Decision Tree Learning

By
Dr. Junaid Qazi, PhD
https://www.linkedin.com/in/jqazi/

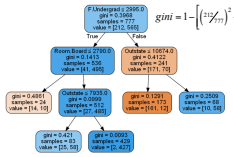
Learning objectives:

- *What are trees, their visual representation and how do they work*
- *What is a split in trees, how to find the best split*
- *Advantages / disadvantages*

✅*Recommended Readings and References:*
sklearn's Official Documentation - 1.10. Decision Trees
Introduction to Statistical Learning - Chapter 8
Machine Learning - A Probabilistic Perspective - Chapter 16

Data Science from Scratch – Part 2: Business Machine Learning

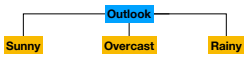
$$gini = 1 - \left[\left(\tfrac{215}{777}\right)^2 + \left(\tfrac{562}{777}\right)^2\right] = 0.3968$$



A typical decision tree visualization using one of the library in Python

Decision Trees - key concepts

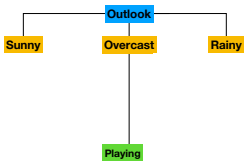
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.



Outlook	Windy	Humidity	Playing
Sunny	T	High	No
Overcast	T	Normal	Yes
Rainy	T	Normal	Yes
Rainy	T	High	No
Sunny	F	Normal	Yes
.	.	.	.
.	.	.	.
.	.	.	.

Decision Trees - key concepts

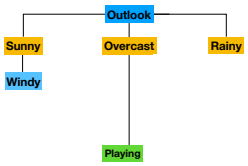
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.



Outlook	Windy	Humidity	Playing
Sunny	T	High	No
Overcast	T	Normal	Yes
Rainy	T	Normal	Yes
Rainy	T	High	No
Sunny	F	Normal	Yes
.	.	.	.
.	.	.	.
.	.	.	.

Decision Trees - key concepts

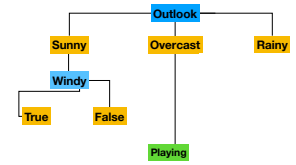
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

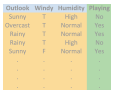


Outlook	Windy	Humidity	Playing
Sunny	T	High	No
Overcast	T	Normal	Yes
Rainy	T	Normal	Yes
Rainy	T	High	No
Sunny	F	Normal	Yes
.	.	.	.
.	.	.	.
.	.	.	.

Decision Trees - key concepts

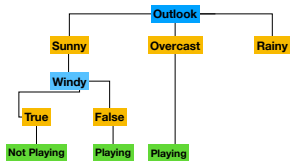
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

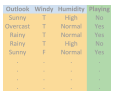




Decision Trees - key concepts

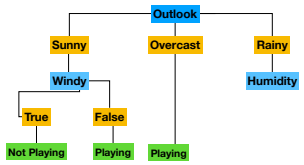
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

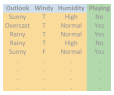




Decision Trees - key concepts

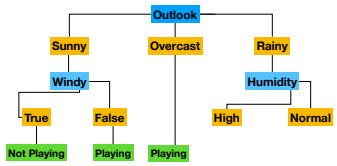
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

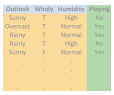




Decision Trees - key concepts

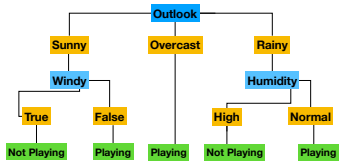
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

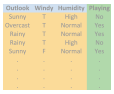




Decision Trees - key concepts

Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.



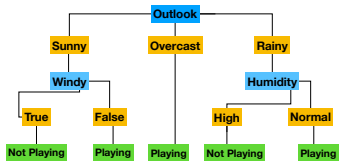


Decision trees:

- type of model is used for both **classification** and **regression** (CART).

Decision Trees - key concepts

Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.

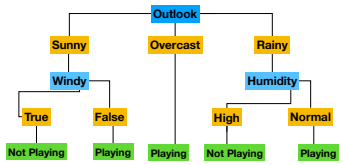


Decision trees:

- type of model is used for both **classification** and **regression** (CART).
- **answer sequential questions** and send us down along a certain route of the tree to find the outcomes / result class.

Decision Trees - key concepts

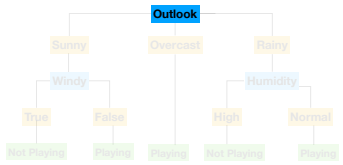
Let's create a decision tree for a very simple situation.
My son wants to practice soccer. Before I decide to go out with him, I have few questions on weather.



