Decision Trees

What are Decision Trees ?

Decision Tree is an algorithm used for building classification and regression models by representing it in the form of a tree structure.

Types of Decision Trees

DATA FOLKZ®
#CATAPULT DATA LEADERS

Categorical Variable Decision Tree: Decision Tree which has categorical target variable then it called as categorical variable decision tree.

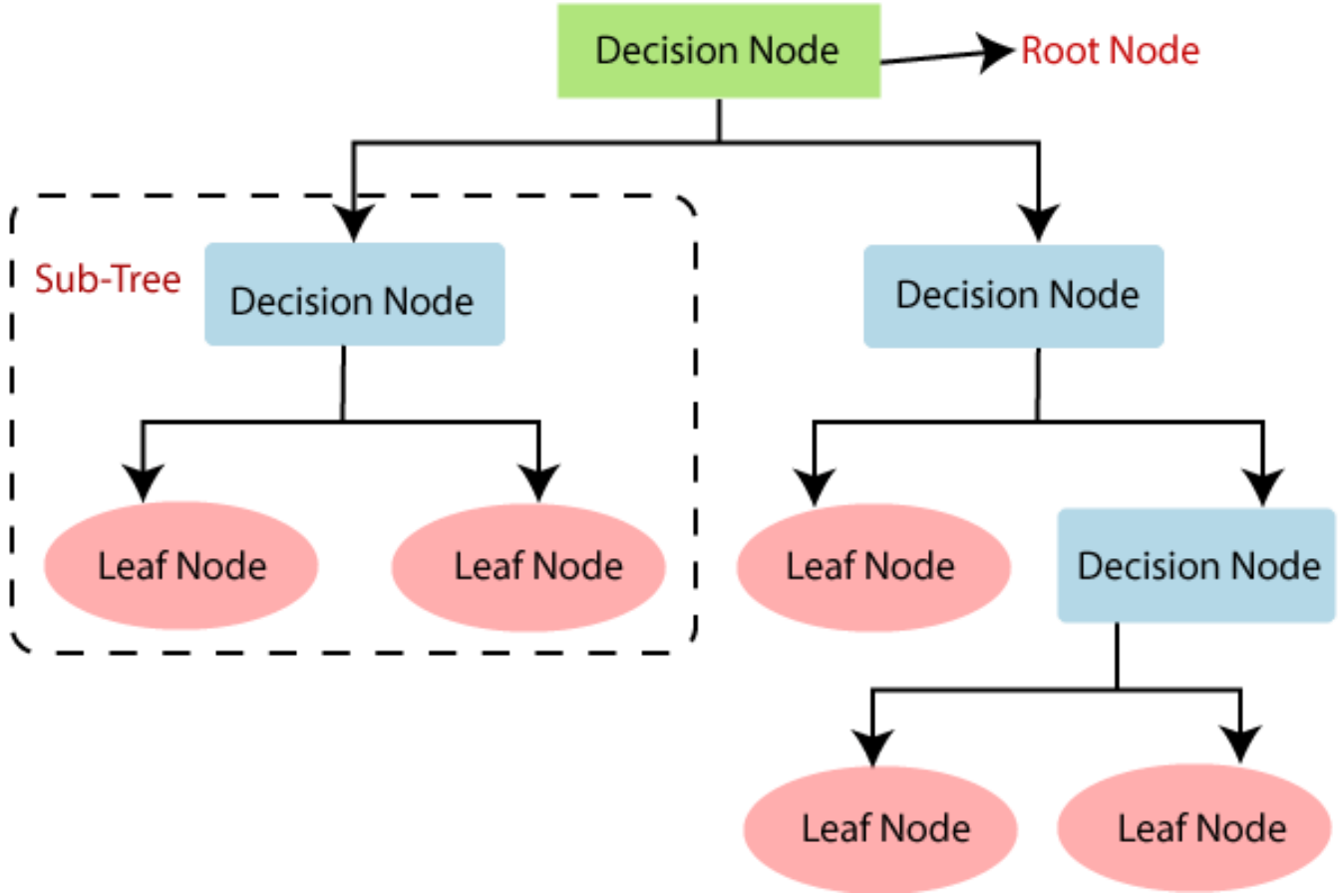
Continuous Variable Decision Tree: Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

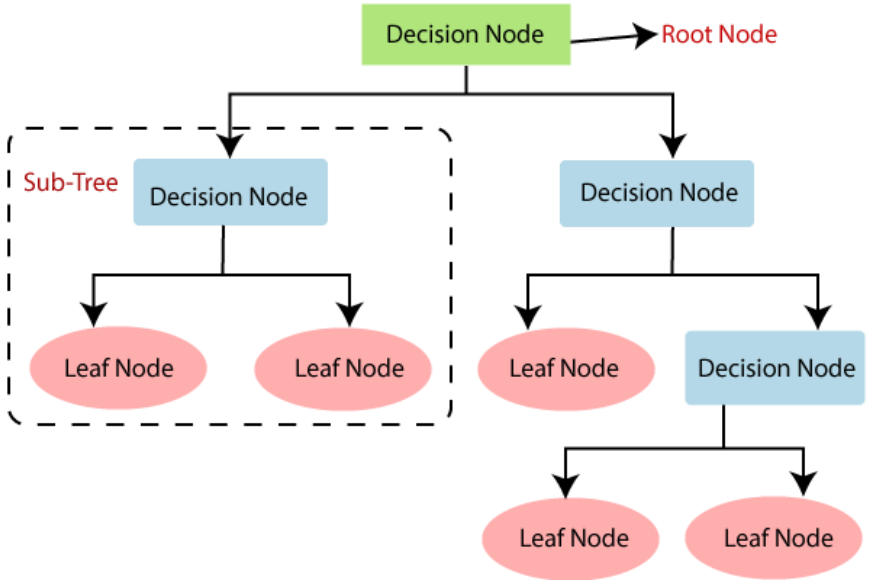
How does a Decision Tree look like ?

Root Node - The root node is the highest node in the tree structure and has no parent. This node is a global element and represents the entire message. It may have one or more child nodes but can never have sibling nodes or be repeating.

Internal Node - An internal node (also known as an inner node, inode for short, or branch node) is any node of a tree that has child nodes.

