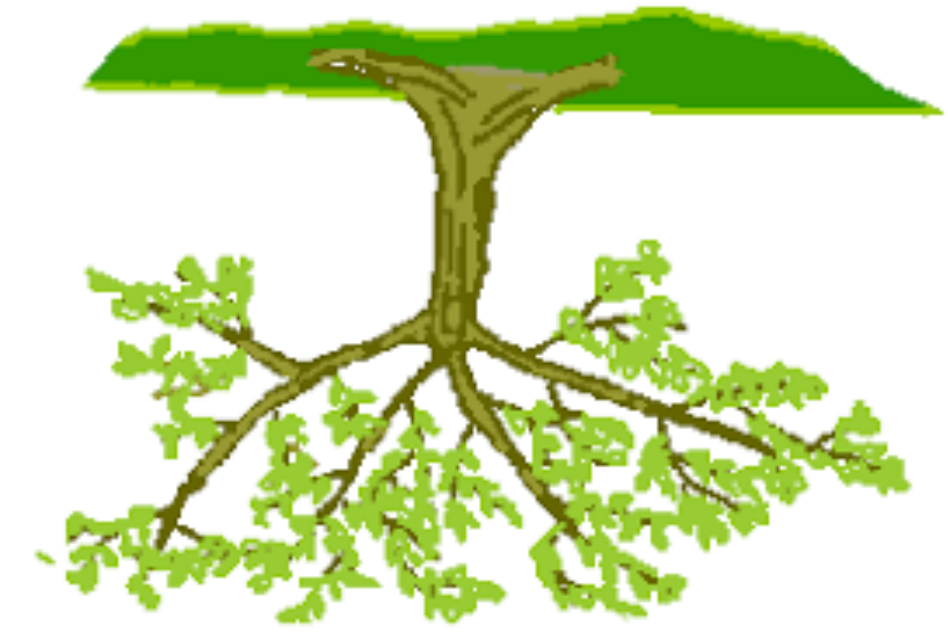
Introduction to Machine Learning Using python: Decision Tree



Objective

- The aim of this lecture is to learn how one of the famous algorithm for constructing decision trees, work.
- At the end of this lecture you should understand how to construct a decision tree using the concept of Information gain.
- Moreover, you will be able to use the decision tree you created to make a decision about new data.

Introduction

- Tree based learning algorithms are considered to be one of the best and most popular supervised learning methods.
- They are popular because the final model is easy to understand.
- Tree based algorithms methods empower predictive models with high accuracy, stability and ease of interpretation.
- They map non-linear relationship quite well.
- These are used in all kinds of data science problems.
- Decision tree is a tree in which each branch node represents a choice between a number of alternatives and each leaf node represents a decision.

- Decision trees are supervised learning algorithms used for both classification and regression tasks.

- The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).

- The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class.

- Decision trees are assigned to the information based learning algorithms which use different measures of information gain for learning.

- We can use decision trees for issues where we have continuous but also categorical input and target features.

- The main idea behind decision trees is to find those variables which contain the most information regarding target feature and then split the dataset along the values of these features such that the target feature values for the resulting sub datasets are as pure as possible

- This process of finding the most informative feature is done until we accomplish a stopping criteria where we finally end up in so called leaf nodes.

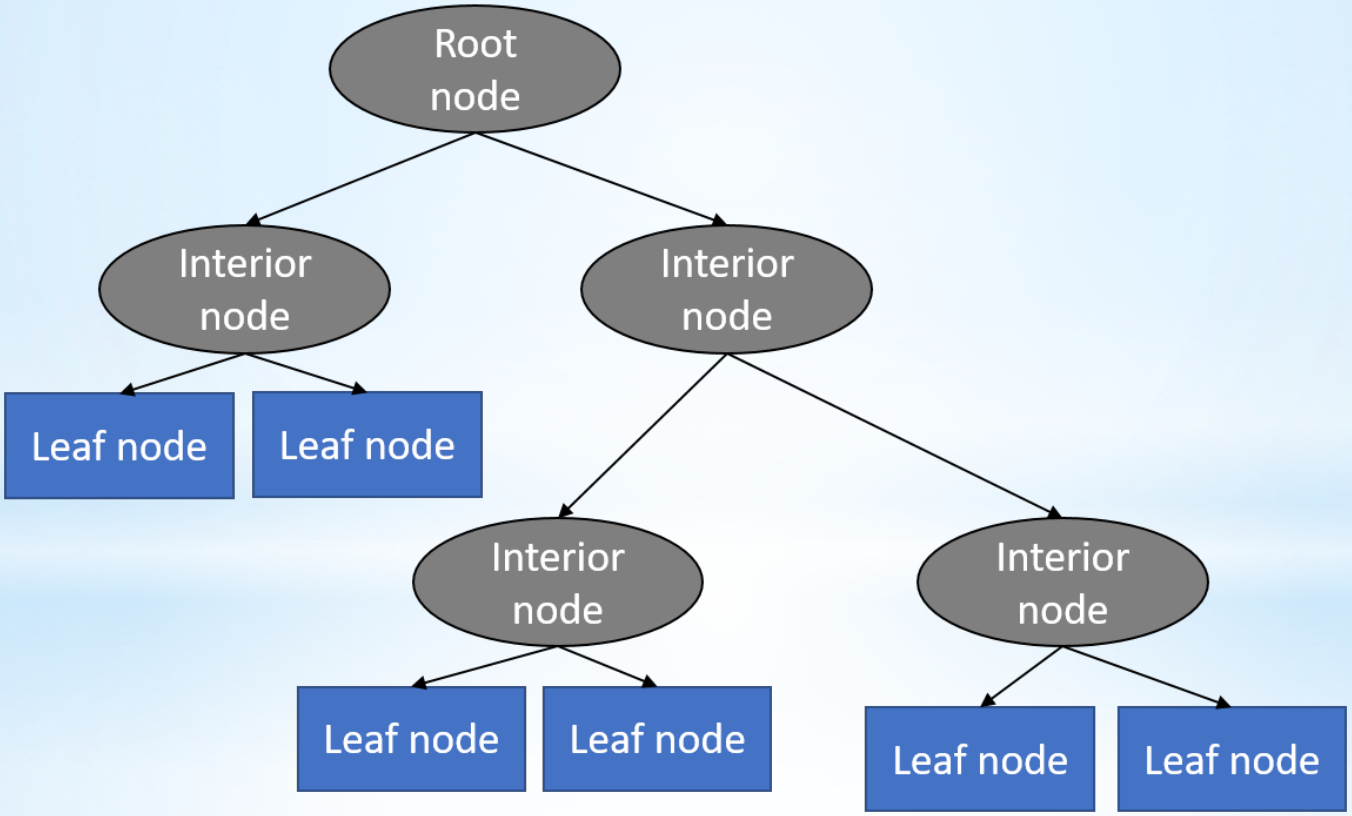
- The leaf nodes contain the predictions we will make for new query instances

- We split the data into two or more homogeneous sets based on most significant splitter in input variables.

