Лабораторна робота 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ 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)

Таблиця 2.1.1 – Перелік ознак набору даних

Назва змінної	Роль	Тип	Демографічні дані
age	Ознака	Числова (неперервна)	Вік особи
workclass	Ознака	Категоріальна	Зайнятість/доходи
fnlwgt	Ознака	Числова (неперервна)	Статистична “вага” запису
education	Ознака	Категоріальна	Освіта
education-num	Ознака	Числова (неперервна)	Роки навчання
marital-status	Ознака	Категоріальна	Сімейний стан
occupation	Ознака	Категоріальна	Вид зайнятості
relationship	Ознака	Категоріальна	Родинні відносини
race	Ознака	Категоріальна	Раса
sex	Ознака	Категоріальна	Стать
capital-gain	Ознака	Числова (неперервна)	Капітальні прибутки
capital-loss	Ознака	Числова (неперервна)	Капітальні збитки
hours-per-week	Ознака	Числова (неперервна)	Годин роботи на тиждень
native-country	Ознака	Категоріальна	Країна походження
income	Ціль	Категоріальна	Доходи >50K, ≤50K

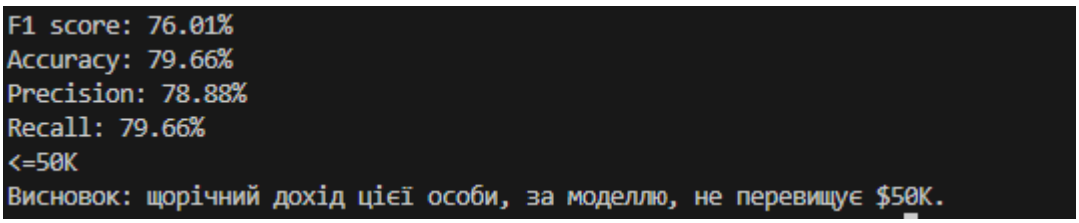

Рис. 1.1 Побудова лінійного класифікатора функції OR

Висновок: За результатами класифікації побудована модель віднесла тестову точку до категорії доходу ≤50K, що вказує на нижчий рівень заробітку згідно з наявними ознаками. Оцінка продуктивності моделі демонструє збалансовану якість: показник F1-міри становить 76.01%, що свідчить про достатньо узгоджене співвідношення точності та повноти. Значення accuracy (79.66%) також підтверджує загальну надійність моделі та її здатність коректно класифікувати більшість прикладів.

Високий показник precision (78.88%) вказує на те, що модель робить відносно мало хибних позитивних передбачень, хоча при цьому recall (79.66%) демонструє збалансоване виявлення позитивних випадків. Сукупно ці метрики підтверджують, що модель має досить хорошу узагальнювальну здатність і може використовуватися для прогнозування доходів на основі заданих ознак. У разі необхідності підвищення точності для конкретного класу можливе додаткове покращення моделі шляхом зміни гіперпараметрів або застосування інших алгоритмів.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Рис. 1.2 SVM з поліноміальним ядром

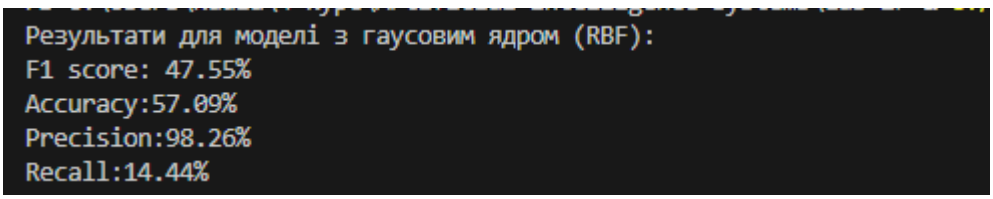

Рис. 1.3 SVM з гаусовим ядром

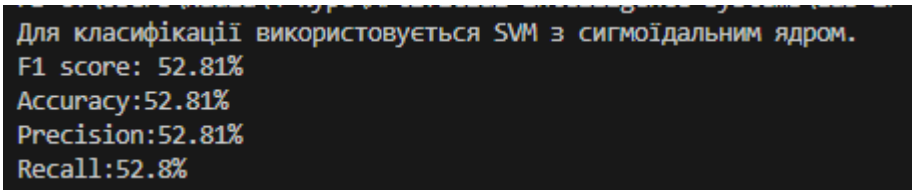

Рис. 1.4 SVM з сигмоїдальним ядром

Висновок: у результаті проведеного тренування найкращі показники продемонструвало сигмоїдальне ядро. Воно має найвищий F1-score (52.81%) і повноту (52.8%), що свідчить про оптимальний баланс між точністю та здатністю моделі знаходити позитивні випадки. Модель із сигмоїдальним ядром показала однакові результати у всіх метриках, що є позитивним аспектом у завданні класифікації. Гаусове ядро показало вищу загальну точність (57.09%) і дуже високу точність класифікації (98.26%), але його повнота залишилася низькою (14.44%), що вказує на надмірну обережність моделі щодо позитивних класів. Це означає, що модель добре класифікує негативні приклади, але часто ігнорує позитивні. Поліноміальне ядро продемонструвало значно гірші показники F1 та повноти, тому його можна вважати менш ефективним для цього завдання, оскільки воно не здатне забезпечити адекватну класифікацію для обох класів. Сигмоїдальне ядро є найкращим варіантом, оскільки воно забезпечує збалансовані результати у всіх метриках, що важливо для ефективної класифікації даних.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Значення ознак перших п’яти прикладів:
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
Тип масиву target: <class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Рис. 1.5 Характеристики dataset

