# BT2101 – TUTORIAL 1: DECISION TREE

Kaustubh Jagtap A0168820B

NATIONAL UNIVERSITY OF SINGAPORE

**Overview**

In this tutorial, I first performed manual calculations for the information gain for all 5 attributes, by constructing frequency tables for all of them. Following this, I abstracted my methodology by writing python functions to calculate entropy and information gain (adapted and modified from the course github page). To address the third question, I directly defined python methods to calculate the gini index. All my findings are presented below.

**Questions 1 & 2 – Manual Calculation**

For this step, I had to break down the given dataset into frequency tables for each attribute. These frequency tables are given below. It is interesting to note that intuitively, we can hypothesize that *gender* would be the best attribute for the first split, since it gives us the cleanest partitioning – we can tell just by visual inspection of the data.

| Split on Passenger class | | |
|---|---|---|
| | Survived = yes | survived = no |
| First | 5 | 1 |
| Second | 3 | 5 |
| Third | 4 | 12 |

| Split on Age class | | |
|---|---|---|
| | Survived = yes | survived = no |
| Childhood | 4 | 4 |
| Young Adulthood | 6 | 9 |
| Middle Adulthood | 2 | 1 |
| Elder Citizen | 1 | 3 |

| Split on Gender | | |
|---|---|---|
| | Survived = yes | survived = no |
| Male | 2 | 16 |
| Female | 10 | 2 |

| Split on no. of siblings/ spouses | | |
|---|---|---|
| | Survived = yes | survived = no |
| 0 | 4 | 13 |
| 1 | 7 | 2 |
| >=2 | 1 | 3 |

| Split on no. of parents/ children | | |
|---|---|---|
| | Survived = yes | survived = no |
| 0 | 9 | 14 |
| 1 | 1 | 2 |
| >=2 | 2 | 2 |

*Sample Calculation for split on passenger class*

Using the entropy formula, I calculated the entropies for passenger class split as follows:

| Split on Passenger class | | | |
|---|---|---|---|
| | Survived = yes | survived = no | Entropy |
| First | 5 | 1 | 0.65 |
| Second | 3 | 5 | 0.954 |
| Third | 4 | 12 | 0.811 |
| Weighted Entropy | | | 0.817 |

*Results*

The above calculations were performed for all 5 attributes, and the results are shown below.

| Entropy Method | | |
|---|---|---|
| **Attribute** | **Weighted Entropy** | **Gain in Information** |
| Age Class | 0.868 | 0.103 |
| Passenger Class | 0.817 | 0.154 |
| Gender | 0.562 | 0.409 |
| No of Siblings or spouses | 0.783 | 0.187 |
| No of parents or children | 0.965 | 0.005 |
| Note: Initial Entropy = 0.971 | | |

We want to maximize the gain in information, hence we shall choose the *Gender* attribute for our first split. This is in line with our initial hypothesis.

**Questions 1 & 2 – Python scripts**

I thought it would be interesting to redo the above using python. I adapted some of the scripts from our course github and came up with an automated process to calculate information gain.

The first script I rewrote is that for entropy calculation. Below is an abstracted version of the function that can calculate entropy for *any* number of input sample labels.

**General function to calculate entropy** ¶

```
def entropy(sample_labels):
    """Input: A vector of sample labels e.g. [A,B,A,C,C,C,A,B,B,A,C,B,A,A].
       This input has to be for a particular segment of the attribute we are splitting on.
       e.g. if splitting on gender attribute, the vector will be the labels for all males. OR all females. etc
       Output: A number between 0 and 1, the entropy"""

    # Assert that it it a numpy array
    sample_labels = np.array(sample_labels)

    # Return 0 if empty array
    if sample_labels.size == 0:
        return 0

    class_values = np.unique(sample_labels)
    entropy = 0  # initialize entropy

    # iterate over every class and get the proportion. From there, add to entropy using formula.
    for label in class_values:
        number_of_instances = len(list(filter(lambda x:x==label, sample_labels)))
        if number_of_instances == 0:   # if no instances of this class, continue
            continue

        proportion = number_of_instances/len(sample_labels)
        entropy -= proportion*log(proportion,2)

    return entropy
```

I then defined a function to calculate information gain – this was done from scratch. Here, it takes a parameter called *method*. This is actually a function to be passed in – either entropy or gini_index. The function uses this *method* to calculate the information gain. Doing this allows for better abstraction.

