Department of
Computer Science and Engineering

Title: Machine Learning Classification: - Decision trees, k-Nearest Neighbor Implementation using Python

Data Mining Lab

CSE 424

Green University of Bangladesh

1 Objective(s)

- To understand few simple machine learning classifiers.
- To understand pruning using decision tree and how to choose appropriate k value in KNN.

2 Classification models

A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "classes." One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam. In this lab, we understand simple two classifiers, along with their respective datasets, pruning using decision tree and how to choose optimum value of parameter k.

2.1 Decision tree Classifier

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Types of Decision Trees

Types of decision trees are based on the type of target variable we have. It can be of two types:

1. **Categorical Variable Decision Tree:** Decision Tree which has a categorical target variable then it called a **Categorical variable decision tree.**
2. **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called **Continuous Variable Decision Tree.**

Decision Tree Algorithm

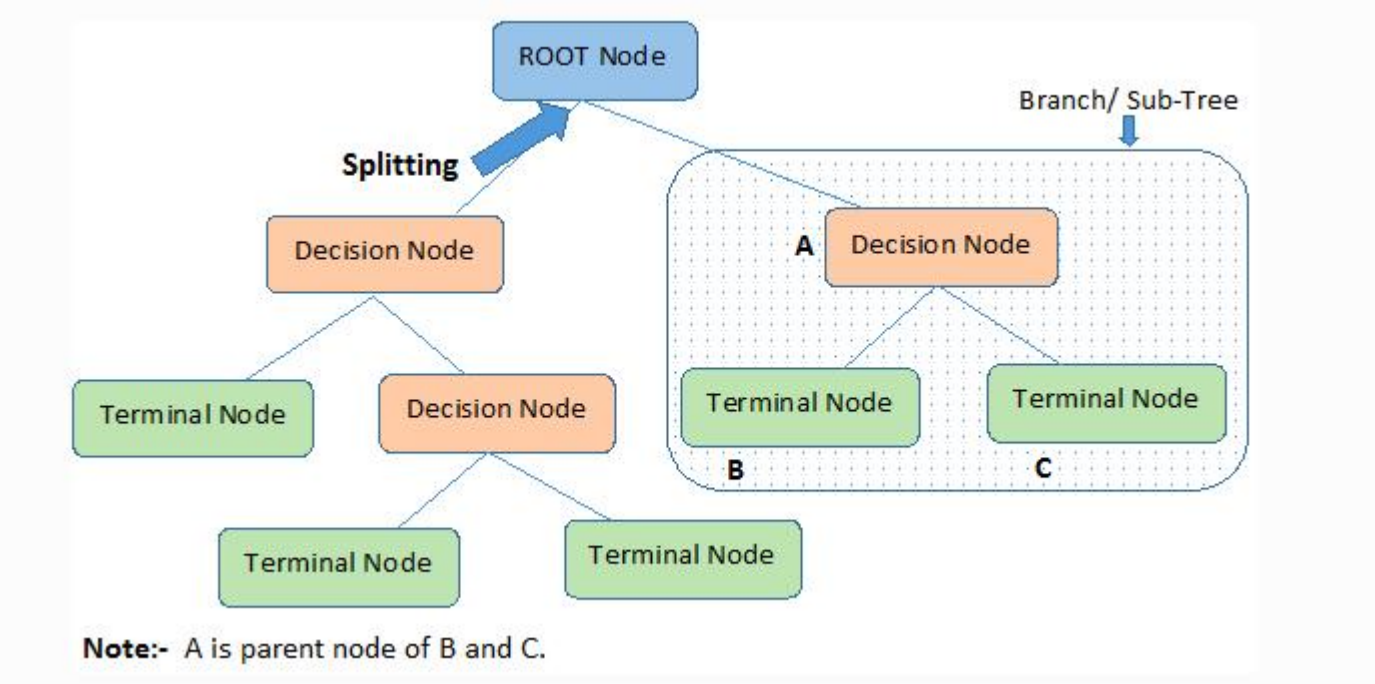
Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Important Terminology related to Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node



Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

Assumptions while creating Decision Tree

Below are some of the assumptions we make while using Decision tree:

- In the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are **distributed recursively** on the basis of attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

The primary challenge in the decision tree implementation is to identify which attributes do we need to consider as the root node and each level. Handling this is to know as the attributes selection. We have different attributes selection measures to identify the attribute which can be considered as the root note at each level.