Decision trees:

- type of model is used for both **classification** and **regression** (CART).
- **answer sequential questions** and send us down along a certain route of the tree to find the outcomes / result class.
- model behaves like "**if this, then that**" conditions ultimately yielding a specific result.

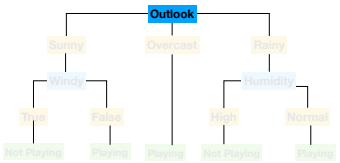
Decision Trees - key concepts



Decision trees:

- **Root Node**: The starting point of the tree, e.g. "Outlook"

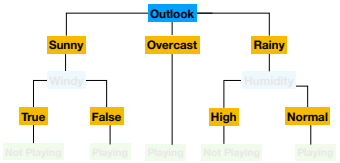
Decision Trees - key concepts



Decision trees:

- **Root Node**: The starting point of the tree, e.g. “Outlook”
- **Branches**: Arrows/lines connecting nodes, s

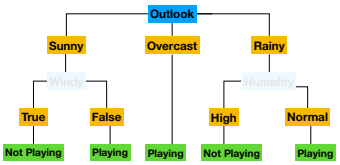
Decision Trees - key concepts



Decision trees:

- **Root Node**: The starting point of the tree, e.g. “Outlook”
- **Branches**: Arrows/lines connecting nodes, showing the flow from question to answer - segments of the trees that connect the nodes

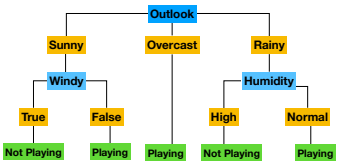
Decision Trees - key concepts



Decision trees:

- **Root Node**: The starting point of the tree, e.g. “Outlook”
- **Branches**: Arrows/lines connecting nodes, showing the flow from question to answer - segments of the trees that connect the nodes
- **Leaf Node**: Terminal nodes that predict the outcomes

Decision Trees - key concepts



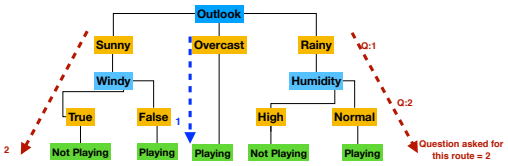
Decision trees:

- **Root Node**: The starting point of the tree, e.g. “Outlook”
- **Branches**: Arrows/lines connecting nodes, showing the flow from question to answer - segments of the trees that connect the nodes
- **Leaf Node**: Terminal nodes that predict the outcomes

Windy
Humidity

Nodes split for the value of a certain feature / attribute are internal nodes

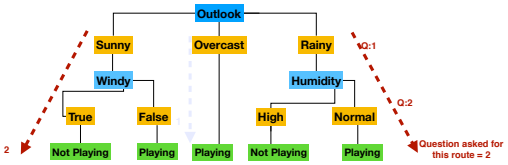
Decision Trees - key concepts



Decision trees:

- Both root and leaf nodes contain questions or criteria to be answered along the route between them.
- **Depth of the tree** is important and represents **how many questions are asked** before we reach the leaf node (prediction class)

Decision Trees - key concepts

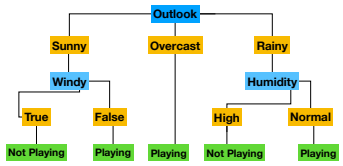


Decision trees:

- Both root and leaf nodes contain questions or criteria to be answered along the route between them.
- **Depth of the tree** is important and represents **how many questions are asked** before we reach the leaf node (prediction class)
- The overall tree depth is denoted by its **longest route**.

Depth is 2 for both Sunny and Rainy routes whereas 1 for the overcast route. **Our tree has a depth of 2.**

Decision Trees - key concepts



Now, another thing that we can notice in the above tree, is **Splitting** at the nodes.
Well, a relevant and the most important question is **"how to decide a split, that could be THE BEST under given circumstances?"**

Let's move on and try to understand this with a simple example!

