

## 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.

Сахно Андрій ПЗ-21-1

<https://github.com/andreylion06/artificial-intelligence>

#### Завдання 2.1. Класифікація за допомогою машин опорних векторів (SVM)

Ознака	Опис	Вид
age	Вік респондента	Числовий
workclass	Тип зайнятості	Категоріальний
fnlwgt	Вага осіб у вибірці	Числовий
education	Освітній рівень респондента	Категоріальний
education-num	Кількість років освіти	Числовий
marital-status	Сімейний стан	Категоріальний
occupation	Професія	Категоріальний
relationship	Відношення до родини	Категоріальний
race	Расова група	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

income	Дохід респондента (>50К або <=50К)	Категоріальний
--------	---------------------------------------	----------------

```

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
from sklearn.model_selection import cross_val_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
        if 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:
        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)

# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
# Навчання класифікатора
classifier.fit(X, y)

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

```

```

classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# Обчислення F-метри для SVM-класифікатора
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([item])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)

# Використання класифікатора для кодованої точки даних та виведення
результату
predicted_class = classifier.predict(input_data_encoded.reshape(1,-1))
print(label_encoder[-1].inverse_transform(predicted_class)[0])

# Обчислити значення інших показників якості класифікації
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")

precision_values = cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")

recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")

```

## Результат виконання:

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab2\LR_2_task_1.py"
F1 score: 76.01%
<=50K
Accuracy: 79.66%
Precision: 78.88%
Recall: 79.66%

Process finished with exit code 0

```

Можу зробити висновок, що тестова точка належить до класу <=50K.

## Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

З поліноміальним ядром:

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

# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 10000 # Зменшую кількість даних для тестування

with open('income_data.txt', 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# 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:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(float) # Тепер використовуємо float, щоб було
легше нормалізувати
y = X_encoded[:, -1].astype(int)

# Нормалізація даних
scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)

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

classifier = SVC(kernel='poly', degree=3)
classifier.fit(X_train, y_train)
```

```

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision_values = cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=3)
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=3)

print(f"SVM with poly kernel:")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
print("Recall: " + str(round(100 * recall_values.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([item])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)
input_data_encoded = scaler.transform(input_data_encoded.reshape(1, -1)) #
Нормалізуємо дані тестової точки

# Використання класифікатора для передбачення класу
predicted_class = classifier.predict(input_data_encoded)
print("Class:", label_encoder[-1].inverse_transform(predicted_class)[0])

```

## Результат виконання:

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\va-un\7th semester\CWI\projs\lab2\LR_2_task_2_1.py"
SVM with poly kernel:
F1 score: 79.33%
Accuracy: 79.34%
Precision: 79.35%
Recall: 79.34%
Class: <=50K
Process finished with exit code 0

```

## З гаусовим ядром:

```

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

# Читання даних
X = []
y = []

```

```

count_class1 = 0
count_class2 = 0
max_datapoints = 10000 # Зменшую кількість даних для тестування

with open('income_data.txt', 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# 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:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(float) # Тепер використовуємо float, щоб було
легше нормалізувати
y = X_encoded[:, -1].astype(int)

# Нормалізація даних
scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)

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

classifier = SVC(kernel='rbf')
classifier.fit(X_train, y_train)

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision_values = cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=3)
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=3)

print(f"SVM with rbf kernel:")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
print("Recall: " + str(round(100 * recall_values.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([item])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)
input_data_encoded = scaler.transform(input_data_encoded.reshape(1, -1)) #
Нормалізуємо дані тестової точки

# Використання класифікатора для передбачення класу
predicted_class = classifier.predict(input_data_encoded)
print("Class:", label_encoder[-1].inverse_transform(predicted_class)[0])

```

## Результат виконання:

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab2\LR_2_task_2_2.py"
SVM with rbf kernel:
F1 score: 80.91%
Accuracy: 80.86%
Precision: 81.07%
Recall: 80.86%
Class: <=50K

Process finished with exit code 0

