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

Ознаки з набору даних – їх назви, що вони позначають та вид.

Variable Name	***Type***	***Description***
age	Integer	(Вік)
workclass	Categorical	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
education	Categorical	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	Integer	(рівень освіти)
marital-status	Categorical	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
race	Categorical	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
relationship	Categorical	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

occupation	Categorical	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.
sex	Binary	Female, Male.
capital-gain	Integer	(прибуток)
capital-loss	Integer	(витрати)
hours-per-week	Integer	(години роботи)
native-country	Categorical	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.
income	Binary	>50K, <=50K.

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
input_file = 'income_data.txt'
# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
```

```
max_datapoints = 25000
input_file = 'income_data.txt'
X = []  # Один масив X для всіх даних
max_datapoints = 25000
count_class1 = 0
count_class2 = 0

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

# Перевірка кількості точок у кожному класі
print("Кількість точок у класі <=50K:", count_class1)
print("Кількість точок у класі >50K:", count_class2)
# Перетворення на масив numpy
X = np.array(X)

# Виведіть інформацію про кількість точок у масиві X
print("Кількість точок у масиві X:", len(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:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)
import warnings
warnings.filterwarnings("ignore")
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
```

```
# Навчання класифікатора
classifier.fit(X, y)

# Розділення даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Створення нового класифікатора для навчання на навчальних даних
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)

# Прогнозування на тестових даних
y_test_pred = classifier.predict(X_test)

from sklearn.model_selection import cross_val_score

# Обчислення F-міри для SVM-класифікатора з використанням крос-валідації
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Han-dlers-cleaners', 'Not-in-family', 'White', 'Male',
'0', '0', '40', 'United-States']

# Кодування тестової точки даних
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])


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

# Використання класифікатора для кодованої точки даних
# та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Отримання передбачених класів для тестових даних
y_test_pred = classifier.predict(X_test)

# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:",100 * accuracy)

# Обчислення точності
```

```
precision = precision_score(y_test, y_test_pred)
print("Precision:",100 * precision)

# Обчислення повноти
recall = recall_score(y_test, y_test_pred)
print("Recall:",100 * recall)
```

```
[ ]  from sklearn.model_selection import cross_val_score

     # Обчислення F-міри для SVM-класифікатора з використанням крос-валідації
     f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
     print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

     F1 score: 56.15%
```

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

# Використання класифікатора для кодованої точки даних
# та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Отримання передбачених класів для тестових даних
y_test_pred = classifier.predict(X_test)

# Обчислення акуратності
accuracy = accuracy_score(y_test, y_test_pred)
print("Accuracy:",100 * accuracy)

# Обчислення точності
precision = precision_score(y_test, y_test_pred)
print("Precision:",100 * precision)

# Обчислення повноти
recall = recall_score(y_test, y_test_pred)
print("Recall:",100 * recall)

<=50K
Accuracy: 77.5070445880988
Precision: 95.09803921568627
Recall: 12.589227774172615
```

Рис.1. Показники якості класифікації та передбачення результату для тестової точки.

Висновки: За отриманими даними можна стверджувати, що тестова точка відноситься до першого класу <=50K.

Завдання 2

Лістинг програм:

2.1

```
from sklearn.svm import SVC
```

```
# Створення класифікатора з поліноміальним ядром
poly_classifier = SVC(kernel='poly', degree=2, random_state=0)
poly_classifier.fit(X_train, y_train)

# Прогнозування на тестових даних
y_test_pred_poly = poly_classifier.predict(X_test)

# Оцінка результатів
accuracy_poly = accuracy_score(y_test, y_test_pred_poly)
precision_poly = precision_score(y_test, y_test_pred_poly)
recall_poly = recall_score(y_test, y_test_pred_poly)
print("Показники для нелінійного класифікатора SVM з поліноміальним ядром:")
print("Accuracy:", 100 * accuracy_poly)
print("Precision:", 100 * precision_poly)
print("Recall:", 100 * recall_poly)
```

