**Lab6:**

**Decision Tree and Random Forest Algorithms**

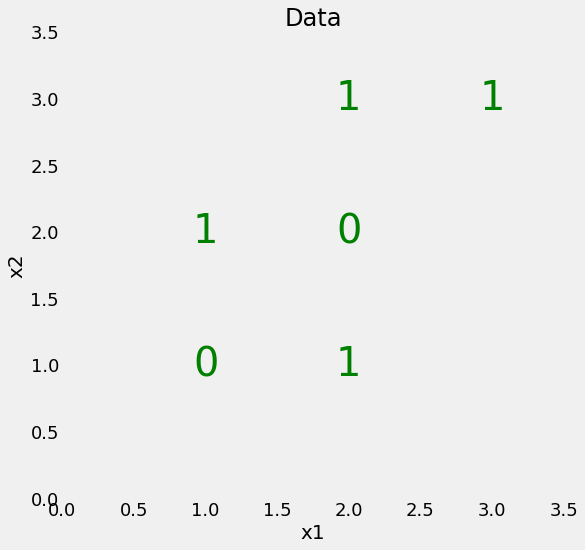
**Introduction:**

A decision tree is the building block of a random forest and by itself is a rather intuitive model. We can think of decision trees as a flowchart of questions asked about our data, eventually leading to a predicted class (or continuous value in the case of regression). This is an interpretable model because it makes decisions how we do in real life: we ask a series of questions about the data until we eventually have arrived at a decision.

The main technical details of a decision tree are in how the questions about the data are constructed. A decision tree is built by forming questions that lead to the greatest reduction in Gini Impurity. We’ll get into Gini Impurity a little later, but what this means is that the decision tree tries to form nodes that are as pure as possible, containing a high proportion of samples (data points) from only one class. Gini Impurity and constructing the tree may be a little tough to understand, so first let’s build a Decision Tree, and then we can work through some simple math.

**Q1: Understand the decision tree with simple example:**

We’ll start with a very simple binary classification problem as shown below:



Our data only has two features (predictor variables) and there are a total of 6 data points with 2 different labels.

Although this problem is simple, it’s not linearly separable, which means that we can’t draw a single straight line through the data to classify the points. We can however draw a series of straight lines that divide the classes, which is essentially what the decision tree will do as it forms the series of questions.

import numpy as np

import pandas as pd

# Set random seed to ensure reproducible runs

RSEED = 50

X = np.array([[2, 2],

[2, 1],

[2, 3],

[1, 2],

[1, 1],

[3, 3]])

y = np.array([0, 1, 1, 1, 0, 1])

**Q1:**

1) Apply decision tree algorithm on this dataset and draw decision and find the number of nodes and maximum depth of tree.

2) Use different depth of tree and check the result. Whether the result is good or bad?