Decision Trees - Splitting at nodes

The best split!
Let's learn with a very simple example in which, we have three features A, B & C with two target classes **Green** and **Blue**

A	B	C	Target
1	1	1	Green
1	1	0	Green
0	0	1	Blue
1	0	0	Blue

Decision Trees - Splitting at nodes

The best split!
Let's learn with a very simple example in which, we have three features A, B & C with two target classes **Green** and **Blue**

A	B	C	Target
1	1	1	Green
1	1	0	Green
0	0	1	Blue
1	0	0	Blue

Splitting at B, gives the clear separation between the target classes

We have 1 for green and 0 for blue

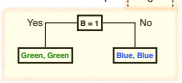
Decision Trees - Splitting at nodes

The best split!
Let's learn with a very simple example in which, we have three features A, B & C with two target classes **Green** and **Blue**

A	B	C	Target
1	1	1	Green
1	1	0	Green
0	0	1	Blue
1	0	0	Blue

Splitting at B, gives the clear separation between the target classes

We have 1 for green and 0 for blue



A		B		C	
0	1	0	1	0	1
	Green		Green		Green
	Green		Green	Green	
Blue		Blue			Blue
	Blue	Blue		Blue	

On **left**, we have created a simple **split chart** for A, B and C. Clearly, A and C do not split the classes perfectly.

(We will do the mathematics on splitting in the coming slides)

Intuition is to choose the feature that gives the best split, this is to maximize the information gain from the split.

Let's talk about the Mathematical Methods behind split:

Entropy & Information Gain

Gini Index

Decision Trees - Splitting at nodes

Entropy & Information Gain

Entropy, H, is a common way to measure the level of impurity in a group, it comes from the information theory* - *higher the entropy is, more the information contents are.*

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i is the probability of i

* Link to the article published in 1948 "A Mathematical Theory of Communication" by Claude E. Shannon, the father of information theory.

Decision Trees - Splitting at nodes

Entropy & Information Gain

Entropy, H, is a common way to measure the level of impurity in a group, it comes from the information theory* - *higher the entropy is, more the information contents are.*

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i is the probability of i

Information Gain, IG, is based on entropy. It decides which feature to split on at each step in building the decision tree.

$IG = H\_$**parent node** $- H\_$**child nodes** (weighted sum of entropy of the children)

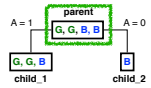
Let's do some mathematics to see the splitting by calculating Entropy and IG for our example dataset.

* Link to the article published in 1948 "A Mathematical Theory of Communication" by Claude E. Shannon, the father of information theory.

Decision Trees - Splitting at nodes

Entropy & Information Gain

Splitting at A:



H (parent) = – (2/4) $\log_2$ (2/4) – (2/4) $\log_2$ (2/4)
= – (0.5) (– 1) – (0.5) (– 1) = 1

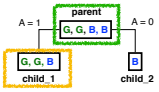
2 out of 4 from blue and 2 out of 4 from green class

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Decision Trees - Splitting at nodes

Entropy & Information Gain

Splitting at A:



H (parent) = – (2/4) $\log_2$ (2/4) – (2/4) $\log_2$ (2/4)
= – (0.5) (– 1) – (0.5) (– 1) = 1

H (child_1) = – (1/3) $\log_2$ (1/3) – (2/3) $\log_2$ (2/3)
= –(1/3) (– 1.58) – (2/3) (– 0.58) = 0.92

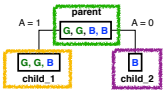
1 out of 3 from blue and 2 out of 3 from green class

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Decision Trees - Splitting at nodes

Entropy & Information Gain

Splitting at A:



H (parent) = – (2/4) $\log_2$ (2/4) – (2/4) $\log_2$ (2/4)
= – (0.5) (– 1) – (0.5) (– 1) = 1

H (child_1) = – (1/3) $\log_2$ (1/3) – (2/3) $\log_2$ (2/3)
= –(1/3) (– 1.58) – (2/3) (– 0.58) = 0.92

H (child_2) = – (1) $\log_2$ (1) = 0

1 out of 1 from blue class

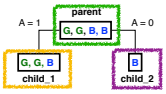
A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Decision Trees - Splitting at nodes

Entropy & Information Gain

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Splitting at A:



H (parent) $= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$
$= -(0.5)(-1) - (0.5)(-1) = 1$

H (child_1) $= -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= -(1/3)(-1.58) - (2/3)(-0.58) = 0.92$

H (child_2) $= -(1)\log_2(1) = 0$

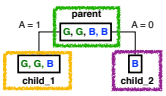
IG = H (parent) – (3/4) H (child_1) – (1/4) H (child_2)
IG = 1 – (3/4) 0.92 – (1/4) 0 = 1 – 0.69 – 0 = **0.31**

Decision Trees - Splitting at nodes

Entropy & Information Gain

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Splitting at A:



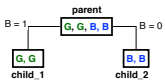
H (parent) $= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$
$= -(0.5)(-1) - (0.5)(-1) = 1$

