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 2.1. Класифікація за допомогою машин опорних векторів (SVM). 14 ознак з набору даних:

1. age – вік особи, числова;
2. workclass – тип зайнятості, категоріальна;
3. fnlwgt – статистична вага вибірки, числова;
4. education – рівень освіти, категоріальна;
5. education-num – кількість років навчання, числова;
6. marital-status – сімейний стан, категоріальна;
7. occupation – рід занять або професія, категоріальна;
8. relationship – сімейний статус, категоріальна;
9. race – раса, категоріальна;
1. 10.sex – стать особи, категоріальна;
2. 11.capital-gain – прибуток від капіталу, числова;
3. 12.capital-loss – збитки від капіталу, числова;
4. 13.hours-per-week – кількість годин роботи на тиждень, числова;
5. 14.native-country – країна народження, категоріальна.

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 cross_val_score, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))
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.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')

print(f"Accuracy: {round(accuracy * 100, 2)}%")
print(f"Precision: {round(precision * 100, 2)}%")
print(f"Recall: {round(recall * 100, 2)}%")
print(f"F1 score: {round(f1 * 100, 2)}%")

f1_cross_val_score = cross_val_score(classifier, X, y, scoring='f1_weighted',
cv=3)
print(f"Cross-validated F1 score: {round(100 * f1_cross_val_score.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 = [-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]])[0])
        count += 1

input_data_encoded = np.array(input_data_encoded)
predicted_class = classifier.predict(input_data_encoded.reshape(1, -1))
print("Predicted class for test data point: ", label_encoder[-
1].inverse_transform(predicted_class)[0])
```

Результат виконання програми:

```
Accuracy: 79.56%
Precision: 79.26%
Recall: 79.56%
F1 score: 75.75%
Cross-validated F1 score: 76.09%
Predicted class for test data point:  <=50K
```

Рис. 1

Передбачуваний клас для тестової точки даних: <=50K, тобто особа з даним набором отримуватиме до 50 000 доларів на рік.

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами.

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

input_file = 'income_data.txt'
X = []
y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 5000

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)

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

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')

print(f"Accuracy: {round(accuracy * 100, 2)}%")
print(f"Precision: {round(precision * 100, 2)}%")
print(f"Recall: {round(recall * 100, 2)}%")
print(f"F1 score: {round(f1 * 100, 2)}%")
```

Результат виконання програми:

```
Accuracy: 52.8%
Precision: 56.68%
Recall: 52.8%
F1 score: 42.63%
```

Рис. 2

Для алгоритму з поліноміальним ядром було зменшено кількіть точок (до 5000), оскільки SVM з даним типом ядра високого степеня має високу обчислювальну складність, а тому великий час обчислення. Таке рішення вплинуло на якість класифікації.

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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)

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

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')

print(f"Accuracy: {round(accuracy * 100, 2)}%")
print(f"Precision: {round(precision * 100, 2)}%")
print(f"Recall: {round(recall * 100, 2)}%")
print(f"F1 score: {round(f1 * 100, 2)}%")
```

Результат виконання програми:

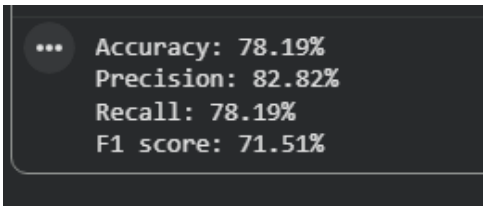

Рис. 3

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_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)

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

accuracy = accuracy_score(y_test, y_test_pred)
precision = precision_score(y_test, y_test_pred, average='weighted')
recall = recall_score(y_test, y_test_pred, average='weighted')
f1 = f1_score(y_test, y_test_pred, average='weighted')

print(f"Accuracy: {round(accuracy * 100, 2)}%")
print(f"Precision: {round(precision * 100, 2)}%")
print(f"Recall: {round(recall * 100, 2)}%")
print(f"F1 score: {round(f1 * 100, 2)}%")
```

Результат виконання програми:

```
Accuracy: 60.47%
Precision: 60.64%
Recall: 60.47%
F1 score: 60.55%
```

Рис. 4

У завданні порівнювалися три ядра для SVM-класифікації: поліноміальне, гаусове та сигмоїдальне. З найкращої сторони алгоритм SVM показав себе з гаусовим ядром, оскільки він має найвищі показники вимірюваних метрик. Проте через значне навантаження на процесор і пам'ять було змінено кількість точок для поліноміального ядра, що відповідно вплинуло на якість тренування, а, отже, і на показники.

