**ЛАБОРАТОРНА РОБОТА № 2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЯ ДАНИХ**

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

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

age: continuous.  
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.  
fnlwgt: continuous.  
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.  
education-num: continuous.  
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.  
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.  
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.  
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.  
sex: Female, Male.  
capital-gain: continuous.  
capital-loss: continuous.

hours-per-week: continuous.  
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran,

Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

classifier = OneVsOneClassifier(LinearSVC(random\_state=0))

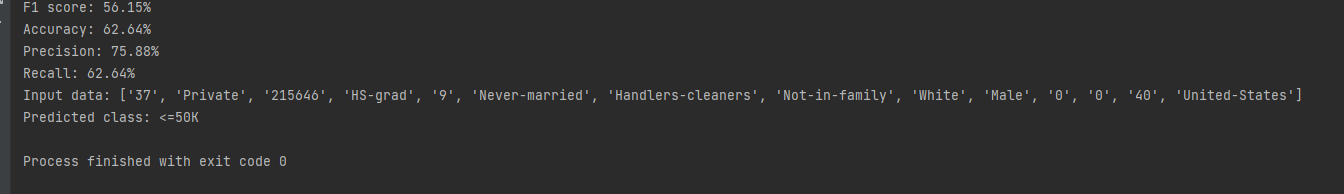


Рис.1 Результат аналізу акуратності, повноти, точності та F1

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

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=2))