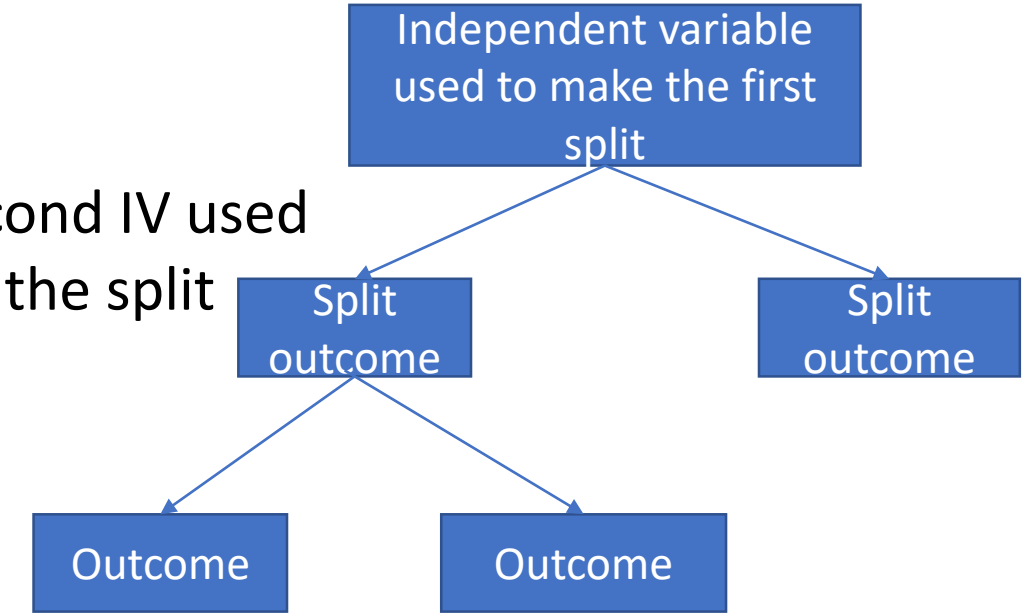
Leaf Nodes - An external node (also known as an outer node, leaf node, or terminal node) is any node that does not have child nodes.



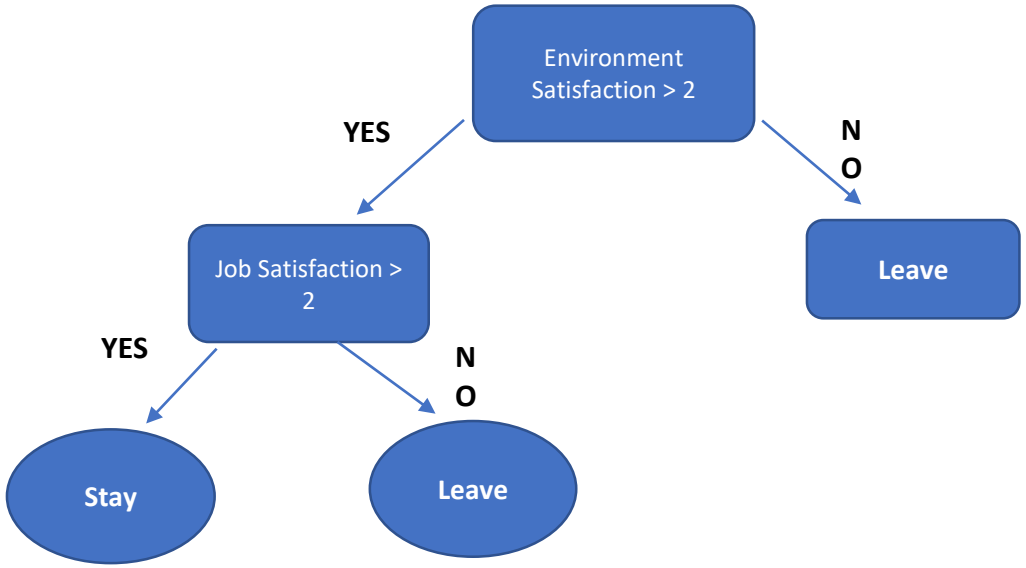


DATA FOLKZ®
#CATAPULT DATA LEADERS

Second IV used for the split



Will the employee stay with the company or leave ?



Example of Decision Tree

Example of Decision tree

- In the above slide , we discussed whether employee will stay or leave the organisation based on the information we have regarding Environment Satisfaction and Job Satisfaction .

- We used Environment Satisfaction as the first criteria to decide whether he will stay or not .

But how do we decide this ?
How do we decide which variable to use first for the decision making ?

How does the attribute for the split is decided ?

The popular attribute selection measures are :

- Entropy / Information Gain
- Gini index

If dataset consists of **“n”** attributes like our dataset consists 35 attributes , then deciding which attribute to place at the root or at different levels of the tree as internal nodes is a n essential step .

For solving this attribute selection problem, we have some criterion like **entropy/ information gain, Gini index,** etc.

Entropy

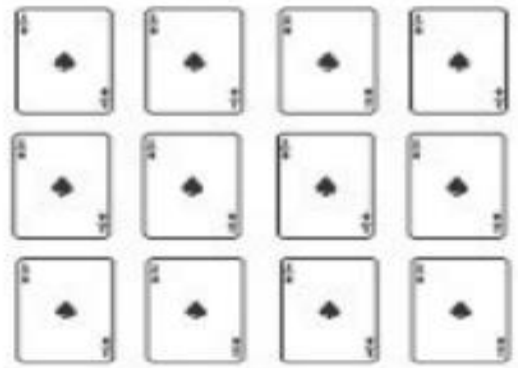
DATA FOLKZ®
#CATAPULT DATA LEADERS

Entropy is the measurement of impurity or randomness in the data . In other words, how much variance the data has or unpredictability of a dataset.

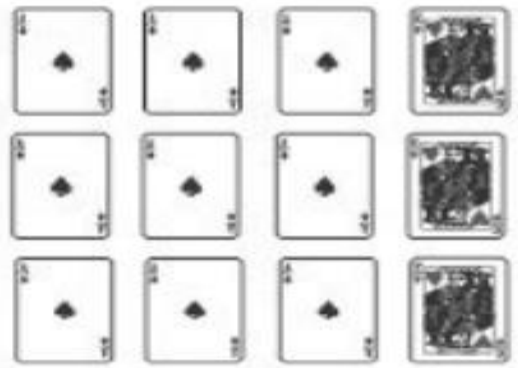
The value of entropy lies between 0 to 1 ,
Where entropy = 0 means no impurity
And entropy = 1 means high impurity .

The variable/ Descriptive feature with the lower entropy is used for split .

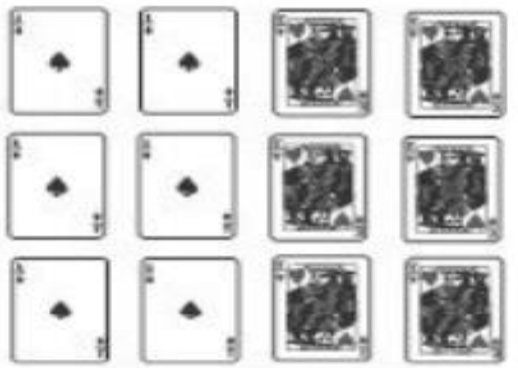
The entropy of different sets of playing cards measured in bits.