- A decision tree leads to a prediction by asking a series of questions on whether you belong to certain groups

- You start at the top question, called the root node, then move through the tree branches according to which groups you belong to, until you reach a leaf node.

A decision tree mainly contains of a root node, interior node, and a leaf node which are then connected by branches.



- Decision tree used in data mining are two main types:
    - **Classification trees** – when the predicted outcome is the class to which the data belongs
    - **Regression trees** – is when the predicted outcome can be considered a real number
- The term classification and regression tree (CART) is an umbrella term used to refer to both of the above procedures.

Algorithm for decision tree

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

Building a decision tree

A decision tree consists of nodes and arcs which connects nodes.

To make a decision tree, one starts at the root node, and asks questions to determine which arc to follow, until one reaches a leaf node and the decision is made.

Each non-leaf of a decision tree corresponds to an input attribute, and each arc to a possible value of that attribute. A leaf node corresponds to the expected value of the output attributes when the input attributes are described by the path from the root node to that leaf node.

Building a decision tree

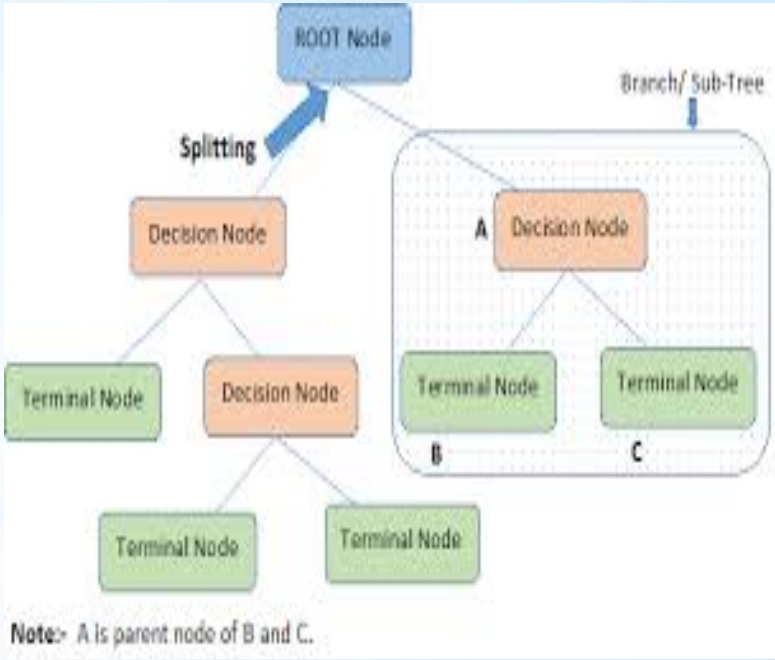
- In a “good” decision tree, each non-leaf node should correspond to the input attribute which is the most informative about the output attribute amongst all the input attributes not yet considered in the path from the root node to that node. This is because we would like to predict the output attribute using the smallest possible number of questions on average.

Building a decision tree

- The classification model is a tree, called decision tree
- Defined by a hierarchy of rules (in form of a tree)
- Rules form the internal nodes of the tree (topmost node = root)
- Each rule (internal node) tests the value of some property of the atrribute

Important Terminology related to Decision Trees

Root Node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.
Splitting: It is a process of dividing a node into two or more sub-nodes.
Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node.
Leaf/ Terminal Node: Nodes do not split is called Leaf or Terminal node.
Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.



What are the key parameters of tree modeling and how can we avoid over-fitting in decision trees?

Overfitting is one of the key challenges faced while modeling decision trees. If there is no limit set of a decision tree, it will give you 100% accuracy on training set because in the worse case it will end up making 1 leaf for each observation. Thus, preventing overfitting is pivotal while modeling a decision tree and it can be done in 2 ways:

1. Setting constraints on tree size
2. Tree pruning

Setting Constraints on Tree Size

There are various parameters in decision trees which can be set. These parameters are available in R & Python.

1. **Minimum samples for a node split**

Defines the minimum number of samples (or observations) which are required in a node to be considered for splitting.

Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.

Too high values can lead to under-fitting hence, it should be tuned using CV.

2. Minimum samples for a terminal node (leaf)

Defines the minimum samples (or observations) required in a terminal node or leaf.

Used to control over-fitting similar to min_samples_split.

Generally lower values should be chosen for imbalanced class problems because the regions in which the minority class will be in majority will be very small.

3. Maximum depth of tree (vertical depth)

The maximum depth of a tree.

Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.

Should be tuned using CV.

4. Maximum number of terminal nodes

The maximum number of terminal nodes or leaves in a tree.

