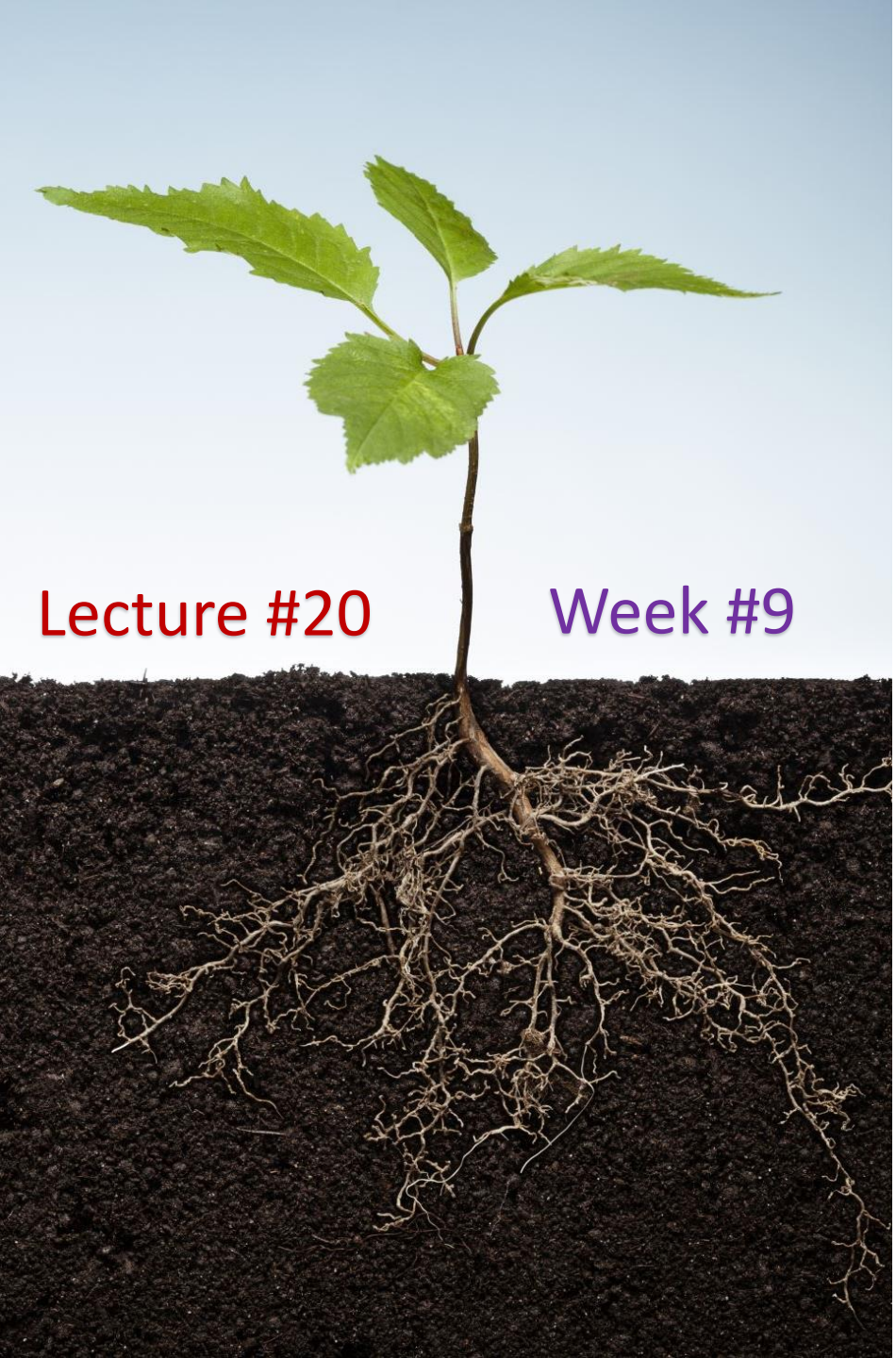


Pattern Recognition

ECSE 4410/6410 CAPA

Spring 2021

Classification and Regression Trees

Course Instructor - Thirimachos Bourlai

January to May 2021

Please Check the 2021 Syllabus

OVERVIEW

Non-linear Algorithms

Classification and Regression Trees

- The many names used to describe the CART algorithm for machine learning
- The representation used by learned CART models that is stored on disk
- How a CART model can be learned from training data
- How a learned CART model can be used to make predictions on unseen data
- How to calculate the Gini index for a given split in a decision tree
- How to evaluate different split points when constructing a decision tree
- How to make predictions on new data with a learned decision tree.

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

About

- Decision Trees → important type of algorithm for predictive modeling ML
- Classical DT algorithms:
  - Have been around for decades and
  - There are modern variations
- Random Forests → among the most powerful techniques available
- We will discuss a DT algorithm known by it's more modern name **CART** or else **Classification And Regression Trees**.

Decision Trees

- **Classification and Regression Trees or CART** → introduced by Leo Breiman
- **CART**: refers to DT algorithms that can be used for classification or regression predictive modeling problems
- In **certain platforms, e.g. R**, they are **referred** to by the more modern term **CART**.
- CART: its an algorithm that provides a **foundation** for other important algorithms:
  - Bagged decision trees
  - Random forests and
  - Boosted decision trees

