ЛАБОРАТОРНА РОБОТА № 2

Варіант 17

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

Завдання 1

Список ознак:

Age – Тип: Числова

Workclass– Тип: Категоріальна

Education – Тип: Категоріальна

Education-num – Тип: Числова

Marital-status – Тип: Категоріальна

Occupation – Тип: Категоріальна

Relationship – Тип: Категоріальна

Race – Тип: Категоріальна

Sex – Тип: Категоріальна

Capital-gain – Тип: Числова

Capital-loss – Тип: Числова

Hours-per-week – Тип: Числова

Native-country – Тип: Категоріальна

Income – Тип: Категоріальна

Лістинг:

```
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 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.strip().split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            Y.append(0)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            Y.append(1)
            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:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))
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: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1 Score: {f1 * 100:.2f}%")

f1_cv = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print(f"F1 score (cross-validation): {f1_cv.mean() * 100:.2f}%")
```

```
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([0] * len(input_data))

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

input_data_encoded = input_data_encoded.reshape(1, -1)
predicted_class = classifier.predict(input_data_encoded)
print(f"Predicted class: {'<=50K' if predicted_class[0] == 0 else '>50K'}")
```

Результат виконання програми:

```
Accuracy: 79.56%
Precision: 79.26%
Recall: 79.56%
F1 Score: 75.75%
F1 score (cross-validation): 76.01%
Predicted class: <=50K
```

Рис. 1

Тестова точка належить до класу: Income (Дохід)

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Завдання було виконано в одному файлі:

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

# Зчитування та попередня обробка даних
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.strip().split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            Y.append(0)
            count_class1 += 1
        elif data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            Y.append(1)
            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:
        le = preprocessing.LabelEncoder()
        X_encoded[:, i] = le.fit_transform(X[:, i])
        label_encoder.append(le)

X = X_encoded[:, :-1].astype(int)
Y = np.array(Y)

scaler = StandardScaler()
X = scaler.fit_transform(X)

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

metrics = {}

poly_svc = SVC(kernel='poly', degree=2, random_state=0)
poly_svc.fit(X_train, y_train)
```

```
y_test_pred = poly_svc.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')

metrics['Polynomial Kernel'] = {
    "Accuracy": accuracy * 100,
    "Precision": precision * 100,
    "Recall": recall * 100,
    "F1 Score": f1 * 100
}

rbf_svc = SVC(kernel='rbf', random_state=0)
rbf_svc.fit(X_train, y_train)
y_test_pred = rbf_svc.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')

metrics['RBF Kernel'] = {
    "Accuracy": accuracy * 100,
    "Precision": precision * 100,
    "Recall": recall * 100,
    "F1 Score": f1 * 100
}

sigmoid_svc = SVC(kernel='sigmoid', random_state=0)
sigmoid_svc.fit(X_train, y_train)
y_test_pred = sigmoid_svc.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')

metrics['Sigmoid Kernel'] = {
    "Accuracy": accuracy * 100,
    "Precision": precision * 100,
    "Recall": recall * 100,
    "F1 Score": f1 * 100
}

for kernel, scores in metrics.items():
```

```
    print(f"\n{kernel}:")
    print(f"Accuracy: {scores['Accuracy']:.2f}%")
    print(f"Precision: {scores['Precision']:.2f}%")
    print(f"Recall: {scores['Recall']:.2f}%")
    print(f"F1 Score: {scores['F1 Score']:.2f}%")
```

Результат виконання програми:

```
Polynomial Kernel:
Accuracy: 80.29%
Precision: 80.28%
Recall: 80.29%
F1 Score: 76.79%

RBF Kernel:
Accuracy: 83.36%
Precision: 82.55%
Recall: 83.36%
F1 Score: 82.45%

Sigmoid Kernel:
Accuracy: 75.27%
Precision: 75.05%
Recall: 75.27%
F1 Score: 75.16%
```

Рис. 2

Краще всіх себе показав RBF, маючи найкращі показники серед усіх.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Тестовий код:

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

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

