ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідити різні методи класифікації даних та навчитися їх порівнювати.

Посилання на проект: https://github.com/ipz202-rev/AI-lab2

Хід роботи:

Завдання №2.1. Класифікація за допомогою машин опорних векторів (SVM).

Ознаки з набору даних:

Бінарні – sex (стать), income (дохід, >50K, <=50K)

Числові – age (вік), fnlwgt, education-num (рівень освіти у вигляді числа), capital-gain (здобуток з капіталу), capital-loss (збиток з капіталу), hours-per-week (годин на тиждень)

Категоріальні – workclass (вид зайнятості, приватний підприємець, безробітний, державна служба тощо), education (рівень освіти у текстовому вигляді), marital-status (сімейний стан), occupation (рід діяльності), relationship (статус у відносинах, чоловік, дружина, не в стосунках тощо), race (раса), native-country (країна походження)

Лістинг файлу LR_2_task_1.py:

```
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

input_file = 'income_data.txt'
```

```
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

X = np.array(X)

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)

scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)

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

scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaler.fit_transform(X_train)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

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)) + "%")

f1_values = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100 * f1_values.mean(), 2)) + "%")
```

```
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-
States']

input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]).item())
        count += 1

input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

```
D:\course-4\semester-1\ai\lab2_proj
Accuracy: 81.95%
Precision: 80.94%
Recall: 81.95%
F1 score: 80.13%
>50K

Process finished with exit code 0
```

Рис.2.1.1. Результат виконання коду.

Враховуючи отримані результати, можна зробити висновок, що тестова точка належить до класу >50K.

Завдання №2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

Лістинг файлу LR_2_task_2_1.py:

```
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, random_state=0))
```

```
D:\course-4\semester-1\ai\lab2_proje
Accuracy: 83.75%
Precision: 83.06%
Recall: 83.75%
F1 score: 83.2%
<=50K

Process finished with exit code 0
```

Рис.2.2.1. Результат виконання коду.

```
classifier = OneVsOneClassifier(SVC(kernel='rbf'))
```

```
D:\course-4\semester-1\ai\lab2_proje
Accuracy: 83.96%
Precision: 83.18%
Recall: 83.96%
F1 score: 82.95%
<=50K

Process finished with exit code 0
```

Рис.2.2.2. Результат виконання коду.

```
classifier = OneVsOneClassifier(SVC(kernel='sigmoid'))
```

```
D:\course-4\semester-1\ai\lab2_proj
Accuracy: 57.26%
Precision: 57.1%
Recall: 57.26%
F1 score: 57.18%
<=50K

Process finished with exit code 0
```

Рис.2.2.3. Результат виконання коду.

SVM з поліноміальним та гаусовим ядром дали приблизно однаковий результат, в той час, як із сигмоїдальним показники вийшли відчутно гірші, тому його відразу відкидаємо. Також варто зазначити, що поліноміальне потребувало значно більше часу порівняно з гаусовим. Виходячи з отриманих результатів, можна зробити висновок, що для даної задачі найкраще підходить SVM з гаусовим ядром.

Завдання №2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

```
D:\course-4\semester-
[[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]]
```

Рис.2.3.1. Виведення ознак для перших п’яти прикладів.

Лістинг файлу LR_2_task_3.py (ознайомлення зі структурою даних):

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

print(iris_dataset['data'][:5])

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']))
```

```
D:\course-4\semester-1\ai\lab2_project\venv\Scripts\python.exe D:\course-4\semester-1\ai\lab2_project\LR_2_task_3.py
[[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]]
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Тип масиву target: <class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

Process finished with exit code 0
```

Рис.2.3.2. Результат виконання коду.

Лістинг файлу LR_2_task_3.py (візуалізація даних):

```
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.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

iris_dataset = load_iris()

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))

# Стастичні 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()
```

```
D:\course-4\semester-1\ai\lab2_project\venv\Scripts\python.exe D:\course-
(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

Process finished with exit code 0
```

Рис.2.3.3. Результат виконання коду.