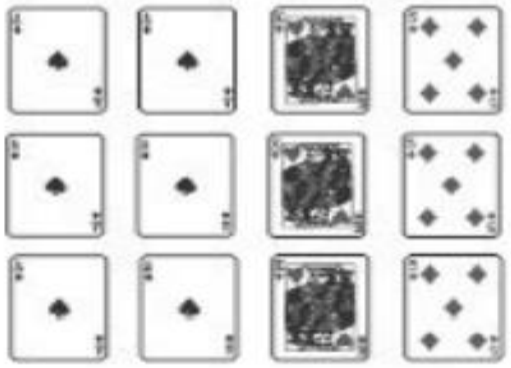
(a) $H(card) = 0.00$



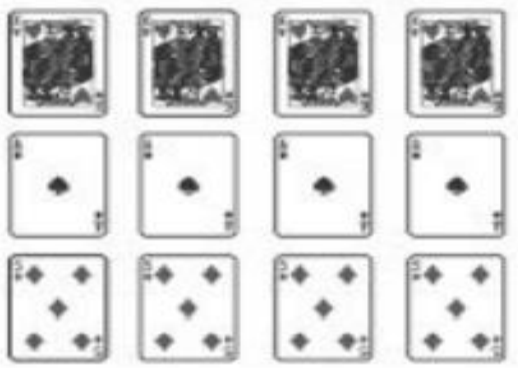
(b) $H(card) = 0.81$



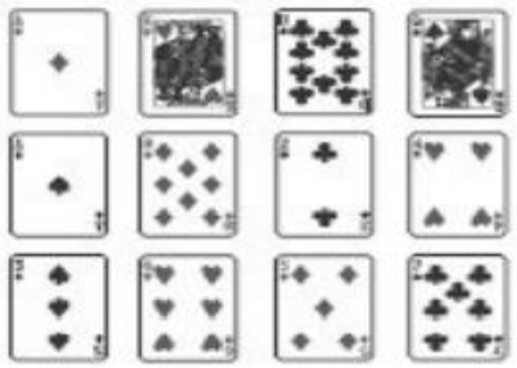
(c) $H(card) = 1.00$



(d) $H(card) = 1.50$



(e) $H(card) = 1.58$



(f) $H(card) = 3.58$

Probability to Entropy

We can transform the probabilitiy of randomly selecting an element from a set to entropy values.

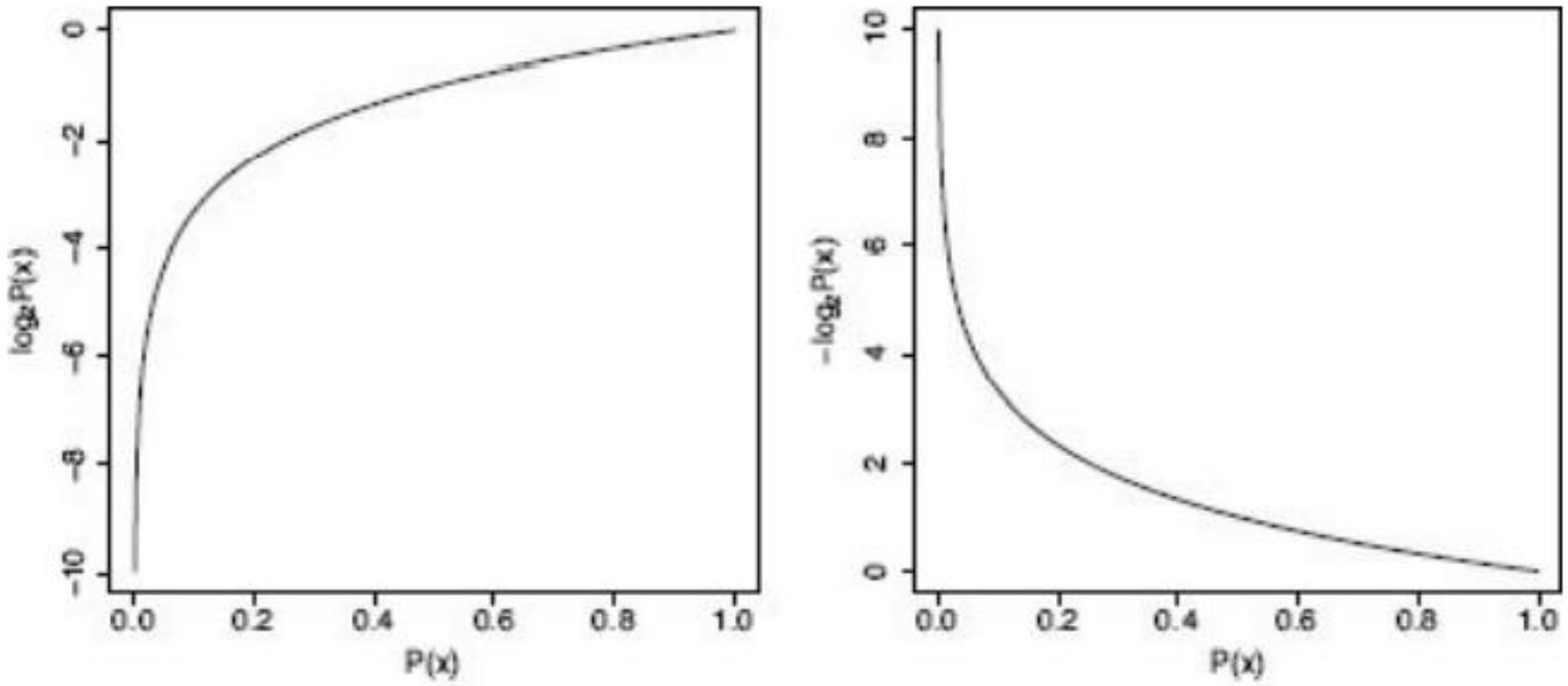
An outcome with a large probability should map to a low entropy value, while an outcome with a small probability should map to a large entropy value.

The binary logarithm (a logarithm to the base 2) of probabilities ranging from 0 to 1 does almost exactly the transformation that we need.

The logarithm function returns large negative numbers for low probabilities, and small negative numbers for high probabilities