Завдання 2.3. 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("Перші 5 прикладів:\n{}".format(iris_dataset['data'][:5]))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Форма масиву target: {}".format(iris_dataset['target'].shape))
print("Відповіді:\n{}".format(iris_dataset['target']))
```

Результат виконання програми:

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
.. _iris_dataset:

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

**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Перші 5 прикладів:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Тип масиву target: <class 'numpy.ndarray'>
Форма масиву target: (150,)
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 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
 2 2]
```

Рис. 5

Лістинг програми:

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

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)

print(dataset.shape)
print(dataset.head(20))
print(dataset.describe())
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]
# Розділення X і Y на навчальний та контрольний набори
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("Точність (SVC):", accuracy_score(Y_validation, predictions))
print("Матриця неточностей (SVC): \n", confusion_matrix(Y_validation,
predictions))
print("Звіт про класифікацію (SVC):\n", classification_report(Y_validation,
predictions))

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

# Прогноз для нової квітки
prediction1 = model.predict(X_new)
print("Прогноз (SVC): {}".format(prediction1))

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, Y_train)
predictions_lda = model.predict(X_validation)

print("Точність  (LDA):", accuracy_score(Y_validation, predictions_lda))
prediction2 = lda.predict(X_new)
print("Прогноз (LDA): {}".format(prediction2))
print("Спрогнозована мітка:", prediction2[0])
```

Результат виконання програми:

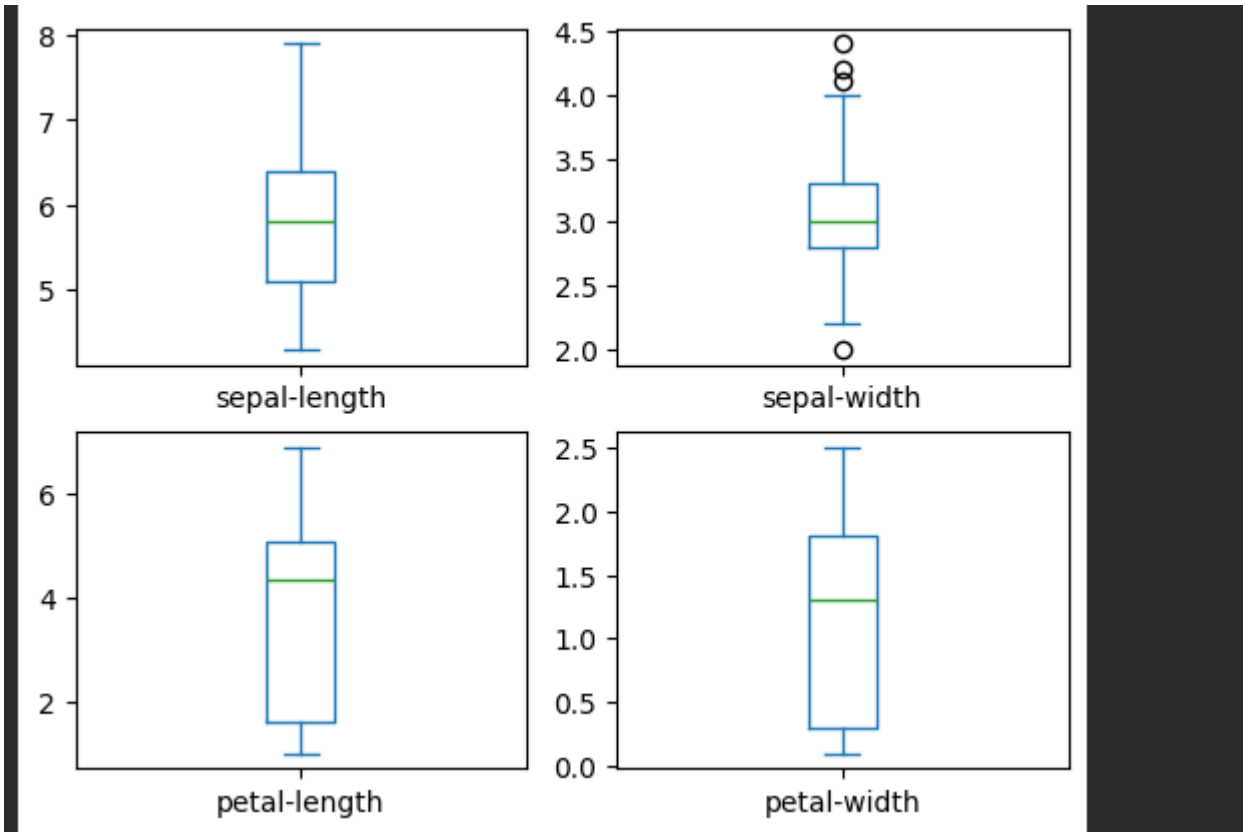

Рис. 6. Діаграма розмаху

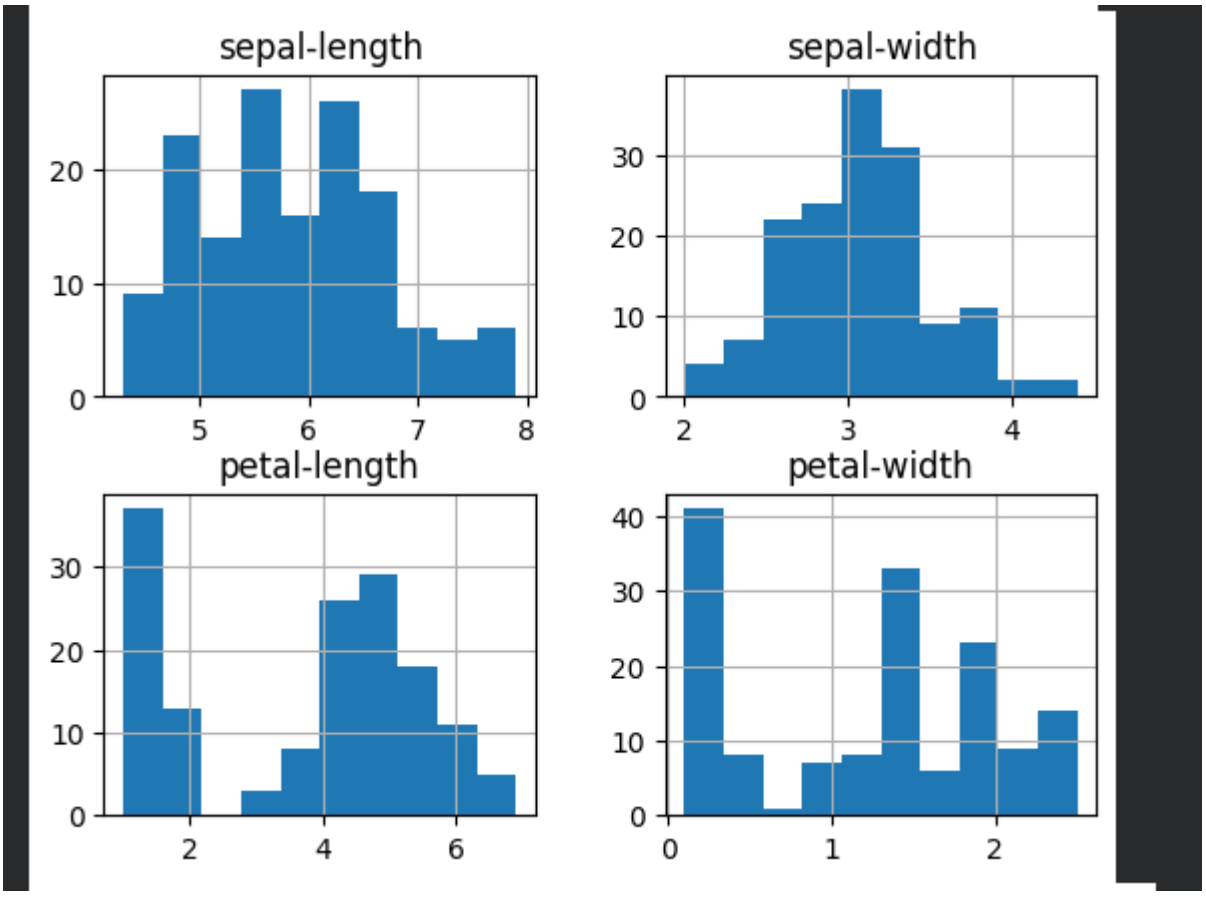

Рис. 7. Гістограма розподілу атрибутів датасета

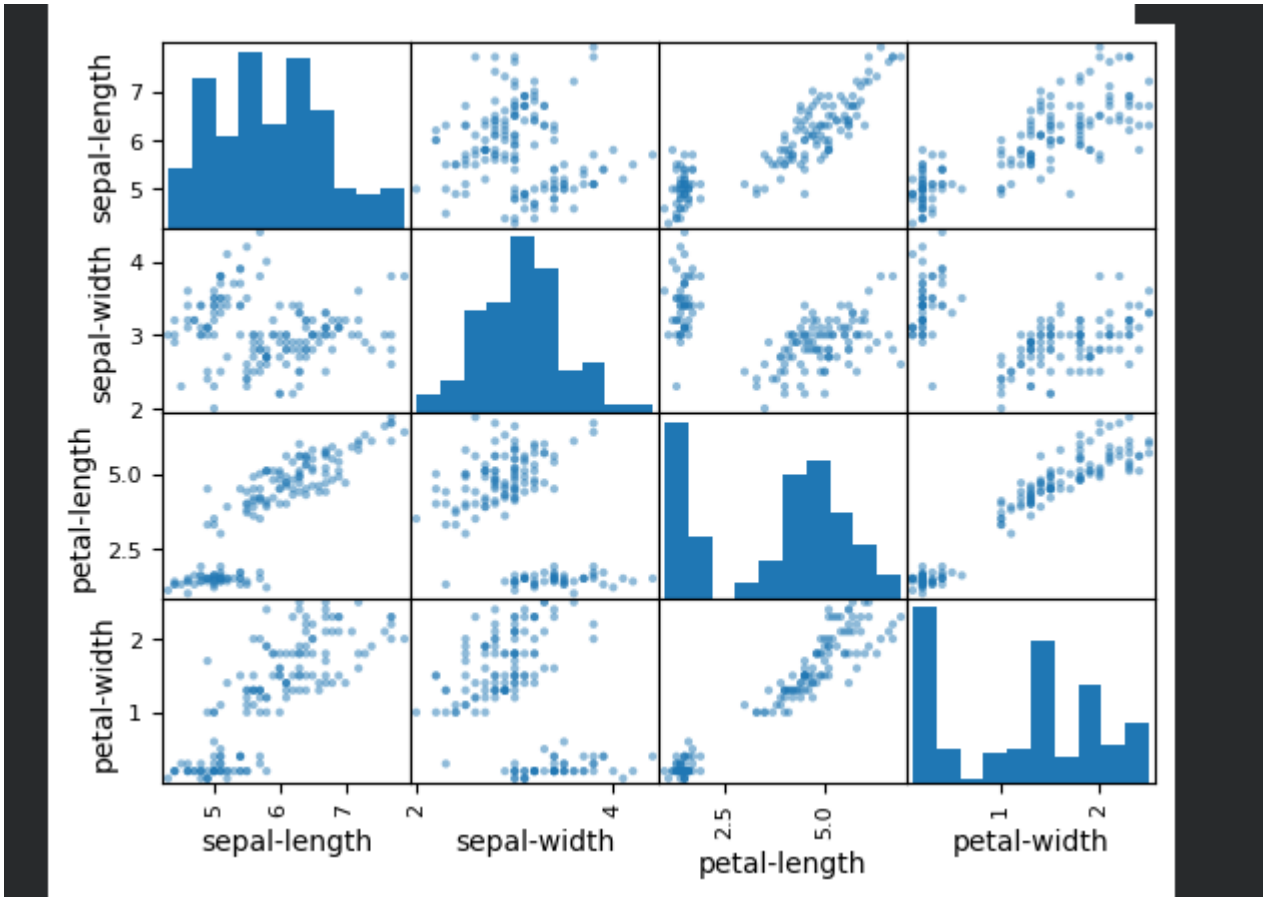

Рис. 8 Матриця діаграм розсіювання

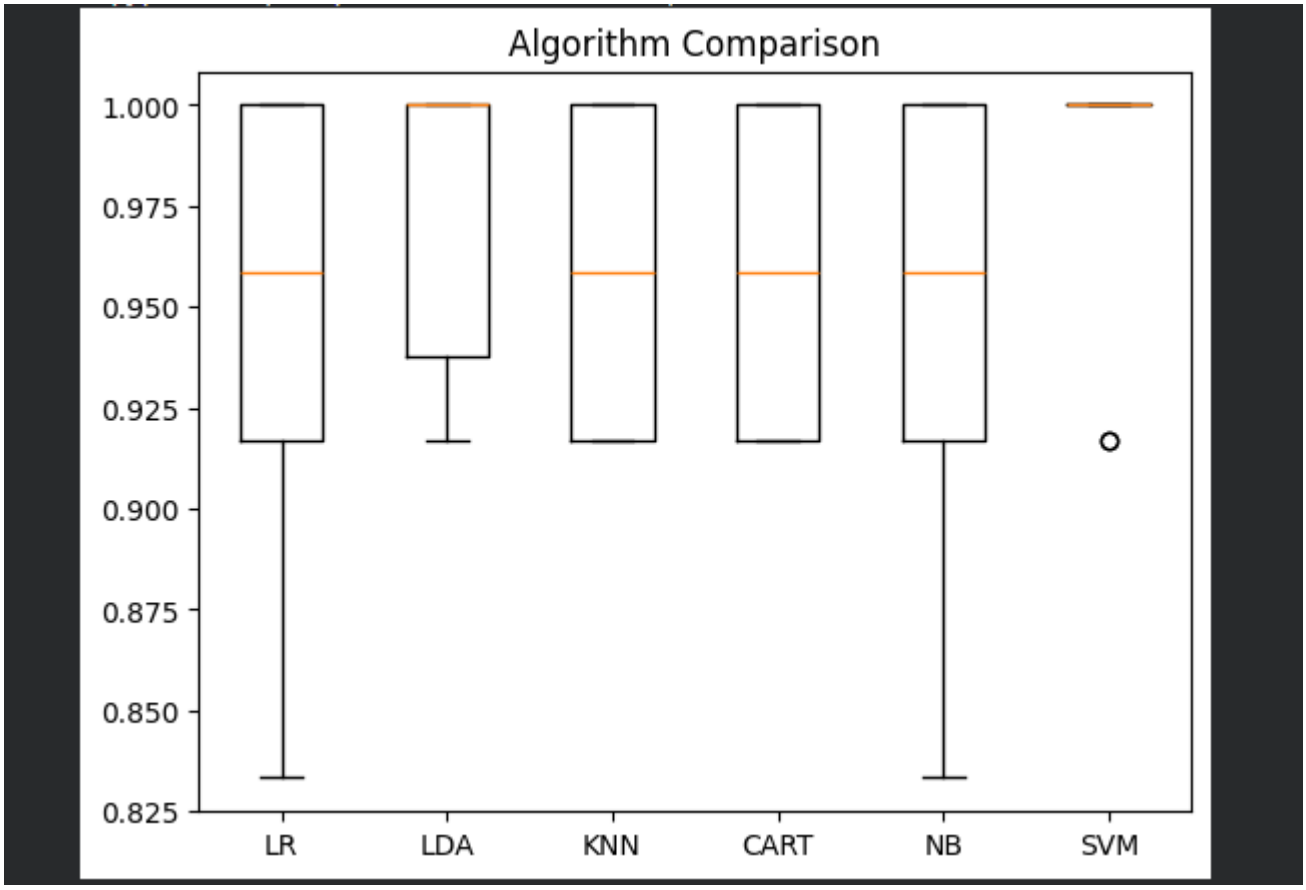

Рис. 9. Порівняння алгоритмів

