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

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

Назви	Що вони означають	Вид
Age	вік особи	числовий
Workclass	тип зайнятості	категоріальний
Fnlwgt	фінальна вага	числовий
Education	рівень освіти	категоріальний
Education-num	кількість років навчання	числовий
Marital-status	сімейний стан	категоріальний
Occupation	сфера зайнятості	категоріальний
Relationship	сімейне становище в до-могосподарстві	категоріальний
Race	раса	категоріальний
Sex	стать	категоріальний
Capital-gain	дохід від капіталу	числовий
Capital-loss	втрати від капіталу	числовий
Hours-per-week	кількість годин роботи на тиждень	числовий
Native-country	країна народження	категоріальний

1.2. Обчисліть значення інших показників якості класифікації (акуратність, повнота, точність) та разом з F1 занесіть їх у звіт.

Лістинг програми:

```
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
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Вхідний файл, який містить дані
input_file = 'income_data.txt'

# Читання даних
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
        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)

# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))

# Розділення даних на навчальні та тестові вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Навчання класифікатора на навчальних даних
classifier.fit(X_train, y_train)

# Обчислення основних показників якості класифікації
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Виведення результатів
print(f"Accuracy (Акуратність): {accuracy * 100:.2f}%")
print(f"Precision (Точність): {precision * 100:.2f}%")
print(f"Recall (Повнота): {recall * 100:.2f}%")
print(f"F1 Score (F-міра): {f1 * 100:.2f}%")

# Передбачення результату для тестової точки даних
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 даних
```

```
input_data_encoded = np.array([-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

# Використання класифікатора для передбачення класу
input_data_encoded = input_data_encoded.reshape(1, -1)  # Має бути 2D масив для
predict
predicted_class = classifier.predict(input_data_encoded)

# Виведення результату для нової тестової точки даних
print("Передбачений клас:", label_encoder[-
1].inverse_transform([predicted_class[0]])[0])
```

Виконання програми:

```
Accuracy (Акуратність): 79.56%
Precision (Точність): 79.26%
Recall (Повнота): 79.56%
F1 Score (F-міра): 75.75%
Передбачений клас: <=50K

Process finished with exit code 0
```

1.3. Зробіть висновок до якого класу належить тестова точка.

З результату ми можемо побачити що наша точка відноситься до класу <=50k

2.1. Використовуючи набір даних та код з попереднього завдання створіть та дослідіть нелінійні класифікатори SVM.

з поліноміальним ядром;

з гаусовим ядром;

з сигмоїдальним ядром.

Для кожного виду класифікатора отримайте та запишіть у звіт показники якості алгоритму класифікації.

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Вхідний файл, який містить дані
input_file = 'income_data.txt'

# Читання даних
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
        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)

# Розділення даних на навчальні та тестові вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Функція для навчання та оцінки SVM
def train_and_evaluate_svm(kernel_type):
    print(f"\nSVM з {kernel_type} ядром:")
    classifier = SVC(kernel=kernel_type)
    classifier.fit(X_train, y_train)
    # Передбачення та оцінка
    y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    # Виведення результатів
    print(f"Accuracy (Акуратність): {accuracy * 100:.2f}%")
    print(f"Precision (Точність): {precision * 100:.2f}%")
    print(f"Recall (Повнота): {recall * 100:.2f}%")
    print(f"F1 Score (F-міра): {f1 * 100:.2f}%")

# Гаусове (радіальне) ядро
train_and_evaluate_svm('rbf')

# Сигмоїдальне ядро
train_and_evaluate_svm('sigmoid')

# Поліноміальне ядро
train_and_evaluate_svm('poly')
```

Виконання програми:

```
SVM з rbf ядром:
Accuracy (Акуратність): 78.19%
Precision (Точність): 82.82%
Recall (Повнота): 78.19%
F1 Score (F-міра): 71.51%

SVM з sigmoid ядром:
Accuracy (Акуратність): 60.47%
Precision (Точність): 60.64%
Recall (Повнота): 60.47%
F1 Score (F-міра): 60.55%

SVM з poly ядром:
Accuracy (Акуратність): 76.71%
Precision (Точність): 79.49%
Recall (Повнота): 76.71%
F1 Score (F-міра): 68.99%