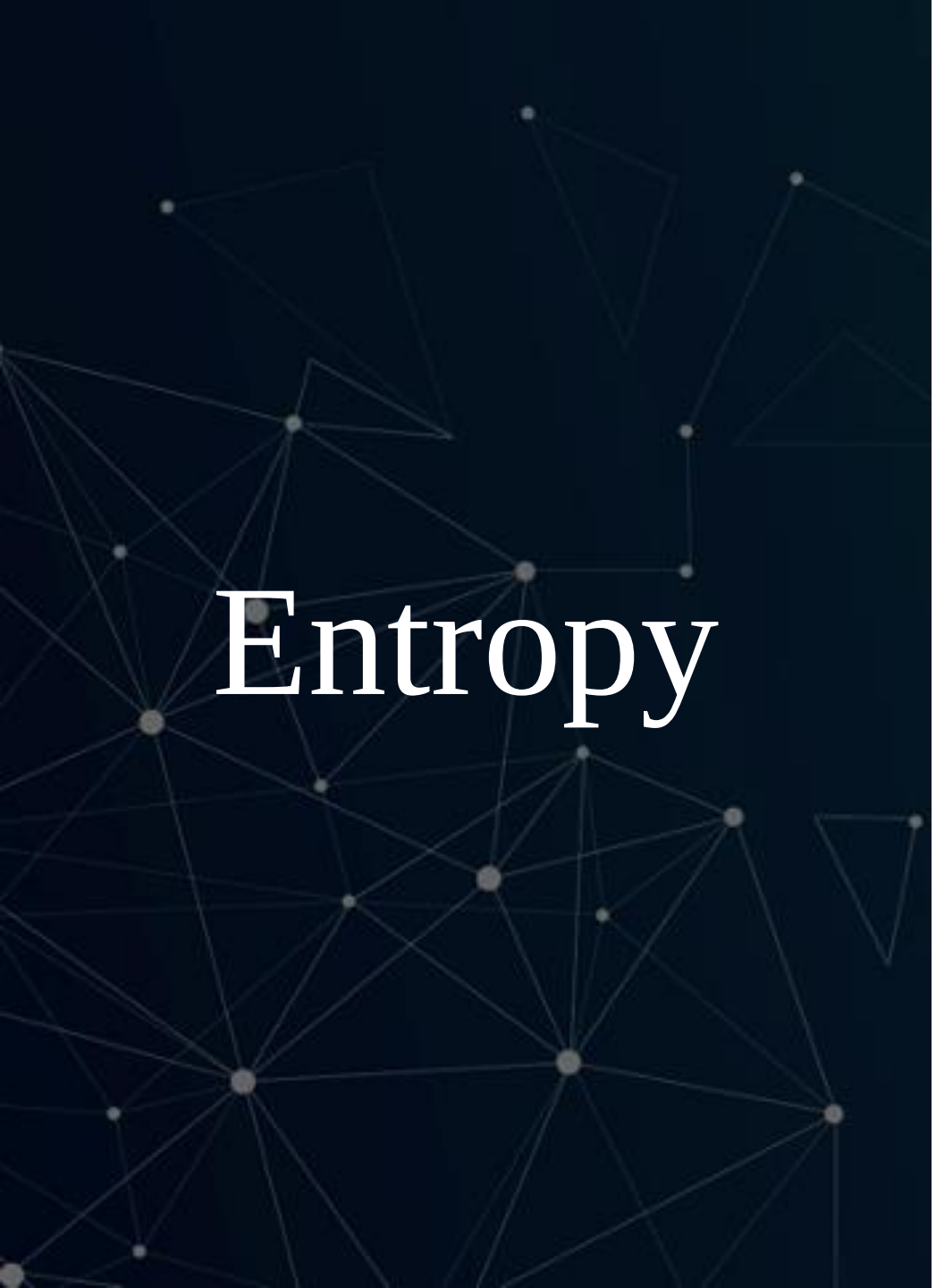
The entropy of different sets of playing cards measured in bits.

We see that the logarithm function returns large negative numbers for low probabilities, and small negative numbers for high probabilities



(a) $P(x)$ and $log_2 P(x)$

(b) $P(x)$ and $-log_2 P(x)$



Entropy

DATA FOLKZ®
#CATAPULT DATA LEADERS

Mathematical equation for entropy:

$$H = -\sum p(x)\ log\ p(x)$$

It is negative summation of probability times the log of probability of item x.

where p(x) is the probability of randomly picking an element of class.

Employee count has single class : 1
It will have no randomness which means it has lowest entropy (almost 0).

Education has multiple classes : 2,1,4,3
It will have high randomness which means it has higher entropy.

Example of Entropy calculation :

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
H(card) &= -\sum_{i=1}^{52} P(card = i) \times log_2(P(card = i)) \\
&= -\sum_{i=1}^{52} 0.019 \times log_2(0.019) \\
&= -\sum_{i=1}^{52} -0.1096 \\
&= 5.700\ bits
\end{aligned}
$$

Example of Entropy calculation :

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
H(suit) &= -\sum_{l \in \{\heartsuit,\clubsuit,\diamondsuit,\spadesuit\}} P(suit = l) \times log_2(P(suit = l)) \\
&= -\big((P(\heartsuit) \times log_2(P(\heartsuit))) + (P(\clubsuit) \times log_2(P(\clubsuit))) \\
&\quad + (P(\diamondsuit) \times log_2(P(\diamondsuit))) + (P(\spadesuit) \times log_2(P(\spadesuit)))\big) \\
&= -\Big(\Big(\tfrac{13}{52} \times log_2\big(\tfrac{13}{52}\big)\Big) + \Big(\tfrac{13}{52} \times log_2\big(\tfrac{13}{52}\big)\Big) \\
&\quad + \Big(\tfrac{13}{52} \times log_2\big(\tfrac{13}{52}\big)\Big) + \Big(\tfrac{13}{52} \times log_2\big(\tfrac{13}{52}\big)\Big)\Big) \\
&= -((0.25 \times -2) + (0.25 \times -2) + (0.25 \times -2) + (0.25 \times -2)) \\
&= 2\ bits
\end{aligned}
$$

Example of Entropy calculation :

DATA FOLKZ®
#CATAPULT DATA LEADERS

An email spam prediction dataset.

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Example of Entropy calculation :

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
H(t,\mathcal{D}) &= -\sum_{l \in \{spam,ham\}} (P(t=l) \times log_2(P(t=l))) \\
&= -\big((P(t=spam) \times log_2(P(t=spam)) \\
&\qquad + (P(t=ham) \times log_2(P(t=ham))\big) \\
&= -\left(\left({}^{3}/_{6} \times log_2({}^{3}/_{6}) \right) + \left({}^{3}/_{6} \times log_2({}^{3}/_{6}) \right) \right) \\
&= 1\ bit
\end{aligned}
$$

Remaining entropy for the SUSPICIOUS WORDS feature

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
&rem(\text{WORDS}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{WORDS}=true}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{WORDS}=true}\right) \right) \\
&\quad + \left(\frac{|\mathcal{D}_{\text{WORDS}=false}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{WORDS}=false}\right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \left(\left({}^{3}/_{3} \times log_2({}^{3}/_{3}) \right) + \left({}^{0}/_{3} \times log_2({}^{0}/_{3}) \right) \right) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \left(\left({}^{0}/_{3} \times log_2({}^{0}/_{3}) \right) + \left({}^{3}/_{3} \times log_2({}^{3}/_{3}) \right) \right) \right) \right) \\
&= 0 \; bits
\end{aligned}
$$

Remaining entropy for the UNKNOWN SENDER feature

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
&rem\,(\text{SENDER}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{SENDER}=true}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{SENDER}=true}\right) \right) \\
&\quad + \left(\frac{|\mathcal{D}_{\text{SENDER}=false}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{SENDER}=false}\right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&= \left({}^{3}/_{6} \times \left(- \left(\left({}^{2}/_{3} \times log_2({}^{2}/_{3}) \right) + \left({}^{1}/_{3} \times log_2({}^{1}/_{3}) \right) \right) \right) \right) \\
&\quad + \left({}^{3}/_{6} \times \left(- \left(\left({}^{1}/_{3} \times log_2({}^{1}/_{3}) \right) + \left({}^{2}/_{3} \times log_2({}^{2}/_{3}) \right) \right) \right) \right) \\
&= 0.9183\ bits
\end{aligned}
$$

