**Практична робота № 2**

**Варіант 13**

**Попередня обробка та контрольована**

**Класифікація даних**

***Мета:*** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.

**Хід роботи:**

**Завдання 2.1:** 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). Випишіть у звіт всі 14 ознак з набору даних – їх назви та що вони позначають та вид (числові чи категоріальні):

* Вік – чисельна
* Тип працевлаштування

– категоріальна

* Рівень освіти – категоріальна
* Безперервність освіти – чисельна
* Сімейний стан – категоріальна
* Професія – категоріальна
* Відносини – категоріальна
* Раса – категоріальна
* Стать – категоріальна
* Приріст капіталу – чисельна
* Витрати – чисельна
* Кількість годин роботи на тиждень – чисельна
* Країна наордження – категоріальна
* Досвід роботи – чисельна

Лістинг файлу task-1.py

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)  
  
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])  
input\_data = ['48', 'State-gov', '102628', 'Doctorate', '19', 'Married-civ-spouse', 'Prof-specialty', 'Husband',   
 'Asian-Pac-Islander', 'Male', '0', '0', '40', 'United-States']  
predict(input\_data)

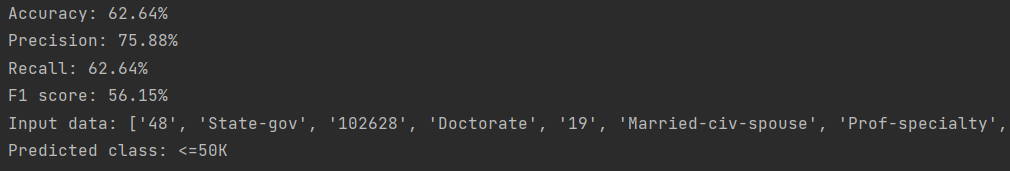


Рис.2.1. task-1.py

**Завдання 2.2:** Порівняння якості класифікаторів SVM з нелінійними ядрами. Порівнювати між собою будемо поліномінальне, гаусове, сигмоїдальне ядра.

Фрагмент лістингу файлу task-2\_1.py (поліномінальне ядро)

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  
 if '?' in line:  
 continue

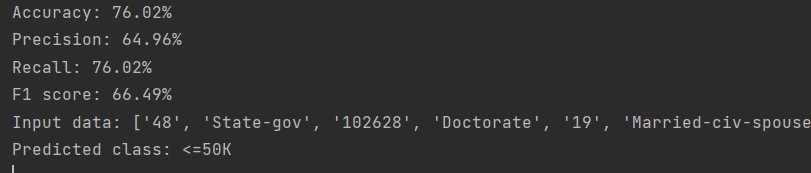


Рис.2.2. task-2\_1.py

Фрагмент лістингу файлу task-2\_2.py (гаусове ядро)

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

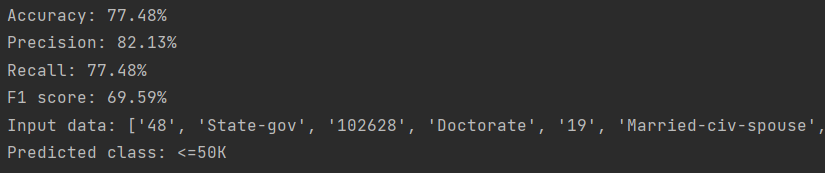


Рис.2.2. task-2\_2.py

Лістинг файлу task-2\_3.py (сигмоїдальне ядро)

