

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.

<i>Variable Name</i>	<i>Type</i>	<i>Description</i>
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.

					ДУ «Житомирська політехніка».22.122.4.000 – Лр2						
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи			Літ.	Арк.	Аркушів	
Розроб.		Дяченко В.В.								1	
Перевір.											
Керівник											
Н. контр.											
Зав. каф.								ФІКТ Гр. КН-20-1(1)			

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, Trinidad&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
```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

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

```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

```

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', 'Handlers-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])
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([input_data[i]]))
        count += 1

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

```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

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%

```

▶ 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)

```

Accuracy: 77.5070445880988
Precision: 95.09803921568627
Recall: 12.589227774172615

Рис.1. Показники якості класифікації.

```

▶ # 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])

```

<=50K

Рис.2. Передбачення результату для тестової точки.

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки: За отриманим результатом можемо сказати, що тестова точка належить до 1 класу $\leq 50K$.

Завдання 2

Лістинг програм:

2.1

```
# Створення класифікатора з поліноміальним ядром
classifier = SVC(kernel='poly', degree=2, random_state=0)
```

2.2

```
# Створення SVM-класифікатора з ядром rbf
classifier = OneVsOneClassifier(SVC(kernel='rbf', random_state=0))
```

2.3

```
# Створення SVM-класифікатора з ядром rbf
classifier = OneVsOneClassifier(SVC(kernel='sigmoid', random_state=0))
```

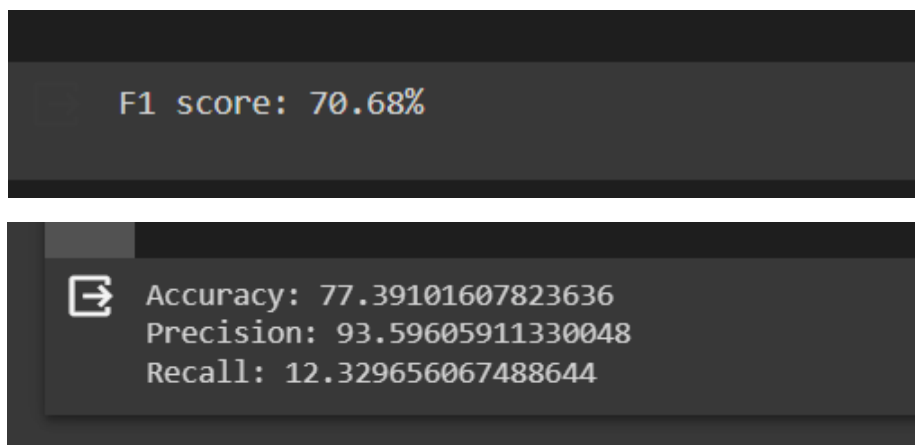


Рис.3. Показники якості класифікації нелінійного класифікатора SVM з поліноміальним ядром.

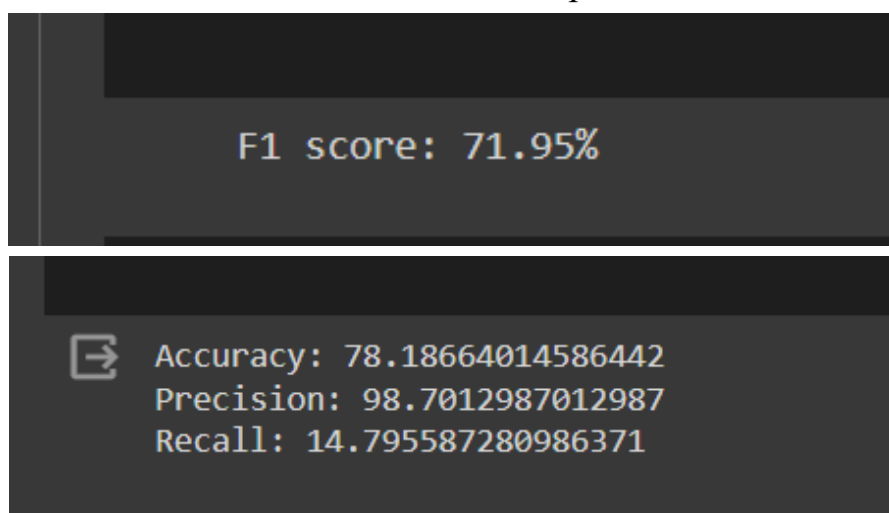


Рис.4. Показники якості класифікації нелінійного класифікатора SVM з гаусовим ядром.