Process finished with exit code 0
```

2.2. У висновках опишіть який з видів SVM найкраще виконує завдання класифікації за результатами тренування.

RBF ядро показало найкращі результати за **точністю (precision)** та **повнотою (recall)**, що робить його найбільш ефективним у класифікації в цьому експерименті. Крім того, це ядро виконувалося найшвидше, що є ще однією перевагою.

3.1. Код для ознайомлення зі структурою даних та результати його виконання занесіть у звіт

Лістинг програми:

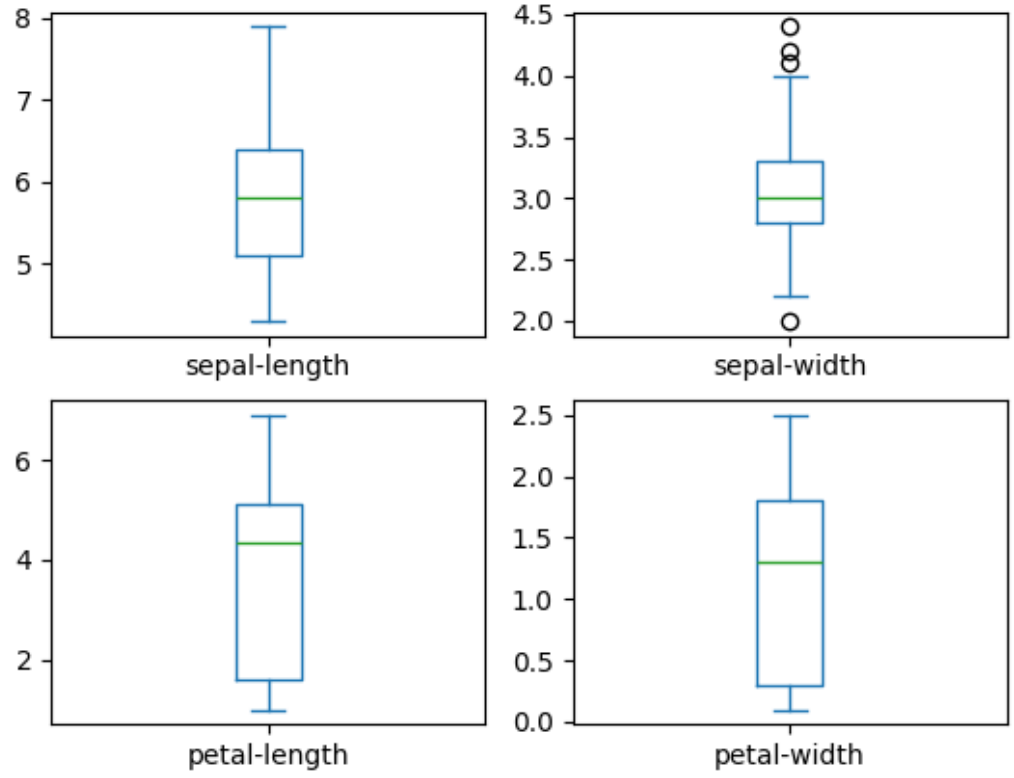
```
from sklearn.datasets import load_iris
iris_dataset = load_iris()

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("Тип масиву target: {}".format(type(iris_dataset['target'])))

print("Відповіді:\n{}".format(iris_dataset['target']))
```

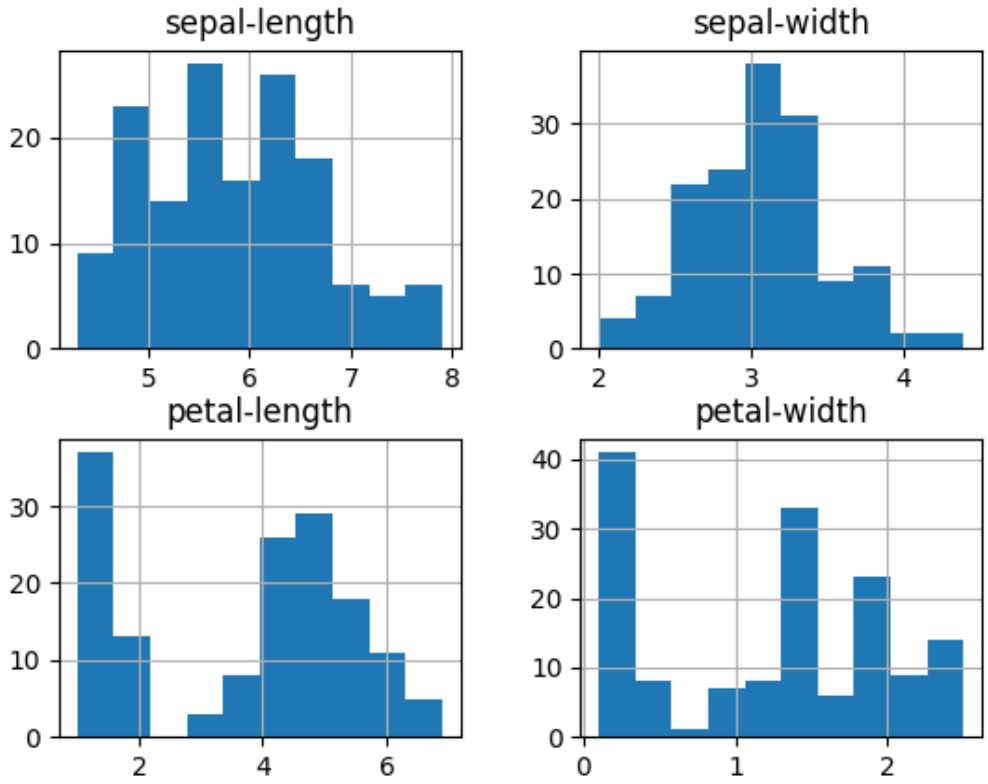
Виконання програми:

```
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'>
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 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 2
 2 2]
```

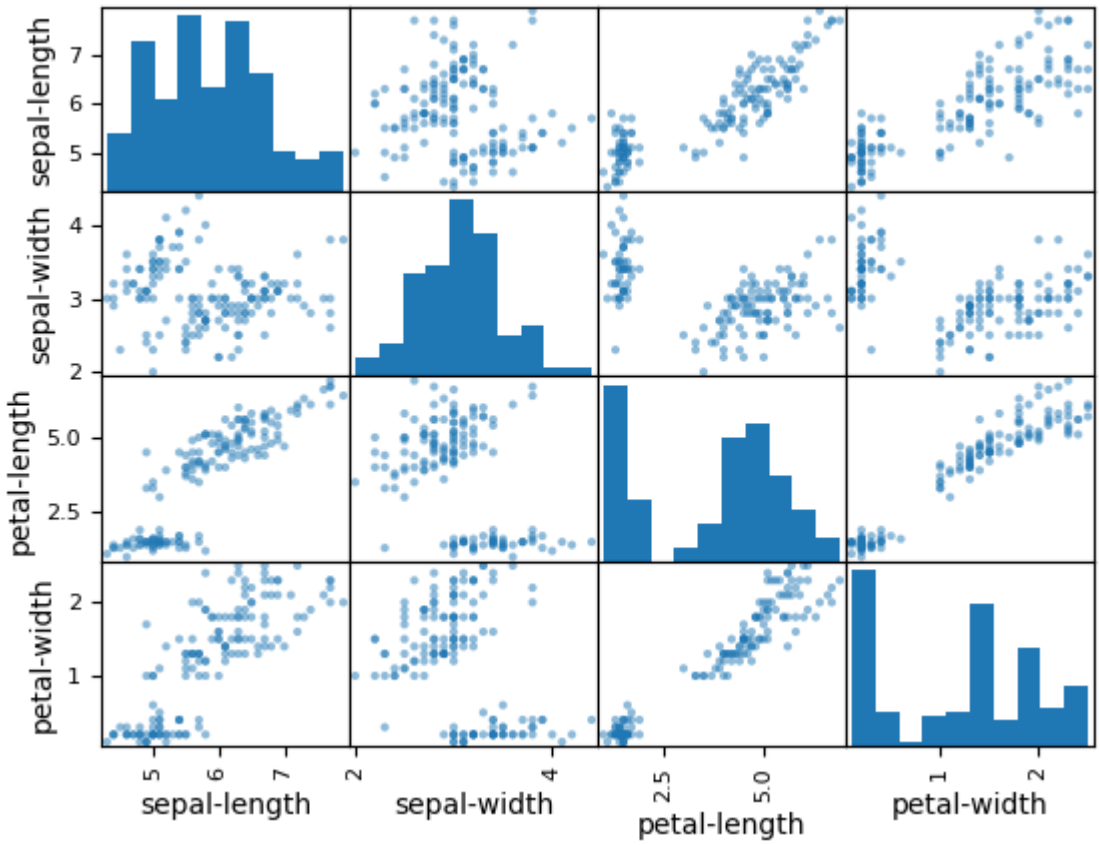
3.2. Графіки функції занесіть у звіт!



3.3. Графіки функції занесіть у звіт!



3.4. Код для візуалізації та отримані графіки занесіть у звіт

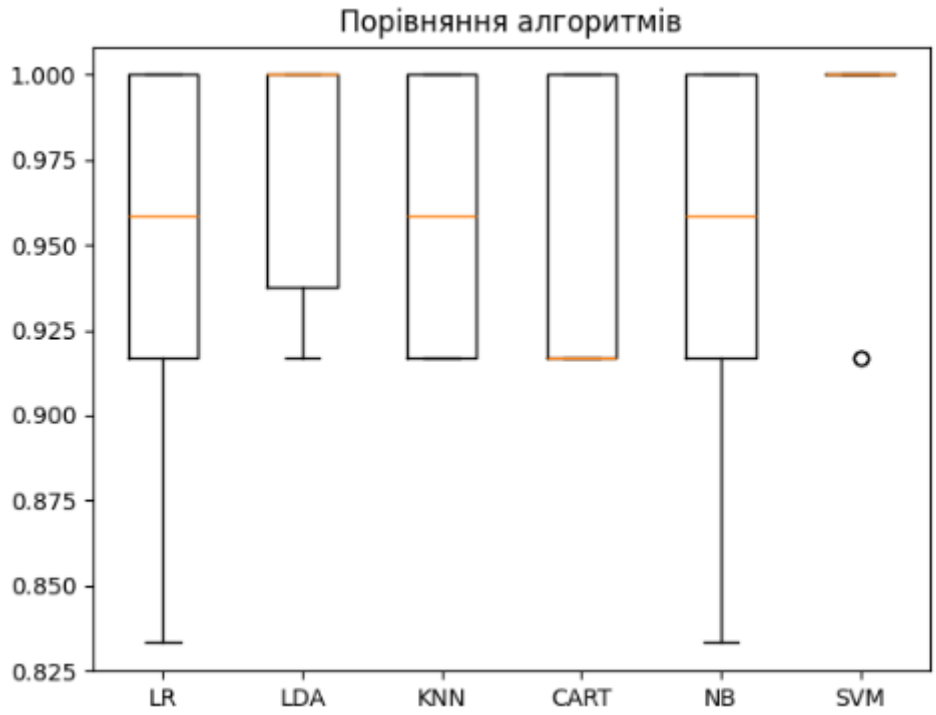


3.5 Отримані графіки та результати занесіть у звіт Виберіть та напишіть чому обраний вами метод класифікації ви вважаєте найкращим

