Assignment

B ANIRUDH SRINIVSAN
COE17B019

1. Decision Tree

1.1 Introduction

Decision Tree Mining is a type of data mining technique that is used to build Classification Models. It builds classification models in the form of a tree-like structure. This type of mining belongs to supervised class learning. Decision trees can be used for both categorical and numerical data.

Decision Tree has three parts,

- An inner node which represents an attribute.
- An edge representing the test on the father node.
- And a leaf node representing one of the classes.

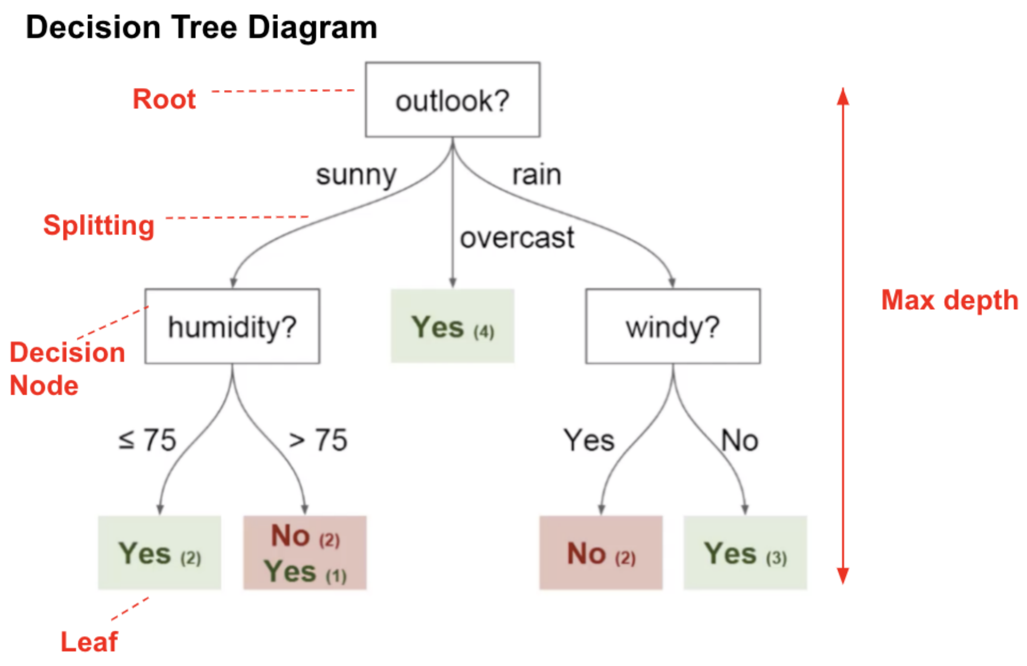
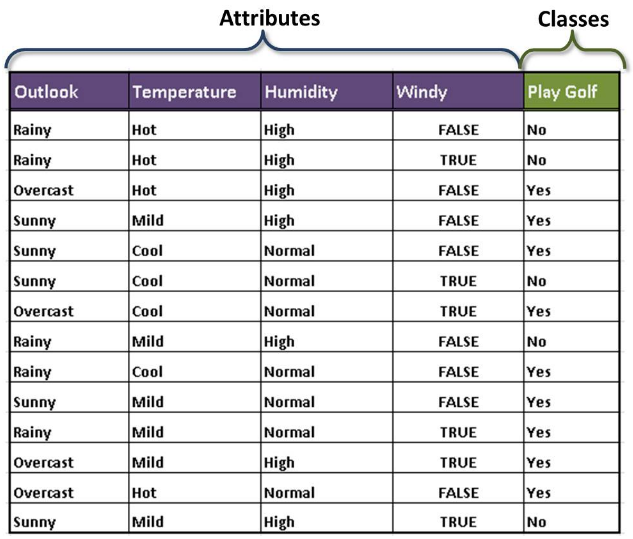


Figure 1: figure explaining different part of a decision tree

1.2 Construction of a decision tree with example

Example Dataset


Attributes
Classes

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

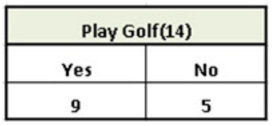
Figure 2: Example dataset

Step 1: Determine the Decision Column

Since decision trees are used for classification, you need to determine the classes which are the basis for the decision.

