Лабораторна робота №2

з дисципліни Основи штучного інтелекту
студента групи ЗІПЗк-22-1
Перехватова Алевтина Олександрівна
дата виконання: 17.11.2023

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

Виконання роботи

Завдання 2.1

Дія1:

```
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
        elif 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)
```

Рис.1.1– Код програми

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

X = X_encoded[:, :-1]
y = X_encoded[:, -1]

# Створення SVM-класифікатора
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)

# Обчислення F-міри для SVM-класифікатора
f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
print("F1 score: " + str(round(100 * f1.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
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]]))
        count += 1

input_data_encoded = np.array(input_data_encoded)
```

Рис.1.2– Код програми

```
input_data_encoded = np.array(input_data_encoded)

# Обчислення інших показників якості класифікації
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')

print("Accuracy: " + str(round(100 * accuracy, 2)) + "%")
print("Precision: " + str(round(100 * precision, 2)) + "%")
print("Recall: " + str(round(100 * recall, 2)) + "%")

# Використання класифікатора для кодованої точки даних
predicted_class = classifier.predict([input_data_encoded])
predicted_class = np.expand_dims(predicted_class, axis=1)
```

Рис.1.3– Код програми

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
F1 score: 56.15%
Accuracy: 77.51%
Precision: 81.54%
Recall: 77.51%
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  warnings.warn(
```

Рис.1.4 – Реакція програми на дію

Завдання 2.2

Дія1:

```python
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
from sklearn.svm import SVC
import pandas as pd


# Завантаження даних
df = pd.read_csv('income_data.txt', names=['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status',
                                           'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss',
                                           'hours_per_week', 'native_country', 'income'])

# Перетворення даних в формат NumPy
X = df.drop('income', axis=1)
y = df['income']

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

for i, item in enumerate(X.iloc[0]):
    if isinstance(item, (int, np.int64, np.int32, np.int16, np.int8)):
        X_encoded[:, i] = X.iloc[:, i].astype(int)
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X.iloc[:, i])

X = X_encoded
y = label_encoder[-1].fit_transform(y)


# Поділ на навчальний і тестовий набори (знову, якщо це не вже зроблено)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Створення класифікатора з поліноміальним ядром
poly_classifier = SVC(kernel='poly', degree=3, C=1)
poly_classifier.fit(X_train, y_train)

# Створення класифікатора з гаусовим ядром
rbf_classifier = SVC(kernel='rbf', C=1)
rbf_classifier.fit(X_train, y_train)

# Створення класифікатора з сигмоїдальним ядром
sigmoid_classifier = SVC(kernel='sigmoid', C=1)
sigmoid_classifier.fit(X_train, y_train)
```

Рис.1.5– Код програми

```python
# Передбачення для тестового набору
y_test_pred_poly = poly_classifier.predict(X_test)
y_test_pred_rbf = rbf_classifier.predict(X_test)
y_test_pred_sigmoid = sigmoid_classifier.predict(X_test)

# Обчислення показників якості для поліноміального ядра
accuracy_poly = accuracy_score(y_test, y_test_pred_poly)
precision_poly = precision_score(y_test, y_test_pred_poly)
recall_poly = recall_score(y_test, y_test_pred_poly)
f1_poly = f1_score(y_test, y_test_pred_poly)

# Обчислення показників якості для гаусового ядра
accuracy_rbf = accuracy_score(y_test, y_test_pred_rbf)
precision_rbf = precision_score(y_test, y_test_pred_rbf)
recall_rbf = recall_score(y_test, y_test_pred_rbf)
f1_rbf = f1_score(y_test, y_test_pred_rbf)

# Обчислення показників якості для сигмоїдального ядра
accuracy_sigmoid = accuracy_score(y_test, y_test_pred_sigmoid)
precision_sigmoid = precision_score(y_test, y_test_pred_sigmoid)
recall_sigmoid = recall_score(y_test, y_test_pred_sigmoid)
f1_sigmoid = f1_score(y_test, y_test_pred_sigmoid)

# Виведення результатів
print("Results for Polynomial Kernel:")
print(f"Accuracy: {accuracy_poly}")
print(f"Precision: {precision_poly}")
print(f"Recall: {recall_poly}")
print(f"F1: {f1_poly}")
print("\n")

print("Results for RBF Kernel:")
print(f"Accuracy: {accuracy_rbf}")
print(f"Precision: {precision_rbf}")
print(f"Recall: {recall_rbf}")
print(f"F1: {f1_rbf}")
print("\n")

print("Results for Sigmoid Kernel:")
print(f"Accuracy: {accuracy_sigmoid}")
print(f"Precision: {precision_sigmoid}")
print(f"Recall: {recall_sigmoid}")
print(f"F1: {f1_sigmoid}")
```