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

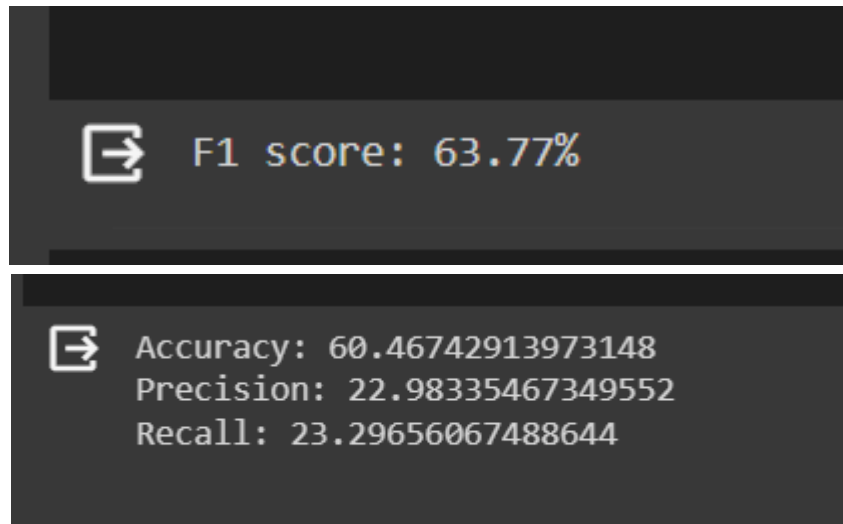


Рис.5. Показники якості класифікації нелінійного класифікатора SVM з сигмоїдальним ядром.

Висновки: за результатами тренування, найкраще завдання класифікації виконує класифікатора SVM з гаусовим ядром. Класифікатор з поліноміальним ядром можливо міг показати кращий результат, якщо збільшити значення degree (ступінь многочлена), проте у нас досить великий набір даних і таке обчислення потребує забагато часу і ресурсів.

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']))
```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

[illegible]

Рис.6. Результат виконання коду для ознайомлення.

KPOK 1

Лістинг програми:

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
         'class']
dataset = read_csv(url, names=names)
# shape
print(dataset.shape)
```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		


```
# Зріз даних head
print(dataset.head(20))
# Статистичні зведення методом describe
print(dataset.describe())
# Розподіл за атрибутом class
print(dataset.groupby('class').size())
```

КРОК 2

Лістинг програми:

```
# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2,2),
sharex=False, sharey=False)
pyplot.show()
# Гістограма розподілу атрибутів датасета
dataset.hist()
pyplot.show()
```