In this case, it it the last column, that is Play Golf column with classes Yes and No.

To determine the rootNode we need to compute the entropy. To do this, we create a frequency table for the classes (the Yes/No column).



Play Golf(14)	
Yes	No
9	5

Figure 3: Frequency table for Play Golf attribute

Step 2: Calculating Entropy for the classes

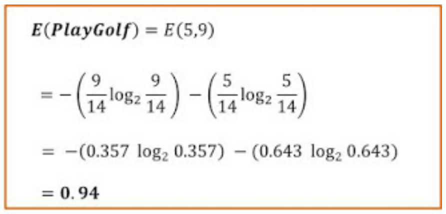
In this step, you need to calculate the entropy for the Decision column Formula for entropy is :

$$Entropy(Attribute) = -\sum_{i \in c} p_i \log_2 p_i \tag{1}$$

where,

- c is set of all classes for the attribute.
- p_i is the probability associated with that class.

The calculation of Play Golf attribute is :



$$\boldsymbol{E(PlayGolf)} = E(5,9)$$
$$= -\left(\frac{9}{14}\log_2\frac{9}{14}\right) - \left(\frac{5}{14}\log_2\frac{5}{14}\right)$$
$$= -(0.357 \log_2 0.357) - (0.643 \log_2 0.643)$$
$$= \mathbf{0.94}$$

Figure 4: Calculating Entropy

Step 3: Calculate Entropy for Other Attributes After Split

We need to calculate the entropy of the resultant tree, wrt each attribute and use it to calculate *Gain.* The entropy for two variables is :

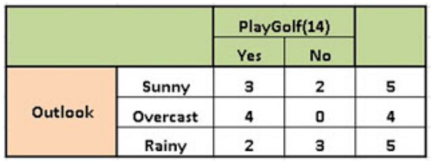
For the other four attributes, we need to calculate the entropy after each of the split.

$$Entropy(A1, A2) = \sum_{i \in c} P(i)E(i) \tag{2}$$

where,

- c is set of all classes for the attribute A2.
- P(i) is the probability associated with the i^th class of attribute A2.
- E(i^th) is the entropy of the i^th class of attribute A2.

The calculation of Play Golf attribute is :
E(PlayGolf, Outloook)
E(PlayGolf, Temperature)
E(PlayGolf, Humidity)
E(PlayGolf,windy)



		PlayGolf(14)		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5

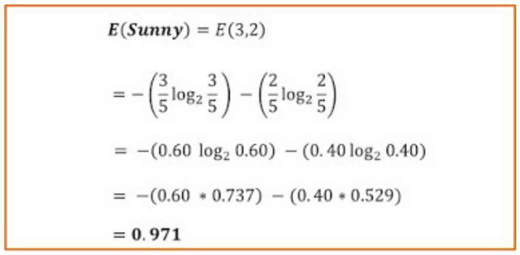
Figure 5: Table for Outlook and Play Golf

E(PlayGolf, Outlook) = P(Sunny) E(Sunny) + P(Overcast) E(Overcast) + P(Rainy) E(Rainy)

Which is same as:

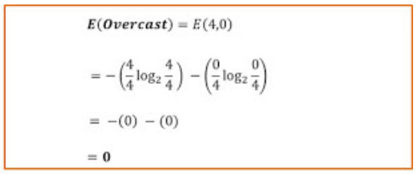
E(PlayGolf, Outlook) = P(Sunny) E(3,2) + P(Overcast) E(4,0) + P(rainy) E(2,3)

E(x,y) = E(y,x)
Therefore, E(2,3) = E(3,2)



$$\boldsymbol{E(Sunny)} = E(3,2)$$
$$= -\left(\frac{3}{5}\log_2\frac{3}{5}\right) - \left(\frac{2}{5}\log_2\frac{2}{5}\right)$$
$$= -(0.60\ \log_2 0.60) - (0.40 \log_2 0.40)$$
$$= -(0.60 * 0.737) - (0.40 * 0.529)$$
$$\boldsymbol{= 0.971}$$

