

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.

Хід роботи:

Завдання 2.1 Класифікація за допомогою машин опорних векторів (SVM)

Назва	Призначення	Вид
age	Вік	числовий
workclass	Робітничий клас	категоріальний
fnlwgt	Фінальна вага (показує, скільки людей є схожими на конкретного респондента з таким набором характеристик)	числовий
education	Освіта	категоріальний
education-num	Рівень освіти	числовий
marital-status	Сімейний стан	категоріальний
occupation	Професія	категоріальний
relationship	Відносини	категоріальний
race	Раса	категоріальний
sex	Стать	числовий
capital-gain	Приріст капіталу	числовий
capital-loss	Втрата капіталу	числовий
hours-per-week	Кількість робочих годин в тиждень	числовий
native-country	Рідна країна	категоріальний
income	Прибуток в рік	числовий

					ДУ «Житомирська політехніка».24.121.07.000 – Лр2						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Волков О.М.			Звіт з лабораторної роботи	Лім.		Арк.	Аркушів		
Перевір.		Іванов Д.А.						1	27		
Керівник						ФІКТ Гр. ІПЗ-21-5[2]					
Н. контр.											
Зав. каф.											

Лістинг коду:

```
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 recall_score, precision_score, accuracy_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)
```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Обчислення F-міри 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:
        try:
            input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        except ValueError as e:
            print(f"Значення '{item}' не було знайдено у label_encoder. Перевірте дані.")
            print(e)

        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])

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X_test)

#Акуратність
accuracy_percentage = accuracy_score(y_test, y_pred) * 100;
print("Accuracy = ", round(accuracy_percentage,2), "%")

#Повнота
recall_percentage = recall_score(y_test, y_pred, average='weighted') * 100
print("Recall: ", round(recall_percentage,2), "%")

#Точність
precision_percentage = precision_score(y_test, y_pred, average='weighted') * 100
print("Precision: ", round(precision_percentage,2), "%")

```

Результат виконання:

Рисунок 1 - Результат виконання програми

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестова точка була класифікована як клас " $\leq 50K$ ", що означає, що модель вважає, що особа, описана у тестовій точці, має дохід не більше \$50K.

Метрики:

F1 score: 76.01% — це свідчить про збалансовану продуктивність моделі в класифікації як позитивних, так і негативних прикладів.

Accuracy (Акуратність): 79.56% — модель правильно класифікує близько 80% прикладів.

Recall (Повнота): 79.56% — це означає, що модель успішно виявляє майже 80% всіх прикладів з будь-яким доходом.

Precision (Точність): 79.26% — коли модель прогнозує, що людина заробляє менше ніж \$50K, вона буде правильною в майже 80% випадків.

Висновок: Тестова точка класифікована до класу " $\leq 50K$ ", тобто модель передбачає, що ця особа має дохід не більше \$50K.

Завдання 2.2 Порівняння якості класифікаторів SVM з нелінійними ядрами

1) Поліноміальне ядро

Лістинг коду:

```
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, cross_val_score
from sklearn.metrics import recall_score, precision_score, accuracy_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)
```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=5)

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, max_iter=6000))
classifier.fit(X_train, y_train)

# Обчислення F-міри 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:
        try:
            input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        except ValueError as e:
            print(f"Значення '{item}' не було знайдено у label_encoder. Перевірте дані.")
            print(e)

        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])

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X_test)

```

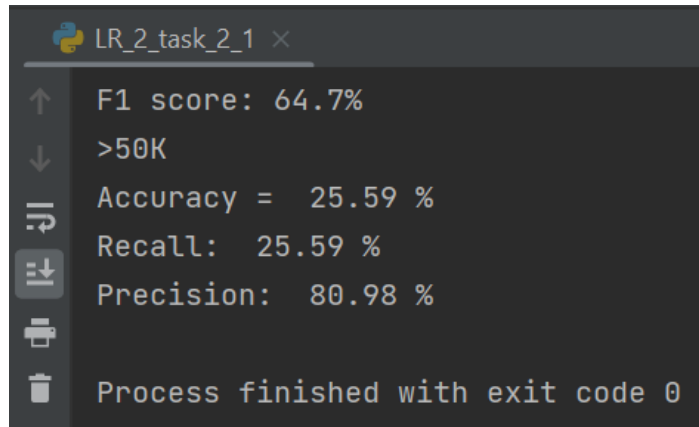
		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
#Акуратність
accuracy_percentage = accuracy_score(y_test, y_pred) * 100;
print("Accuracy = ", round(accuracy_percentage,2), "%")