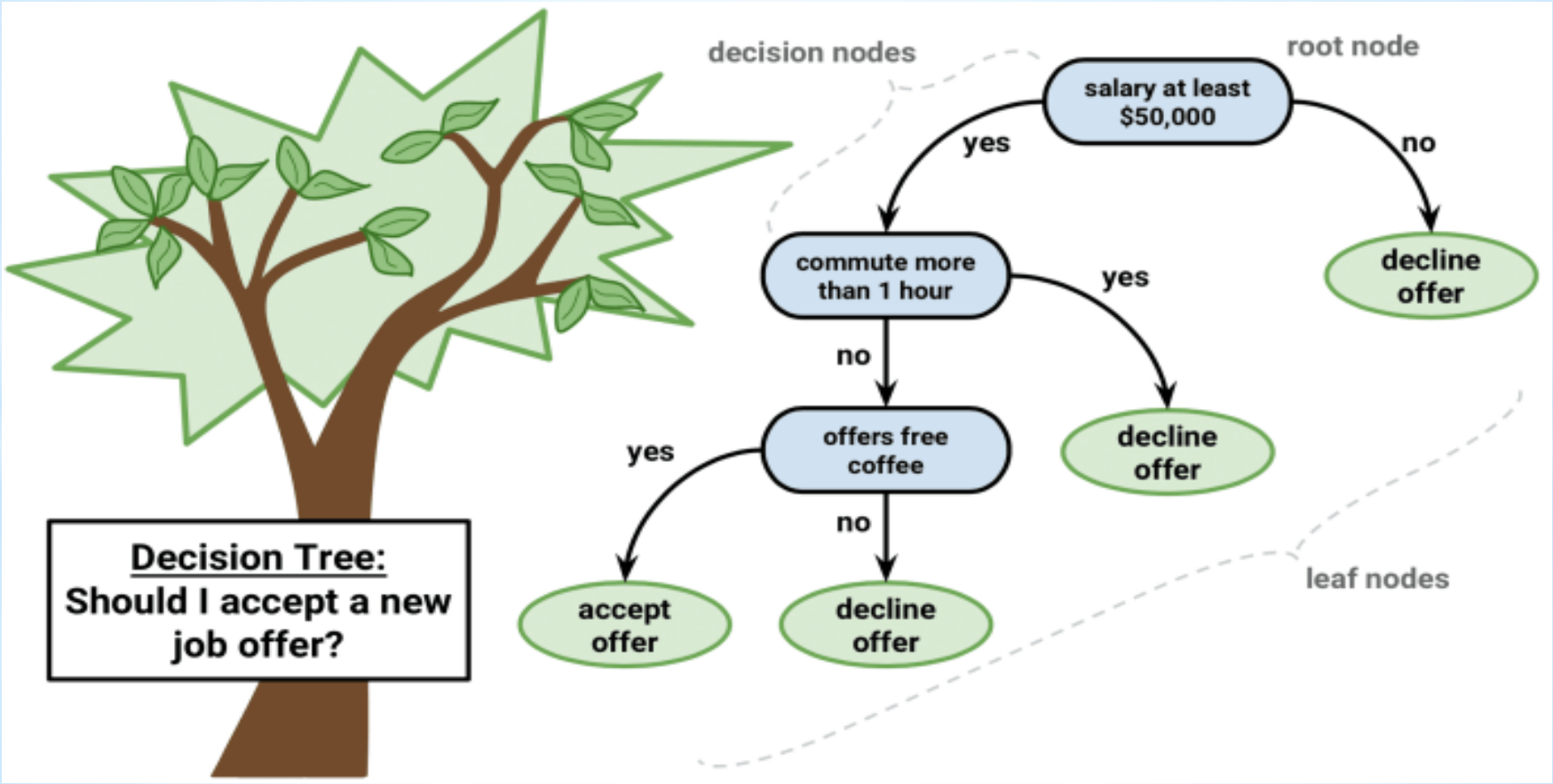
Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves.

5. Maximum features to consider for split

The number of features to consider while searching for a best split. These will be randomly selected.

As a thumb-rule, square root of the total number of features works great but we should check upto 30-40% of the total number of features.

Higher values can lead to over-fitting but depends on case to case.

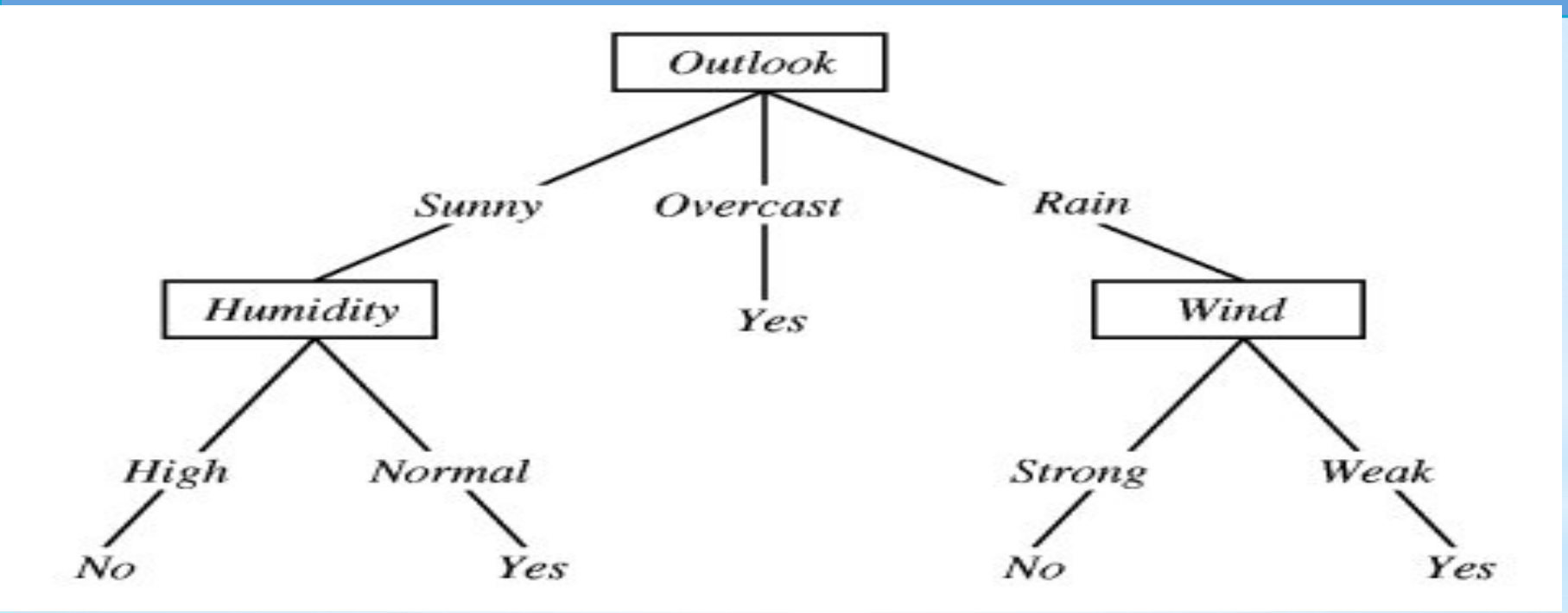

decision nodes
root node
salary at least $50,000
yes
no
commute more than 1 hour
yes
decline offer
no
offers free coffee
yes
decline offer
no
accept offer
decline offer
leaf nodes
Decision Tree:
Should I accept a new job offer?