H (child_1) $= -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= -(1/3)(-1.58) - (2/3)(-0.58) = 0.92$

H (child_2) $= -(1)\log_2(1) = 0$

IG = H (parent) – (3/4) H (child_1) – (1/4) H (child_2)
IG = 1 – (3/4) 0.92 – (1/4) 0 = 1 – 0.69 – 0 = **0.31**

Splitting at B:



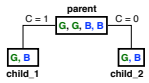
H (parent) $= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$
H (child_1) $= -(2/2)\log_2(2/2) = 0$
H (child_2) $= -(2/2)\log_2(2/2) = 0$
IG = H (parent) – (2/4) H (child_1) – (2/4) H (child_2)
IG = 1 – 0 – 0 = **1**

Decision Trees - Splitting at nodes

Entropy & Information Gain

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Splitting at C:



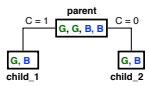
H (parent) $= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$
H (child_1) $= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
H (child_2) $= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
IG = H (parent) – (3/4) H (child_1) – (1/4) H (child_2)
IG = 1 – (2/4) 1 – (2/4) 1 = 1 – 0.5 – 0.5 = 1 – 1 = **0**

Decision Trees - Splitting at nodes

Entropy & Information Gain

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Splitting at C:



H (parent) $= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$
H (child_1) $= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
H (child_2) $= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
IG = H (parent) – (3/4) H (child_1) – (1/4) H (child_2)
IG = 1 – (2/4) 1 – (2/4) 1 = 1 – 0.5 – 0.5 = 1 – 1 = **0**

	Splitting at		
Features	**A**	**B**	**C**
IG	**0.31**	**1**	**0**
Quality of split	poor	**Best**	worst

This might be to much on mathematics, don't worry, you don't need to do these calculations for split, your algorithm will do this automatically for you! However, its good to know how the things work at the backend :)

Decision Trees - Splitting at nodes

Gini Split / Gini Index

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

$$Gini = 1 - \left(\begin{matrix} Sum\ of\ squared \\ probabilities \\ of\ each\ class \end{matrix} \right)$$

- Favours larger partitions.
- Uses squared proportion of classes.
- **Perfect classification, GI = 0.**
- Evenly distributed would be 1 – (1/n_classes).
- We want a variable split that has a low Gini Index.

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

Decision Trees - Splitting at nodes

Gini Split / Gini Index

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

$$Gini = 1 - \left(\begin{matrix} Sum\ of\ squared \\ probabilities \\ of\ each\ class \end{matrix} \right)$$

- Favours larger partitions.
- Uses squared proportion of classes.
- **Perfect classification, GI = 0.**
- Evenly distributed would be 1 – (1/n_classes).
- We want a variable split that has a low Gini Index.

A	B	C	Target
1	1	1	G
1	1	0	G
0	0	1	B
1	0	0	B

Splitting at A:

A = 1 & T = G => 2/3
A = 1 & T = B => 1/3

GI = 1 - {2/3^2 + 1/3^2}
1 - {0.44 + 0.11}
1 - 0.55 = 0.45
GI = 0.45

Poor

Splitting at B:

B = 1 & T = G => 2/2 = 1
B = 1 & T = B => 0

GI = 1 - 1 - 0 = 0

Best

Splitting at C:

C = 1 & T = G => 1/2
C = 1 & T = B => 1/2

GI = 1 - {1/2^2 + 1/2^2}
1 - {0.25 + 0.25}
1 - 0.50 = 0.50
GI = 0.50

Worst

	Splitting at		
Features	**A**	**B**	**C**
IG	**0.31**	**1**	**0**
Quality of split	poor	**Best**	worst

From Entropy & Information Gain

http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

Decision Trees - Advantages / disadvantages

Advantages:
*A decision tree is a **flowchart-like diagram** that shows the various outcomes from a series of decisions. A primary advantage for using a decision tree is that it is **easy to follow and understand**. They are fast and can **handle both numerical and categorical data** in the large datasets.*

Other than machine learning, decision trees are commonly used as a decision-making tool, for research analysis, or for planning strategy.

Decision Trees - Advantages / disadvantages

Advantages:
*A decision tree is a **flowchart-like diagram** that shows the various outcomes from a series of decisions. A primary advantage for using a decision tree is that it is **easy to follow and understand**. They are fast and can **handle both numerical and categorical data** in the large datasets.*

Disadvantages:
While decision trees are computationally cheap for predictions, training the decision tree can be computationally expensive.