CART Model Representation

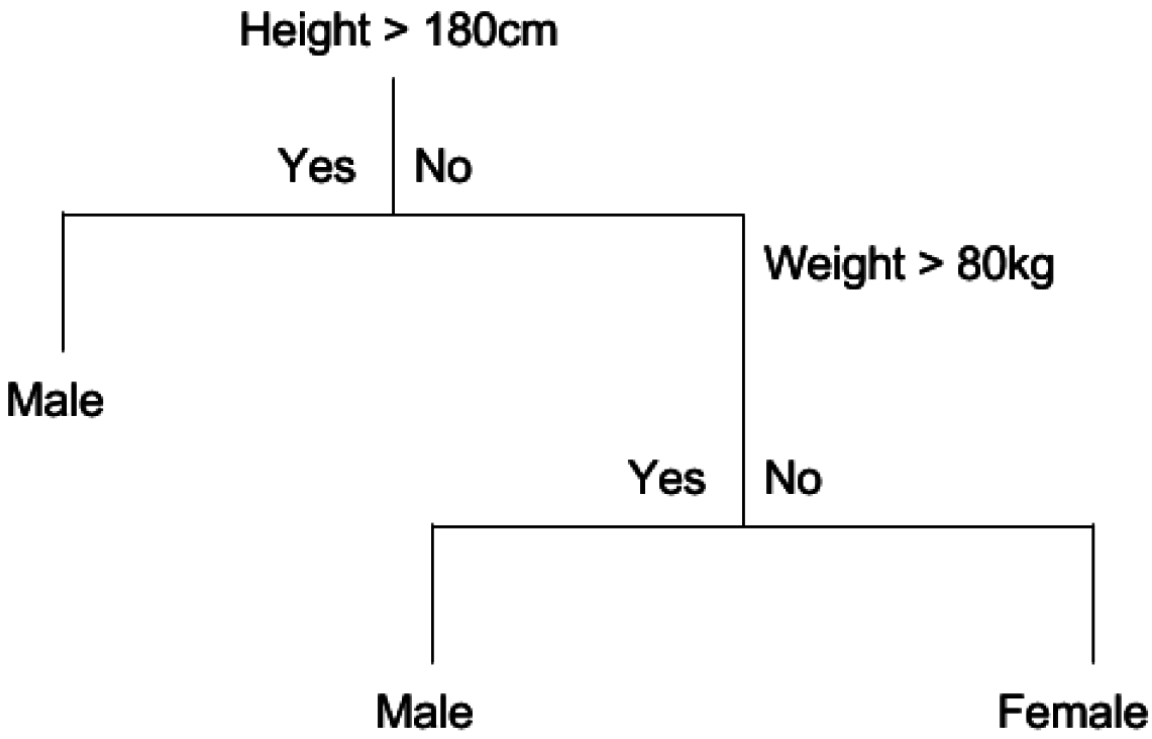
- CART model representation → binary tree
- It's the same binary tree we see in algorithms and data structures
- **Each tree node represents** a single input variable (x) and a split point on a numeric variable
- **Leaf nodes** of the **tree contain an output variable** (y) → used to **make a prediction**

- **DEMO Example**:
  - Two inputs: height (cm) and weight (Kgr)
  - One Output: Gender (M / F)
  - Let's see how a binary decision tree works (non real example – just a demo).

❖ A learned binary tree is a partitioning of the input space. Each input variable is 1D on a p-D space.

❖ The decision tree split the p-D space into rectangles (when p = 2 input variables) or hyper-rectangles when p > 2.

CART Model Representation



The tree can be stored to be as a graph or a set of rules:

If Height >	180 cm	Then Male
If Height <=	180 cm AND Weight > 80 kg	Then Male
If Height <=	180 cm AND Weight <= 80 kg	Then Female

Let's see now how we can "Make Predictions With CART Models"

CART: Making Predictions

Using the previous CART model representation → making predictions is relatively straightforward

PROCESS – Very Simple:

Given a new input / data → we evaluate it at the root node of the tree and filtered through the tree → lands in one of the rectangles → the output value for that rectangle is the prediction made by the model.

Thus, CART models can make boxy decision boundaries.

Example:

Given the input of height = 160 cm and weight = 65 kg, we would traverse the above tree as follows:

- **Is the Height > 180 cm? No**
- **Is the Weight > 80 kg? No**
- **Therefore → Female**

CART Model Representation

- **Creating a binary decision tree**: the input space is divided via a **greedy approach**
  - Namely: **recursive binary** splitting - a numerical procedure where all the values are lined up
  - *Different split points are tried and tested* using a cost function
  - The split with the lowest cost is selected
- **All input variables** and **all possible split points** are evaluated and chosen in a greedy manner (e.g., the very best split point is chosen each time)

CART Model Representation

- For regression predictive modeling problems, the cost function that is minimized, to choose split points, is the **sum squared error across all training samples** that fall within the rectangle:

$$\sum_{i=1}^{n}(y_i - prediction_i)^2$$

, where

- ❖ Y…….. = output for the training sample
- ❖ prediction = predicted output for the rectangle

CART Model Representation

- For **classification**, the **Gini index function** is used, which provides an indication of how **pure the leaf nodes are** (how mixed the training data assigned to each node is).

$$G = \sum_{k=1}^{n} p_k \times (1 - p_k)$$

- G= Gini index over all classes
- pk =the proportion of training instances with class k in the rectangle of interest.

- G=0 → A node that has all classes of the same type (perfect class purity)
- G = 0.5 → a G that has a 50-50 split of classes for a binary classification problem (worst purity)

- For a binary classification problem, this can be re-written as:

$$G = 2 \times p_1 \times p_2$$
$$G = 1 - ({p_1}^2 + {p_2}^2)$$

Learn a CART Model From Data

- The **Gini index calculation** for each node is weighted by the total number of instances in the parent node.
- The **Gini score** for a chosen split point in a **binary classification** problem is therefore calculated as follows:

$$G = ((1 - (g1_1{}^2 + g1_2{}^2)) \times \frac{n_{g1}}{n}) + ((1 - (g2_1{}^2 + g2_2{}^2)) \times \frac{n_{g2}}{n})$$

- G = Gini index for the split point
- $g1_1$ = proportion of instances in group 1 for class 1
- $g1_2$ = proportion of instances in group 1 for class 2
- $g2_1$ = proportion of instances in group 2 for class 1
- $g2_2$ = proportion of instances in group 2 for class 2 a
- ng1 and ng2 = the total # of instances in group 1 and 2
- n = total # of instances we need to group from the parent node.

CART #1: Stopping Criterion

Recursive binary splitting procedure --- It needs to know when to stop splitting *as it works its way down the tree with the training data*

- **Most common stopping procedure**: use a minimum count on the # of training instances assigned to each leaf node
  - If the count < pre-assigned minimum → the split is not accepted, and the node is taken as a final leaf node.
  - Else – it continues
- **Count of training members:** tuned to the dataset, e.g., 5 or 10
  - It defines how specific to the training data the tree will be
  - **Too specific (e.g., a count of 1) → tree will overt the training data** and likely have **poor performance** on the **test s**et.