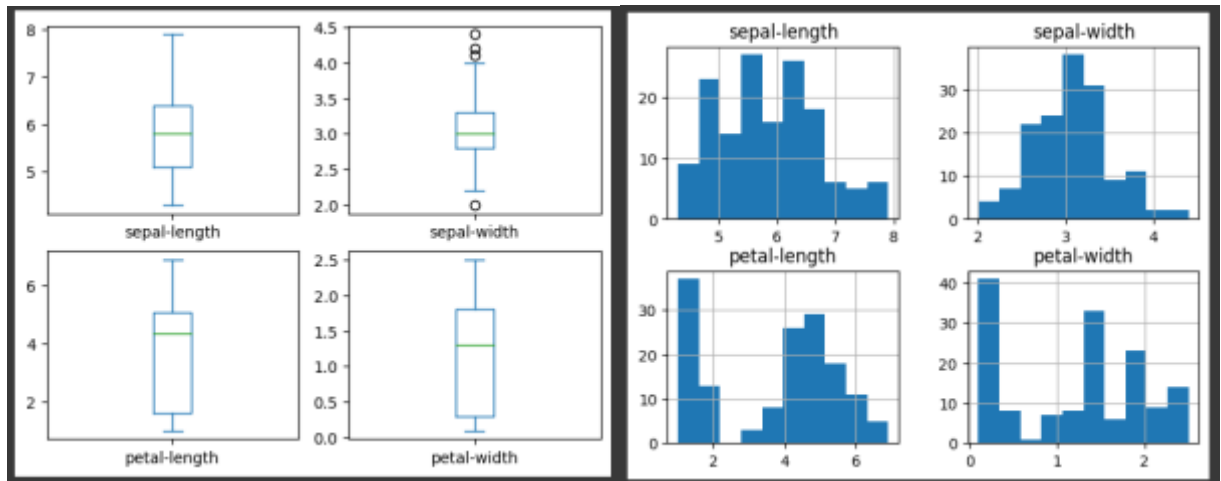


Рис.7. Діаграма розмаху та гістограма розподілу атрибутів датасету.

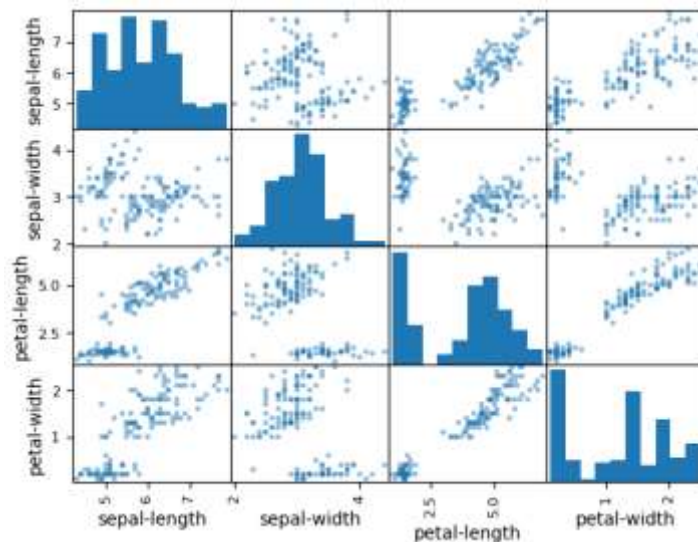


Рис.8. Матриця діаграм розсіювання.

КРОК 3

Лістинг програми:

```
# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
# Вибір перших 4-х стовпців
X = array[:,0:4]
# Вибір 5-го стовпця
Y = array[:,4]
# Разделение X и y на обучающую и контрольную выборки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,
test_size=0.20, random_state=1)
```