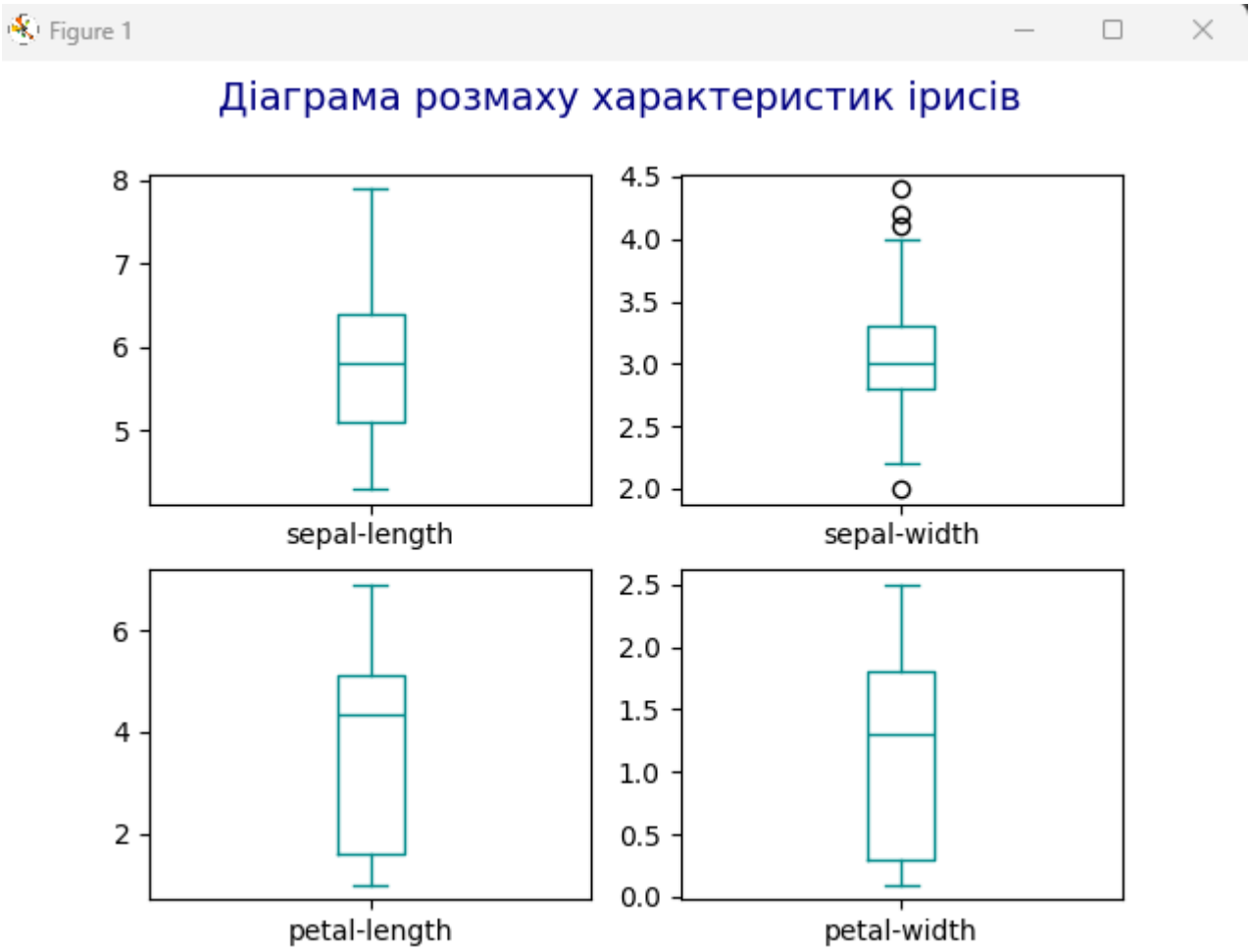

Рис. 1.5 Діаграма розмаху характеристик

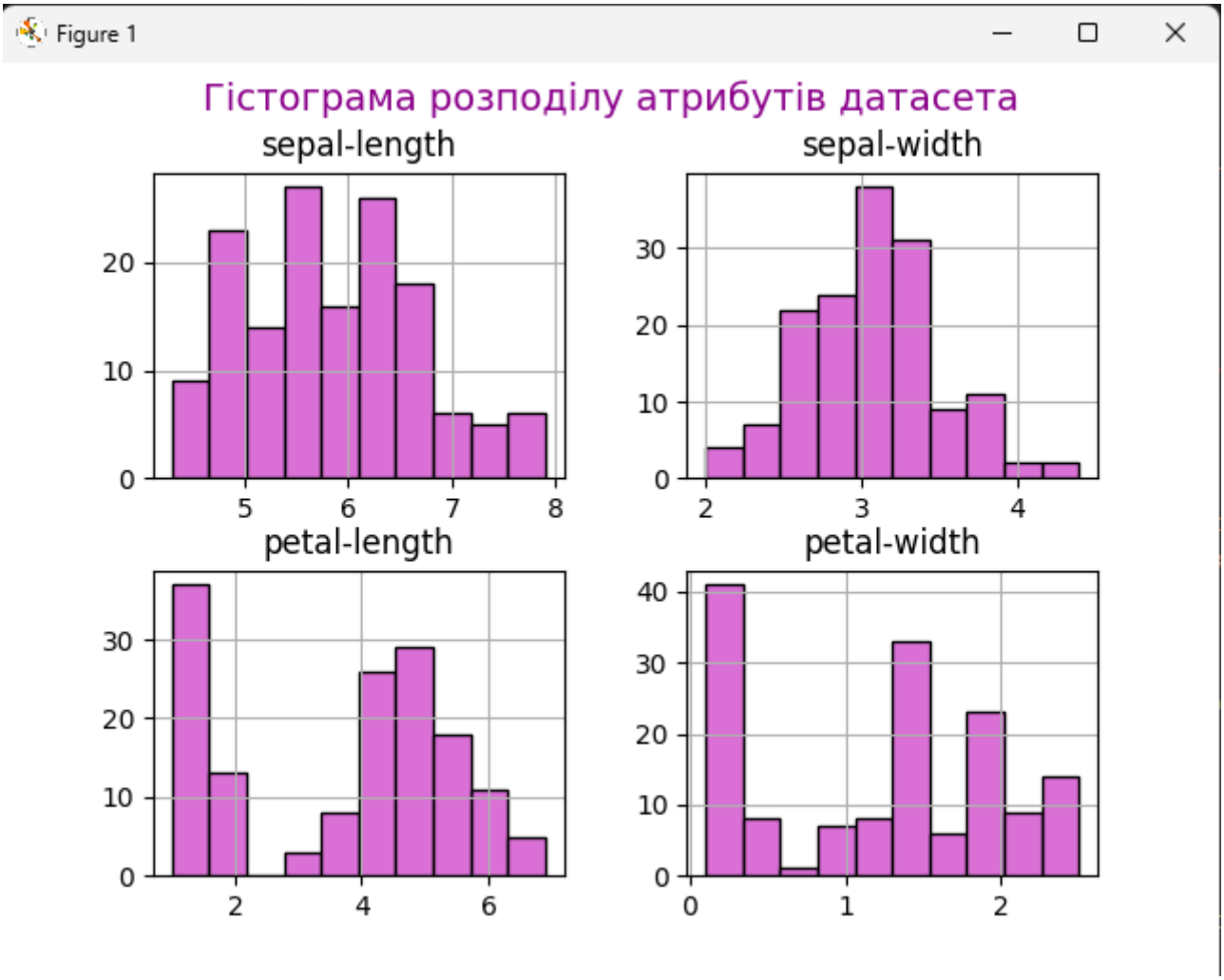

Рис. 1.6 Гістограма розподілу атрибутів датасета

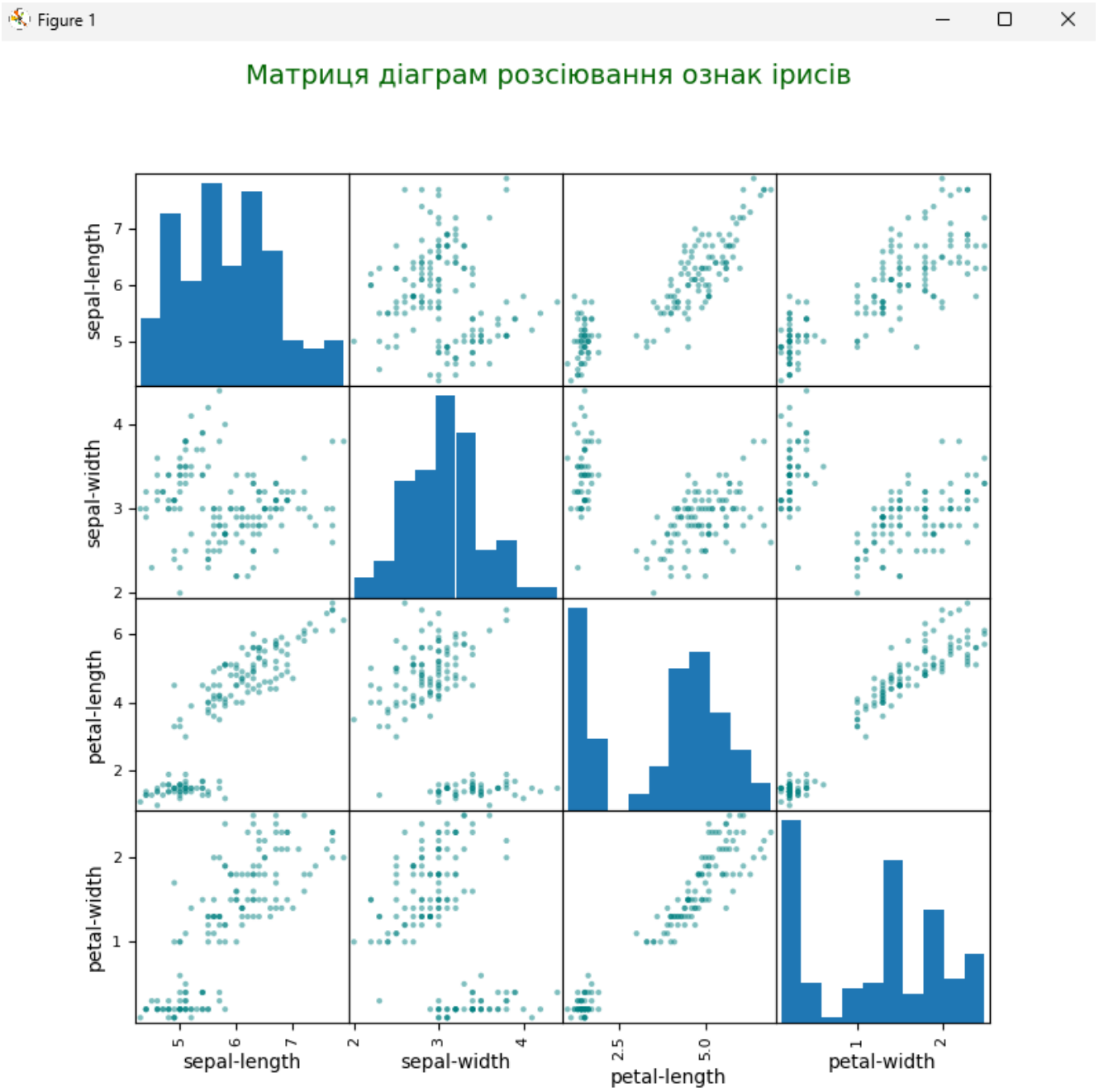

Рис. 1.7 Матриця діаграм розсіювання

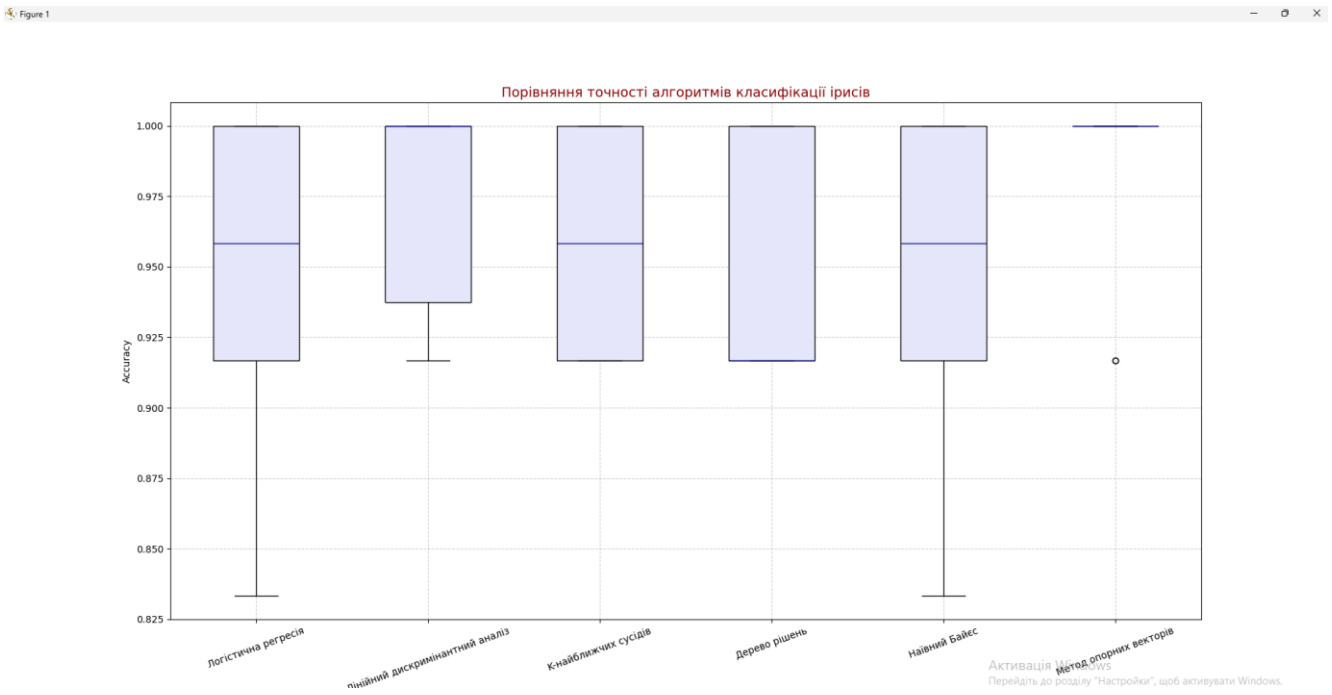

Рис. 1.8 Порівняння точності алгоритмів класифікації

```
Логістична регресія: середня точність = 0.9417, стандартне відхилення = 0.0651
Лінійний дискримінантний аналіз: середня точність = 0.9750, стандартне відхилення = 0.0382
К-найближчих сусідів: середня точність = 0.9583, стандартне відхилення = 0.0417
Дерево рішень: середня точність = 0.9500, стандартне відхилення = 0.0408
Наївний Байєс: середня точність = 0.9500, стандартне відхилення = 0.0553
