DATA ANALYTICS

DECISION TREE BASED CLASSIFIER MODEL BUILDING USING CART ALGORITHM

DATA ANALYTICS PROJECT

Group ID: 85

Project ID: 10

GAUTAM SHARMA

16EC10019

GURSHARAN AHIR

16EC10021

Introduction

Decision Tree is one of the most powerful and popular tools for classification and prediction. A Decision Tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

A tree can be *"learned"* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called *recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. Decision tree induction is a typical inductive approach to learn knowledge on classification.

An instance is classified by starting at the root node of the tree,testing the attribute specified by this node,then moving down the tree branch corresponding to the value of the attribute as shown in the above figure.This process is then repeated for the subtree rooted at the new node.

Gini impurity

Used by the CART (classification and regression tree) algorithm for classification trees, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity can be computed by summing the probability p_i being chosen times the probability

$$\sum_{k \neq i} p_k = 1 - p_i$$

of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

G(D) measures the “impurity” of data set D.The smallest value of G(D) is zero which it takes when all the classifications are the same.

It takes its largest value =1-1/k when the classes are evenly distributed between tuples, that is the frequency of each class is 1/k.

IMPLEMENTATION

Implementation is done as a binary tree implementation where two possible branches are true or false. At each node, Gini Index is calculated along with the split of the possible values of an attribute to find the best possible split and thus identify the attribute which reduces the impurity.

Part (A): After taking a look at the given data, we observed that it was well structured but there were many instances of missing values or '?' as the values. So preprocessing the data was required to get rid of these values. As most of the features were categorical, so we used LabelEncoder to get rid of the '?' values. To effectively train the model we shuffled the data and different functions for classifier were made.

Part (B): After making the required functions for decision tree two functions for k-Fold cross validation and bootstrap estimation functions were made. Scikit learn was "not" used for any of these functions.

Part (C): Different metrics were used to analyze the performance of the decision tree.

k-Fold cross validation code :

```
def kfold(data,k):


    X= shuffle(data,random_state=42)
    X=X.to_numpy()
    n = len(data)/k
    if(n>int(n)):
        n= (int(n)+1)
    trainingData = X[0:n*(k-1)]
    test = X[n*(k-1):len(data)]

    testData = test[:,:test.shape[1]-1]
    y_test = test[:,test.shape[1]-1:]
    decisionTree = growDecisionTreeFrom(trainingData, evaluationFunction=gini)
    prune(decisionTree, 0.8, notify=True)
    count=0
    count1=0
    true=[]
    pred=[]
    for i in range(testData.shape[0]):
        count1 +=1
        t = classify(testData[i], decisionTree)
        for key, value in t.items():
                #print(key,y_test[i])
            pred.append(key)
            true.append(y_test[i])
            if(key==y_test[i]):
                count +=1

    print("Predictive accuracy for k = ",k," is ",count/count1)
```

BootStrap Estimation Function:

```
def bootstrap(data,n):
    data = data.to_numpy()
    for j in range(n):
        trainingData = resample(data,n_samples=250)
        testData = resample(data,n_samples=50)
        y_test = testData[:,testData.shape[1]-1:]
        testData = testData[:,:testData.shape[1]-1]
        decisionTree = growDecisionTreeFrom(trainingData, evaluationFunction=gini)
        prune(decisionTree, 0.8, notify=True)
        count=0
        count1=0
        true=[]
        pred=[]
        for i in range(testData.shape[0]):
            count1 +=1
            t = classify(testData[i], decisionTree)
            for key, value in t.items():
                #print(key,y_test[i])
                pred.append(key)
                true.append(y_test[i])
                if(key==y_test[i]):
                    count +=1

        print("Predictive accuracy for Bootstrap = ",j+1," is ",count/count1)
```

K-fold Cross validation results with confusion matrix, Precision, Recall, F1-score, Accuracy for k=2,3,4

```
Predictive accuracy for k =  2  is  0.8431372549019608
[[17  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0]
 [ 0  4  1  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  2  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  1  1  3  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 15  0  0  0  0  0  0  0  0  0  4  0  0  0  0]
 [ 0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  2  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0]
 [ 2  0  0  0  2  1  0  0  0  0  0 17  0  0  0  0  0  0  0]
 [ 0  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  4  0  0  0  0  0  0  0  0  5  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 22  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  7]]
precision =  0.7990788162031488 Recall =  0.7867007586305832  F1-score =  0.7893938888391019
Predictive accuracy for k =  3  is  0.8910891089108911
[[13  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  0  3  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  9  0  0  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  2  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  1  1  0  0  0  0  0 12  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0  2  0  0  0  0  0  0  0  0  2  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 16  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4]]
precision =  0.9095933387600054 Recall =  0.8682539682539682  F1-score =  0.857896787308552
Predictive accuracy for k =  4  is  0.9210526315789473
[[ 7  0  0  0  4  0  0  0  0  0  0  2  0  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0]
 [ 0  0  2  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0]
 [ 1  0  0  0  0  1  0  0  0  0  0 10  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  0  0]
 [ 0 11  0  0  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3]]
precision =  0.8726851851851852 Recall =  0.8447293447293447  F1-score =  0.8344622172208379
```

K-fold Cross validation results with confusion matrix, Precision, Recall, F1-score, Accuracy for k=5,6,7

```
Predictive accuracy for k =  5  is  0.8813559322033898
[[ 5  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0]
 [ 0  3  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0]
 [ 0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0]
 [ 1  0  0  0  1  1  0  0  0  0  0  5  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2]]
precision =  0.9314814814814815 Recall =  0.8921296296296296  F1-score =  0.8954345037678372
Predictive accuracy for k =  6  is  0.8936170212765957
[[4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0 0 0 0 4 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 1 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 7 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8885802469135802 Recall =  0.8703703703703703  F1-score =  0.86413139329806
Predictive accuracy for k =  7  is  0.9069767441860465
[[4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0 0 0 4 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]]
```

K-fold Cross validation results with confusion matrix, Precision, Recall, F1-score, Accuracy for k=8,9,10,11

```
Predictive accuracy for k =  8  is  0.9411764705882353
[[3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 0 0 4 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 4 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 0 0 4 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8755555555555555 Recall =  0.8666666666666667  F1-score =  0.8548148148148148
Predictive accuracy for k =  9  is  0.9629629629629629
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 0 4 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 3 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8680555555555555 Recall =  0.8541666666666666  F1-score =  0.8460317460317461
Predictive accuracy for k =  10  is  0.9642857142857143
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 4 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 3 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8782051282051282 Recall =  0.8653846153846154  F1-score =  0.8578754578754579
Predictive accuracy for k =  11  is  0.9629629629629629
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 4 0 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 3 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 2 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8680555555555555 Recall =  0.8541666666666666  F1-score =  0.8460317460317461
```

BootStrap Estimation

```
Predictive accuracy for Bootstrap =  1  is  0.96
[[6 0 0 0 1 0 0 0 0 0 0 2 0 2 0 0 0 0]
 [0 3 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0]
 [0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [2 0 0 0 4 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 5 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.810978835978836 Recall =  0.8305675805675805  F1-score =  0.8090193432298695
```

For K=10, we got the best predictive accuracy of 96.4%

```
Predictive accuracy for k =  10  is  0.9642857142857143
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 4 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 3 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 3 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 2 0 0 0 0 0]
 [0 2 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2]]
precision =  0.8782051282051282 Recall =  0.8653846153846154  F1-score =  0.8578754578754579
```