КРОК 4

Лістинг програми:

```
# Завантажуємо алгоритми моделі
models = []
models.append(('LR', LogisticRegression(solver='liblinear', mul-
ti_class='ovr'))))
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, scor-
ing='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

# Порівняння алгоритмів
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

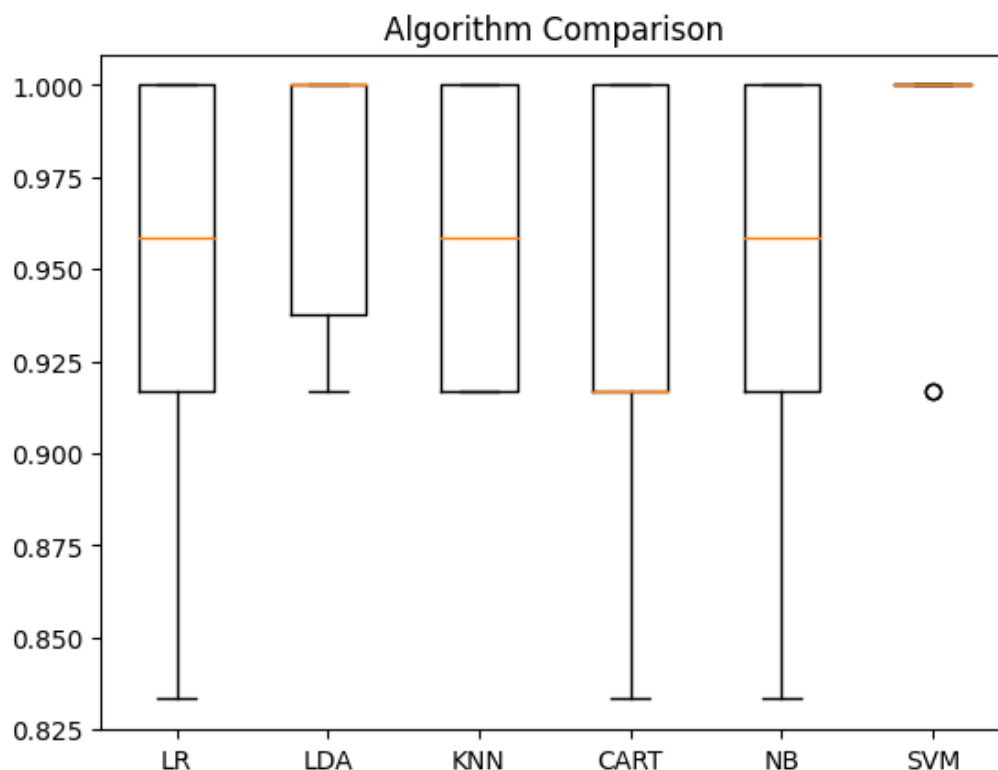


Рис.11. Порівняння алгоритмів.

```

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)

```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

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         0.97         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: За результатами розрахування точності, матриці помилок та звіту про класифікацію, можна стверджувати, що на наборі даних Iris була досягнута висока якість класифікації (точність становить приблизно 96.7). Квітка з 8 кроку 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 = []
```

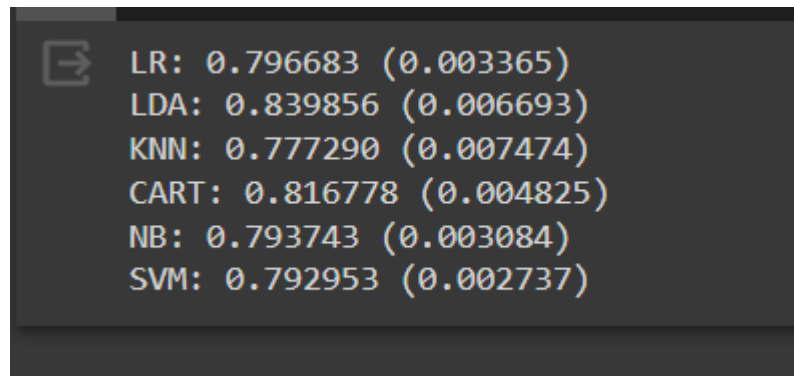
		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

```

models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('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()))

```



```

LR: 0.796683 (0.003365)
LDA: 0.839856 (0.006693)
KNN: 0.777290 (0.007474)
CART: 0.816778 (0.004825)
NB: 0.793743 (0.003084)
SVM: 0.792953 (0.002737)

```

Рис.12. Результат виконання.

Висновки: з результатів видно, що найкращий алгоритм з точки зору середньої точності на тестових наборах даних - це Linear Discriminant Analysis (LDA). Ця модель добре враховує взаємозв'язок між ознаками та цільовою змінною і крім того, стандартне відхилення для LDA також низьке, що свідчить про стабільність моделі. Розрахунок не потребує багато часу та ресурсів.

