

Building a Decision Tree: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2019

Syntax

- Using Python to calculate entropy:

```
def calc_entropy(column):  
    """  
    Calculate entropy given a pandas series, list, or numpy array.  
    """  
    counts = numpy.bincount(column)  
    probabilities = counts / len(column)  
    entropy = 0  
    for prob in probabilities:  
        if prob > 0:  
            entropy += prob * math.log(prob, 2)  
    return -entropy
```

- Using Python to calculate information gain:

```
def calc_information_gain(data, split_name, target_name):  
    """  
    Calculate information gain given a data set, column to split on, and target.  
    """  
    original_entropy = calc_entropy(data[target_name])  
    column = data[split_name]  
    median = column.median()  
    left_split = data[column <= median]  
    right_split = data[column > median]  
    to_subtract = 0  
    for subset in [left_split, right_split]:  
        prob = (subset.shape[0] / data.shape[0])  
        to_subtract += prob * calc_entropy(subset[target_name])  
    return original_entropy - to_subtract
```

-Finding the best column to split on:

```
def find_best_column(data, target_name, columns):  
    """  
    Find the best column to split on given a data set, target variable, and list of columns.  
    information_gains = []  
    for col in columns:
```

- Applying a function to a data frame:

```
information_gain = calc_information_gain(data, col, "high_income")
```

```
df.apply(find_best_column, axis=0)
```

```
highest_gain_index = information_gains.index(max(information_gains))
```

```
highest_gain = columns[highest_gain_index]
```

```
return highest_gain
```

Concepts

- Pseudocode is a piece of plain-text outline of a piece of code explaining how the code works. Exploring the pseudocode is a good way to understand it before trying to code it.

- Pseudocode for the ID3 algorithm:

```
def id3(data, target, columns)

    1 Create a node for the tree

    2 If all values of the target attribute are 1, Return the node, with label = 1

    3 If all values of the target attribute are 0, Return the node, with label = 0

    4 Using information gain, find A, the column that splits the data best

    5 Find the median value in column A

    6 Split column A into values below or equal to the median (0), and values above the
median (1)

    7 For each possible value (0 or 1), vi, of A,

    8     Add a new tree branch below Root that corresponds to rows of data where A = vi

    9     Let Examples(vi) be the subset of examples that have the value vi for A

    10     Below this new branch add the subtree id3(data[A==vi], target, columns)

    11 Return Root
```

- We can store the entire tree in a nested dictionary by representing the root node with a dictionary and branches with keys for the left and right node.
- Dictionary for a decision tree:

```
{
  "left":{
    "left":{
      "left":{
        "number":4,
        "label":0
      },
      "column":"age",
      "median":22.5,
      "number":3,
      "right":{
        "number":5,
        "label":1
      }
    },
    "column":"age",
    "median":25.0,
    "number":2,
    "right":{
      "number":6,
      "label":1
    }
  }
```

```
}
```

Resources

- [Recursion](#)
- [ID3 Algorithm](#)

```
"number":1,
```

```
"right":{
```

```
"left":{
```



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2019

```
"left":{
```