2.2

```
from sklearn.svm import SVC

# Створення класифікатора з гаусовим (RBF) ядром
rbf_classifier = SVC(kernel='rbf', random_state=0)
rbf_classifier.fit(X_train, y_train)

# Прогнозування на тестових даних
y_test_pred_rbf = rbf_classifier.predict(X_test)

# Оцінка результатів
accuracy_rbf = accuracy_score(y_test, y_test_pred_rbf)
precision_rbf = precision_score(y_test, y_test_pred_rbf)
recall_rbf = recall_score(y_test, y_test_pred_rbf)
print("Показники для нелінійного класифікатора SVM з гаусовим ядром:")
print("Accuracy:", 100 * accuracy_rbf)
print("Precision:", 100 * precision_rbf)
print("Recall:", 100 * recall_rbf)
```

2.3

```
from sklearn.svm import SVC

# Створення класифікатора з сигмоїдальним ядром
sigmoid_classifier = SVC(kernel='sigmoid', random_state=0)
sigmoid_classifier.fit(X_train, y_train)

# Прогнозування на тестових даних
y_test_pred_sigmoid = sigmoid_classifier.predict(X_test)

# Оцінка результатів
accuracy_sigmoid = accuracy_score(y_test, y_test_pred_sigmoid)
precision_sigmoid = precision_score(y_test, y_test_pred_sigmoid)
```

```
recall_sigmoid = recall_score(y_test, y_test_pred_sigmoid)
print("Показники для нелінійного класифікатора SVM з сигмоїдальним ядром:")
print("Accuracy:", 100 * accuracy_sigmoid)
print("Precision:", 100 * precision_sigmoid)
print("Recall:", 100 * recall_sigmoid)
```

```
Показники для нелінійного класифікатора SVM з поліноміальним ядром:
Accuracy: 77.39101607823636
Precision: 93.59605911330048
Recall: 12.329656067488644
```

Рис.3. Показники якості класифікації нелінійного класифікатора SVM з поліноміальним ядром.

```
Показники для нелінійного класифікатора SVM з гаусовим ядром:
Accuracy: 78.18664014586442
Precision: 98.7012987012987
Recall: 14.795587280986371
```

Рис.4. Показники якості класифікації нелінійного класифікатора SVM з гаусовим ядром.

```
Показники для нелінійного класифікатора SVM з сигмоїдальним ядром:
Accuracy: 60.46742913973148
Precision: 22.98335467349552
Recall: 23.29656067488644
```

Рис.5. Показники якості класифікації нелінійного класифікатора SVM з сигмоїдальним ядром.

Висновки: За результатами навчання, найкращою моделлю для завдання класифікації виявився класифікатор SVM з гаусовим ядром. Варто відзначити, що класифікатор з поліноміальним ядром може показати кращі результати, якщо збільшити ступінь полінома, проте у нас є великий обсяг даних, і такий підхід вимагає значних обчислювальних ресурсів та часу.

Завдання 3

Лістинг програми для ознайомлення:

```
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("Форма масиву data: {}".format(iris_dataset['data'].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

```
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("Форма масиву data: {}".format(iris_dataset['data'].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

```
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, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Форма масиву data: (150, 4)
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 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 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]
```

Рис.6. Результат виконання коду для ознайомлення.

КРОК 1

Лістинг програми:

```
import pandas as pd

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
```

```
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pd.read_csv(url, names=names)

# shape
print(dataset.shape)

# Зріз даних head
print(dataset.head(10))

# Стастичні зведення методом describe
print(dataset.describe())

# Розподіл за атрибутом class
print(dataset.groupby('class').size())
```

КРОК 2

Лістинг програми:

```
import pandas as pd
import matplotlib.pyplot as plt
dataset.plot(kind='box', subplots=True, layout=(2,2),
sharex=False, sharey=False)
plt.show()
dataset.hist()
plt.show()
```

