# Procedure and Results

## Procedure:

This section explains in detail about the Methodology used in building the Decision tree:

Attributes used: ConfirmedIndiaNATIONAL, ConfirmedForeignNational, Confirmed,States,Cured
Target : Deaths - Seen as a discrete classification variable that can take a value from 0-5.

1. **Finding Best Splits:** As per the ID3 algorithm we build the DT using a greedy approach - At each step we find the best possible split among all possible splits.
2. We perform only binary splits (for continuous variables it is x > value or not )

   For getting the best possible split we first compute ALL POSSIBLE Binary splits for EACH attribute in the dataset.

   - For each continuous attribute:
     - Iterate through all the values and store it in a list
     - Sort all the values
     - Now the candidates for splitting are all unique values of (i + j)/2 where i and j are 2 different consecutive values in the sorted list.
     - For Example: [2 1 3] - Possible splits are [1.5 2.5]
   - For discret attributes: In our case Only States - We just perform Binary splits
     - The possible candidates are the set of all states in data set

Now with this list of ALL POSSIBLE SPLIT CANDIDATE LIST we iterate through each one of them to perform the split and compute the following heuristic. The best split is the one with the smallest value of the following heuristic.

Heuristic = (ypredict - yTarget)^2 (left) + (ypredict - yTarget)^2 (right)

This heuristic has to be minimized because we want the ypredict to be as close as possible to yTarget.

The ypredict is simply the most occurring target class in the particular child node.

On Comparing with standard heuristics such as Gini Index and Entropy of mixture, our heuristic gave a superior validation accuracy.

So we perform all possible splits of each attribute and select the one which gave the lowest above defined heuristic. Concretely, it can be concluded that the split giving the lowest heuristic has the ypredict closest to the desired ytarget. Though the target variable is discrete, we minimise the root mean square between the desired output and the predicted

Once the best split for the current level is chosen, we recursively build the tree by a Depth First Method (left side): We stop under 2 conditions: 1. If max depth has reached, 2. If no elements are there in the child node. In these cases the node is made a terminal node with a prediction value of MODE OF THE CLASSES OCCURING.

Now, the tree is evaluated using 10Fold Cross validation technique. We varied the Depth to find out that we get best performance at **Depth  = 12.(Result plot attached)**
**Accuracy Obtained  = 95.74%**

Pruning Step: Reduced Error Pruning

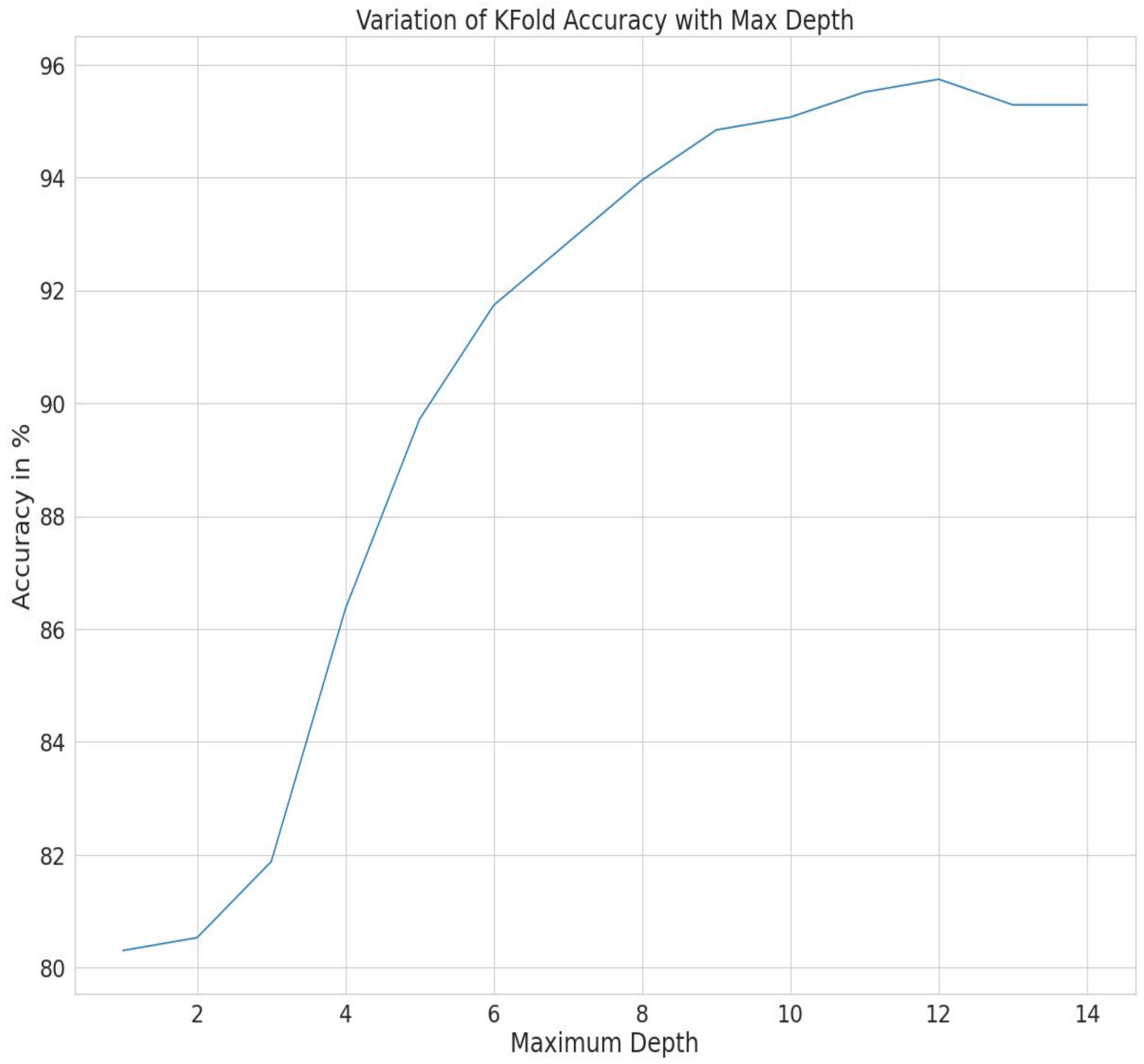
We used the Reduced Error PruningL In this case we delete each non leaf node in the tree recursively and replace it with a terminal node. For each of these single node pruned trees we compute the validation accuracy. If the best validation accuracy from this set outputs a higher validation accuracy than the BENCHMARK validation accuracy, then permanently prune that subtree and new benchmark is the validation accuracy of this tree. Repeat the process until the best validation accuracy from the pruning step is less than the benchmark accuracy

