ЛАБОРАТОРНА РОБОТА № 2

ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

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

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

14 ознак набору даних:

1. Age – Вік (числовий)
2. Workclass – Клас зайнятості (категоріальний)
3. Fngwgt (числовий)
4. Education – Рівень освіти (категоріальний)
5. Education-num (числовий)
6. Marital-status – Сімейний стан (категоріальний)
7. Occupation – Професія (категоріальний)
8. Relationship – Відношення в сім’ї (категоріальний)
9. Race – Раса (категоріальний)
10. Sex – Стать (бінарний)
11. Capital-gain – Капітальні прибутки (числовий)
12. Capital-loss – Капітальні втрати (числовий)
13. Hours-per-week – Кількість годин роботи на тиждень (числовий)
14. Native-country – Рідна країна (категоріальний)

Лістинг програми LR_2_task_1:

```
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
# 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)
```

Рис 2.1 – лістинг програми

```python
# Створення SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X, y)

# Поділ на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)

# Створення та тренування SVM-класифікатора
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score (cross-validated): " + str(round(100 * f1.mean(), 2)) + "%")

# Обчислення інших показників
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_test = f1_score(y_test, y_test_pred, average='weighted')
print(f"Accuracy: {round(100 * accuracy, 2)}%")
print(f"Precision: {round(100 * precision, 2)}%")
print(f"Recall: {round(100 * recall, 2)}%")
print(f"F1 score (test set): {round(100 * f1_test, 2)}%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40',
'United-States']

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

# Використання класифікатора для кодованої точки даних та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рис 2.2 – лістинг програми

```
F1 score (cross-validated): 76.01%
Accuracy: 79.56%
Precision: 79.26%
Recall: 79.56%
F1 score (test set): 75.75%
<=50K
```

Рис 2.3 – результат виконання

Тестова точка належить до класу з доходом нижче-рівно 50,000 (з ймовірно близько 79%)

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

```python
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

# 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])
```

Рис 2.4 – лістинг програми

```python
scaler = preprocessing.StandardScaler()
X_encoded = scaler.fit_transform(X_encoded)

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

# Поділ на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)

# Створення та тренування SVM-класифікатора
classifier = SVC(kernel='poly', degree=8, class_weight='balanced')
classifier.fit(X_train, y_train)

y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score (cross-validated): " + str(round(100 * f1.mean(), 2)) + "%")

# Обчислення інших показників
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_test = f1_score(y_test, y_test_pred, average='weighted')
print(f"Accuracy: {round(100 * accuracy, 2)}%")
print(f"Precision: {round(100 * precision, 2)}%")
print(f"Recall: {round(100 * recall, 2)}%")
print(f"F1 score (test set): {round(100 * f1_test, 2)}%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40',
'United-States']

# Кодування тестової точки даних
input_data_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = int(input_data[i])
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([input_data[i]])[0])
        count += 1  # Збільшуємо тільки для категоріальних ознак
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

# Використання класифікатора для кодованої точки даних та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рис 2.5 – лістинг програми

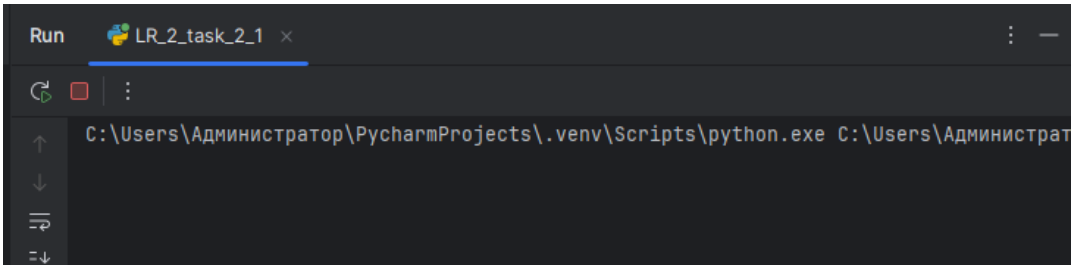

Рис 2.6 – результат виконання

Код так і не виконався.

```
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

# 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)
```

