Assignment - 1

CS60050 - Machine Learning

Group - 1
Haasita Pinnepu (19CS30021)
Piriya Sai Swapnika (19CS30035)

Abstract

This report is submitted as part of an assignment of the course CS60050 – Machine Learning, IIT Kharagpur. The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. The task was to build a decision-tree classifier, analyze it and print the most accurate one. This report briefly explains the procedure we followed to achieve the goal and the results obtained.

1. Generating Containers

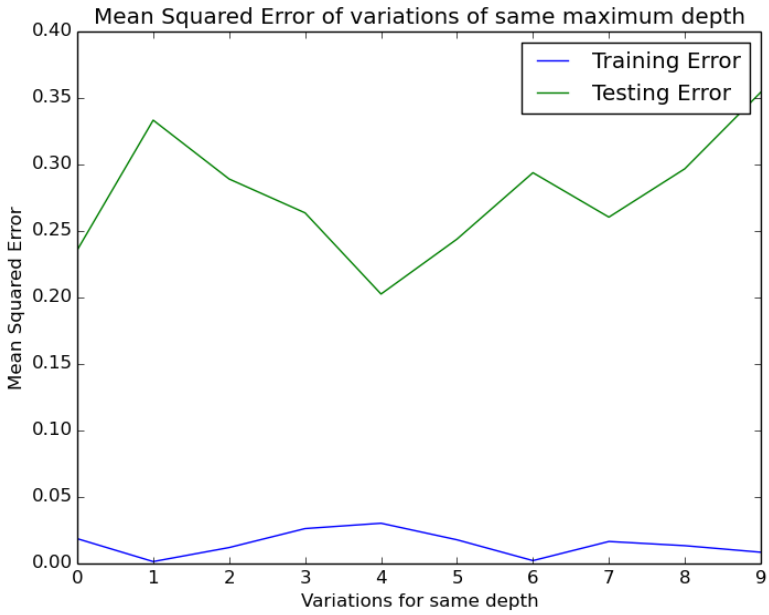
We first created a class DataClass to store the records of the dataset in a Python virtual environment. The attributes of the class Record are Pregnancies (Number of times pregnant), Glucose (Plasma glucose concentration), BloodPressure (Diastolic blood pressure), SkinThickness (Triceps skinfold thickness), Insulin (2-Hour serum insulin), BMI (Body mass index), DiabetesPedigreeFunction, Age and Outcome. Later, the records (which are now Record instances) are fused to form a pandas.DataFrame object. This is done because it is quite simple to handle data using such objects.

2. Using the impurity measures: 1) Gini index and 2) information gain.

We assume a max_depth of 10. While building the trees of a given maximum depth, the entire dataset is divided into training set, validation set and test set in the ratio 60:20:20. This is done 10 times with independent random splitting. Each tree remembers what was the partition used for its training and testing (and validation, however validation set is not used in this part). For choosing the best attribute for a given split, 2 impurity measures are employed, namely, 1. Gini Index and 2. Information Gain. The measure with the higher average accuracy is chosen and continued with for the rest of the question. The average test accuracy is reported as the test accuracy of these 10 trees, while the best one is chosen for further tasks. To define the accuracy of a decision tree, we used mean squared error over the predicted values and actual values.

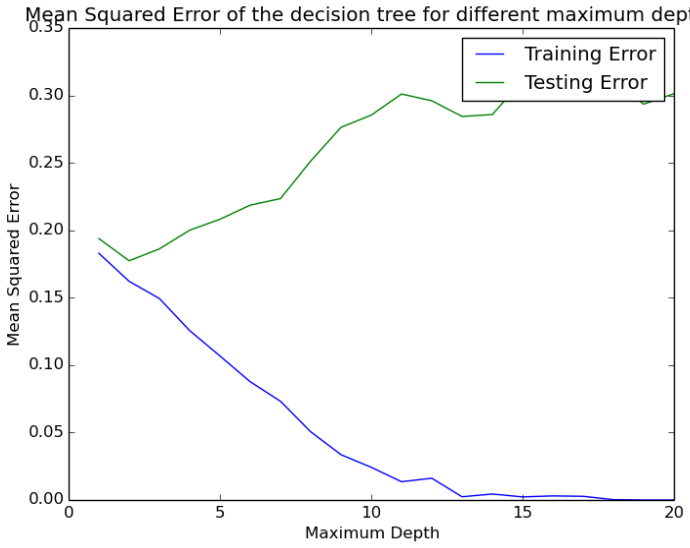
3. Calculating the accuracy by averaging over 10 random 80/20 splits

Max_depth assumed here is 10.



4. Selecting the best possible depth limit

Here, the best possible depth has been chosen in the range 1 to 20.



Therefore, bestDepth = 2

5. Pruning the Best Tree over Validation Set

We use the tree which gave the best results (i.e. its depth is 10). It can be further pruned so as to improve its performance. For this case, the validation set is used. Note that it is not necessary that the pruned tree will perform better than the unpruned one over the test set (as pruning is done only on the validation set). For pruning, we are using a reduced-error pruning algorithm. It is done in a top-down manner recursively. For a given node, it checks whether after pruning this node, the error on the validation set reduces or not. If it does, the node is pruned by making it a leaf node and its prediction is set to the mean of training instances that were incident on this node. In our case, the unpruned tree performed better on the test set as compared to the pruned one. (But the scenario may be different if the program is executed again.)

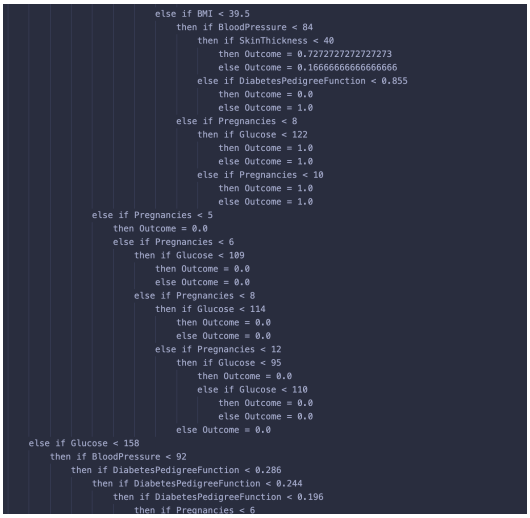
```
Solution to Q4:
    Before pruning:
        trainError = 0.0352696216827, valError = 0.315222783328, testError = 0.237708012528

    Refer to prunedTree.txt file

    After pruning:
        trainError = 0.141343169848, valError = 0.185705961628, testError = 0.168253023062
```

6. Printing the Decision Tree:

A recursive approach is used to print the decision tree.



Part of the final tree printed in the file “tree.txt”.