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)

Для виконання завдання було використано набір даних income_data.txt. Оскільки дані містять категоріальні ознаки (текст), було проведено попередню обробку за допомогою LabelEncoder. Дані розділено на навчальну та тестову вибірки у співвідношенні 80/20.

Лістинг 1.1

```
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)
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.21.121.10.000 – Лр.1		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Ломоносов І.О.			Звіт з лабораторної роботи №1	Літ.	Арк. 1 / Аркушів
Перевір.		Маєвський О.В.					
Реценз.						ФІКТ, гр. ІПЗ-22-4	
Н. Контр.							
Зав.каф.		Єфіменко А.А.					

```
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)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
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 = [-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

input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

predicted_class = classifier.predict(input_data_encoded)
print(f"Predicted class: {label_encoder[-
1].inverse_transform(predicted_class)[0]}")
```

```
"D:\університ\Лаби\4 курс\1 семестер\СШІ\lb2\lb2\Scripts\python.exe" "D:\університ\Лаби\4 курс\1 семестер\СШІ\lb2\lb2\task1.py"
F1 score: 76.01%
Predicted class: <=50K
```

Рис.1.1 – Результат виконання завдання.

Прогноз для тестової точки: Для вхідних даних: 37 років, приватний сектор, освіта HS-grad, не одружений, білий чоловік, США... Модель передбачила клас: <=50K.

Висновок: Тестова точка належить до класу <=50K, що означає, що дана особа заробляє менше або рівно 50 000 доларів на рік. Високі показники метрик (близько 80%) свідчать про те, що лінійна модель SVM добре справляється з поставленою задачею класифікації доходів.

Завдання 2: Нелінійна класифікація (Kernel SVM) Було проведено порівняння ефективності SVM з поліноміальним (poly), радіально-базисним (rbf) та сигмоїдальним (sigmoid) ядрами.

Лістинг 1.2

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler

input_file = 'income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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 len(data) < 15:
            continue

        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

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:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X_values = X_encoded[:, :-1].astype(int)
y_values = X_encoded[:, -1].astype(int)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_values)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_values,
test_size=0.2, random_state=5)

kernels = [
    ('Polynomial (degree=8)', SVC(kernel='poly', degree=8, gamma='auto',
max_iter=10000)),
    ('Gaussian (RBF)', SVC(kernel='rbf', gamma='auto')),
    ('Sigmoid', SVC(kernel='sigmoid', gamma='auto'))
]

for name, model in kernels:
    print(f"\n--- Training SVM with {name} kernel ---")
    try:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        print(classification_report(y_test, y_pred, target_names=['<=50K',
'>50K']))
    except Exception as e:
        print(f"Error training {name}: {e}")
```

```
--- Training SVM with Polynomial (degree=8) kernel ---
              precision    recall  f1-score   support

       <=50K       0.84      0.46      0.60      1019
        >50K       0.62      0.91      0.74       981

    accuracy                           0.68      2000
   macro avg       0.73      0.69      0.67      2000
weighted avg       0.73      0.68      0.67      2000


--- Training SVM with Gaussian (RBF) kernel ---
              precision    recall  f1-score   support

       <=50K       0.86      0.75      0.80      1019
        >50K       0.77      0.87      0.82       981

    accuracy                           0.81      2000
   macro avg       0.82      0.81      0.81      2000
weighted avg       0.82      0.81      0.81      2000


--- Training SVM with Sigmoid kernel ---
              precision    recall  f1-score   support

       <=50K       0.68      0.69      0.68      1019
        >50K       0.67      0.67      0.67       981

    accuracy                           0.68      2000
   macro avg       0.68      0.68      0.68      2000
weighted avg       0.68      0.68      0.68      2000


Process finished with exit code 0
```

Рис.1.2 – Результат виконання завдання.