Remaining entropy for the CONTAINS IMAGES feature is

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
&rem(\text{IMAGES}, \mathcal{D}) \\
&\quad = \left(\frac{|\mathcal{D}_{\text{IMAGES}=true}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{IMAGES}=true}\right) \right) \\
&\qquad + \left(\frac{|\mathcal{D}_{\text{IMAGES}=false}|}{|\mathcal{D}|} \times H\left(t, \mathcal{D}_{\text{IMAGES}=false}\right) \right) \\
&\quad = \left({}^{2}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&\qquad + \left({}^{4}/_{6} \times \left(- \sum_{l \in \{spam, ham\}} P(t = l) \times log_2(P(t = l)) \right) \right) \\
&\quad = \left({}^{2}/_{6} \times \left(- \left(\left({}^{1}/_{2} \times log_2({}^{1}/_{2}) \right) + \left({}^{1}/_{2} \times log_2({}^{1}/_{2}) \right) \right) \right) \right) \\
&\qquad + \left({}^{4}/_{6} \times \left(- \left(\left({}^{2}/_{4} \times log_2({}^{2}/_{4}) \right) + \left({}^{2}/_{4} \times log_2({}^{2}/_{4}) \right) \right) \right) \right) \\
&\quad = 1 \; bit
\end{aligned}
$$

Information gain calculation for each descriptive feature

DATA FOLKZ®
#CATAPULT DATA LEADERS

$$
\begin{aligned}
IG(\text{WORDS}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{WORDS}, \mathcal{D}) \\
&= 1 - 0 \\
&= 1\ bit \\
IG(\text{SENDER}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{SENDER}, \mathcal{D}) \\
&= 1 - 0.9183 \\
&= 0.0817\ bits \\
IG(\text{IMAGES}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - rem(\text{IMAGES}, \mathcal{D}) \\
&= 1 - 1 \\
&= 0\ bits
\end{aligned}
$$

Information Gain

DATA FOLKZ®
#CATAPULT DATA LEADERS

The concept of entropy plays an important role in calculating Information Gain.

Information Gain: Information gain (IG) measures how much "information" a feature gives us about the class. It Quantifies the quality of a split.

Information Gain = how much Entropy we removed
Higher information gain = More entropy removed

Decision tree at every stage selects the one feature that gives best information gain.

When information gain is 0 means the feature does not divide the working set at all.

An attribute with low entropy and high Information gain will be used to split first.

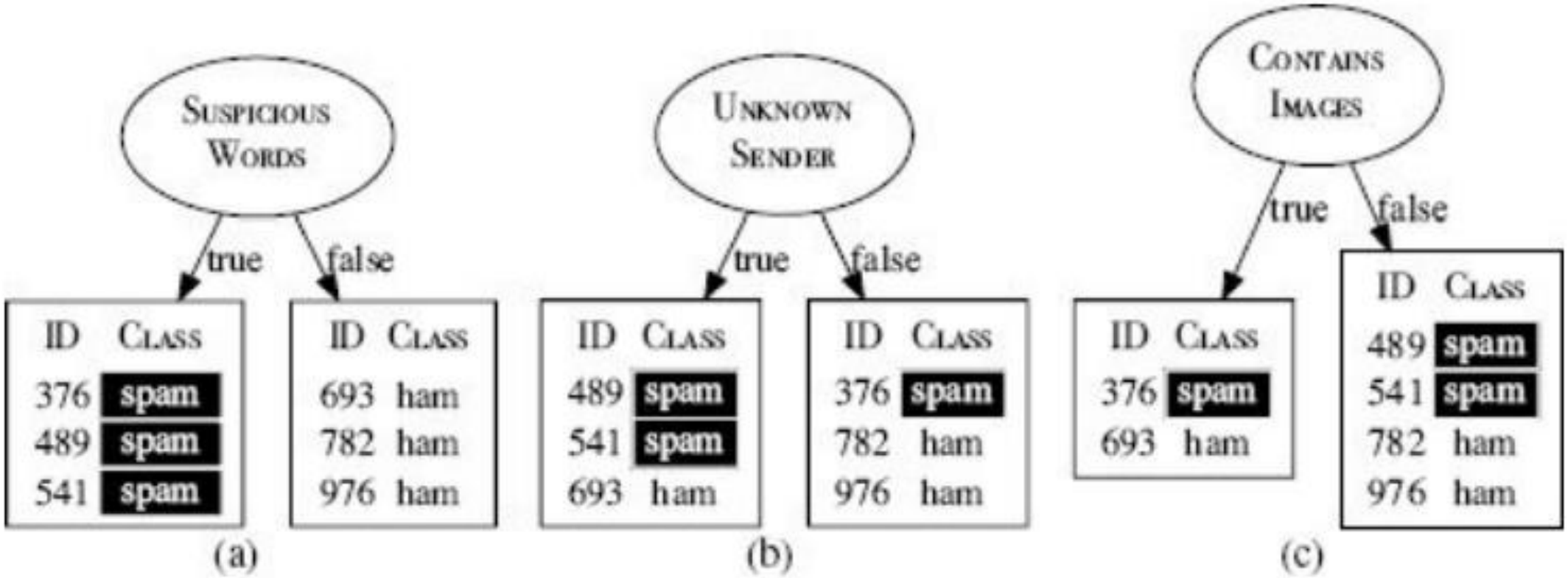
Example: Information Gain

DATA FOLKZ®
#CATAPULT DATA LEADERS

An email spam prediction dataset.

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Example: Information Gain



Information Gain

DATA FOLKZ®
#CATAPULT DATA LEADERS

. Computing information gain is a three-step process:

1. Compute the entropy of the original dataset with respect to the target feature. This gives us an measure of how much information is required in order to organize the dataset into pure sets.

2. For each descriptive feature, create the sets that result by partitioning the instances in the dataset using their feature values, and then sum the entropy scores of each of these sets.

3. This gives a measure of the information that remains required to organize the instances into pure sets after we have split them using the descriptive feature.

4. Subtract the remaining entropy value (computed in step 2) from the original entropy value (computed in step 1) to give the information gain.

Equations: Information Gain

The first equation calculates the entropy for a dataset with respect to a target feature

$$H(t, \mathcal{D}) = -\sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l)))$$

The second equation defines how we compute the entropy remaining after we partition the dataset using a particular descriptive feature d.

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D})$$

Gini Index

DATA FOLKZ®
#CATAPULT DATA LEADERS

Gini Index, also known as Gini impurity, can be understood as calculating how often the target levels of instances in a dataset would be misclassified if predictions were made based only on the distribution of the target levels in the dataset

The value of Gini index lies between 0 to 1
Where 0 means the **purity** of classification and 1 means **randomness** in the classification .

The attribute with the least Gini index is preferred for the split .