```
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
```

```
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
```

```
Iris-versicolor    50
Iris-virginica     50
dtype: int64
SVM: 0.983333 (0.033333)

Process finished with exit code 0
```



Серед усіх доступних методів класифікації, я б виділив **Support Vector Machine (SVM)** із нелінійним ядром як оптимальний вибір для задач, де дані мають складну структуру. Завдяки можливості використовувати ядрові функції (наприклад, RBF), цей метод ефективно визначає нелінійні межі між класами, забезпечуючи високу точність класифікації. SVM також відзначається своєю стабільністю, гнучкістю та здатністю добре працювати навіть у випадках, коли класи мають складну геометрію. Це робить його одним із найкращих варіантів для задач, що вимагають високої якості результатів.

3.6. Коди та результати занесіть у звіт. У висновках опишіть яку якість класифікації за результатами тренування вдалося досягти та до якого класу належить квітка з кроку 8.

Лістинг програми:

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import 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
from sklearn.multiclass import OneVsRestClassifier
import numpy as np

# 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("Розмірність датасету: {}".format(dataset.shape))

# Зріз даних head
print("\nПерші 20 рядків даних:")
print(dataset.head(20))

# Стастичні зведення методом describe
print("\nСтатистичні зведення даних:")
print(dataset.describe())

# Розподіл за атрибутом class
print("\nРозподіл класів:")
print(dataset.groupby('class').size())

# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)
pyplot.title('Діаграма розмаху для атрибутів')
pyplot.show()

# Гістограма розподілу атрибутів датасета
dataset.hist()
pyplot.title('Гістограма розподілу атрибутів')
pyplot.show()

# Матриця діаграм розсіювання
scatter_matrix(dataset)
```

```
pyplot.title('Матриця діаграм розсіювання')
pyplot.show()

# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
X = array[:, 0:4]  # Вибір перших 4-х стовпців
Y = array[:, 4]  # Вибір 5-го стовпця
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(('LR', OneVsRestClassifier(LogisticRegression(solver='liblinear'))))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('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('{}: {:.2f} ± {:.2f}'.format(name, cv_results.mean(), cv_results.std()))

# Порівняння алгоритмів
pyplot.boxplot(results, tick_labels=names)  # Зміна labels на tick_labels
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("\nОцінка моделі на контрольній вибірці:")
```

```
print("Точність: {:.2f}".format(accuracy_score(Y_validation, predictions)))
print("\nМатриця плутанини:")
print(confusion_matrix(Y_validation, predictions))
print("\nЗвіт про класифікацію:")
print(classification_report(Y_validation, predictions))

# Прогноз для нових даних
X_new = np.array([[5, 2.9, 1, 0.2]])
print("\nФорма масиву X_new: {}".format(X_new.shape))
prediction = model.predict(X_new)

# Виводимо результати прогнозу
print("Прогноз для нових даних: {}".format(prediction))
print("Спрогнозована метка класу: {}".format(prediction[0]))
```

```
Оцінка моделі на контрольній вибірці:
Точність: 0.97

Матриця плутанини:
[[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


Форма масиву X_new: (1, 4)
Прогноз для нових даних: ['Iris-setosa']
Спрогнозована мітка класу: Iris-setosa

Process finished with exit code 0
```

Модель передбачила, що ця квітка належить до класу **Iris-setosa**.