```
print("Опис набору даних:\n{}".format(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("Значення ознак для перших п'яти
прикладів:\n{}".format(iris_dataset['data'][:5]))

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

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

Результат виконання програми:

```
Ключі iris_dataset:
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
Опис набору даних:
.. _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)
Значення ознак для перших п'яти 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'>
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 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 2 2
 2 2]
```

Рис. 3

Код для візуалізації (деяки фрагменти кода з л.р. були винесені в окремі функцій):

```
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot

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)


def print_array_data():
    print(dataset.shape)
    print(dataset.head(20))
    print(dataset.describe())
    print(dataset.groupby('class').size())

def show_array_data_in_chart():
    dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
    pyplot.show()
    dataset.hist()
    pyplot.show()
    scatter_matrix(dataset)
    pyplot.show()


show_array_data_in_chart()
```

Результат виконання програми:

```
(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. 4

Результат виконання програми:

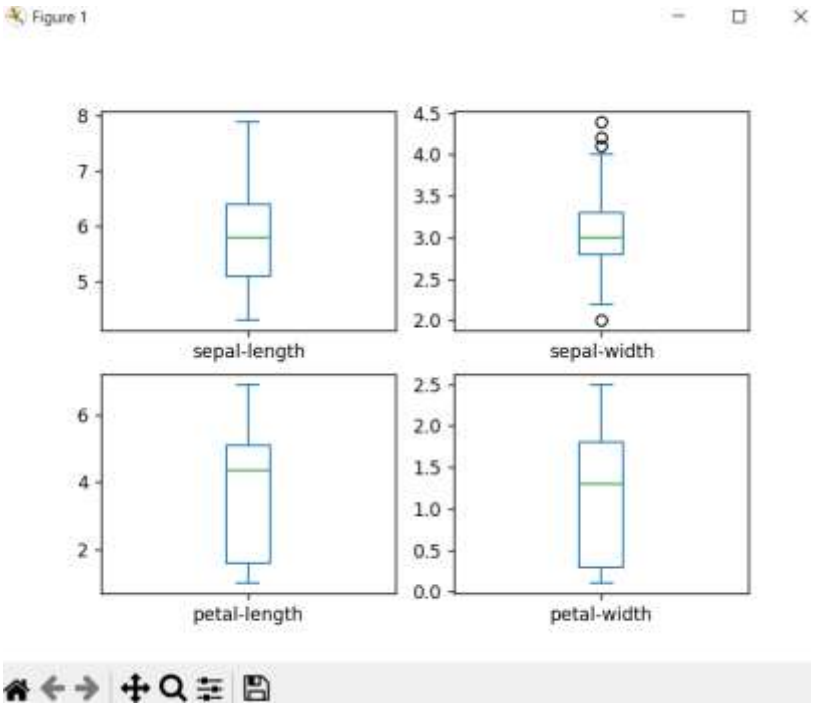

Рис. 5

Результат виконання програми:

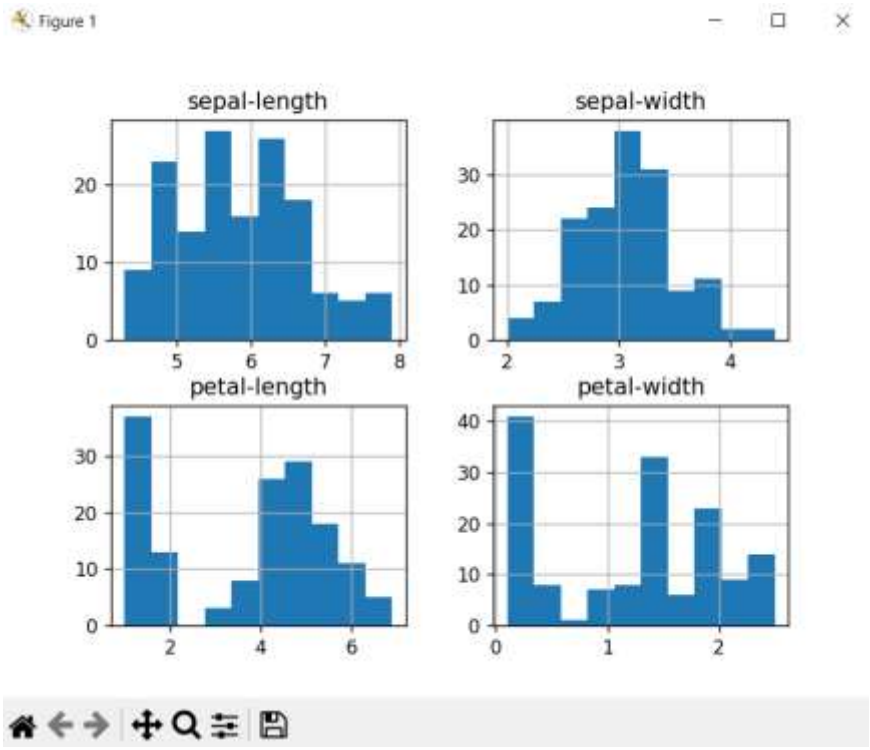

Рис. 6

Результат виконання програми:

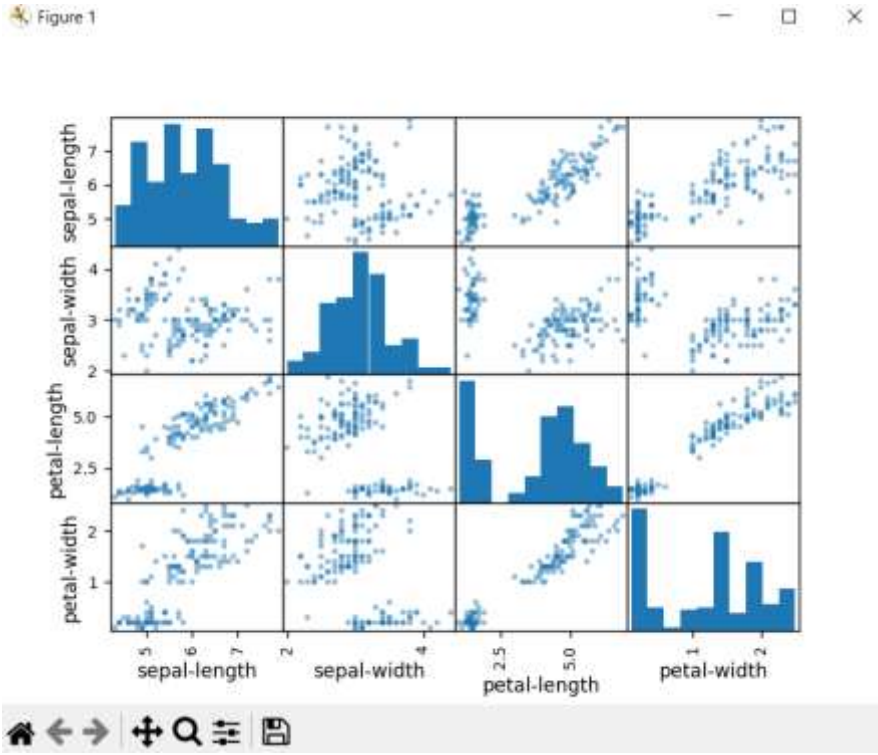

Рис. 7

Вибір та порівняння підходящої моделі

Результат виконання програми:

```
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)
```

Рис. 8

Результат виконання програми:

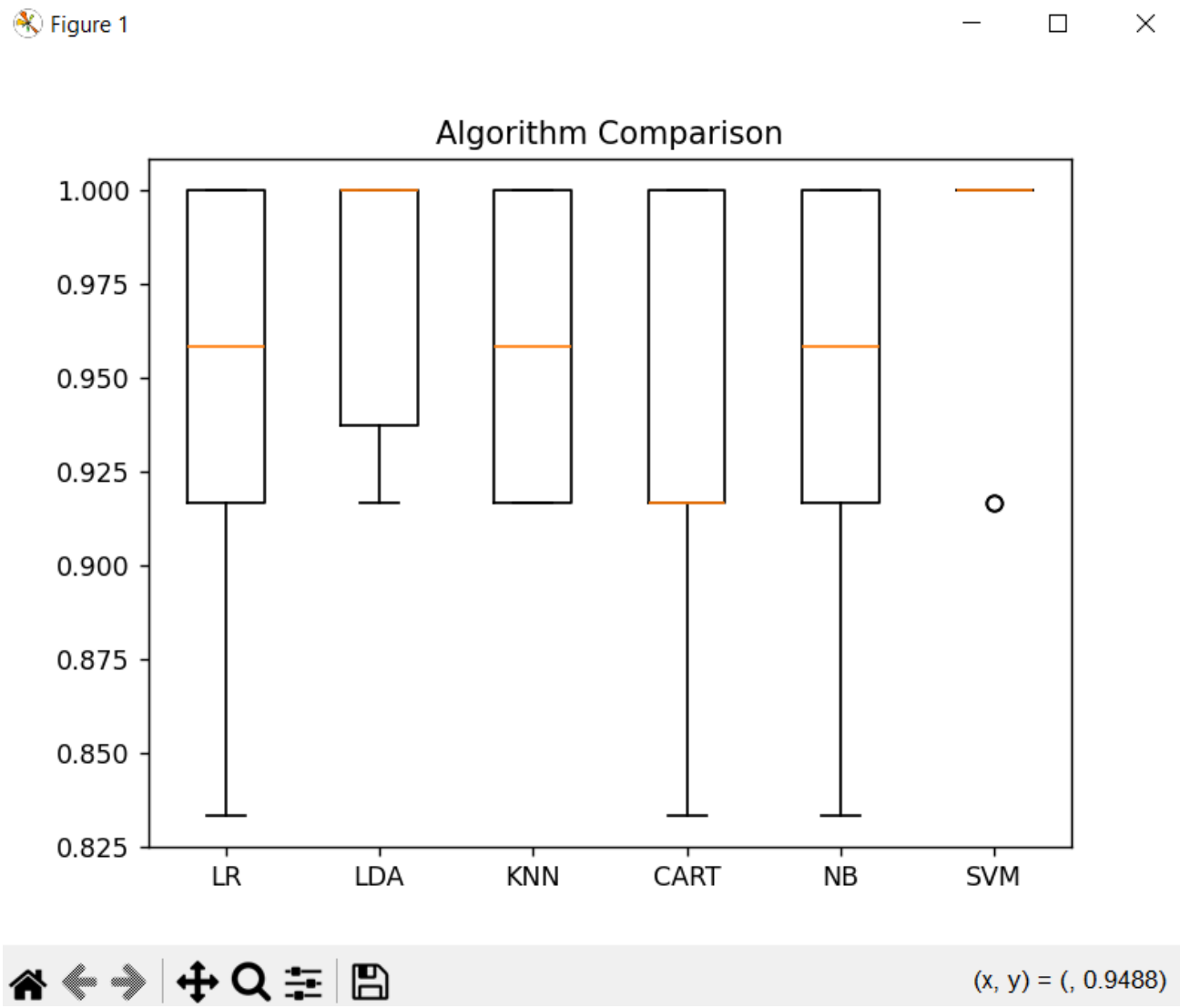

Рис. 9

SVM має найбільшу точність та найменше відхилення серед решти, тому вважаю що варто обрати його.

Оцінка якості:

Результат виконання програми:

```
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
```

Рис. 10

Також виявлено до якого класу належить квітка з кроку 8:

Результат виконання програми:

```
X_new shape: (1, 4)
Prediction: ['Iris-setosa']
Prediction: Iris-setosa
```

Рис. 11

Лістинг:

```
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)


def print_array_data():
    print(dataset.shape)
    print(dataset.head(20))
    print(dataset.describe())
    print(dataset.groupby('class').size())

def show_array_data_in_chart():
    dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
    pyplot.show()
    dataset.hist()
    pyplot.show()
    scatter_matrix(dataset)
    pyplot.show()

#print_array_data()
#show_array_data_in_chart()

array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=0.20, random_state=1)

def models_test():
    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()))

    pyplot.boxplot(results, labels=names)
    pyplot.title('Algorithm Comparison')
    pyplot.show()


#models_test()

model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

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("X_new shape: {}".format(X_new.shape))

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

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг:

Результат виконання програми:

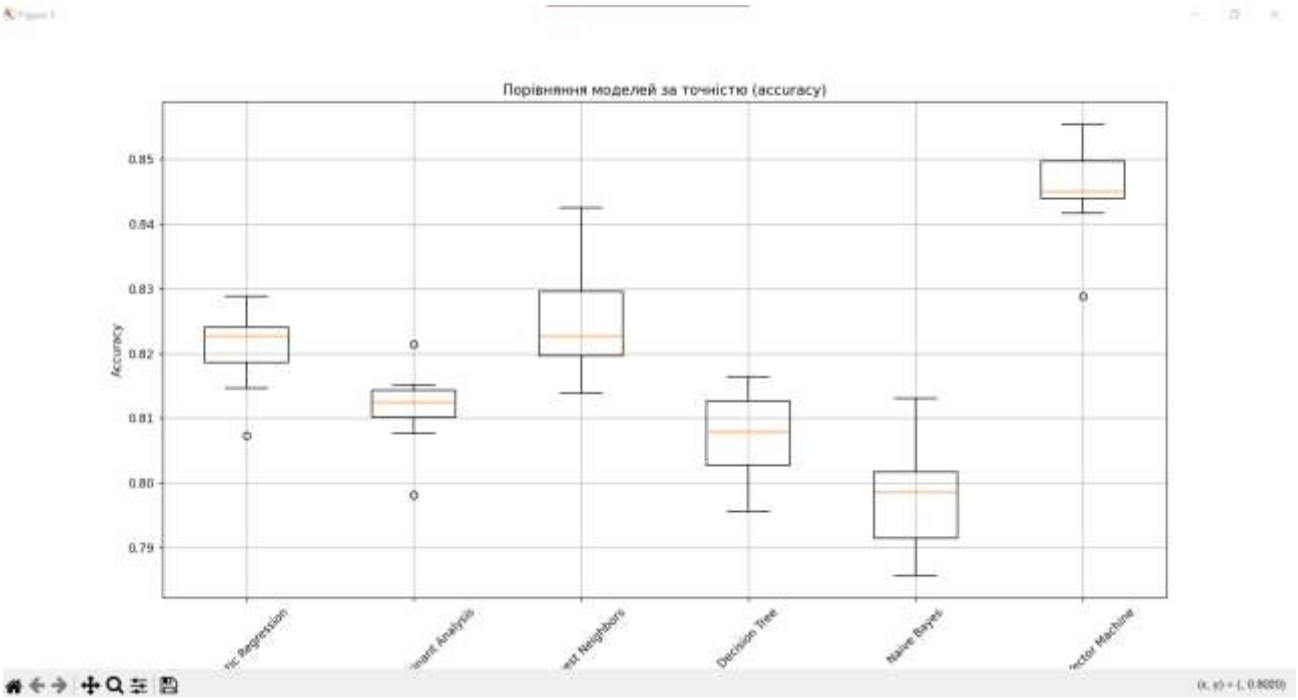

Рис. 12

Результат виконання програми:

```
Logistic Regression: Mean Accuracy = 0.8208 (0.0058)
Linear Discriminant Analysis: Mean Accuracy = 0.8116 (0.0057)
K-Nearest Neighbors: Mean Accuracy = 0.8254 (0.0096)
Decision Tree: Mean Accuracy = 0.8075 (0.0063)
Naive Bayes: Mean Accuracy = 0.7979 (0.0081)
Support Vector Machine: Mean Accuracy = 0.8456 (0.0070)
```

Рис. 13

В цій ситуації найкращий результат показав SVM, можна зробити висновок що для цієї задачі він підходить найкраще.