#Повнота
recall_percentage = recall_score(y_test, y_pred, average='weighted') * 100
print("Recall: ", round(recall_percentage,2), "%")

#Точність
precision_percentage = precision_score(y_test, y_pred, average='weighted') * 100
print("Precision: ", round(precision_percentage,2), "%")
```

Результат виконання:



```
LR_2_task_2_1 ×
↑ F1 score: 64.7%
↓ >50K
⇌ Accuracy = 25.59 %
⇌ Recall: 25.59 %
⇌ Precision: 80.98 %
🖨️
🗑️ Process finished with exit code 0
```

Рисунок 2 - Результат виконання програми

2) Гаусове ядро

Лістинг коду:

```
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, cross_val_score
from sklearn.metrics import recall_score, precision_score, accuracy_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
```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    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)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=5)

classifier = OneVsOneClassifier(SVC(kernel='rbf', max_iter=6000))
classifier.fit(X_train, y_train)

# Обчислення F-міри 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:
        try:
            input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        except ValueError as e:
            print(f"Значення '{item}' не було знайдено у label_encoder. Перевірте дані.")
            print(e)

        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])

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X_test)

#Акуратність
accuracy_percentage = accuracy_score(y_test, y_pred) * 100;
print("Accuracy = ", round(accuracy_percentage,2), "%")

```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```
#Повнота
recall_percentage = recall_score(y_test, y_pred, average='weighted') * 100
print("Recall: ", round(recall_percentage,2), "%")

#Точність
precision_percentage = precision_score(y_test, y_pred, average='weighted') * 100
print("Precision: ", round(precision_percentage,2), "%")
```

Результат виконання:

```
LR_2_task_2_2 ×
↑ F1 score: 71.95%
↓ <=50K
↻ Accuracy = 78.22 %
↻ Recall: 78.22 %
↻ Precision: 82.84 %
🖨️
🗑️ Process finished with exit code 0
```

Рисунок 3 - Результат виконання програми

3) Сигмоїдне ядро

Лістинг коду:

```
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, cross_val_score
from sklearn.metrics import recall_score, precision_score, accuracy_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
```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		


```

# 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)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=5)

classifier = OneVsOneClassifier(SVC(kernel='sigmoid', max_iter=6000))
classifier.fit(X_train, y_train)

# Обчислення F-міри 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:
        try:
            input_data_encoded[i] =
int(label_encoder[count].transform([input_data[i]])[0])
        except ValueError as e:
            print(f"Значення '{item}' не було знайдено у label_encoder. Перевірте дані.")
            print(e)

        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])

# Прогнозування значень для тренувальних даних
y_pred = classifier.predict(X_test)

#Акуратність
accuracy_percentage = accuracy_score(y_test, y_pred) * 100;
print("Accuracy = ", round(accuracy_percentage,2), "%")

#Повнота
recall_percentage = recall_score(y_test, y_pred, average='weighted') * 100
print("Recall: ", round(recall_percentage,2), "%")

```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				9
Змн.	Арк.	№ док.ум.	Підпис	Дата		