Figure 6: Calculating E(3, 2)



$$\boldsymbol{E(Overcast)} = E(4,0)$$
$$= -\left(\frac{4}{4}\log_2\frac{4}{4}\right) - \left(\frac{0}{4}\log_2\frac{0}{4}\right)$$
$$= -(0) - (0)$$
$$\boldsymbol{= 0}$$

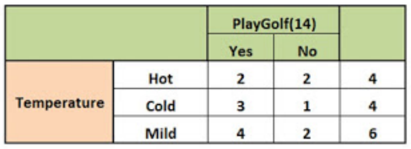
Figure 7: Calculating E(4, 0)

Therefore, the E(Play Golf, Outlook) is :

$$
\begin{aligned}
\boldsymbol{E}(PlayGolf, Outlook) &= \frac{5}{14}\boldsymbol{E}(3,2) + \frac{4}{14}\boldsymbol{E}(4,0) + \frac{5}{14}\boldsymbol{E}(2,3) \\
&= \frac{5}{14}0.971 + \frac{4}{14}0.0 + \frac{5}{14}0.971 \\
&= 0.357 * 0.971 + 0.0 + 0.357 * 0.971 \\
&= \mathbf{0.693}
\end{aligned}
$$

Figure 8: Calculating Entropy

Similarly,



		PlayGolf(14)		
		Yes	No	
Temperature	Hot	2	2	4
	Cold	3	1	4
	Mild	4	2	6

Figure 9: Table for Temperature and Play Golf

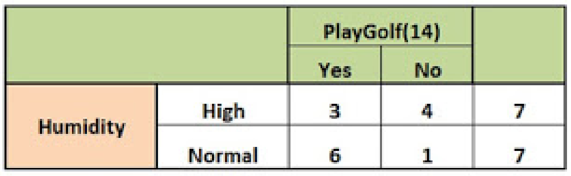
$$\text{E (PlayGolf, Temperature)} = \mathbf{4/14 * E}(\text{Hot}) + \mathbf{4/14 * E}(\text{Cold}) + \mathbf{6/14 * E}(\text{Mild})$$

$$\text{E (PlayGolf, Temperature)} = \mathbf{4/14 * E}(2, 2) + \mathbf{4/14 * E}(3, 1) + \mathbf{6/14 * E}(4, 2)$$

$$
\begin{aligned}
\text{E (PlayGolf, Temperature)} = \; & 4/14 * -(2/4 \log 2/4) - (2/4 \log 2/4) \\
+ \; & 4/14 * -(3/4 \log 3/4) - (1/4 \log 1/4) \\
+ \; & 6/14 * -(4/6 \log 4/6) - (2/6 \log 2/6)
\end{aligned}
$$

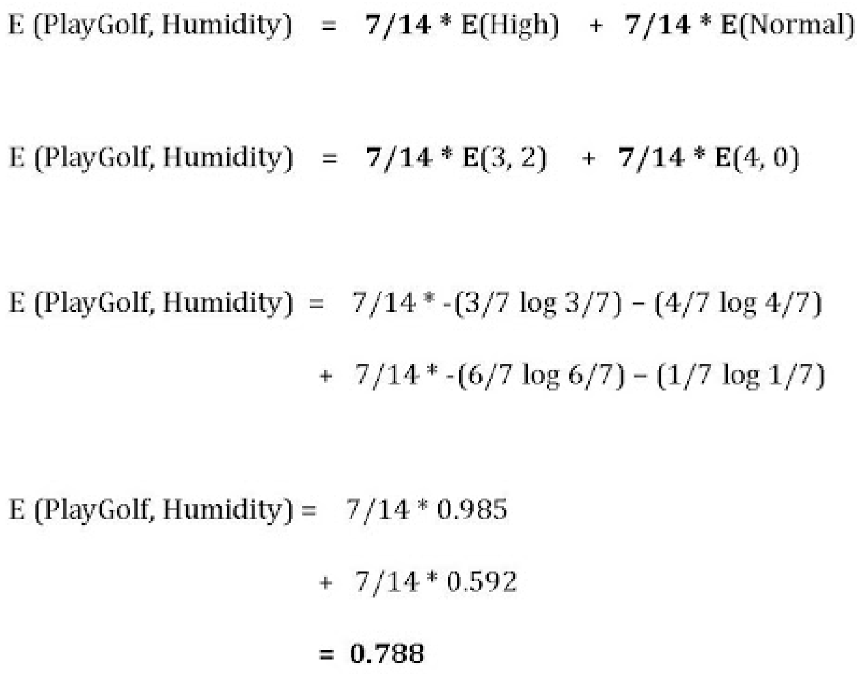
$$
\begin{aligned}
\text{E (PlayGolf, Temperature)} = \; & 5/14 * 1.0 \\
+ \; & 4/14 * 1.811 \\
+ \; & 5/14 * 0.918 \\
= \; & \mathbf{0.911}
\end{aligned}
$$