Gini Index

The Gini Index is calculated by deducting the sum of squared of probabilities of each class from one .
Mathematically, Gini Index can be expressed as:

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

Where Pi denotes the probability of an element/attribute

Example:

DATA FOLKZ®
#CATAPULT DATA LEADERS

An email spam prediction dataset.

ID	Suspicious Words	Unknown Sender	Contains Images	Class
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Gini Index calculation

DATA FOLKZ®
#CATAPULT DATA LEADERS

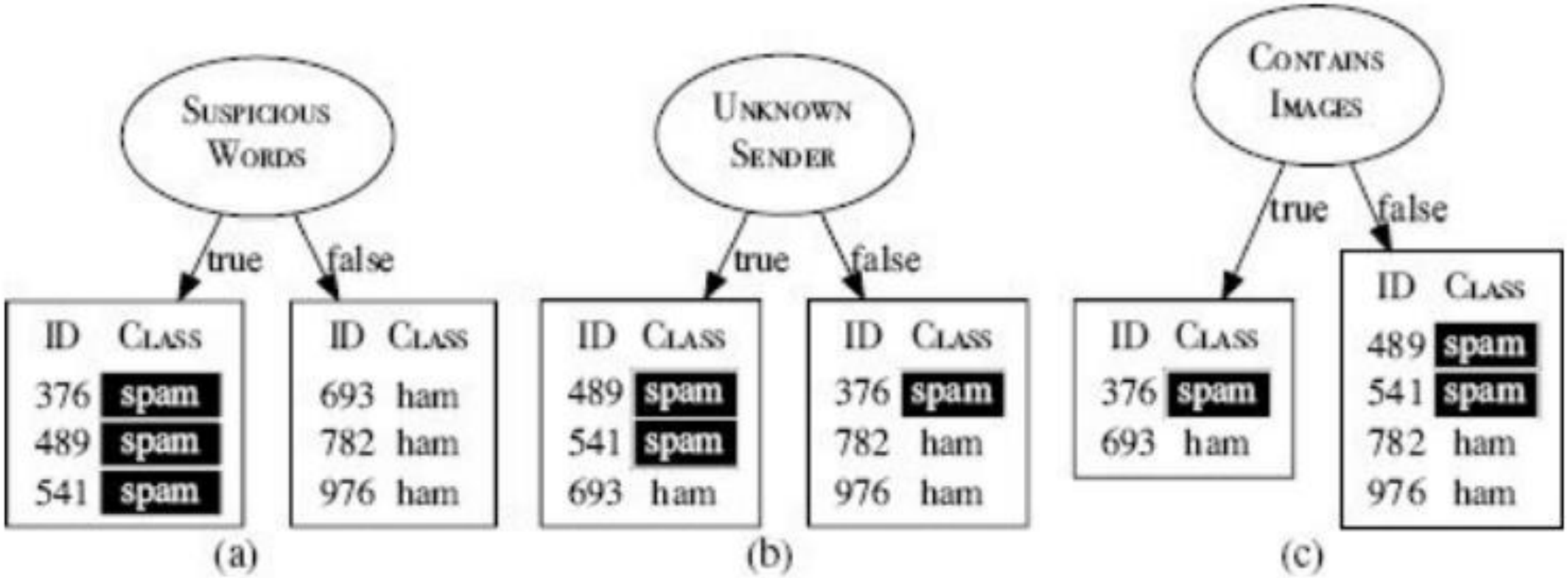
Gini Index calculation

DATA FOLKZ®
#CATAPULT DATA LEADERS

Gini Index calculation

DATA FOLKZ®
#CATAPULT DATA LEADERS

Example: Information Gain

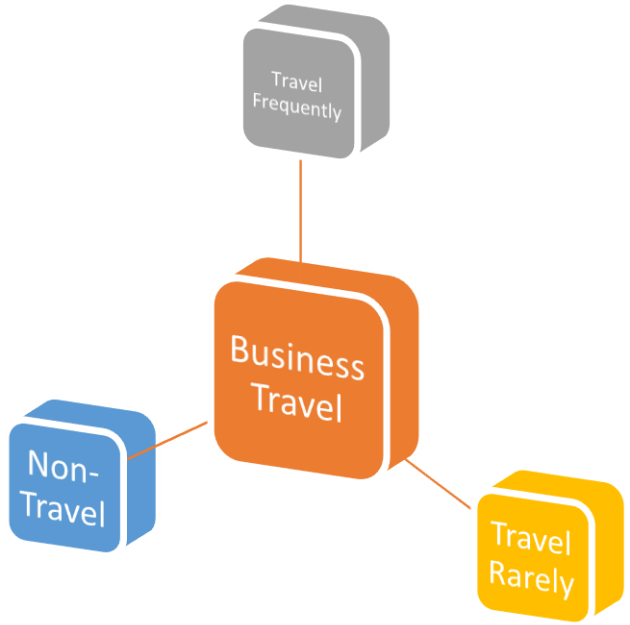


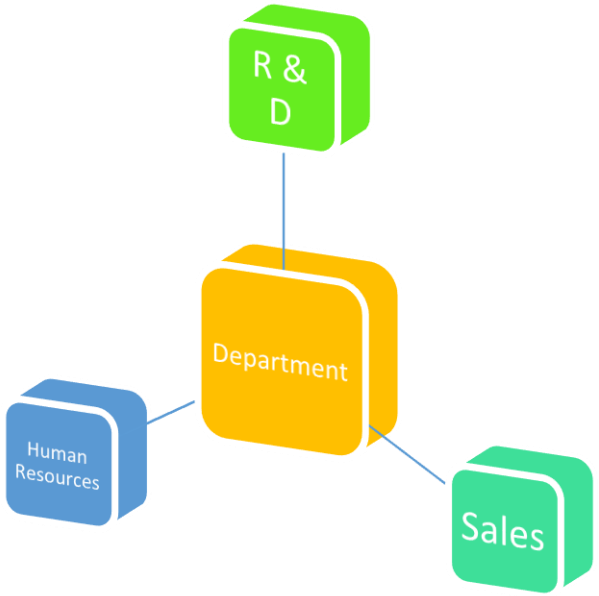
Example

DATA FOLKZ®
#CATAPULT DATA LEADERS

Split on Business Travel

Samples: 1470
Travel Frequently:277 (0.18)
Travel Rarely: 1043 (0.71)
Non Travel: 150 (0.10)