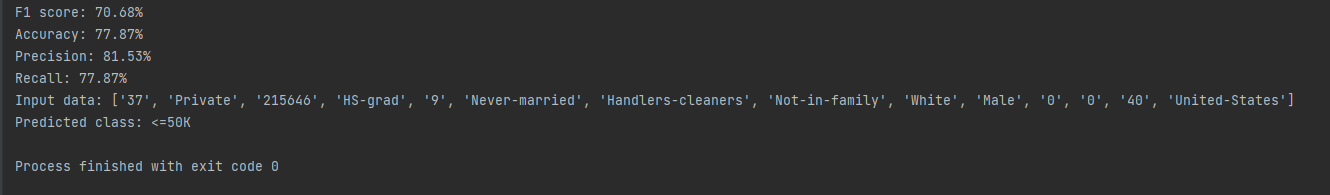


Рис.2 Результат аналізу за використання поліноміального ядра

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

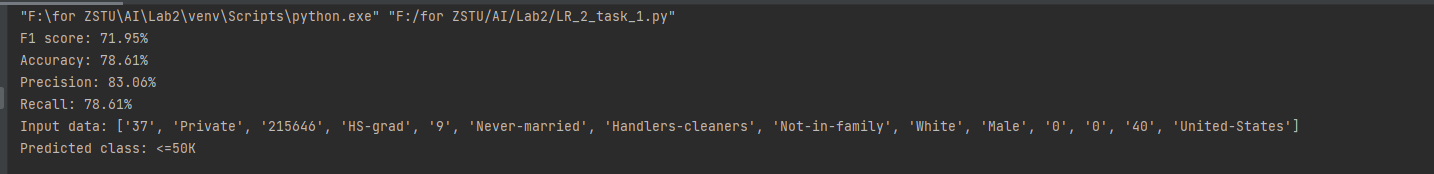


Рис.3 Результат аналізу за використання ядра Гауса

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

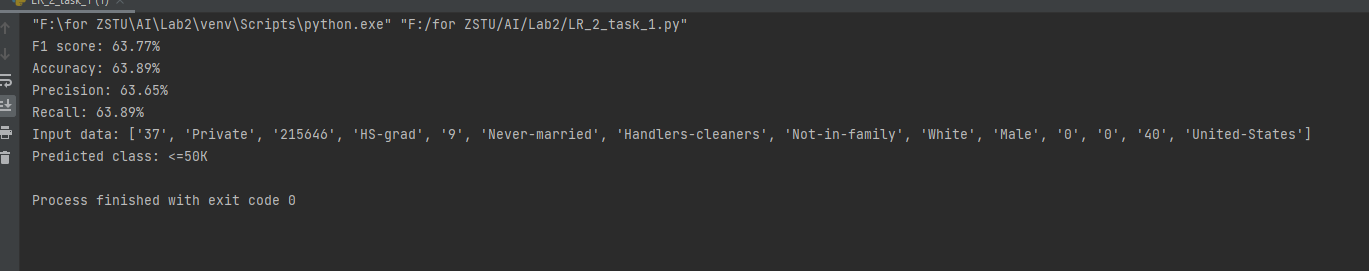
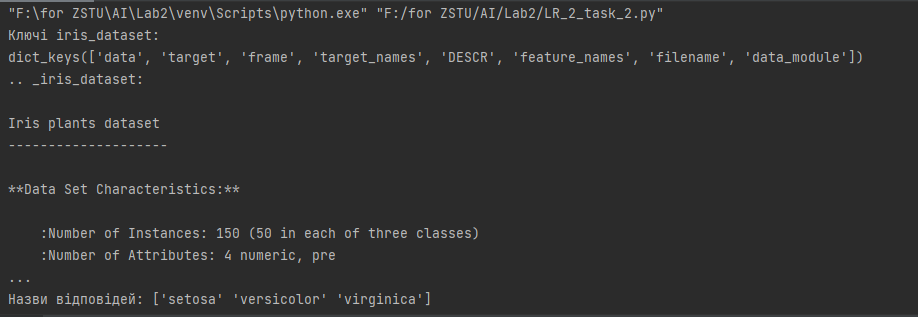


Рис.4 Результат аналізу за використання сигмоїдального ядра

За даних умов нерівномірності використання даних, найкращий результат надає ядро Гауса. За використання поліноміального ядра з максимальним значенням degree його ефективність може бути значно вищою, але це буде потребувати неймовірно потужний комп’ютер

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

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 масиву data: {type(iris\_dataset['data'])}")  
print(f"Форма масиву data: {iris\_dataset['data'].shape}")  
print(f"Перші 5 рядків ознак: \n{iris\_dataset['data'][:5]}")  
  
print(f"Тип масиву відповідей: {type(iris\_dataset['target'])}")  
print(f"Форма масиву відповідей: {iris\_dataset['target'].shape}")  
print(f"Відповіді: {iris\_dataset['target']}")