CART #2: Pruning the Tree

The stopping criterion is important as it strongly influences the performance of your tree.

- You can use **pruning** *after learning your tree* to further lift performance.

- **Decision tree complexity =** the # of splits in the tree

- **Simpler trees are preferred**
  - Easy to understand (you can print them out and show them to SMEs)
  - They are less likely to overfit your data

CART #2: Pruning the Tree

The fastest and simplest pruning method

- Work through each leaf node in the tree
- Evaluate the effect of removing it using a hold-out test set

- **Remove Leaf Nodes**
  - Only if it results in a drop in the overall cost function on the entire test set
- **Stop removing nodes**
  - When no further improvements can be made

More sophisticated pruning methods can be used such as **cost complexity pruning** (also called **weakest link pruning - WLP**)

- **WLP**: where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree

CART: Preparing the Data

- CART does not require any special data preparation other than a good representation of the problem.

CART Model Representation

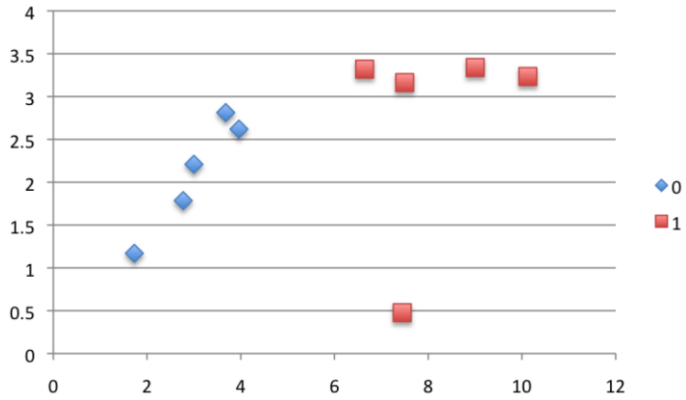
- Decision Tree → Modern name CART
- The representation used for CART is a binary tree
- Predictions are made with CART by traversing the binary tree given a new input record
- The **tree is learned** using a **greedy algorithm** on the training data to pick splits in the tree
- **Stopping criteria** dene how much a tree learns
- **Tree Pruning**
  - Can improve generalization on a learned tree

Simple EXAMPLE

X1	X2	Y
2.771244718	1.784783929	0
1.728571309	1.169761413	0
3.678319846	2.81281357	0
3.961043357	2.61995032	0
2.999208922	2.209014212	0
7.497545867	3.162953546	1
9.00220326	3.339047188	1
7.444542326	0.476683375	1
10.12493903	3.234550982	1
6.642287351	3.319983761	1

The CART model is learned by looking for split points in the data

- **Split point**: a single value of a single attribute, e.g. the 1st value of the X1 attribute = 2.77
- Partitioning data at a split point → separating all data at that node into 2 groups, left and right of the split point

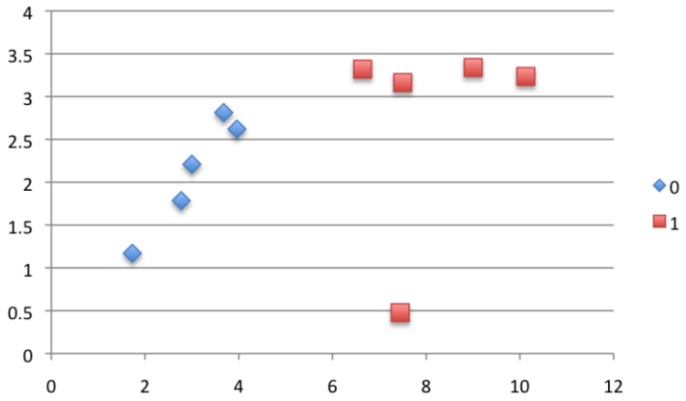


- If we work on the 1st split point → all data of the DS is affected
- If we are working on at level deep → then only the data that has **filtered down the tree** from nodes above and is sitting at that node is affected by the split point

Simple EXAMPLE

- We do not care about the class value of the chosen split point BUT the composition of the data assigned to the LEFT and to the RIGHT child nodes of the split point
- A cost function is used to evaluate the mix of classes of training data assigned to each side of the split
- In classification problems the Gini index cost function is used.

X1	X2	Y
2.771244718	1.784783929	0
1.728571309	1.169761413	0
3.678319846	2.81281357	0
3.961043357	2.61995032	0
2.999208922	2.209014212	0
7.497545867	3.162953546	1
9.00220326	3.339047188	1
7.444542326	0.476683375	1
10.12493903	3.234550982	1
6.642287351	3.319983761	1



Simple EXAMPLE: Gini Index Cost Function

$$G = 1 - ({p_1}^2 + {p_2}^2)$$

- p_1 is the proportion of instances in the node with class 1
- p_2 is the proportion of instances in the node for class 2

- We have 2 groups: a LEFT and RIGHT group because we are using a binary tree

- We know from our dataset that we only have two classes.

Simple EXAMPLE : Gini Index Cost Function

- The proportion of data instances of a class is easy to calculate.
- If a LEFT group has 3 instances with class 0 and 4 instances with class 1 → the proportion of data instances with class 0 would be 3/7=0.4285
- To get a feeling for Gini index scores for child nodes → see table below (GINI Calculations)
- It provides 7 different scenarios for mixes of 0 and 1 classes in a single group.

Class 0	Class 1	Count	Class 0/Count	Class 1/Count	Gini
10	10	20	0.5	0.5	0.5
19	1	20	0.95	0.05	0.095
1	19	20	0.05	0.95	0.095
15	5	20	0.75	0.25	0.375
5	15	20	0.25	0.75	0.375
11	9	20	0.55	0.45	0.495
20	0	20	1	0	0

- When the group has a 50-50 mix (1st row) → **Gini=0.5 → worst possible split**
- When the group only has data instances with class 0 (see last row) → **Gini= 0 → perfect split**.