How do Decision Trees work?

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes. The algorithm selection is also based on the type of target variables. Let us look at some algorithms used in Decision Trees:

ID3 → (extension of D3)
C4.5 → (successor of ID3)
CART → (Classification And Regression Tree)
CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
MARS → (multivariate adaptive regression splines)

The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment.

Steps in ID3 algorithm:

1. It begins with the original set S as the root node.
2. On each iteration of the algorithm, it iterates through the very unused attribute of the set S and calculates **Entropy(H)** and **Information gain(IG)** of this attribute.
3. It then selects the attribute which has the smallest Entropy or Largest Information gain.
4. The set S is then split by the selected attribute to produce a subset of the data.
5. The algorithm continues to recur on each subset, considering only attributes never selected before.

Attribute Selection Measures

If the dataset consists of N attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can't solve the issue. If we follow a random approach, it may give us bad results with low accuracy. For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some *criteria* like :

Entropy,
Information gain,
Gini index,
Gain Ratio,
Reduction in Variance
Chi-Square

These criteria will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root.
While using Information Gain as a criterion, we assume attributes to be categorical, and for the Gini index, attributes are assumed to be continuous.

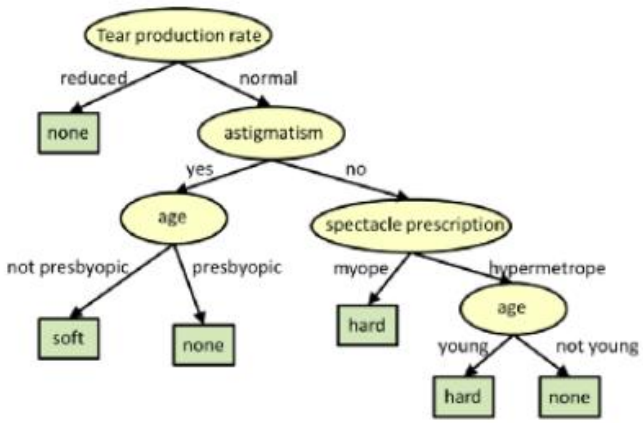
How to avoid/counter Overfitting in Decision Trees?

The common problem with Decision trees, especially having a table full of columns, they fit a lot. Sometimes it looks like the tree memorized the training data set. If there is no limit set on a decision tree, it will give you 100% accuracy on the training data set because in the worse case it will end up making 1 leaf for each observation. Thus this affects the accuracy when predicting samples that are not part of the training set.

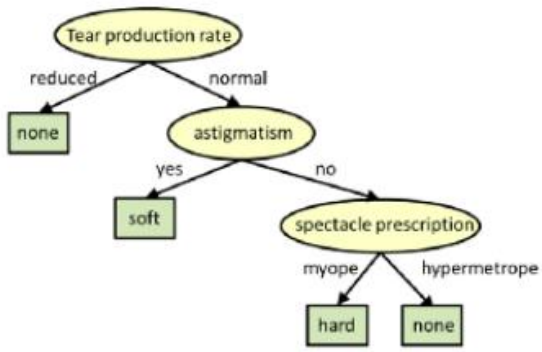
Here are two ways to remove overfitting:

1. Pruning Decision Trees.
2. Random Forest

In **pruning**, you trim off the branches of the tree, i.e., remove the decision nodes starting from the leaf node such that the overall accuracy is not disturbed. This is done by segregating the actual training set into two sets: training data set, D and validation data set, V. Prepare the decision tree using the segregated training data set, D. Then continue trimming the tree accordingly to optimize the accuracy of the validation data set, V.



