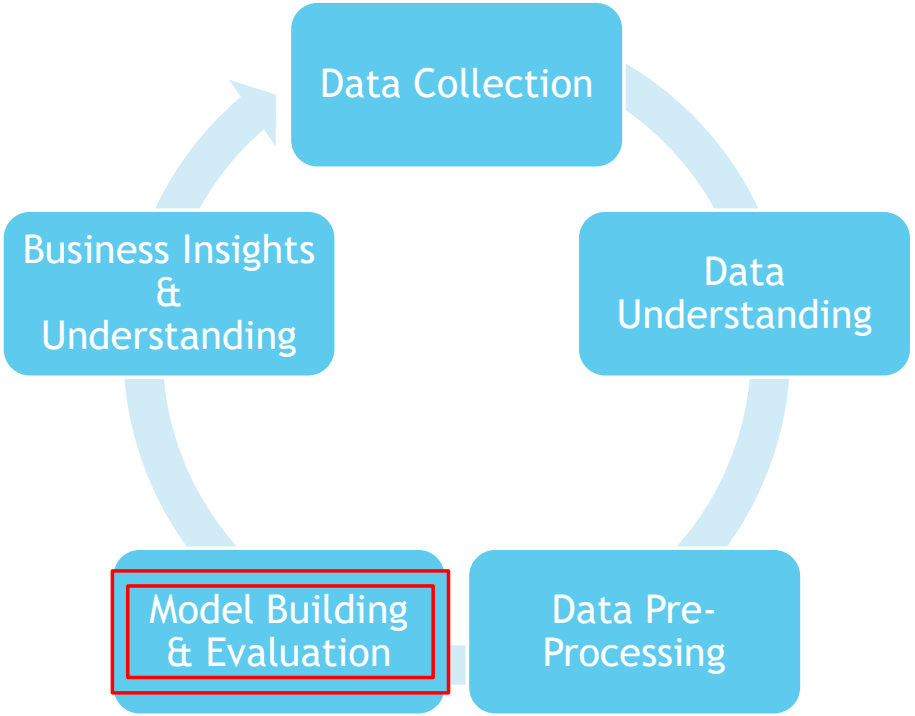
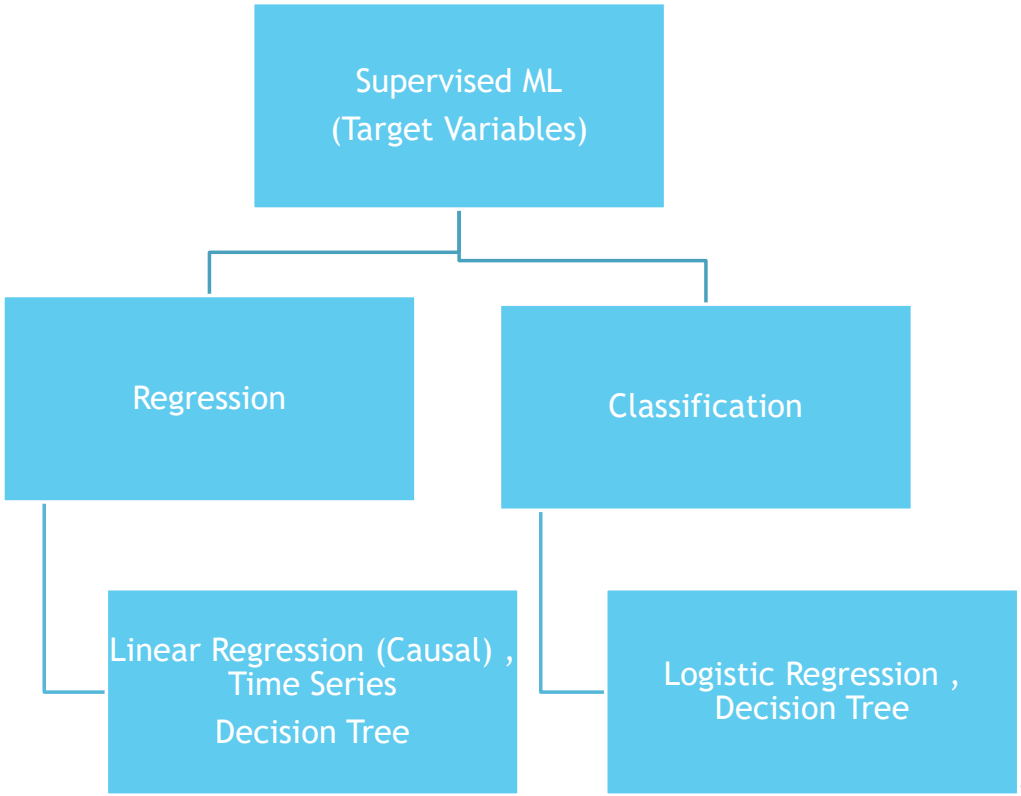
Decision Tree

DataDNA

Typical Data Science Cycle



Machine Learning



Ideation

- Categorical Target Variable

Binary Variable (*Two categories / Two Classes / Two Labels)*

e.g. Yes/No, Churn/No-Churn, Spam/Non-Spam

Whether he will attend class tomorrow or not?