Simple EXAMPLE

For a chosen split point → **calculate** the Gini index **for each child node** and **weighting** the **scores** by the number of instances in the parent node, as follows:

$$G = ((1 - ({g1_1}^2 + {g1_2}^2)) \times \frac{n_{g1}}{n}) + ((1 - ({g2_1}^2 + {g2_2}^2)) \times \frac{n_{g2}}{n})$$

- G = Gini index for the split point
- $g1_1 / g1_2$ = the proportion of instances in group 1 for classes 1 and 2 respectfully
- $g2_1 / g2_2$ = the proportion of instances in group 2 for classes 1 and 2 respectfully
- n = total # of instances we are trying to group from the parent node

GOAL in selecting a split point:

- Evaluate the Gini index (of all possible split points)
- Select the split point with the lowest cost

Simple EXAMPLE: 1st Candidate Split Point

- **STEP 1**: choose a split → it becomes the root node of our decision tree
- **STEP 2**: start with the 1st candidate split point, i.e X1 = 2,7712
- **STEP 3**:
  - **IF** $X1 < 2.7712$ **THEN** LEFT
  - **IF** $X1 \geq 2.7712$ **THEN** RIGHT
- **STEP 4**: Apply Step 3 to all data

X1	X2	Y
2.771244718	1.784783929	0
1.728571309	1.169761413	0
3.678319846	2.81281357	0
3.961043357	2.61995032	0
2.999208922	2.209014212	0
7.497545867	3.162953546	1
9.00220326	3.339047188	1
7.444542326	0.476683375	1
10.12493903	3.234550982	1
6.642287351	3.319983761	1

X1	Y	Group
2.771244718	0	RIGHT
1.728571309	0	LEFT
3.678319846	0	RIGHT
3.961043357	0	RIGHT
2.999208922	0	RIGHT
7.497545867	1	RIGHT
9.00220326	1	RIGHT
7.444542326	1	RIGHT
10.12493903	1	RIGHT
6.642287351	1	RIGHT

- **STEP 5: Assess the split --** We can evaluate the mixture of the classes in each of the LEFT and RIGHT nodes as a single cost of choosing this split point for our root node
  - LEFT group: 1 member
  - RIGHT group: 9 members

Source: ML-Mastery – ML Algorithms 2020 by J. Brownlee

Simple EXAMPLE: 1st Candidate Split Point

STEP 5 / Cont:

- **LEFT group**, we can calculate the proportion of training instances that have each class

- $Y = 0$: $\frac{1}{1}$ or 1.0
- $Y = 1$: $\frac{0}{1}$ or 0.0

- Right Group:

- $Y = 0$: $\frac{4}{9}$ or 0.4444
- $Y = 1$: $\frac{5}{9}$ or 0.5555

X1	Y	Group
2.771244718	0	RIGHT
1.728571309	0	LEFT
3.678319846	0	RIGHT
3.961043357	0	RIGHT
2.999208922	0	RIGHT
7.497545867	1	RIGHT
9.00220326	1	RIGHT
7.444542326	1	RIGHT
10.12493903	1	RIGHT
6.642287351	1	RIGHT

Step 6: Calculate the Gini index for this split

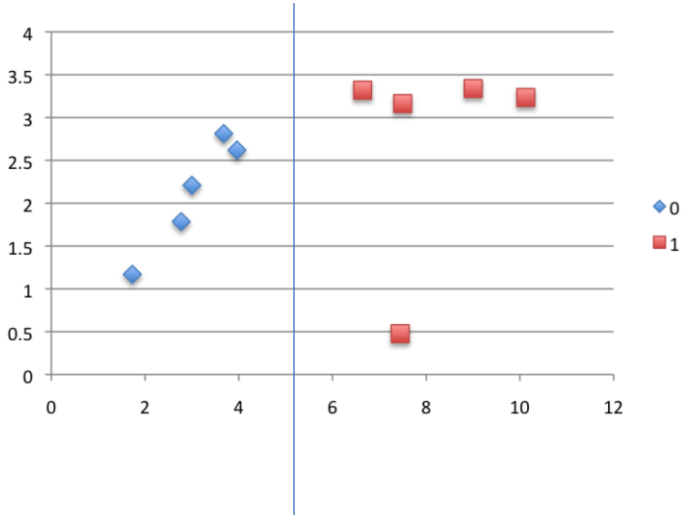
$$Gini(X1 = 2.7712) = ((1 - (\frac{1}{1}^2 + \frac{0}{1}^2)) \times \frac{1}{10}) + ((1 - (\frac{4}{9}^2 + \frac{5}{9}^2)) \times \frac{9}{10})$$

or

$$Gini(X1 = 2.7712) = 0.4444$$

Simple EXAMPLE: Best Candidate Split Point

- We can evaluate each candidate split point using the process above with the values from X1 and X2
- If we look at the graph of the data, we can see that we can probably draw a vertical line to separate the classes
- This would translate to a **split point** for **X1 ~ 5.0**
- **X1 = 6.6422** is a close value



- **IF** $X1 < 6.6422$ **THEN** LEFT
- **IF** $X1 \geq 6.6422$ **THEN** RIGHT

Apply this rule to all instances (Left/Right Groups):

X1	Y	Group
2.771244718	0	LEFT
1.728571309	0	LEFT
3.678319846	0	LEFT
3.961043357	0	LEFT
2.999208922	0	LEFT
7.497545867	1	RIGHT
9.00220326	1	RIGHT
7.444542326	1	RIGHT
10.12493903	1	RIGHT
6.642287351	1	RIGHT

- $Y = 0$: $\frac{5}{5}$ or 1.0
- $Y = 1$: $\frac{0}{5}$ or 0.0

L

- $Y = 0$: $\frac{0}{5}$ or 0.0
- $Y = 1$: $\frac{5}{5}$ or 1.0

R

$$Gini(X1 = 6.6422) = ((1 - (\frac{5^2}{5} + \frac{0^2}{5})) \times \frac{5}{10}) + ((1 - (\frac{0^2}{5} + \frac{5^2}{5})) \times \frac{5}{10})$$

$$Gini(X1 = 6.6422) = 0.0$$

classes are perfectly separated.

