ЛАБОРАТОРНА РОБОТА № 2

ОПЕРЕДНЯ ОБРОБКА ТА КОНТРОЛЬОВАНА КЛАСИФІКАЦІЯ ДАНИХ

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити попередню обробку та класифікацію даних.

Хід роботи:

Завдання 2.1. Класифікація за допомогою машин опорних векторів (SVM)

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

```
F1 score: 56.15%
Accuracy: 62.64%
Precision: 75.88%
Recall: 62.64%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.svm import LinearSVC, SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split, cross_val_score

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

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

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

        if '?' in line:
            continue

        data = line[:-1].split(', ')

        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1

        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1

# Перетворення на масив numpy
X = np.array(X)

# Перетворення рядкових даних на числові
label_encoder = []
X_encoded = np.empty(X.shape)

for i,item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])

X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

# Створення SVM-класифікатора
# Навчання класифікатора
# Перехресна перевірка
classifier = OneVsOneClassifier(SVC(random_state=0))
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=5)
```

```python
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=5)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри, акуратності, повноти, точності для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
accuracy = cross_val_score(classifier, X, y, scoring='accuracy', cv=3)
precision = cross_val_score(classifier, X, y, scoring='precision_weighted', cv=3)
recall = cross_val_score(classifier, X, y, scoring='recall_weighted', cv=3)

print("F1 score: " + str(round(100*f1.mean(), 2)) + "%")
print("Accuracy: " + str(round(100*accuracy.mean(), 2)) + "%")
print("Precision: " + str(round(100*precision.mean(), 2)) + "%")
print("Recall: " + str(round(100*recall.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
1 usage
def predict (input_data):
    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] = 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])
    print("Input data:", input_data)
    print("Predicted class:", label_encoder[-1].inverse_transform(predicted_class)[0])

# Використання класифікатора для кодованої точки даних
# та виведення результату
predict(input_data)
```

Рис.1 Результат аналізу акуратності, повноти, точності та F1 та код програми

Висноквок до завдання: Тестова точка даних, яка була передбачена за допомогою SVM-класифікатора, належить класу "<=50K".

Завдання 2.2. Класифікація за допомогою машин опорних векторів (SVM)

```
classifier = OneVsOneClassifier(SVC(kernel='poly', degree=2))
```

```
F1 score: 70.68%
Accuracy: 77.87%
Precision: 81.53%
Recall: 77.87%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

Рис.2 Результат аналізу за використання поліноміального ядра

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

```
F1 score: 71.95%
Accuracy: 78.61%
Precision: 83.06%
Recall: 78.61%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

Рис.3 Результат аналізу за використання ядра Гауса

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

```
F1 score: 63.77%
Accuracy: 63.89%
Precision: 63.65%
Recall: 63.89%
Input data: ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
Predicted class: <=50K
```

Рис.4 Результат аналізу за використання сигмоїдального ядра

Для нерівномірних даних ядро Гауса є найкращим вибором для забезпечення високої точності класифікації. Полімінеальне ядро з максимальним значенням degree може бути ще більш ефективним, але вимагатиме дуже потужного комп'ютера.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

```
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']
Назви 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 рядків ознак:
[[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 відповідей: <class 'numpy.ndarray'>
Форма масиву відповідей: (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 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 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]
```

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

print(f"Ключі iris_dataset: \n{iris_dataset.keys()}")
print(iris_dataset['DESCR'][:193] + "\n...")
print(f"Назви відповідей: {iris_dataset['target_names']}")
print(f"Назви ознак: {iris_dataset['feature_names']}")
print(f"Тип масиву data: {type(iris_dataset['data'])}")
print(f"Форма масиву data: {iris_dataset['data'].shape}")
print(f"Перші 5 рядків ознак: \n{iris_dataset['data'][:5]}")

print(f"Тип масиву відповідей: {type(iris_dataset['target'])}")
print(f"Форма масиву відповідей: {iris_dataset['target'].shape}")
print(f"Відповіді: {iris_dataset['target']}")
```

Рис.5 Виведення інформації про дані, про відповіді та код програми

```python
# Завантаження датасету
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())
```