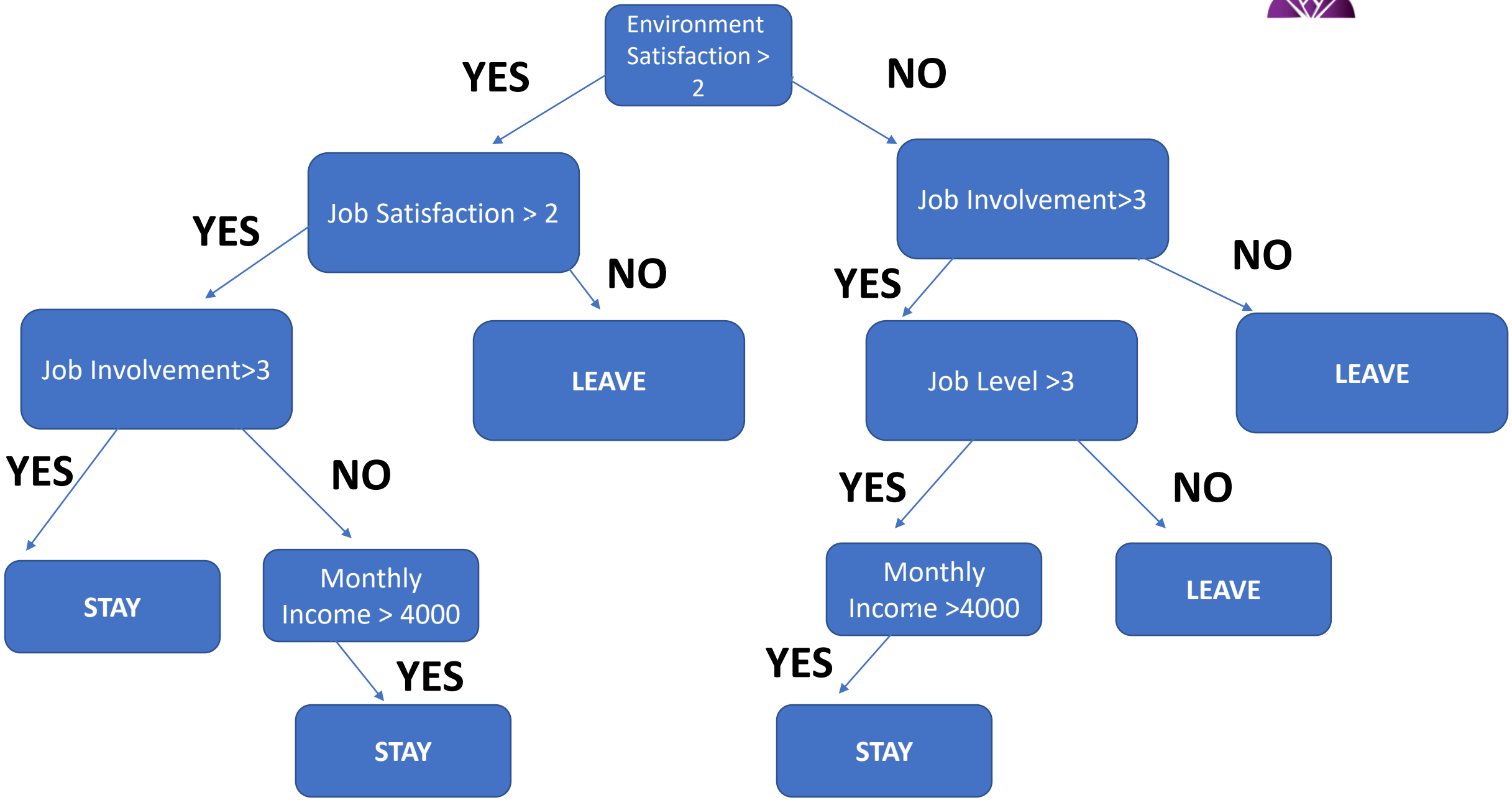




Split on Department

Samples: 1470
R & D: 961 (0.65)
Sales: 446 (0.30)
Human Resources: 63 (0.04)

Overfitting in Decision Tree



Pruning

DATA FOLKZ®
#CATAPULT DATA LEADERS

Reference:
https://pdfs.semanticscholar.org/025b/8c109c38dc115024e97eb0ede5ea873fffdb.pdf

Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances. Decision trees are the most susceptible out of all the machine learning algorithms to overfitting and effective pruning can reduce this likelihood.

Pruning are of two types:
1. Forward Pruning
2. Backward pruning

Forward Pruning

DATA FOLKZ®
#CATAPULT DATA LEADERS

To prevent overfitting we stop the tree-building process early, before it produces leaves with very small samples.

An early stopping criteria can be induced

This heuristic is known as forward pruning but is also sometimes known as pre-pruning decision trees.

Forward Pruning Strategies

DATA FOLKZ®
#CATAPULT DATA LEADERS

There are a range of simple pre-pruning strategies. We can stop creating subtrees

- when the number of instances in a partition falls below a threshold
- when the information gain (or whatever other feature selection metric is being used) measured at a node is not deemed to be sufficient to make partitioning the data worthwhile
- when the depth of the tree goes beyond a predefined limit.

