CMPE480 Decision Tree Implementation Report

Elif Çalışkan 2016400183

Introduction:

The problem is to implement a decision tree algorithm and use iris dataset in order to train it.

Analysis:
- Download the dataset
- For 10 times:
  - Shuffle the data
  - Divide your dataset into training (20%), validation (40%) and test (40%) sets
  - Apply decision tree learning using the training set. Stop splitting based on the loss in the validation set.
  - Plot the change in training and validation loss given depth of the trees during training
  - Report the loss in the test set.
- Provide a final comparison between performances of information gain and Gini impurity metrics and plot/provide the mean and variance in error for these two different metrics.

Implementation:

Main method:

This method stores the iris data using load_iris function. Then, targets are added to the last column of each row. Then this data is shuffled and splited into three sets: training, validation and test. Then the tree is built using training data and the tree is validated and tested.

Label_counts:

This function returns the histogram of rows.

Gini:

Finds the entropy of the node using the rows. The entropy formula is used.

Info_gain:

Finds the gain if the split is made. This function is used in order to find the correct split parameter.

Decision_Node:

This class stores the information of each decision. Each node keeps the depth of itself. Every node stores train, validation and test rows in order to make comparisons easier. If a node is pruned then it becomes a leaf. So there is a leaf flag for that. It keeps true and false branches as its children.

SplitParameter:

It keeps the value of parameter and its column.

Leaf node keeps the depth and rows.

Split:

This function partitions the data given the parameter and rows to split. Returns true and false branches.

Find_Split:

This function tries each parameters and finds the one with the highest gain. Returns the gain and selected parameter.

Make_Tree:

This function builds the tree recursively by creating a new Node or Leaf after the controls.

BFS():

This makes breadth first search which I have used in order to checks the creation of tree.

Find_Error():

This function returns the error given train and validation/test labels. The correct result is assumed to be the most frequent label in trained data. So in validation, the data which are not correct are assumed to be erroneous.

Validation():

This iterates through the tree using dfs and finds the nodes to prune. If any child has higher error than the current node, children are pruned and current node is assumed to be the leaf.

Test():

Using the created and pruned tree, the test data is predicted. Then their error values are computed.

Results Error Values on Leaves:

Count 0

```
Validation error in depth  0 ->0.7

Validation error in depth  1 ->0.4888888888888889

Validation error in depth  2 ->0.13636363636363635

Validation error leaf in depth  3 ->0.13043478260869565

Validation error leaf in depth  2 ->0.0

Test error in depth  4 ->0.0

Test error in depth  3 ->0.5

Test error in depth  3 ->0.21052631578947367

Test error in depth  2 ->0.0
```

Count 1

```
Validation error in depth  0 ->0.7333333333333333

Validation error in depth  1 ->0.4772727272727273

Validation error leaf in depth  3 ->0.13043478260869565

Validation error leaf in depth  3 ->0.047619047619047616

Validation error leaf in depth  2 ->0.0

Test error in depth  3 ->0.047619047619047616

Test error in depth  3 ->0.1111111111111111

Test error in depth  2 ->0.0
```

Count 2:

```
Validation error in depth  0 -> 0.6833333333333333

Validation error in depth  1 -> 0.43902439024390244

Validation error leaf in depth  3 ->0.0625

Validation error leaf in depth  3 ->0.12

Validation error leaf in depth  2 ->0.0

Test error in depth  3 ->0.1111111111111111

Test error in depth  3 ->0.0

Test error in depth  2 ->0.0
```

Count 3:

```
Validation error in depth  0 ->0.6666666666666666

Validation error leaf in depth  2 ->0.09523809523809523

Validation error in depth  1 ->0.48717948717948717

Validation error leaf in depth  3 ->0.05263157894736842

Validation error leaf in depth  3 ->0.0

Test error in depth  2 ->0.19047619047619047

Test error in depth  3 ->0.0

Test error in depth  3 ->0.0
```

Count 4:

```
Validation error in depth  0 ->0.75

Validation error in depth  1 ->0.5333333333333333

Validation error leaf in depth  3 ->0.08333333333333333

Validation error leaf in depth  3 ->0.09523809523809523

Validation error leaf in depth  2 ->0.0

Test error in depth  3 ->0.10526315789473684

Test error in depth  3 ->0.15789473684210525

Test error in depth  2 ->0.0
```

Count 5:

```
Validation error in depth  0 ->0.6833333333333333

Validation error in depth  1 ->0.4864864864864865

Validation error leaf in depth  3 ->0.1

Validation error leaf in depth  3 ->0.058823529411764705
Validation error leaf in depth  2 ->0.0
Test error in depth  3 ->0.17391304347826086
Test error in depth  3 ->0.0
Test error in depth  2 ->0.0
```

Count 6:

```
Validation error in depth  0 ->0.6833333333333333
Validation error in depth  1 ->0.46511627906976744
Validation error leaf in depth  3 ->0.058823529411764705
Validation error leaf in depth  3 ->0.15384615384615385
Validation error leaf in depth  2 ->0.0
Test error in depth  3 ->0.045454545454545456
Test error in depth  3 ->0.2727272727272727
Test error in depth  2 ->0.0
```

Count 7:

```
Validation error in depth  0 ->0.65
Validation error leaf in depth  2 ->0.0
Validation error in depth  1 ->0.5348837209302325
Validation error leaf in depth  3 ->0.17391304347826086
Validation error leaf in depth  3 ->0.0
Test error in depth  2 ->0.05555555555555555
Test error in depth  3 ->0.045454545454545456
Test error in depth  3 ->0.0
```

Count 8:

```
Validation error in depth  0 ->0.7
Validation error in depth  1 ->0.5813953488372093
Validation error leaf in depth  3 ->0.0
Validation error leaf in depth  3 ->0.25
Validation error leaf in depth  2 ->0.0
Test error in depth  3 ->0.08333333333333333
Test error in depth  3 ->0.21052631578947367
Test error in depth  2 ->0.0
```

Count 9:

```
Validation error in depth  0 ->0.7166666666666667

Validation error in depth  1 ->0.5405405405405406

Validation error leaf in depth  3 ->0.15789473684210525

Validation error leaf in depth  3 ->0.1111111111111111

Validation error leaf in depth  2 ->0.0

Test error in depth  3 ->0.05

Test error in depth  3 ->0.125

Test error in depth  2 ->0.0
```