Example

- Deciding whether to play or not to play on a certain day
  - A classification problem (play vs. no play)
  - Each input has 4 features (outlook, Temp, Humidity, Wind)

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Example



- DT can be used to predict play or no-play for a new day
  - By testing the features of that day
  - in the order defined by DT

- Question: Why does it make more sense to test the feature “outlook” first?
- Answer: Of all the 4 features, it is most informative
- We will see shortly how to quantify the informative-ness
- Information content of a feature decides its position in the DT

Attributes Selection

If dataset consists of “n” attributes then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a complicated step. By just randomly selecting any node to be the root can’t solve the issue. If we follow a random approach, it may give us bad results with low accuracy.

For solving this attribute selection problem, researchers worked and devised some solutions. They suggested using some criterion like information gain, gini index, etc. These criterions will calculate values for every attribute. The values are sorted, and attributes are placed in the tree by following the order i.e, the attribute with a high value(in case of information gain) is placed at the root. There are also other types of measures which can be used to calculation most informative attribute such as information gain ratio, variance reduction etc.

Choose an attribute to partition data

- The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria is different for classification and regression trees.
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.

Choose an attribute to partition data

$$Entropy = \sum_j -p_j log_2 p_j$$

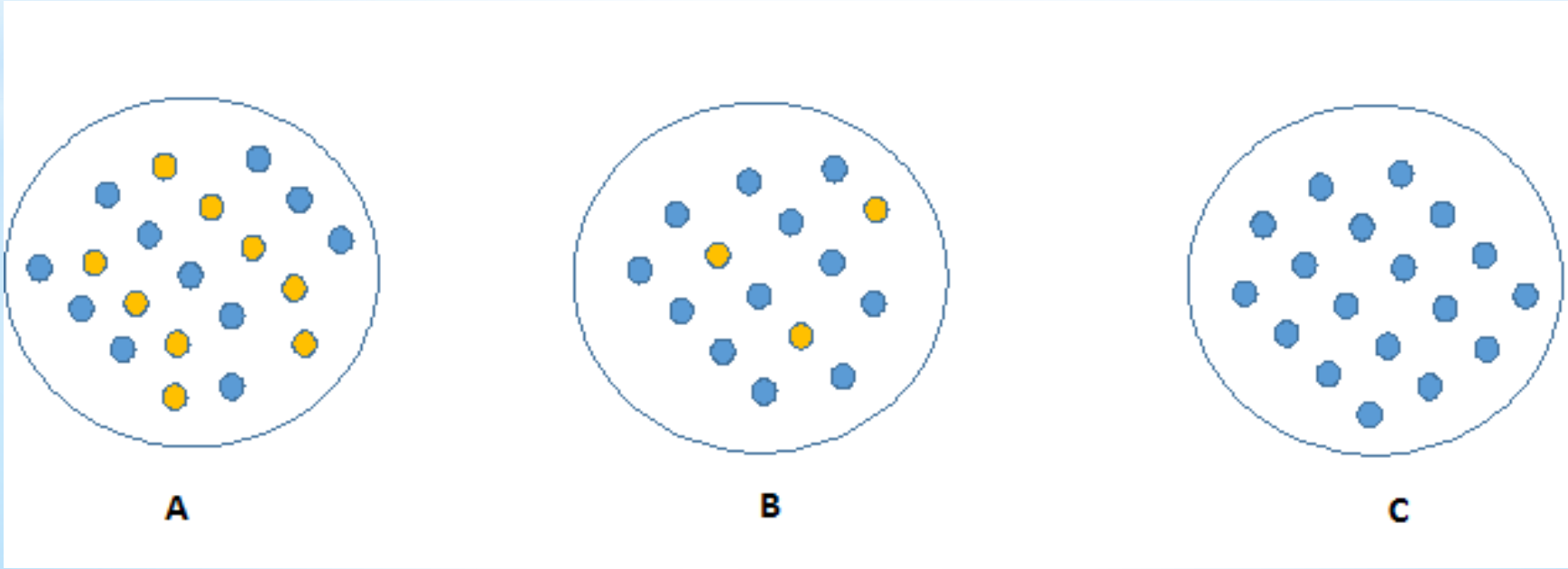
$$Gini\ Index = 1 - \sum_j p_j^2$$

$$Classification\ Error = 1 - max\{p_j\}$$

All above formulas contain values of probability of p_j of class j

Choose an attribute to partition data

Which node is described easily or pure. Clearly it is C because it requires less information to describe as all values are similar. On the other hand, B requires more information to describe it and A requires the maximum information. In other words, we can say that C is a Pure node, B is less impure and A is more impure.



Choose an attribute to partition data

Entropy refers to disorder or uncertainty.

Entropy is the measure of impurity, disorder or uncertainty in a bunch of examples. It gives us the degree of disorganization in out data.

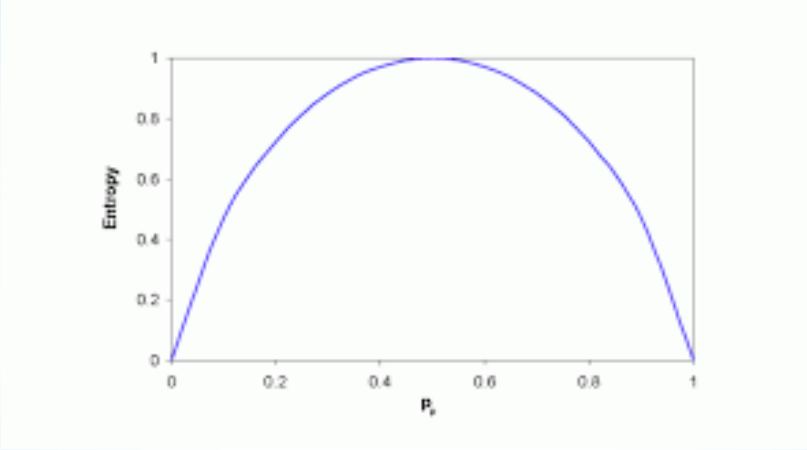
Entropy controls how a decision tree decides to split the data. It actually effects how a decision tree draws its boundaries.

Choose an attribute to partition data

Entropy & Information Gain

The word entropy is borrowed from Thermodynamics which is a measure of variability or chaos or randomness.

The graph shows the variation of entropy with the probability of a class, we can clearly see that entropy is maximum when probability of either of the classes is equal.