Figure 10: Calculating E(Play Golf, Temperature)



		PlayGolf(14)		
		Yes	No	
Humidity	High	3	4	7
	Normal	6	1	7

Figure 11: Table for Humidity and Play Golf



E (PlayGolf, Humidity) = **7/14 * E**(High) + **7/14 * E**(Normal)

E (PlayGolf, Humidity) = **7/14 * E**(3, 2) + **7/14 * E**(4, 0)

E (PlayGolf, Humidity) = 7/14 * -(3/7 log 3/7) – (4/7 log 4/7)

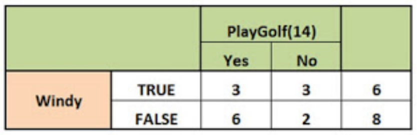
+ 7/14 * -(6/7 log 6/7) – (1/7 log 1/7)

E (PlayGolf, Humidity) = 7/14 * 0.985

+ 7/14 * 0.592

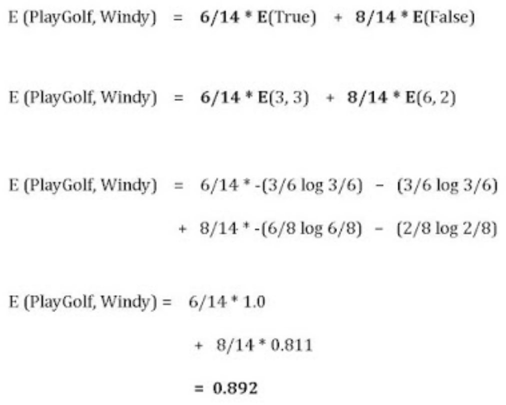
= 0.788

Figure 12: Calculating E(Play Golf, Humidity)



		PlayGolf(14)		
		Yes	No	
Windy	TRUE	3	3	6
	FALSE	6	2	8

Figure 13: Table for windy and Play Golf



E (PlayGolf, Windy) = **6/14 * E**(True) + **8/14 * E**(False)

E (PlayGolf, Windy) = **6/14 * E**(3, 3) + **8/14 * E**(6, 2)

E (PlayGolf, Windy) = 6/14 * -(3/6 log 3/6) – (3/6 log 3/6)
+ 8/14 * -(6/8 log 6/8) – (2/8 log 2/8)

E (PlayGolf, Windy) = 6/14 * 1.0
+ 8/14 * 0.811
= 0.892

Figure 14: Calculating E(Play Golf, y)

So now that we have all the entropies for all the four attributes, let's go ahead to summarize them as shown in below:

E(PlayGolf, Outloook) = 0.693
E(PlayGolf, Temperature) = 0.911
E(PlayGolf, Humidity) = 0.788
E(PlayGolf,windy) = 0.892

Step 4: Calculating Information Gain for Each Split

The next step is to calculate the information gain for each of the attributes. The information gain is calculated from the split using each of the attributes. Then the attribute with the largest information gain is used for the split.

The information gain is calculated using the formula:

$$Gain(S,T) = Entropy(S) – Entropy(S,T) \tag{3}$$

where,

- Entropy(S) is the entropy associated with attribute S.
- Entropy(S, T) is the joint entropy of attributes, S and T.