```
(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. 10

```
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)
```

Рис. 11

```
Точність (SVC): 0.9666666666666667
Матриця неточностей (SVC):
 [[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
Звіт про класифікацію (SVC):
                  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 (SVC): ['Iris-setosa']
Точність  (LDA): 0.9666666666666667
Прогноз (LDA): ['Iris-setosa']
Спрогнозована мітка: Iris-setosa
```

Рис. 12

На рис. 9 продемонстровано діаграму з порівнянням алгоритмів. З нього можна зробити такий висновок: алгоритм SVM показує найвищу медіану точності (помаранчева лінія, типове значення) і найменший розкид результатів. Тому для прогнозу сорту квітки вибираємо дану модель. Іншим варіантом може бути алгоритм LDA, який має високу стабільність (через маленький розкид) і високу точність.

Для прогнозу сорту квітки було обрано дві моделі SVM і LDA для порівняння їх результатів. Обидві моделі показали якість класифікації (точність) близько 0.97. Дана квітка належить до класу «Iris-setosa».

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1.

Лістинг програми:

```
import numpy as np
from matplotlib import pyplot
from sklearn import preprocessing
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 accuracy_score, precision_score, recall_score,
f1_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

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)

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

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')))

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()

def evaluate_classification_metrics(name, model, X_test, y_test):
    y_test_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_test_pred)
    precision = precision_score(y_test, y_test_pred, average='weighted')
    recall = recall_score(y_test, y_test_pred, average='weighted')
    f1 = f1_score(y_test, y_test_pred, average='weighted')
    print(f"Model name: {name}")
    print(f"Accuracy: {round(accuracy * 100, 2)}%")
    print(f"Precision: {round(precision * 100, 2)}%")
    print(f"Recall: {round(recall * 100, 2)}%")
    print(f"F1 score: {round(f1 * 100, 2)}% \n")


model_LR = LogisticRegression(solver='liblinear', multi_class='ovr')
model_LR.fit(X_train, y_train)
evaluate_classification_metrics('LR', model_LR, X_test, y_test)

model_LDA = LinearDiscriminantAnalysis()
model_LDA.fit(X_train, y_train)
evaluate_classification_metrics('LDA', model_LDA, X_test, y_test)

model_KNN = KNeighborsClassifier()
model_KNN.fit(X_train, y_train)
evaluate_classification_metrics('KNN', model_KNN, X_test, y_test)

model_CART = DecisionTreeClassifier()
model_CART.fit(X_train, y_train)
evaluate_classification_metrics('CART', model_CART, X_test, y_test)

model_NB = GaussianNB()
model_NB.fit(X_train, y_train)
```

```
evaluate_classification_metrics('NB', model_NB, X_test, y_test)

model_SVM = SVC(gamma='auto')
model_SVM.fit(X_train, y_train)
evaluate_classification_metrics('SVM', model_SVM, X_test, y_test)
```

Результат виконання програми:

```
LR: 0.792283 (0.006167)
LDA: 0.811637 (0.005701)
KNN: 0.767748 (0.003026)
CART: 0.808115 (0.008070)
NB: 0.789133 (0.006934)
```