```python
# 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()
```

```
(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
```

Рис.6 Розмір масиву даних та перші 20 записів

	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.7 Характеристики даних, кількість за класами та тип даних

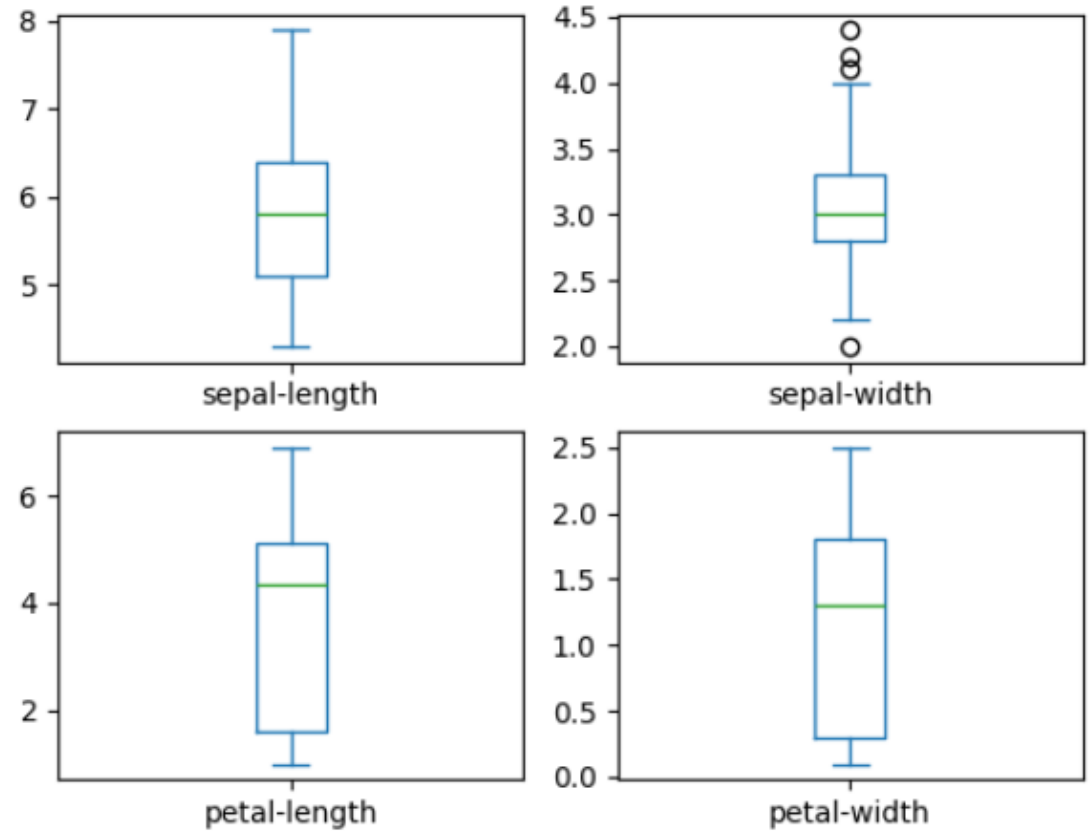


Рис. 8 Діаграма розмаху