Choose an attribute to partition data

Now, you can understand that when a decision is taken to split the data, it selects the variable which will give us maximum reduction in system entropy.

Choose an attribute to partition data

It can be concluded that less impure node requires less information to describe it and , more impure node requires more information.

Entropy is a measure of degree of randomness in a system. If the sample is completely homogenous, then the entropy is zero and if the sample is an equally divided it has entropy of one.

Steps to calculate entropy for a split

1. Calculate entropy for parent node
2. Calculate entropy for each individual node of split and calculate weighted average of all sub-nodes available in split.

We will split where entropy is lowest or information gain (1-entropy) is maximum

Choose an attribute to partition data

What is Information gain and why it is matter in Decision Tree?

Information gain measures how much “information” a feature gives us about the class. Features that perfectly partition should give maximum information whereas unrelated features should give no information.

Information gain is the expected reduction in entropy caused by partitioning the examples according to a given feature.

Information gain is the main key that is used by Decision Tree Algorithms. It decides which feature goes into a decision node. To minimize the decision tree depth the feature with the most entropy reduction is the best choice.

Decision Trees algorithm will always tries to maximize information gain.

Choose an attribute to partition data

An attribute with highest Information gain will split first.

The equation of Information gain:

Information gain = entropy(parent)-[weights average] * entropy(children)

Choose an attribute to partition data

The IG (S,A) of an attribute an attribute A relative to a collection of examples is defined as

$$Gain(S,A) = \underbrace{Entropy(S)}_{\text{original entropy of S}} - \underbrace{\sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)}_{\text{relative entropy of S}}$$

S = Each value v of all possible values of attribute A

Sv = Subset of S for which attribute A has value v

|Sv| = Number of elements in Sv

|S| = Number of elements in S

Example

We will consider a toy data set to understand how a decision tree is built.

In this dataset, there are five categorical attributes *gender*, *car ownership*, *travel cost, Income level,* and *transportation mode.*

We are interested in building a system which will enable us to decide which mode of transportation to take on the basis of other four variables*, i.e.*, we wish to predict the value of transportation mode using *gender, car ownership*, *travel cost* and *income level*.

We can think of the attribute we wish to predict, *i.e., transportation mode,* as the **output** attribute and the other attributes as **input** attributes

Example

Attributes				Classes
Gender	Car ownership	Travel Cost ($)/km	Income Level	Transportation mode
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car
Female	1	Cheap	Medium	Train
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train

The attributes are Gender, Car ownership, Travel cost, and Income level. They can have the following values:

Gender = { male, female }

Car ownership = {0,1,2 }

Travel cost = { cheap, standard, expensive }

Income level = { low, medium, high }

We need to find which attribute will be the root node in our decision tree.

Example

In example, the classes of transportation mode consists of three groups of Bus, Car and Train. In this case we have 4 buses, 3 cars, and 3 trains. Based on this data, we can calculate the probability of each class. We have

prob(bus) = 4/10 = 0.4
prob(car) = 3/10 = 0.3
prob (Train) = 3/10 + 0.3

Having the probability of each class, we are ready to compute the quantitative indices of impurity

Example

Entropy:

Entropy = -0.4log(0.4)-0.3log(0.3)-0.3log(0.3) = 1.571

Entropy of a pure table(consist of single class) is zero because the probability is 1 and log(1) = 0. Entropy reaches maximum value when all classes in the table have equal prob.

Gini Index:

Gini index = 1-(0.4^2+0.3^2+0.3^2) = 0.660

Gini index of a pure table(consist of single class) is zero because the prob. Is 1 and 1-1^2 = 0.

Classification Error

Classification error = 1-max(0.4, 0.3, 0.3) = 0.60

Travel Cost ($)/km	Transportation mode
Cheap	Bus
Cheap	Bus
Cheap	Bus
Cheap	Bus
Cheap	Train
Expensive	Car
Expensive	Car
Expensive	Car
Standard	Train
Standard	Train

Travel Cost ($)/km	Classes
Cheap	Bus
Cheap	Bus
Cheap	Bus
Cheap	Bus
Cheap	Train

4B, 1T

Entropy	0.722
Gini index	0.320
classification error	0.200

Travel Cost ($)/km	Classes
Expensive	Car
Expensive	Car
Expensive	Car

3C

Entropy	0.000
Gini index	0.000
classification error	0.000

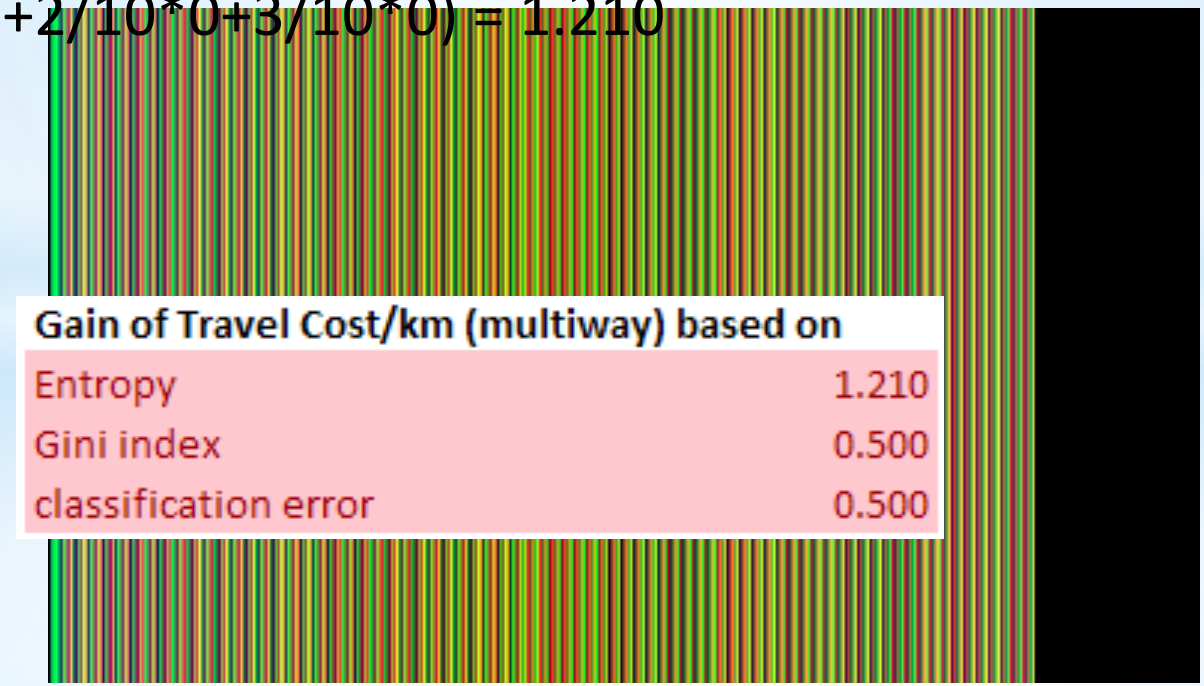
Travel Cost ($)/km	Classes
Standard	Train
Standard	Train