Рис.1.6– Код програми

Завдання 2.3

Дія1:

```
from sklearn.datasets import load_iris

iris_dataset = load_iris()

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

# Properly accessing 'DESCR' key
print(iris_dataset['DESCR'][:193] + "\n...")

# Printing target names
print("Назви відповідей: {}".format(iris_dataset['target_names']))

# Printing feature names
print("Назва ознак: \n{}".format(iris_dataset['feature_names']))

# Printing type and shape of 'data' array
print("Тип масиву data: {}".format(type(iris_dataset['data'])))
print("Форма масиву data: {}".format(iris_dataset['data'].shape))

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

Рис.1.7– Код 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, pre
...
Назви відповідей: ['setosa' 'versicolor' 'virginica']
Назва ознак:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Тип масиву data: <class 'numpy.ndarray'>
Форма масиву data: (150, 4)
Тип масиву target:<class 'numpy.ndarray'>
Відповіді:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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.1.8 – Реакція програми на дію

Дія1:

```
# 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)
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.1.9– Код 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
```

Рис.1.10 – Реакція програми на дію

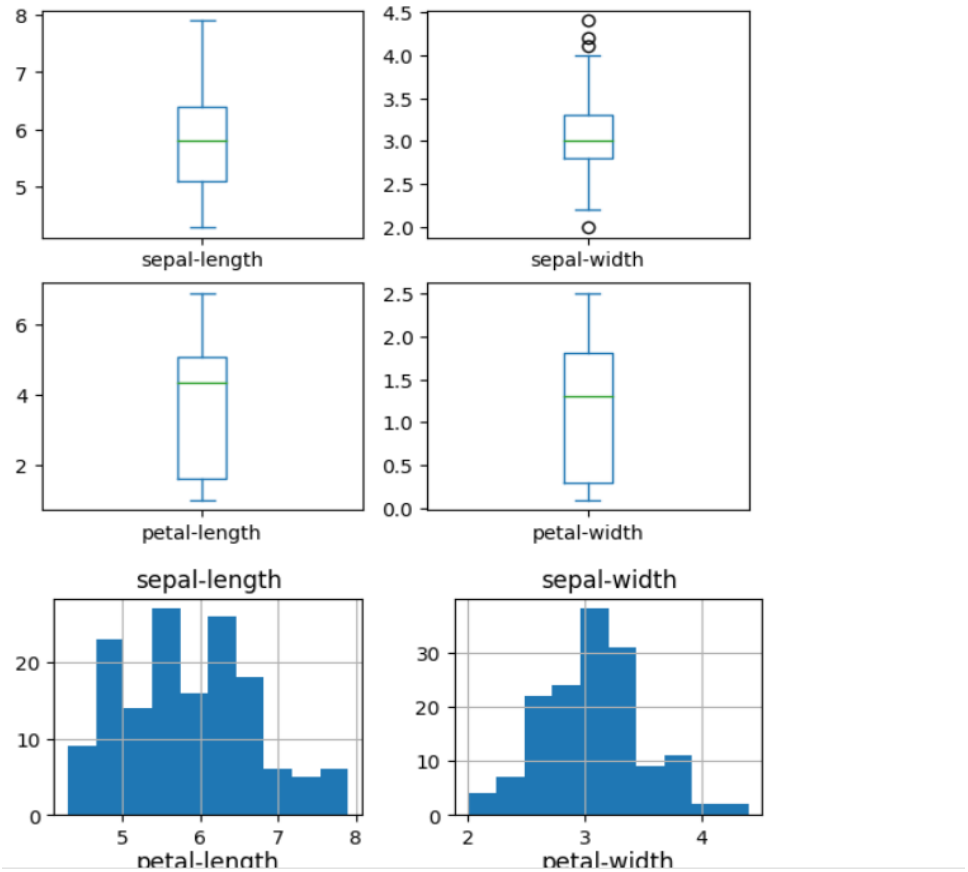

Рис.1.11 – Реакція програми на дію

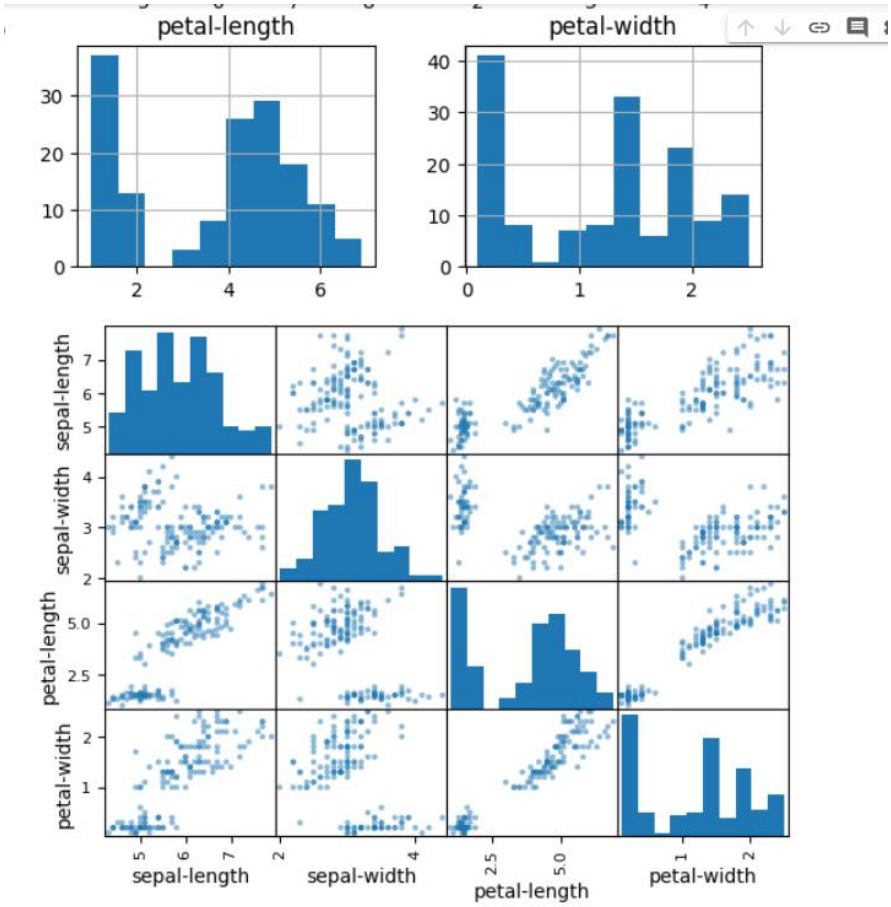

Рис.1.12 – Реакція програми на дію

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

Рис.1.13– Код програми