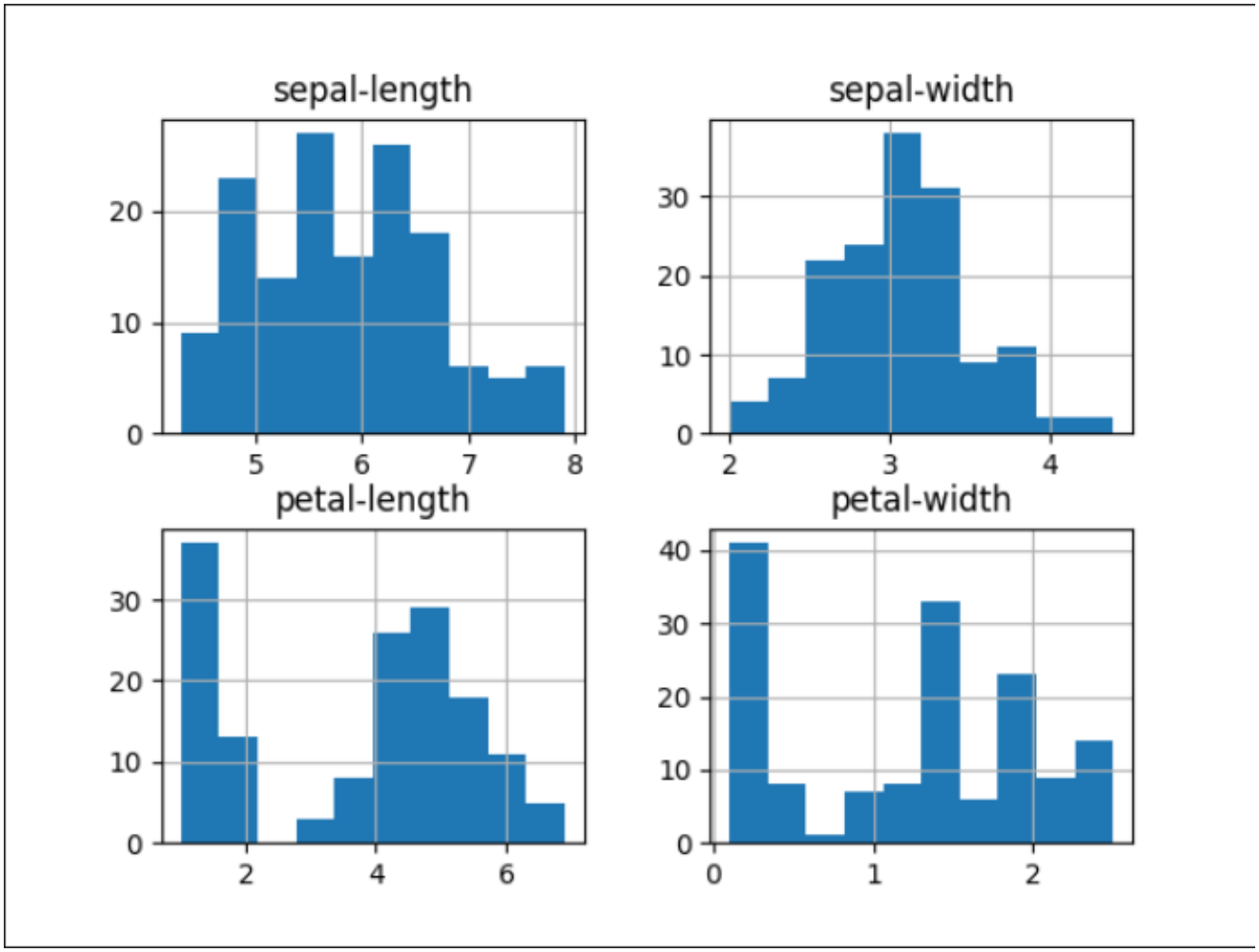


Рис.9 Гістограма розподілу атрибутів датасета

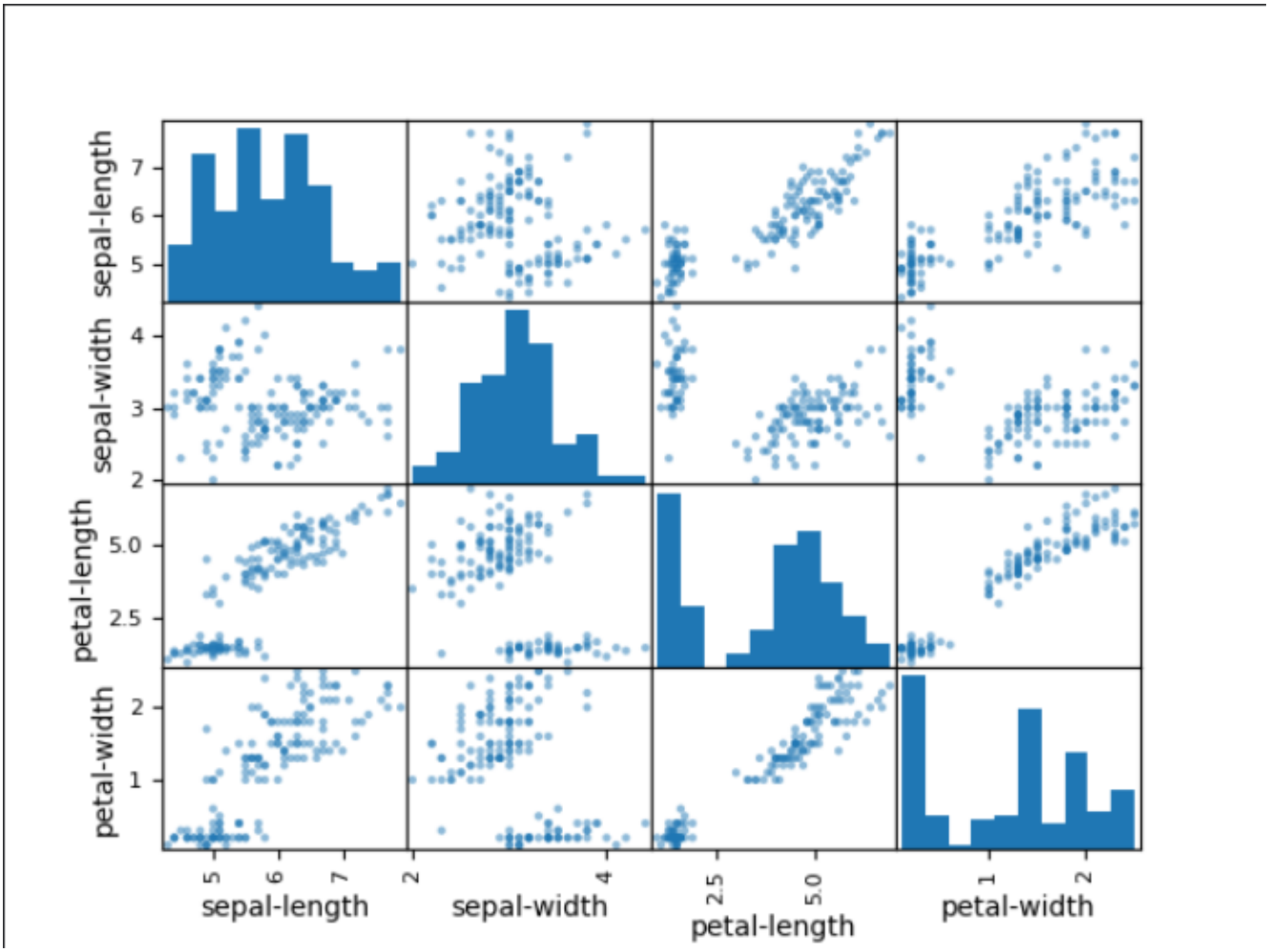


Рис.10 Матриця діаграм розсіювання

Висновок: Квітка належала до класу Iris-setosa

Можна зробити висновок, що модель лінійного дискримінантного аналізу показала найкращі результати в порівнянні з іншими моделями, проте для її роботи знадобилося найбільше ресурсів.