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

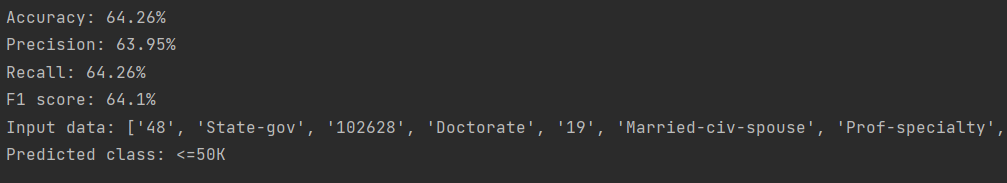


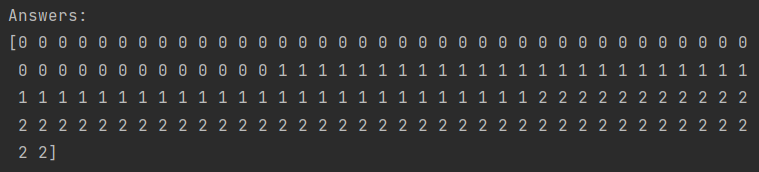
Рис.2.2. task-2\_3.py

**Завдання 2.3:** Порівняння якості класифікаторів на прикладі класифікації сортів ірисів. Необхідно класифікувати сорти ірисів за деякими їх характеристиками: довжина та ширина пелюсток, а також довжина та ширина чашолистків.

Також, в наявності є вимірювання цих же характеристик ірисів, які раніше дозволили досвідченому експерту віднести їх до сортів: setosa, versicolor і virginica.

Лістинг файлу task-2\_4.py

from sklearn.datasets import load\_iris  
iris\_dataset = load\_iris()  
  
print(f"Iris\_dataset keys: \n{iris\_dataset.keys()}")  
print(iris\_dataset['DESCR'][:193] + "\n...")  
  
print(f"Response names: {iris\_dataset['target\_names']}")  
print(f"Feature names: {iris\_dataset['feature\_names']}")  
print(f"Data type: {type(iris\_dataset['data'])}")  
print(f"Data size: {iris\_dataset['data'].shape}")  
print(f"The first five lines of data:\n{iris\_dataset['data'][:5]}")  
print(f"Response array type: {type(iris\_dataset['target'])}")  
print(f"Size of response array: {iris\_dataset['target'].shape}")  
print(f"Answers:\n{iris\_dataset['target']}")