Other than machine learning, decision trees are commonly used as a decision-making tool, for research analysis, or for planning strategy.

Decision Trees - Advantages / disadvantages

Advantages:
*A decision tree is a **flowchart-like diagram** that shows the various outcomes from a series of decisions. A primary advantage for using a decision tree is that it is **easy to follow and understand**. They are fast and can **handle both numerical and categorical data** in the large datasets.*

Disadvantages:
While decision trees are computationally cheap for predictions, training the decision tree can be computationally expensive.

In decision trees, utilizing different training and test subsets of the same dataset leads to have high variance (different splits in the training data can lead to very different trees). Decision trees can be very easy to overfit (Overfitting is creating over-complex trees that do not generalize the data well), and the results are "poor performance" on the new and unseen data. This primary weakness limits their usage in predictive modelling.

Other than machine learning, decision trees are commonly used as a decision-making tool, for research analysis, or for planning strategy.

Decision Trees - Advantages / disadvantages

Advantages:
*A decision tree is a **flowchart-like diagram** that shows the various outcomes from a series of decisions. A primary advantage for using a decision tree is that it is **easy to follow and understand**. They are fast and can **handle both numerical and categorical data** in the large datasets.*

Disadvantages:
While decision trees are computationally cheap for predictions, training the decision tree can be computationally expensive.

In decision trees, utilizing different training and test subsets of the same dataset leads to have high variance (different splits in the training data can lead to very different trees). Decision trees can be very easy to overfit (Overfitting is creating over-complex trees that do not generalize the data well), and the results are "poor performance" on the new and unseen data. This primary weakness limits their usage in predictive modelling.

However, using ensemble methods, we can create a model that utilizes underlying decision trees as a foundation to produce excellent results.

Other than machine learning, decision trees are commonly used as a decision-making tool, for research analysis, or for planning strategy.

Decision Trees - Advantages / disadvantages

Advantages:
*A decision tree is a **flowchart-like diagram** that shows the various outcomes from a series of decisions. A primary advantage for using a decision tree is that it is **easy to follow and understand**. They are fast and can **handle both numerical and categorical data** in the large datasets.*

Disadvantages:
While decision trees are computationally cheap for predictions, training the decision tree can be computationally expensive.

In decision trees, utilizing different training and test subsets of the same dataset leads to have high variance (different splits in the training data can lead to very different trees). Decision trees can be very easy to overfit (Overfitting is creating over-complex trees that do not generalize the data well), and the results are "poor performance" on the new and unseen data. This primary weakness limits their usage in predictive modelling.

However, using ensemble methods, we can create a model that utilizes underlying decision trees as a foundation to produce excellent results.

Excellent, we got good understand of Decision Trees! If you want to learn more, please go through the suggested readings.
Let's move on and discuss **The bootstrap, bagging & Random Forests!**

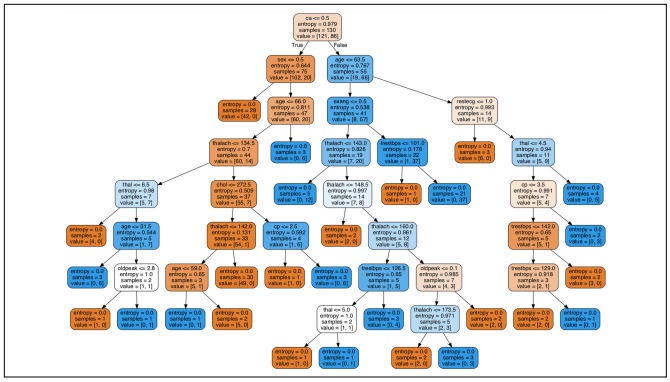
Other than machine learning, decision trees are commonly used as a decision-making tool, for research analysis, or for planning strategy.

Next Lecture:

- ***The Bootstrap***
- ***Bagging (Bootstrap aggregating)***
- ***Random Forests!***

Thank you!

✅*Recommended Readings and References:*
sklearn's Official Documentation - 1.10. Decision Trees
Introduction to Statistical Learning - Chapter 8
Machine Learning - A Probabilistic Perspective - Chapter 16



The Bootstrap

The bootstrap is widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.

The Bootstrap

The bootstrap is widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.

Consider, we have a dataset with 100 values and we want to estimate the mean of the sample

mean = sum of samples or value / no. of samples or values

The Bootstrap

The bootstrap is widely applicable and extremely powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.