2T

Entropy	0.000
Gini index	0.000
classification error	0.000

For example: our data table has 4 bus, 3 cars, and 3 trains. Now we try the attribute Travel cost per km which we split into three: cheap that has classes of 4 bus, 1 train (thus entropy of 0.722), Standard that has classes of 2 trains (entropy = 0) and expensive with 3 Cars (Entropy =0)

The IG of attribute Travel cost per km = 1.571-(5/10*0.722+2/10*0+3/10*0) = 1.210



Gain of Travel Cost/km (multiway) based on	
Entropy	1.210
Gini index	0.500
classification error	0.500

For each attribute in our data, we compute IG.

Subset

Gender	Classes
Female	Bus
Female	Car
Female	Car
Female	Train
Female	Train

1B, 2C, 2T

Entropy	1.522
Gini index	0.640
classification error	0.600

Gender	Classes
Male	Bus
Male	Bus
Male	Bus
Male	Car
Male	Train

3B, 1C, 1T

Entropy	1.371
Gini index	0.560
classification error	0.400

Gain of Gender based on

Entropy	0.125
Gini index	0.060
classification error	0.100

Car ownership	Classes
0	Bus
0	Bus
0	Train

2B, 1T

Entropy	0.918
Gini index	0.444
classification error	0.333

Car ownership	Classes
1	Bus
1	Bus
1	Car
1	Train
1	Train

2B, 1C, 2T

Entropy	1.522
Gini index	0.640
classification error	0.600

Car ownership	Classes
2	Car
2	Car

2C

Entropy	0.000
Gini index	0.000
classification error	0.000

Gain of Car ownership (multiway) based on

Entropy	0.534
Gini index	0.207
classification error	0.200

Income Level	Classes
High	Car
High	Car

2C

Entropy	0.000
Gini index	0.000
classification error	0.000

Income Level	Classes
Low	Bus
Low	Bus

2B

Entropy	0.000
Gini index	0.000
classification error	0.000

Income Level	Classes
Medium	Bus
Medium	Bus
Medium	Car
Medium	Train
Medium	Train
Medium	Train

2B, 1C, 3T

Entropy	1.459
Gini index	0.611
classification error	0.500

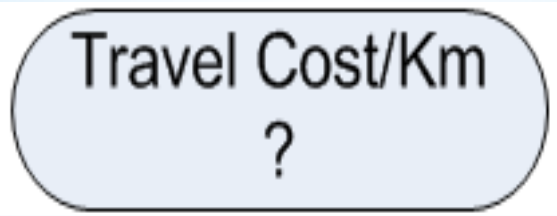
Gain of Income Level (multiway) based on

Entropy	0.695
Gini index	0.293
classification error	0.300

Results of first Iteration

Gain	Gender	Car ownership	Travel Cost/KM	Income Level
Entropy	0.125	0.534	**1.210**	0.695
Gini index	0.060	0.207	**0.500**	0.293
Classification error	0.100	0.200	**0.500**	0.300

Once we have IG for all the attributes we find the optimum attribute that produce the max. IG. In our case, travel cost per km produces the maximum IG. We put this attribute into the node of our decision tree.



Once we obtain the optimum attribute, we can split the data table according to that attribute.

Data

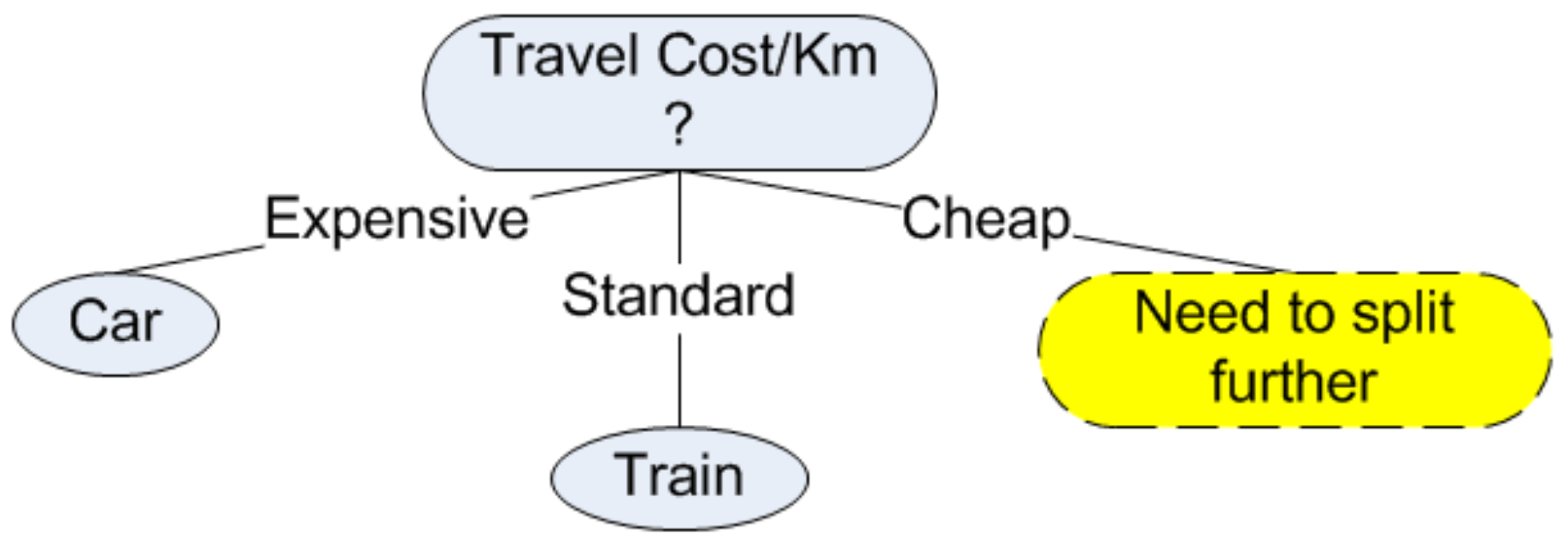
Attributes				Classes
Gender	Car	Travel Cost	Income Level	Transportation
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train
Female	1	Expensive	High	Car
Male	2	Expensive	Medium	Car
Female	2	Expensive	High	Car
Male	0	Standard	Medium	Train
Female	1	Standard	Medium	Train

