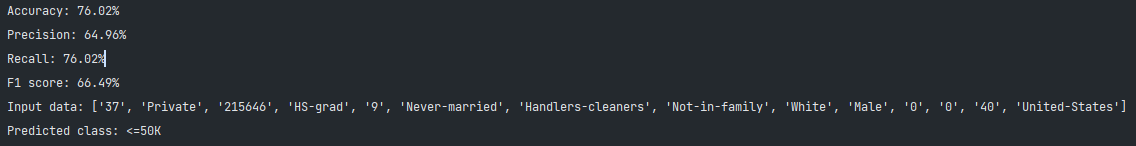


Рис.2.2 – Результат роботи поліномінального ядра

Лістинг змін для гаусового ядра, файл Task\_2\_2.py:

classifier = OneVsOneClassifier(SVC(kernel='rbf'))

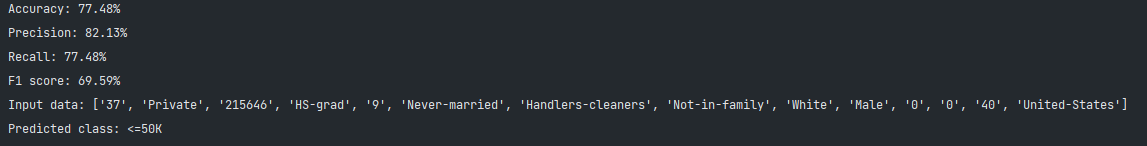


Рис.2.3 – Результат роботи гаусового ядра

Лістинг змін для сигмоїдального ядра, файл Task\_2\_3.py:

classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))

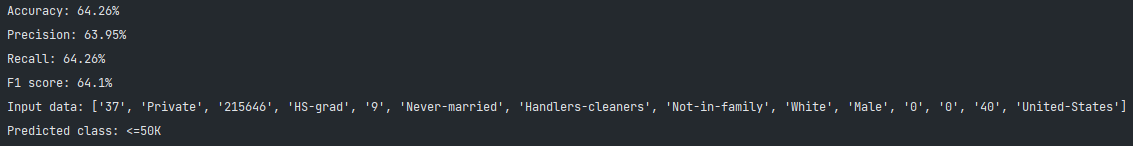


Рис.2.4 – Результат роботи сигмоїдального ядра

За даних умов нерівномірності використання даних, найкращий результат надає гаусове ядро. За використання всіх наявних даних, результати можуть бути іншими, проте для їх отримання необхідно надати дуже багато часу на навчання поліномінальому ядру.

Завдання 3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Для аналізу вхідних даних було створено окремий файл Task\_3\_data.py, лістинг файлу:

from sklearn.datasets import load\_iris  
iris\_dataset = load\_iris()  
  
print(f"Ключі iris\_dataset: \n{iris\_dataset.keys()}")  
print(iris\_dataset['DESCR'][:193] + "\n...")  
  
print(f"Назви відповідей: {iris\_dataset['target\_names']}")  
print(f"Назви ознак: {iris\_dataset['feature\_names']}")  
print(f"Тип даних: {type(iris\_dataset['data'])}")  
print(f"Розмір даних: {iris\_dataset['data'].shape}")  
print(f"Перші п'ять рядків даних:\n{iris\_dataset['data'][:5]}")  
print(f"Тип масиву відповідей: {type(iris\_dataset['target'])}")  
print(f"Розмір масиву відповідей: {iris\_dataset['target'].shape}")  
print(f"Відповіді:\n{iris\_dataset['target']}")

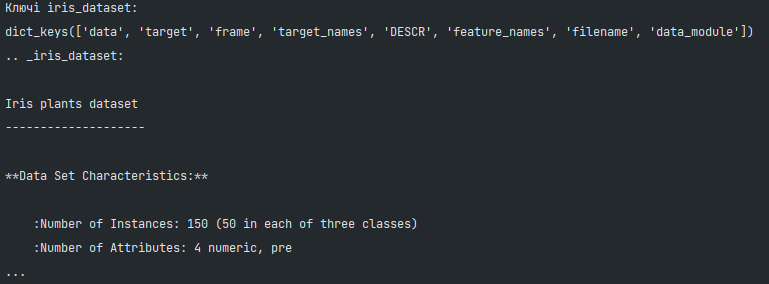


Рис.2.5 – Ключі словника вхідних даних та опис

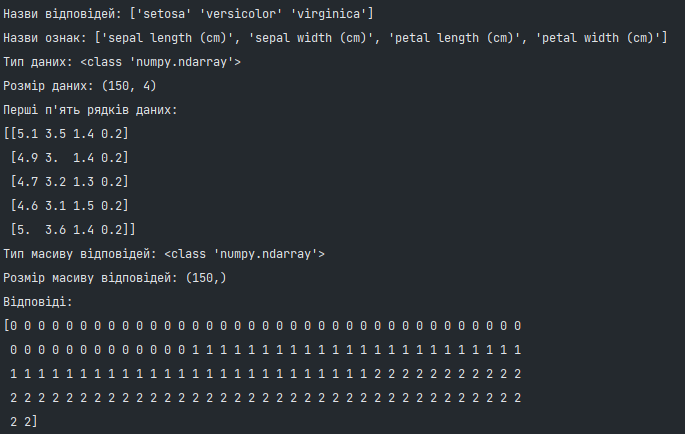


Рис.2.6 – Детальна інформація про відповіді, ознаки та дані

Для класифікації даних було створено файл Task\_3\_classify.py, лістинг файлу:

from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot as plt  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score, StratifiedKFold  
from sklearn.metrics import classification\_report, confusion\_matrix, 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)  
  