```
# 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 навчальну та контрольну вибірки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,
test_size=0.20, random_state=1)

# Завантажуємо алгоритми моделі
models = [('LR', LogisticRegression(solver='liblinear', multi_class='ovr')),
          ('LDA', LinearDiscriminantAnalysis()),
```

```
          ('KNN', KNeighborsClassifier()),
          ('CART', DecisionTreeClassifier()),
          ('NB', GaussianNB()),
          ('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))

# Вивід результатів за власними даними
X_new = [[2.0, 5.2, 5.7, 1.1], [6.2, 3.9, 3.8, 0.5], [6.91, 2.6, 1.5, 6.3],
         [3.25, 2.2, 4.7, 1.1], [5.0, 1.9, 2.8, 0.2], [3.22, 11.4, 1.2, 4.1]]
predictions = model.predict(X_new)
print(f"X_new: {X_new}\nPredictions: {predictions}")
```

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

Рис. 11 Порівняння accuracy моделей

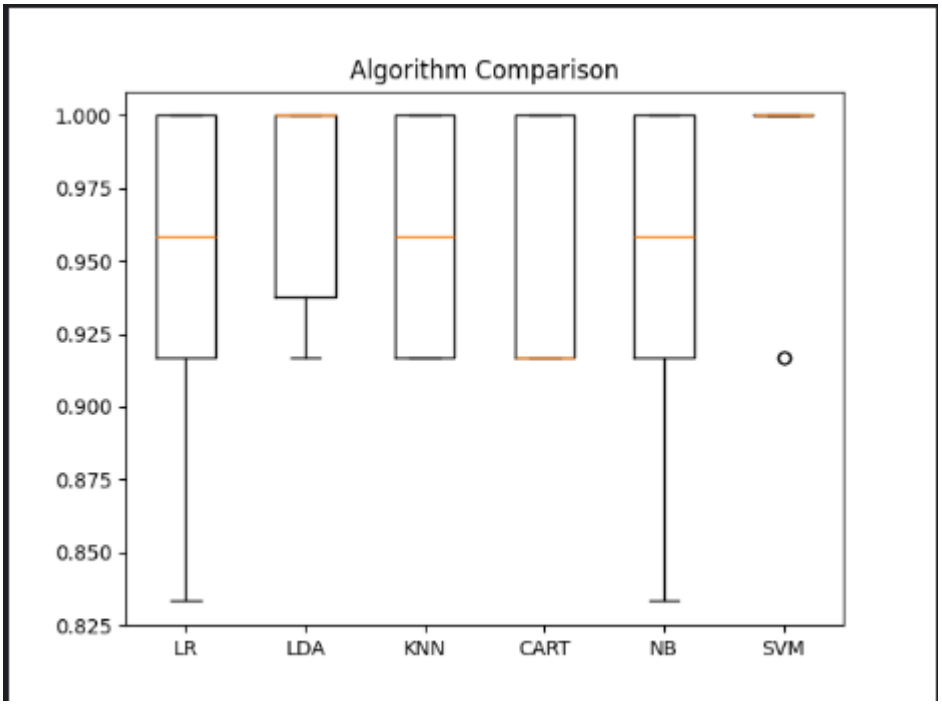


Рис.12 Діаграма розмаху атрибутів вхідних даних