4.1. Порівняння якості класифікаторів для набору даних income_data.txt

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
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
import pandas as pd

# Вхідний файл, який містить дані
input_file = 'income_data.txt'

# Читання даних
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.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, dtype=object)
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)  # Вхідні ознаки
y = X_encoded[:, -1].astype(int)    # Мітки класу

# Завантажуємо алгоритми моделі
models = []
models.append(('LR', LogisticRegression(solver='liblinear')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', OneVsOneClassifier(LinearSVC(random_state=0))))

# Оцінка кожної моделі
results = []
names = []

for name, model in models:
    # Розділення даних на навчальні та тестові вибірки
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

    # Навчання моделі
    model.fit(X_train, y_train)

    # Передбачення на тестових даних
    y_pred = model.predict(X_test)
```

```
    # Обчислення показників якості
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    # Додавання результатів
    results.append((accuracy, precision, recall, f1))
    names.append(name)

    print(f"Модель: {name}")
    print(f"Accuracy: {accuracy * 100:.2f}%")
    print(f"Precision: {precision * 100:.2f}%")
    print(f"Recall: {recall * 100:.2f}%")
    print(f"F1 Score: {f1 * 100:.2f}%\n")

# Порівняння алгоритмів
labels = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
metrics = np.array(results)

for i, label in enumerate(labels):
    plt.figure(i)
    plt.bar(names, metrics[:, i])
    plt.title(label)
    plt.ylabel('Оцінка')
    plt.xticks(rotation=45)
    plt.ylim(0, 1)
    plt.show()

# Передбачення результату для тестової точки даних
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 даних
input_data_encoded = np.array([-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

# Використання класифікатора для передбачення класу
input_data_encoded = input_data_encoded.reshape(1, -1)  # Має бути 2D масив для predict
predicted_class = models[-1][1].predict(input_data_encoded)  # Остання модель - SVM

# Виведення результату для нової тестової точки даних
print("Передбачений клас:", label_encoder[-1].inverse_transform([predicted_class[0]])[0])
```

Виконання програми:

```
Модель: LR
Accuracy: 78.55%
Precision: 76.88%
Recall: 78.55%
F1 Score: 75.33%
```