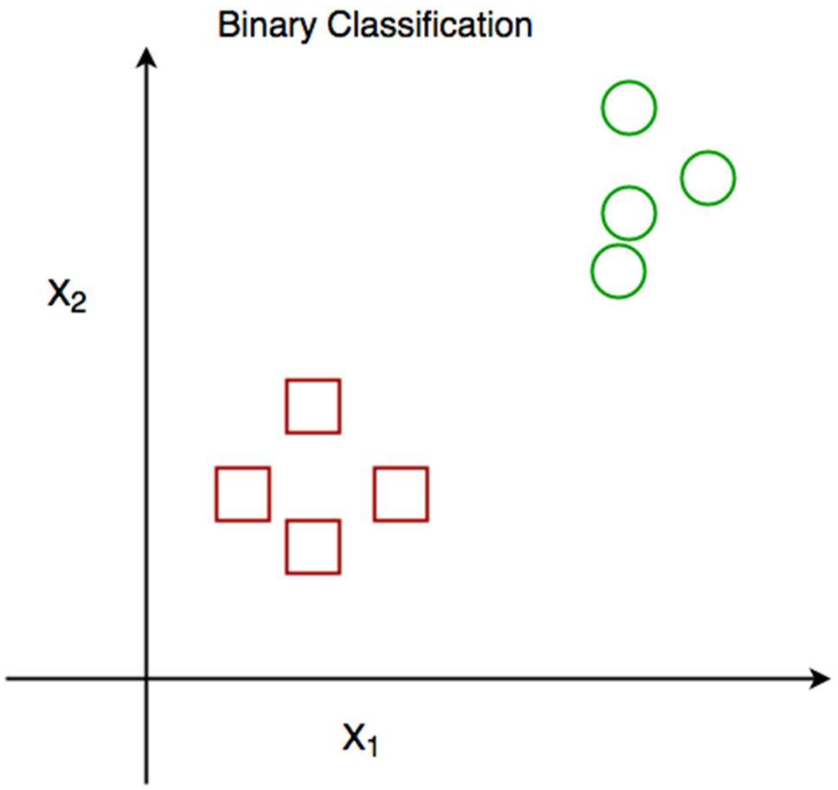
Whether he will pass exam or not?

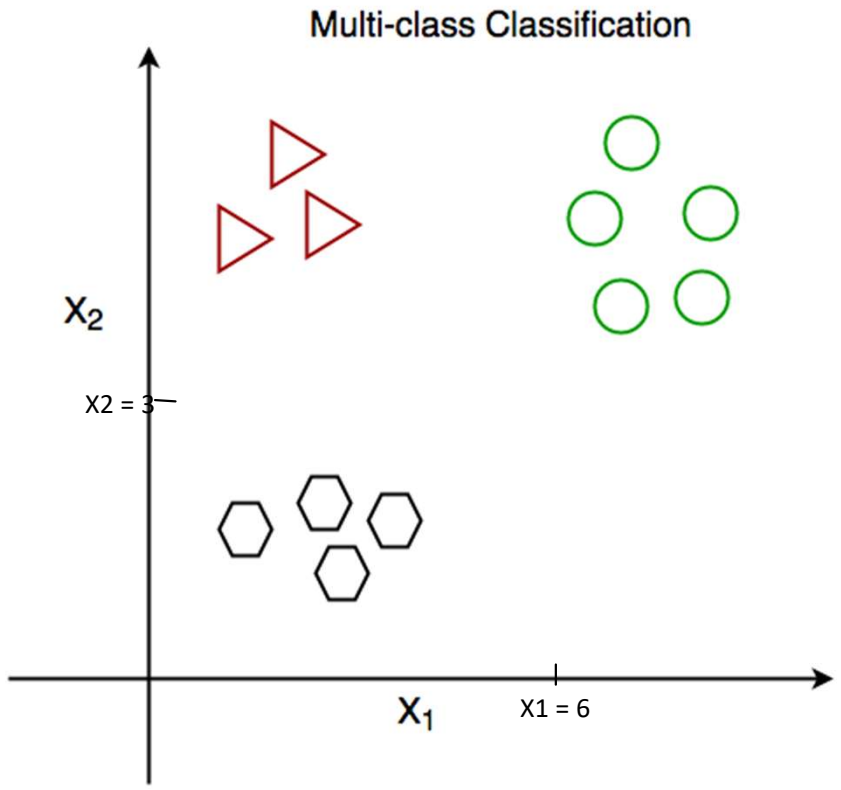
Multi-class Variable (*More than two categories / Classes / Labels)*

e.g.
Size of an object in an image (small / medium / large)
Type of species in IRIS data set (virginica / setosa / versicolor)
Grouping related articles / books together (Google News)
Type of banking customer (low-affluent/ medium affluent / high net worth / super affluent)

Ideation

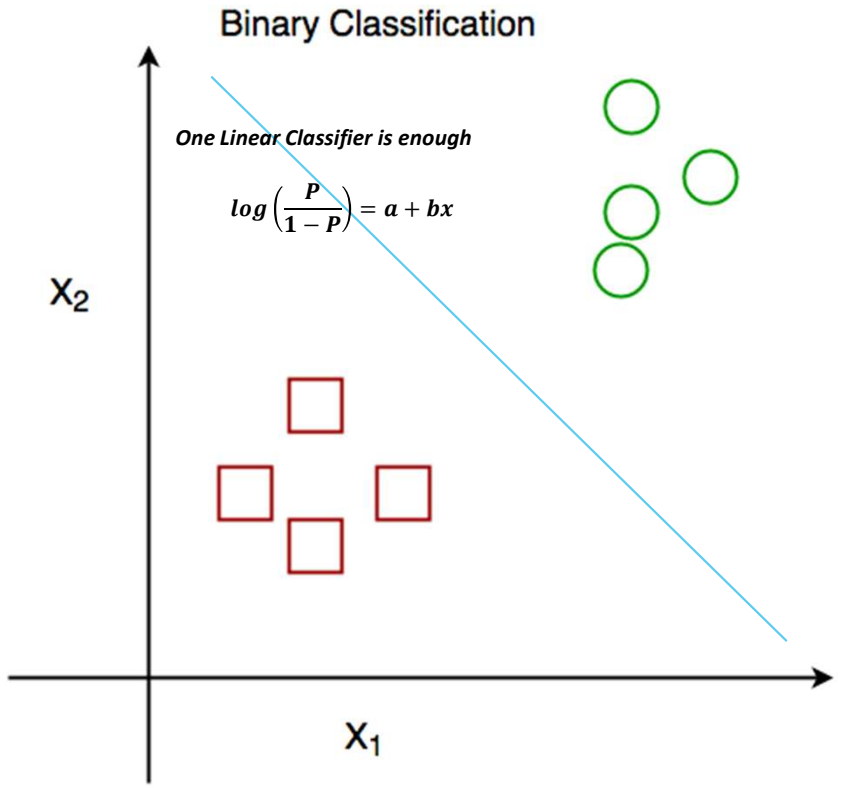
What is the effective way to classify data into each category for both cases?