```

## З сигмоїдальним ядром:

```

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

# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 10000 # Зменшую кількість даних для тестування

with open('income_data.txt', 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)

```

```

        count_class2 += 1

# 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:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(float) # Тепер використовуємо float, щоб було
легше нормалізувати
y = X_encoded[:, -1].astype(int)

# Нормалізація даних
scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)

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

classifier = SVC(kernel='sigmoid')
classifier.fit(X_train, y_train)

f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision_values = cross_val_score(classifier, X, y,
scoring='precision_weighted', cv=3)
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=3)

print(f"SVM with sigmoid kernel:")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
print("Recall: " + str(round(100 * recall_values.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([item])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)
input_data_encoded = scaler.transform(input_data_encoded.reshape(1, -1)) #
Нормалізуємо дані тестової точки

```



```
# Використання класифікатора для передбачення класу
predicted_class = classifier.predict(input_data_encoded)
print("Class:", label_encoder[-1].inverse_transform(predicted_class)[0])
```

## Результат виконання:

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab2\LR_2_task_2_3.py"
SVM with sigmoid kernel:
F1 score: 68.57%
Accuracy: 68.6%
Precision: 68.56%
Recall: 68.6%
Class: <=50K
Process finished with exit code 0
```

Для економії часу під часу тестування - я зменшив кількість точок даних.

Протестувавши виконання коду з різними видами SMV видно, що нелінійний класифікатор з гаусовим ядром показує найкращі результати, якісні показники є найвищими.

## Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

### КРОК 1. ЗАВАНТАЖЕННЯ ТА ВИВЧЕННЯ ДАНИХ

```
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("Форма масиву data: {}".format(iris_dataset['data'].shape))

print("Перші п'ять прикладів:\n{}".format(iris_dataset['data'][:5]))

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

### Результат виконання:

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\proj\lab2\LR_2_task_3_1.py"
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
-----