For example, the information gain after splitting using the Outlook attribute is given by:

Gain(PlayGolf, Outlook) = Entropy(PlayGolf) – Entropy(PlayGolf, Outlook)
= 0.94 – 0.693
= 0.247

Gain(PlayGolf, Temperature) = Entropy(PlayGolf) – Entropy(PlayGolf, Tempara-ture)
= 0.94 – 0.911
= 0.029

Gain(PlayGolf, Humidity) = Entropy(PlayGolf) – Entropy(PlayGolf, Humidity)
= 0.94 – 0.788
= 0.152

Gain(PlayGolf, windy) = Entropy(PlayGolf) – Entropy(PlayGolf, windy)
= 0.94 – 0.892
= 0.048

Having calculated all the information gain, we now choose the attribute that gives the highest information gain after the split.

Step 5: Perform the First Split

Now that we have all the information gain, we then split the tree based on the attribute with the highest information gain.

From our calculation, the highest information gain comes from Outlook. Therefore the split will look like this:

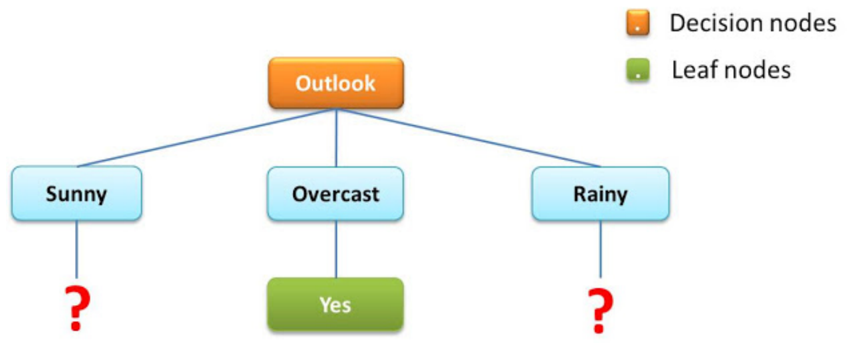
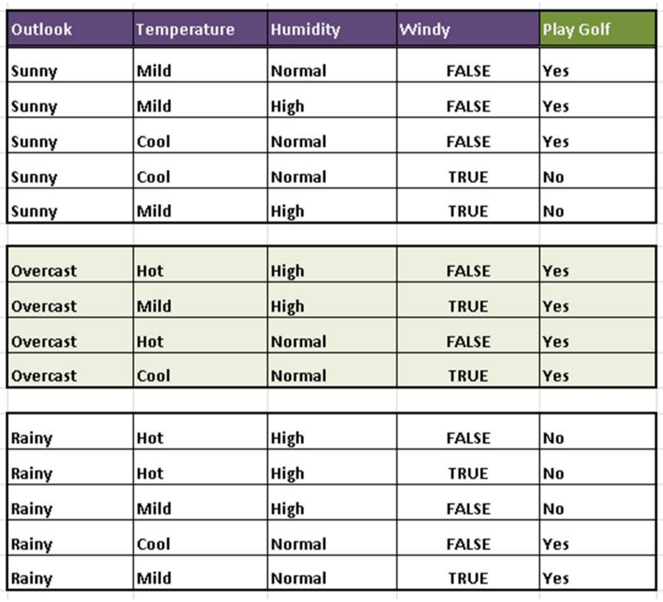


Figure 15: Decision Tree after first split

Now that we have the first stage of the decison tree, we see that we have one leaf node. But we still need to split the tree further.

To do that, we need to also split the original table to create sub tables. This sub tables are given in below.



Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Figure 16: Sub Tables

Step 6: Perform Further Splits

Now, we need to recalculate the entropies and the gain for the remaining attributes, and split with the attribute that has the maximum gain.

Note that, an attribute wont repeat in the subtree with that attribute as node.
Splitting for Rainy

From the above table its clear that Humidity gives an homogeneous split, i.e Entropy(Outlook = Rainy, Humidity) = 0
Therefore, lets split Rainy with Humidity

Similarly, when the Outlook is Sunny, the attribute windy gives an homogeneous split, thus we split Sunny with windy.