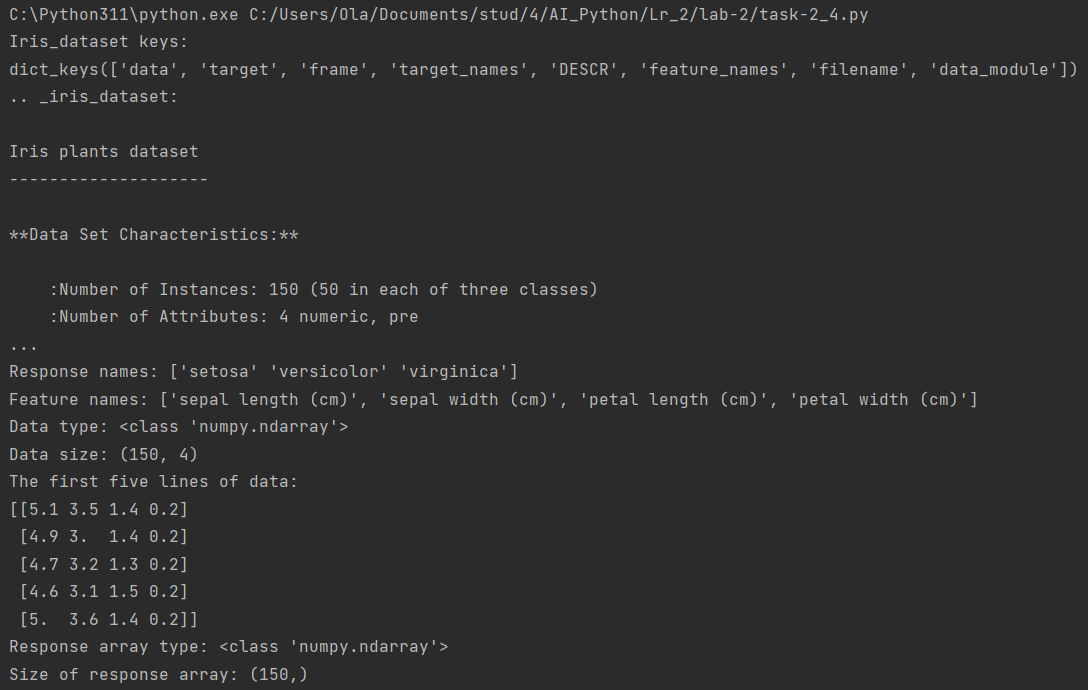
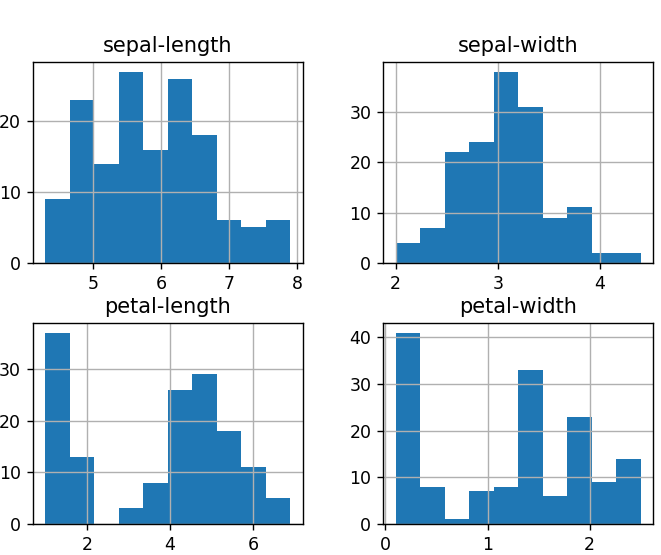
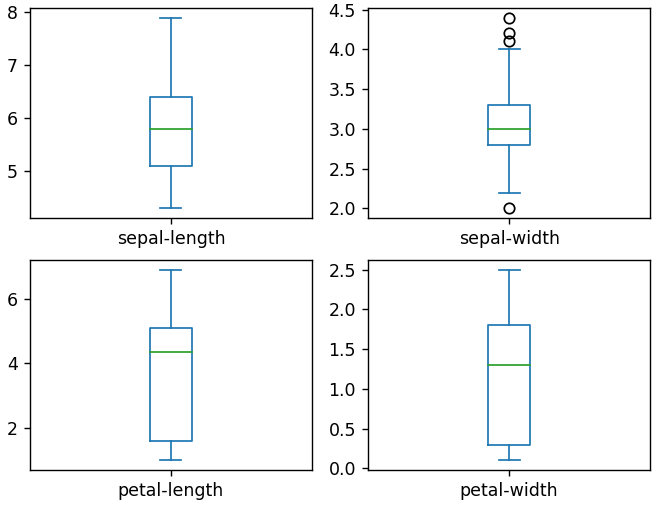


