Introduction to Artificial Intelligence - Project 3

M.Akın Elden
2015402072

1 Introduction

The decision tree learning implementation is performed with Python during the project. Some of the outputs are placed in report but since 20 trials needed, all plots couldn't be placed into the report. The remaining outputs, including plots, can be easily seen using the Jupyter Notebook submitted with the report.

2 Implementation

First of all, the "Iris" dataset is loaded using sklearn package and than reformatted for easier usage in the functions. Then a class named Node is created to represent the each split node, depth, of the tree. This class includes train, validation and test data for each depth of the tree. After that, impurity measurement and information gain functions are implemented.

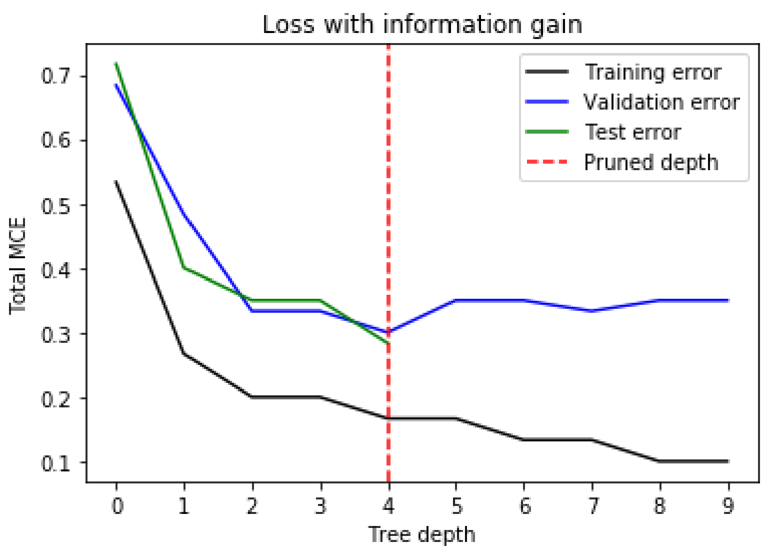
Two different impurity measures (entropy and Gini index) are used to calculate the impurity of a cluster. Then information gain and Gini index gain values are used to determine splitting feature and value. For that purpose, at each depth, the train data of the current node evaluated for all unique values in all columns and the one with most information gain or Gini index gain is selected as the splitting parameter. Since these measures don't consider the predicted class in their calculation, misclassification error(MCE) is used as loss function to evaluate the tree after the training. When tree is completely trained, misclassification is calculated as $MCE = 1 - P_o$ where P_o is equal to $\frac{Number\ of\ predicted\ class\ in\ leaf\ node}{Total\ number\ of\ instances\ in\ leaf\ node}$. Tree is trained until there exists no possible split with data. During this process, train set and validation set loss values (MCE) are calculated at each split and stored in a list. When all the splits are done, the number of splits which minimizes the validation loss is selected as the depth of tree and test set is splitted until that depth and loss in the test set is calculated at each step. Also the accuracy of the prediction (number true and false estimations) is calculated and returned. When all these operations are done; loss values of training, validation and test sets are plotted against depths of tree and the best splitting number is showed with a red dashed line.

3 Results

Example of a tree trained by entropy measurement:

```python
results = decisionLearning(iris, "entropy", seed=2)
print("Test error at pruned depth: {0}".format(results[3][-1]))
results[-1]
```

Outputs:
Test error at pruned depth: 0.28333

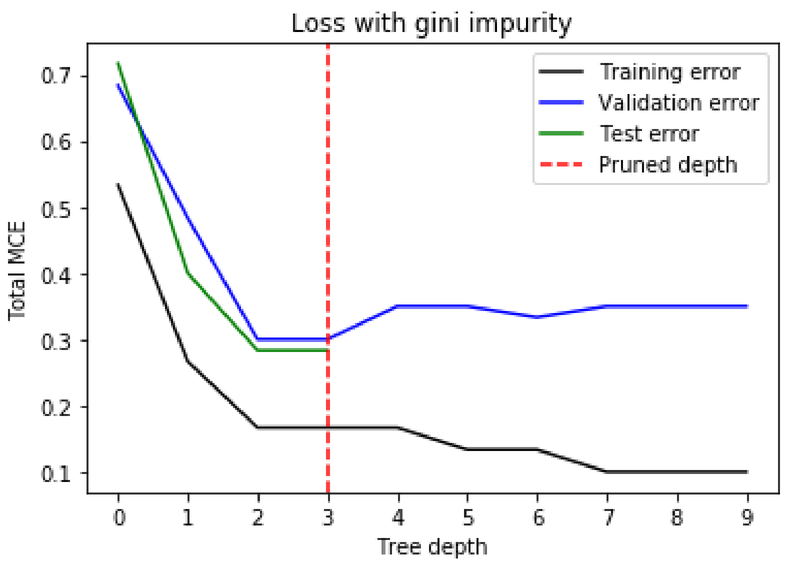


True estimate	False estimate
43	17

Example of a tree trained by Gini index measurement (with same seed value):

```python
results = decisionLearning(iris, "gini", seed=2)
print("Test error at pruned depth: {0}".format(results[3][-1]))
results[-1]
```

Outputs:
Test error at pruned depth: 0.28333



True estimate	False estimate
43	17

The mean and variance of error values obtained from 10 entropy and 10 Gini index performed trees (with same seed values).

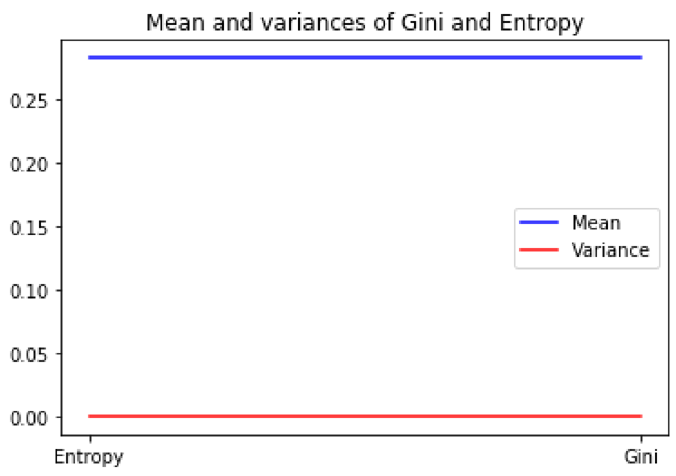
```python
entropy_results = []
gini_results = []
seed = 2
for i in range(10):
    e_res = decisionLearning(iris, "entropy", seed=seed)
    g_res = decisionLearning(iris, "gini", seed=seed)
    entropy_results.append(e_res[3][-1])
    gini_results.append(g_res[3][-1])

e_mean = np.mean(entropy_results)
e_variance = np.std(entropy_results)

g_mean = np.mean(gini_results)
g_variance = np.std(gini_results)
```

Outputs:

Impurity	Mean	Variance
Entropy	0.28332999999999997	5.551115123125783e-17
Gini	0.28332999999999997	5.551115123125783e-17



4 Comparison

As we can see from the above results, even if the optimal depth is different for each of the methods, the resulting error values are the same. The reason is both the similarity of measurements and the size of the data since it's a small data and actually it's the more dominant reason.