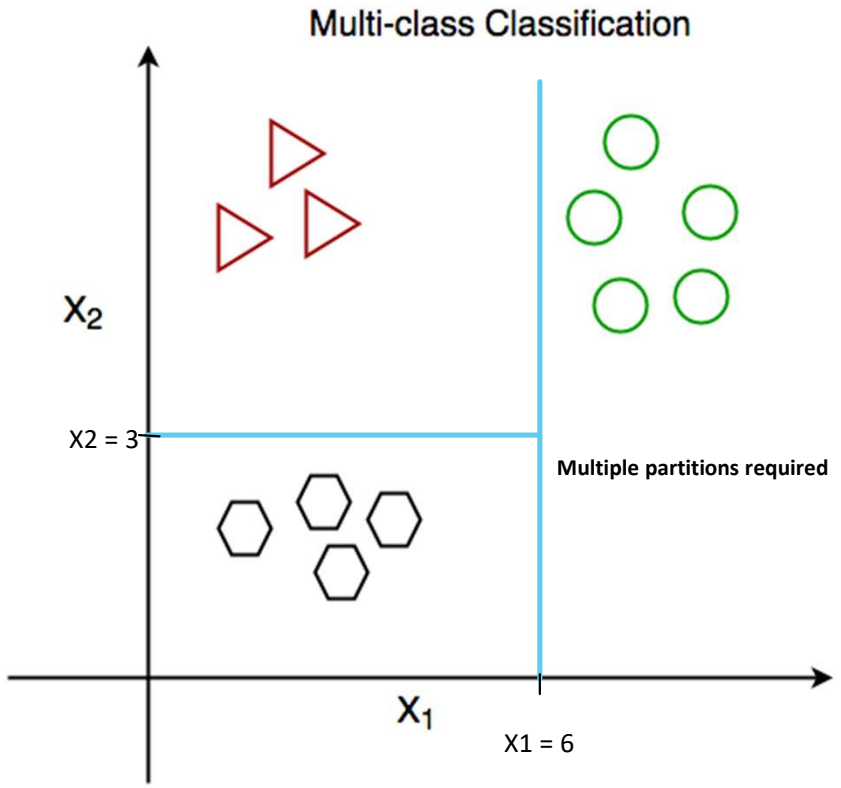




Ideation

What is the effective way to classify data into each category for both cases?





DataDNA

Ideation

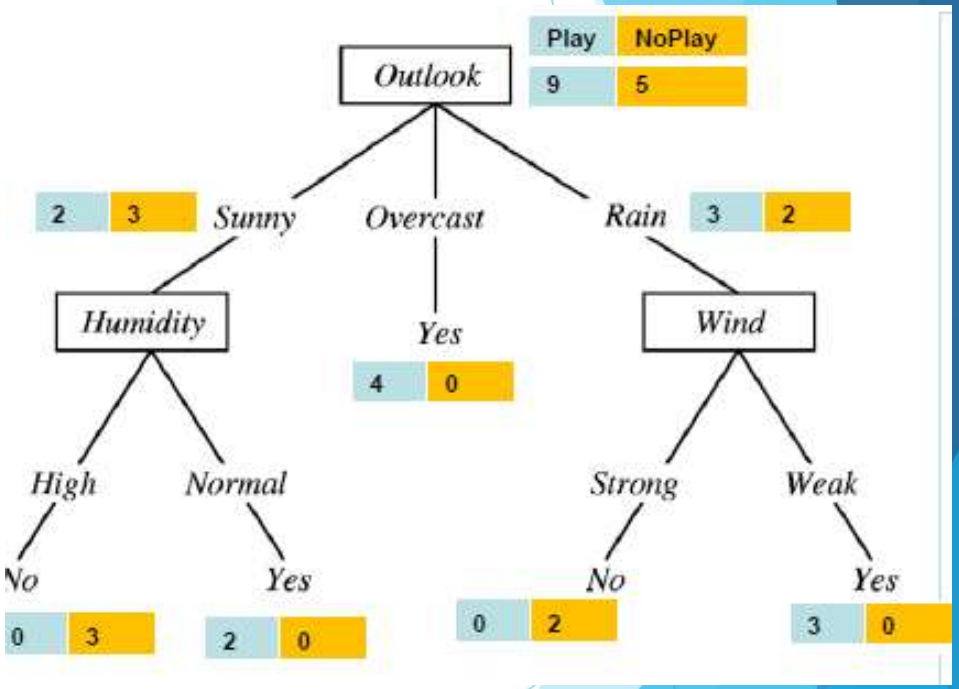
Day	Outlook	humidity	Wind	PlayCricket
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	yes
4	Rain	High	Weak	yes
5	Rain	Normal	Weak	yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	yes
10	Rain	Normal	Weak	yes
11	Sunny	Normal	Strong	yes
12	Overcast	High	Strong	yes
13	Overcast	Normal	Weak	yes
14	Rain	High	Strong	No