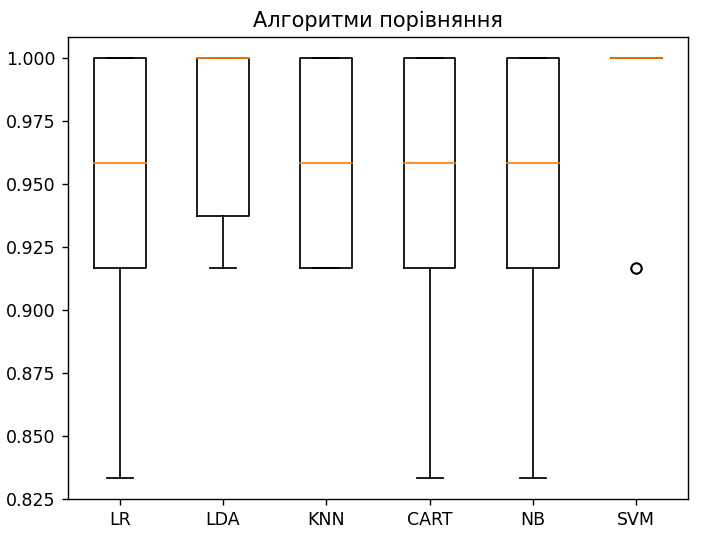
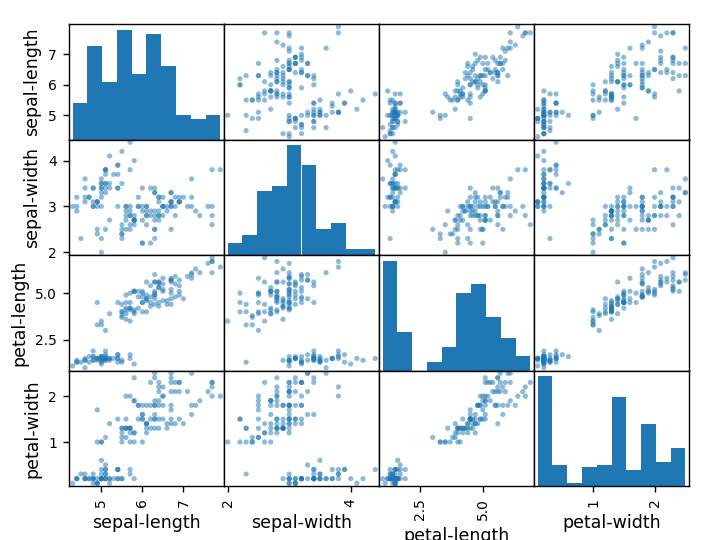
Рис.2.2. task-2\_4.py