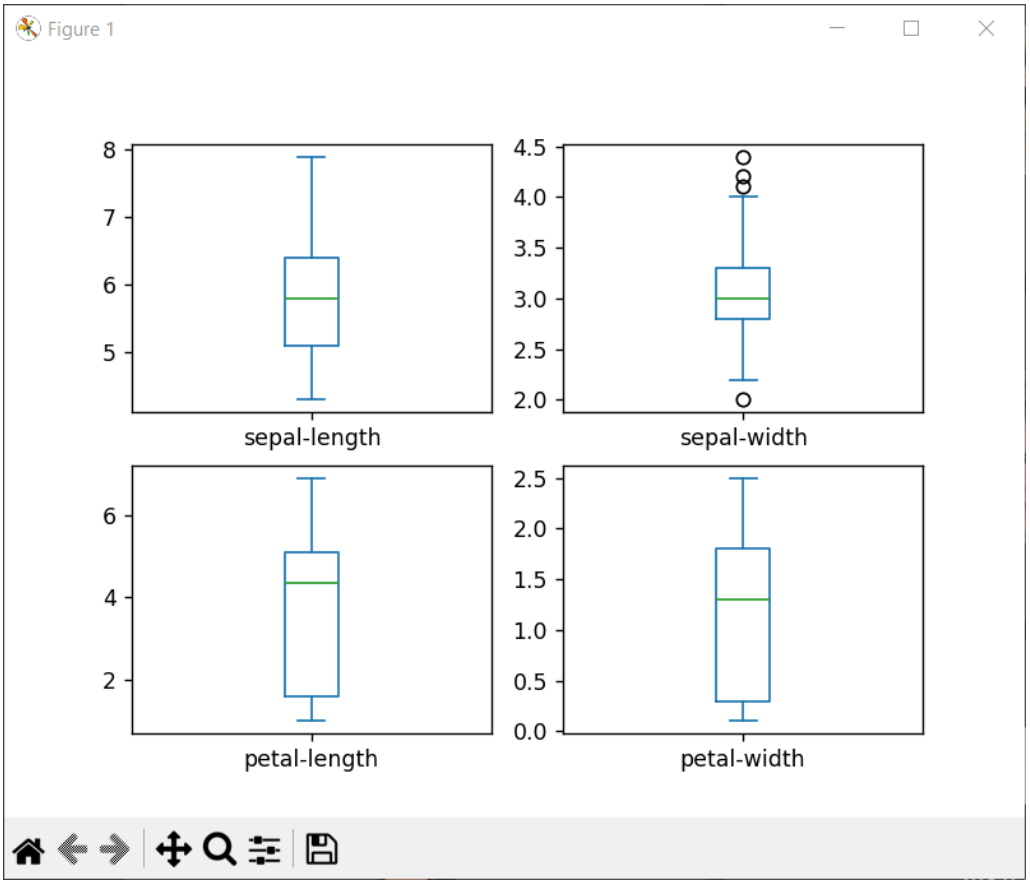

Рис.2.3.4. Діаграма розмаху кожної змінної.