Hard to visualize through charts so let's create some rules to classify

DataDNA

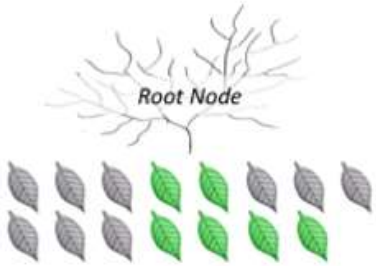
Ideation

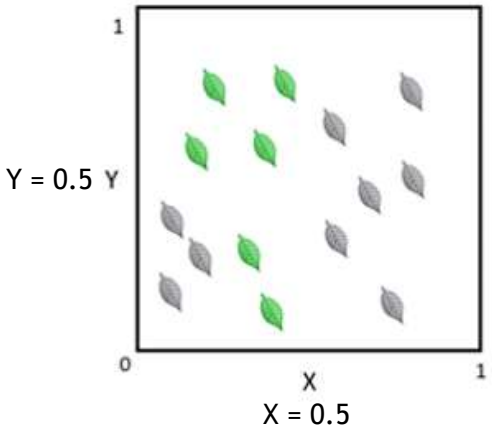
Day	Outlook	humidity	Wind	PlayCricket
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	yes
4	Rain	High	Weak	yes
5	Rain	Normal	Weak	yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	yes
10	Rain	Normal	Weak	yes
11	Sunny	Normal	Strong	yes
12	Overcast	High	Strong	yes
13	Overcast	Normal	Weak	yes
14	Rain	High	Strong	No



