COS30018 – Intelligent Systems

Week 5 tutorial

This week we will cover the following items:

- Naïve Bayes (NB) with sklearn
- Decision Tree (DT) with sklearn

1. Naïve Bayes (NB) with sklearn

Naive Bayes is the most straightforward and fast classification algorithm based on Bayes' Theorem. Naive Bayes classifier is used in various applications such as spam filtering, text classification, sentiment analysis, and recommender systems (of course just in the past, now we have more powerful algorithms LOL).

The Bayes's theorem says:

$$P(c|x) = \frac{P(x|c).P(c)}{P(x)}$$

Where:

- $P(c|x)$: is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(x|c)$: is the probability of predictor given class.
- $P(c)$: is the prior probability of class.
- $P(x)$: is the prior probability of predictor.

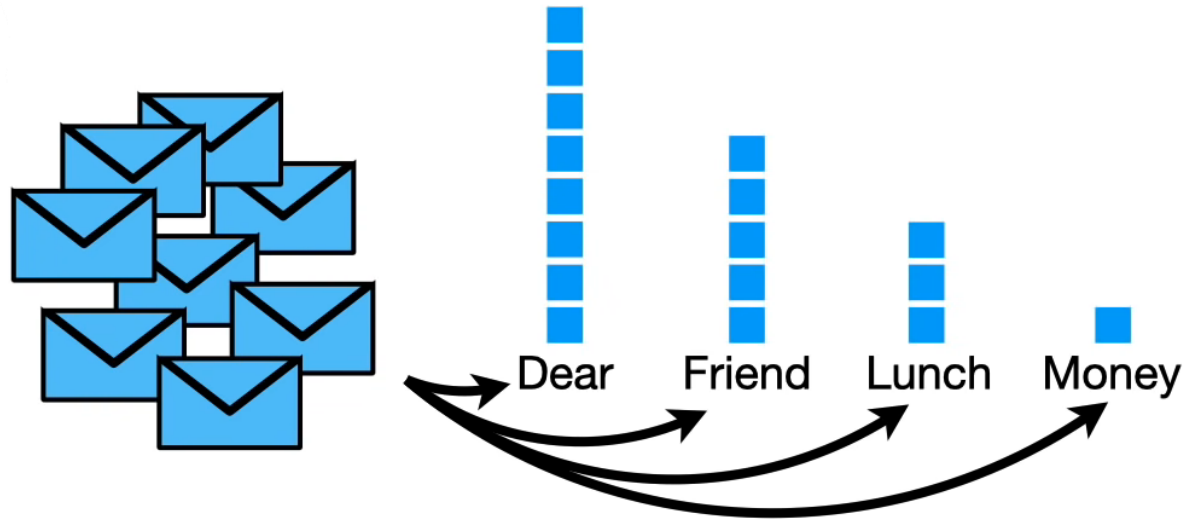
We will understand more about the meaning of each component when we look at an example.

Let's go straight to an example of how NB works as a supervised classification algorithm.

Scenario: We work for Gmail and our task is to classify if an email is spam or not.

Let's say in our dataset, we have **12 emails**: 8 normal emails and 4 spam emails.

The first thing to do is to count the occurrences of all words in the normal emails. Let's say there are only 4 words in the 8 normal emails:



Then calculate the probabilities of seeing each word, given that it was in normal email:

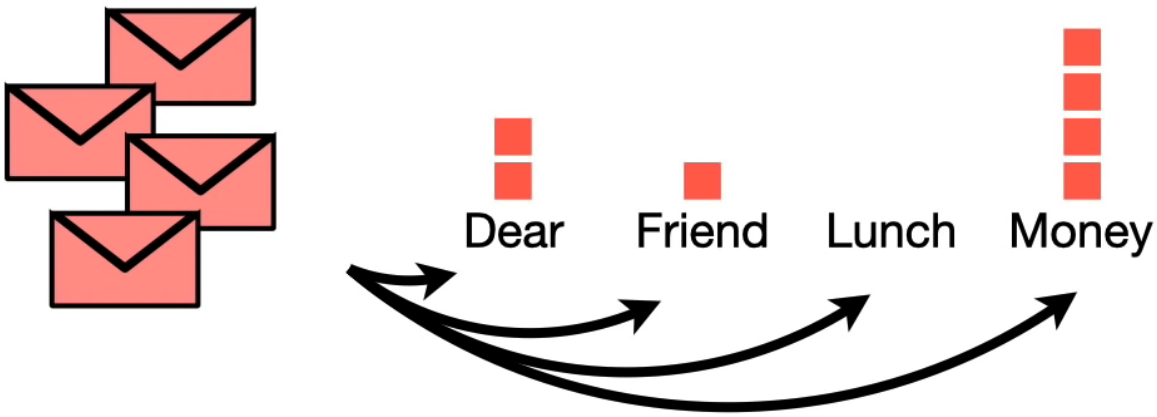
$$p(Dear|N) = \frac{8}{17} = 0.47$$

$$p(Friend|N) = \frac{5}{17} = 0.29$$

$$p(Lunch|N) = \frac{3}{17} = 0.18$$

$$p(Money|N) = \frac{1}{17} = 0.06$$

We do the same thing for spam emails:



$$p(Dear|S) = \frac{2}{7} = 0.29$$

$$p(Friend|S) = \frac{1}{7} = 0.14$$

$$p(Lunch|S) = \frac{0}{7} = 0$$

$$p(Money|S) = \frac{4}{7} = 0.57$$

Now, imagine if we got a new message that said “Dear Friend”. How can we predict if this is a normal email or a spam?

First, we calculate the score that “Dear Friend” gets if it is a normal email:

$$score(N) = p(N).p(Dear|N).p(Friend|N) = \frac{8}{12}.\frac{8}{17}.\frac{5}{17} = 0.09$$

This score is proportional to the probability that the message is normal, given that it says “Dear Friend”:

$$score(N) \sim p(N|Dear\ Friend)$$

We do the same to find the score that “Dear Friend” gets if it is a spam email:

$$score(S) = p(S).p(Dear|S).p(Friend|S) = \frac{4}{12}.\frac{2}{7}.\frac{1}{7} = 0.01$$

Again, this score is proportional to the probability that the message is spam, given that it says “Dear Friend”:

$$score(S) \sim p(S|Dear\ Friend)$$

And because $score(N) = 0.09 > score(S) = 0.01$, we can predict that “Dear Friend” is a normal email.

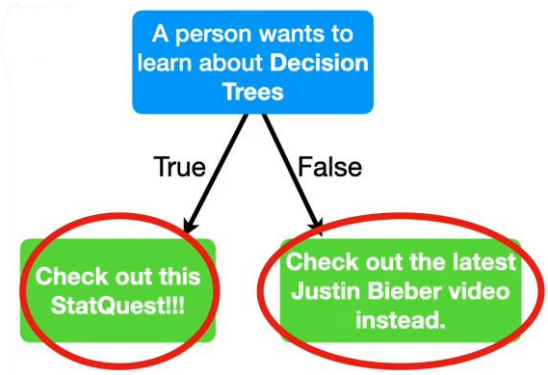
Implementation:

The implementation of Naïve Bayes classifier and explanations for each step are given in the notebook “Naive_Bayes.ipynb”. In this notebook, we use the library Scikit-learn to create and train the classifier.

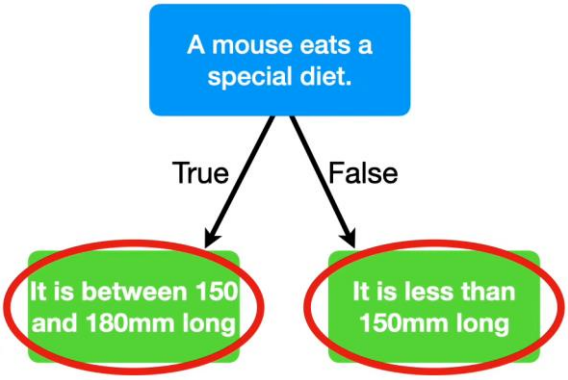
2. Decision Tree (DT) with sklearn

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.

When a Decision Tree classifies things into categories => it’s called a Classification Tree:



When a Decision Tree predicts numeric values => it’s called a Regression Tree:



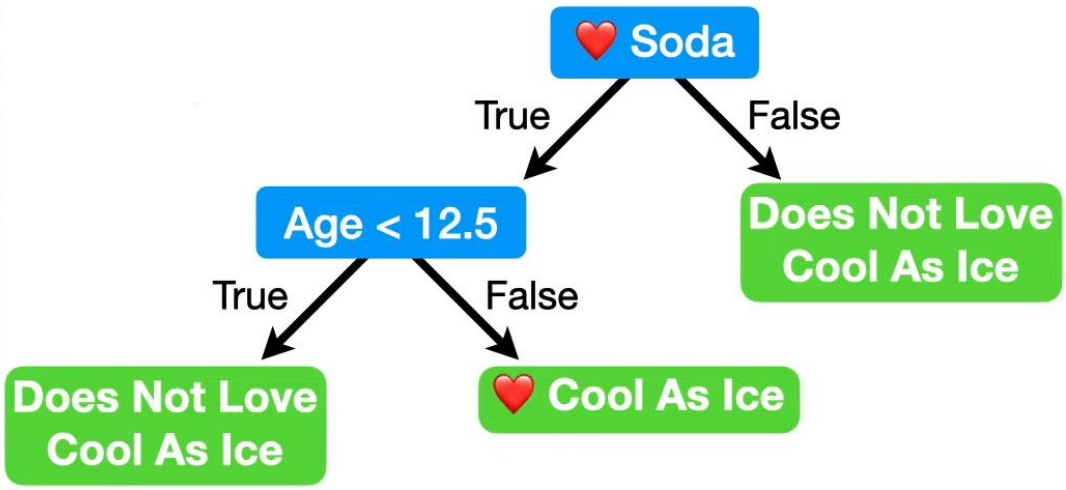
In this tutorial, we will be focusing on Classification Tree. So, how to create a Decision Tree?

Consider we have a dataset as below:

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

The blue columns are the features, and the green column is the outcome that we are interested in. Given a set of features of a person, we want to predict if that person loves Cool As Ice.

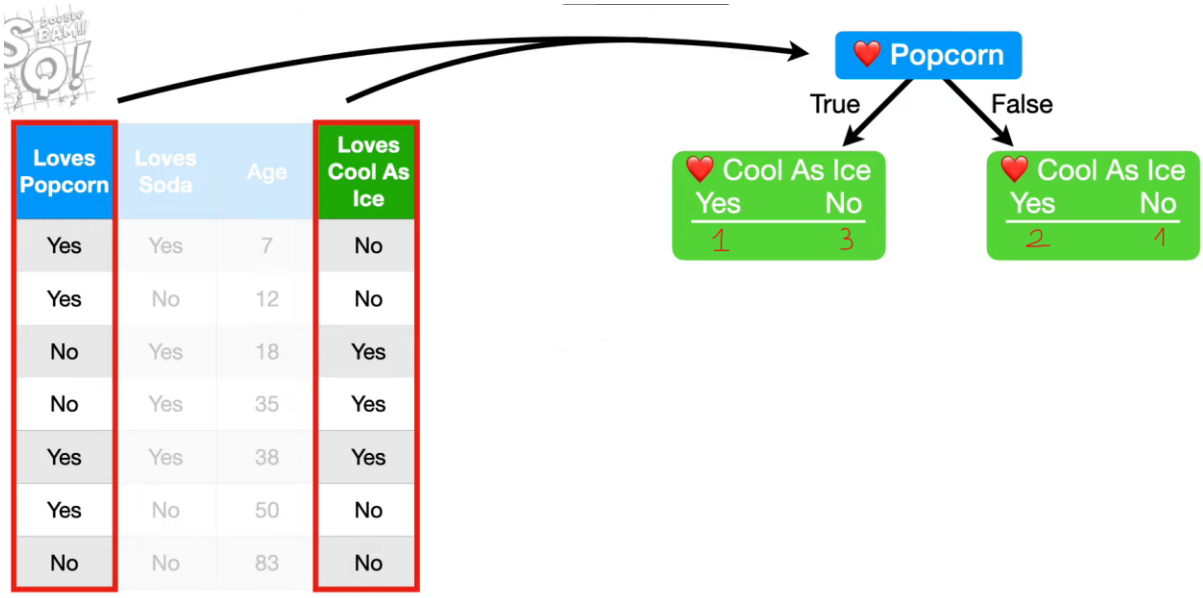
The output tree is something like this:



OK! Let's see how we can build this tree from the given dataset.

The 1st question you may ask is, how do we know whether Loves Popcorn, Loves Soda, or Age should be placed at the very top of the tree? The answer is we need to find the **"impurity"** of each feature, and the lower the impurity is, the higher that feature is placed in the tree. One of the most popular method to find impurity is called **Gini Impurity**. There are also other fancy sounding methods like **Entropy** and **Information Gain**. We will demonstrate the example with **Gini Impurity** since it is very popular and straightforward.

