# Usage instructions for Question 4

The only program that is to be executed is **validateTree.py**. Rest of the programs will be called by this. The changes that are to be made for running the program are listed out in the description for this program.

*Note: All programs must be in the same folder*.

## validateTree.py

Inputs variables in the program:

1. **"datasets"** in dataset = **"datasets"** on **line 9** specifies the folder where the input file is present.
2. **"iris.txt"** in fileName = dataset+**"\\"**+**"iris.txt"** on **line 10** specifies the name of the input file.
3. **','** in delim = **','** on **line 11** specifies the delimiter used in the input file.
4. cols in cls = cols on **line 14** specifies the column number of the class variable in the input file. If the class variable is the last column, the statement need not be changed. (cls has to be column number, not column index… i.e if it is the first column, cls has to be 1, not 0.)
5. ignore = [] on **line 16** indicates the columns that are to be ignored(if any). Column numbers must be separated by commas.
6. measure = **'i'** on **line 18** specifies the purity measure to be used. **i** is for information gain and **g** is for Gini.

The program splits the input file into 10 partitions and uses a 10 fold cross validation.

For every iteration, the program builds a decision tree by calling the function **buildTree** from **classification.py** using the training data for that iteration.

Once the tree is built, a class label is predicted for each of the records in the test data for that iteration by calling **predictClass** from **prediction.py**.

At the end of all 10 iterations, accuracy is calculated based on number of correct predictions made.

For every run of this program, accuracy is displayed as well as logged in a file named k-fold log.txt in the current working directory.

## classification.py

This is used from validateTree.py, but can be executed directly if desired by modifying the main section as required.

The buildTree function takes a training data matrix, purity measure, class column and optional ignore list as arguments and returns a decision tree in the form of a dictionary.

## prediction.py

This is used from validateTree.py, but can be executed directly if desired by modifying the main section as required.

The predictClass function in this program takes test data matrix and a decision tree as inputs and returns the predicted class label.

## splitFile.py

This is used from validateTree.py, but can be executed directly if desired by modifying the main section as required.

The function splitFile is takes as inputs a file and number of partitions it has to make and returns a list of all file names.