Printing the Tree using a Recursive step and proper indentation:

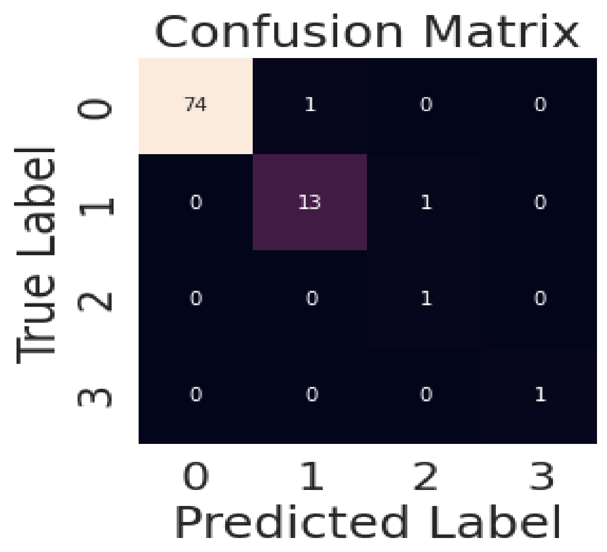
Apart from the Overview given here, each function in the code file has a proper **Docstring** and explanation as to what the function exactly does, what it takes and what it returns. SO please refer the code for further explanation on how the assignment has been done.

## RESULTS:

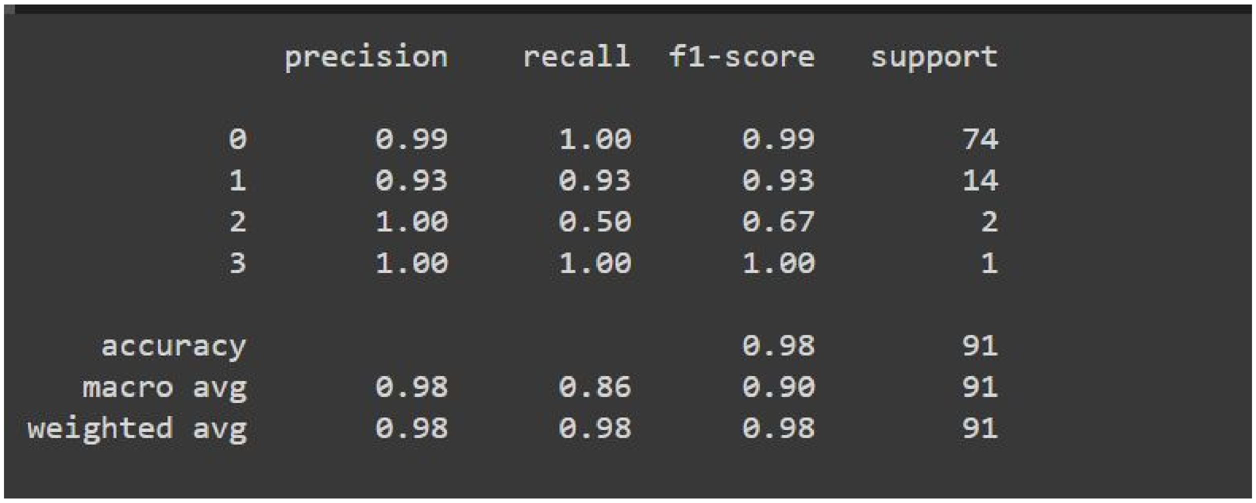
**The Highest Accuracy Obtained is 95.74% at a depth of 12.**



Confusion Matrix on Test data:



Classification report on Test data:



```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99        74
           1       0.93      0.93      0.93        14
           2       1.00      0.50      0.67         2
           3       1.00      1.00      1.00         1

    accuracy                           0.98        91
   macro avg       0.98      0.86      0.90        91
weighted avg       0.98      0.98      0.98        91
```

Printed Tree after pruning:

Can be seen after running the cell in the Notebook