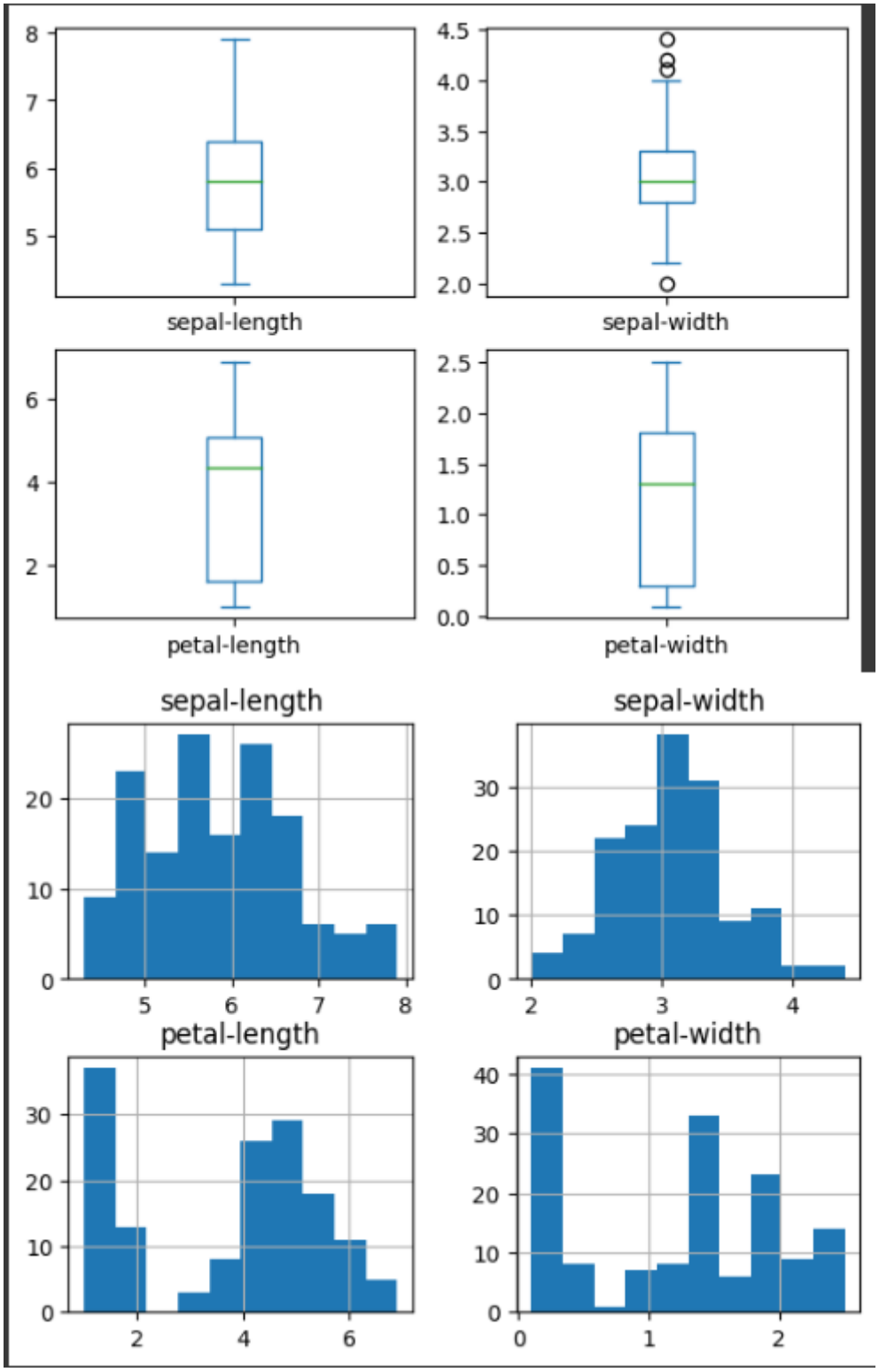

Рис.7. Діаграма розмаху та гістограма розподілу атрибутів датасету.

КРОК 3

Лістинг програми:

```
import numpy as np
from sklearn.model_selection import train_test_split

# Розділення датасету на навчальну та контрольну вибірки
X = dataset.iloc[:, 0:4].values  # Вибираємо перші 4 стовпці як ознаки
```

```
Y = dataset.iloc[:, 4].values  # Вибираємо 5-ий стовпець як цільову змінну

# Розділення X і 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)
```

КРОК 4

Лістинг програми:

```
# Створення моделей
models = [
    ('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('SVM', SVC(gamma='auto'))
]

# Оцінка моделей
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)
    mean_accuracy = cv_results.mean()
    std_accuracy = cv_results.std()
    print(f'{name}: Середнє {mean_accuracy:.4f}, Стандартне відхилення
{std_accuracy:.4f}')

# Порівняння алгоритмів
plt.boxplot(results, labels=names)
plt.title('Порівняння алгоритмів')
plt.ylabel('Точність')
plt.show()
```