DataDNA

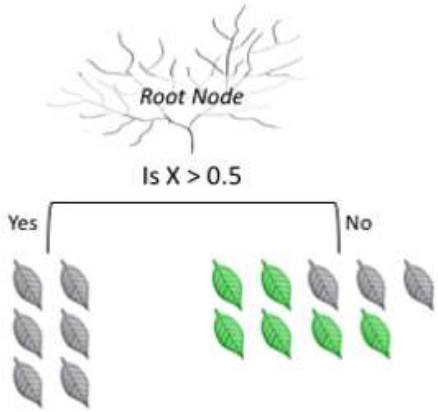
Ideation

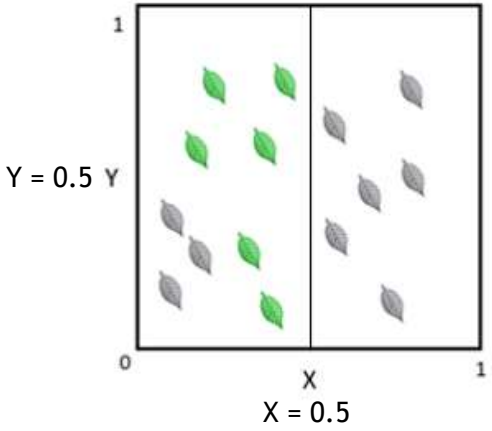




DataDNA

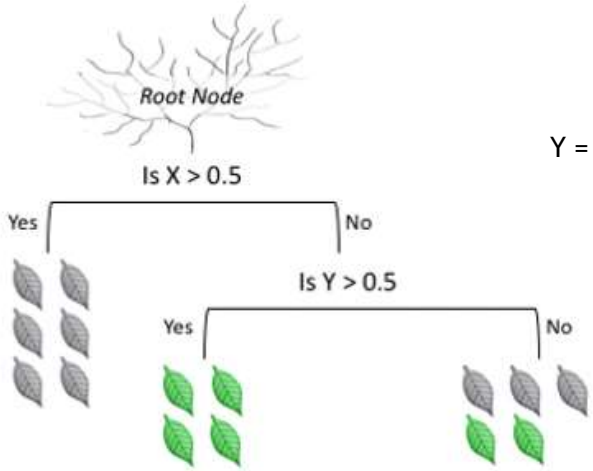
Ideation

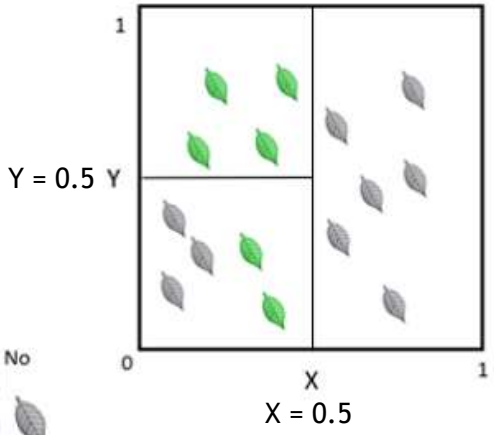




DataDNA

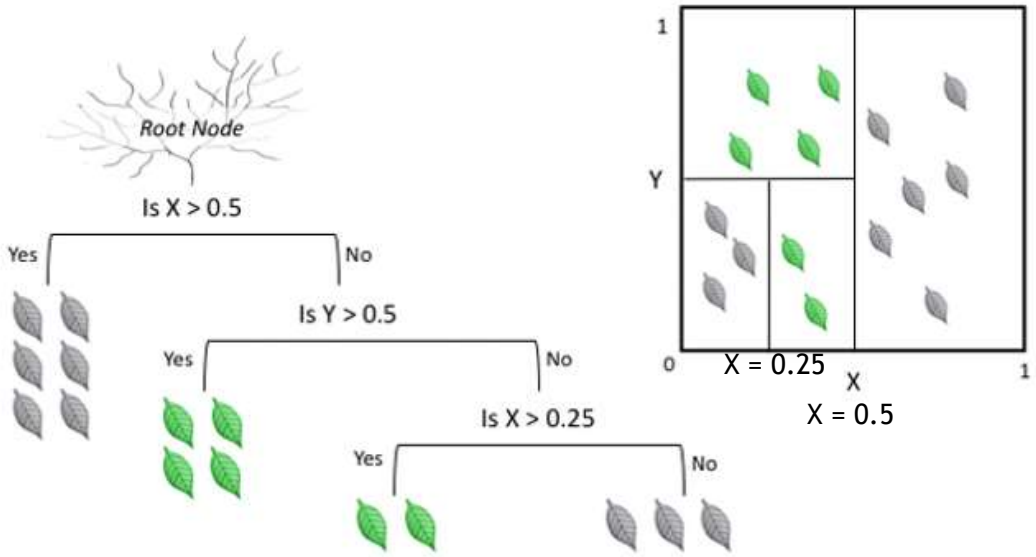
Ideation





DataDNA

Ideation



Introduction

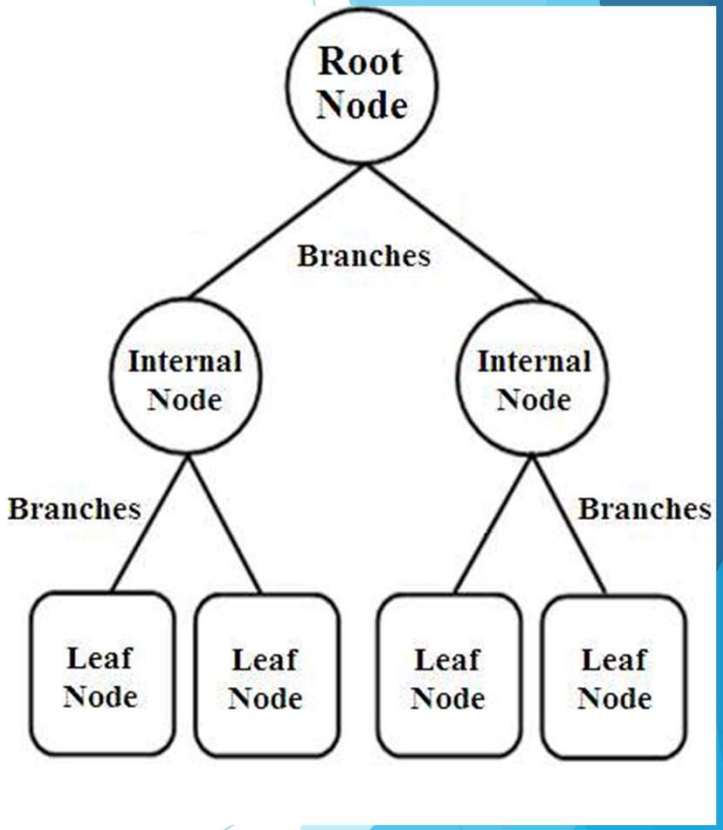
- A Decision tree is a flowchart like tree structure, which contains **nodes** and **branches**.

- Internal node denotes a test on an attribute.
- First internal node is called **root node**. Remaining internal nodes

are called **decision node**. These nodes performs splitting tests.

- Each branch represents an outcome of the test

- If any node is not further partitioned then it is called **leaf node (terminal node)**. Each leaf node denotes the final results of a particular rule i.e. (label/class distribution)



DataDNA

Introduction

- Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods.
- Tree based methods empower predictive models with high accuracy, stability and ease of interpretation.
- Unlike linear models, they map **non-linear relationships** quite well.
- They are adaptable at solving any kind of problem at hand (**classification or regression**).
- Methods like decision trees, random forest, gradient boosting are being popularly used in all kinds of data science problems.

Common terms used with 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.

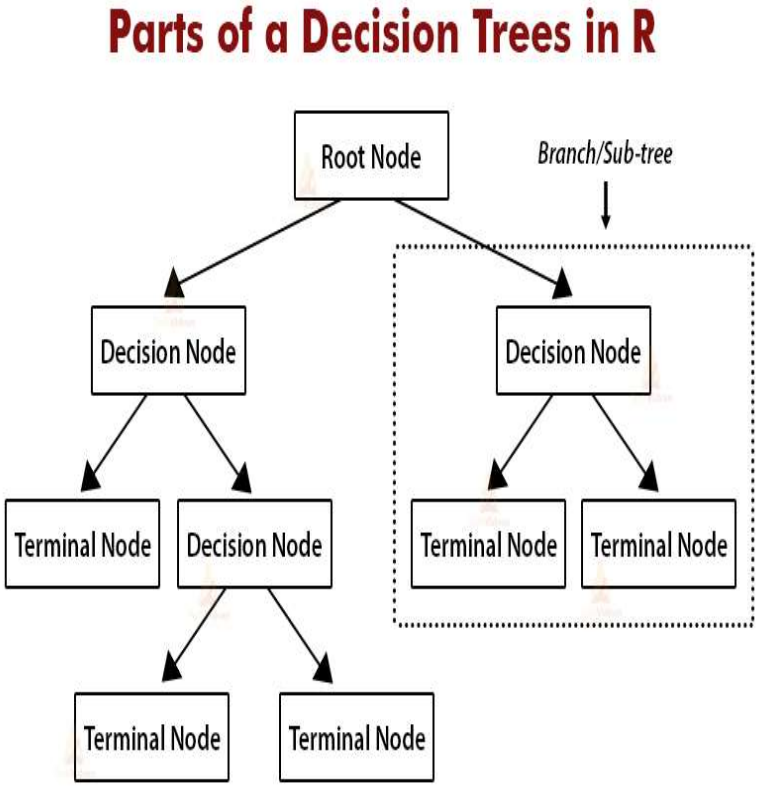
Decision Node: When a sub-node splits into further sub-nodes, then it is called decision node.

Leaf/ Terminal Node: Nodes do not split is called Leaf or Terminal node.

Branch / Sub-Tree: A sub section of entire tree is called branch or sub-tree.

Parent and Child Node: A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

Pruning: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.



Construction of decision tree

Decision tree works on the principle of **recursive partitioning** where Recursive partitioning strives to correctly classify members of the population by **splitting it into sub-populations** based on more than one independent variables. The process is termed recursive because each sub-population may in turn be split an indefinite number of times until the splitting process terminates after a particular **stopping criterion** is reached.