We will start by looking at how well Loves Popcorn predicts whether or not someone loves Cool As Ice:



Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

The impurity of the left leaf is:

$$Impurity\ (left) = 1 - (probability\ of\ \text{"yes"})^2 - (the\ probability\ of\ \text{"}no\text{"})^2$$

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375$$

And the impurity of the right leaf is:

$$Impurity(right) = 1 - (probability\ of\ \text{"yes"})^2 - (the\ probability\ of\ \text{"}no\text{"})^2$$

$$= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444$$

Thus, the total Gini Impurity of the parent node is:

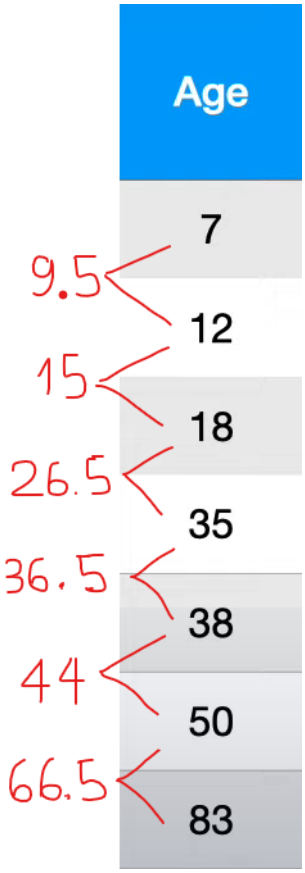
$$Impurity(parent) = weighted\ average\ of\ Gini\ Impurities\ of\ children\ nodes$$

$$= \left(\frac{4}{4+3}\right)0.375 + \left(\frac{3}{4+3}\right)0.444 = 0.405$$

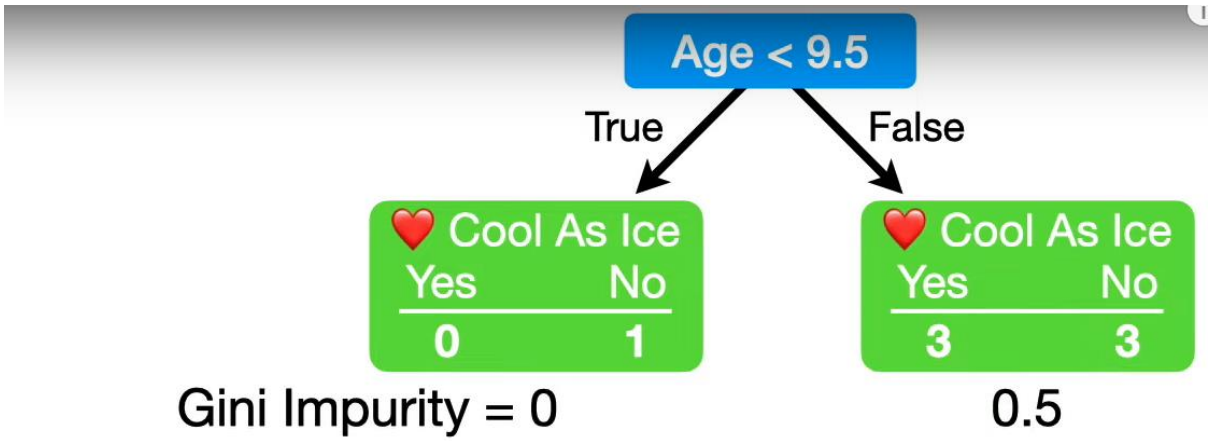
That is, the Gini Impurity of **Love Popcorn** feature is 0.405.

Likewise, the Gini Impurity of **Love Soda** feature is 0.214.

On the other hand, the Gini Impurity of **Age** is different, because Age contains numeric data, not categorical data like Yes/No. The first thing is to **sort** the Age from lowest to highest value, then calculate the **average** age for all adjacent numbers:

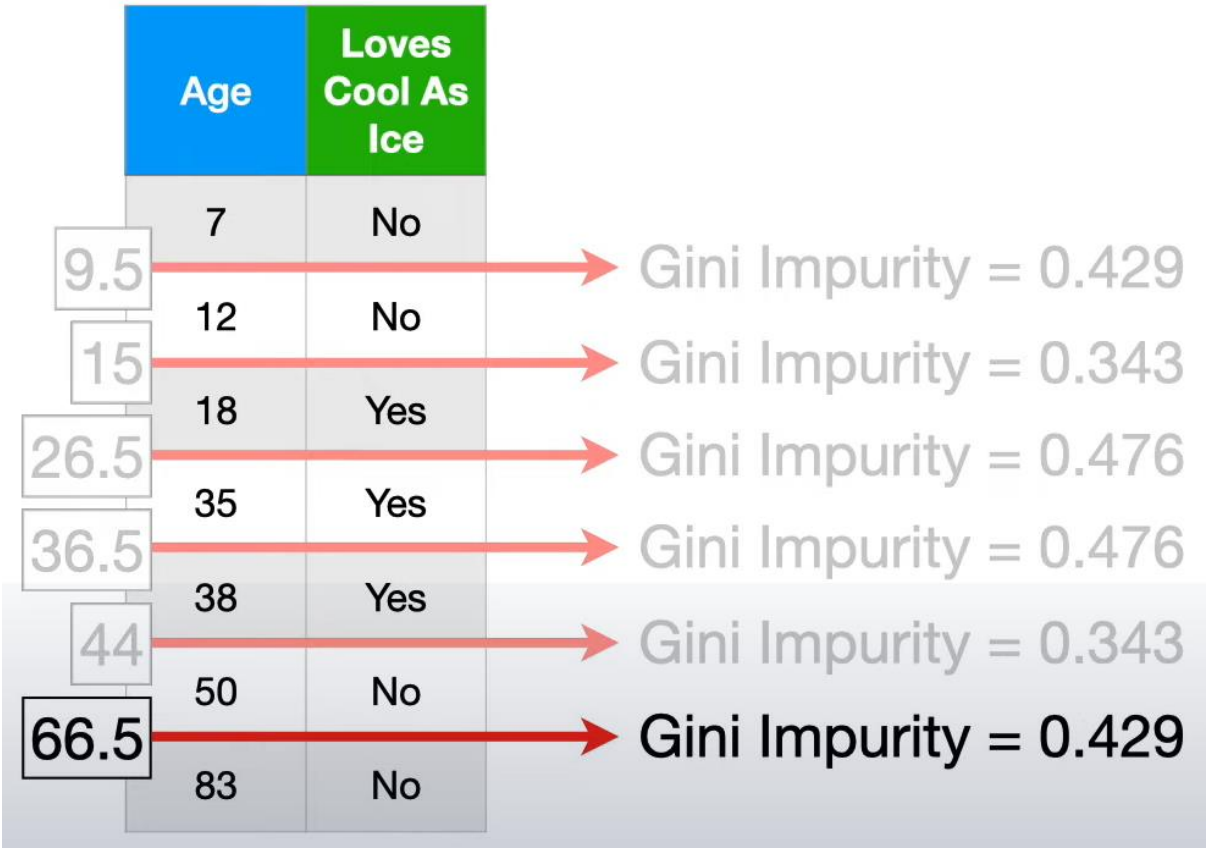


For each average age, we will calculate a Gini Impurity of it, follow the same steps we did for Loves Popcorn and Love Soda. For example, a Gini Impurity of Age < 9.5 can be calculated as below:



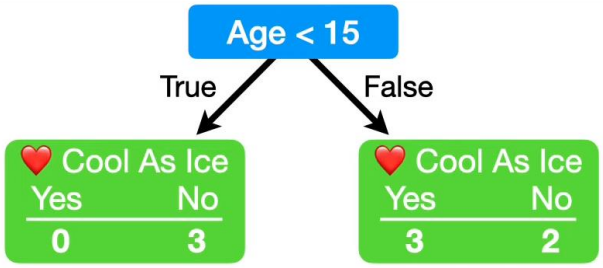
$$\text{Total Gini Impurity} = \left(\frac{1}{1+6}\right) 0 + \left(\frac{6}{1+6}\right) 0.5 = 0.429$$

Likewise, we calculate the Gini Impurity for all other age values:



Since 15 and 44 are tied for the lowest Impurity => we can pick either one, we will choose 15 for this example:

Gini Impurity for Age < 15 = 0.343

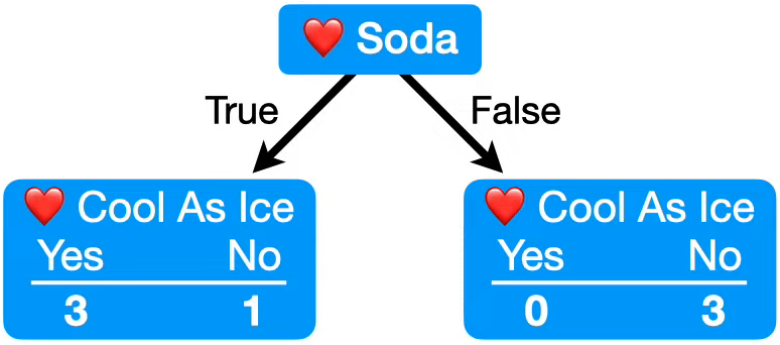


Now we can compare the Gini Impurity of 3 features to decide which one to go to the top of the tree:

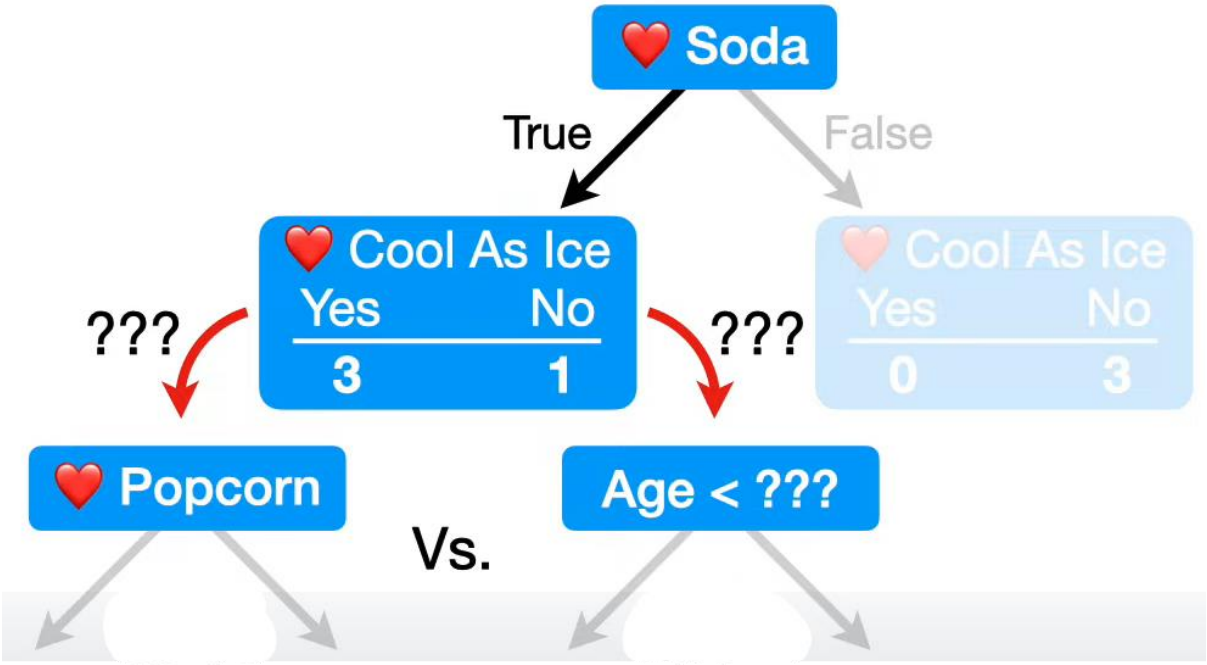
- Gini Impurity of Loves Popcorn = 0.405
- Gini Impurity of Loves Soda = 0.214
- Gini Impurity of Age < 15 = 0.343

Because Love Soda has the lowest Impurity => we put it at the top of the tree.

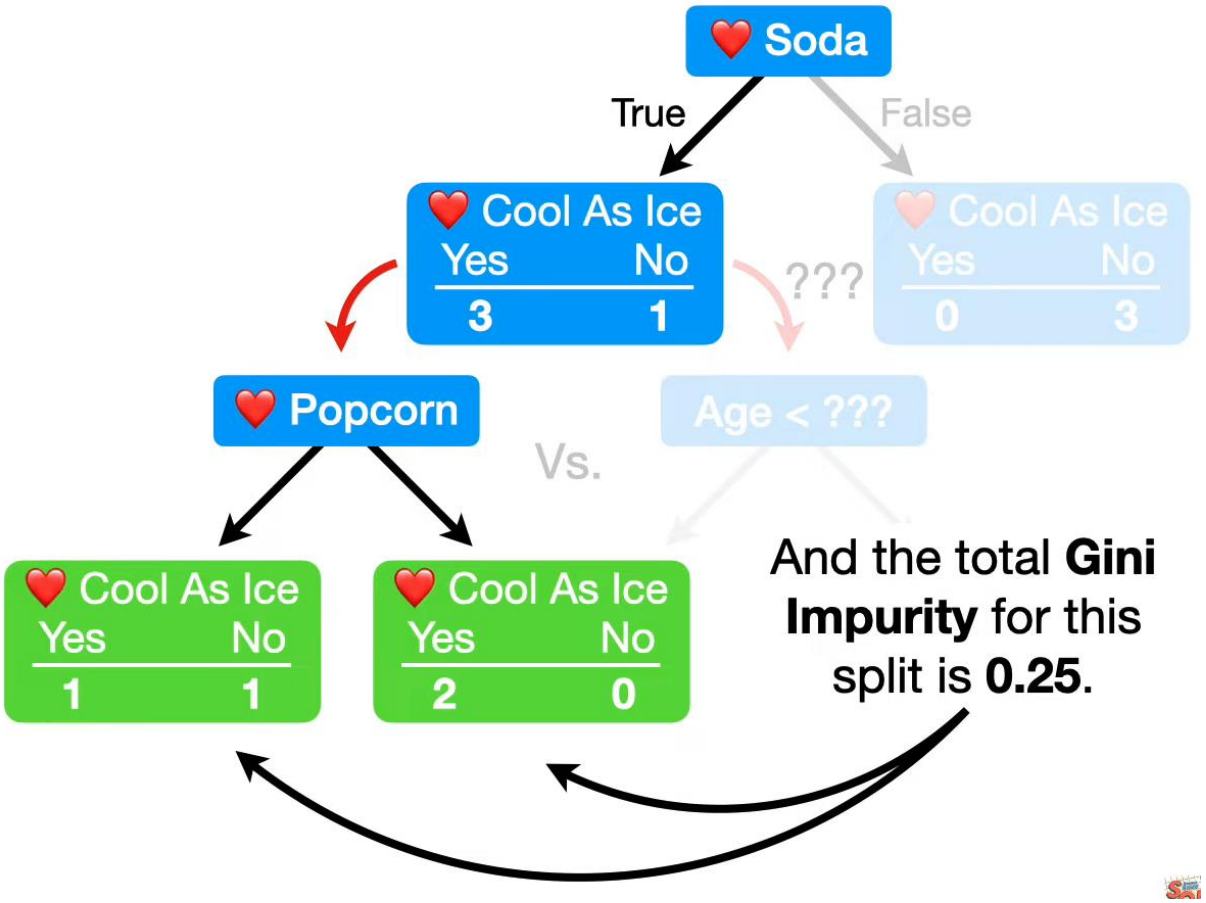
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