Рис 2.7 – лістинг програми

```python
# Поділ на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=5)

# Створення та тренування SVM-класифікатора
classifier = SVC(kernel='rbf')
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score (cross-validated): " + str(round(100 * f1.mean(), 2)) + "%")

# Обчислення інших показників
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_test = f1_score(y_test, y_test_pred, average='weighted')
print(f"Accuracy: {round(100 * accuracy, 2)}%")
print(f"Precision: {round(100 * precision, 2)}%")
print(f"Recall: {round(100 * recall, 2)}%")
print(f"F1 score (test set): {round(100 * f1_test, 2)}%")

# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
    'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']

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

# Збільшуємо тільки для категоріальних ознак
input_data_encoded = np.array(input_data_encoded).reshape(1, -1)

# Використання класифікатора для кодованої точки даних та виведеннярезультату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рис 2.8 – лістинг програми

```
C:\Users\Администратор\PycharmProjects\.venv\Scripts\pyt
F1 score (cross-validated): 71.95%
Accuracy: 78.19%
Precision: 82.82%
Recall: 78.19%
F1 score (test set): 71.51%
<=50K
```

Рис 2.9 – результат виконання

```python
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

# 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)
# Поділ на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=5)
```

Рис 2.10 – лістинг програми

```python
# Створення та тренування SVM-класифікатора
classifier = SVC(kernel='sigmoid')
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score (cross-validated): " + str(round(100 * f1.mean(), 2)) + "%")
# Обчислення інших показників
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_test = f1_score(y_test, y_test_pred, average='weighted')
print(f"Accuracy: {round(100 * accuracy, 2)}%")
print(f"Precision: {round(100 * precision, 2)}%")
print(f"Recall: {round(100 * recall, 2)}%")
print(f"F1 score (test set): {round(100 * f1_test, 2)}%")
# Передбачення результату для тестової точки даних
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
            'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
# Кодування тестової точки даних
input_data_encoded = [-1] * len(input_data)
count = 0
for i, item in enumerate(input_data):
        if item.isdigit():
            input_data_encoded[i] = int(input_data[i])
        else:
            input_data_encoded[i] = int(label_encoder[count].transform([input_data[i]])[0])
            count += 1  # Збільшуємо тільки для категоріальних ознак

input_data_encoded = np.array(input_data_encoded).reshape(1, -1)
# Використання класифікатора для кодованої точки даних та виведення результату
predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рис 2.11 – лістинг програми

```
F1 score (cross-validated): 63.77%
Accuracy: 60.47%
Precision: 60.64%
Recall: 60.47%
F1 score (test set): 60.55%
<=50K
```

Рис 2.12 – результат виконання

У загальному, класифікатори з лінійним та гаусовим ядром демонструють схожі результати, проте класифікатор з гаусовим ядром зазвичай має значно вищу точність. Якщо основним критерієм є точність, то класифікатор з гаусовим ядром показує найкращі результати у задачах класифікації.

Завдання 2.3. Порівняння якості класифікаторів

```
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'])))
# Виведемо форму масиву data
print("Форма масиву data: {}".format(iris_dataset['data'].shape))
# Виведемо значення ознак для перших п'яти прикладів
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']))
```

Рис 2.13 – лістинг програми для ознайомлення зі структурою даних

```
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)
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 2.14 – результат виконання

```python
# Завантаження бібліотек
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)

