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.

GitHub: https://github.com/ingaliptn/AI

Хід роботи:

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

Код програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
input_file = "income_data.txt"
X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000
with open(input_file, "r") as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1
X = np.array(X)
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
```

```
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
= train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted',
cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners',
'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])
```

Результат виконання програми зображено на рисунку 1.

```
Accuracy: 81.95%
Precision: 80.94%
Recall: 81.95%
F1: 80.13%
F1 score: 80.13%
>50K

Process finished with exit code 0
```

Рис. 1 Результат виконання програми

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

```
Accuracy 83.99%
Precision: 83.21%
Recall: 83.99%
F1: 82.99%
F1 score: 82.99%
<=50K

Process finished with exit code 0
```

Рис. 2 Поліноміальне ядро

```
Accuracy: 83.96%
Precision: 83.18%
Recall: 83.96%
F1: 82.95%
F1 score: 82.95%
<=50K

Process finished with exit code 0
```

Рис. 3 Гаусове ядро

```
Accuracy: 57.26%
Precision: 57.1%
Recall: 57.26%
F1: 57.18%
F1 score: 57.18%
<=50K

Process finished with exit code 0
```

Рис. 4 Сигмоїдальне ядро

RFB дає хороший результат, але менш точний перед поліноміальним ядром. Його перевага – швидкодія. Сигмоїдальне ядро дає більш низький результат. Для нашого випадку кращим буде RFB.

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Лістинг програми:

```
from sklearn.datasets import load_iris
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
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("Назви ознак: \n{}".format(iris_dataset["feature_names"]))
print("Тип масиву date: {}".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']))
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()
# 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_state=1)
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
# Порівняння алгоритмів
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
# 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]])
for name, model in models:
    model.fit(X_train, Y_train)
    prediction = model.predict(X_new)
    print("Прогноз: {}".format(prediction))
    print(accuracy_score(Y_validation, predictions))
    print(confusion_matrix(Y_validation, predictions))
    print(classification_report(Y_validation, predictions))
```

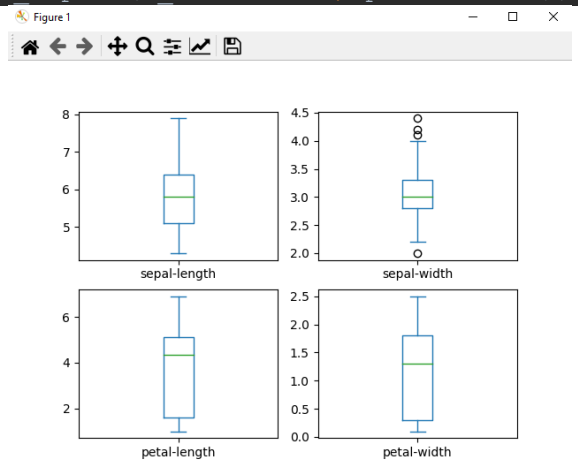

Рис. 5 Результат діаграми розмаху

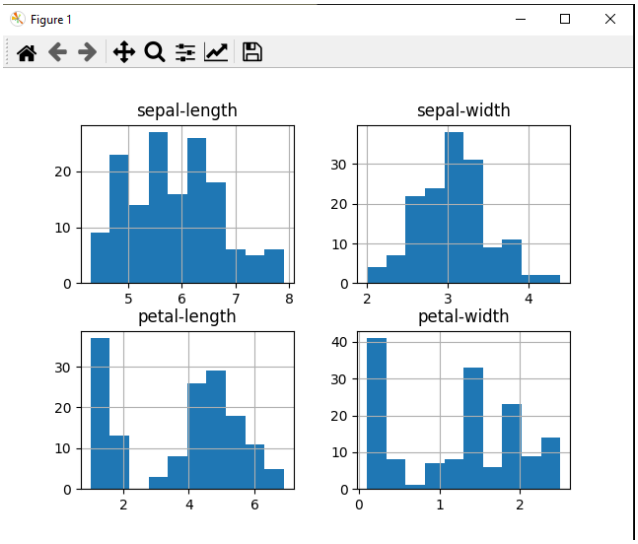

Рис. 6 Гістрограма розподілу атрибутів

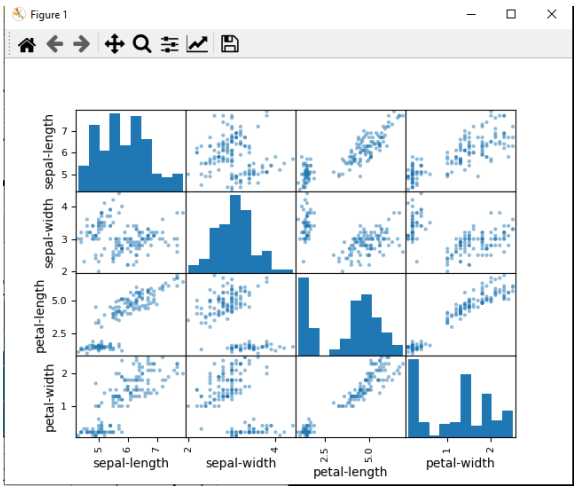

Рис. 7 Матриця діаграми розсіювання

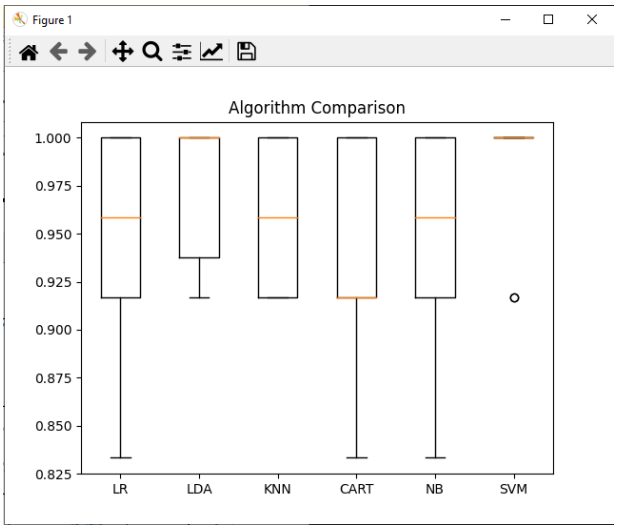

Рис. 8 Рисунок порівняння алгоритмів