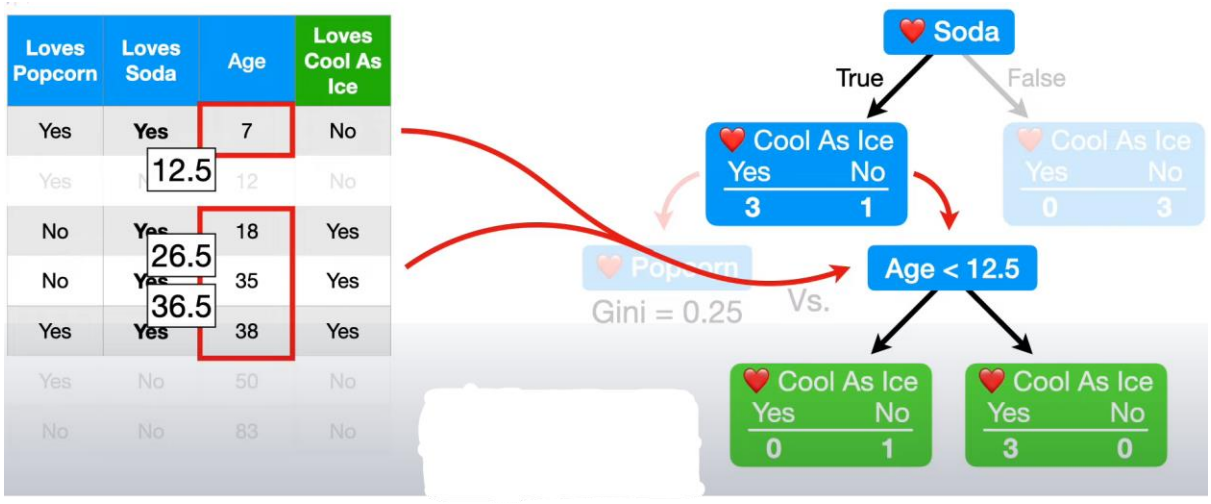
Since the right node is pure already and the left node is impure, so we need to reduce the Impurity by splitting people that Loves Soda based on Loves Popcorn and Age. In order to know which next feature we should use to split, we will calculate the Gini Impurity of Loves Popcorn and Age:



For Loves Popcorn, we can calculate the Gini Impurity and get 0.25:

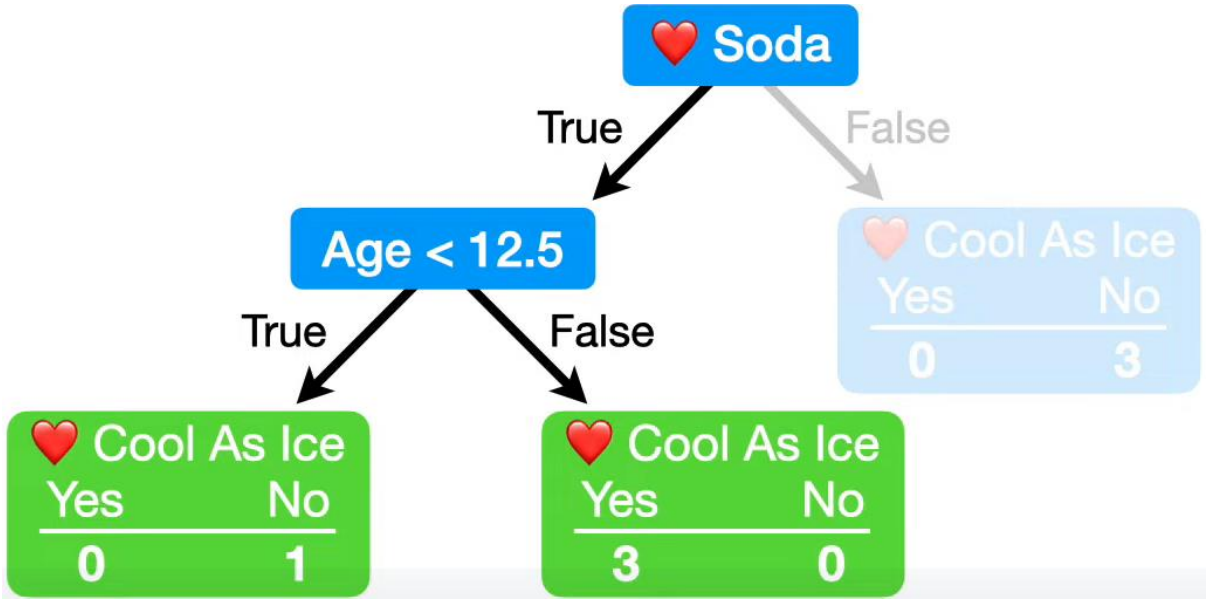


For Age, we just take ages of people who love soda, because we are under the branch "Loves Soda – Yes":



And Age < 12.5 has the lowest impurity = 0, because both leaves have no impurity at all.

Since the Gini Impurity of Loves Popcorn (0.25) > the Gini Impurity of Age < 12.5 (0), we will use "Age < 12.5" to split this node into leaves:

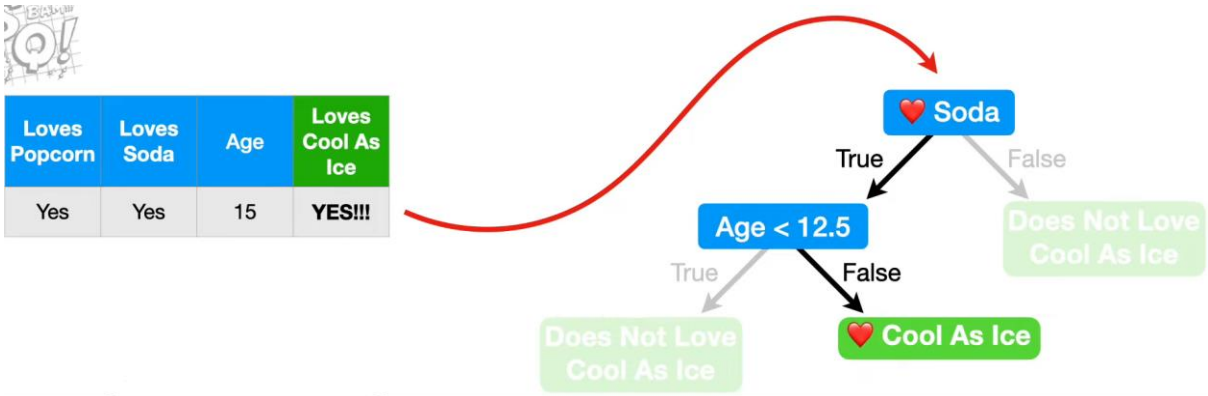


All leaves are pure, so there is no reason to continue splitting any more leaves.

Now, if someone new comes along and given the data as below:

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	15	???

We can use our Decision Tree to predict if he/she loves Cool As Ice:



Implementation:

The implementation of a Decision Tree and explanations for each step are given in the notebook "Decision_Tree.ipynb". In this notebook, we use the library Scikit-learn to create and train the model.

References:

- StatQuest: Decision and Classification Trees, Clearly Explained!!! https://www.youtube.com/watch?v=_L39rN6gz7Y
- StatQuest: Naive Bayes, Clearly Explained!!! https://www.youtube.com/watch?v=O2L2Uv9pdDA