```
#Точність
precision_percentage = precision_score(y_test, y_pred, average='weighted') * 100
print("Precision: ", round(precision_percentage,2), "%")
```

Результат виконання:

Рисунок 4 - Результат виконання програми

За результатами тренування SVM з Гаусовим ядром найкраще виконує завдання класифікації за результатами тренування.

Метрики:

- Поліноміальне ядро:

F1 score: 64.7% — це свідчить про середній рівень збалансованості моделі у класифікації позитивних прикладів (більше ніж \$50K) з певними недоліками в класифікації негативних прикладів.

Accuracy (Акуратність): 25.59% — модель правильно класифікує лише близько 26% прикладів, що свідчить про загальний низький рівень продуктивності.

Recall (Повнота): 25.59% — це означає, що модель успішно виявляє лише близько 26% всіх прикладів з доходом більше ніж \$50K.

Precision (Точність): 80.98% — коли модель прогнозує, що людина заробляє більше ніж \$50K, вона буде правильною в майже 81% випадків, що свідчить про високу точність при позитивних передбаченнях.

- Гаусове ядро

F1 score: 76.01% — це свідчить про збалансовану продуктивність моделі в класифікації як позитивних, так і негативних прикладів.

Accuracy (Акуратність): 79.56% — модель правильно класифікує близько 80% прикладів.

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Recall (Повнота): 79.56% — це означає, що модель успішно виявляє майже 80% всіх прикладів з будь-яким доходом.

Precision (Точність): 79.26% — коли модель прогнозує, що людина заробляє менше ніж \$50К, вона буде правильною в майже 80% випадків.

- Сигмоїдне ядро

F1 score: 63.77% — це свідчить про середній рівень збалансованості моделі у класифікації як позитивних, так і негативних прикладів.

Accuracy (Акуратність): 60.47% — модель правильно класифікує близько 60% прикладів.

Recall (Повнота): 60.47% — це означає, що модель успішно виявляє близько 60% всіх прикладів, незалежно від класу.

Precision (Точність): 60.64% — коли модель прогнозує, що людина заробляє менше ніж \$50К, вона буде правильною в майже 61% випадків.

Завдання 2.3 Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

КРОК 1

Лістинг коду:

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

print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))
print("Опис ", 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))

for i in range(5):
    print(iris_dataset['data'][i])

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

Результат виконання:

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Project
LR_2_task_3
"D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-2\Scripts\python.exe" "D:\ЖДТУ\1 семестр\Системи штучного
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
Опис .. _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)
[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]
Тип масиву target: <class 'numpy.ndarray'>