Attributes				Classes
Gender	Car ownership	Travel Cost /km	Income Level	Transportation mode
Female	0	Cheap	Low	Bus
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train

Attributes				Classes
Gender	Car ownership	Travel Cost /km	Income Level	Transportation mode
Female	1	Expensive	High	Car
Female	2	Expensive	High	Car
Male	2	Expensive	Medium	Car

Attributes				Classes
Gender	Car ownership	Travel Cost /km	Income Level	Transportation mode
Female	1	Standard	Medium	Train
Male	0	Standard	Medium	Train

After the split of the data, we can see clearly that the value of Expensive travel cost/km is associated only with pure class of car while standard travel cost/km is only related to pure class of Rrain. Pure class is always assigned to leaf node.



For cheap travel cost/km, the classes are not pure, thus we need to split further in the next iteration. For second iteration our data is only from the Cheap travel cost/km. We remove attribute travel cost/km from the data because they are equal and redundant

Attributes				Classes
Gender	Car ownership	Travel Cost /km	Income Level	Transportation mode
Female	0	Cheap	Low	Bus
Male	0	Cheap	Low	Bus
Male	1	Cheap	Medium	Bus
Male	1	Cheap	Medium	Bus
Female	1	Cheap	Medium	Train

Data

Attributes			Classes
Gender	Car ownership	Income Level	Transportation mode
Female	0	Low	Bus
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus
Female	1	Medium	Train

Now we have three attributes: gender, car ownership and Income level .

Data second iteration

Attributes			Classes
Gender	**Car ownership**	**Income Level**	**Transportation mode**
Female	0	Low	Bus
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus
Female	1	Medium	Train

4B, 1T

Entropy	0.722
Gini index	0.320
classification error	0.200

Again calculate IG.

Subsets of second iterations

Gender	Classes
Female	Bus
Female	Train

1B, 1T

Entropy	1.000
Gini index	0.500
classification	0.500

Car ownership	Classes
0	Bus
0	Bus

2B

Entropy	0.000
Gini index	0.000
classification error	0.000

Income Level	Classes
Low	Bus
Low	Bus

2B

Entropy	0.000
Gini index	0.000
classification error	0.000

Gender	Classes
Male	Bus
Male	Bus
Male	Bus

3B

Entropy	0.000
Gini index	0.000
classification	0.000

Car ownership	Classes
1	Bus
1	Bus
1	Train

2B, 1T

Entropy	0.918
Gini index	0.444
classification error	0.333

Income Level	Classes
Medium	Bus
Medium	Bus
Medium	Train

2B, 1T

Entropy	0.918
Gini index	0.444
classification error	0.333

Gain of Gender based on

Entropy	0.322
Gini index	0.120
classification	0.000

Gain of Car ownership based on

Entropy	0.171
Gini index	0.053
classification error	0.000

Gain of Income Level based on

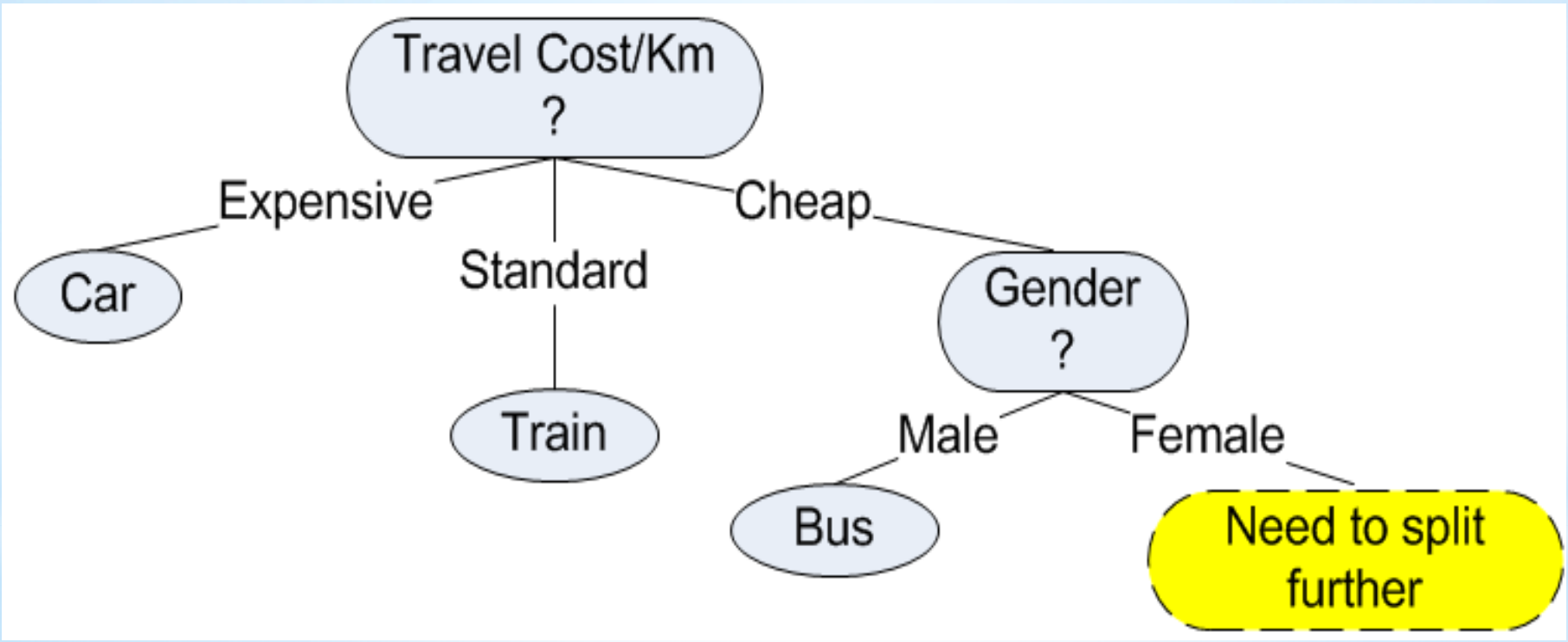
Entropy	0.171
Gini index	0.053
classification error	0.000

Max. gain is for gender.