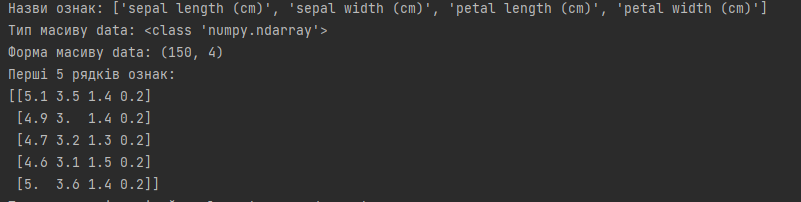


Рис.5 - 6 Виведення інформації про дані

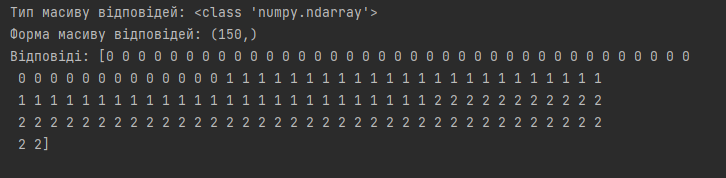


Рис.7 Виведення інформації про відповіді

# 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()

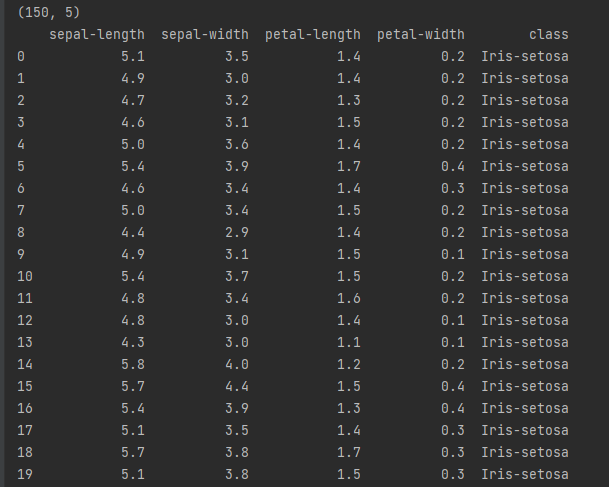


Рис.8 Розмір масиву даних та перші 20 записів

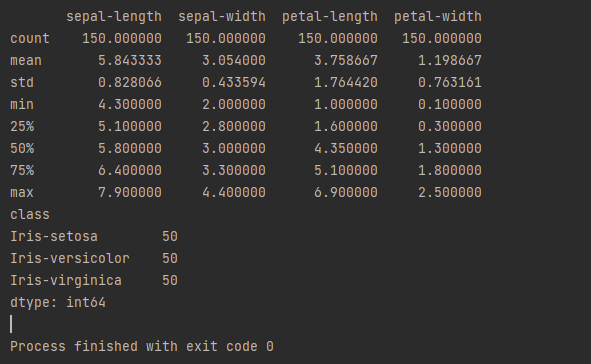


Рис.9 Характеристики даних, кількість за класами та тип даних

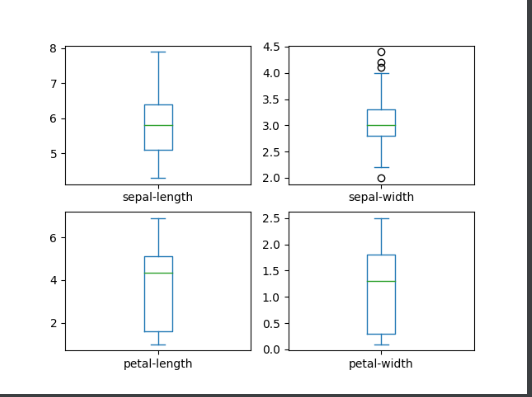


Рис. 10 Діаграма розмаху

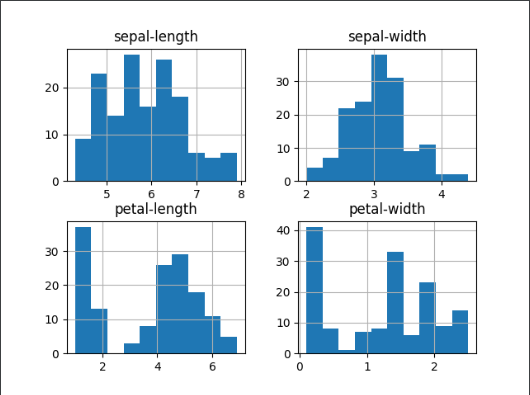


Рис.11 Гістограма розподілу атрибутів датасета

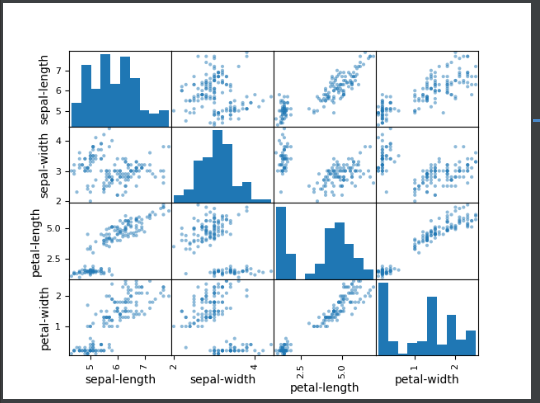


Рис.12 Матриця діаграм розсіювання

Квітка належала до класу Iris-setosa

Можна зробити висновок, що найкраще показала себе модель лінійного дискримінантного аналізу, проте вона потребувала найбільшої кількості ресурсів.

# 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 навчальну та контрольну вибірки  
X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X, Y, test\_size=0.20, random\_state=1)  
  
# Завантажуємо алгоритми моделі  
models = [('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')),  
 ('LDA', LinearDiscriminantAnalysis()),  
 ('KNN', KNeighborsClassifier()),  
 ('CART', DecisionTreeClassifier()),  
 ('NB', GaussianNB()),  
 ('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('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(accuracy\_score(Y\_validation, predictions))  
print(confusion\_matrix(Y\_validation, predictions))  
print(classification\_report(Y\_validation, predictions))