# 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()
```

Рис 2.15 – лістинг програми

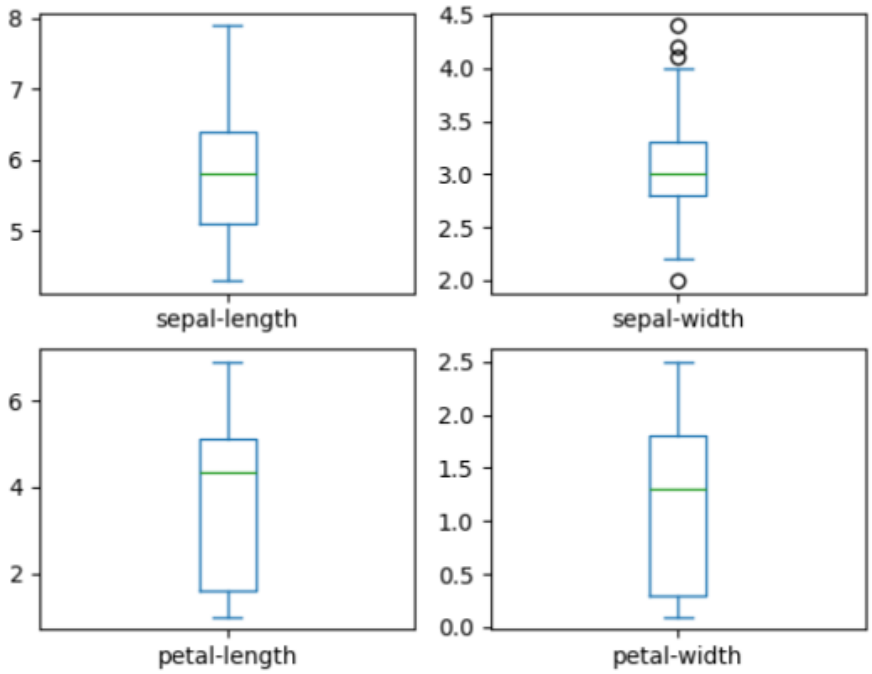

Рис 2.16 – результат виконання

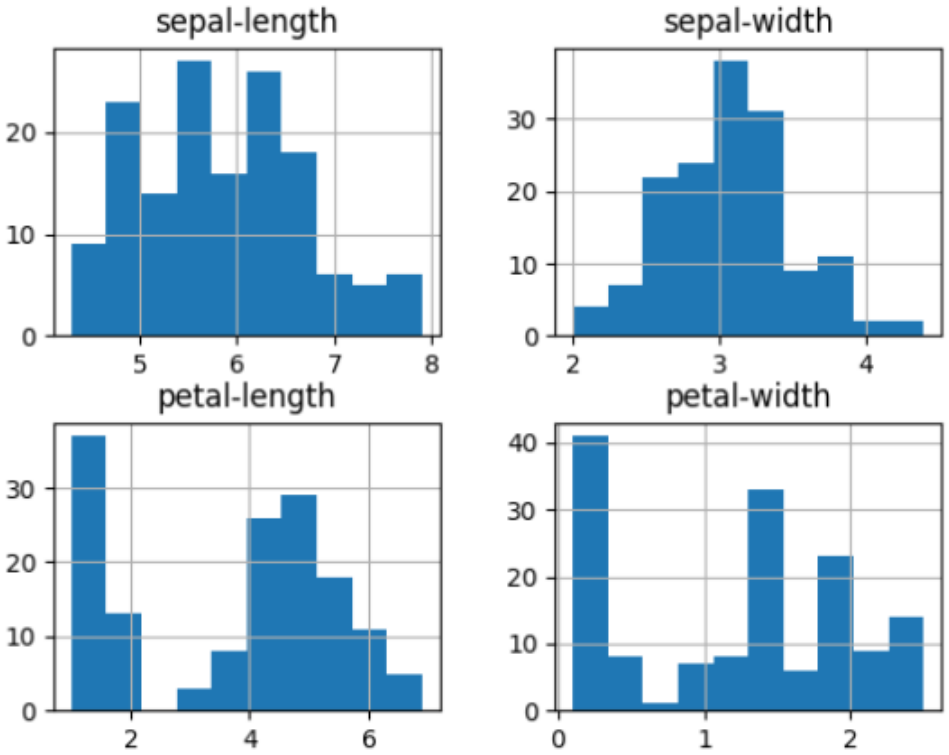

Рис 2.17 – результат виконання

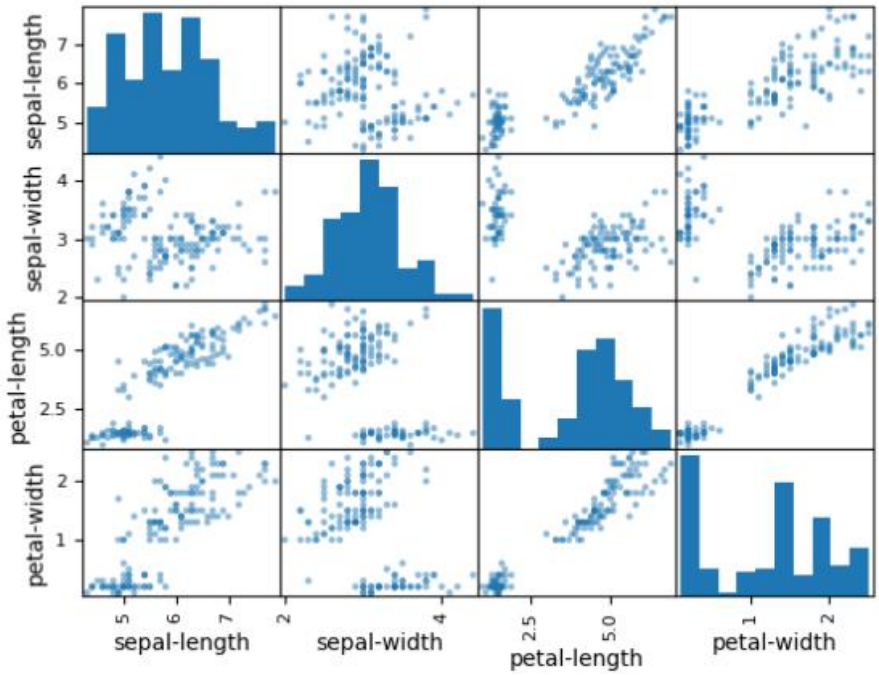

Рис 2.18 – результат виконання