Source: ML-Mastery – ML Algorithms 2020 by J. Brownlee

Make Predictions – Test Data

X1	X2	Y
2.343875381	2.051757824	0
3.536904049	3.032932531	0
2.801395588	2.786327755	0
3.656342926	2.581460765	0
2.853194386	1.052331062	0
8.907647835	3.730540859	1
9.752464513	3.740754624	1
8.016361622	3.013408249	1
6.58490395	2.436333477	1
7.142525173	3.650120799	1

Using the decision tree with a single split at, X1 = 6.6422 we can classify the test instances as follows ----------→

Y	Prediction
0	0
0	0
0	0
0	0
0	0
1	1
1	1
1	1
1	0
1	1

Near perfect classification accuracy of 90% accurate.

NEW - CART EXAMPLE

- "Decision tree algorithms still keep their popularity because they can produce transparent decisions.
  - ID3 Decision Trees (uses information gain)
  - C4.5 uses gain ratio for splitting

- CART is an alternative decision tree building algorithm: It can handle both classification and regression tasks.

- This algorithm uses a metric named GINI index to create decision points for classification tasks.

- Here is a step-by-step CART decision tree example by hand from scratch."

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

"You can find the *python implementation* of CART algorithm here.

You can build CART decision trees with a few lines of code.

The package in the link below supports the most common decision tree algorithms such as ID3, C4.5, CHAID or Regression Trees

It also supports bagging methods such as random forest and some boosting methods such as gradient boosting and adaboost."

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Objective

Decision rules will be found by GINI index value.

Data set

We will work on same dataset in ID3. There are 14 instances of golf playing decisions based on outlook, temperature, humidity and wind factors.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini index

Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

$$Gini = 1 - \Sigma (Pi)^2 \text{ for i=1 to number of classes}$$

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. I will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

$Gini(Outlook=Sunny) = 1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$

$Gini(Outlook=Overcast) = 1 - (4/4)^2 - (0/4)^2 = 0$

$Gini(Outlook=Rain) = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$

Then, we will calculate weighted sum of gini indexes for outlook feature.

$Gini(Outlook) = (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 = 0.171 + 0 + 0.171 = 0.342$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Temperature

Similarly, temperature is a nominal feature and it could have 3 different values: Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

$Gini(Temp=Hot) = 1 - (2/4)^2 - (2/4)^2 = 0.5$

$Gini(Temp=Cool) = 1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$

$Gini(Temp=Mild) = 1 - (4/6)^2 - (2/6)^2 = 1 - 0.444 - 0.111 = 0.445$

We'll calculate weighted sum of gini index for temperature feature

$Gini(Temp) = (4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 = 0.142 + 0.107 + 0.190 = 0.439$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = $1 - (3/7)^2 - (4/7)^2 = 1 - 0.183 - 0.326 = 0.489$

Gini(Humidity=Normal) = $1 - (6/7)^2 - (1/7)^2 = 1 - 0.734 - 0.02 = 0.244$

Weighted sum for humidity feature will be calculated next

Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

$Gini(Wind=Weak) = 1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.062 = 0.375$