Рис. 13

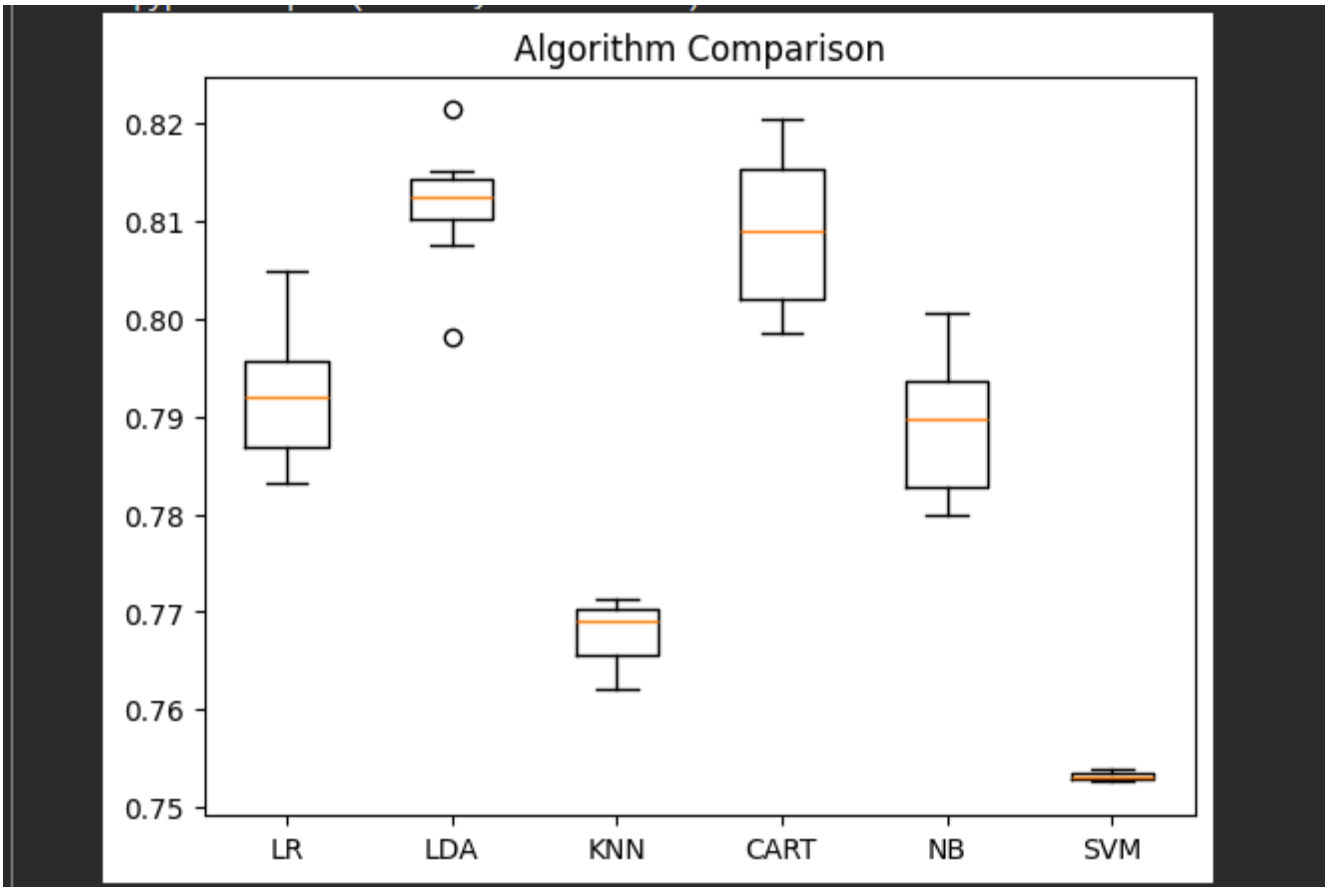

Рис. 14

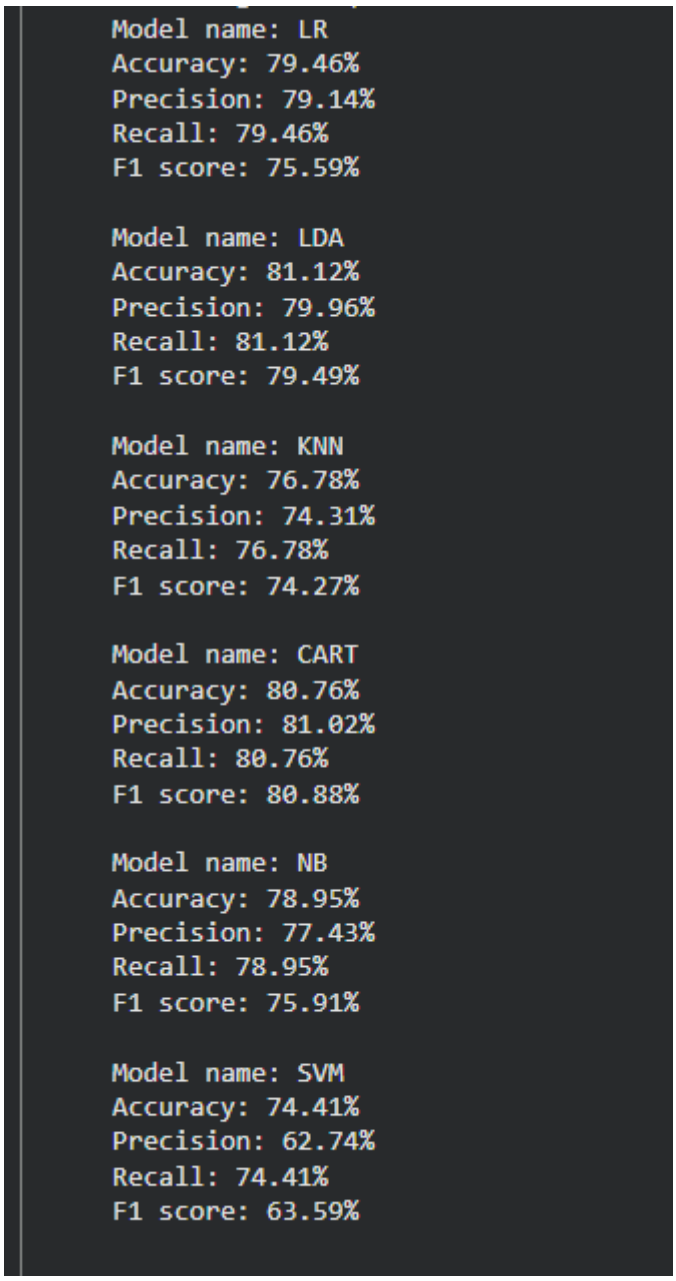

Рис. 15

На рис. 14 продемонстровано діаграму з порівнянням алгоритмів. З нього можна зробити такий висновок: алгоритм LDA є найкращим для рішення даної задачі, оскільки має найвищу точність (на це вказує медіана точності, помаранчева лінія) і високу стабільність (невеликий розмах, тобто різниця між результатами на різних розбиттях мала).