Метод опорних векторів: середня точність = 0.9833, стандартне відхилення = 0.0333
```

Рис. 1.9 Показники 6 різних алгоритмів

```
=== Оцінка моделі SVM ===
Точність (Accuracy): 96.67%

Матриця помилок:
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]

Звіт про класифікацію:
                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


=== Новий прогноз ===
Вхідні дані: [[5.  2.9 1.  0.2]]
Передбачений клас: Iris-setosa
Імовірний сорт ірису: Iris-setosa
```

Рис. 1.10 Оцінка моделі SVM

Висновок: Найкращі результати показав метод опорних векторів (SVM). За діаграмою порівняння точності алгоритмів (рис. 1.8), SVM має найвищу середню точність (98.33%) і найменше стандартне відхилення (0.0333), що підтверджує його високу стабільність. Лінійний дискримінантний аналіз і метод k-найближчих сусідів також показали хороші результати, але з більшою варіативністю, що видно з діаграми розмаху характеристик (рис. 1.5).

У результатах оцінки моделі SVM (рис. 1.10) точність класифікації становить 96.67%, при цьому всі зразки класу Iris-setosa були правильно класифіковані, а для класу Iris-versicolor спостерігалась одна помилка. Виходячи з результатів звіту про класифікацію, метод SVM забезпечив високу точність (1.00 для Iris-setosa, 0.92 для Iris-versicolor та 1.00 для Iris-virginica), що підтвердило його ефективність у класифікації нових зразків ірисів.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