$Gini(Wind=Strong) = 1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$

$Gini(Wind) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

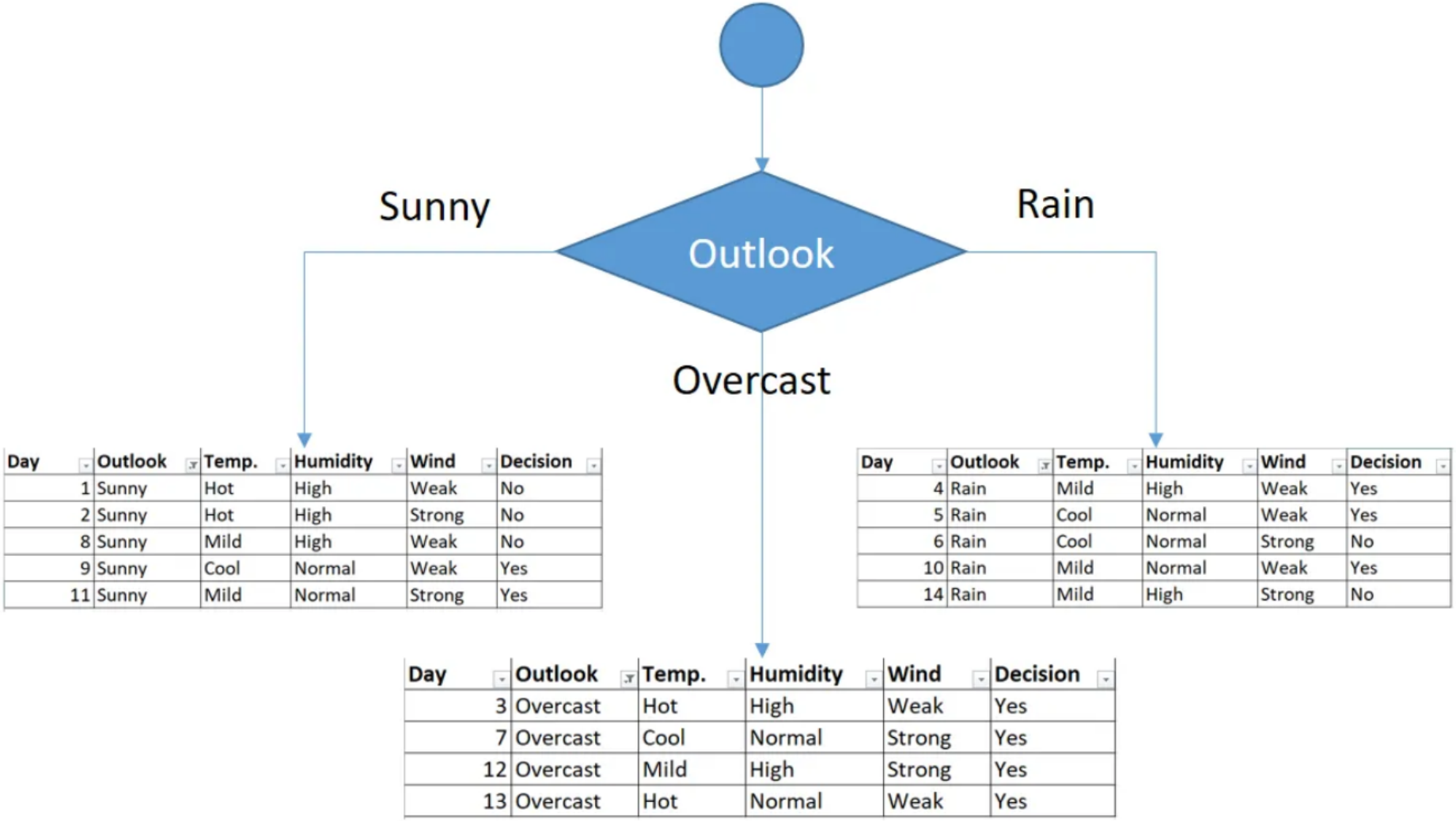
Time to decide

We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

We'll put outlook decision at the top of the tree.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

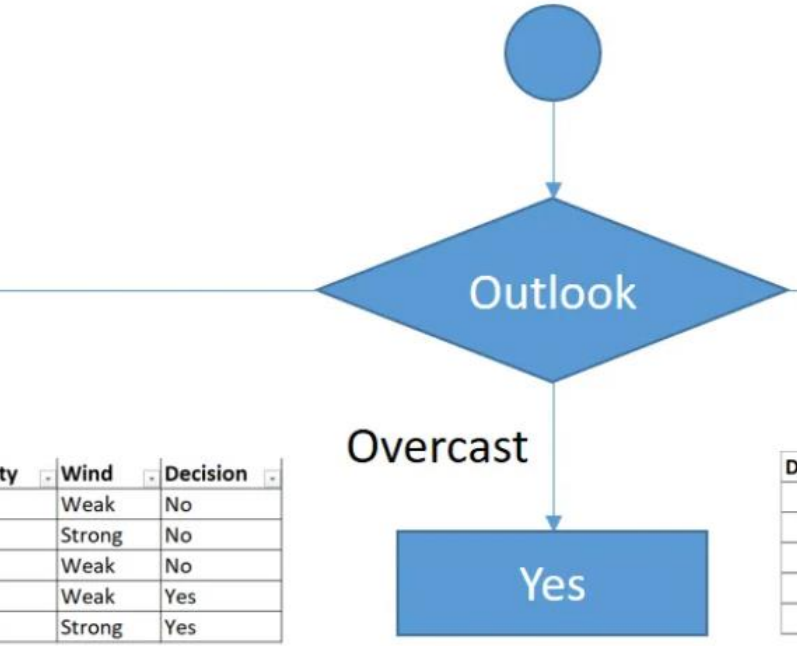
Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

First decision would be outlook feature

You might realize that sub dataset in the overcast leaf has only yes decisions. This means that overcast leaf is over.

CART EXAMPLE



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Tree is over for overcast outlook leaf

We will apply same principles to those sub datasets in the following steps.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Focus on the sub dataset for sunny outlook. We need to find the gini index scores for temperature, humidity and wind features respectively.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

$$Gini(Outlook=Sunny \text{ and } Temp.=Hot) = 1 - (0/2)^2 - (2/2)^2 = 0$$

$$Gini(Outlook=Sunny \text{ and } Temp.=Cool) = 1 - (1/1)^2 - (0/1)^2 = 0$$

$$Gini(Outlook=Sunny \text{ and } Temp.=Mild) = 1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$$

$$Gini(Outlook=Sunny \text{ and } Temp.) = (2/5)x0 + (1/5)x0 + (2/5)x0.5 = 0.2$$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

$$Gini(Outlook=Sunny \text{ and } Humidity=High) = 1 - (0/3)^2 - (3/3)^2 = 0$$

$$Gini(Outlook=Sunny \text{ and } Humidity=Normal) = 1 - (2/2)^2 - (0/2)^2 = 0$$

$$Gini(Outlook=Sunny \text{ and } Humidity) = (3/5)x0 + (2/5)x0 = 0$$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak) = $1 - (1/3)^2 - (2/3)^2 = 0.266$

Gini(Outlook=Sunny and Wind=Strong) = $1- (1/2)^2 - (1/2)^2 = 0.2$

Gini(Outlook=Sunny and Wind) = $(3/5)x0.266 + (2/5)x0.2 = 0.466$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Decision for sunny outlook

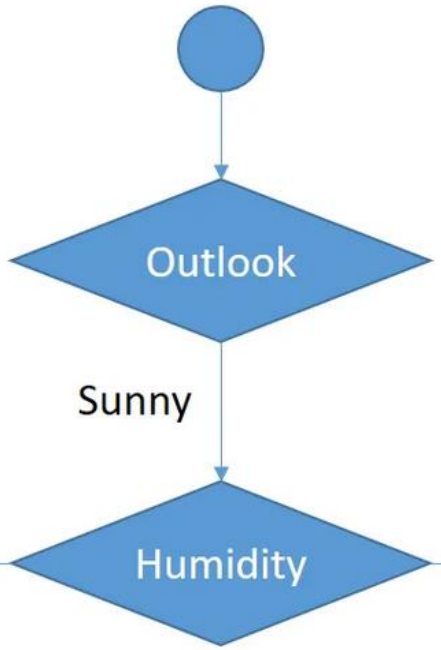
We've calculated gini index scores for feature when outlook is sunny. The winner is humidity because it has the lowest value.