```
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 date: <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 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 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
 2 2]
(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
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. 9 Результат програми

Квітка належала до класу Iris-setosa.

З діаграм можемо зробити висновок, що найкраще показала себе модель лінійного дискримінантного аналізу.

Завдання 2.4. Порівняння якості класифікаторів для набору даних завдання 2.1

Лістинг програми:

```
import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import cross_val_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

input_file = "income_data.txt"
X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000
with open(input_file, "r") as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(', ')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1
X = np.array(X)
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)

#classifier = LogisticRegression(solver='liblinear', multi_class='ovr')

#classifier = LogisticRegression(solver='liblinear', multi_class='ovr')
#classifier = LinearDiscriminantAnalysis()
#classifier = KNeighborsClassifier()
#classifier = DecisionTreeClassifier()
#classifier = GaussianNB()
classifier = SVC(gamma='auto')

classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
    = train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(
```

```
    classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married',
'Handlers-cleaners',
              'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])
```

```
Accuracy: 81.82%
Precision 80.69%
Recall: 81.82%
F1: 80.25%
F1 score: 80.25%
>50K
```

Рис.10 Точність класифікатора LR

```
Accuracy: 81.14%
Precision 79.86%
Recall: 81.14%
F1: 79.35%
F1 score: 79.35%
>50K
```

Рис. 11 Точність класифікатора LDA

```
Accuracy: 82.16%
Precision 81.53%
Recall: 82.16%
F1: 81.75%
F1 score: 81.75%
<=50K
```

Рис. 12 Точність класифікатора KNN

```
Accuracy: 80.55%
Precision 80.76%
Recall: 80.66%
F1: 80.84%
F1 score: 80.77%
>50K
```

Рис. 13 Точність класифікатора CART

```
Accuracy: 79.76%
Precision 78.2%
Recall: 79.76%
F1: 77.13%
F1 score: 77.13%
<=50K
```

Рис. 14 Точність класифікатора NB

```
Accuracy: 82.38%
Precision 81.51%
Recall: 82.38%
F1: 80.6%
F1 score: 80.6%
>50K
```

Рис. 15 Точність класифікатора SVM

Завдання 2.5. Класифікація даних лінійним класифікатором Ridge

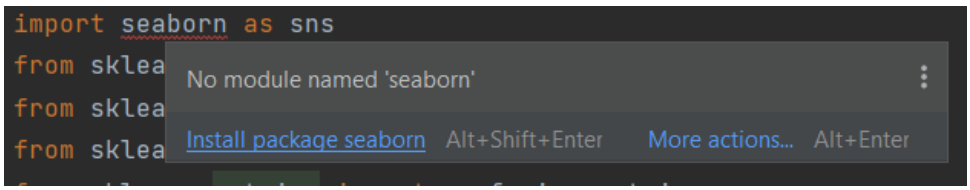

Рис 16. Інсталюємо seaborn

Лістинг програми:

```
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


Process finished with exit code 0
```

Рис. 16 Результат виконання

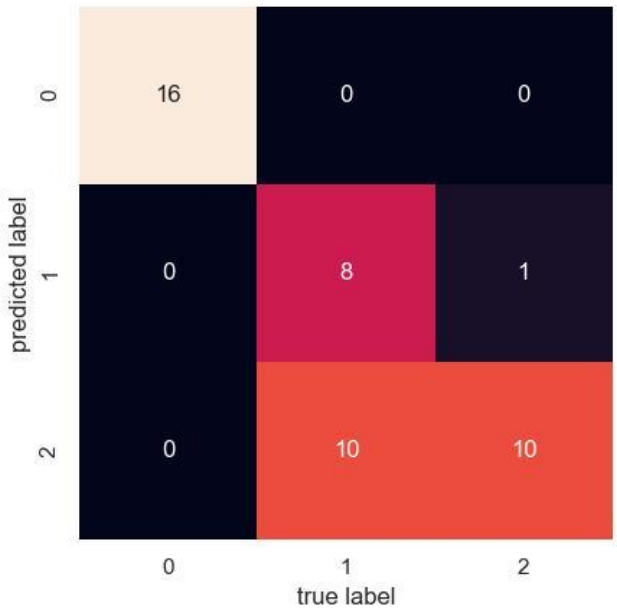

Рис. 17 Матриця невідповідності

З отриманого результату видно, що було отримано r1, recall, коеф. Коена Каппа – це стат. значення, що вимірює міжрегіональну згоду на категоріальні предметі і вважається більш надійнішим аніж розрахунок у відсотках. Також було отримано коеф. кореляції Метьюза – використовується в машинному навчанні, як міра якості бінарних мультикласних класифікацій.

Матриця невідповідності – це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання. Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу.

Висновки: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.