```

Рисунок 5 - Результат виконання програми

КРОК 2

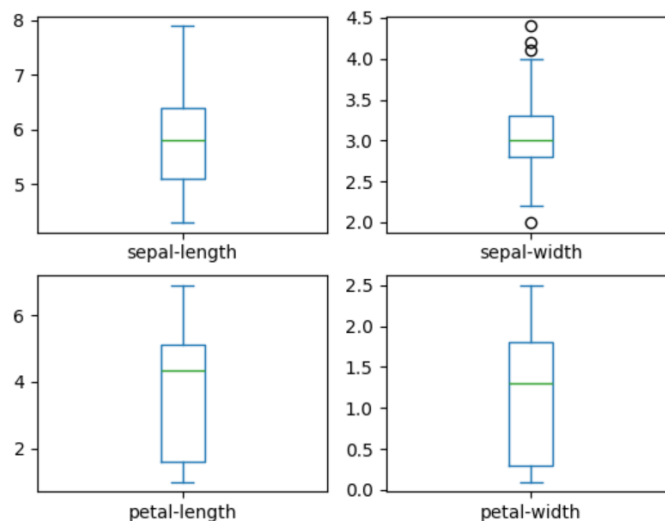


Рисунок 6 – Діаграма розмаху

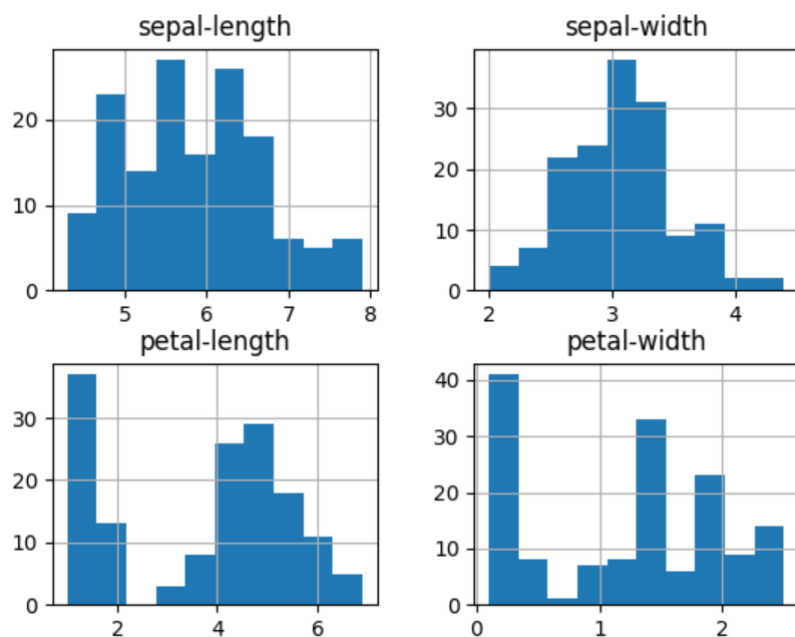


Рисунок 7 – Гістограма розподілу атрибутів датасета

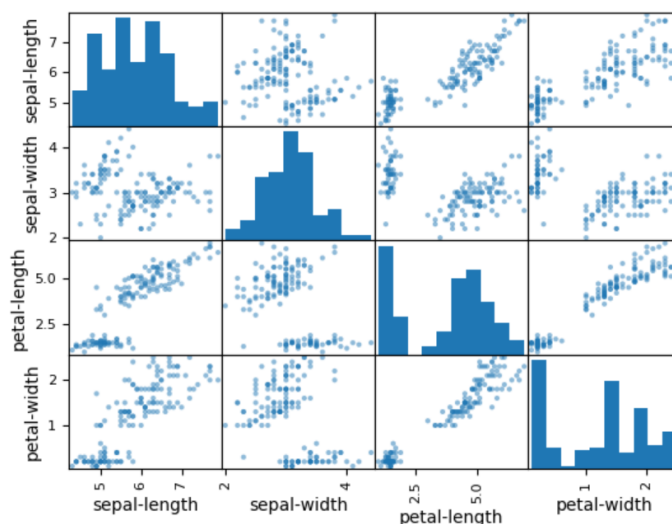


Рисунок 8 – Матриця діаграм розсіювання

Лістинг коду:

```
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
```

```

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())

# КРОК 2
# 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 4

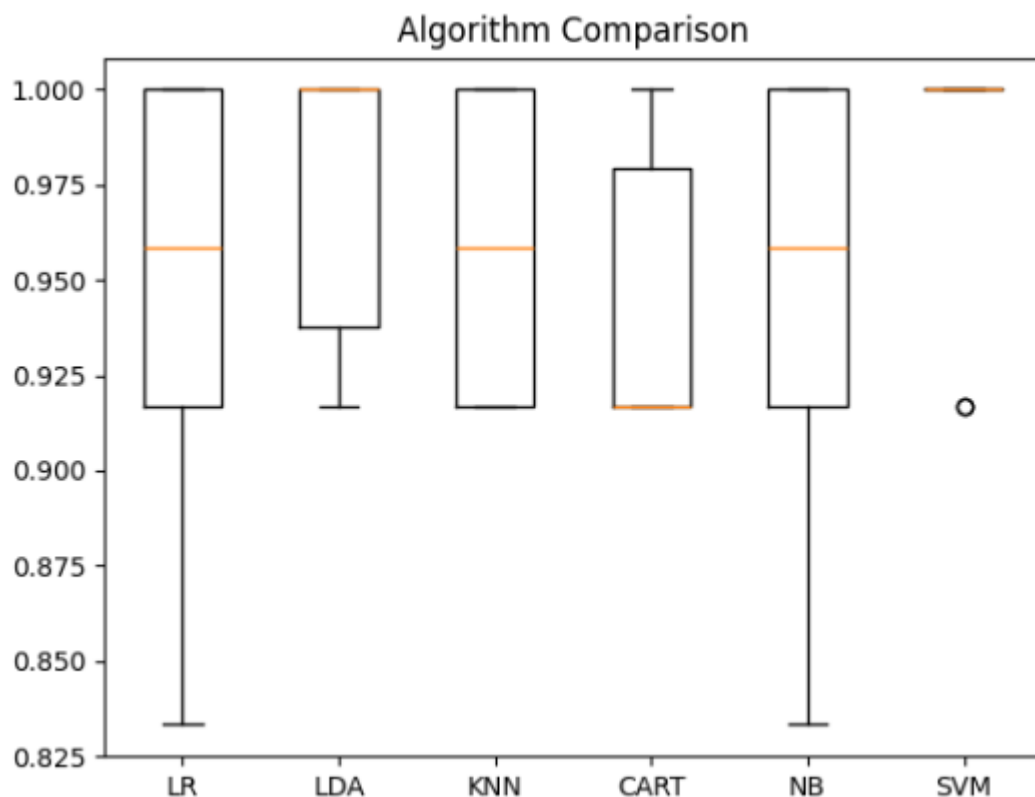


Рисунок 9 – Графік порівняння алгоритмів

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

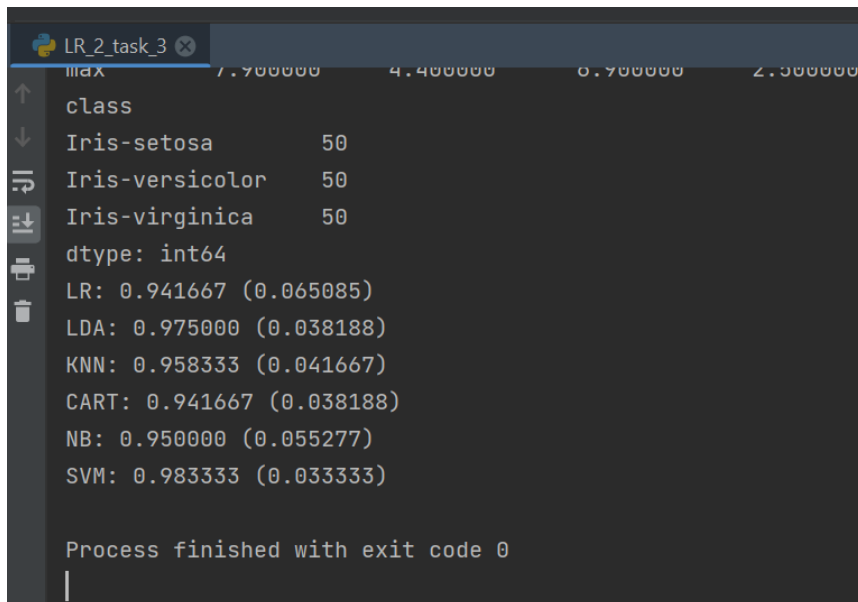


Рисунок 10 – Графік порівняння алгоритмів

Було проведено порівняння декількох методів класифікації на датасеті Iris, і результат показав наступні середні значення точності:

Logistic Regression (LR): 94.17%

Linear Discriminant Analysis (LDA): 97.50%

K-Nearest Neighbors (KNN): 95.83%

Decision Tree (CART): 94.17%

Naive Bayes (NB): 95.00%

Support Vector Machine (SVM): 98.33%

Найкращим методом класифікації я вважаю SVM, тому що він показав найвищу точність серед усіх випробуваних моделей — 98.33%. Це свідчить про його здатність добре розділяти класи на основі особливостей у датасеті.

КРОК 6-7

Лістинг коду:

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

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

```
Run: LR_2_task_3 x
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        30
 weighted avg           0.97        30

Process finished with exit code 0
```

Рисунок 10 – Оцінка якості моделі

КРОК 8

Лістинг коду:

```
# КРОК 8
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масиву X_new: {}".format(X_new.shape))
# Отримуємо прогноз для нового зразка
prediction = model.predict(X_new)
# Виводимо результат прогнозу
print("Прогноз: {}".format(prediction))
```

```
LR_2_task_3 x
Форма масиву X_new: (1, 4)
Прогноз: Iris-setosa

Process finished with exit code 0
```

Рисунок 11 – Оцінка якості моделі

Моя модель передбачила клас квітів Iris-setosa.

Загальний лістинг коду:

```
# from sklearn.datasets import load_iris
# iris_dataset = load_iris()
#
# print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))
#
# print("Опис ",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))
```



```

# for i in range(5):
#     print(iris_dataset['data'][i])
#
# print("Тип масиву target: {}".format(type(iris_dataset['target'])))

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

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())

# КРОК 2
# 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
# 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]