```
Модель: LDA
Accuracy: 81.12%
Precision: 79.96%
Recall: 81.12%
F1 Score: 79.49%
```

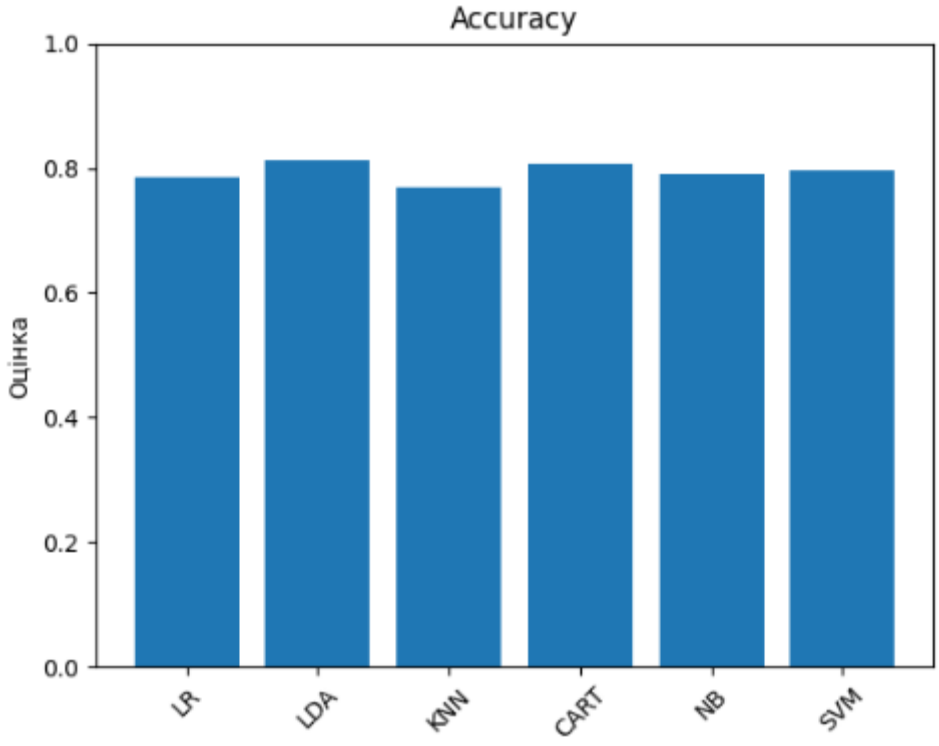
```
Модель: KNN
Accuracy: 76.78%
Precision: 74.31%
Recall: 76.78%
F1 Score: 74.27%
```

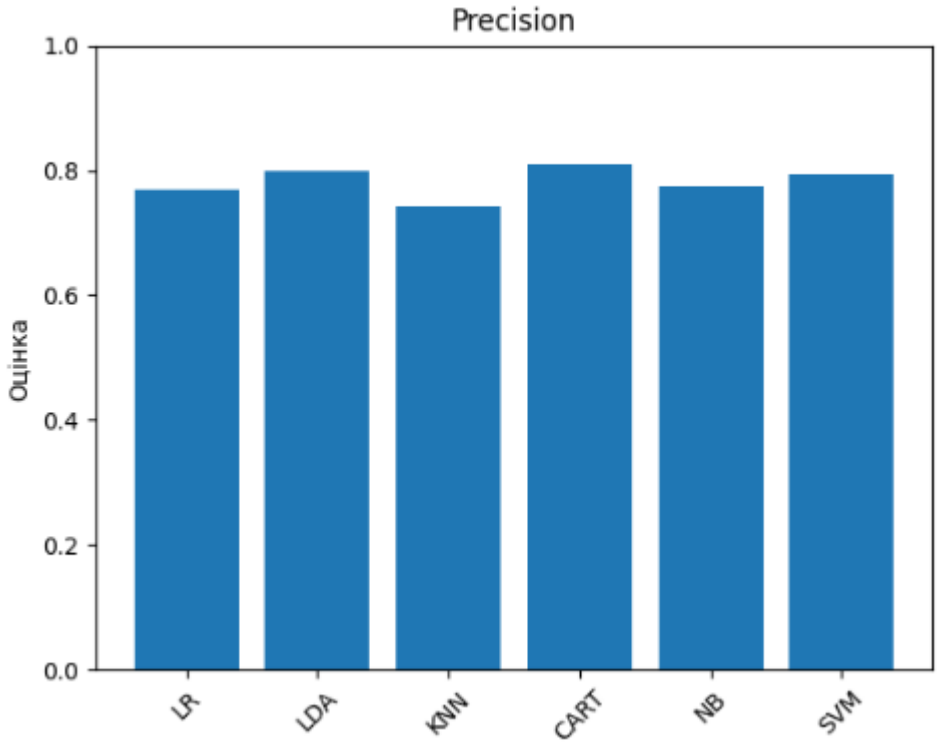
```
Модель: CART
Accuracy: 80.59%
Precision: 80.92%