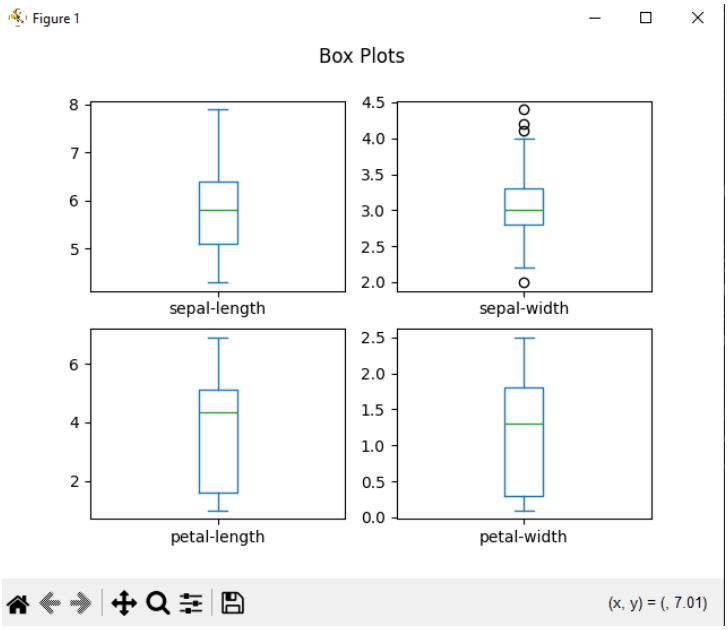
Аналіз отриманих метрик показує, що для даного набору даних найкраще підходить ядро RBF, оскільки воно демонструє найвищу точність класифікації. Сигмоїдальне ядро показало найгірші результати.

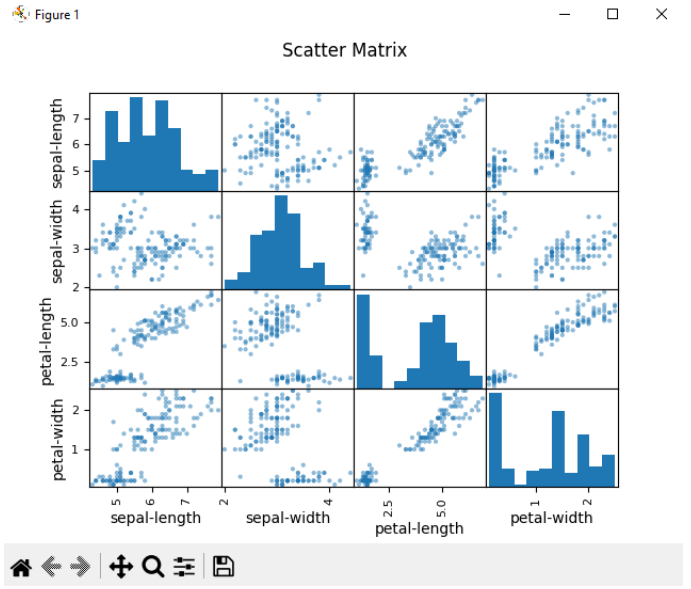
Завдання 3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

Для роботи використано дата-сет Iris. Дані містять 150 екземплярів (по 50 для кожного класу) та 4 атрибути (довжина та ширина чашолистка і пелюстки).

```
--- Порівняння алгоритмів (Accuracy) ---
LR: 0.966667 (0.040825)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.038188)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