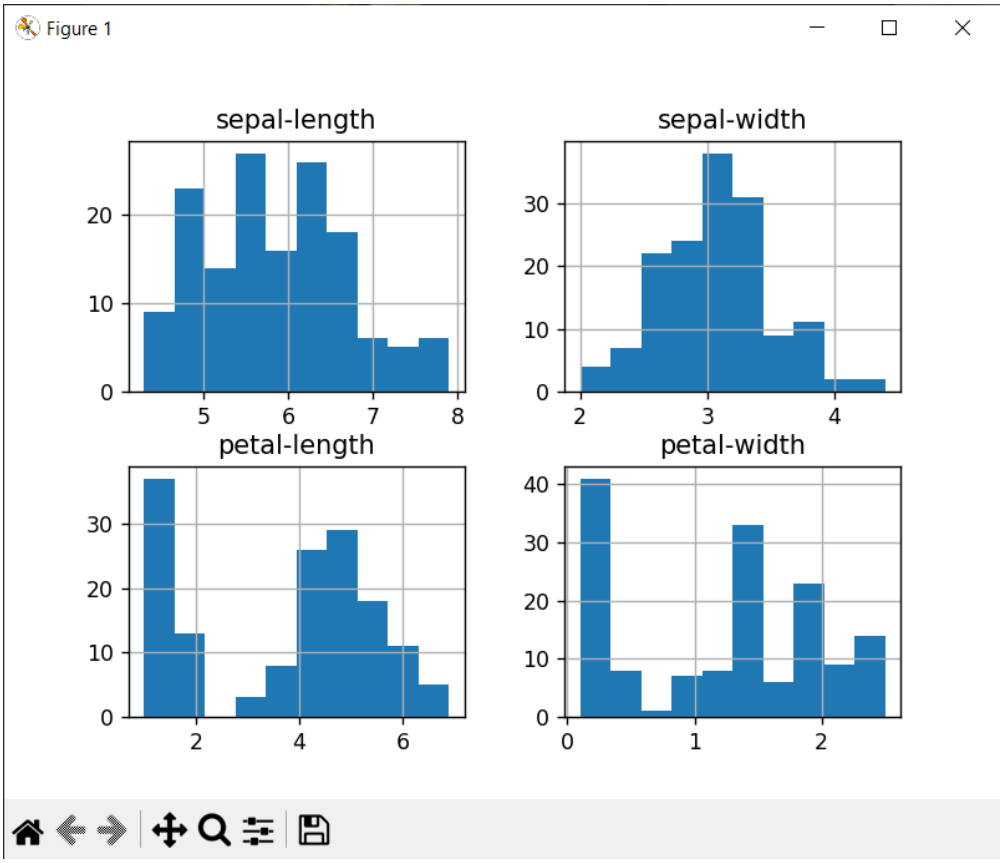

Рис.2.3.5. Гістограма вхідних даних кожної змінної.

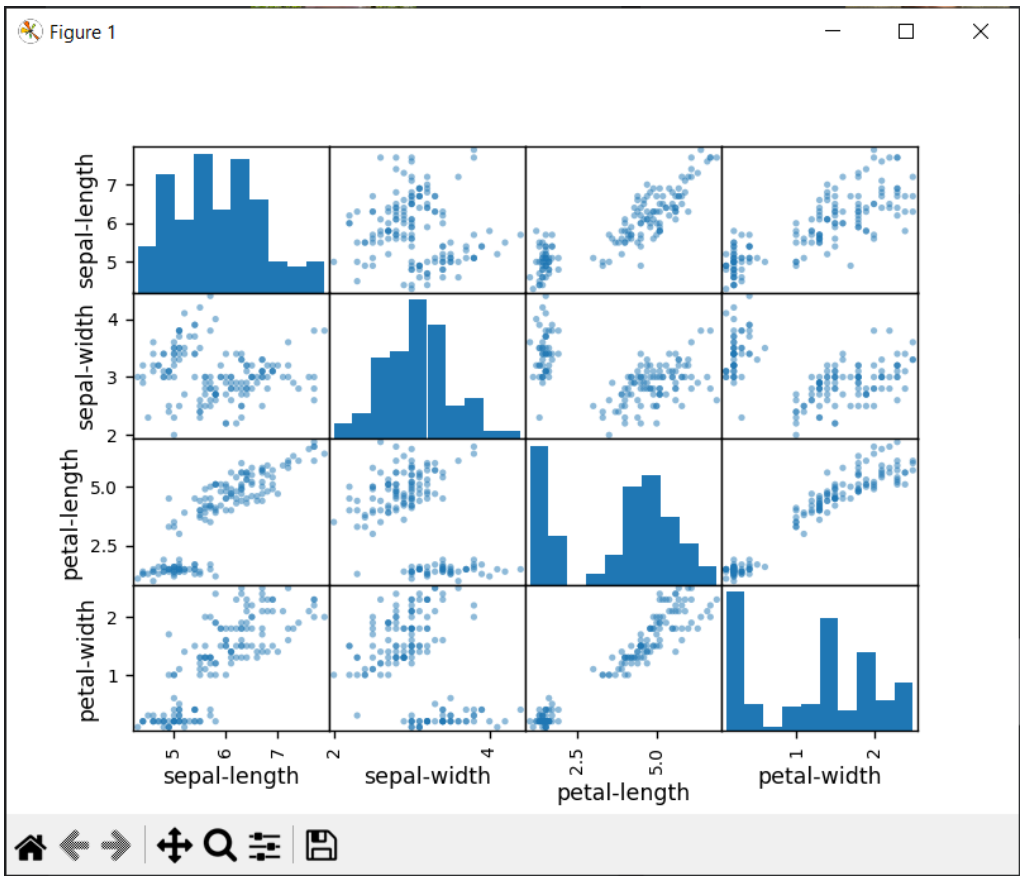

Рис.2.3.6. Матриця діаграм розсіювання.

Лістинг файлу LR_2_task_3.py (створення навчального та тестового наборів і побудова моделі):

```
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.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

iris_dataset = load_iris()

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)

# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
```

```
# Вибір перших 4-х стовпців
X = array[:, 0:4]
# Вибір 5-го стовпця
y = array[:, 4]
# Разделение X и y на обучающую и контрольную выборки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

# 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()
```

```
D:\course-4\semester-1\ai\lab2_proje
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)

Process finished with exit code 0
```

Рис.2.3.7. Результат виконання коду.