Recall: 80.59%
F1 Score: 80.74%
```

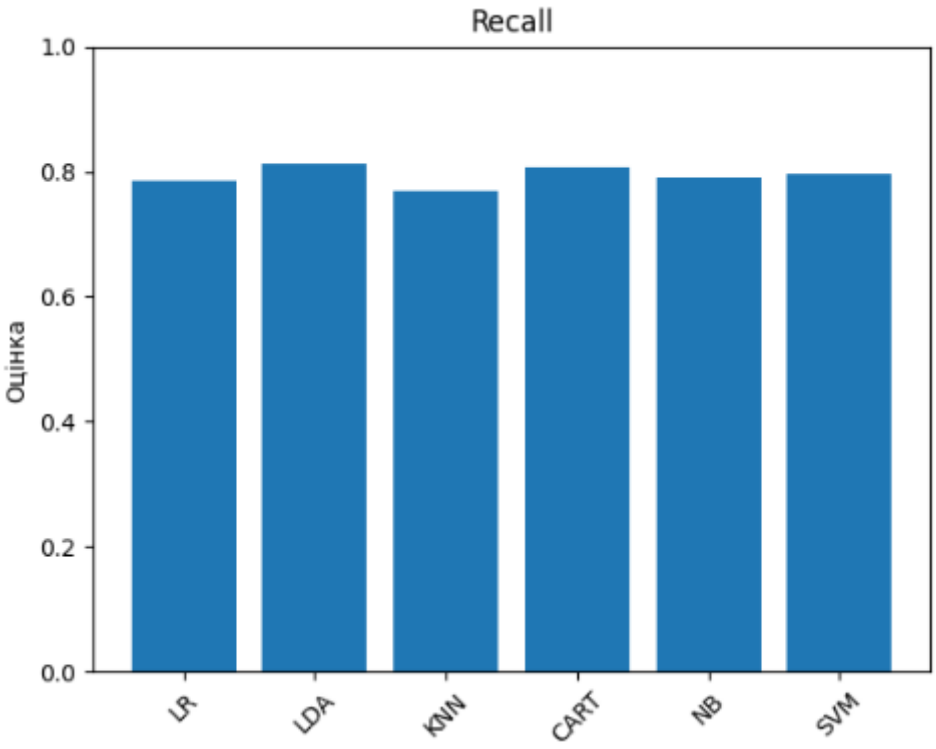
```
Модель: NB
Accuracy: 78.95%
Precision: 77.43%
Recall: 78.95%
F1 Score: 75.91%
```

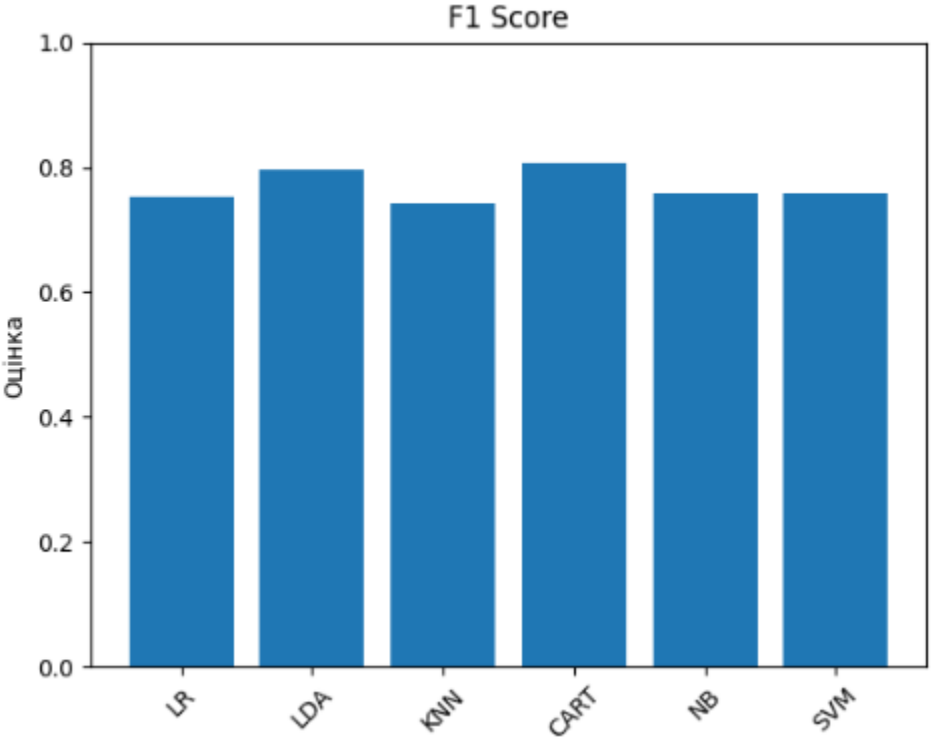
```
Модель: SVM
Accuracy: 79.56%
Precision: 79.26%
Recall: 79.56%
F1 Score: 75.75%
```