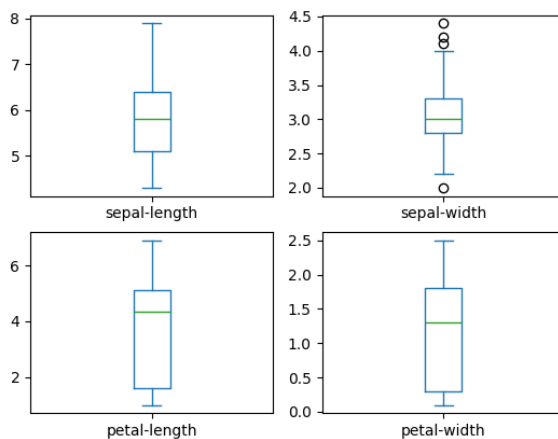
**Data Set Characteristics:**

: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)
Перші п'ять прикладів:
[[5.1 3.5 1.4 0.2]
 [4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5. 3.6 1.4 0.2]]
```

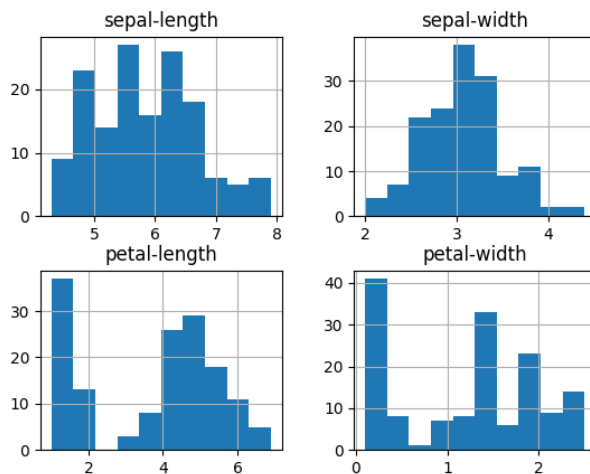
[illegible]

## КРОК 2. ВІЗУАЛІЗАЦІЯ ДАНИХ

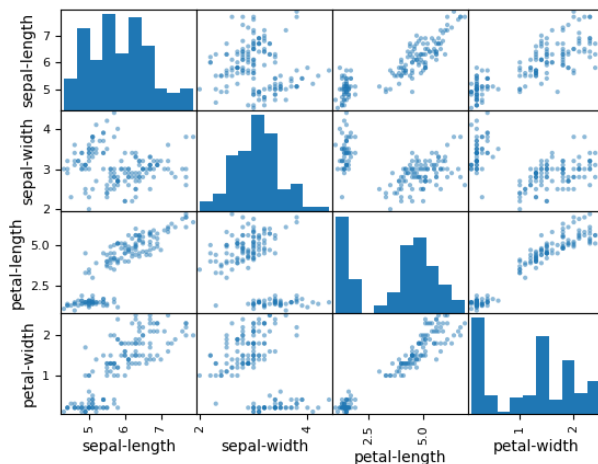
Діаграма розмаху:



Гістограма розподілу атрибутів датасету:



Матриця діаграм розсіювання:



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

```

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)

# Зріз даних head
print(dataset.head(20))

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

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

# 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()

#Матриця діаграм розсіювання
scatter_matrix(dataset)
pyplot.show()

```

### КРОК 3. СТВОРЕННЯ НАВЧАЛЬНОГО ТА ТЕСТОВОГО НАБОРІВ

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\va-un\7th semester\СМІ\projs\lab2\LR_2_task_3_2.py"
(150, 5)

```

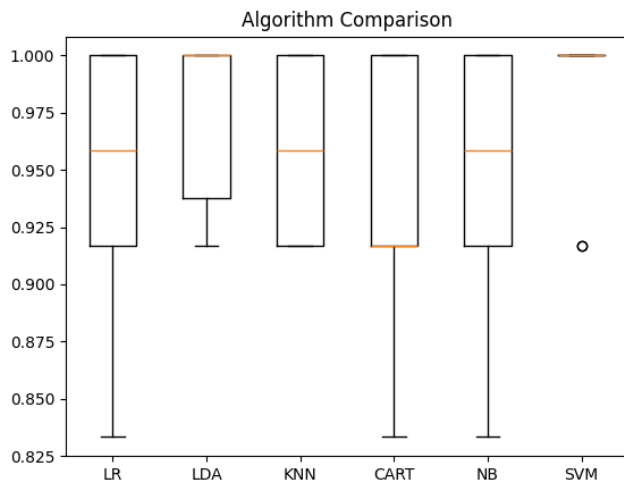
	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	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

```

## КРОК 4. КЛАСИФІКАЦІЯ (ПОБУДОВА МОДЕЛІ)



```
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)
```

Метод опорних векторів (SVM) демонструє високу стабільність результатів, з мінімальною варіацією, хоча є один винятковий випадок поза межами основного діапазону. За результатами тренування він показав найбільшу точність – 98,333%.

## КРОК 5. ОПТИМІЗАЦІЯ ПАРАМЕТРІВ МОДЕЛІ

## КРОК 6. ОТРИМАННЯ ПРОГНОЗУ (ПЕРЕДБАЧЕННЯ НА ТРЕНУВАЛЬНОМУ НАБОРІ)

## КРОК 7. ОЦІНКА ЯКОСТІ МОДЕЛІ

```
Accuracy: 0.9666666666666667
Confusion Matrix:
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]

Classification Report:
              precision    recall  f1-score   support

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

   accuracy          0.97
   macro avg          0.95
   weighted avg          0.97
```

Звіт по класифікації показує, що прогноз буде дуже точним і вірогідність похибки мала.

## КРОК 8. ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ)