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

We'll put humidity check at the extension of sunny outlook.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

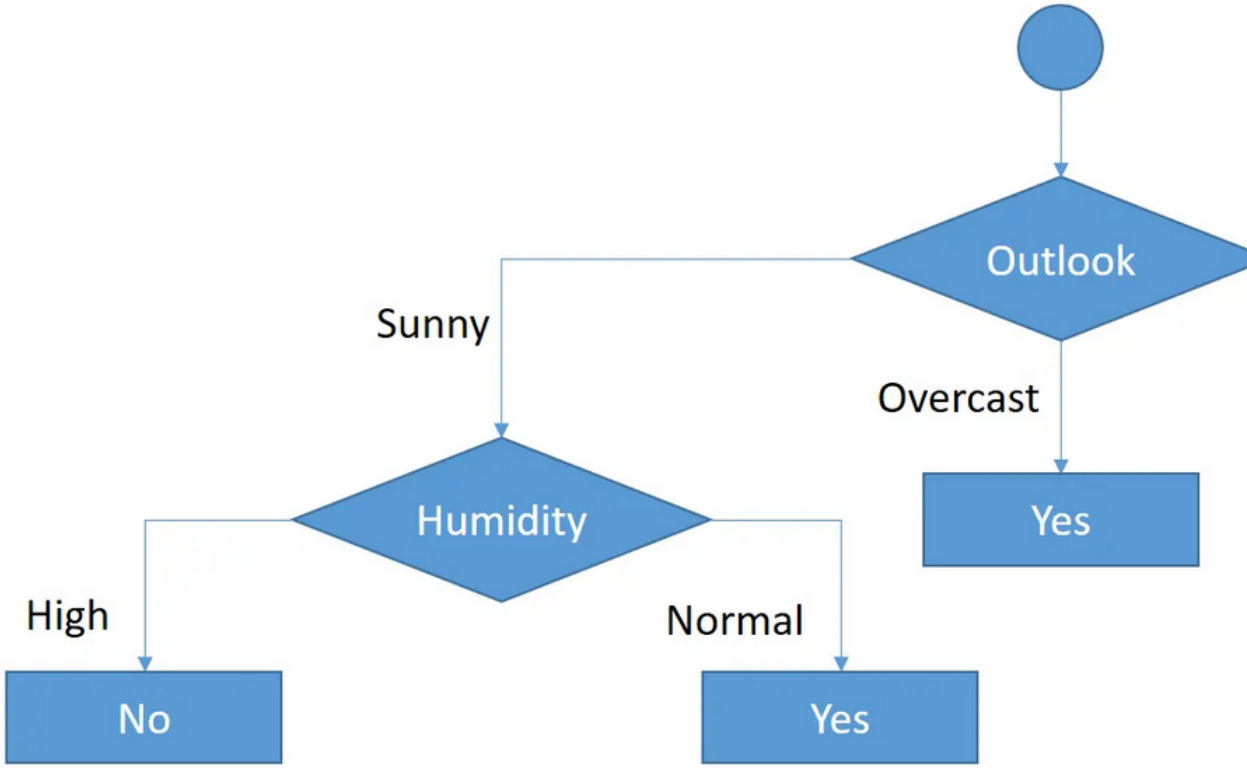
Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Sub datasets for high and normal humidity

As seen, decision is always no for high humidity and sunny outlook. On the other hand, decision will always be yes for normal humidity and sunny outlook. This branch is over.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decisions for high and normal humidity

Now, we need to focus on rain outlook.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Rain outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We'll calculate gini index scores for temperature, humidity and wind features when outlook is rain.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini of temprature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

$Gini(Outlook=Rain\ and\ Temp.=Cool) = 1 - (1/2)^2 - (1/2)^2 = 0.5$

$Gini(Outlook=Rain\ and\ Temp.=Mild) = 1 - (2/3)^2 - (1/3)^2 = 0.444$

$Gini(Outlook=Rain\ and\ Temp.) = (2/5)x0.5 + (3/5)x0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

$Gini(Outlook=Rain\ and\ Humidity=High) = 1 - (1/2)^2 - (1/2)^2 = 0.5$

$Gini(Outlook=Rain\ and\ Humidity=Normal) = 1 - (2/3)^2 - (1/3)^2 = 0.444$

$Gini(Outlook=Rain\ and\ Humidity) = (2/5)x0.5 + (3/5)x0.444 = 0.466$

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak) = $1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong) = $1 - (0/2)^2 - (2/2)^2 = 0$

Gini(Outlook=Rain and Wind) = $(3/5)x0 + (2/5)x0 = 0$

Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Put the wind feature for rain outlook branch and monitor the new sub data sets.

CART EXAMPLE

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

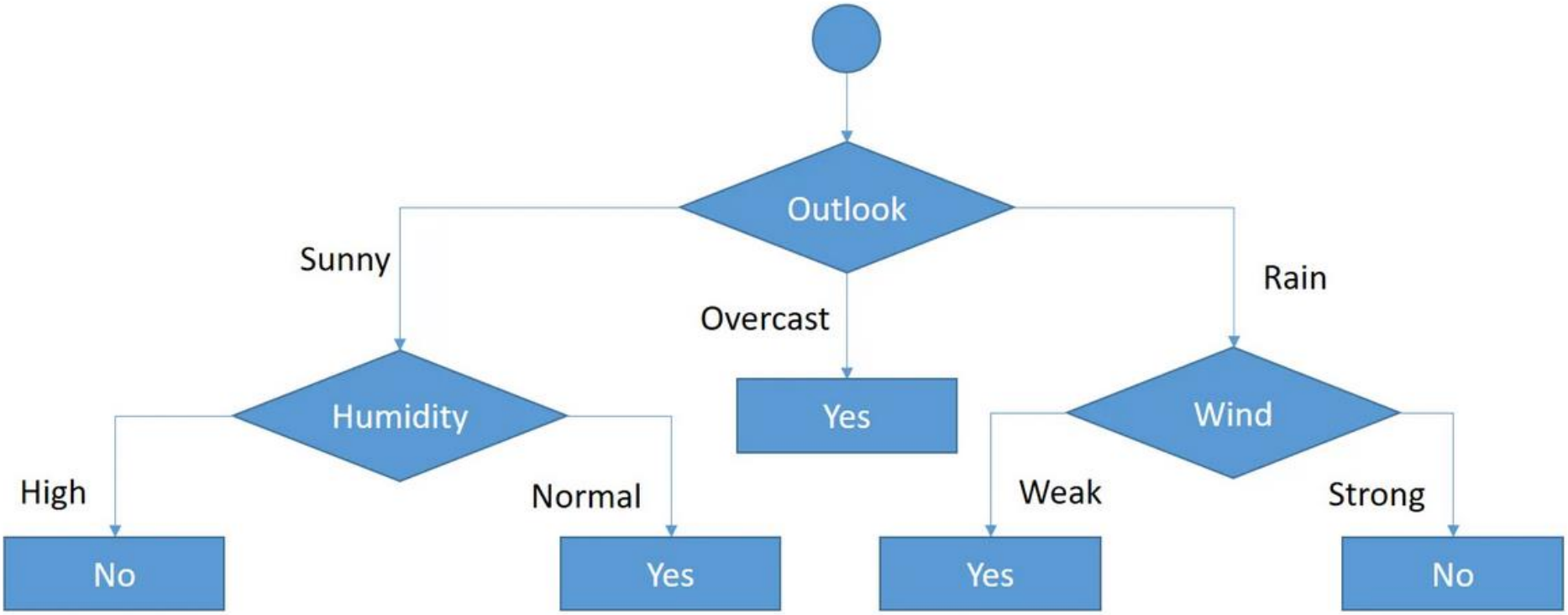
Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