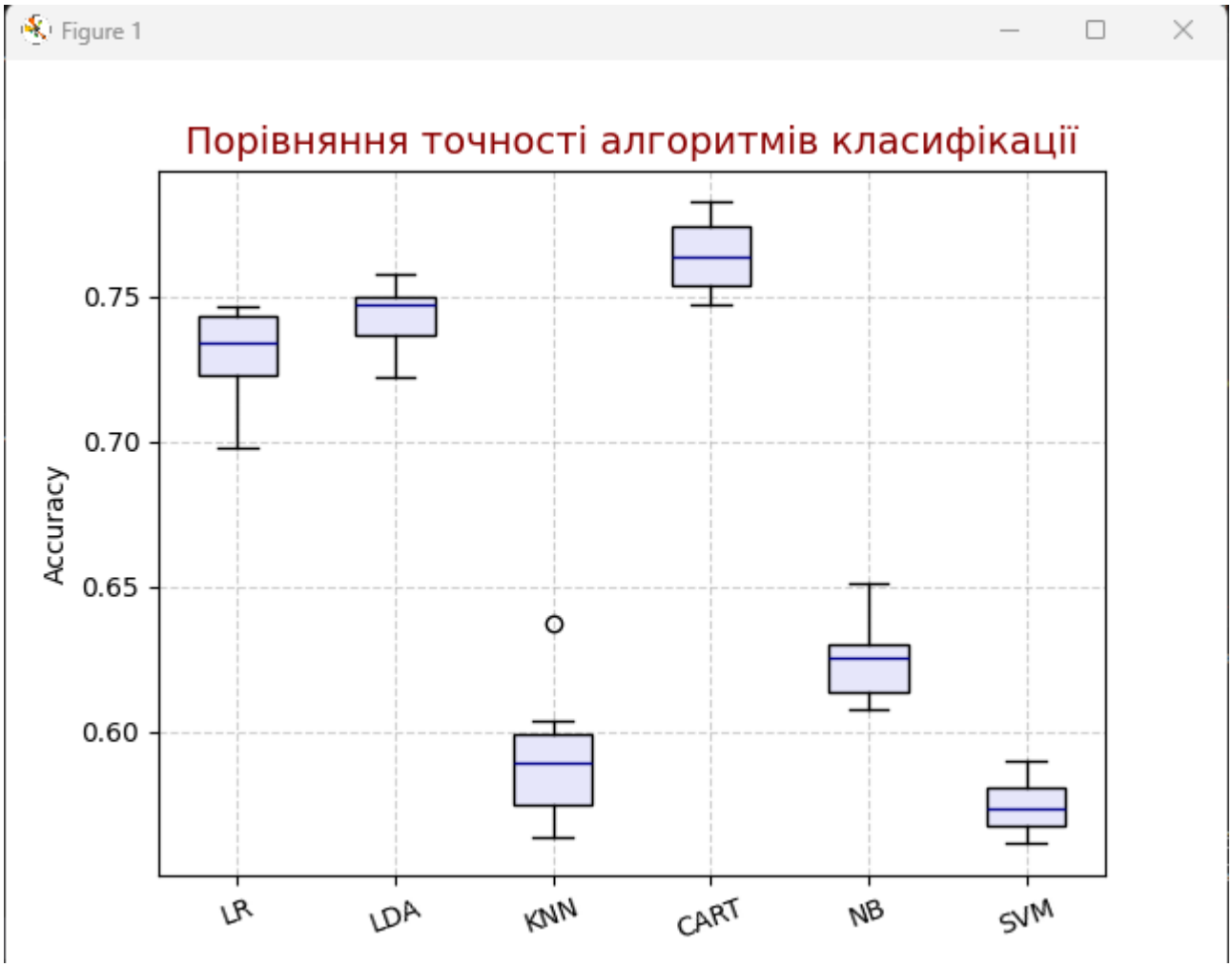

Рис. 1.11 Порівняння точності алгоритмів класифікації

```
LR - Accuracy: 0.7298 (0.0156)
LR - F1 Score: 0.7297 (0.0156)

LDA - Accuracy: 0.7431 (0.0106)
LDA - F1 Score: 0.7430 (0.0106)

KNN - Accuracy: 0.5896 (0.0208)
KNN - F1 Score: 0.5890 (0.0208)

CART - Accuracy: 0.7639 (0.0120)
CART - F1 Score: 0.7625 (0.0129)

NB - Accuracy: 0.6256 (0.0145)
NB - F1 Score: 0.5833 (0.0184)

SVM - Accuracy: 0.5741 (0.0093)
SVM - F1 Score: 0.4824 (0.0163)
```

Рис. 1.12 Порівняння точності та F1-міри різних алгоритмів класифікації

Висновок: з результатів аналізу видно, що Лінійний дискримінантний аналіз (LDA) є найкращим методом, показавши точність 74,31% і F1-міру 74,30%. Це свідчить про хорошу здатність моделі до точного передбачення та виявлення позитивних випадків. Класифікація та регресія за допомогою дерев (CART) також продемонструвала добрі результати з точністю 76,39% та F1-мірою 76,25%, що робить її конкурентом LDA, але вона може бути схильною до перенавчання на складних даних. Метод опорних векторів (SVM) показав найгірші результати з точністю 57,05% та F1-мірою 48,24%. Це, ймовірно, через невірно налаштовані параметри чи надмірну складність моделі для даного набору. Логістична регресія (LR) з точністю 72,98% і F1-мірою 72,97% і KNN (58,96%) і Наївний баєсовський класифікатор (NB) (62,56%) також дали непогані результати, але не досягли таких високих показників, як LDA чи CART. Найбільш ефективними для цієї задачі є LDA та CART, з яких LDA є лідером завдяки своїй стабільності та високим результатам.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