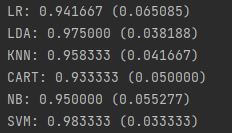


Рис. 13 Порівняння accuracy моделей

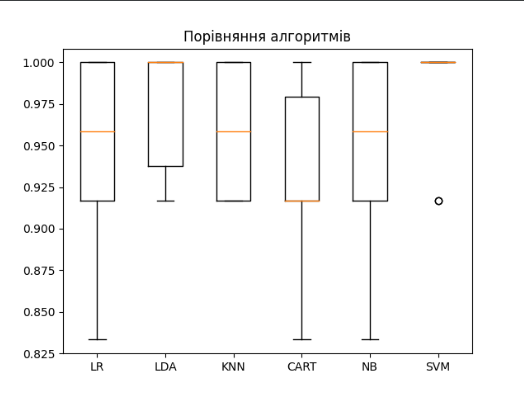


Рис.14 Діаграма розмаху атрибутів вхідних даних

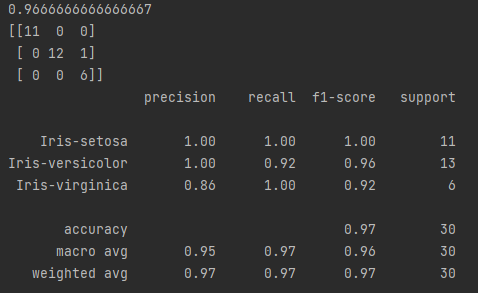


Рис.15 Якість, матриця помилок та звіт по класифікації даних через SVC

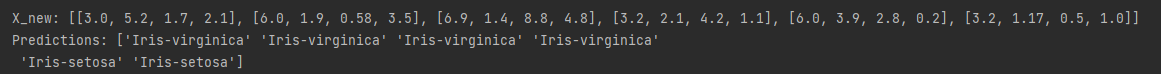


Рис. 16 Прогнозування класів власних даних

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

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")

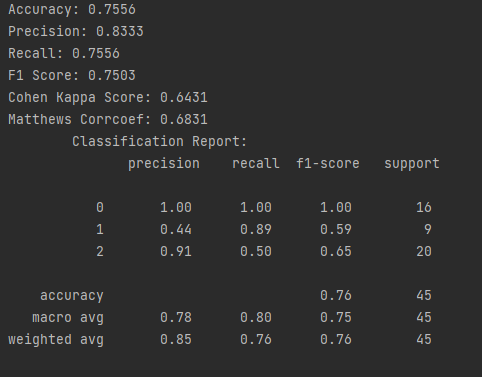


Рис.17 Результат виконання завдання 2.5

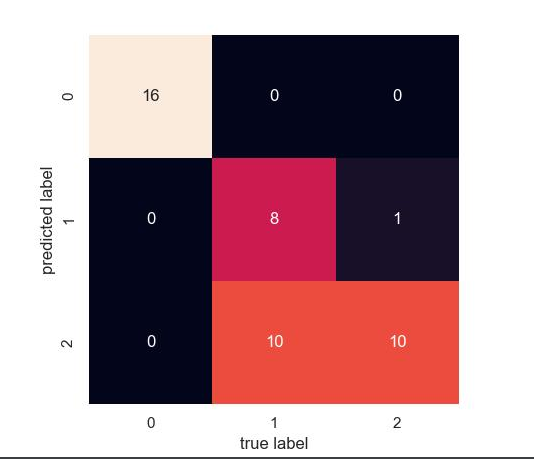


Рис.18 Confusion.jpg

В результаті було отримано:

r1, recall, коефіцієнт Коена Каппа —  статистика що використовується для вимірювання надійності між оцінювачами (а також Надійність внутрішньої оцінки ) для якісних (категоріальних) предметів, коефіцієнт кореляції Метьюса – або коефіцієнт фі використовується в машинному навчанні як міра якості двійкової (двокласної) класифікації, матриця невідповідності – це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання; кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу (або навпаки), назва походить від того факту, що вона дає можливість просто бачити, чи допускає система невідповідності між цими двома класами (наприклад, часто помилково маркуючи один як інший).

**Репозиторій:**<https://github.com/ipz192dyad/Artificial-intelligence-systems>

**Висновок:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та отримано навички для їх порівняння.