Congress Decision Tree: Predicting Republican on Democrat.

Fall 2018

Oamar Kanji

1. Project description

The aim of this project is to predict whether the member of congress is a Republican or a Democrat based on a voting record for the USA congress. The chosen machine learning algorithm is a decision tree. A decision tree was decided upon as this is a classification problem and all the decisions fall into either of two categories (yes or no).

1. Description of training data set (if applicable).

U.S. congress voting record from 1984. The data set consists of the votes (yes or no) on sixteen issues for each of the 435 members of congress.

Attribute Information:

1. Class Name: 2 (democrat, republican)   
2. handicapped-infants: 2 (y,n)   
3. water-project-cost-sharing: 2 (y,n)   
4. adoption-of-the-budget-resolution: 2 (y,n)   
5. physician-fee-freeze: 2 (y,n)   
6. el-salvador-aid: 2 (y,n)   
7. religious-groups-in-schools: 2 (y,n)   
8. anti-satellite-test-ban: 2 (y,n)   
9. aid-to-nicaraguan-contras: 2 (y,n)   
10. mx-missile: 2 (y,n)   
11. immigration: 2 (y,n)   
12. synfuels-corporation-cutback: 2 (y,n)   
13. education-spending: 2 (y,n)   
14. superfund-right-to-sue: 2 (y,n)   
15. crime: 2 (y,n)   
16. duty-free-exports: 2 (y,n)   
17. export-administration-act-south-africa: 2 (y,n)

1. Description of algorithm

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf nodes represent a classification or decision.

* List of files

|  |  |  |
| --- | --- | --- |
| Name | Description | List of functions |
| Main | Splits the data into testing and training. Imports the Node class and builds the decision tree using the training data. Tests the decision tree on the testing data. | data\_proportions  entropy  split\_data  gain  labels\_all\_same  majority  make\_decision\_tree  decision\_tree  classify |
| Node | The machine learning algorithm |  |

Structure of Data

Below is an example of the raw data where the first 4 rows of data from house-votes-1984.txt:

R,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y  
R,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?  
D,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n  
D,n,y,y,n,?,y,n,n,n,n,y,n,y,n,n,y

The program reads each row and stores the features in an array. The features array and the corresponding label are combined in a tuple. This tuple is appended to an array called **data** that stores the whole data set. If a row contains a ‘?’ which is symbolizes missing data, it is discarded.

Below is an example of the first element of **data.**



List of functions

1)

data\_proportions(data)

Data proportions takes an array of n rows of data and returns the proportions of each label in the data.

For example, if this is the input (not actual data from house-votes-1984.txt):



Then the corresponding output will be:



entropy(proportions)

Entropy takes a list of proportions and computes the entropy.

2)

split\_data(data, feature\_index)

Split data groups the data based on a feature index. Grouping all the data into separate lists based on the value of the feature at that index.

For example, consider the data below is passed to split data with a feature index of 0, and then passed to split data with an index of 1:



A list of lists will be returned, the first list will contain all the tuples which have ‘Y’ in their first position and the second list will contain the tuples with ‘N’ in their first position.



3)

gain(data, feature\_index)

Splits the data on the feature index provided (uses the split\_data function) and returns the information gain of that split.

4)

labels\_all\_same(data)

Takes data in the form as described above and checks if the labels are all the same. This is used to check if a meaningful split can be made.

5)

majority(data, node)

Takes data and node instance of Node class. If there are no remaining features to split on then classify that node with the simple majority of the labels in the data set as this is our next best option.

6)

make\_decision\_tree(data, root, remaining\_features)

Makes the decision tree recursively by appending children to the root node. On each recursion it finds the best feature to split on and recursively computes the best feature of the remaining nodes, thus building the decision tree.

How to use program in terminal

Install python 3 change directory to the program directory containing Node.py, Tree.py and the data set. Run Tree.py from command line and follow prompts, in macOS/ Linux:

>python3 Tree.py