Step 7: Complete the Decision Tree

Keep on splitting the decision tree, until no other attribute is left, or if we get an homogeneous split.
Therefore, the final table is :

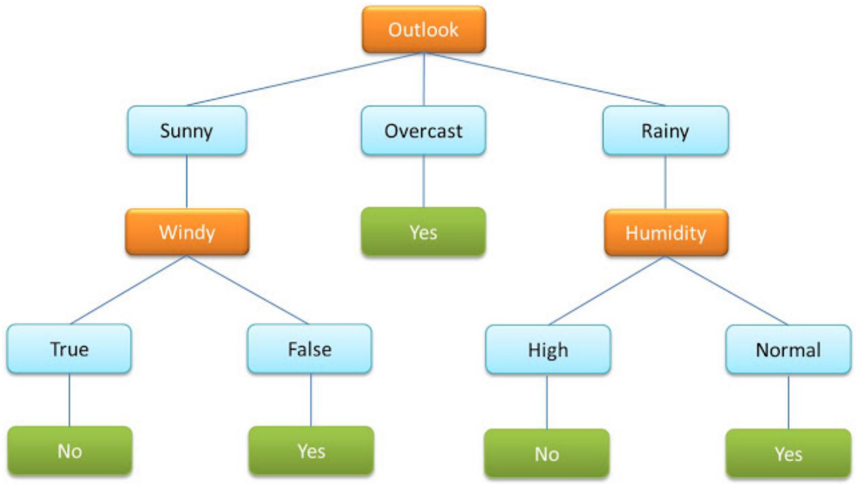


Figure 17: Sub Tables

1.3 Pseudo-Code for Decision Tree construction

```
Begin
Load learning sets  and create decision tree root node(rootNode), add learning
     set S into root not as its subset
For rootNode, compute Entropy(rootNode.subset) first
If Entropy(rootNode.subset) == 0 (subset is homogenious)
      return a leaf node

If Entropy(rootNode.subset)!= 0 (subset is not homogenious)
     compute Information Gain for each attribute left (not been used for
    spliting)
     Find attibute A with Maximum(Gain(S, A))
     Create child nodes for this root node and add to rootNode in the decision
     tree

For each child of the rootNode
   Apply ID3(S, A, V)
   Continue until a node with Entropy of 0 or a leaf node is reached
End
```

1.4 Additional Notes

1. For numerical attributes,

   Step 1: Sort the Attribute in ascending order.

   Step 2: Calculate the avg value for two adjacent datapoints.

   Step 3: Now consider each avg value to be a class, and calculate the Gain. Choose the average value with the most gain.

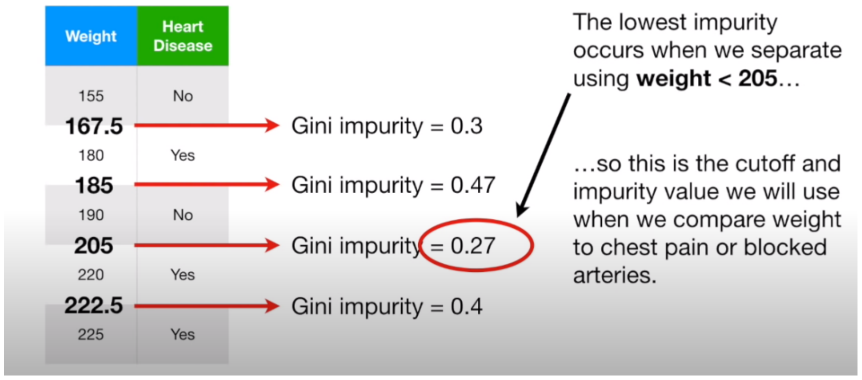
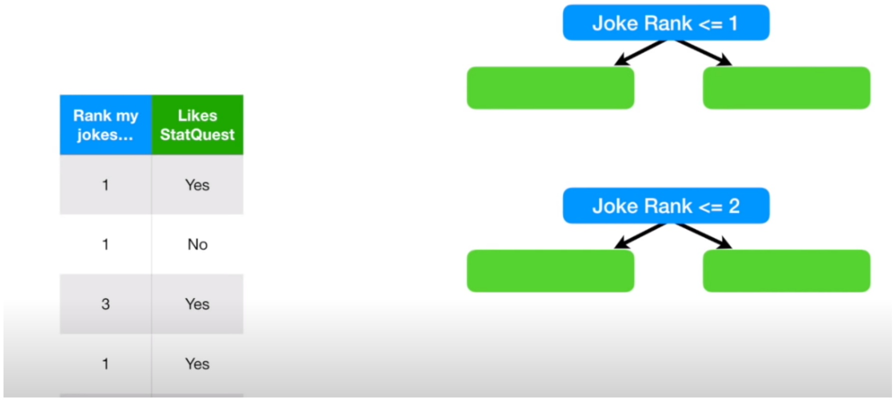