```
Форма масиву X_new: (1, 4)
Прогноз: Iris-setosa
```

З прогнозу, квітка з класу Iris-setosa.

## Завдання 2.4. Порівняння якості класифікаторів для набору даних

### завдання 2.1

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_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.model_selection import StratifiedKFold
from sklearn.multiclass import OneVsRestClassifier

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

with open('income_data.txt', 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# 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:
        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)

# Поділ даних на навчальну і тестову вибірку
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=5)

# Підвантажую 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()))

# 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)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

```

### Результат виконання:

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab2\LR_2_task_4.py"
LR: 0.791993 (0.005400)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.807286 (0.007008)
NB: 0.789133 (0.006934)
SVM: 0.788512 (0.002538)

Process finished with exit code 0

```

Результати показують, що LDA має найвищу середню точність серед усіх моделей, хоча різниця з CART та LR невелика. На основі отриманих результатів Linear Discriminant Analysis (LDA) обрано найкращою моделлю для задачі класифікації.

### Завдання 2.5. Класифікація даних лінійним класифікатором Ridge.

```

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

iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)

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

```



```

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

mat = metrics.confusion_matrix(y_test, y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True label')
plt.ylabel('Predicted label')

plt.savefig("Confusion.jpg")
f = BytesIO()
plt.savefig(f, format="svg")

plt.show()

```

## Результат виконання:

```

C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab2\LR_2_task_4_1.py"
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoeff: 0.6831

Classification Report:

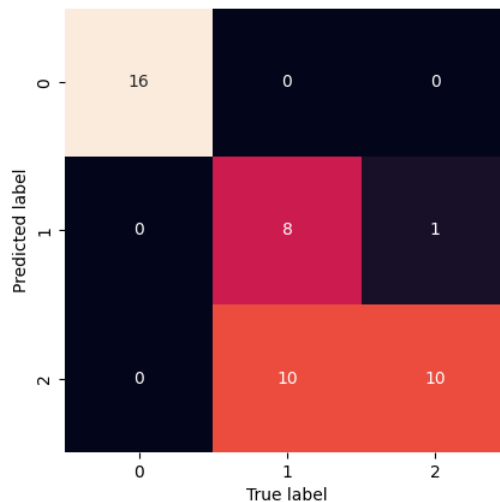
```

	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

```

Process finished with exit code 0

```



### Налаштування RidgeClassifier

- `tol=1e-2`: допустиме значення для перевірки збіжності алгоритму. Якщо зміна втрат менша за це значення, ітерації припиняються.
- `solver="sag"`: оптимізатор SAG (Stochastic Average Gradient), який швидко працює на великих наборах даних.

### Показники якості

- Accuracy: частка правильно передбачених класів — 0.9778.
- Precision: точність передбачення кожного класу, враховуючи хибнопозитивні передбачення — 0.978.
- Recall: відсоток правильно передбачених позитивних прикладів — 0.978.
- F1 Score: середнє значення Precision і Recall, враховуючи їх баланс — 0.978.
- Cohen Kappa Score: показує узгодженість передбачень з істинними мітками, враховуючи випадкову згоду — 0.9667.
- Matthews Corrccoef: кореляція між передбаченнями і справжніми мітками, враховуючи усі типи помилок — 0.9661.

### Коефіцієнт Коена Каппа

- Вимірює рівень узгодженості між передбаченнями та реальними класами з урахуванням випадкової угоди.
- Значення від -1 (повна невідповідність) до 1 (повна відповідність).
- У цьому випадку 0.9667 показує майже ідеальну узгодженість.

### Коефіцієнт Метьюза

- Вимірює кореляцію між передбаченнями та істинними мітками для багатокласових даних.
- Значення від -1 до 1: 1 означає ідеальну класифікацію, 0 — випадковий розподіл, -1 — повна невідповідність.
- Значення 0.9661 свідчить про високу якість класифікації.

**Висновок:** На лабораторній роботі, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив різні методи класифікації даних та навчився їх порівнювати.