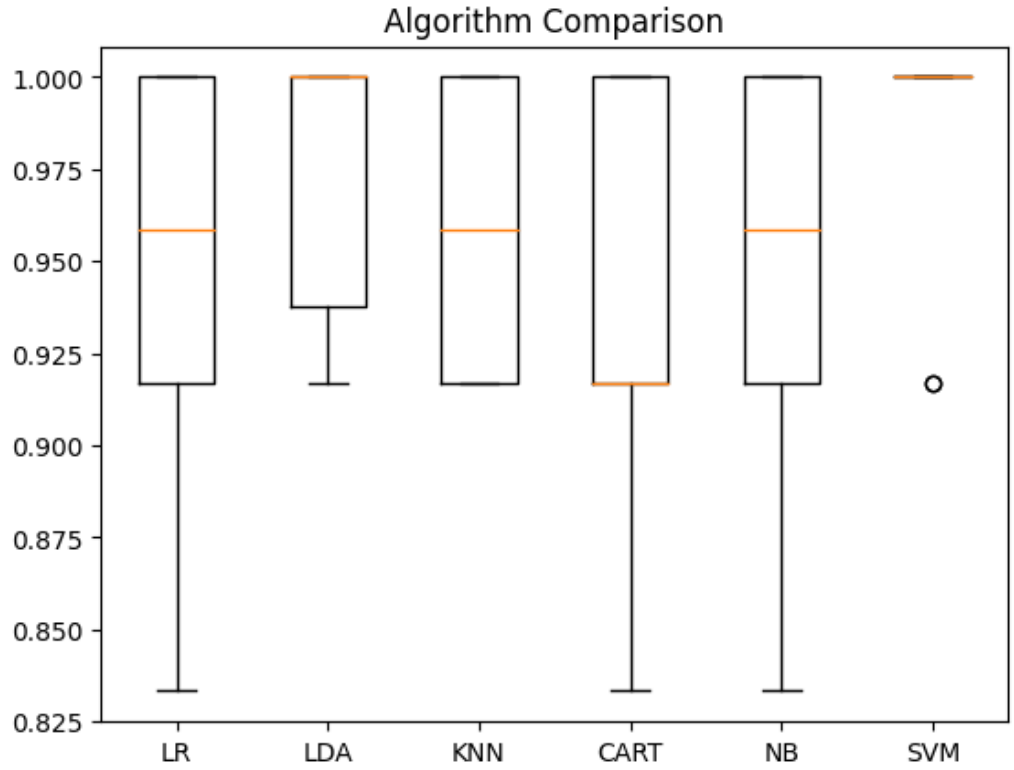

Рис.11. Порівняння алгоритмів.

```
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.941667 (0.053359)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
```

Рис.10. Результат оцінок кожного алгоритму.

Серед розглянутих методів класифікації, найкращим в даному випадку є метод опорних векторів (SVM). Він показав найвищу середню точність, також найменше стандартне відхилення. Ще метод опорних векторів відомий своєю здатністю працювати добре на різних типах даних та у складних задачах класифікації.

КРОК 6

Лістинг програми:

```
# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
```

КРОК 7

Лістинг програми:

```
# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 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
```

Рис.11. Результат оцінки прогнозу. Точність, матриця помилок та звіт про класифікацію.

КРОК 8

Лістинг програми:

```
import numpy as np

# Створюємо новий масив з даними для передбачення
X_new = np.array([[5, 2.9, 1, 0.2]])

# Виводимо форму масиву X_new
print("Форма масиву X_new:", X_new.shape)
prediction = model.predict(X_new)
print("Прогноз класу: {}".format(prediction))
# Отримуємо мітку класу на основі прогнозу
predicted_class = prediction[0]

# Виводимо мітку класу
print("Мітка класу: {}".format(predicted_class))
```

Висновки: на підставі результатів розрахунків щодо точності, матриці помилок і звіту про класифікацію можна встановити, що на наборі даних Iris досягнута висока якість класифікації, імовірність правильної класифікації

становить близько 96.7%. Квітка, яка знаходиться на 8-му рядку, належить до класу Iris-setosa.