Attributes			Classes
Gender	Car ownership	Income Level	Transportation mode
Female	0	Low	Bus
Female	1	Medium	Train
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus

Attributes			Classes
Gender	Car ownership	Income Level	Transportation mode
Female	0	Low	Bus
Female	1	Medium	Train

Attributes			Classes
Gender	Car ownership	Income Level	Transportation mode
Male	0	Low	Bus
Male	1	Medium	Bus
Male	1	Medium	Bus


Travel Cost/Km
?
Expensive
Standard
Cheap
Car
Train
Gender
?
Male
Female
Bus
Need to split
further

Third iteration

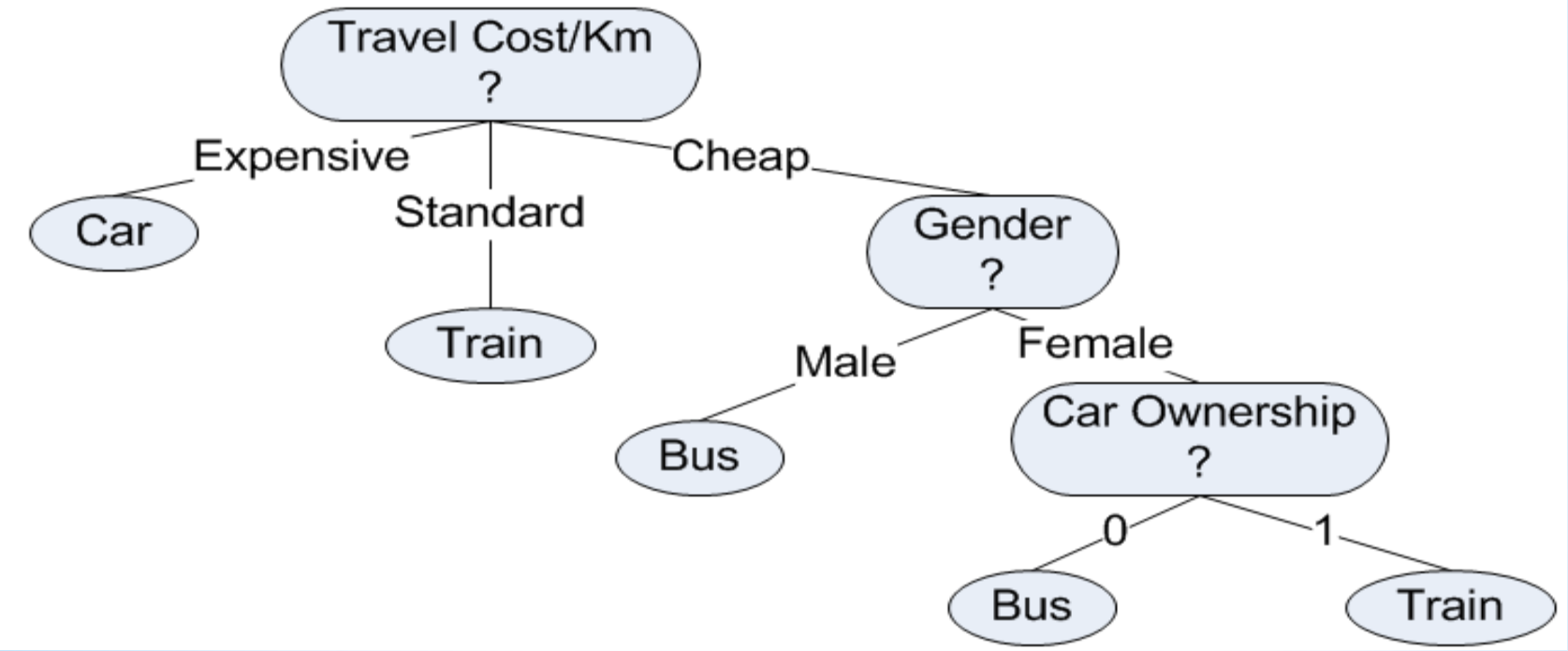
Attributes			Classes
Gender	Car ownership	Income Level	Transportation mode
Female	0	Low	Bus
Female	1	Medium	Train

Data third iteration

Attributes		Classes
Car ownership	Income Level	Transportation mode
0	Low	Bus
1	Medium	Train

Third iteration

If you observe it consists of two rows. Each row has distinct values. If we use attribute car ownership, we get pure class for each of its value. Similarly, attribute income level will also give pure class for each value. Therefore we can use either one of the two attributes.



The decision tree can also be expressed in rule format:

IF travel cost = expensive THEN transportation mode = Car

IF travel cost = standard THEN transportation mode = Train

IF travel cost = cheap AND gender = male THEN transportation mode = Bus

IF travel cost = cheap AND gender = female and car ownership = 0 THEN transportation mode = Bus

IF travel cost = cheap AND gender = female and car ownership = 1 THEN transportation mode = Bus

Advantages

- Simple to understand and interpret: Decision tree output is very easy to understand even for people from non-analytical background.

- It does not require any statistical knowledge to read and interpret them. Its graphical representation is very intuitive and users can easily relate their hypothesis.

- Requires little data preparation: It requires less data cleaning compared to some other modeling techniques. It is not influenced by outliers and missing values to a fair degree.

- It can take different types of input and output variables which can be categorical, binary and numeric value

- Non Parametric Method: Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Disadvantages

- DT only considers one input attribute at a time but not a combination of multiple input variables.
- Once learned it cannot be updated incrementally. When new training data arrives, you have to throw away the old tree and retrain every data from scratch
- Overfitting: Can create over-complex tree that do not generalize well
- For data including categorical variables with different number of levels, information gain in decision trees is biased in favor of those attributes with more levels

Other issues in decision tree learning

In practice standalone decision tree are rarely used in practice. Tree ensemble are common way to use decision trees.

Instead of picking a single model, Ensemble method combines multiple models in a certain way to fit the training data. There are two primary ways: “Bagging and Boosting”

Another approach “Random Forest” are also used.