Consider, we have a dataset with 100 values and we want to estimate the mean of the sample

mean = sum of samples or value / no. of samples or values

For such a small sample, we expect error in the mean, however, using bootstrap procedure, we can improve the estimate of our mean using the following steps:

- Create many random sub-sets (say 500) of our dataset with replacement — selecting the same value multiple times.
- Calculate mean of each subset.
- Calculate the average of all of our collected means and use that as our estimate mean for the data.

For example, we had 5 subsets which gave the mean values 2.5, 3.5, 5.5, 4.3 & 2.9. The estimated mean in this case will be 3.74

Bagging (Bootstrap aggregating)

Bagging *(an application of the bootstrap)* is a general purpose procedure for reducing the high variance of machine learning algorithm, typically in decision trees. This simple and very powerful ensemble method combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model.

Bagging (Bootstrap aggregating)

Bagging *(an application of the bootstrap)* is a general purpose procedure for reducing the high variance of machine learning algorithm, typically in decision trees. This simple and very powerful ensemble method combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model.

Let's consider that we have a sample with 5,000 instances or values. We want to use decision tree (CART) algorithm. The Bagging procedure will work as follow in this situation:

- Create many random sub-sets (say 500) of our dataset with replacement.
- Train the model on each sub-set.
- For the new dataset, calculate and output the average prediction from each model.

*For example, we had 5 bagged decision trees, making predictions for class **G**, **G**, **B**, **G**, **B**, the most request (mode) **G** will be there final prediction.*

Decision trees are greedy!

They choose the variable (node) to split on using a greedy algorithm that minimizes error. Even with Bagging, the decision trees can have lots of structural similarities and in turn have high correlation in their predictions. Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or at best weakly correlated.

This is where the Random Forests comes in as improvement over bagged decision trees!

Random Forests

Random Forests is one of the most popular and very powerful machine learning algorithm which ***corrects the habit of overfitting of decision trees*** to their training dataset. The essence of the method is to ***assemble multiple trees in randomly selected subsets*** at training time and ***outputting the class that is the mode*** *(set of data values is the value that appears most often)* of the classes *(in case of classification problem)* or mean prediction *(in case of regression problem)* of the individual trees.

The general method of random decision forests was first proposed by Tin Kam Ho in 1995, Here is the original article "Random Decision Forests" An extension of the algorithm was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark.

Random Forests

Random Forests is one of the most popular and very powerful machine learning algorithm which ***corrects the habit of overfitting of decision trees*** to their training dataset. The essence of the method is to ***assemble multiple trees in randomly selected subsets*** at training time and ***outputting the class that is the mode*** *(set of data values is the value that appears most often)* of the classes *(in case of classification problem)* or mean prediction *(in case of regression problem)* of the individual trees.

Suppose, *one* of the *feature* in our dataset is very *strong at predicting a certain class*. With bagging, most of the trees will use that feature as their *top split to minimize the error*. This will *result* in ensemble of *very similar and highly correlated trees*. Averaging highly correlated quantities does not significantly reduce variance and this is what we don't want.

Random Forests

Random Forest does the trick:
By randomly leaving out candidate features from each split, Random Forests "de-correlates" the trees and make them independent from each other. So that, the averaging process can reduce the variance of the resulting model and it has no effect by the features that are strong at predicting a certain class.

Random Forests

Random Forest does the trick:
By randomly leaving out candidate features from each split, Random Forests "de-correlates" the trees and make them independent from each other. So that, the averaging process can reduce the variance of the resulting model and it has no effect by the features that are strong at predicting a certain class.

A new random sample of features is chosen for every single tree at every single split. The number of randomly selected features (m) that can be searched at each split point is specified as a parameter to the algorithm.

Typically if p is the number of full set of the features:
- For classification a good default value for m is = sqrt p *(If we have 25 features, m will be 5 for classification problem)*
- For regression a good default value for m is = p/3

Where m is the number of randomly selected features that can be searched at a split point and p is the number of input variables (full set of features)

Excellent job!
We have covered the theoretical background and key concepts that governs the decision tree learning and random forests.
Let's move on and work with data using Python's Machine Learning library, scikit-learn.

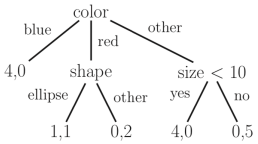


Figure 16.2 A simple decision tree for the data in Figure 1.1. A leaf labeled as (n_1, n_0) means that there are n_1 positive examples that match this path, and n_0 negative examples. In this tree, most of the leaves are "pure", meaning they only have examples of one class or the other; the only exception is leaf representing red ellipses, which has a label distribution of $(1, 1)$. We could distinguish positive from negative red ellipses by adding a further test based on size. However, it is not always desirable to construct trees that perfectly model the training data, due to overfitting.