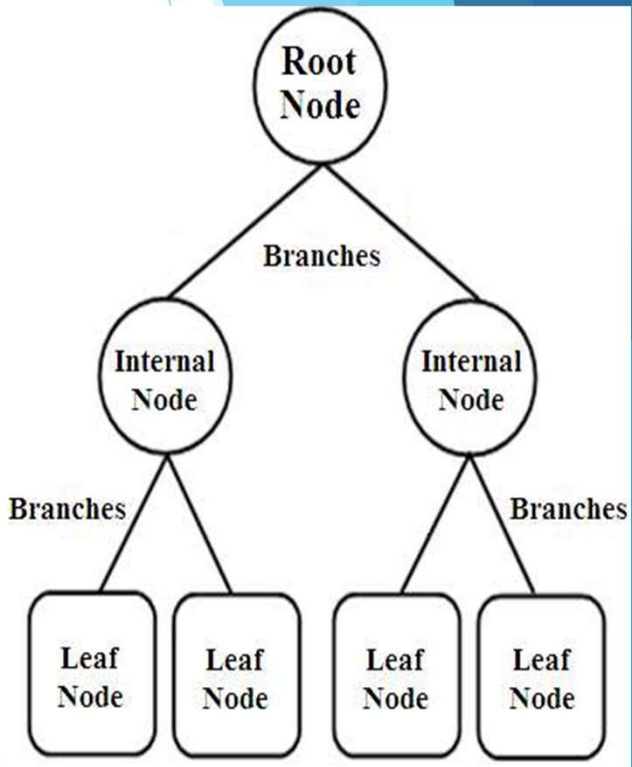
Process :-

1. Place the **best attribute** (most **informative**) of the dataset at the root of the tree. Order to placing attributes at root node of the tree is done by using some statistical approach.

2. Split the Data set into subsets. Subsets should be made in such a way that each subset contains **homogeneous** values for target variable.

3. Repeat step 1 and step 2 on each subsequent decision node until you find leaf nodes in all the branches of the tree.

4. Process **terminates** after some predefined criteria achieved its threshold



Construction of decision tree

Identify target variable

Decision Tree works both for classification problem as well as regression problem.

Decision Tree Classifier : Target variable is categorical

e.g.
What is the outcome of cricket match (win /lose)
Whether car will breakdown (yes / no)

Decision Tree Regressor : Target variable is numerical
e.g.
Total income of customer
Total revenue from a single customer
Total likes on a facebook post given information about fb user, post characteristics

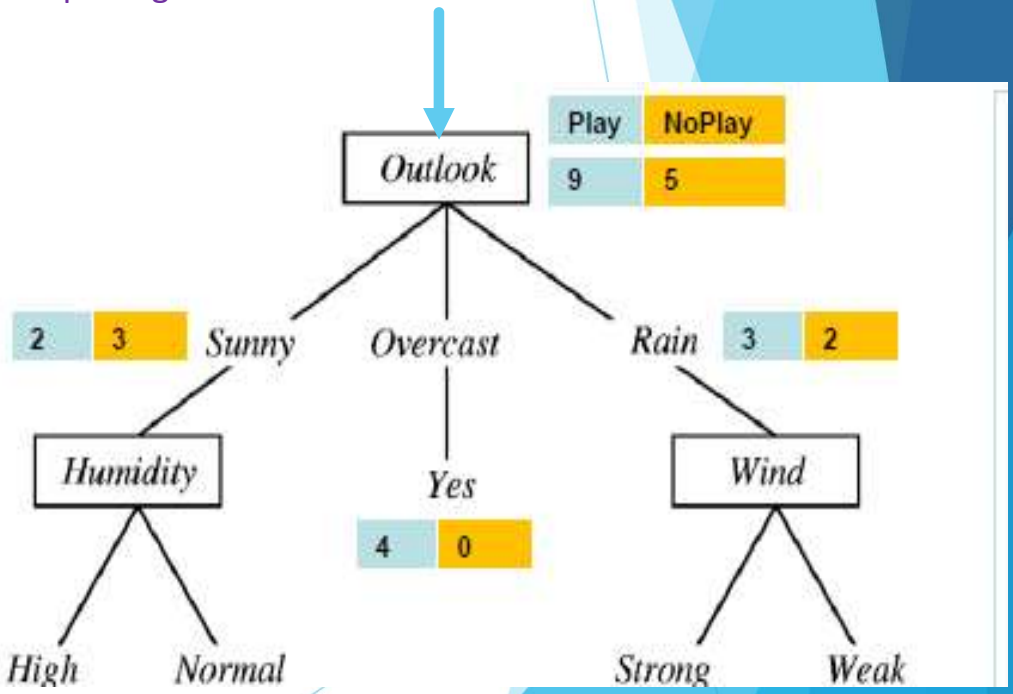
Construction of decision tree

Place the best attribute at root node /internal node

We do not choose random independent variable from dataset to perform splitting

Day	Outlook	humidity	Wind	PlayCricket
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	yes
4	Rain	High	Weak	yes
5	Rain	Normal	Weak	yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	yes
10	Rain	Normal	Weak	yes
11	Sunny	Normal	Strong	yes
12	Overcast	High	Strong	yes
13	Overcast	Normal	Weak	yes
14	Rain	High	Strong	No

Why Did we choose variable "Outlook" at root node



Construction of decision tree

Place the best attribute at root node /internal node

Objectives :

1. To achieve homogeneous data set after splitting

2. We select attributes which are most informative

Construction of decision tree

Place the best attribute at root node /internal node

Methods to select best attribute for splitting when target variable is categorical :

1. Information gain

2. Gini Impurity (Gini index based method)

Construction of decision tree

Place the best attribute at root node /internal node

Information gain

$$Information\,Gain = 1 - Entropy$$

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

Entropy is used for calculating the purity of a node.

Lower the value of entropy, higher is the purity of the node.