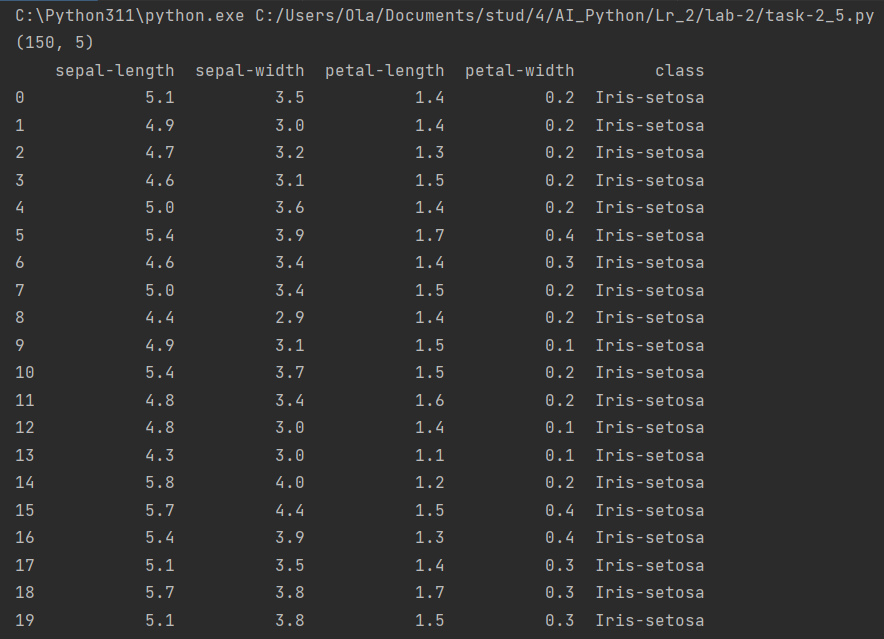
Лістинг файлу task-2\_5.py

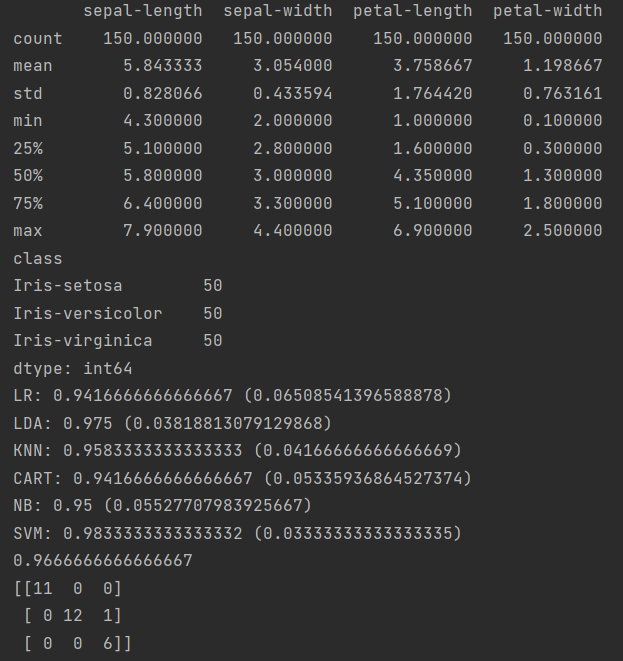
from pandas import read\_csv  
from pandas.plotting import scatter\_matrix  
from matplotlib import pyplot as plt  
from sklearn.model\_selection import train\_test\_split, cross\_val\_score, StratifiedKFold  
from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score  
from 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())  
  
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)  
plt.show()  
  
dataset.hist()  
plt.show()

scatter\_matrix(dataset)  
plt.show()  
  
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)  
  
models = [  
 ('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')),  
 ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()),  
 ('CART', DecisionTreeClassifier()),  
 ('NB', GaussianNB()),  
 ('SVM', SVC(gamma='auto'))  
]  
  
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()})')  
  
plt.boxplot(results, labels=names)  
plt.title('Алгоритми порівняння')  
plt.show()  
  
model = SVC(gamma='auto')  
model.fit(X\_train, Y\_train)  
predictions = model.predict(X\_validation)  
  
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}")









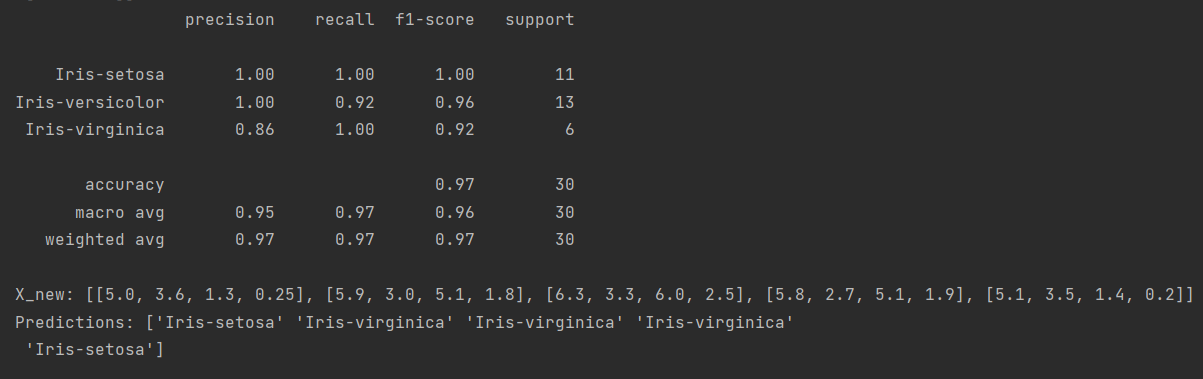


Рис.2.2. task-2\_5.py

[**https://github.com/avrorilka/AI\_Python**](https://github.com/avrorilka/AI_Python)

**Висновки:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python ми дослідити попередню обробку та класифікацію даних.

Класифікацію та кінцевий аналіз даних ми проводили на прикладах класифікатору у вигляді машини опорних векторів, призначений для прогнозування меж доходу заданої фізичної особи на основі 14 атрибутів.

Після чого провели перехресну перевірку, розбивши дані на навчальний та тестовий набори у пропорції 80/20. А для порівняння якостей класифікаторів SVM з нелінійними ядрами ми порівнювали між собою поліномінальне, гаусове та сигмоїдальне ядра.