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.

Хід роботи

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

```
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, 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)

classifier = OneVsOneClassifier(LinearSVC(random_state=0))

classifier.fit(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)

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)

# 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. 1.1 Код програми

Рис. 1.2 Результат виконання програми

Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними ядрами

Ядрами - це найкраще підходить для нашого завдання.

1. Поліноміальне ядро. У разі поліноміального ядра ви також повинні передати значення для параметра degree класу SVC. Це переважно ступінь многочлена. Фактично у попередньому коді вам необхідно замінити лінійний параметр на: KernelSVC(kernel='poly', degree=8):

Не забудьте імпортувати відповідну функцію з бібліотеки. Вся решта коду повинна працювати.

2. Гаусове ядро. Ми можемо використовувати гаусове ядро для реалізації kernel SVM: KernelSVC(kernel='rbf'). Щоб використовувати ядро Гауса, ви повинні вказати 'rbf' як значення параметра ядра класу SVC.

3. Сигмоїдальне ядро. Щоб використовувати сигмоїдальне ядро, ви повинні вказати 'sigmoid' як значення для параметра kernel класу SVC .

```
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import LinearSVR
from sklearn.multiclass import OneVsOneClassifier



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)


classifier = OneVsOneClassifier(LinearSVR(random_state=0))


classifier.fit(X, y)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
classifier = OneVsOneClassifier(LinearSVR(random_state=0))
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)


f1 = cross_val_score(classifier, X, y, scoring='f1_weighted', cv=3)
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 = [-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)

predicted_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicted_class)[0])
```

Рис. 1.3 Код програми

```
F1 score: 66.01%
```

Рис. 1.4 Результат виконання програми

Завдання 2.3. Порівняння якості класифікаторів на прикладі класифікації сортів ірисів.

```
from pandas import read_csv
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

# Load dataset
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)

# print(dataset.head(20))
# descriptions
print(dataset.describe())

# classdistribution
print(dataset.groupby('class').size())

# boxand whisker plots
# dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
# pyplot.show()

# histograms
# dataset.hist()
# pyplot.show()
```

```
# scatter plot matrix
# scatter_matrix(dataset)
# pyplot.show()

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

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')))
# evaluate each model in turn
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()))

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

# Make predictions on validation dataset
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Evaluate predictions
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

from sklearn.neighbors import KNeighborsClassifier
import numpy as np
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train, Y_train)
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма массива X_new: {}".format(X_new.shape))
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))


y_pred = knn.predict(X_validation)
print("Прогнозы для тестовго набора:\n {}".format(y_pred))

print("Правильность на тестовом наборе: {:.2f}".format(np.mean(y_pred == Y_validation)))

print("Правильность на тестовом наборе: {:.2f}".format(knn.score(X_validation, Y_validation)))
```

Рис. 1.5 Код програми

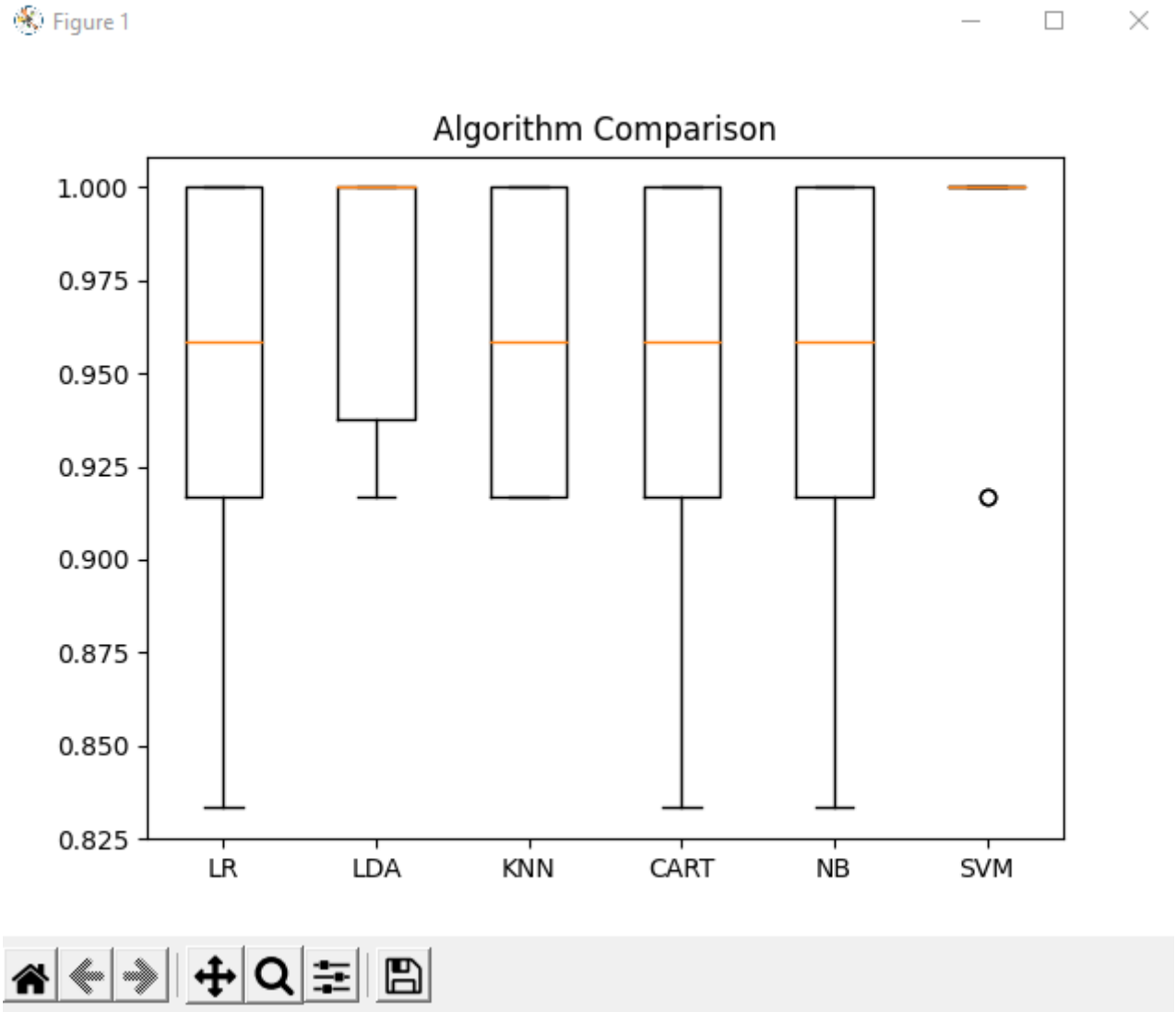

Рис. 1.6 Результат виконання програми