### Function to calculate information gain ¶

```
def calc_info_gain(df, selected_feature, label, method):
    """df: Takes in the whole dataframe
       Selected Feature: Name of the column which we want to split on
       Label: Name of column which contains the labels
       method: The function we are using. eg entropy or gini_index. These are in fact functions.
       This function outputs a tuple (weighted_entropy, info_gain)"""

    initial_measure = method(df[label])    # calculate overall entropy/ gini-index of dataframe
    attribute_segments = df[selected_feature].unique()  # stores a list of all segments within chosen attribute eg.[male, female]

    weighted_measure = 0   # initialise weighted entropy/ gini-index

    for segment in attribute_segments:
        data_for_segment = df[df[selected_feature] == segment]
        labels_for_segment = data_for_segment[label]
        measure_for_this = method(labels_for_segment)
        weighted_measure += measure_for_this * len(data_for_segment)

    weighted_measure /= len(df)
    info_gain = initial_measure - weighted_measure

    return (weighted_measure, info_gain)
```

Running the function gives the following result.

```
                (Weighted Measure, Information Gain)
Age Class : (0.8675277182987636, 0.10342287615590495)
Passenger Class : (0.8172018848211989, 0.15374870963346965)
Gender : (0.5619639695247292, 0.4089866249299394)
No of Siblings or Spouses on Board : (0.7834701673945229, 0.18748042706014567)
No of Parents or Children on Board : (0.9654839523229165, 0.005466642131752075)
```

---

It is good to note that these results match with the ones calculated by hand above. Once again, it is affirmed that *Gender* would be the best attribute to split on.

**Question 3 – Using the Gini Index**

Below is a function to calculate the gini index for an attribute, given *any* number of class labels. This was written from scratch.

**General function to calculate Gini Index**

```
def gini_index(sample_labels):
    """Input: A vector of sample labels e.g. [A,B,A,C,C,C,A,B,B,A,C,B,A,A].
       This input has to be for a particular segment of the attribute we are splitting on.
       e.g. if splitting on gender attribute, the vector will be the labels for all males. OR all females. etc
       Output: The gini index for this split"""

    # Assert that it it a numpy array
    sample_labels = np.array(sample_labels)

    # Return 0 if empty array
    if sample_labels.size == 0:
        return 0

    class_values = np.unique(sample_labels)
    intermediate_sum = 0  # initialize 1 - (gini index)

    # iterate over every class and get the proportion. From there, add to entropy using formula.
    for label in class_values:
        number_of_instances = len(list(filter(lambda x:x==label, sample_labels)))

        proportion = number_of_instances/len(sample_labels)
        intermediate_sum += proportion**2

    return 1 - intermediate_sum
```

*Results*

Running the above script on the data produces the following raw results:

```
                (Weighted Gini Index, Reduction in Gini Index)
Age Class : (0.4277777777777778, 0.05222222222222217)
Passenger Class : (0.38055555555555554, 0.09944444444444445)
Gender : (0.22962962962962957, 0.2503703703703704)
No of Siblings or Spouses on Board : (0.3576252723311547, 0.12237472766884527)
No of Parents or Children on Board : (0.4763285024154589, 0.0036714975845411058)
```

The consolidated results are presented in the table below:

| Gini Index Method | | |
|---|---|---|
| **Attribute** | **Weighted Gini Index** | **Reduction from start** |
| Age Class | 0.428 | 0.052 |
| Passenger Class | 0.381 | 0.099 |
| Gender | 0.230 | 0.250 |
| No of siblings or spouses | 0.358 | 0.122 |
| No of parents or children | 0.476 | 0.004 |
| Note: Initial Gini Index = 0.480 | | |

The split on the *Gender* attribute produces the highest reduction in Gini Index. Hence, even using this method, *Gender* is the best root attribute to split upon.