Original Tree



Pruned Tree

In the above diagram, the 'Age' attribute in the left-hand side of the tree has been pruned as it has more importance on the right-hand side of the tree, hence removing overfitting.

Random Forest

Random Forest is an example of ensemble learning, in which we combine multiple machine learning algorithms to obtain better predictive performance.

2.2 k- Nearest Neighbor Classifier

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following −

2.2.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

2.2.2 − Now, based on the distance value, sort them in ascending order.

2.2.3 − Next, it will choose the top K rows from the sorted array.

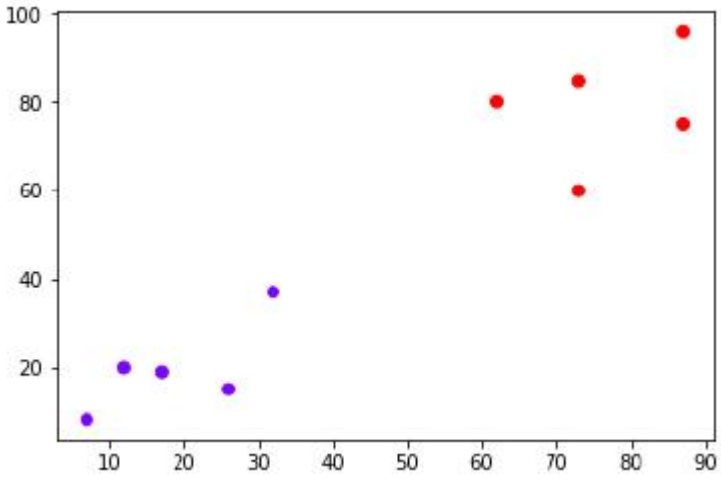
2.2.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 − End

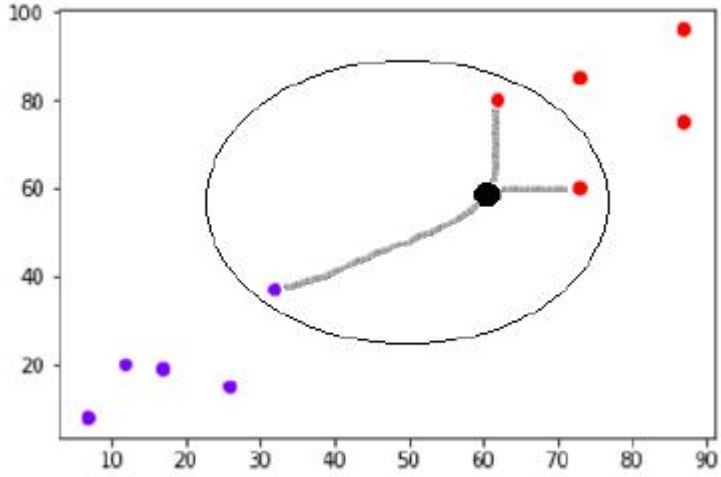
Example

The following is an example to understand the concept of K and working of KNN algorithm −

Suppose we have a dataset which can be plotted as follows −



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

how to select the optimal K value?

- There are no pre-defined statistical methods to find the most favorable value of K.
- Initialize a random K value and start computing.
- Choosing a small value of K leads to unstable decision boundaries.
- The substantial K value is better for classification as it leads to smoothening the decision boundaries.
- **Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.**

3.Implementation of decision tree and KNN using python:

Step 1: Import necessary libraries:

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from sklearn.metrics import  accuracy_score
from matplotlib import pyplot as plt
from sklearn import tree
```

Step 2: Load dataset

```
#Loading datasets
iris_data = load_iris()
iris=pd.DataFrame(iris_data.data)
iris_targets=pd.DataFrame(iris_data.target)

#priting features name of iris data
print ("Features Name : ", iris_data.feature_names)

#priting targets name of iris data
print ("Targets Name : ", iris_data.target_names)

#shape of datasets
print ("Dataset Shape: ", iris.shape)

#first five sample features
print ("Dataset: ",iris.head())

#first five sample targets
print ("Dataset: ",iris_targets.head())