```
E:\ЖДУ\Python\Work_Drobotun_2\venv\Scripts\python.exe E:/ЖДУ/Python/Work_Dr
(150, 5)
       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
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)
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
```

Рис. 1.7 Результат виконання програми

```
                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. 1.8 Результат виконання програми

ОТРИМАННЯ ПРОГНОЗУ (ЗАСТОСУВАННЯ МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ)

```
from pandas import read_csv
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

# Load dataset
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)

# print(dataset.head(20))
# descriptions
print(dataset.describe())

# classdistribution
print(dataset.groupby('class').size())

# boxand whisker plots
# dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
# pyplot.show()

# histograms
# dataset.hist()
# pyplot.show()

# scatter plot matrix
# scatter_matrix(dataset)
# pyplot.show()

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

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')))
```

```
# evaluate each model in turn
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()))

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

# Make predictions on validation dataset
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)

# Evaluate predictions
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

from sklearn.neighbors import KNeighborsClassifier
import numpy as np
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train, Y_train)
X_new = np.array([[5, 2.9, 1, 0.2]])
print("Форма массива X_new: {}".format(X_new.shape))
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))


y_pred = knn.predict(X_validation)
print("Прогнозы для тестовго набора:\n {}".format(y_pred))

print("Правильность на тестовом наборе: {:.2f}".format(np.mean(y_pred == Y_validation)))

print("Правильность на тестовом наборе: {:.2f}".format(knn.score(X_validation, Y_validation)))
```

Рис. 1.9 Код програми

```
Форма массива X_new: (1, 4)
Прогноз: ['Iris-setosa']
Прогнозы для тестовго набора:
 ['Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica']
Правильность на тестовом наборе: 1.00
Правильность на тестовом наборе: 1.00
```

Рис. 2.1 Результат виконання програми

Завдання 2.4. Порівняння якості класифікаторів для набору даних

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split

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)
from sklearn import metrics
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))
from sklearn.metrics import confusion_matrix
from io import BytesIO #neded for plot
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
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. 2.2 Код програми