# Розділення X и y на навчающую и контрольную выборки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# КРОК 4
# 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('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))

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

# КРОК 6
# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# КРОК 7
# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

# КРОК 8
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масиву X_new: {}".format(X_new.shape))
# Отримуємо прогноз для нового зразка
prediction = model.predict(X_new)
# Виводимо результат прогнозу
print("Прогноз: {}".format(prediction))

```

За результатами тренування, найкращої точності вдалося досягти за допомогою моделі SVM (Support Vector Machine), яка показала середню точність (accuracy) на рівні **98.33%** (0.983333). Інші моделі також продемонстрували високі результати, зокрема LDA (97.5%) та KNN (95.83%).

Для квітки з вимірами чашолистка (5, 2.9) і пелюстки (1, 0.2), модель передбачила клас Iris-setosa. Це відповідає правильному результату, оскільки такі параметри дійсно характерні для цього класу.

Завдання 2.4 Порівняння якості класифікаторів для набору даних завдання 2.1

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг коду:

```

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, precision_score, recall_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
from sklearn import preprocessing
import numpy as np

# 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)

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

```

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# Завантажуємо алгоритми моделей
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
accuracies = []
percisions = []
recalls = []
names = []

for name, model in models:
    # Навчання моделі
    model.fit(X_train, y_train)

    # Отримання прогнозу
    y_pred = model.predict(X_test)

    # Оцінка акуратності моделі
    accuracy = round(accuracy_score(y_test, y_pred) * 100, 2)

    # Оцінка точності моделі
    precision = round(precision_score(y_test, y_pred, average='weighted')*100,
2)

    # Оцінка повноти моделі
    recall = round(recall_score(y_test, y_pred, average='weighted') * 100, 2)

    # Збереження результатів
    accuracies.append(accuracy)
    percisions.append(precision)
    recalls.append(recall)
    names.append(name)

    print('Accuracy of %s model -> %s, percision -> %s, recall -> %s' % (name,
accuracy, precision, recall))

# Вибір найкращого алгоритму
best_model_index = np.argmax(accuracies)
best_model_name = names[best_model_index]
best_model_accuracy = accuracies[best_model_index]
model_percision = percisions[best_model_index]
model_recall = recalls[best_model_index]

print('\nThe best algoritm is %s with accuracy -> %s, percision -> %s, recall ->
%s' % (best_model_name, best_model_accuracy, model_percision, model_recall))

```

Результат виконання:

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_2_task_2_3 x LR_2_task_4 x
"D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-2\Scripts\python.exe" "D:\ЖДТУ\1 се
Accuracy of LR model -> 78.55, percision -> 76.88, recall -> 78.55
Accuracy of LDA model -> 81.12, percision -> 79.96, recall -> 81.12
Accuracy of KNN model -> 76.78, percision -> 74.31, recall -> 76.78
Accuracy of CART model -> 80.57, percision -> 80.84, recall -> 80.57
Accuracy of NB model -> 78.95, percision -> 77.43, recall -> 78.95
Accuracy of SVM model -> 74.41, percision -> 62.74, recall -> 74.41

The best algoitm is LDA with accuracy -> 81.12, percision -> 79.96, recall -> 81.12