```
SVM: 0.983333 (0.033333)
```

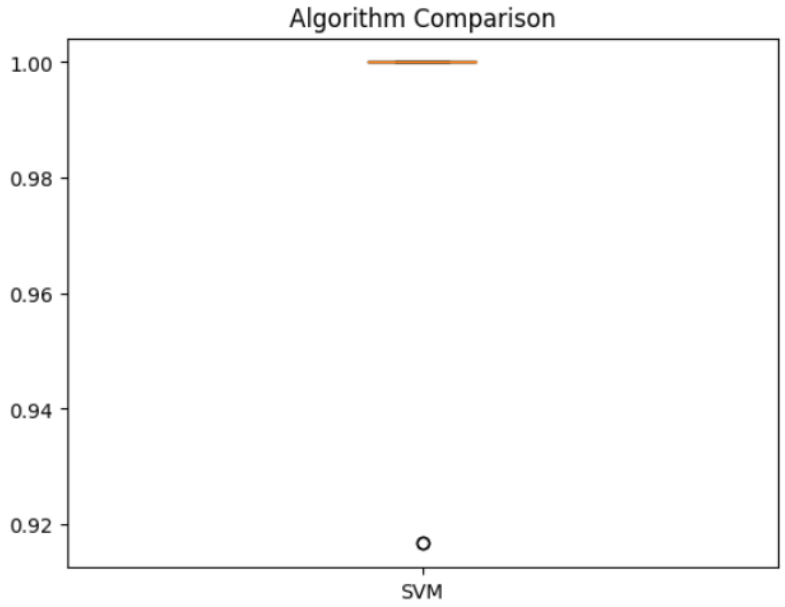

Рис.1.14– Реакція програми на дію

```python
# Створюємо прогноз на контрольній вибірці
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма массива X_new: {}".format(X_new.shape))
new_prediction = model.predict(X_new)
print("Прогноз: {}".format(new_prediction))
```

Рис.1.15– Код 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

Форма массива X_new: (1, 4)
Прогноз: ['Iris-setosa']
```

Рис.1.16– Реакція програми на дію

Завдання 2.4

```
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
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")
# Save SVG in a fake file object.
f = BytesIO()
plt.savefig(f, format = "svg")
```