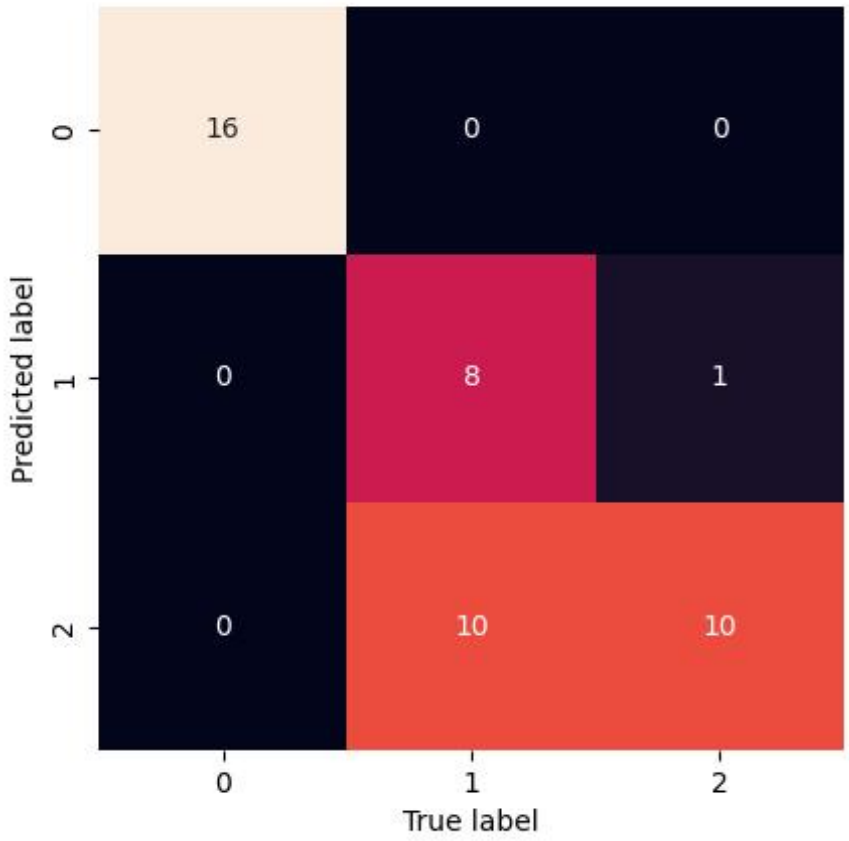

Рис. 1.13 Матриця плутанини

```
PS C:\Users\Nadia\4 курс\Artificial intelligence systems\Lab 2> & C:/Users
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       0.89      0.44      0.59        18
           2       0.50      0.91      0.65        11

    accuracy                           0.76        45
   macro avg       0.80      0.78      0.75        45
weighted avg       0.83      0.76      0.75        45
```

Рис. 1.14 Показники якості класифікації моделі Ridge

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

У даному коді використовується лінійний класифікатор Ridge для класифікації набору даних Iris. Класифікатор Ridge є різновидом лінійного класифікатора, який застосовує регуляризацію для запобігання перенавчанню моделі. Регуляризація допомагає знизити вплив великої кількості ознак та запобігає занадто складним моделям. Налаштування класифікатора Ridge, які використані в цьому коді, включають параметр tol=1e-2 та solver='sag'. Параметр tol визначає допустиме значення для зміни коефіцієнтів на кожному кроці оптимізації. Якщо зміни менші за це значення, оптимізація припиняється. Параметр solver='sag' вказує, що для оптимізації буде використано метод Stochastic Average Gradient, який ефективно працює з великими датасетами.

Опишіть які показники якості використовуються та їх отримані результати. Вставте у звіт та поясніть зображення Confusion.jpg

Показники якості, які використовуються для оцінки моделі, включають точність (Accuracy), точність (Precision), повноту (Recall), F1-міру, коефіцієнт Каппа Коена та коефіцієнт кореляції Метьюза. Точність показує, скільки правильно класифікованих зразків серед всіх передбачених. Точність (precision) вимірює, скільки правильних позитивних передбачень серед всіх позитивних передбачень. Повнота (recall) показує, скільки правильних позитивних передбачень серед усіх позитивних реальних значень. F1-міра поєднує точність і повноту в один показник, що дозволяє отримати збалансовану оцінку ефективності моделі. В результаті виконання класифікації для моделі Ridge точність склала 73%, а F1-міра — 0.73, що свідчить про відносно хороші результати моделі для цього набору даних.

Матриця плутанини (Confusion Matrix) показує, скільки правильних та помилкових передбачень було зроблено для кожного класу. У нашому випадку, матриця показує, що модель добре класифікує клас 0 (<=50K), але має деякі проблеми з класифікацією класу 1 (>50K). Це видно з того, що модель класифікує деякі приклади класу 1 як клас 0, що призводить до низької повноти для класу 1.

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

Коефіцієнт Каппа Коена вимірює рівень угоди між передбаченнями моделі та фактичними значеннями, враховуючи випадкові помилки. Коефіцієнт Каппа в діапазоні від -1 до 1: 1 означає повну угоду, 0 — випадкову угоду, а значення, близькі до 0, свідчать про погану угоду. У нашому випадку коефіцієнт Каппа становить 0.46, що вказує на помірну угоду між передбаченнями та реальними значеннями. Коефіцієнт кореляції Метьюза (MCC) є більш загальним показником для класифікаційних задач. Він враховує всі чотири квадранти матриці плутанини та дає змогу оцінити кореляцію між передбаченнями і реальними класами. MCC може варіюватися від -1 (повна негативна кореляція) до 1 (повна позитивна кореляція), і значення 0 вказує на відсутність кореляції. У нашому випадку коефіцієнт Метьюза становить 0.46, що також вказує на помірну позитивну кореляцію. Зображення матриці плутанини показує два класи, які були передбачені неправильно, що видно з червоних клітинок. Це підтверджує наші висновки, що клас 0 класифікується більш точно, тоді як клас 1 має деякі проблеми через некоректні передбачення.