Process finished with exit code 0

```

Рисунок 12 – Оцінка якості моделі

LDA модель:

- Точність (81.12%) — найвища серед усіх моделей.
- Precision (79.96%) та Recall (81.12%) також на високому рівні, що свідчить про збалансовану продуктивність моделі. Модель LDA має високу точність передбачення і добре розпізнає всі класи.

CART модель:

- Точність (80.57%) майже на рівні з LDA.
- Precision (80.84%) навіть вища, ніж у LDA, що означає, що ця модель робить менше помилкових позитивних передбачень.
- Recall (80.57%) також на високому рівні, але трохи нижча за LDA.

LR модель:

- Точність (78.55%), Precision (76.88%) і Recall (78.55%) показують, що ця модель добре працює, але поступається LDA і CART.

KNN модель:

- Точність (76.78%), Precision (74.31%) і Recall (76.78%) є нижчими порівняно з іншими моделями, що свідчить про те, що ця модель не є найкращим варіантом для цього набору даних.

NB модель:

- Точність (78.95%), Precision (77.43%) і Recall (78.95%) показують, що модель є досить збалансованою, але все ж поступається LDA і CART.

SVM модель:

- Точність (74.41%) і Precision (62.74%) є найнижчими серед усіх моделей. Це свідчить про те, що SVM на цьому наборі даних працює гірше порівняно з іншими методами.

Найкраща модель для вирішення задачі: LDA. Ця модель має найвищу загальну точність (81.12%) та хороші показники precision і recall. Вона забезпечує збалансовану продуктивність і робить мало помилок як при передбаченні позитивних класів, так і при розпізнаванні всіх класів.

Завдання 2.5 Класифікація даних лінійним класифікатором Ridge

Лістинг коду:

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from io import BytesIO #needed for plot
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt

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)
clf = RidgeClassifier(tol = 1e-2, solver = "sag")
clf.fit(X_train,y_train)
y_pred = clf.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 Corrcoef:',
np.round(metrics.matthews_corrcoef(y_test,y_pred),4))
print('\t\tClassification Report:\n',
metrics.classification_report(y_pred,y_test))

mat = 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")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format = "svg")
```

Результат виконання:

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR_2_task_2_3 x LR_2_task_5 x
"D:\ЖДТУ\1 семестр\Системи штучного інтелекту\Lab-2\Script
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               0.44      0.89      0.59         9
    2               0.91      0.50      0.65        20

   accuracy               0.76        45
  macro avg               0.78        45
 weighted avg              0.85        45