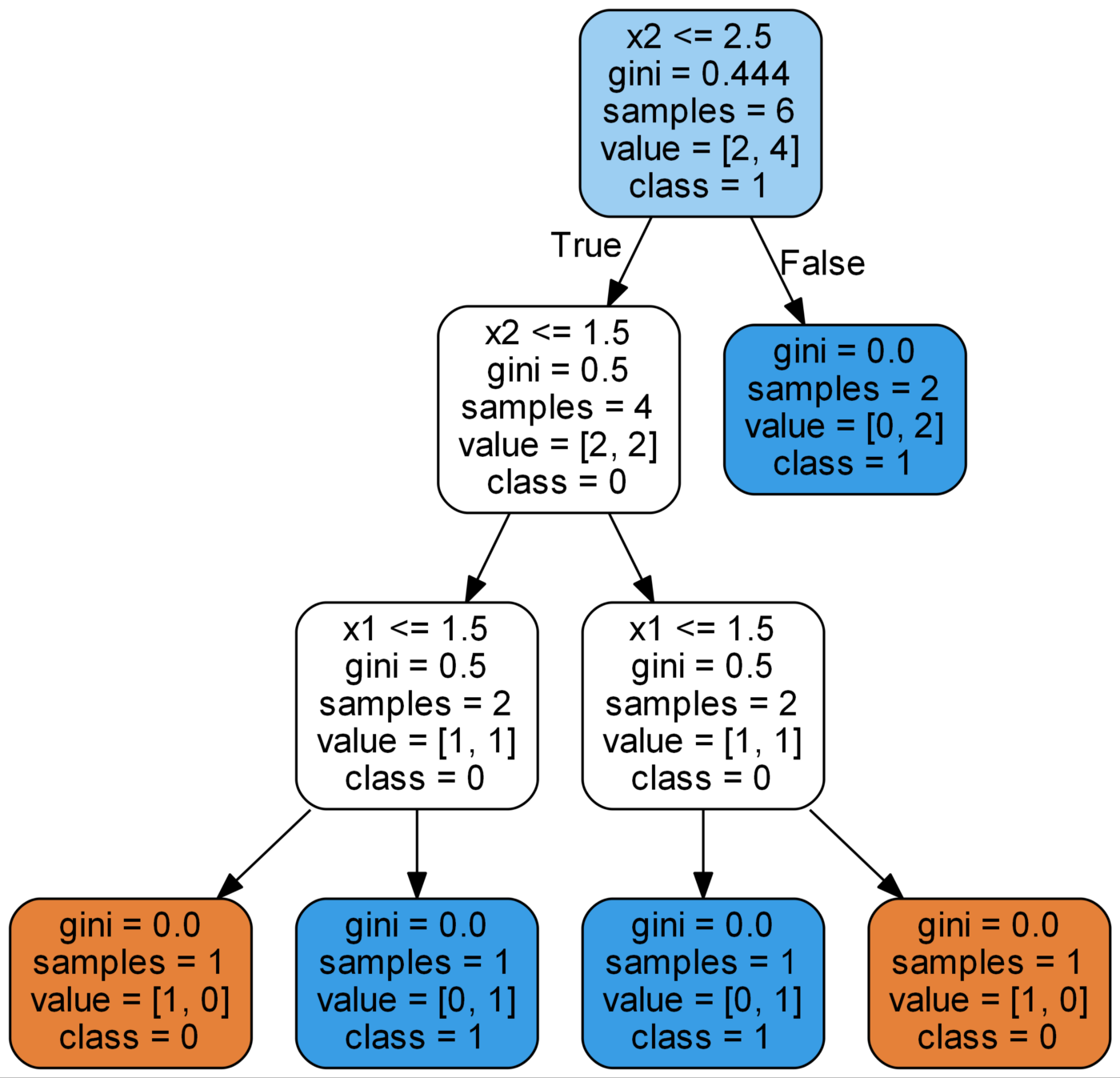
Simply install graphviz and set path using following command to show the decision tree.

# conda install python-graphviz

import os

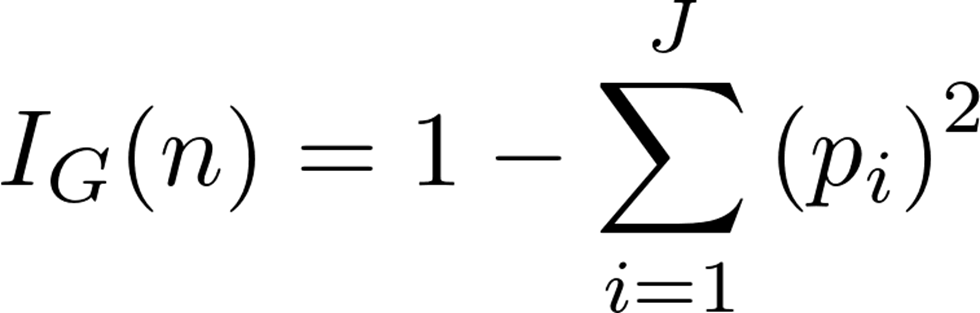
os.environ['PATH'] = os.environ['PATH']+';'+os.environ['CONDA\_PREFIX']+r"\Library\bin\graphviz"

The graph should look like this.

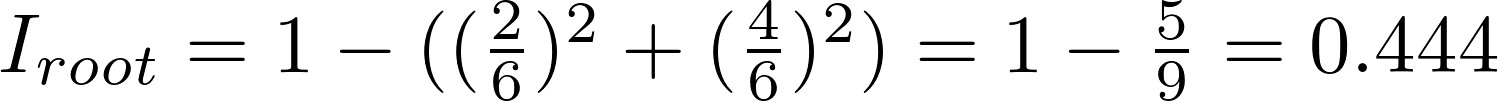


**Gini Impurity**

At this point it’ll be helpful to dive into the concept of Gini Impurity (the math is not intimidating!) The Gini Impurity of a node is the probability that a randomly chosen sample in a node would be incorrectly labeled if it was labeled by the distribution of samples in the node. For example, in the top (root) node, there is a 44.4% chance of incorrectly classifying a data point chosen at random based on the sample labels in the node. We arrive at this value using the following equation:



The Gini Impurity of a node n is 1 minus the sum over all the classes J (for a binary classification task this is 2) of the fraction of examples in each class p\_i squared. That might be a little confusing in words, so let’s work out the Gini impurity of the root node.