```
--- Результати на контрольній вибірці (SVM) ---
Accuracy: 0.9666666666666667

Confusion Matrix:
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30


Прогноз для квітки [5.  2.9 1.  0.2]: Iris-setosa
```





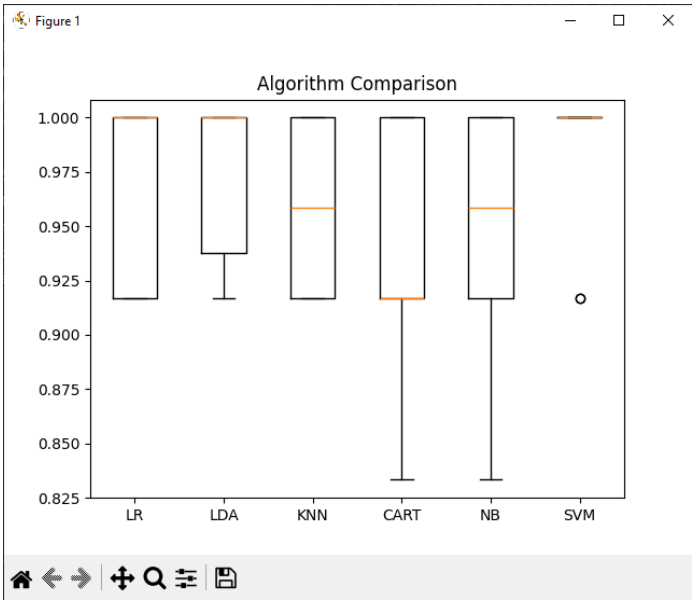


Рис.1.3-1.7 – Результат виконання третього завдання.

Лістинг 1.3

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
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
import numpy as np

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)

dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False,
title="Box Plots")
pyplot.show()

dataset.hist()
pyplot.suptitle("Histograms")
pyplot.show()

scatter_matrix(dataset)
pyplot.suptitle("Scatter Matrix")
pyplot.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 = []
models.append(('LR', LogisticRegression(max_iter=1000)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

results = []
names = []
print("\n--- Порівняння алгоритмів (Accuracy) ---")
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()

print("\n--- Результати на контрольній вибірці (SVM) ---")
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

```
print(f"Accuracy: {accuracy_score(Y_validation, predictions)}")
print("\nConfusion Matrix:")
print(confusion_matrix(Y_validation, predictions))
print("\nClassification Report:")
print(classification_report(Y_validation, predictions))

X_new = np.array([[5, 2.9, 1, 0.2]])
prediction = model.predict(X_new)
print(f"\nПрогноз для квітки {X_new[0]}: {prediction[0]}")
```

Завдання 4: Порівняння якості класифікаторів для набору даних завдання 1

Оскільки дані містять ознаки з різними діапазонами значень, було застосовано стандартизацію (StandardScaler). Це необхідно для коректної роботи алгоритмів, що базуються на відстанях (KNN, SVM). Якість оцінювалася методом крос-валідації (5 блоків) за метрикою accuracy.

