
Data Mining: Assignment 2

=====

Krishna Mahajan, 0003572903

Q4

Implementing classification trees and evaluating their accuracy

(a) Implement the greedy algorithm that learns a classification tree given a data set. Assume that all features are numerical and properly find the best threshold for each split. Use Gini and information gain, as specified by user, to decide on the best attribute to split in every step. Stop growing the tree when all examples in a node belong to the same class or the remaining examples contain identical features

(soln)

Here are following explanation of python Modules that i coded while implementing the decision tree ,in sequence.

Step 1) Reading raw data: (1) Functionality of this module is to open the raw data file containing all the observations with final target variable (Class Variable). The module read all the data and do the necessary cleaning (such as converting numerical attributed which are loaded as strings and convert them back to numerical type).

(2) Module 'll also add header to the raw data and colnames need to be passed as an input.

(3) Finally after all the cleaning ,Module convert the loaded the dataset into python list of list format (Data Frame) and dump this list in pickle format so that it can be later used readily while building the decision tree.

[Module: readdata.py](#)

Step 2) Gini Measure to split dataset: I have used giniimpurity instead of entropy as asked to select the criteria to split the dataset. Giniimpurity is simply the expected error rate of assigning wrong class to a random observation.

[Module: impurity.py](#)

Step 3) Building the Tree: (1) I have implemented decision tree in recursive manner.

Here are brief steps of the Algorithm.

1. The Algorithm iterate over all the features in data and within each feature find the best split among all the possible values of that feature such that information gain is maximum if the data is splitted on that particular feature's particular value.

2. So now all the observations are divided either into True Branch (which are greater or equal to feature value) or False Branch (which are less than feature value)

3. Now recursively the same Algorithm repeats for each branch until there's only one class variable in that branch.

(2) At the completion ,the algorithm returns the root decision node which was initial node at which dataset was splitted first. If we traverse through the root decision node we can reach all other decision node and the final leafs (classes).

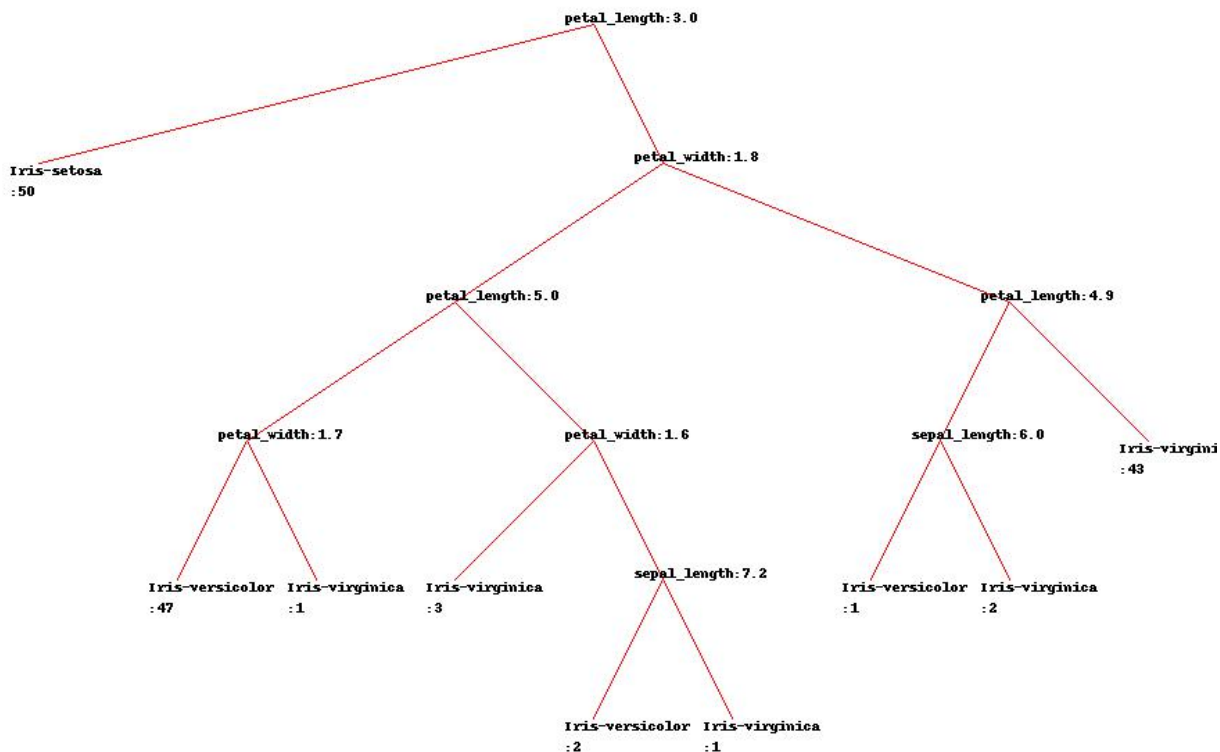
[Module: buildtree.py](#)