Завдання 5

Лістинг програми:

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split # Додали імпорт
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 # Додали імпорт

# 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) # Змінено X_test на 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 Corrccoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n', metrics.classification_report(ypred, ytest))

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

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

```

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

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

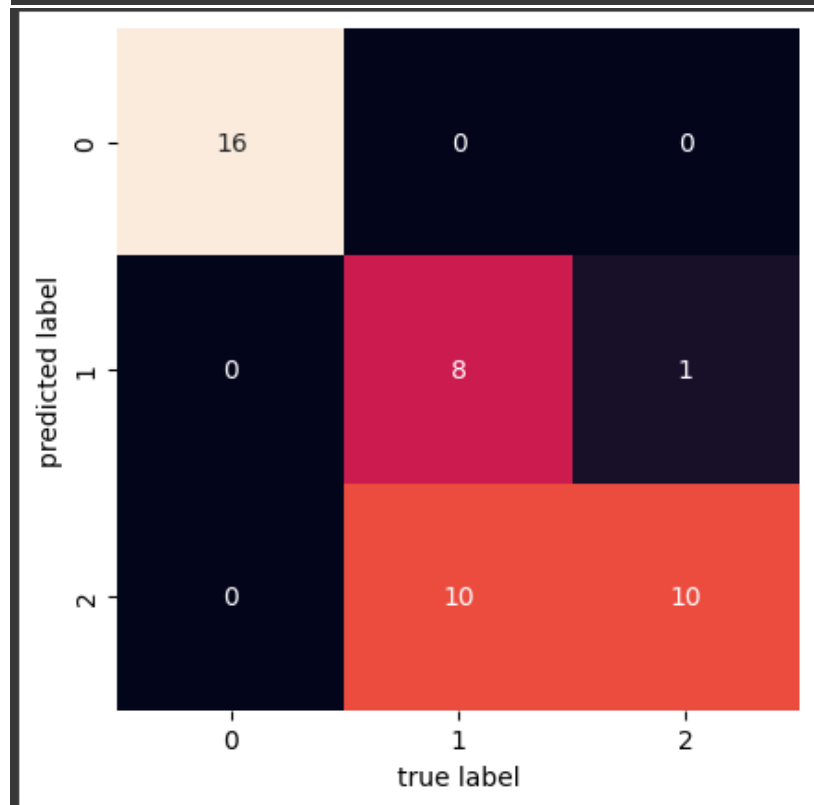


Рис.13. Результат виконання.

Висновки: **Ridge** використовується для класифікації даних в наведеному коді. $\text{tol}=1\text{e-}2$ - це параметр tolerance, який вказує точність обчислення. Він використовується для критерію зупинки оптимізаційного алгоритму. $\text{solver}=\text{"sag"}$ - це параметр, що вказує метод оптимізації для Ridge-класифікатора. "sag" означає Stochastic Average Gradient Descent.

Показники якості. Ассурасу: Визначає, частку точних передбачень відносно загальної кількості прикладів. Результат 75%. Precision: Вимірює точність позитивних передбачень. Результат 83%. Recall: Вимірює здатність моделі виявляти всі позитивні приклади. Результат 75%. F1 Score: Гармонічне середнє точності і повноти. Результат 75%.

Confusion.jpg представляє з себе матрицю плутанини. По вертикалі (вісь y) розташовані передбачені класи. По горизонталі (вісь x) розташовані фактичні класи. Діагональ матриці відображає кількість правильних класифікацій для кожного класу. Тобто 16 екземплярів були правильно класифіковані як перший клас, 8 як другий та 10 як третій. Елементи поза діагоналлю вказують на кількість неправильних класифікацій. Тобто 1 екземпляр неправильно класифіковано як дрийгий та 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 - повна протилежність. У нашому 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 вона більше враховує дисбаланс класів у вибірці. Зазвичай використовується для бінарної класифікації. У нашому випадку, значення Метьюза дорівнює 0.6831, що вказує на добрий рівень узгодженості між фактичними та передбаченими класами.

		Дяченко В.В.			ДУ «Житомирська політехніка».22.122.4.000 – Лр2	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		