Gini Impurity of the root node

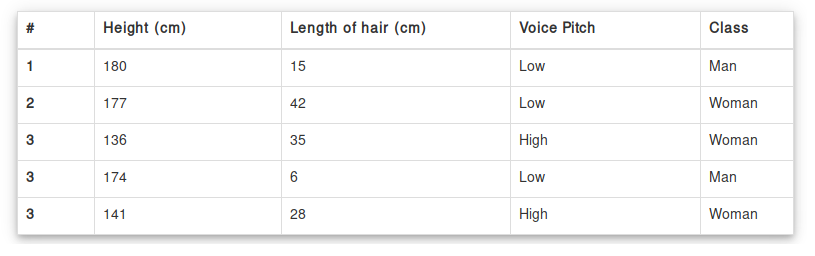
It then repeats this splitting process in a greedy, [recursive procedure](http://scikit-learn.org/stable/modules/tree.html#tree) until it reaches a maximum depth, or each node contains only samples from one class. The weighted total Gini Impurity at each level of tree must decrease. At the second level of the tree, the total weighted Gini Impurity is 0.333:



(The Gini Impurity of each node is weighted by the fraction of points from the parent node in that node.) You can continue to work out the Gini Impurity for each node (check the visual for the answers).

**Q2: We have the training dataset in the following example:**

Three features and two labels



# Data Collection

X = [ [180, 15,0],

[177, 42,0],

[136, 35,1],

[174, 65,0],

[141, 28,1]]

Y = ['man', 'woman', 'woman', 'man', 'woman']

data\_feature\_names = [ 'height', 'hair length', 'voice pitch' ]

Perform decision tree algorithm on the health dataset. Do the complete analysis.

I have provided the python file.

**Random Forest**

Now we can move on to a more powerful model, the random forest. This takes the idea of a single decision tree, and creates an ensemble model out of hundreds or thousands of trees to reduce the variance. Each tree is trained on a random set of the observations, and for each split of a node, only a subset of the features are used for making a split. When making predictions, the random forest averages the predictions for each of the individual decision trees for each data point in order to arrive at a final classification.

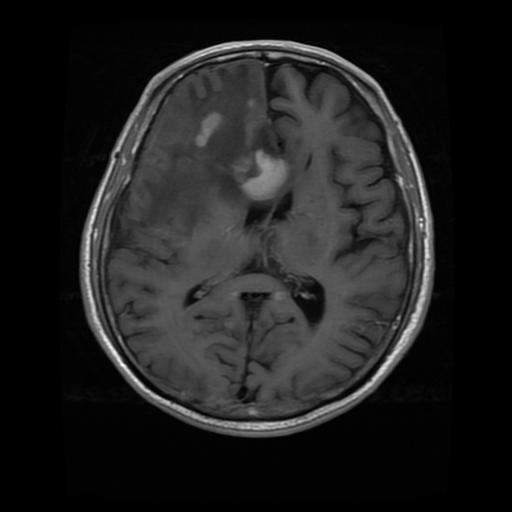
**Q3: perform random forest as provided the health dataset. Do complete analysis.**

**Q4: perform decision tree and random forest for regression dataset. Do the analysis and optimize various parameters.**

**Q5: I have provided the brain tumor dataset for classification.**

1. Construct Feature matrix using BOW or other image-based features.

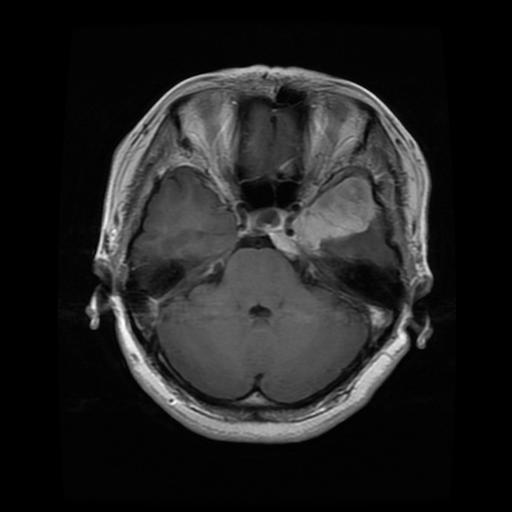
2. Compare the decision tree and random forest. Do the analysis and check various number of parameters.



**Bag of visual words or other features**

Split features into training and testing

Random Forest and Decision Tree classifiers



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Samples | F1 | F2 | fn | labels |
| 1 | -- | -- | -- | 1 |
| 2 | -- | -- | -- | 1 |
| … | -- |  |  | 2 |
| n | -- | -- | -- | 3 |