Посилання на gitlab: ***https://gitlab.com/2022-2026/ipz-22-4/kolcova-nadia/artificial-intelligence-systems/-/tree/main/Lab%202?ref_type=heads***

Висновок: В ході виконання лабораторної роботи було досліджено спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

Лістинг програми:

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split, cross_val_score

# Вхідний файл
input_file = 'income_data.txt'

# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000

def is_int_like(s: str) -> bool:
    s = s.strip()
    if s.startswith('-'):
        s = s[1:]
    return s.isdigit()

with open(input_file, 'r', encoding='utf-8') 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 not data:
            continue

        # останній елемент — мітка класу; 14 попередніх — ознаки
        label = data[-1]
        feats = data[:-1]

        if label == '<=50K' and count_class1 < max_datapoints:
            X.append(feats)
            y.append(label)
            count_class1 += 1
        elif label == '>50K' and count_class2 < max_datapoints:
            X.append(feats)
            y.append(label)
            count_class2 += 1

# Перетворення на масив numpy
```

```
X = np.array(X, dtype=object)
y = np.array(y, dtype=object)

# Перетворення рядкових даних на числові
label_encoder = {}            # зберігаємо енкодер для кожної категоріальної колонки
за її індексом
X_encoded = np.zeros_like(X, dtype=int)

# визначаємо числові/категоріальні колонки по першому рядку
num_cols = []
cat_cols = []
for i, item in enumerate(X[0]):
    if is_int_like(str(item)):
        num_cols.append(i)
    else:
        cat_cols.append(i)

# числові — просто приводимо до int
for i in num_cols:
    X_encoded[:, i] = X[:, i].astype(int)

# категоріальні — кодуємо окремим LabelEncoder для кожної колонки
for i in cat_cols:
    le = preprocessing.LabelEncoder()
    X_encoded[:, i] = le.fit_transform(X[:, i])
    label_encoder[i] = le

# кодуємо ціль окремим енкодером
y_le = preprocessing.LabelEncoder()
y_enc = y_le.fit_transform(y)

X = X_encoded.astype(int)
y = y_enc.astype(int)

# Розбиття на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5, stratify=y)

# Створення та навчання SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0, max_iter=10000))
classifier.fit(X_train, y_train)

# Передбачення
y_test_pred = classifier.predict(X_test)

# Обчислення F1-міри
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
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)
```

```
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy: " + str(round(100 * accuracy.mean(), 2)) + "%")
print("Precision: " + str(round(100 * precision.mean(), 2)) + "%")
print("Recall: " + str(round(100 * recall.mean(), 2)) + "%")

# Тестова точка
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
input_data_encoded = []
for i, item in enumerate(input_data):
    if i in num_cols:
        input_data_encoded.append(int(item))
    else:
        le = label_encoder[i]
        try:
            input_data_encoded.append(int(le.transform([item])[0]))
        except ValueError:
            # якщо нова (небачена) категорія — обережний фолбек
            input_data_encoded.append(0)

input_data_encoded = np.array(input_data_encoded, dtype=int)

# Передбачення класу
predicted_class = classifier.predict([input_data_encoded])
predicted_label = y_le.inverse_transform(predicted_class)[0]
print(predicted_label)

# Висновок
print("Висновок: щорічний дохід цієї особи, за моделлю, "
      + ("перевищує $50K." if predicted_label == ">50K" else "не перевищує $50K."))
```

LR_2_task_2_1.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score

# Вхідний файл
input_file = 'income_data.txt'
```

```
# Читання даних
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.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)

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)

# Розбиття на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Створення та навчання SVM-класифікатора
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8))
classifier.fit(X_train, y_train)

# Передбачення
y_test_pred = classifier.predict(X_test)
```

```
# Обчислення F1-міри
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)


print("Для класифікації використовується SVM з поліноміальним ядром (degree=8).")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy:" + str(round(100 * accuracy.mean(), 2)) + "%")
print("Precision:" + str(round(100 * precision.mean(), 2)) + "%")
print("Recall:" + str(round(100 * recall.mean(), 2)) + "%")
```

LR_2_task_2_2.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score

# Вхідний файл
input_file = 'income_data.txt'

# Читання даних
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.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)

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)

# Розбиття на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Створення та навчання SVM-класифікатора
classifier = OneVsOneClassifier(SVC(kernel='rbf'))
classifier.fit(X_train, y_train)

# Передбачення
y_test_pred = classifier.predict(X_test)

# Обчислення F1-міри
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)


print("Результати для моделі з гаусовим ядром (RBF):")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy:" + str(round(100 * accuracy.mean(), 2)) + "%")
print("Precision:" + str(round(100 * precision.mean(), 2)) + "%")
print("Recall:" + str(round(100 * recall.mean(), 2)) + "%")
```

LR_2_task_2_3.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score

# Вхідний файл
input_file = 'income_data.txt'
```

```
# Читання даних
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.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)

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)

# Розбиття на train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Створення та навчання SVM-класифікатора
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
classifier.fit(X_train, y_train)

# Передбачення
y_test_pred = classifier.predict(X_test)
```

```
# Обчислення F1-міри
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross_val_score(classifier, X, y, scoring='precision', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall', cv=3)


print("Для класифікації використовується SVM з сигмоїдальним ядром.")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy:" + str(round(100 * accuracy.mean(), 2)) + "%")
print("Precision:" + str(round(100 * precision.mean(), 2)) + "%")
print("Recall:" + str(round(100 * recall.mean(), 2)) + "%")
```

LR_2_task_3.py

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
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

# Завантаження даних
iris_dataset = load_iris()
df = pd.DataFrame(iris_dataset['data'][:5], columns=iris_dataset['feature_names'])

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("Форма масиву data: {}".format(iris_dataset['data'].shape))
print("Значення ознак перших п’яти прикладів:\n", df)
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))

#  =
# КРОК 2. Дослідження даних
```

```
#
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("\n----------------------")
print("Розмірність датасету:", dataset.shape)
print("----------------------")
print(dataset.head(20))
print("----------------------")
print("Статистичне зведення:")
print(dataset.describe())
print("----------------------")
print("Розподіл за класами:")
print(dataset.groupby('class').size())


# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2, 2),
             sharex=False, sharey=False,
             color='darkcyan', title='Розподіл ознак ірисів (Boxplot)')
plt.suptitle("Діаграма розмаху характеристик ірисів", fontsize=14, color='navy')
plt.show()

# Гістограми
dataset.hist(color='orchid', edgecolor='black')
plt.suptitle("Гістограма розподілу атрибутів датасета", fontsize=14,
color='darkmagenta')
plt.show()