```
Передбачений клас: <=50K

Process finished with exit code 0
```


Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Оцінка
LR
LDA
KNN
CART
NB
SVM


Precision
1.0
0.8
0.6
0.4
0.2
0.0
Оцінка
LR
LDA
KNN
CART
NB
SVM





Класифікація даних лінійним класифікатором Ridge

5.1. Виправте код та виконайте класифікацію.

Лістинг програми:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
```

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from io import BytesIO  # needed for plot
import seaborn as sns
import matplotlib.pyplot as plt

# Завантаження даних
iris = load_iris()
X, y = iris.data, iris.target

# Розбиття даних на навчальну і тестову вибірки
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,
random_state=0)

# Ініціалізація та навчання класифікатора Ridge
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)

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

# Розрахунок показників якості
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(ytest, ypred))

# Побудова матриці 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')

# Збереження матриці плутанини у файл
plt.savefig("Confusion.jpg")
```

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

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

1. **tol=1e-2** — Це параметр толерантності (tolerance), який визначає точність збіжності алгоритму. Якщо різниця в значеннях функції втрат між ітераціями менша за це значення, процес навчання зупиняється.
2. **solver="sag"** — Це вибір алгоритму оптимізації. SAG (Stochastic Average Gradient) — це варіант градієнтного спуску, який обчислює середній градієнт на кожному кроці, що дозволяє працювати з великими наборами даних ефективніше.

5.3. Опишіть які показники якості використовуються та їх отримані результати.

Точність (Accuracy): Частка правильних передбачень від загальної кількості. Вона відображає загальну ефективність моделі в класифікації всіх прикладів.

Точність (Precision): Відсоток правильних позитивних прогнозів серед усіх передбачених як позитивні. Цей показник оцінює, наскільки добре модель виявляє саме позитивні класи.

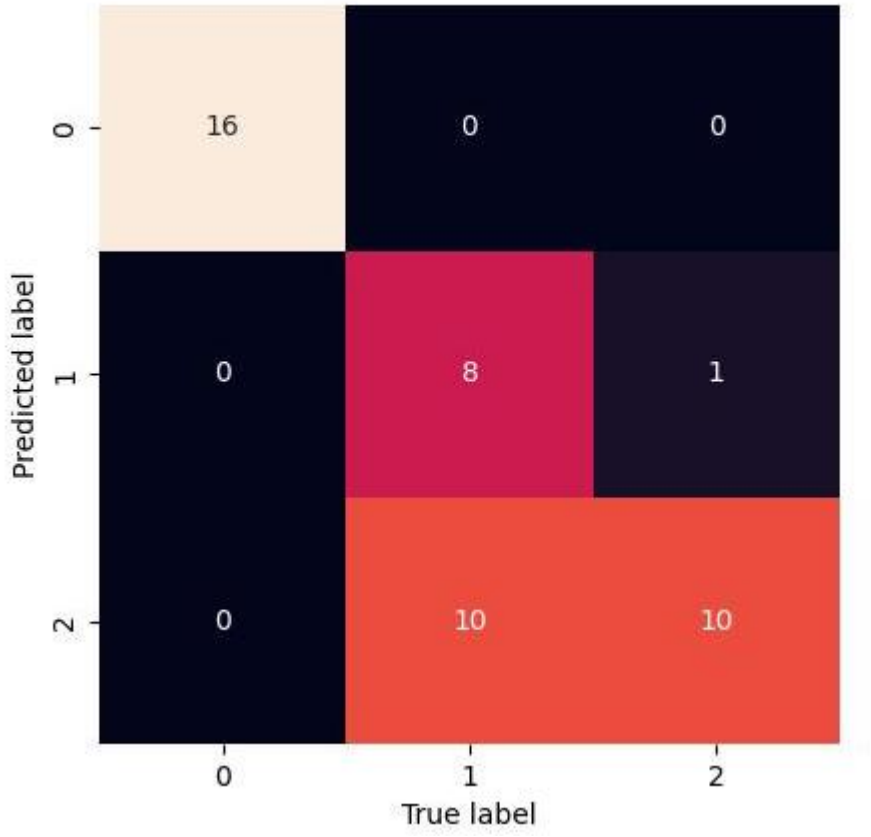
Повнота (Recall): Частка правильно визначених позитивних прикладів серед усіх фактично позитивних. Ця метрика показує, наскільки добре модель знаходить усі позитивні випадки.

F1-міра (F1 Score): Гармонічне середнє між точністю та повнотою, яке дає збалансовану оцінку роботи моделі в класифікації.

Коефіцієнт Коена Каппа (Cohen Kappa Score): Показник узгодженості між передбаченнями моделі та реальними значеннями, з урахуванням випадкових збігів. Він показує, наскільки модель краще за випадкове вгадування.

Кореляція Метьюза (Matthews Corrcoef): Метрика, що враховує всі компоненти матриці плутанини (TP, FP, TN, FN) для оцінки якості класифікації. Особливо корисна для незбалансованих наборів даних, оскільки забезпечує більш точний аналіз.

5.4. Вставте у звіт та поясніть зображення Confusion.jpg



матриця плутанини показує результат роботи класифікатора Ridge на наборі даних Iris. Візуально відображені три класи (0, 1, 2), які представляють різні види квітки Iris. Ось що означають дані у цій матриці:

- **На діагоналі**: елементи (0,0), (1,1), (2,2) — це правильно передбачені класи. Тобто, модель правильно передбачила 16 квіток класу 0, 8 квіток класу 1, і 10 квіток класу 2.
- **Позадіагональні елементи**: це неправильно передбачені класи.
    - Елемент (2,1) означає, що модель передбачила 10 квіток як клас 2, тоді як вони належали до класу 1.

- Елемент (1,2) означає, що одна квітка з класу 2 була неправильно класифікована як клас 1.

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

Коефіцієнт Коена Каппа:

- Використовується для вимірювання ступеня узгодженості між двома джерелами оцінювання (наприклад, між прогнозами моделі та реальними значеннями) з урахуванням випадкових збігів.
- Метрика варіюється від -1 (повна незгода) до 1 (повна узгодженість).
- У нашому випадку значення, близьке до 1, демонструє помірний рівень відповідності між прогнозами моделі та реальними даними.

Коефіцієнт кореляції Метьюза:

- Збалансована метрика для оцінки класифікації, яка враховує всі чотири елементи матриці плутанини (TP, FP, TN, FN), що робить її особливо цінною для аналізу незбалансованих даних.
- Значення змінюється від -1 (невдала модель) до 1 (ідеальна модель).
- У нашому випадку значення, близьке до 1, вказує на помірний рівень якості класифікації.

Посилання на ГітХаб: https://github.com/Kn211mna/AI-YT

Висновок: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчитися їх порівнювати.