

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

Лістинг програми:

```
import numpy as np
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
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning

simplefilter("ignore", category=ConvergenceWarning)

# 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():
```

					ДУ «Житомирська політехніка».23.121.16.000			
					– Пн 1			
Змн.	Арк.	№ докум.	Підпис	Дата	Звіт з лабораторної роботи	Літ.	Арк.	Аркушів
Розроб.		Нагорний В.В.						
Перевір.		Іванов Д.А.					1	20
Керівник						ФІКТ Гр. ІПЗ-20-4		
Н. контр.								
Зав. каф.								

```

        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-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:
        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 кодової точки даних # та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```
"D:\4Course\без гита\Lab2\venv\Sc
F1 score: 56.15%
<=50K
Accuracy: 62.64%
Precision: 75.88%
Recall: 62.64%
```

Рис. 1. Результати виконання програми

Ознаки з набору даних:

Вік (числова), робочий клас (категоріальна), fnlwgt – вага вибірки (числова), освіта (категоріальна), education-num – найвищий рівень освіти (числова), сімейний стан (категоріальна), сфера роботи (категоріальна), взаємовідносини (категоріальна), раса (категоріальна), стать (категоріальна), приріст капіталу (числова), збиток капіталу (числова), годин на тиждень (числова), рідна країна (категоріальна),

Тестова точка належить до класу “<=50K”.

**Завдання №2: Порівняння якості класифікаторів SVM з нелінійними ядрами.**

Лістинг програми:

Поліноміальне ядро:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning

simplefilter("ignore", category=ConvergenceWarning)

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

# Читання даних
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 1000

with open(input_file, 'r') as f:
    for line in f.readlines():
```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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(SVC(kernel='poly', degree=8))

# 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(SVC(kernel='poly', degree=8))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# 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:
        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)

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```
print(label_encoder[-1].inverse_transform(predicted_class)[0])
num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy',
cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted',
cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
```

```
"D:\4Course\без гита\Lab2
F1 score: 36.67%
<=50K
Accuracy: 51.35%
Precision: 69.52%
Recall: 51.35%
```

Рис. 2. Результати виконання програми (Поліноміальне ядро для 1000 точок)

Кількість точок для даного алгоритму було зменшено до тисячі для отримання хоча б якогось результату, бо даний алгоритм є дуже вимогливим до апаратного забезпечення. Зрозуміло, що у разі зменшення кількості точок, впадуть і показники метрик.

Лістинг програми:

Гаусове ядро:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning

simplefilter("ignore", category=ConvergenceWarning)

# 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
```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        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(SVC(kernel='rbf'))

# 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(SVC(kernel='rbf'))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# 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:
        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 кодової точки даних # та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])

num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy',

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

```
cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted',
cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted',
cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
```

```
"D:\4Course\без гита\Lab2\venv\Scripts\
F1 score: 71.95%
<=50K
Accuracy: 78.61%
Precision: 83.06%
Recall: 78.61%
```

Рис. 3. Результати виконання програми (Гаусове ядро)

Лістинг програми:

Сигмоїдальне ядро:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from warnings import simplefilter
from sklearn.exceptions import ConvergenceWarning

simplefilter("ignore", category=ConvergenceWarning)

# 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)
```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

# 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(SVC(kernel='sigmoid'))

# 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(SVC(kernel='sigmoid'))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# 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:
        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])
num_folds = 3
accuracy_values = cross_val_score(classifier, X, y, scoring='accuracy', cv=num_folds)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=num_folds)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=num_folds)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```
"D:\4Course\без гита\Lab2\ven
F1 score: 71.95%
<=50K
Accuracy: 78.61%
Precision: 83.06%
Recall: 78.61%
```

Рис. 4. Результати виконання програми (Сигмоїдальне ядро)

Згідно отриманих результатів тренувань гаусове ядро найкраще виконує завдання класифікації для даного завдання. Можливо поліноміальне ядро продемонструвало б кращі результати для 25000 точок, але швидкодія даного алгоритму не дає змоги перевірити це на практиці.

### Завдання №3: Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

Лістинг програми:

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()
print("Ключі iris_dataset: \n{}".format(iris_dataset.keys()))
print(iris_dataset['DESCR'][:193] + "\n...")
print("Назви відповідей: {}".format(iris_dataset['target_names']))
print("Назва ознак: \n{}".format(iris_dataset['feature_names']))
print("Тип масиву data: {}".format(type(iris_dataset['data'])))
print("Форма масиву data: {}".format(iris_dataset['data'].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

[illegible]

Рис. 5. Результати виконання програми (Завантаження та вивчення даних)

### Лістинг програми:

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