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

Лістинг:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
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 io import BytesIO
from sklearn.metrics import confusion_matrix

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('\nClassification Report:\n', metrics.classification_report(y_test, y_pred))

mat = confusion_matrix(y_test, y_pred)
sns.set()
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True Label')
plt.ylabel('Predicted Label')
plt.title("Confusion Matrix of Ridge Classifier")
plt.savefig("Confusion.jpg")
f = BytesIO()
plt.savefig(f, format="svg")
plt.show()
```

Результат виконання програми:

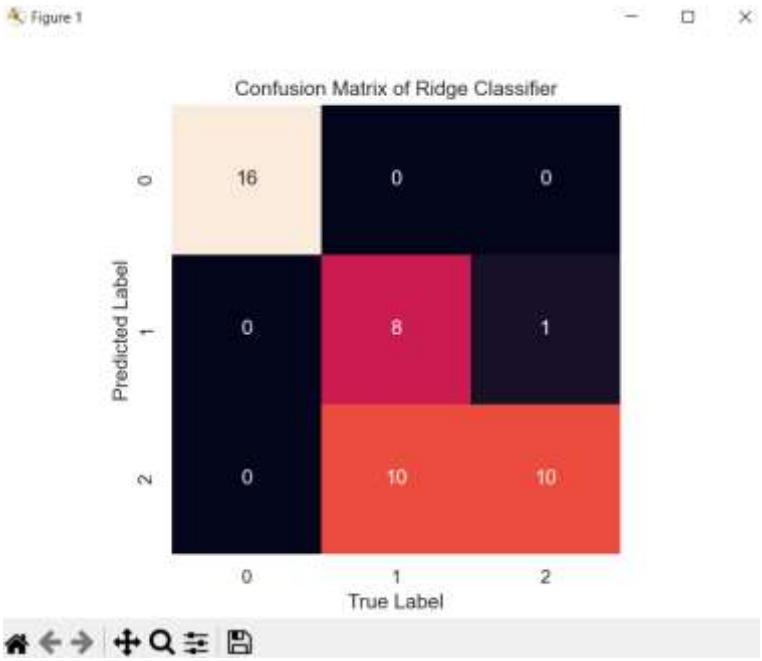

Рис. 14

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       0.89      0.44      0.59        18
           2       0.50      0.91      0.65        11

    accuracy                           0.76        45
   macro avg       0.80      0.78      0.75        45
weighted avg       0.83      0.76      0.75        45
```

Рис. 15

Налаштування класифікатора Ridge:

1. tol=1e-2: Параметр tol визначає допустиму похибку для зупинки ітерацій алгоритму.

2. solver='sag': solver визначає метод розв’язання задачі оптимізації. 'sag' - Stochastic Average Gradient: є ітеративним методом, що ефективно працює на великих наборах даних. Він використовує стохастичний середній градієнт для мінімізації функції втрат.

Матриця плутанини показує наступне:

0: Класифікатор правильно передбачив всі 16 зразків.

1: З 18 зразків 8 було передбачено правильно, 10 зразків передбачено як клас 2, а 1 як клас 1.

2: Класифікатор правильно передбачив 10 зразків, 1 зразок було передбачено як клас 1.

Ця матриця плутанини показує, що класифікатор добре працює для класу 0, при цьому досить погано з іншими.

Коефіцієнт кореляції Метьюза – це оцінення якості матриці плутанини, отримане значення каже про те що модель демонструє не дуже хороший результат класифікування.

Коефіцієнта Каппа Коена – це ступінь збігів передбачень класифікатора з реальними мітками зазначених класів (враховуючи факт, що частина збігів це випадковість). Саме в цьому коді ми звіряємо показники передбачень з тестового набору ірисів.

Git: https://github.com/PavlenkoOks/AI