```
C:\Users\Администратор\PycharmProjects\.venv\Scripts\python.exe C:\Users\Админи
(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 2.19 – результат виконання

```
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'])))

# Виведемо форму масиву data
print("Форма масиву data: {}".format(iris_dataset['data'].shape))

# Виведемо значення ознак для перших п'яти прикладів
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']))
# 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)

# Перегляд основних параметрів
print(dataset.shape)
print(dataset.head(20))
print(dataset.describe())
print(dataset.groupby('class').size())

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

# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
X = array[:, 0:4]
y = array[:, 4]

# Розподіл X і y на вибірки
X_train, X_validation, y_train, y_validation = train_test_split(X, y, test_size=0.20, random_state=1)
# Використання алгоритму SVM
model = SVC(gamma='auto')
# Налаштування стратифікованої крос-валідації
stratified_kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
```

Рис 2.20 – лістинг програми

```
# Оцінка моделі з використанням метрики точності
cv_results = cross_val_score(model, X_train, y_train, cv=stratified_kfold, scoring='accuracy')
# Виведення результатів точності
print(f"Середня точність: {cv_results.mean() * 100:.2f}%")
print(f"Стандартне відхилення точності: {cv_results.std() * 100:.2f}%")

# Навчання моделі на всіх навчальних даних
model.fit(X_train, y_train)
# Передбачення на тестовій вибірці
y_pred = model.predict(X_validation)
# Обчислення точності на тестових даних
accuracy = accuracy_score(y_validation, y_pred)
print(f"Точність на тестових даних: {accuracy * 100:.2f}%")
```

Рис 2.21 – лістинг програми

```
# Навчання моделі на всіх навчальних даних
model.fit(X_train, y_train)
# Передбачення на тестовій вибірці
y_pred = model.predict(X_validation)
# Обчислення точності на тестових даних
accuracy = accuracy_score(y_validation, y_pred)
print(f"Точність на тестових даних: {accuracy * 100:.2f}%")
# Завантажуємо моделі
models = []
models.append(('LR', LogisticRegression(solver='liblinear')))

# multi_class removed
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# Оцінка моделей
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(f'{name}: {cv_results.mean():.6f} ({cv_results.std():.6f})')

# Порівняння алгоритмів
pyplot.boxplot(results, tick_labels=names) # Changed to tick_labels
pyplot.title('Algorithm Comparison')
pyplot.show()

# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, y_train)
predictions = model.predict(X_validation)
# Оцінюємо прогноз
print(accuracy_score(y_validation, predictions))
print(confusion_matrix(y_validation, predictions))
print(classification_report(y_validation, predictions))
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

Рис 2.22 – лістинг програми