```
(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
```

Рис. 6. Результати виконання програми (Завантаження та вивчення даних)

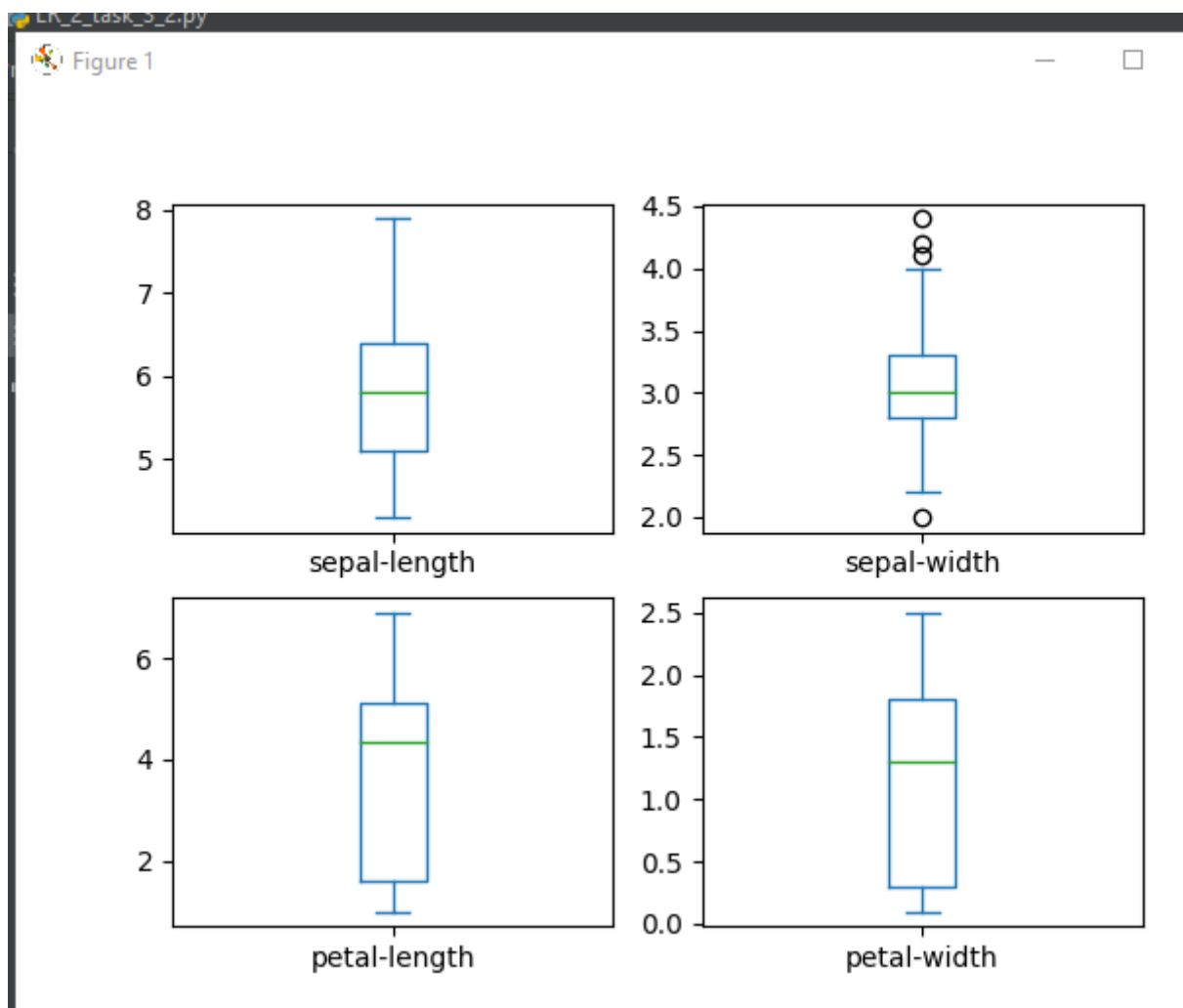


Рис. 7. Результати виконання програми (Одновимірні графіки)