Process finished with exit code 0

```

Рисунок 13 – Результат виконання програми

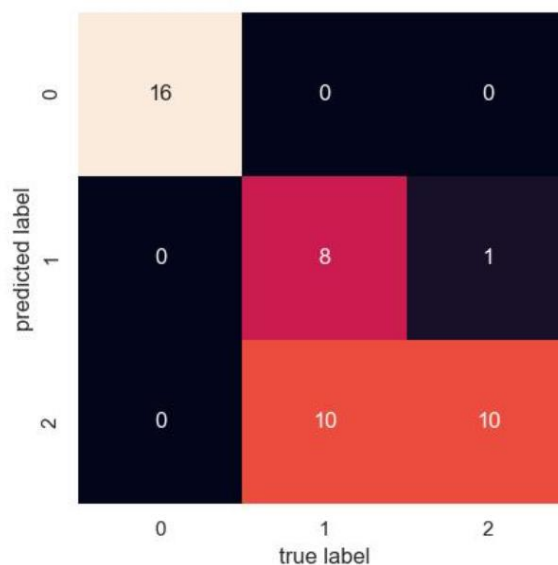


Рисунок 14 – Результат виконання програми (Confusion.jpg)

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

1 $\text{tol}=1\text{e-}2$:

- Це параметр "tolerance" (толерантність).
- Він визначає допустимий рівень похибки для зупинки алгоритму. Коли зміни в функції вартості під час ітерацій стають меншими за це значення, алгоритм зупиняється.

- У даному випадку $1e-2$ (або 0.01) означає, що якщо зміна в результаті оптимізації стає менше ніж 0.01, ітерації завершаться.

2 solver="sag":

- Це параметр, який визначає алгоритм оптимізації для навчання моделі.
- "sag" означає Stochastic Average Gradient. Це метод градієнтного спуску, який добре підходить для великих датасетів, оскільки він є стохастичним і оновлює градієнт на кожній ітерації, використовуючи середнє значення попередніх ітерацій.
- Він прискорює процес навчання порівняно з традиційними методами для великих вибірок.

Опишіть які показники якості використовуються та їх отримані результати.

Вставте у звіт та поясніть зображення Confusion.jpg

Основні показники:

1. Accuracy (Акуратність):

- Це частка правильно передбачених класів серед усіх передбачень.
- **Результат:** 0.7556 (або 75.56%)
- Це означає, що 75.56% усіх передбачень моделі були правильними.

2. Precision (Точність передбачення):

- Частка правильних позитивних передбачень серед усіх передбачених позитивних прикладів.
- **Результат:** 0.8333 (або 83.33%)
- Це означає, що з усіх прикладів, які модель класифікувала як позитивні, 83.33% були правильними.

3. Recall (Чутливість або повнота):

- Частка правильних позитивних передбачень серед усіх реальних позитивних прикладів.
- **Результат:** 0.7556 (або 75.56%)
- Це означає, що з усіх реальних позитивних прикладів, модель правильно класифікувала 75.56%.

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

4. F1 Score:

- Гармонійне середнє між precision і recall. F1 Score важливий, коли потрібно знайти баланс між точністю та чутливістю.
- **Результат:** 0.7503 (або 75.03%)
- Це показує, наскільки добре модель знаходить баланс між точністю і чутливістю.

5. Classification Report (Звіт про класифікацію):

Цей звіт містить докладну інформацію про precision, recall і F1-score для кожного з класів:

- **Клас 0:**
 - **Precision:** 1.00 (ідеальна точність для цього класу)
 - **Recall:** 1.00 (усі реальні приклади класу 0 були правильно класифіковані)
 - **F1-score:** 1.00 (ідеальний баланс між precision і recall)
- **Клас 1:**
 - **Precision:** 0.44 (модель часто помилялася при передбаченні класу 1)
 - **Recall:** 0.89 (але модель виявляла майже всі реальні приклади класу 1)
 - **F1-score:** 0.59 (низький через дисбаланс між precision і recall)
- **Клас 2:**
 - **Precision:** 0.91 (модель зробила мало помилок при передбаченні класу 2)
 - **Recall:** 0.50 (лише половина реальних прикладів класу 2 були виявлені)
 - **F1-score:** 0.65 (середній показник через низький recall)
- **Macro avg:**
 - Середнє значення для precision, recall і F1-score для кожного класу.
 - **Precision:** 0.78, **Recall:** 0.80, **F1-score:** 0.75.
- **Weighted avg:**
 - Зважене середнє precision, recall і F1-score з урахуванням кількості

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

прикладів у кожному класі.

- **Precision:** 0.85, **Recall:** 0.76, **F1-score:** 0.76.

Пояснення зображення Confusion.jpg

Матриця на малюнку – це матриця плутанини. Вона використовується для оцінки ефективності класифікаційної моделі та дає можливість зрозуміти, наскільки добре модель справляється з передбаченням кожного класу.

- **Стовпці (true label):** Це реальні значення класів, які модель мала передбачити.
- **Рядки (predicted label):** Це значення класів, які модель передбачила.
- **Числа в клітинках:** Кількість прикладів, що належать до певної категорії (реальний клас), які були передбачені певною категорією (передбачений клас).

Пояснення матриці:

- Верхній лівий кут: 16 — це кількість випадків, коли клас "0" був правильно передбачений як "0".
- У центральній частині: 8 — це кількість випадків, коли клас "1" був правильно передбачений як "1", але 10 випадків класу "1" були помилково класифіковані як "2".
- У нижній частині: для класу "2" модель передбачила правильно 10 разів, але зробила помилку ще в 10 випадках, класифікуючи їх як клас "1".

Правильні передбачення знаходяться на діагоналі матриці (клітинки: [0,0], [1,1], [2,2]).

Помилки знаходяться поза діагоналлю. Наприклад, модель часто плутала клас "1" з класом "2" (10 випадків).

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

1. Cohen Kappa Score:

- Оцінка узгодженості між передбаченнями моделі та реальними значеннями, з урахуванням випадкових збігів.
- **Результат:** 0.6431 (або 64.31%)

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

- Це середній рівень узгодженості між прогнозами моделі та реальними класами.

2. Matthews Corrcoef:

- Коефіцієнт кореляції Метьюза вимірює зв'язок між передбачуваними та фактичними класами і є більш інформативним для задач з незбалансованими класами.
- **Результат:** 0.6831 (або 68.31%)
- Це свідчить про помірну кореляцію між передбаченнями моделі та реальними класами.

Висновок: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.

		Волков О.М.			ДУ «Житомирська політехніка».24.121.07.000 – Лр2	Арк.
		Іванов Д.А.				27
Змн.	Арк.	№ докум.	Підпис	Дата		