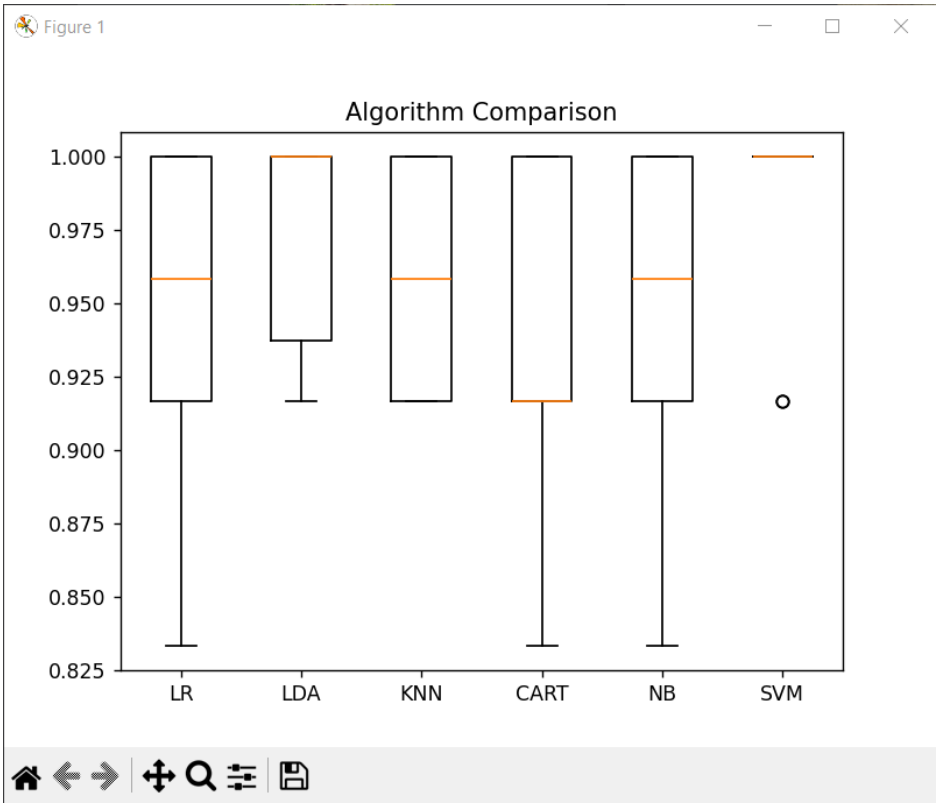

Рис.2.3.8. Діаграма порівняння алгоритмів.

За отриманими результатами можна сказати, що найкращий метод класифікації у даному випадку буде SVM, оскільки він має найвищу оцінку точності (0.983333) та найнижчий показник стандартного відхилення (0.033333). Також можна використати LDA, враховуючи, що його показники теж достатньо високі – 0.975000 і 0.038188 відповідно.

Лістинг файлу LR_2_task_3.py:

```
import numpy as np
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.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

iris_dataset = load_iris()
```

```
print(iris_dataset['data'][:5])

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']))

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))

# Стастичні 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()

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

# 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))

X_new = np.array([[5, 2.9, 1, 0.2]])
print("форма масиву X_new: {}".format(X_new.shape))

prediction = model.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Спрогнозована мітка: {}".format(prediction[0]))
```

```
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

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

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30

форма масиву X_new: (1, 4)
Прогноз: ['Iris-setosa']
Спрогнозована мітка: Iris-setosa

Process finished with exit code 0
```

Рис.2.3.9. Результат виконання коду.

У результаті тренування було досягнуто якість класифікації 97% та визначено, що квітка належить до класу Iris-setosa.

Завдання №2.4. Порівняння якості класифікаторів для набора даних завдання 2.1.