Page 545, Machine Learning - A probabilistic approach

Pruning a tree

To prevent overfitting, we can stop growing the tree if the decrease in the error is not sufficient to justify the extra complexity of adding an extra subtree. However, this tends to be too myopic. For example, on the xor data in Figure 14.2(c), it would might never make any splits, since each feature on its own has little predictive power.

The standard approach is therefore to grow a "full" tree, and then to perform **pruning**. This can be done using a scheme that prunes the branches giving the least increase in the error. See (Breiman et al. 1984) for details.

To determine how far to prune back, we can evaluate the cross-validated error on each such subtree, and then pick the tree whose CV error is within 1 standard error of the minimum. This is illustrated in Figure 16.4(b). The point with the minimum CV error corresponds to the simple tree in Figure 16.6(a).

Pros and cons of trees

CART models are popular for several reasons: they are easy to interpret[2], they can easily handle mixed discrete and continuous inputs, they are insensitive to monotone transformations of the inputs (because the split points are based on ranking the data points), they perform automatic variable selection, they are relatively robust to outliers, they scale well to large data sets, and they can be modified to handle missing inputs.[3]

However, CART models also have some disadvantages. The primary one is that they do not predict very accurately compared to other kinds of model. This is in part due to the greedy nature of the tree construction algorithm. A related problem is that trees are **unstable**: small changes to the input data can have large effects on the structure of the tree, due to the hierarchical nature of the tree-growing process, causing errors at the top to affect the rest of the tree. In frequentist terminology, we say that trees are high variance estimators. We discuss a solution to this below.

Building a regression tree:

We divide the predictor space — that is, the set of possible value for X1, X2, ... Xp — into J distinct and non-overlapping regions, R1, R2, ... Rj
For every observation that falls into the region Rj, we make the same predictions, which is simply the mean of the response value for the training observation in RJ.

Regression Tree, we look for RSS and for Classification tree we look for classification error rate.

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{8.1}$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box. Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes. For this reason, we take a *top-down, greedy* approach that is known as *recursive binary splitting*. The approach is *top-down* because it begins at the top of the tree (at which point all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree. It is *greedy* because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

recursive binary splitting

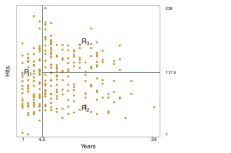


FIGURE 8.2. *The three-region partition for the* Hitters *data set from the regression tree illustrated in Figure 8.1.*

From ISLR (An introduction to statistical learning) Ch#8

Generally, your performance will not change whether you use Gini impurity or Entropy.

Laura Elena Raileanu and Kilian Stoffel compared both in "Theoretical comparison between the gini index and information gain criteria". The most important remarks were:

- It only matters in 2% of the cases whether you use gini impurity or entropy.
- Entropy might be a little slower to compute (because it makes use of the logarithm).

I was once told that both metrics exist because they emerged in different disciplines of science.

If the multiple feature are giving same IG, I believe the first will be selected. The feature will be updated only if the split value is better than the previous. However, in this case, selection is subjective and will not effect. See the links in useful links slide

Gini Index vs Entropy - Information Gain

Gini impurity and Information Gain Entropy are pretty much the same. And people do use the values interchangeably. Below are the formulae of both:

1. $Gini: Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$
2. $Entropy: H(E) = -\sum_{j=1}^{c} p_j \log p_j$

Given a choice, I would use the Gini impurity, as it doesn't require me to compute logarithmic functions, which are computationally intensive. The closed form of it's solution can also be found.

Which metric is better to use in different scenarios while using decision trees ?

The Gini impurity, for reasons stated above.

So, **they are pretty much same when it comes to CART analytics.**

Helpful reference for computational comparison of the two methods

Useful links!

https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/

https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

Gini vs entropy:
https://datascience.stackexchange.com/questions/10228/when-should-i-use-gini-impurity-as-opposed-to-information-gain

Theoretical Comparisons b/w gini index and information gain:
https://www.unine.ch/files/live/sites/imi/files/shared/documents/papers/Gini_index_fulltext.pdf

https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

Same IG:
https://stats.stackexchange.com/questions/23157/choosing-between-best-two-attributes-with-the-same-information-gain-when-buildin/103377
https://stats.stackexchange.com/questions/166560/cart-selection-of-best-predictor-for-splitting-when-gains-in-impurity-decrease/166914#166914