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)  
plt.show()  
  
# Гістограми  
dataset.hist()  
plt.show()  
  
# Матриця розсіювання  
scatter\_matrix(dataset)  
plt.show()  
  
# Розділення набору даних на навчальний та тестовий  
array = dataset.values  
X = array[:, 0:4]  
Y = array[:, 4]  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, Y, test\_size=0.20, random\_state=1)  
  
# Перевірка алгоритмів  
models = [  
 ('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')),  
 ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()),  
 ('CART', DecisionTreeClassifier()),  
 ('NB', GaussianNB()),  
 ('SVM', SVC(gamma='auto'))  
]  
  
# Оцінка кожного алгоритму  
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(f'{name}: {cv\_results.mean()} ({cv\_results.std()})')  
  
# Порівняння алгоритмів  
plt.boxplot(results, labels=names)  
plt.title('Алгоритми порівняння')  
plt.show()  
  
# Побудова моделі на основі алгоритму SVM  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
# Оцінка моделі  
print(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))  
  
# Вивід результатів по власним даним  
X\_new = [[5.0, 3.6, 1.3, 0.25], [5.9, 3.0, 5.1, 1.8], [6.3, 3.3, 6.0, 2.5], [5.8, 2.7, 5.1, 1.9], [5.1, 3.5, 1.4, 0.2]]  
predictions = model.predict(X\_new)  
print(f"X\_new: {X\_new}\nPredictions: {predictions}")