```
# Створюємо прогноз на нових даних (X_new має бути визначено)
X_new = [[5.0, 2.9, 1.0, 0.2]] # приклад нових даних
prediction = knn.predict(X_new)
# Оскільки prediction є масивом, потрібно використовувати перший елемент
predicted_class = prediction[0]
# Прогноз
print("Прогноз: {}".format(prediction))
# сам прогноз (наприклад, ['setosa'])
print("Спрогнозована мітка: {}".format(predicted_class))
# відображення класу
```

Рис 2.23 – лістинг програми

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
Середня точність: 98.33%
Стандартне відхилення точності: 3.33%
Точність на тестових даних: 96.67%
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.950000 (0.040825)
NB: 0.950000 (0.055277)
SVM: 0.983333 (0.033333)
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

Прогноз: ['Iris-setosa']
Спрогнозована мітка: Iris-setosa
```

Рис 2.24 – результат виконання

Модель SVM продемонструвала високу ефективність класифікації із середньою точністю близько 98%, що вказує на здатність моделі чітко розрізняти класи у наборі даних. Прогноз для нового зразка з характеристиками [5.0, 2.9, 1.0, 0.2] визначає, що квітка належить до класу 'setosa'. Це свідчить про надійність моделі у розпізнаванні видів рослин на основі їхніх ознак.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

```python
import numpy as np
from sklearn import preprocessing
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.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Читання 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[:-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])
```

Рис 2.25 – лістинг програми

```
scaler = preprocessing.StandardScaler()
X_encoded = scaler.fit_transform(X_encoded)
X = X_encoded[:, :-1].astype(int)
y = X_encoded[:, -1].astype(int)

# Поділ на тренувальні та тестові дані
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
# Список моделей для тестування
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Linear Discriminant Analysis': LinearDiscriminantAnalysis(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB(),
    'SVM (Linear)': SVC(kernel='linear', class_weight='balanced'),
    'SVM (Polynomial)': SVC(kernel='poly', degree=8, class_weight='balanced')
}
# Функція для оцінки моделі
def evaluate_model(model, X_train, y_train, X_test, y_test):  1 usage  new *
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')
    return accuracy, precision, recall, f1
# Порівняння моделей
for model_name, model in models.items():
    accuracy, precision, recall, f1 = evaluate_model(model, X_train,
y_train, X_test, y_test)
    print(f"{model_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)}%")
    print('-' * 50)
```

Рис 2.26 – лістинг програми

```
Logistic Regression:
 Accuracy: 79.15%
 Precision: 77.47%
 Recall: 79.15%
 F1 Score: 77.13%
--------------------------------------------------
Linear Discriminant Analysis:
 Accuracy: 78.3%
 Precision: 76.38%
 Recall: 78.3%
 F1 Score: 75.42%
--------------------------------------------------
```

Рис 2.27 – результат виконання

```
K-Nearest Neighbors:
 Accuracy: 75.93%
 Precision: 76.52%
 Recall: 75.93%
 F1 Score: 76.2%
-------------------------------------------------
Decision Tree:
 Accuracy: 79.13%
 Precision: 77.94%
 Recall: 79.13%
 F1 Score: 78.25%
-------------------------------------------------
Naive Bayes:
 Accuracy: 77.67%
 Precision: 76.7%
 Recall: 77.67%
 F1 Score: 72.58%
```

Рис 2.28 – результат виконання

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

```python
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
import seaborn as sns
import matplotlib.pyplot as plt
from io import BytesIO

# Завантаження даних
iris = load_iris()
X, y = iris.data, iris.target

# Розділення на тренувальні та тестові дані
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3, random_state=0)

# Ініціалізація Ridge-класифікатора
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)

# Прогнозування
ypred = clf.predict(Xtest)

# Метрики якості
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(ytest, ypred))
```

Рис 2.29 – лістинг програми

```python
# Матриця плутанини
mat = metrics.confusion_matrix(ytest, ypred)

# Візуалізація матриці плутанини
sns.set()
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True label')
plt.ylabel('Predicted label')

# Збереження графіку
plt.savefig("Confusion.jpg", dpi=300)

# Збереження в SVG
f = BytesIO()
plt.savefig(f, format="svg")
```

Рис 2.30 – лістинг 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 2.31 – результат виконання