Завдання 4

Лістинг програми:

```
# Завантаження бібліотек
from pandas import read_csv
import pandas as pd
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
import numpy as np
from sklearn import metrics

# Завантаження даних з файлу 'income_data.txt'
input_file = 'income_data.txt'
dataset = pd.read_csv(input_file, sep=',', header=None, names=[
    'Age', 'Workclass', 'fnlwgt', 'Education', 'Education_Num',
'Marital_Status',
    'Occupation', 'Relationship', 'Race', 'Sex', 'Capital_Gain',
'Capital_Loss',
    'Hours_Per_Week', 'Native_Country', 'Income'
])

# Підготовка даних: кодування категоріальних змінних
dataset_encoded = pd.get_dummies(dataset, columns=[
    'Workclass', 'Education', 'Marital_Status',
    'Occupation', 'Relationship', 'Race', 'Sex', 'Native_Country'
])

# Розділення на ознаки і цільову змінну
X = dataset_encoded.drop('Income', axis=1)
y = dataset_encoded['Income']

# Розділення даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)
```

```
# Створення моделей
models = [
    ('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', KNeighborsClassifier()),
    ('CART', DecisionTreeClassifier()),
    ('NB', GaussianNB()),
    ('SVM', SVC(gamma='scale'))
]

# Оцінка якості моделей
results = []
names = []
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    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()))
```

```
    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()))

LR: 0.796683 (0.003365)
LDA: 0.839856 (0.006693)
KNN: 0.777290 (0.007474)
CART: 0.818007 (0.006005)
NB: 0.793743 (0.003084)
SVM: 0.792953 (0.002737)
```

Рис.12. Результат виконання.

Висновки: З аналізу результатів стає очевидним, що Linear Discriminant Analysis (LDA) є найкращим алгоритмом з точки зору середньої точності на тестових наборах даних. Ця модель добре ураховує взаємозв'язок між ознаками та цільовою змінною і, крім того, має низьке стандартне відхилення, що свідчить про стабільність моделі. При цьому розрахунки вимагають мало часу та ресурсів.

Завдання 5

Лістинг програми:

```
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
from sklearn.metrics import confusion_matrix
from io import BytesIO
import seaborn as sns
import matplotlib.pyplot as plt

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

# Розділення даних на навчальний та тестовий набори
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3,
random_state=0)

# Створення моделі лінійного класифікатора Ridge
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)

# Прогнозування на тестовому наборі
ypred = clf.predict(Xtest)

# Виведення метрик 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(ypred,
ytest))

# Побудова матриці плутанини з кольоровою шкалою
mat = confusion_matrix(ytest, ypred)
plt.figure(figsize=(8, 6))
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=True, cmap='Blues')
```

```
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.title('Confusion Matrix')
plt.savefig("Confusion.jpg")

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

# Збереження графіку у файл (необов'язково)
plt.savefig("Confusion.svg")
```

```
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
```