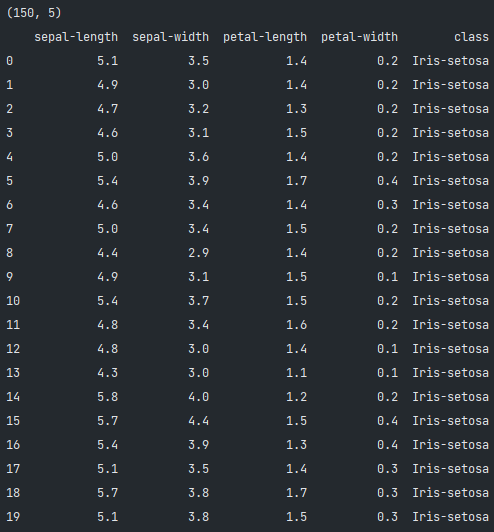


Рис.2.7 – Розмір масиву даних та перші 20 записів

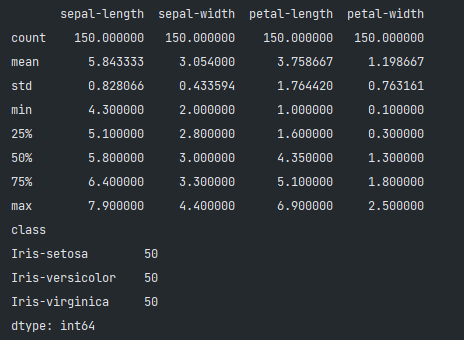


Рис.2.8 – Характеристики даних, кількість по класам та тип даних

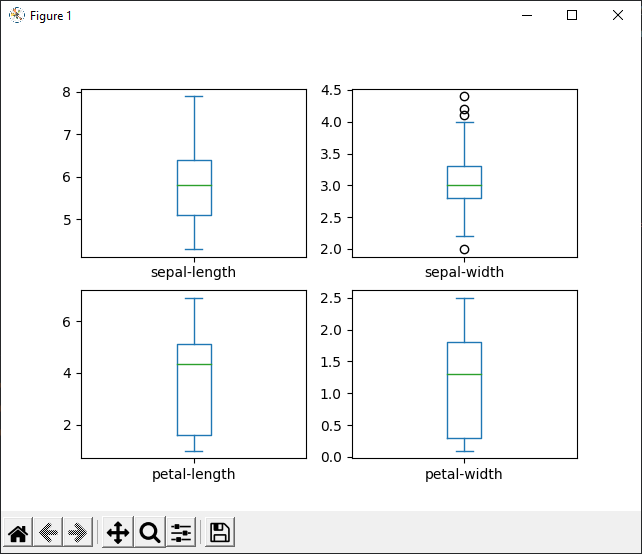


Рис.2.9 – Діаграма розмаху

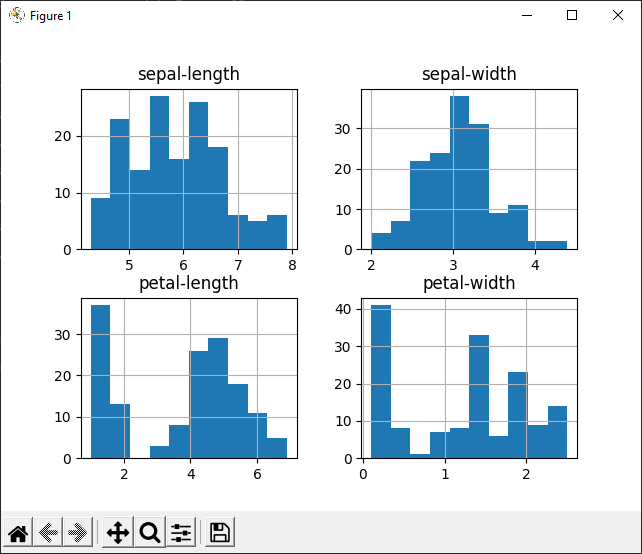


Рис.2.10 – Гістограма даних

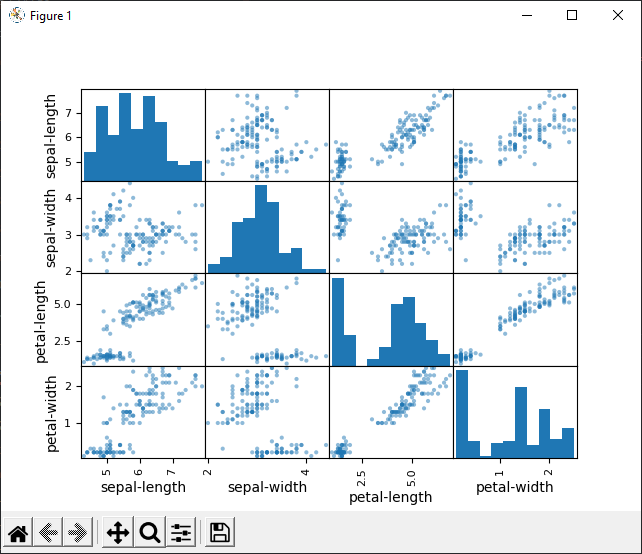


Рис.2.11 – Багатовимірна діаграма розсіювання

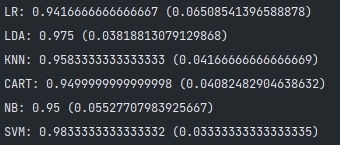


Рис.2.12 – Отримані результати навчання моделей (лише accuracy)

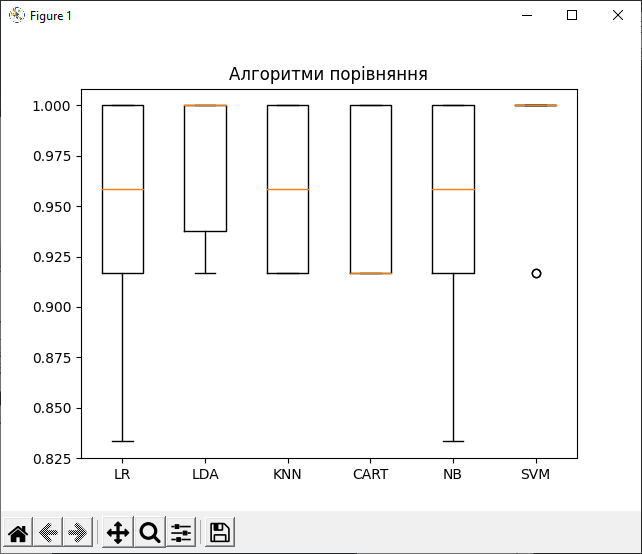


Рис.2.13 – Діаграма розмаху атрибутів вхідних даних

Найкращим за власною думкою є SVM через найкращі результати класифікації та якості роботи.

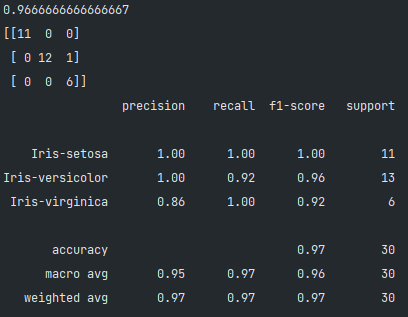


Рис.2.14 – Якість, матриця помилок та звіт по класифікації даних через SVC

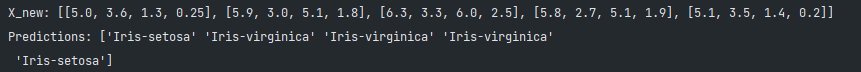


Рис.2.15 – Прогнозування класів власних даних

***Висновок:*** під час виконання завдань лабораторної із використанням спеціалізованих бібліотек та мови програмування Python було досліджено різні методи класифікації даних та отримано навички їх порівнювати.

Протягом роботи було створено кілька файлів формату .py для кожного завдання для окремого порівняння роботи різних класифікаторів між собою та окремого відображення даних та їх обробки.

Для виконання робіт було використано функції бібліотек pandas, matplotlib, scikit-learn та відображено результати класифікації з характеристиками різних способів класифікації.