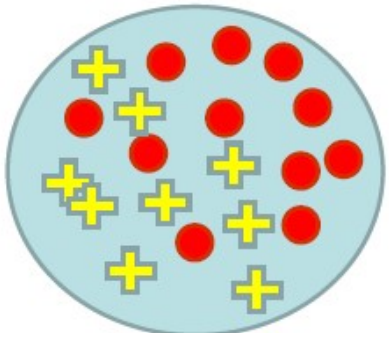
The entropy of a homogeneous node is zero. Since we subtract entropy from 1, the Information Gain is higher for the purer nodes with a maximum value of 1

Construction of decision tree

Place the best attribute at root node /internal node

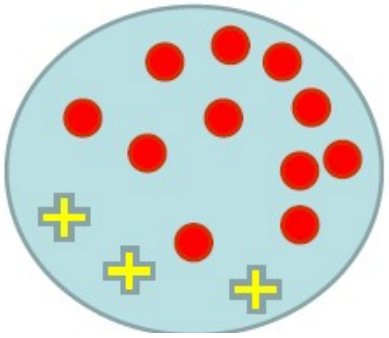
Entropy: A measure of homogeneity

More confusion



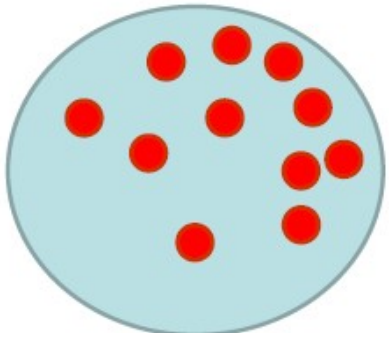
Entropy: $-1 * \left(\frac{11}{20} log \frac{11}{20} + \frac{9}{20} log \frac{9}{20}\right)$
$= 0.47 + .52 = 0.99$

Less confusion



Entropy: $-1 * \left(\frac{11}{14} log \frac{11}{14} + \frac{3}{14} log \frac{3}{14}\right)$
$= 0.27 + .48 = 0.75$

No confusion



Entropy: $-1 * \left(\frac{11}{11} log \frac{11}{11}\right)$
$= 0$

Construction of decision tree

Place the best attribute at root node /internal node

Steps to decide best attribute using Information gain

1. For each split, individually calculate the entropy of each child node

2. Calculate the entropy of each split as the weighted average entropy of child nodes

3. Select the split with the lowest entropy or highest information gain

Information gain = (Entropy of system before split – Entropy of system after split)

4. Until you achieve homogeneous nodes, repeat steps 1-3

Day	Outlook	humidity	Wind	PlayCricket
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	yes
4	Rain	High	Weak	yes
5	Rain	Normal	Weak	yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	yes
10	Rain	Normal	Weak	yes
11	Sunny	Normal	Strong	yes
12	Overcast	High	Strong	yes
13	Overcast	Normal	Weak	yes
14	Rain	High	Strong	No

Construction of decision tree

Place the best attribute at root node /internal node

Information gain in Case 'Outlook'

Entropy = 0.9402

Yes = 9 No = 5

Outlook

Sunny

Rain

Overcast

Yes = 2 No = 3

Entropy = 0.97

Yes = 3 No = 2

Entropy = 0.97

Yes = 4 No = 0

Entropy = 0

Entropy of Total Split = (0.97*5+0.97*5+0*5)/14 = 0.69

Information gain = 0.9402 – 0.69 = 0.2473

Information gain in Case 'humidity'

Entropy = 0.9402

Yes = 9 No = 5

humidity

SPLIT USING 'HUMIDITY' IS BETTER

High

Normal

Yes = 3 No = 4

Entropy = 0.98

Yes = 6 No = 1

Entropy = 0.59

Entropy of Total Split = (0.98*7+0.59*7)/14 = 0.5607

Information gain = 0.9402 – 0.56 = 0.379

Construction of decision tree

Place the best attribute at root node /internal node

Gini Impurity

$$Gini\ Impurity = 1 - Gini$$

$$Gini = \sum_{i=1}^{n} p_i^2$$

$$Gini\ Impurity = 1 - \sum_{i=1}^{n} p_i^2$$

Gini index do not need logarithm hence it is simpler to calculate w.r.t. information gain

Gini Impurity is used for calculating the purity of a node.

Lower the Gini Impurity, higher is the homogeneity of the node.

The Gini Impurity of a pure node is zero.

Construction of decision tree

Place the best attribute at root node /internal node

Steps to decide best attribute using gini impurity

1. Similar to what we did in information gain. For each split, individually calculate the Gini Impurity of each child node

2. Calculate the Gini Impurity of each split as the weighted average Gini Impurity of child nodes

3. Select the split with the lowest value of Gini Impurity

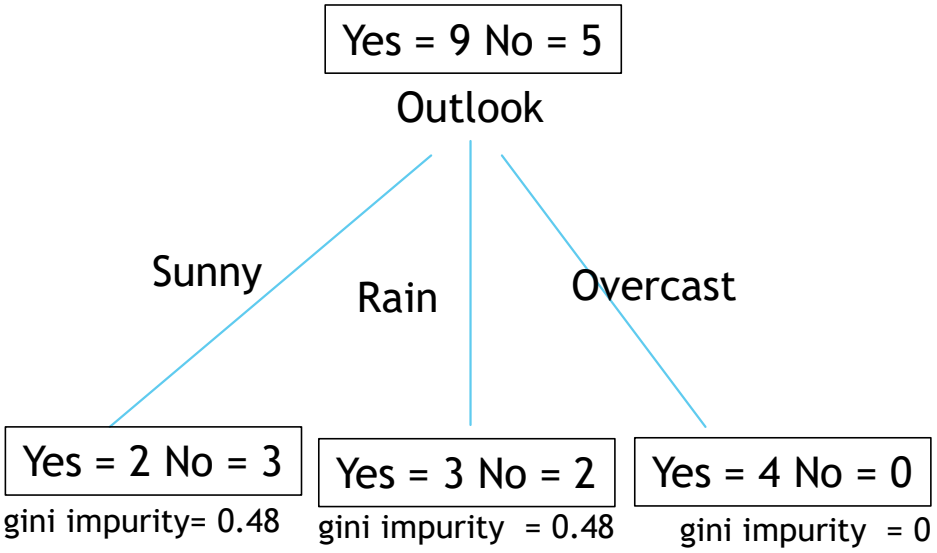
4. Until you achieve homogeneous nodes, repeat steps 1-3