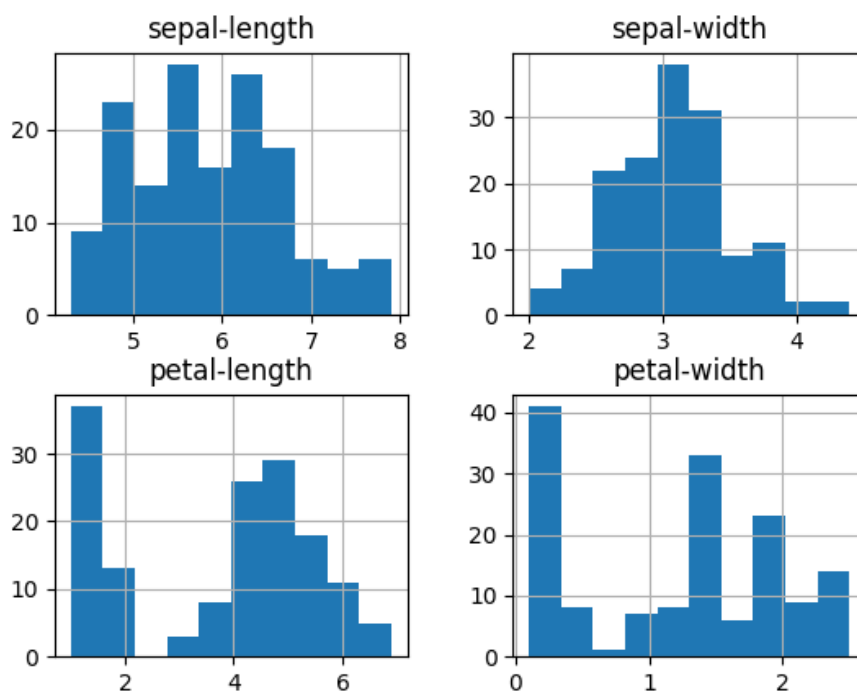


Рис. 8. Результати виконання програми  
(Діаграма розмаху атрибутів вхідних даних)

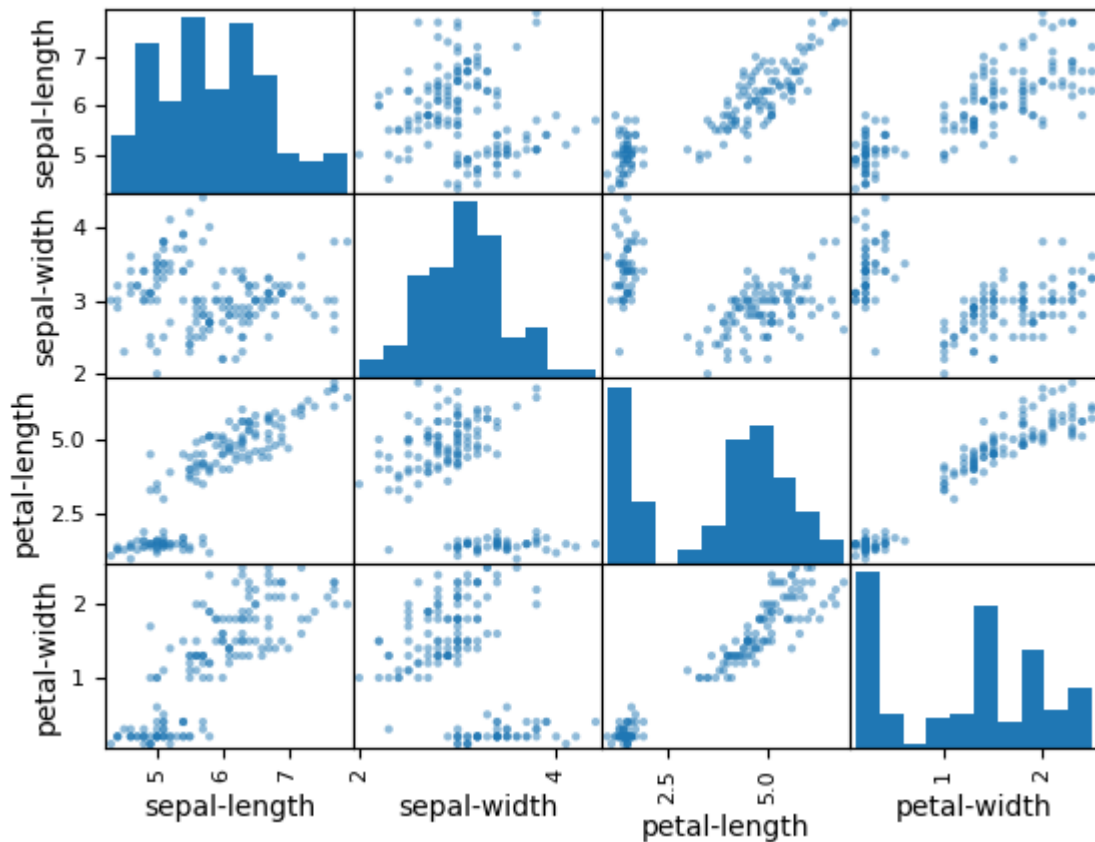


Рис. 9. Результати виконання програми (Багатовимірні графіки)

Лістинг програми:

```
# Розділення дасету на навчальну та контрольну вибірки
import numpy as np
from matplotlib import pyplot
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

from LR_2_task_3_2 import dataset

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
```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

```

X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

# Завантажуємо алгоритми моделі
models = []
models.append(('LR', 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='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, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()

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

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

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, Y_train)
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма масива X_new: {}".format(X_new.shape))
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Оцінка тестового набору: {:.2f}".format(knn.score(X_validation, Y_validation)))

```

```

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

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

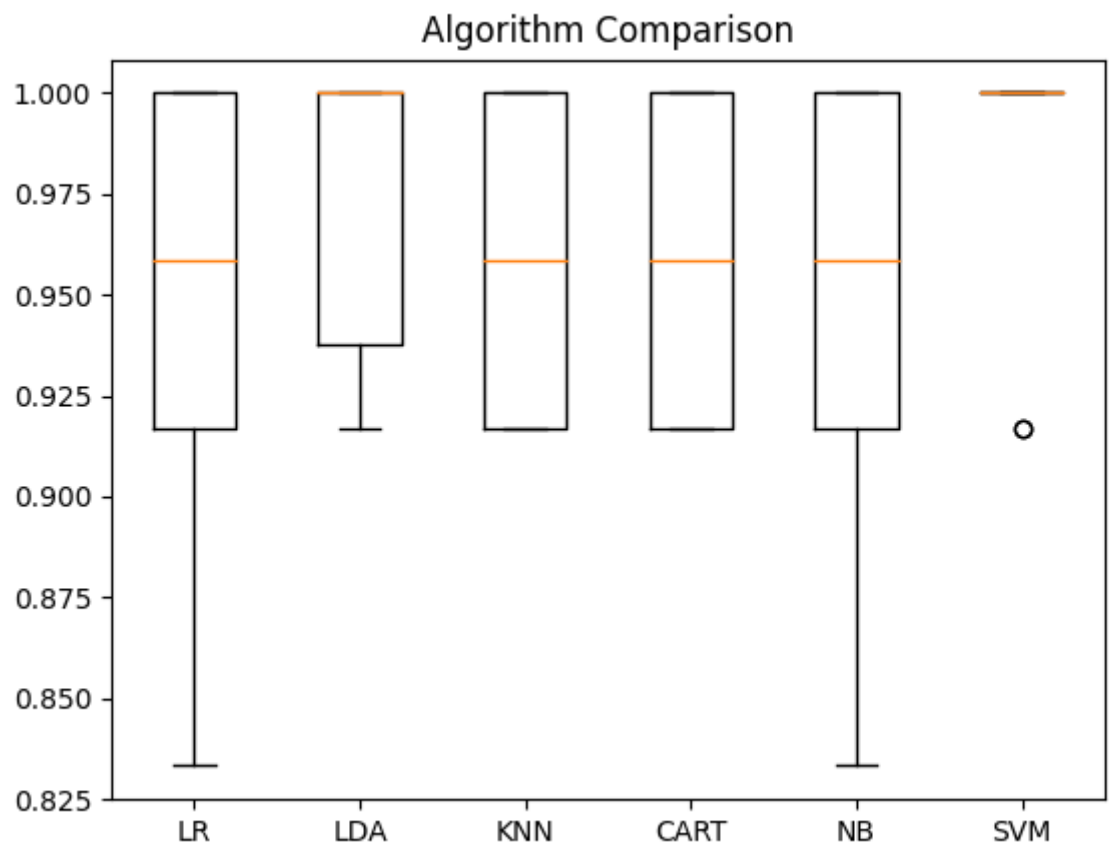


Рис. 10. Результати виконання програми (Порівняння алгоритмів)

```
0.9666666666666667
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.92	0.96	13
Iris-virginica	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Рис. 11. Результати виконання програми (Предбачення на тренувальному наборі)