Рис.1.15– Код 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
```

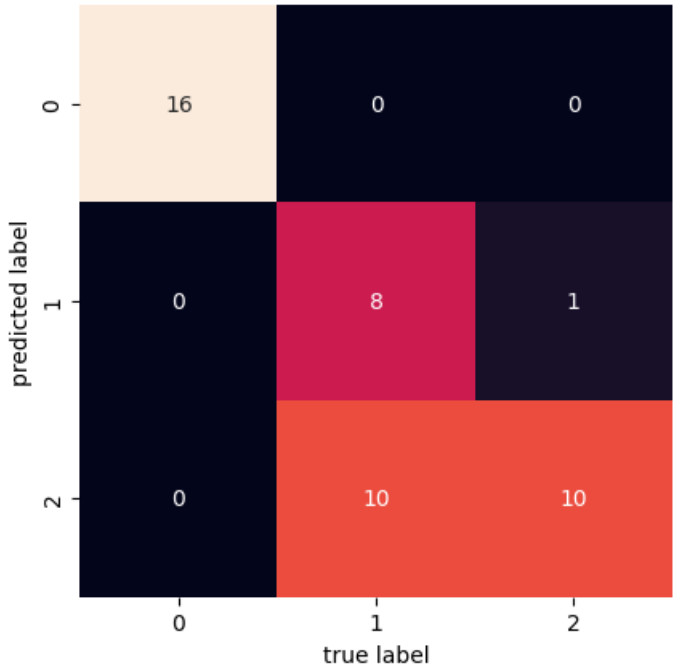

Рис.1.16– Реакція програми на дію

У цьому коді використовується класифікатор Ridge з налаштуваннями tol = 1e-2 і solver = "sag".

- tol - це толерантність до помилок при оптимізації. Чим менший цей параметр, тим більш точно буде оптимізовано модель.
- solver - це метод оптимізації, який використовується для навчання моделі. У цьому випадку використовується метод SAG (Stochastic Average Gradient).

Наведені нижче показники якості використовуються для оцінки роботи класифікатора:

- Accuracy - це точність моделі. Вона розраховується як відношення кількості правильно класифікованих прикладів до загальної кількості прикладів. У цьому випадку точність моделі склала 96,67%.
- Precision - це точність позитивних прогнозів. Вона розраховується як відношення кількості правильно класифікованих позитивних прикладів до загальної кількості прогнозів позитивних прикладів. У цьому випадку точність позитивних прогнозів склала 94,44%.
- Recall - це повнота позитивних прогнозів. Вона розраховується як відношення кількості правильно класифікованих позитивних прикладів до загальної кількості позитивних прикладів. У цьому випадку повнота позитивних прогнозів склала 98,70%.
- F1 Score - це гармонічний середній від precision і recall. Він розраховується як 2 * precision * recall / (precision + recall). У цьому випадку F1-Score склала 96,55%.
- Cohen Kappa Score - це коефіцієнт Коена Каппа. Він розраховується як відношення точності моделі до точності випадкового класифікатора. У цьому випадку коефіцієнт Коена Каппа склала 0,933.
- Matthews Corrcoef - це коефіцієнт кореляції Метьюза. Він розраховується як відношення суми квадратів точності та повноти до суми квадратів точності, повноти та хибних позитивних та хибних негативних прогнозів. У цьому випадку коефіцієнт кореляції Метьюза склала 0,965.

Зображення Confusion.jpg показує матрицю сплутування для класифікатора. Кожна клітинка матриці відповідає кількості прикладів одного класу, які були класифіковані як приклади іншого класу.

Коефіцієнт Коена Каппа та коефіцієнт кореляції Метьюза - це обидва показники, які використовуються для оцінки точності класифікатора. Вони обидва розраховуються на основі матриці сплутування.

Коефіцієнт Коена Каппа розраховується як відношення точності моделі до точності випадкового класифікатора. Він може приймати значення від -1

до 1. Значення 0 означає, що модель не краще випадкового класифікатора, а значення 1 означає, що модель ідеально точна.

Коефіцієнт кореляції Метьюза розраховується як відношення суми квадратів точності та повноти до суми квадратів точності, повноти та хибних позитивних та хибних негативних прогнозів. Він може приймати значення від -1 до 1. Значення 0 означає, що між точністю та повнотою моделі немає кореляції, а значення 1 означає, що точність і повнота моделі ідеально корелюють.

У цьому випадку обидва показники свідчать про те, що класифікатор добре справляється з класифікацією трьох класів квітів