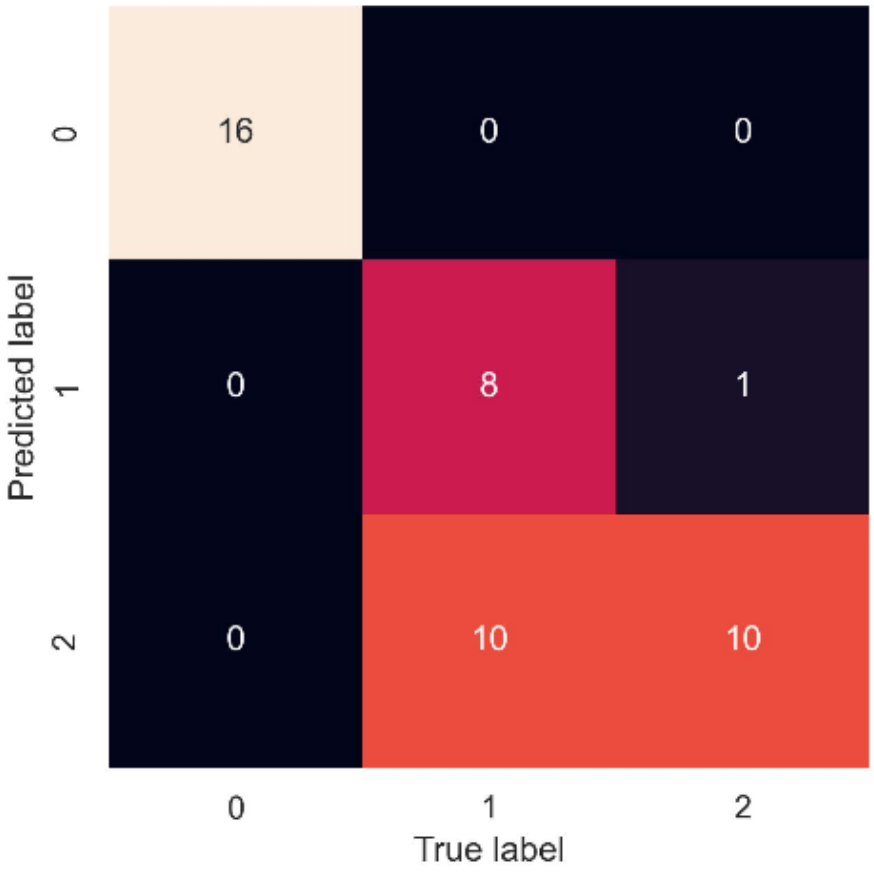

Рис 2.32 – результат виконання

Налаштування Ridge-класифікатора:

Tol=1e-2 – це допустиме значення похибки для зупинки ітерацій. Чим менше значення, тим точніше розв'язок, але час виконання може збільшитися

Solve=`sag` - це алгоритм оптимізації, що працює ефективно для великих наборів даниих

Показники якості:

1. Accuracy – Частка правильно класифікованих зразків (0.7556 або 75.56%).
2. Precision – Відображає частку істинно позитивних результатів серед усіх передбачених як позитивні (0.8333).
3. Recall – Частка правильно передбачених позитивних класів серед усіх фактичних позитивних зразків (0.7556).
4. F1 Score – Гармонійне середнє Precision і Recall (0.7503).
5. Cohen Kappa Score – Статистичний показник узгодженості між реальними та передбаченими значеннями (0.6431).

6. Matthews Correlation Coefficient – Відображає силу зв’язку між реальними та передбаченими значеннями (0.6831).

Пояснення матриці плутанини:

Матриця плутанини показує, як добре модель класифікує дані:

- **Рядки:** Істинні мітки.

- **Стовпці:** Передбачені мітки.

На основі матриці:

- Клас 0 має 100% точність.
- Клас 1 часто плутається з класом 2.
- Клас 2 має високу Recall, але низьку Precision через плутанину з класом 1.

Опис коефіцієнтів:

Cohen Kappa Score: Вимірює угоду між двома наборами класифікацій, скориговану на випадкову угоду. Значення близьке до 1 вказує на високу узгодженість.

Matthews Correlation Coefficient: Розраховує баланс між точністю і повнотою для бінарної або багатокласової класифікації. Значення вищі за 0.5 вказують на добру модель.

Висновок: На лабораторній роботі мала змогу використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати

Посилання на репозиторій:

https://github.com/ipz211/shi_prokopchuk_oleksandra_ipz-21-1_