Лістинг файлу LR_2_task_4.py:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import pyplot
from sklearn import preprocessing
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC, SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score,
StratifiedKFold
from sklearn.tree import DecisionTreeClassifier

input_file = 'income_data.txt'

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

X = np.array(X)

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)

scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
```

```
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')))


for name, model in models:
    print(name)

    model.fit(X, y)

    accuracy_values = cross_val_score(model, X, y, scoring='accuracy', cv=3)
    print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")

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

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

    f1_values = cross_val_score(model, X, y, scoring='f1_weighted', cv=3)
    print("F1 score: " + str(round(100 * f1_values.mean(), 2)) + "%")

    input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family',
                  'White', 'Male', '0', '0', '40', 'United-States']

    input_data_encoded = np.array([-1] * len(input_data))
    count = 0
    for i, item in enumerate(input_data):
        if item.isdigit():
            input_data_encoded[i] = item
        else:
            input_data_encoded[i] =
int(label_encoder[count].transform([item]).item())
            count += 1

    input_data_encoded = input_data_encoded.astype(int)
    input_data_encoded = [input_data_encoded]

    predicted_class = model.predict(input_data_encoded)
    print(label_encoder[-1].inverse_transform(predicted_class)[0])

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

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()))

pyplot.boxplot(results, labels=names)
```

```
pyplot.title('Algorithm Comparison')
pyplot.show()
```

```
D:\course-4\semester-1\ai\lab2_projec
LR
Accuracy: 81.82%
Precision: 80.69%
Recall: 81.82%
F1 score: 80.25%
>50K
LDA
Accuracy: 81.14%
Precision: 79.86%
Recall: 81.14%
F1 score: 79.35%
>50K
KNN
Accuracy: 82.16%
Precision: 81.53%
Recall: 82.16%
F1 score: 81.75%
<=50K
CART
Accuracy: 80.58%
Precision: 80.75%
Recall: 80.72%
F1 score: 80.82%
<=50K
```

```
NB
Accuracy: 79.76%
Precision: 78.2%
Recall: 79.76%
F1 score: 77.13%
<=50K
SVM
Accuracy: 82.38%
Precision: 81.51%
Recall: 82.38%
F1 score: 80.6%
>50K
LR: 0.818849 (0.004427)
LDA: 0.812176 (0.003802)
KNN: 0.817606 (0.003760)
CART: 0.804509 (0.005393)
NB: 0.799080 (0.005377)
SVM: 0.824112 (0.005380)
```

Рис.2.4.1 – 2.4.2. Результат виконання коду.

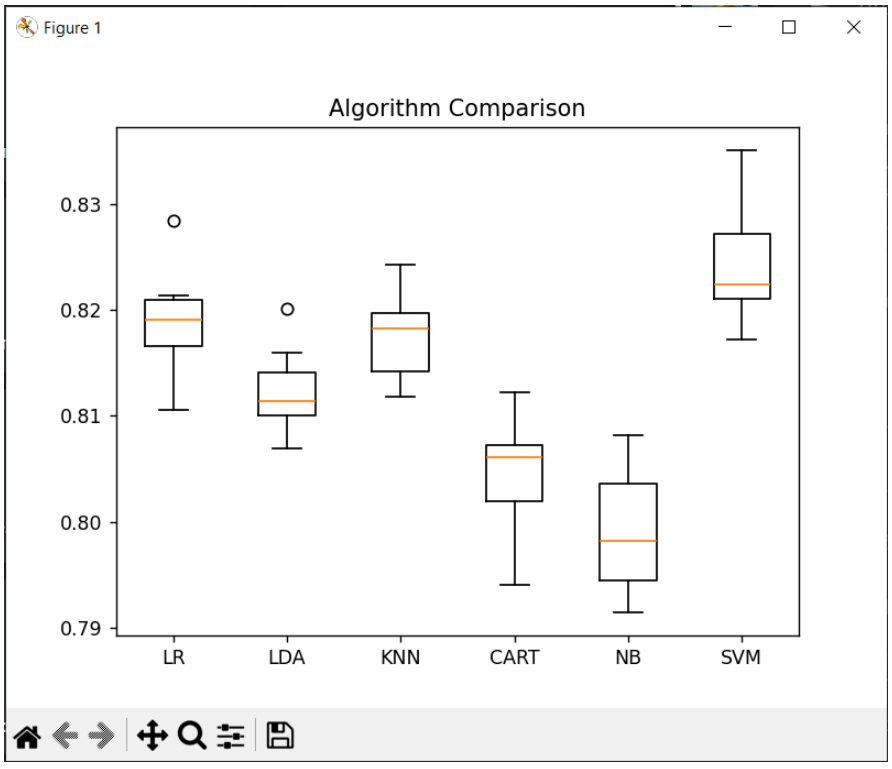

Рис.2.4.3. Діаграма порівняння алгоритмів.

Найкращу оцінку якості, як і в попередньому завданні, має метод SVM (0.824112). Найближчі до нього – LR (0.818849) і KNN (0.817606). LDA показав результат 0.812176. Найгірші показники у методів CART (0.804509) і NB (0.799080). За результатами аналізу варто вибрати SVM, якщо час виконання завдання не має значення, в іншому краще підходять LR чи KNN, оскільки вони мають вищу швидкодію.

Завдання №2.5. Класифікація даних лінійним класифікатором Ridge.

Лістинг файлу LR_2_task_5.py:

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

sns.set()
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))
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))

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:\course-4\semester-1\ai\lab2_project\venv\Scripts\pyth
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


Process finished with exit code 0
```

Рис.2.5.1. Результат виконання коду.