```

форма массива X_new: (1, 4)
Прогноз: ['Iris-setosa']
Оцінка тестового набору: 1.00

Process finished with exit code 0

```

Рис. 12. Результати виконання програми (Застосування моделі для передбачення)

Метод класифікації SVM – найкращий для рішення цієї задачі, бо метрика асигуру найбільша і стандартне відхилення найменше.

Квітка з кроку 8 належить до класу setosa. Для цієї моделі точність тестового набору становить 1.

#### Завдання №4: Порівняння якості класифікаторів для набору даних завдання 2.1.

Лістинг програми:

```

# Завантаження бібліотек
from pandas import read_csv
import matplotlib
import numpy as np
from sklearn import preprocessing

matplotlib.use('TkAgg')
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.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

# Завантаження датасету
# names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'mar-
ital-status', 'occupation', 'relationship',
#         'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week',
'native-country']
dataset = read_csv('income_data.txt')

input_file = 'income_data.txt'

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

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				16
Змн.	Арк.	№ докум.	Підпис	Дата		



```

with open(input_file, 'r') as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_data-
points:
            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)

# Розделение 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)

# Завантажуємо алгоритми моделі
models = []
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='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()

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LR: 0.793070 (0.006099)
LDA: 0.812176 (0.003802)
KNN: 0.766961 (0.006871)
CART: 0.805006 (0.007407)
NB: 0.789796 (0.004791)
SVM: 0.788677 (0.003972)