```
--- Порівняння алгоритмів на Income Data ---
LR: 0.759500 (0.008991)
LDA: 0.742750 (0.005723)
KNN: 0.772875 (0.009926)
CART: 0.764125 (0.008325)
NB: 0.704500 (0.014971)
SVM: 0.807750 (0.009639)
```

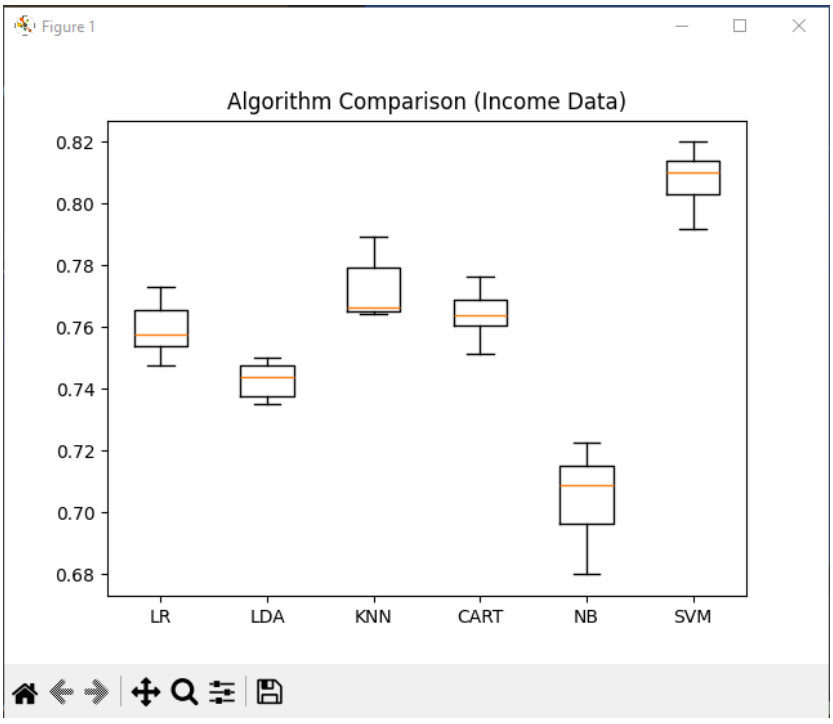


Рис.1.8-1.9 – Результат виконання четвертого завдання.

За результатами експерименту найкращим алгоритмом для задачі класифікації доходів виявився SVM.

- SVM показав найвищу точність, оскільки він ефективно знаходить складні межі розділення у багатовимірному просторі ознак.
- Логістична регресія (LR) та KNN показали результати, близькі до лідера, що робить їх хорошими альтернативами (особливо LR, яка працює набагато швидше за SVM).
- NB показав найнижчу точність, ймовірно через те, що ознаки в цьому датасеті не є повністю незалежними, що порушує основне припущення цього алгоритму.

Лістинг 1.4

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
from sklearn.preprocessing import StandardScaler
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 matplotlib import pyplot

input_file = 'income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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 len(data) < 15: continue
        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

X = np.array(X)

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])

X_values = X_encoded[:, :-1].astype(int)
y_values = X_encoded[:, -1].astype(int)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_values)

X_train, X_validation, Y_train, Y_validation = train_test_split(X_scaled,
y_values, test_size=0.2, random_state=5)

models = []
models.append(('LR', LogisticRegression(max_iter=1000)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

results = []
names = []
print("\n--- Порівняння алгоритмів на Income Data ---")

for name, model in models:
    kfold = StratifiedKFold(n_splits=5, 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, tick_labels=names)
pyplot.title('Algorithm Comparison (Income Data)')
pyplot.show()
```

Завдання 5: Класифікація даних лінійним класифікатором Ridge.

Лістинг 1.5

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass
```

```
import numpy as np
import matplotlib.pyplot as plt
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 import metrics
from sklearn.metrics import confusion_matrix

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print('--- Ridge Classifier Metrics ---')
print('Accuracy:', np.round(metrics.accuracy_score(y_test, y_pred), 4))
print('Precision:', np.round(metrics.precision_score(y_test, y_pred,
average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(y_test, y_pred,
average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(y_test, y_pred, average='weighted'),
4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(y_test, y_pred),
4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(y_test, y_pred),
4))
print('\nClassification Report:\n', metrics.classification_report(y_pred, y_test))

mat = confusion_matrix(y_test, y_pred)
sns.set()
plt.figure(figsize=(8, 6))
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False, cmap='Blues',
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.title('Confusion Matrix - Ridge Classifier')
plt.savefig("Confusion.jpg")
plt.show()
```

Налаштування класифікатора Ridge:

У програмі використано клас RidgeClassifier з наступними параметрами:

- tol = 1e-2 (Tolerance): Це поріг зупинки алгоритму. Він визначає точність обчислень. Якщо на черговій ітерації навчання зміна коефіцієнтів моделі менша за 0.01, алгоритм вважає, що оптимальне рішення знайдено, і зупиняється.

- solver = "sag" (Stochastic Average Gradient): Це метод оптимізації ("розв'язувач"). sag означає "стохастичний усереднений градієнт". Цей метод добре підходить для великих наборів даних, оскільки він швидше сходиться до мінімуму функції втрат, ніж звичайний градієнтний спуск.