# features and targets
X = iris_data.data
Y = iris_data.target
```

Step 3: Splitting the dataset

```
# Splitting the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

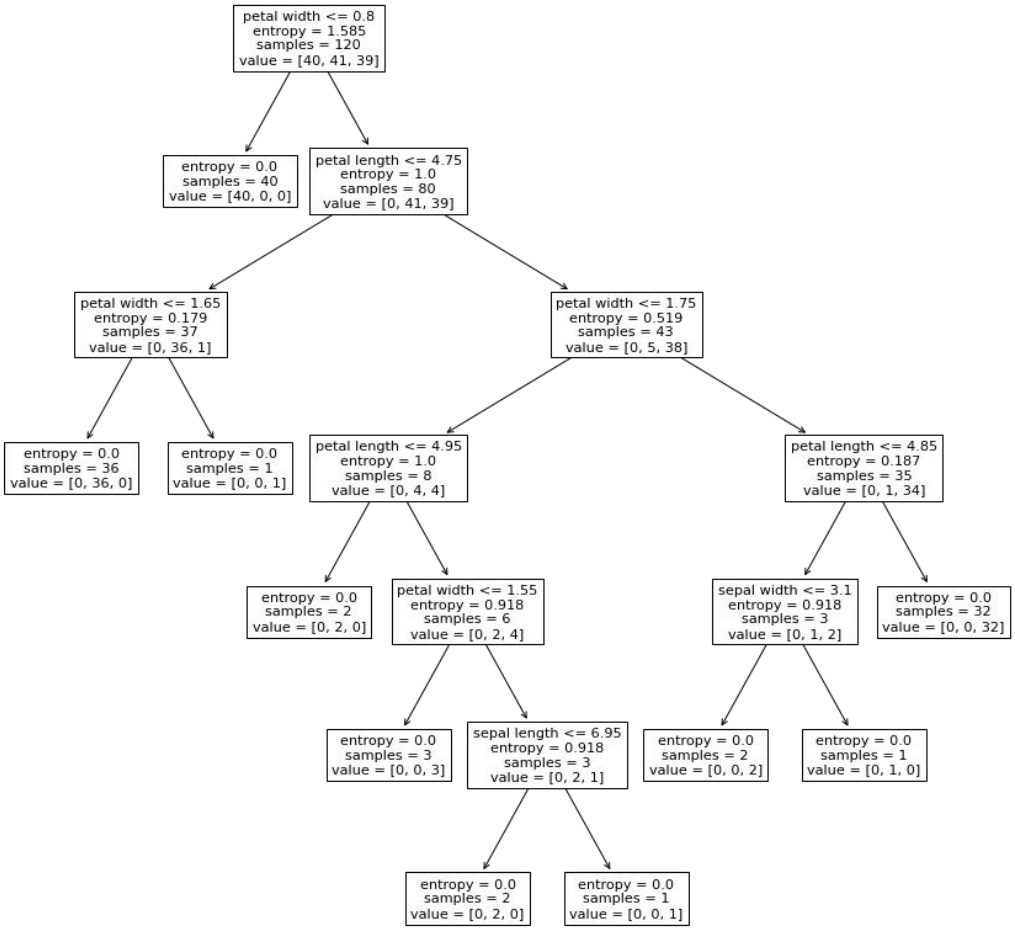
Step 4: Implementing decision tree classifier

```
# Decision tree classifier
DT = DecisionTreeClassifier(criterion='entropy')

#fitting the training data
DT.fit(X_train, y_train)
# prediction on X_test (testing data )
Y_pred=DT.predict(X_test)
```

```
#Accuray of the model
print("Accuracy:", accuracy_score(y_test, Y_pred))
#confusion matrix
cm=np.array(confusion_matrix(y_test, Y_pred))

#plot decision tree
fig, ax = plt.subplots(figsize=(15, 15)) #figsize value changes the size of plot
tree.plot_tree(DT,ax=ax,feature_names=['sepal length','sepal width','petal length','petal width'])
plt.show()
```

Output:

Accuracy: 1.0
array([[10, 0, 0],
[0, 9, 0],
[0, 0, 11]])



Step 5: Implementing KNN classifier

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors=5)
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_test)
from sklearn.metrics import confusion_matrix
#Accuray of the model
print("Accuracy:", accuracy_score(y_test, Y_pred))
print(confusion_matrix(y_test, y_pred))
```

Output:

```
Accuracy: 1.0
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

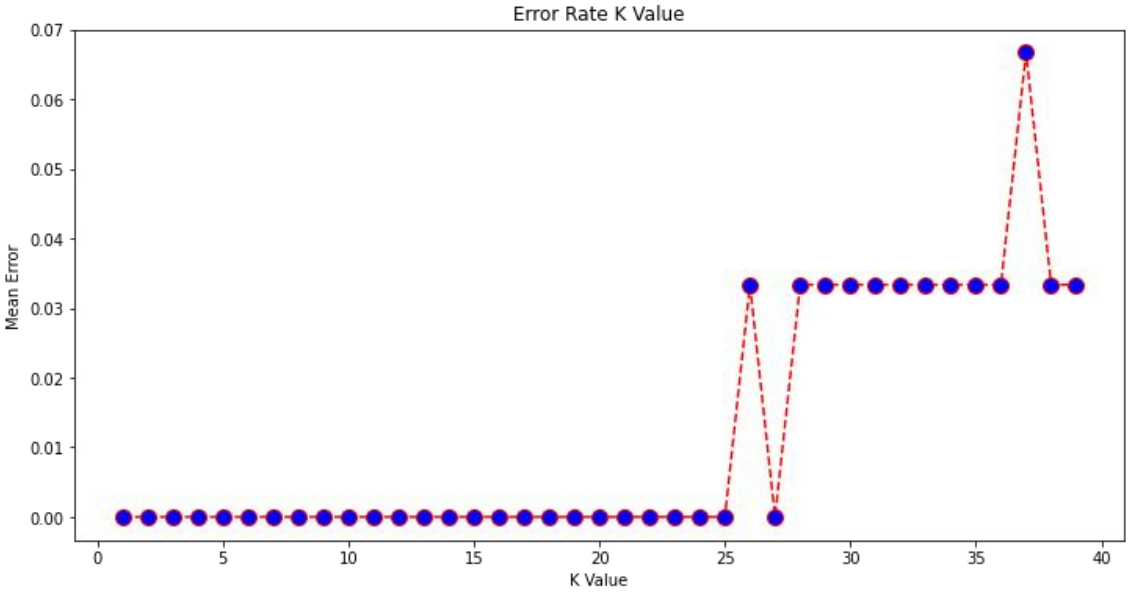
Step 5.1: Calculating error to get optimized k value

```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
    print(np.mean(pred_i != y_test))
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Output:

```
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
0.03333333333333333
0.0
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.03333333333333333
0.06666666666666667
0.03333333333333333
0.03333333333333333



4 Discussion & Conclusion

Based on the focused objective(s) to understand the use of different classifiers, we get to observe different simplest classifiers, their variations, problems, pruning and getting optimized parameter value separately. The additional lab exercise will increase confidence towards the fulfilment of the objectives(s).

5 Lab Exercise

Please run the classifiers using different datasets and implement pruning using decision tree classifier and show the output to your instructor.

6 Lab Report

Please construct a lab report on pruning a decision tree. The report must have the following points:

- Exclusive and details theory regarding decision tree in short and its pruning reasons, along with types of pruning. [in theory part]
- implement a decision tree classifier in python in a certain dataset, different from your classmates.
- The chosen dataset should be described, about tuples, attributes, class with screenshots.
- Implement the decision tree classifier with 15-fold cross validation.
- In output, must be discussed the two results separately: one with a unpruned tree, another with pruned tree result.
- Unpruned and pruned results must include screenshots of output, along with the full tree figures for both cases.
- A little discussion at the end and your opinion on pruning.

7 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be deducted if any such copying is detected for lab exercise.