Good on Decision Tree: https://www.saedsayad.com/categorical_categorical.htm

---

The primary weakness of the decision trees is that they don't tend to have the best predictive accuracy, this is partially due to the high variance, meaning that different splits in the training data can lead to very different trees. Bagging is a general purpose procedure for reducing the variance of machine learning method that's discussed in the reading assignment.
We can build off the idea of bagging by utilizing random forest. Its just a slight variation of these bag to trees that has even better performance.
What we do of a random forest as we create ensemble of decision trees using bootstrap samples of the training set.
Bootstrap sample of the training set just means sampling from the training set with replacement. However, we are building each tree each time a split is considered a random sample of M features is chosen as a split candidate from the full set of p features. The split is only allowed to use one of those m features. A new random sample of features is chosen for every single tree every single split.
For classification, this m, random sample of m features is typically chosen to be the square root of p where p is the full set of features.

The general method of random decision forests was first proposed by Tin Kam Ho in 1995, Here is the original article "Random Decision Forests"! An extension of the algorithm was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark.

Random Forests

Wondering, What is the point of using Radom Forest?
Suppose, there is one very strong feature in the data set, a feature that is really strong at prediction a certain class. When using bagged trees, that's that bootstrap sampling, most of the trees will use that feature as the top split. That is going to result in ensemble of really similar trees that are highly correlated, which is something you want to avoid. Averaging highly correlated quantities does not significantly reduce variance.
By randomly leaving out candidate features from each split, Random Forest "de-correlates" the trees (meaning making the trees independent of each other) , such the averaging process can reduce the variance of the resulting model. So, we wont be affected by features that really strongly predict the class data.

The general method of random decision forests was first proposed by Tin Kam Ho in 1995, Here is the original article "Random Decision Forests"! An extension of the algorithm was developed by Leo Breiman and Adele Cutler, and "Random Forests" is their trademark.

Decision Trees - Entropy & Information Gain

Information gain (IG) is used to decide which feature to split on at each step in building the decision tree. Information gain is based on the entropy*

Entropy, H(T) is defined as: $H(T) = -\sum_{i=1}^{J} p_i \log_2 p_i$

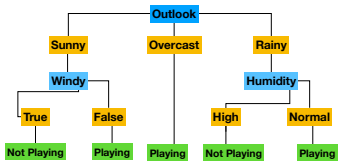
IG = Entropy_before (parent node) - Entropy_after (child node)

At each node, IG for every feature is calculated, the feature with the largest IG value is chosen for the split. This favours features that produce pure splits with low uncertainty/entropy.

This process is applied recursively from the root-node down, and stops when a leaf node contains instances all having the same class (no need to split it further).

* Entropy is a common way to measure the level of impurity in a group, it comes from the information theory - higher the entropy is, more the information contents are.

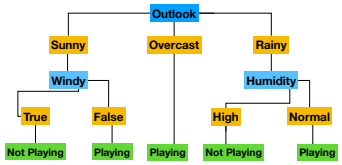
Decision Trees - key concepts



Decision trees:

- **Root Node**: The starting point of the tree, e.g. "Outlook"

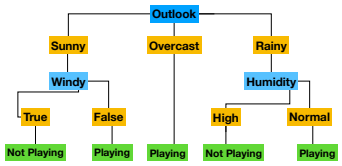
Decision Trees - key concepts



Decision trees:

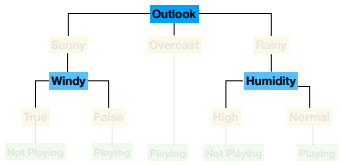
- **Root Node**: The starting point of the tree, e.g. "Outlook"
- **Branches**: Arrows/lines connecting nodes, showing the flow from question to answer - segments of the trees that connect the nodes

Decision Trees - key concepts



Now, another thing that we can notice in the above tree, is **Splitting** at the nodes.

Decision Trees - key concepts



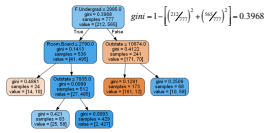
Now, another thing that we can notice in the above tree, is **Splitting** at the nodes.

Decision Tree Learning

In this lecture, we will talk about decision tree learning with some practical examples.

This type of learning uses decision tree as a predictive model to learn from observations about the item, represented in branches, and reach to the conclusions about its target value, represented in leaves.

Dr. Junaid Qazi, *PhD*



$$gini = 1 - \left[\left(\frac{512}{777}\right)^2 + \left(\frac{265}{777}\right)^2\right] = 0.3968$$

A typical decision tree visualization using one of the library in Python