```

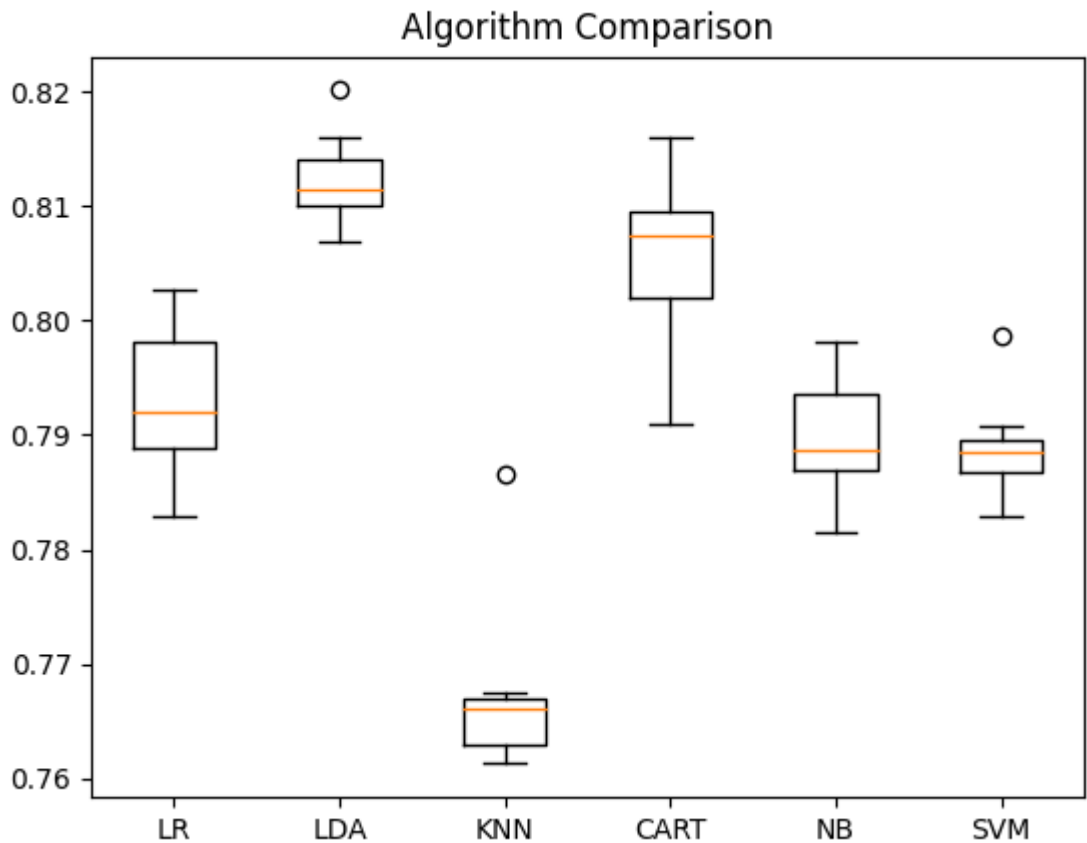


Рис. 13. Результати виконання програми

### Завдання №5: Класифікація даних лінійним класифікатором Ridge.

Лістинг програми:

```

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

iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_state=0)
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)
ypred = clf.predict(Xtest)
print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))

```

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

```

print('Precision:', np.round(metrics.precision_score(ytest, ypred, average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(ytest, ypred, average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4))
print('Cohen Kappa Score:', np.round(metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrcoef:', np.round(metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n', metrics.classification_report(ypred, ytest))

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

```

```

"D:\4Course\без гита\Lab2\venv\Scripts\python.exe" "D:/4Course
Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrcoef: 0.6831
      Classification Report:

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	0.44	0.89	0.59	9
2	0.91	0.50	0.65	20
accuracy			0.76	45
macro avg	0.78	0.80	0.75	45
weighted avg	0.85	0.76	0.76	45

Рис. 14. Результати виконання програми

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

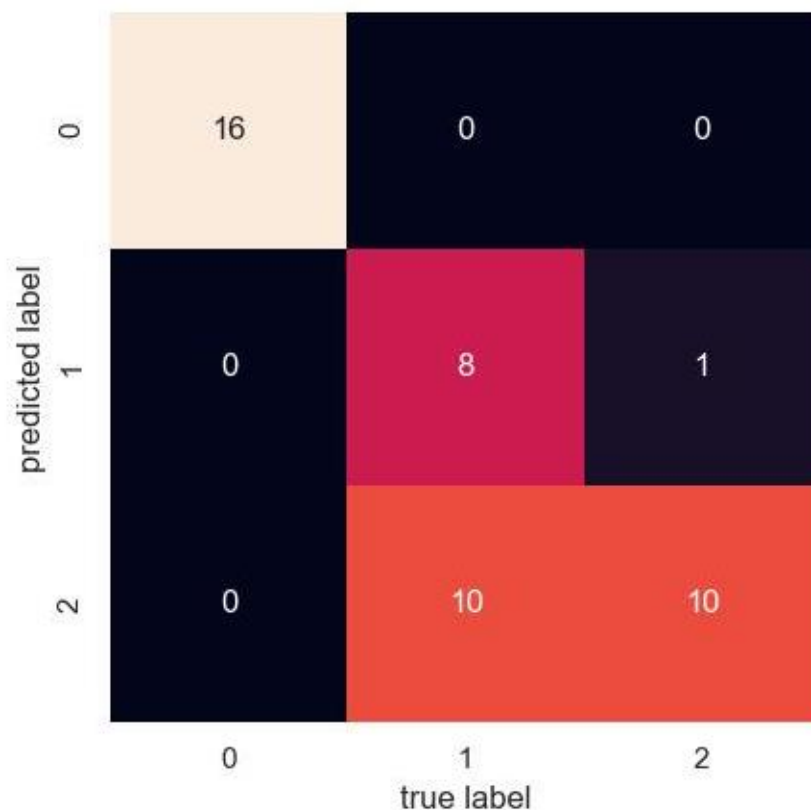


Рис. 15. Зображення Confusion.jpg

**Висновок:** під час виконання завдань лабораторної роботи, було використано спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та навчилися їх порівнювати.

Проект до лабораторної роботи можна переглянути за посиланням: [https://github.com/Xatiko17/AI\\_Labs](https://github.com/Xatiko17/AI_Labs)

		Нагорний В.В.			Житомирська політехніка.23.121.16.000 -	Арк.
		Іванов Д.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		