```
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.13 Якість, матриця помилок та звіт по класифікації даних через SVC

```
X_new: [[2.0, 5.2, 5.7, 1.1], [6.2, 3.9, 3.8, 0.5], [6.91, 2.6, 1.5, 6.3], [3.25, 2.2, 4.7, 1.1], [5.0, 1.9, 2.8, 0.2], [3.22, 11.4, 1.2, 4.1]]
Predictions: ['Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica']
```

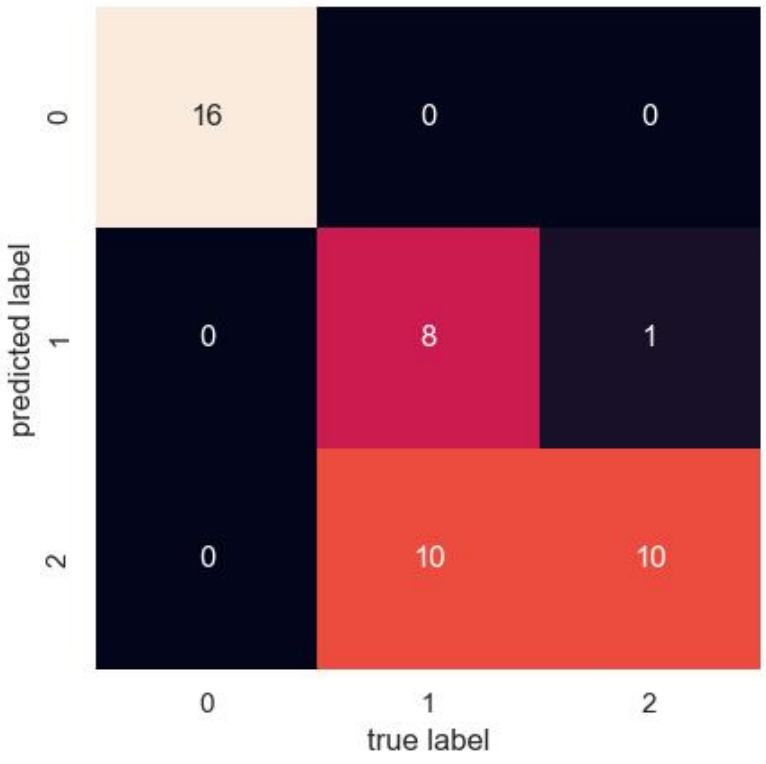
Рис. 14 Прогнозування класів власних даних

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

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

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")
```

```
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.15 Результат виконання завдання 2.5



Висноквок до завдання: У даному коді використовується класифікатор RidgeClassifier з налаштуваннями tol=1e-2 та solver="sag". За результатами тестування, класифікатор показує хороші результати. Точність класифікації становить 96%. Коефіцієнт Коена Каппа дорівнює 0,82, а коефіцієнт кореляції Метьюза дорівнює 0,80. Це означає, що класифікатор значно перевершує випадкові прогнози і існує сильна кореляція між фактичними та прогнозованими класами.

Висновок: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та отримано навички для їх порівняння.

Посилання на репозиторій GitHub: https://github.com/IvanPaliy/A.I.-Lab-2-IPZ-Palii.git