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge.

Лістинг програми:

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

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")

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

Результат виконання програми:

```
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
```

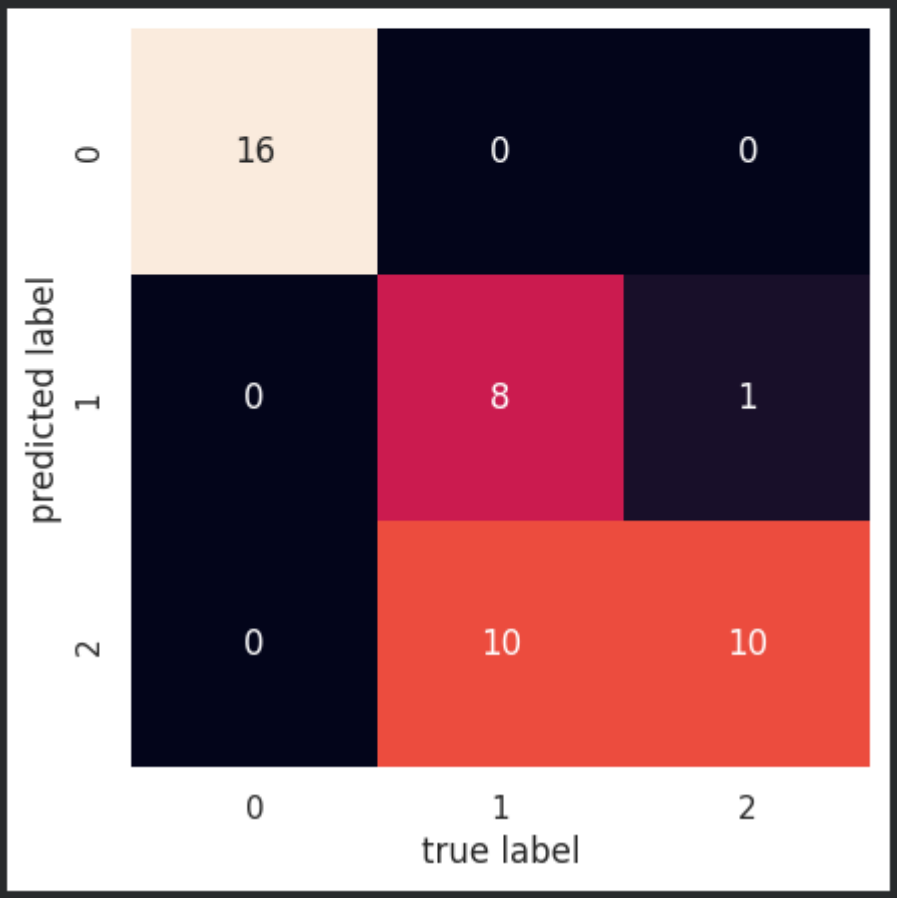

Рис. 16

У коді програми використані наступні налаштування класифікатора Ridge:

- tol=1e-2 — це точність зупинки алгоритму;

- solver='sag' — метод для виконання обчислювальних процедур ('sag' – стохастичний градієнтний метод, який підходить для великих наборів даних).

Показники якості, які використовуються:

- Accuracy ≈ 76%,
- Precision ≈ 83%,
- Recall ≈ 76%,
- F1 Score ≈ 76%,
- Cohen's Kappa ≈ 64%,
- Matthews Corrcoef ≈ 68%.

На зображенні «Confusion.jpg» продемонстровано матрицю плутанини (матрицю неточностей), яка показує, як часто модель плутає класи. Діагональні елементи матриці показують правильно класифіковані приклади, інші клітинки – це помилки класифікації.

Коефіцієнт Коена Каппа вимірює наскільки узгоджені передбачення моделі з істинними значеннями, з урахуванням випадкових збігів. Значення даного показника 64%, тобто це гарний, але не ідеальний результат.

Коефіцієнт кореляції Метьюза вимірює кореляцію між передбаченими та фактичними мітками, тобто статистичний зв'язок між ними. Значення даного показника 68%, тобто це часткова відповідність прогнозів дійсним міткам.

Висновок: в ході виконання лабораторної роботи ми дослідили різні методи класифікації даних та навчилися їх порівнювати, використовуючи спеціалізовані бібліотеки та мову програмування Python.

Репозиторій: https://github.com/Vanchik21/AI