Налаштування класифікатора Ridge:

tol=1e-2: Це параметр, який вказує, наскільки точним повинен бути результат перед зупинкою процесу навчання. У цьому випадку, tol встановлений на значення 1e-2, що означає, що навчання буде зупинено, якщо зміна значення функції втрат менше за 0.01. Це допомагає підвищити швидкість навчання, оскільки можна зупинити процес, коли досягнуто задану точність.

solver="sag": Цей параметр вказує на вибір методу оптимізації для навчання моделі. У цьому випадку, solver встановлений на "sag", що означає "Stochastic Average Gradient".

Використані показники якості:

1. Accuracy – якість оцінки
2. Precision – точність
3. Recall – повнота
4. F1 – гармонійне середнє між точністю і повнотою

5. Cohen Kappa Score – міра ступеня узгодження між прогнозованими та спостережуваними класифікаціями, при цьому враховуючи можливість випадкового узгодження.
6. Matthews Corrcoef – міра узгодження між прогнозованими та спостережуваними класифікаціями, яка враховує всі чотири можливі результати класифікації: правильні позитивні, правильні негативні, помилкові позитивні і помилкові негативні.

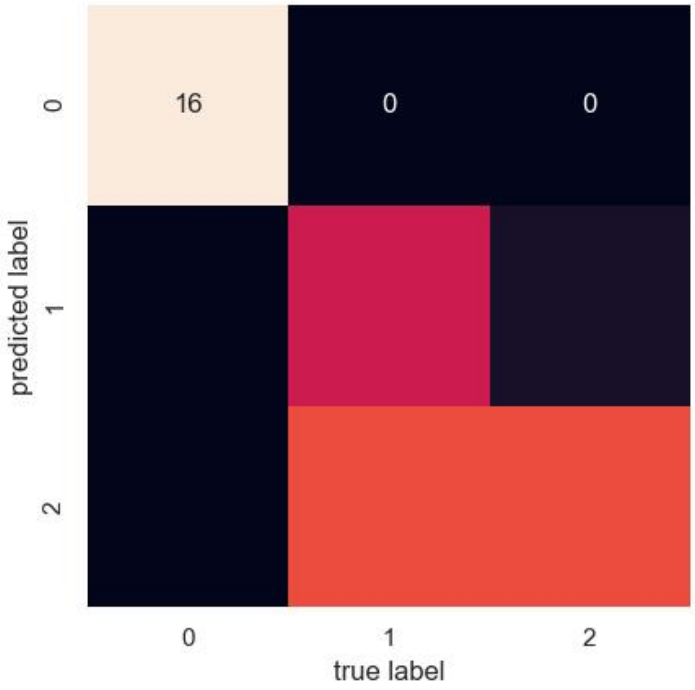

Рис.2.5.2. Матриця помилок.

Матриця помилок – матриця розміром N на N, де N – кількість класів, яка являється табличним представленням прогнозованих і фактичних значень для кожного можливого класу. Вона дозволяє візуалізувати кількість правильно та неправильно класифікованих прикладів для кожного класу у задачі класифікації.

Коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза - це метрики, які використовуються для вимірювання ступеня узгодженості між прогнозованими та спостережуваними класифікаціями в задачах класифікації. Обидві метрики дозволяють враховувати можливість випадкової згоди та допомагають визначити, наскільки ефективно класифікаційна модель вирішує завдання. Значення цих коефіціє-

нтів може лежати від -1 до 1, де 1 означає ідеальну узгодженість між спостережуваними та прогнозованими класифікаціями, 0 вказує на згоду, яка може бути досягнута випадково, а від'ємні значення вказують на гіршу узгодженість, ніж випадкова класифікація.

Висновки: в ході виконання лабораторної роботи було досліджено різні методи класифікації даних та здійснено їх порівняння, використовуючи спеціалізовані бібліотеки та мову програмування Python.