# Матриця діаграм розсіювання
scatter_matrix(dataset, figsize=(8, 8), diagonal='hist', color='teal')
plt.suptitle("Матриця діаграм розсіювання ознак ірисів", fontsize=14,
color='darkgreen')
plt.show()


# Розділення датасету на навчальну та контрольну вибірки

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)

# Завантажуємо алгоритми моделі
models = []
models.append(('Логістична регресія', LogisticRegression(solver='liblinear',
multi_class='ovr')))
```

```
models.append(('Лінійний дискримінантний аналіз', LinearDiscriminantAnalysis()))
models.append(('K-найближчих сусідів', KNeighborsClassifier()))
models.append(('Дерево рішень', DecisionTreeClassifier()))
models.append(('Наївний Байєс', GaussianNB()))
models.append(('Метод опорних векторів', SVC(gamma='auto')))

# Оцінювання моделей
results = []
names = []
print("\n=== Оцінка якості моделей ===")
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}: середня точність = {cv_results.mean():.4f}, стандартне відхилення
= {cv_results.std():.4f}")

# Порівняння алгоритмів (оновлений стиль)
plt.boxplot(results, labels=names, patch_artist=True,
            boxprops=dict(facecolor='lavender'),
            medianprops=dict(color='darkblue'))
plt.title('Порівняння точності алгоритмів класифікації ірисів', fontsize=14,
color='darkred')
plt.ylabel('Accuracy')
plt.xticks(rotation=20)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

# Тренування кращої моделі (SVM) на навчальних даних
model = SVC(gamma='auto')
model.fit(X_train, y_train)
predictions = model.predict(X_validation)

# Оцінка точності прогнозу
print("\n=== Оцінка моделі SVM ===")
print("Точність (Accuracy): {:.2f}%".format(accuracy_score(y_validation, predictions)
* 100))
print("\nМатриця помилок:")
print(confusion_matrix(y_validation, predictions))
print("\nЗвіт про класифікацію:")
print(classification_report(y_validation, predictions))


# Прогноз для нового зразка ірису
X_new = np.array([[5.0, 2.9, 1.0, 0.2]])
prediction = model.predict(X_new)
```

```
predicted_class =
iris_dataset['target_names'][list(iris_dataset['target_names']).index(prediction[0])]
\
    if prediction[0] in iris_dataset['target_names'] else prediction[0]

print("\n=== Новий прогноз ===")
print("Вхідні дані:", X_new)
print("Передбачений клас:", prediction[0])
print("Імовірний сорт ірису:", predicted_class)
```

LR_2_task_4.py

```
import numpy as np
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import make_scorer, f1_score, accuracy_score, confusion_matrix,
classification_report
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.model_selection import train_test_split, cross_val_score,
StratifiedKFold
import matplotlib.pyplot as plt

# Зчитування даних
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.strip().split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            y.append(0)
            count_class1 += 1
```

```
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            y.append(1)
            count_class2 += 1

X = np.array(X)

#  Перетворення категоріальних даних у 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 = np.array(y)

#  Розбиття на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

#   Порівняння моделей
models = [
    ('LR', LogisticRegression(solver='liblinear')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('SVM', SVC(kernel='rbf', gamma='scale'))  # покращений варіант
]

results = []
names = []
f1_scorer = make_scorer(f1_score, average='weighted')

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')
    f1_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring=f1_scorer)
    results.append(cv_results)
    names.append(name)
    print(f"{name} - Accuracy: {cv_results.mean():.4f} ({cv_results.std():.4f})")
    print(f"{name} - F1 Score: {f1_results.mean():.4f} ({f1_results.std():.4f})\n")
```

```
# Побудова графіку точності
plt.boxplot(results, tick_labels=names, patch_artist=True,
            boxprops=dict(facecolor='lavender'),
            medianprops=dict(color='darkblue'))
plt.title('Порівняння точності алгоритмів класифікації', fontsize=14,
color='darkred')
plt.ylabel('Accuracy')
plt.xticks(rotation=20)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

#  Тренування найкращої моделі (SVM)
model = SVC(kernel='rbf', gamma='scale')
model.fit(X_train, y_train)
predictions = model.predict(X_test)

#   Оцінка моделі
print("\n   Оцінка моделі SVM   ")
print("Точність (Accuracy): {:.2f}%".format(accuracy_score(y_test, predictions) *
100))
print("\nМатриця помилок:")
print(confusion_matrix(y_test, predictions))
print("\nЗвіт про класифікацію:")
print(classification_report(y_test, predictions, zero_division=0))

#  Прогноз для нового зразка
# Використаємо реальний приклад із тестових даних
X_new = X_test[0].reshape(1, -1)
prediction = model.predict(X_new)
predicted_class = '<=50K' if prediction[0] == 0 else '>50K'

print("\n   Новий прогноз   ")
print("Приклад із тестових даних:", X_new)
print("Імовірний дохід:", predicted_class)
```

LR_2_task_5.py

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import RidgeClassifier
from sklearn import metrics
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from io import BytesIO

# Завантаження даних Iris
```

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

# Розділення на тренувальні та тестові дані
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_state=0)

# Ініціалізація та налаштування класифікатора Ridge
clf = RidgeClassifier(tol=1e-2, solver='sag')  # tol - точність розв'язку, solver -
алгоритм оптимізації
clf.fit(Xtrain, ytrain)

# Прогнозування на тестових даних
ypred = clf.predict(Xtest)

# Оцінка якості класифікації
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))

# Звіт про класифікацію
print('\t\tClassification Report:\n', metrics.classification_report(ytest, ypred))

# Матриця 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')
plt.savefig("Confusion.jpg")

# Збереження SVG
f = BytesIO()
plt.savefig(f, format="svg")
```