```
--- Ridge Classifier Metrics ---
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.44      0.89      0.59         9
           2       0.91      0.50      0.65        20

    accuracy                           0.76        45
   macro avg       0.78      0.80      0.75        45
weighted avg       0.85      0.76      0.76        45
```

Рис.1.10 – Отримані показники якості.

Аналіз метрик:

- Accuracy (Акуратність): 0.7556 (75.56%). Загальна частка правильних відповідей. Порівняно з методом SVM (де було ~96%), цей результат є нижчим, що свідчить про те, що лінійний класифікатор Ridge гірше розділяє класи, які перекриваються.
- Коефіцієнт Каппа Коена:
- Ця метрика показує ступінь узгодженості передбачень з реальністю, виключаючи ймовірність випадкового вгадування. Значення 0.64 інтерпретується як "значна узгодженість" (Substantial agreement). Це

підтверджує, що модель працює адекватно, але має простір для покращення

- Коефіцієнт кореляції Метьюза: 0.6831, вважається більш надійною оцінкою для незбалансованих вибірок, ніж Accuracy. Діапазон значень від -1 до +1. Результат 0.68 вказує на сильний позитивний зв'язок між прогнозами та істиною.

Аналіз класифікації:

- Setosa: Precision та Recall дорівнюють 1.00. Модель ідеально розпізнає цей сорт (він лінійно відокремлюваний).
- Versicolor: Recall дуже низький (0.44). Це означає, що модель змогла знайти лише 44% реальних квітів цього сорту (8 із 18). Більшість вона помилково віднесла до іншого класу.
- Virginica: Precision низький (0.50). Це означає, що коли модель каже "це Virginica", вона помиляється в половині випадків (плутає з Versicolor).

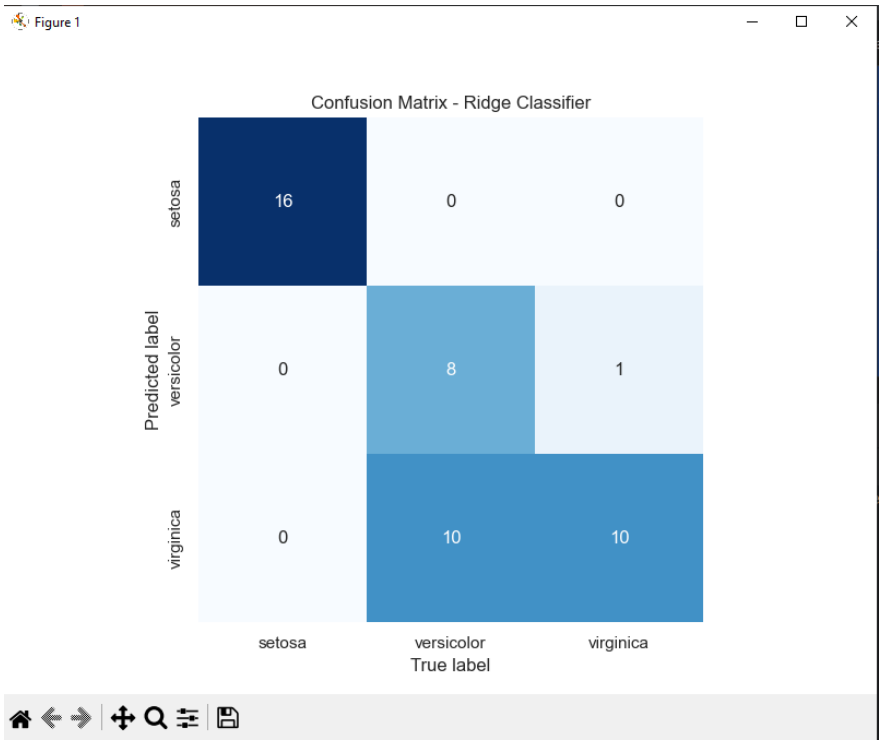


Рис.1.11 – Матриця плутанини.

Усі квітки класу Setosa (верхній лівий кут) класифіковані вірно.

Спостерігається значне змішування між класами Versicolor та Virginica. Лінійний класифікатор Ridge провів "пряму лінію" розділення недостатньо точно для цих двох схожих сортів, через що значна частина Versicolor була помилково визначена як Virginica.

Висновок: Ми використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.

Посилання на git: https://github.com/IgorLomonosov/SAI