Backward Pruning

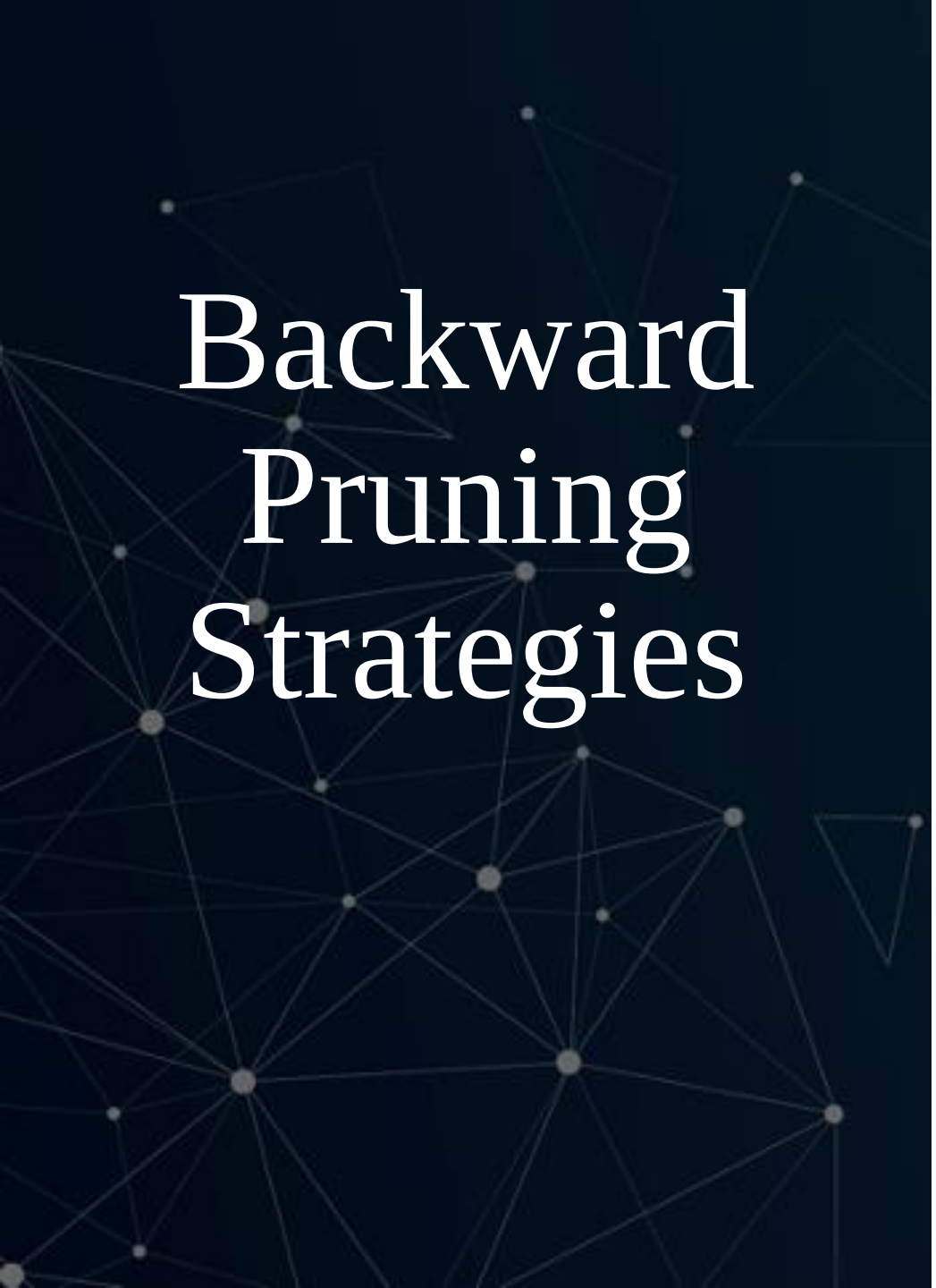
DATA FOLKZ®
#CATAPULT DATA LEADERS

Backward pruning is also known as Post-pruning.

In this, first generate the decision tree and then remove non-significant branches.

Post-pruning a decision tree implies that we begin by generating the (complete) tree and then adjust it with the aim of improving the accuracy on unseen instances.

Post-pruning relies on a criteria that can distinguish between subtrees that model relevant aspects of the data and subtrees that



Backward Pruning Strategies

DATA FOLKZ®
#CATAPULT DATA LEADERS

Simple threshold on the number of instances at a node in the tree

Reduced error pruning is a popular version of post-pruning based on error rates. Here, a decision tree is built to completion and then the tree is searched in an iterative, bottom-up, left-to-right manner for subtrees that can be pruned.

If the error rate in the validation dataset at the subtree root node is less than or equal to the combined error rate at the leaves, the subtree is pruned.

How Pruning Works

DATA FOLKZ®
#CATAPULT DATA LEADERS

To avoid the problem of overfitting we use pruning.

In order to prune a decision tree we use GridSearch to find the optimal Hyper Parameters and then use those parameters to build our model with best scores.

Hyperparameter Tuning

Hyperparameters are the parameters whose values are set before the model building begins, hyperparameters are crucial to the performance, speed, and quality of the machine learning models.

Some examples of common hyperparameters of decision tree include:

- **criterion** : It defines the function to measure the quality of a split. E.g. Entropy, IG
- **max_features**: It defines the no. of features to consider when looking for the best split.
- **max_depth**: The max_depth parameter denotes maximum depth of the tree. It can take any integer value or None
- **min_samples_split**: This tells above the minimum no. of samples reqd. to split an internal node.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node.

Grid Search

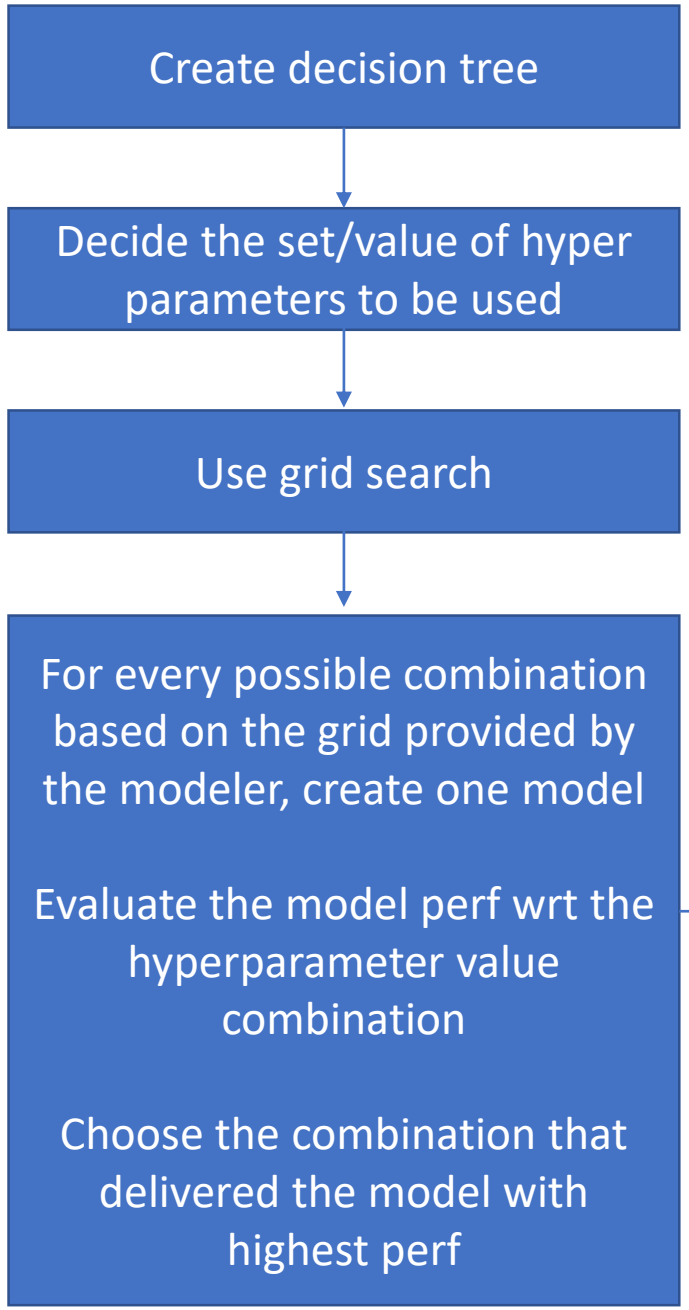
Grid search is way to perform hyperparameter optimization.

It takes a dictionary i.e different values of all of the different hyperparameters that we want to test, and then feeds all of the different combinations through the algorithm for us and then reports back to us with the optimal parameters.

max_features = [1, 3, 10,12,14]
Criterion = ["gini", "entropy"]
min_samples_split =[4,6,10,12,14]
min_samples_leaf =[2,4,5,7,8]

So, in above case Grid search will perform on every combination of the parameters.

i.e. it will perform (3*2*3*3) = 54 iterations and will give us the best parameter values.



Decision tree with default HP values	1. root Node,2.X1 decision,3.Entropy,4.Gini
Optimize it’s hypeparameter value	Use GS and create many more DT to identify the best one


DATA FOLKZ®
#CATAPULT*DATA LEADERS
Thank You !