Weight
Heart Disease
155
No
167.5
Gini impurity = 0.3
180
Yes
185
Gini impurity = 0.47
190
No
205
Gini impurity = 0.27
220
Yes
222.5
Gini impurity = 0.4
225
Yes
The lowest impurity occurs when we separate using weight < 205...
...so this is the cutoff and impurity value we will use when we compare weight to chest pain or blocked arteries.

Figure 18: Illustration of the above steps. Note that here gini impurity is used and accordingly the one with the least impurity is chosen.

2. For attributes with rank as their value, we try to split based on each possible rank.



Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes

Figure 19: Illustration of the above steps. Note that the highest rank is avoided as all other ranks are less than it.

3. We can use other parameters to identify the attribute which splits the tree.
   The commonly used statistical measures are :

   - **Entropy and Information Gain**
     Entropy quantifies how much information there is in a random variable, or more specifically its probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy. Information gain is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees.

     $$Entropy(Attribute) = -\sum_{i \in c} p_i \log_2 p_i \tag{4}$$

     where,

     – c is set of all classes for the attribute.
     – p_i is the probability associated with that class.

     And then we find the information gain, and choose the attribute with the highest gain.

     $$Gain(S,T) = Entropy(S) – Entropy(S,T) \tag{5}$$

     where,

     – Entropy(S) is the entropy associated with attribute S.
     – Entropy(S, T) is the joint entropy of attributes, S and T.

- **Gini Impurity and Gini Gain**

  Gini Impurity is a measurement of the likelihood of an incorrect classification of a new instance of a random variable, if that new instance were randomly classified according to the distribution of class labels from the data set.

  $$G(Attribute) = \sum_{i \in c} p_i * (1 - p_i) \tag{6}$$

  where,

  - c is set of all classes for the attribute.
  - p_i is the probability associated with that class.

  And then we find the gini gains, similar to information gain, and choose the attribute with the highest gain.

  $$GG(Attribute) = 1 - Remainder_Impurity(attribute) \tag{7}$$

  $$RemainderImpurity(attribute) = \sum_{i \in B} p_i * (G(i)) \tag{8}$$

  where,

  - B is the, set of all branches for the attribute.
  - p_i is the probability associated with that branch.

  It seems that gini impurity and entropy are often interchanged in the construction of decision trees. Neither metric results in a more accurate tree than the other.

  All things considered, a slight preference might go to gini since it doesn't involve a more computationally intensive log to calculate.

- **Gain Ratio**

  In decision tree learning, Information gain ratio is a ratio of information gain to the intrinsic information.

  Information gain ratio biases the decision tree against considering attributes with a large number of distinct values. So it solves the drawback of information gain—namely, information gain applied to attributes that can take on a large number of distinct values might learn the training set too well. For example, suppose that we are building a decision tree for some data describing a business's customers. Information gain is often used to decide which of the attributes are the most relevant, so they can be tested near the root of the tree. One of the input attributes might be the customer's credit card number. This attribute has a high information gain, because it uniquely identifies each customer, but we do not want to include it in the decision tree: deciding how to treat a customer based on their credit card number is unlikely to generalize to customers we haven't seen before.

$$IntrinsicInforamtion(E_x, a) = -\sum_{v \in values(a)} \frac{|\{x \in E_x | value(x, a) = v\}|}{|E_x|} * log_2(\frac{|\{x \in E_x | value(x, a)}{|E_x|} \quad (9)$$

where,

- E_x is the dataset.
- value(x, a) returns the value of attribute a for the datapoint x.
- a A, where A is the set of all attributes of E_x.

$$GainRatio(S, A) = \frac{Gain(S, A)}{IntrinsicInformation(S, A)}) \quad (10)$$

where,

- S is the Target Variable.
- A is an attribute.