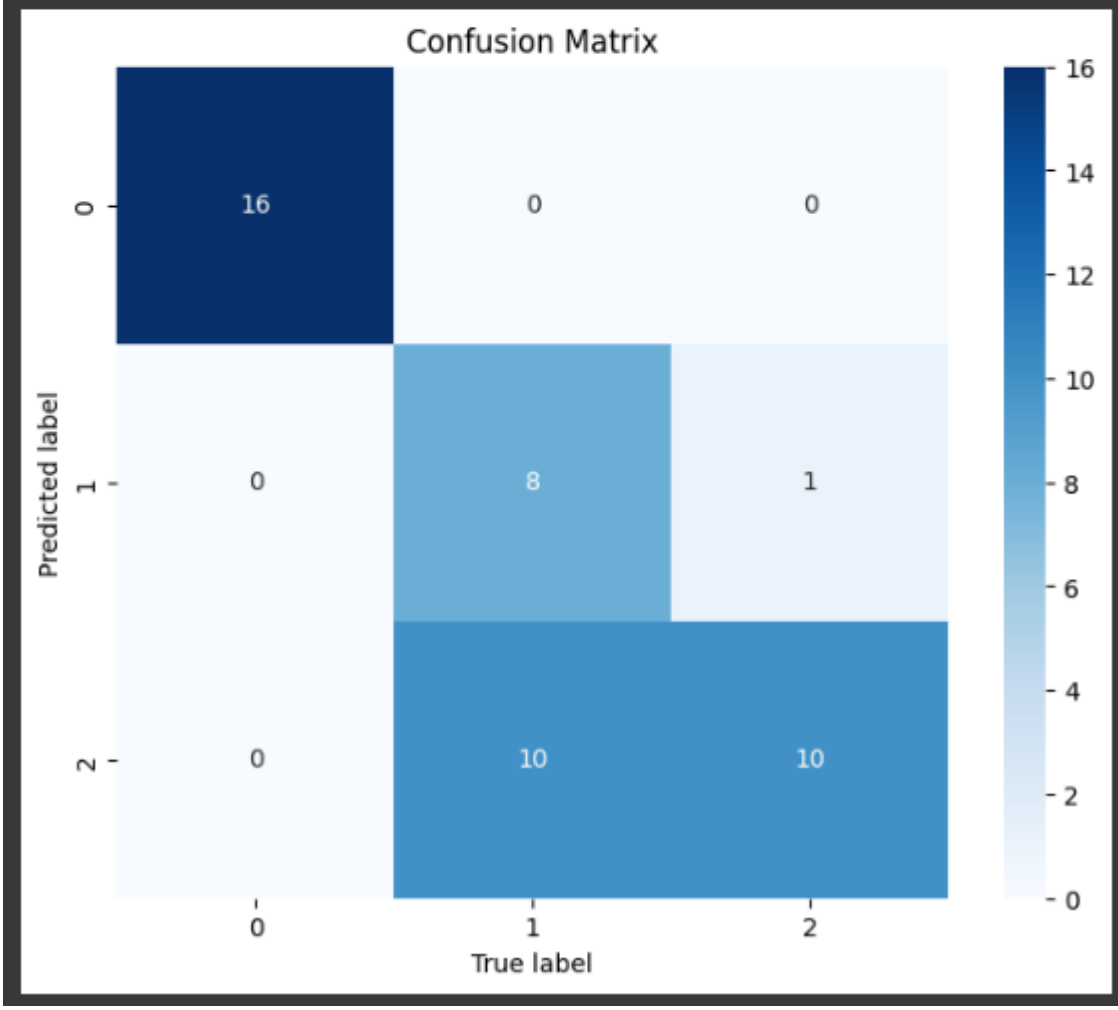

Рис.13. Результат виконання.

Висновки: Ridge використовується для класифікації даних в наведеному коді. Параметр tol=1e-2 визначає точність обчислення і використовується для критерію зупинки оптимізаційного алгоритму. solver="sag" вказує метод оптимізації для Ridge-класифікатора, де "sag" означає Stochastic Average Gradient Descent.

Щодо показників якості:

1. Accuracy (точність): Визначає частку точних передбачень відносно загальної кількості прикладів і становить 75%.
2. Precision (точність): Вимірює точність позитивних передбачень і становить 83%.
3. Recall (повнота): Вимірює здатність моделі виявляти всі позитивні приклади і становить 75%.
4. F1 Score (гармонічне середнє точності і повноти): Становить 75%.

Файл "Confusion.jpg" представляє матрицю плутанини. По вертикалі (вісь y) розташовані передбачені класи, а по горизонталі (вісь x) - фактичні класи. Діагональ матриці показує кількість правильних класифікацій для кожного класу. Наприклад, 16 екземплярів були правильно класифіковані як перший клас, 8 як другий та 10 як третій. Елементи поза діагоналлю вказують на кількість неправильних класифікацій. Наприклад, 1 екземпляр неправильно класифіковано як другий, а 10 як третій.

Коефіцієнт Коена Каппа (Cohen Kappa Score) - це міра узгодженості між реальними і передбаченими мітками, яка враховує випадковий вибір. Результат Каппа лежить в діапазоні від -1 до 1, де 1 означає ідеальну узгодженість, 0 - випадковий результат, а -1 - повну протилежність. У цьому випадку, значення Каппа дорівнює 0.6431, що вказує на помірний рівень узгодженості між фактичними та передбаченими класами.

Коефіцієнт кореляції Метьюза (Matthews Correlation Coefficient) - це також міра узгодженості між реальними і передбаченими мітками, але враховує дисбаланс класів у вибірці. Зазвичай використовується для бінарної класифікації. У цьому випадку, значення Метьюза дорівнює 0.6831, що вказує на добрий рівень узгодженості між фактичними та передбаченими класами.