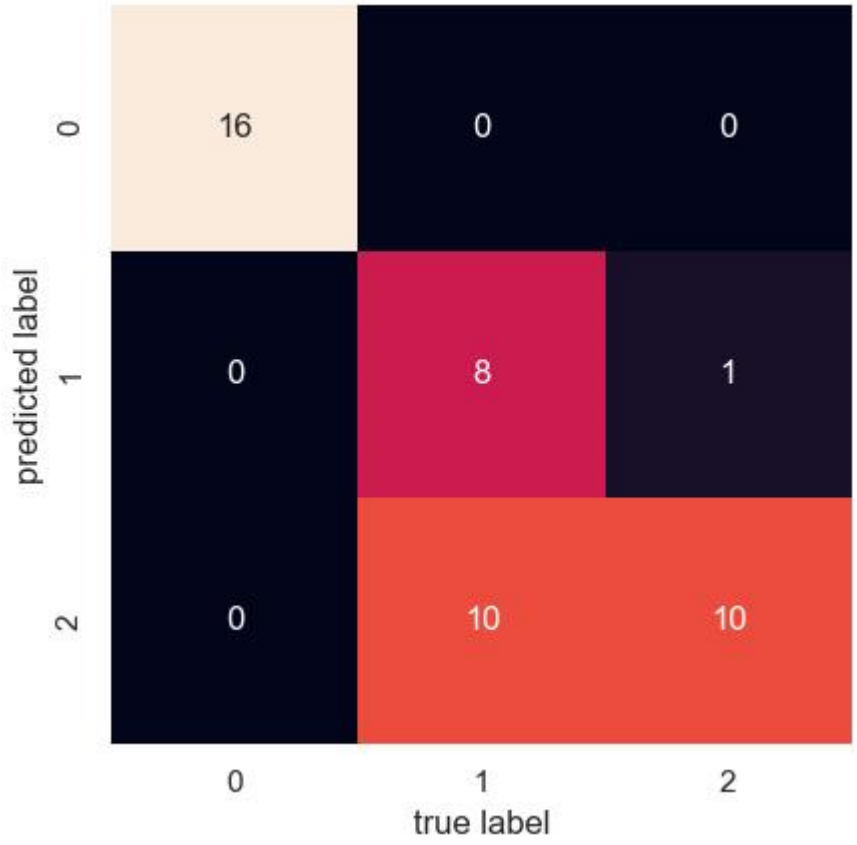

Рис. 2.3 Результат виконання програми

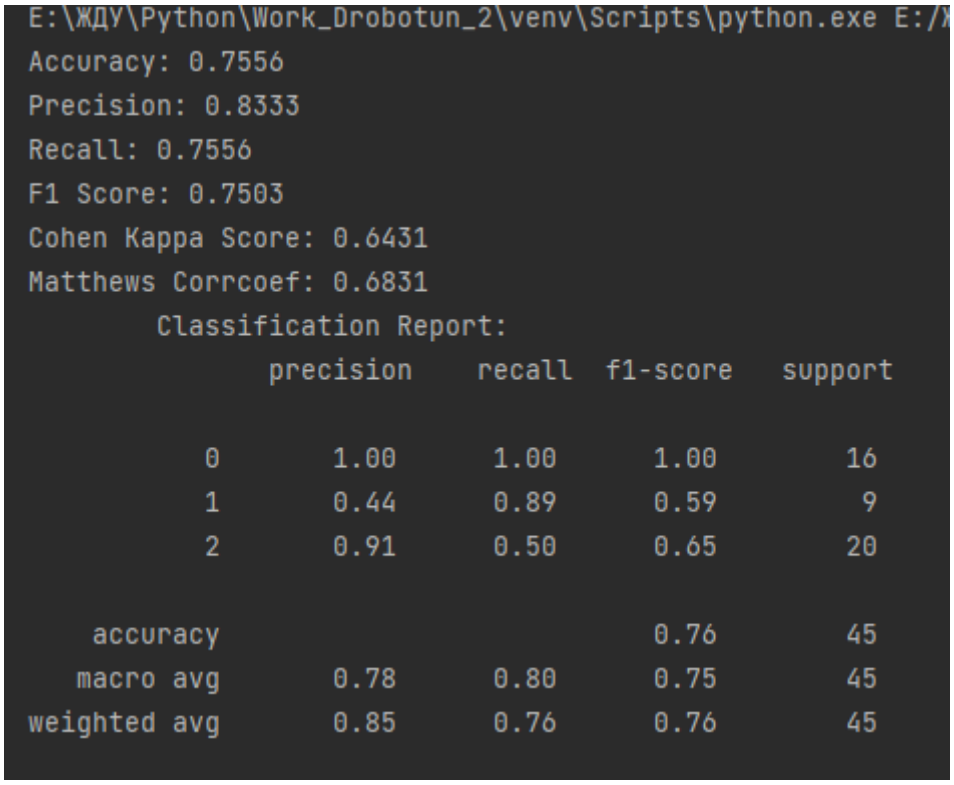

Рис. 2.4 Результат виконання програми

Висновок: я використовував спеціалізовані бібліотеки та мову програмування Python дослідивши різні методи класифікації даних та навчився їх порівнювати.

Основна ідея лінійного класифікатора полягає в тому, що ознаковий простір може бути розділений гіперплощиною на дві напівплощини, у кожній з яких прогнозується одне з двох значень цільового класу. Якщо це можна зробити без помилок, то навчальна вибірка називається лінійно розділеною.

Використовує лінійний класифікатор Ridge за допомогою API бібліотеки scikit-learn. Набір даних Iris класифікується за допомогою лінійного класифікатора Ridge. Розраховуються показники якості.

Коефіцієнт Каппа Коена це статистика, яка вимірює міжрегіональну згоду на якісні (категоріальні) предмети. Зазвичай вважається, що це надійніший захід, ніж простий розрахунок угоди про відсотки, оскільки k враховує випадкову угоду.

Model.fit () друкує – імовірно відповідно до метрики = ['matthews_correlation'] – прогрес і коефіцієнт кореляції Matthews (MCC). Але вони сильно відрізняються від того, що зрештою повертає MCC. Функція MCC в кінці дає загальний MCC прогнозу та узгоджується з функцією MCC sklearn (тобто я довіряю значенню).