Sub data sets for weak and strong wind and rain outlook

As seen, decision is always yes when wind is weak. On the other hand, decision is always no if wind is strong. This means that this branch is over.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE



Final form of the decision tree built by CART algorithm

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Feature Importance

- Decision trees are naturally explainable and interpretable algorithms
- Besides, we can find the feature importance values as well to understand how model works

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

CART EXAMPLE

Conclusion

We have built a decision tree by hand.

We can create the same tree using the ID3 example

https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/

This does not mean that ID3 and CART algorithms produce same trees always.

Reports state that: CART is easier than ID3 and C4.5

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

ID3 EXAMPLE

https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/

We can summarize the ID3 algorithm as illustrated below

$$Entropy(S) = \sum - p(I) \cdot \log_2 p(I)$$

$$Gain(S, A) = Entropy(S) - \sum [\, p(S|A) \cdot Entropy(S|A) \,]$$

Entropy

We need to calculate the entropy first. Decision column consists of 14 instances and includes two labels: yes and no. There are 9 decisions labeled yes, and 5 decisions labeled no.

$$Entropy(Decision) = - p(Yes) \cdot \log_2 p(Yes) - p(No) \cdot \log_2 p(No)$$

$$Entropy(Decision) = - (9/14) \cdot \log_2 (9/14) - (5/14) \cdot \log_2 (5/14) = 0.940$$

Wind factor on decision

$$Gain(Decision, Wind) = Entropy(Decision) - \sum [\, p(Decision|Wind) \cdot Entropy(Decision|Wind) \,]$$

Wind attribute has two labels: weak and strong. We would reflect it to the formula.

$$Gain(Decision, Wind) = Entropy(Decision) - [\, p(Decision|Wind=Weak) \cdot Entropy(Decision|Wind=Weak) \,] - [\, p(Decision|Wind=Strong) \cdot Entropy(Decision|Wind=Strong) \,]$$

Now, we need to calculate (Decision|Wind=Weak) and (Decision|Wind=Strong) respectively.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

ID3 EXAMPLE

Weak wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

There are 8 instances for weak wind. Decision of 2 items are no and 6 items are yes as illustrated below.

There are 8 instances for weak wind. Decision of 2 items are no and 6 items are yes as illustrated below.

1- Entropy(Decision|Wind=Weak) = – p(No) . $\log_2$p(No) – p(Yes) $\log_2$p(Yes)

2- Entropy(Decision|Wind=Weak) = – (2/8) . $\log_2$(2/8) – (6/8) . $\log_2$(6/8) = 0.811

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

ID3 EXAMPLE

Strong wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

Here, there are 6 instances for strong wind. Decision is divided into two equal parts.

1- Entropy(Decision|Wind=Strong) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes)

2- Entropy(Decision|Wind=Strong) = – (3/6) . $\log_2$(3/6) – (3/6) . $\log_2$(3/6) = 1

Now, we can turn back to Gain(Decision, Wind) equation.

Gain(Decision, Wind) = Entropy(Decision) – [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] – [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)] = 0.940 – [(8/14) . 0.811] – [(6/14). 1] = 0.048

Calculations for wind column is over. Now, we need to apply same calculations for other columns to find the most dominant factor on decision.

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

ID3 EXAMPLE

Other factors on decision

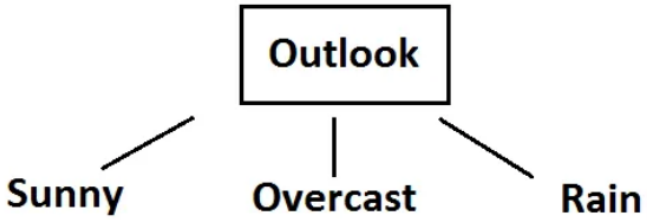
We have applied similar calculation on the other columns.

1- Gain(Decision, Outlook) = 0.246

2- Gain(Decision, Temperature) = 0.029

3- Gain(Decision, Humidity) = 0.151

As seen, outlook factor on decision produces the highest score.
That's why, outlook decision will appear in the root node of the tree.



Root decision on the tree

https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/

ID3 EXAMPLE – mode details

More on the link below – home study

https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/

More Complicated EXAMPLE - Matlab

https://www.mathworks.com/help/stats/fitrtree.html?docviewer=null

What have we learned

- The model representation for LDA and what is distinct about a learned model
- How the parameters of the LDA model can be estimated from training data
- How the model can be used to make predictions on new data
- How to prepare your data to get the most from the method
- How to calculate the statistics from your dataset required by the LDA model
- How to use the LDA model to calculate a discriminant value for each class and make a prediction.
- CARD and ID3 DT examples

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Questions?

THANK YOU!