Step 4)Printing the tree: (1) I have used python's PIL library to print the decision tree as per colnames to visualise all the decision node and their corresponding branches.

[Module:drawtree.py](#)

Here are some examples of final decision tree model when i ran them on the following famous datasets:

Iris



Wine

Step 2) Create stratified 10 Folds: (1) This module splits the dataset in 10 training,testing folds while preserving the original distribution of classes in the original data at each fold.

I used python's scikit library's stratified K Fold to implement this module.

[Module:createfolds.py](#)

Step 3) Calculate Final efficiency: (1) This module builds a decision tree on every folds training data and then predicts final classes of each observation in testing data and then calculate the efficiency.

(2) The Algorithm iterate 10 times for each folds and at the end calculate the final efficiency which is mean efficiency of all 10 folds.

I used python's scikit library's stratified K Fold to implement this module.

[Module:accuracy.py](#)

Efficiency= (observation predicted correctly in Test Set)/(Total observation in test set)

Here are the results when i tested Decision Tree on 10 different datasets with 10 fold cross validation


[Iris](#)

```
C:\ Anaconda - python
>>> reload(accuracy)
<module 'accuracy' from 'accuracy.py'>
>>> a=accuracy.accuracy10Fold(dataset="iris")
('Accuracy for Testing fold', 1, 1.0)
('Accuracy for Testing fold', 2, 0.9333333333333333)
('Accuracy for Testing fold', 3, 1.0)
('Accuracy for Testing fold', 4, 0.9333333333333333)
('Accuracy for Testing fold', 5, 0.9333333333333333)
('Accuracy for Testing fold', 6, 0.8666666666666667)
('Accuracy for Testing fold', 7, 0.9333333333333333)
('Accuracy for Testing fold', 8, 0.9333333333333333)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 0.9333333333333333)
('Final Accuracy of ', 'iris', 0.9466666666666669)
>>>
```

[Wine](#)

```
C:\ Anaconda - python
>>> a=accuracy.accuracy10Fold(dataset="wine")
('Accuracy for Testing fold', 1, 0.9473684210526315)
('Accuracy for Testing fold', 2, 0.8333333333333334)
('Accuracy for Testing fold', 3, 0.7777777777777778)
('Accuracy for Testing fold', 4, 0.8888888888888888)
('Accuracy for Testing fold', 5, 0.8333333333333334)
('Accuracy for Testing fold', 6, 1.0)
('Accuracy for Testing fold', 7, 1.0)
('Accuracy for Testing fold', 8, 0.9444444444444444)
('Accuracy for Testing fold', 9, 1.0)
('Accuracy for Testing fold', 10, 1.0)
('Final Accuracy of ', 'wine', 0.9225146198830408)
>>>
```

BankNote

 Anaconda - python

```
>>> a=accuracy.accuracy10Fold(dataset="banknote")
('Accuracy for Testing fold', 1, 0.9927536231884058)
('Accuracy for Testing fold', 2, 0.9710144927536232)
('Accuracy for Testing fold', 3, 0.9854014598540146)
('Accuracy for Testing fold', 4, 0.9781021897810219)
('Accuracy for Testing fold', 5, 0.9854014598540146)
('Accuracy for Testing fold', 6, 0.9635036496350365)
('Accuracy for Testing fold', 7, 0.9927007299270073)
('Accuracy for Testing fold', 8, 0.9781021897810219)
('Accuracy for Testing fold', 9, 0.9927007299270073)
('Accuracy for Testing fold', 10, 0.9781021897810219)
('Final Accuracy of ', 'banknote', 0.9817782714482176)
>>>
```