Day	Outlook	humidity	Wind	PlayCricket
1	Sunny	High	Weak	No
2	Sunny	High	Strong	No
3	Overcast	High	Weak	yes
4	Rain	High	Weak	yes
5	Rain	Normal	Weak	yes
6	Rain	Normal	Strong	No
7	Overcast	Normal	Strong	yes
8	Sunny	High	Weak	No
9	Sunny	Normal	Weak	yes
10	Rain	Normal	Weak	yes
11	Sunny	Normal	Strong	yes
12	Overcast	High	Strong	yes
13	Overcast	Normal	Weak	yes
14	Rain	High	Strong	No

Construction of decision tree

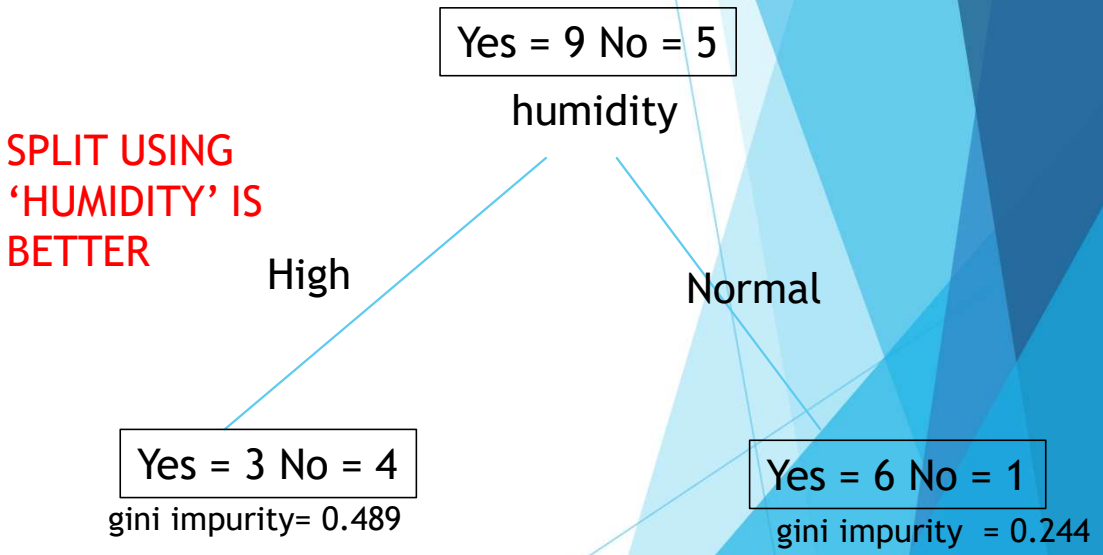
Place the best attribute at root node /internal node

Gini impurity in Case 'Outlook'



Gini Impurity of Total Split = (0.48*5+0.48*5+0*5)/14 = 0.34

Gini Impurity in Case 'humidity'



Gini Impurity of Total Split = (0.489*5+0.244*5)/14 = 0.261

Construction of decision tree

Process Termination

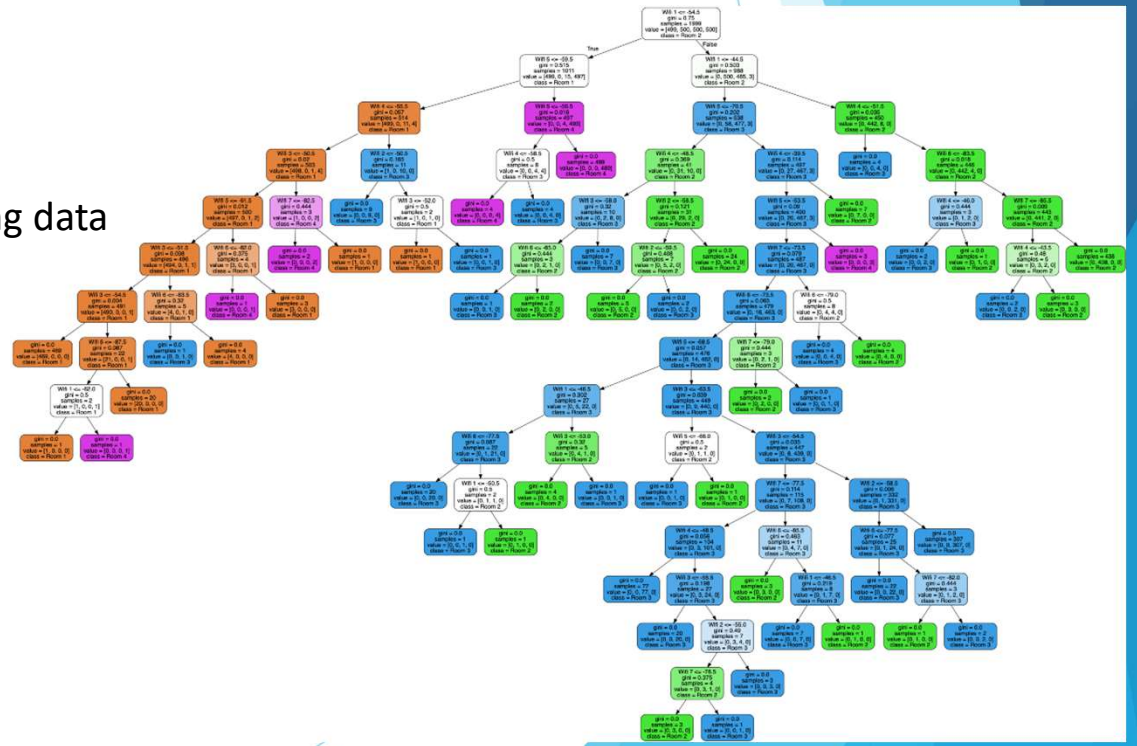
For a large data set , If we keep continue splitting to reach perfect homogeneous leaf node, then tree might get complex

Disadvantages of complex Tree

1. Chances of overfitting (Good performance on training data but poor performance on test/unseen data)

2. Very difficult to read and understand

3. Very computational heavy



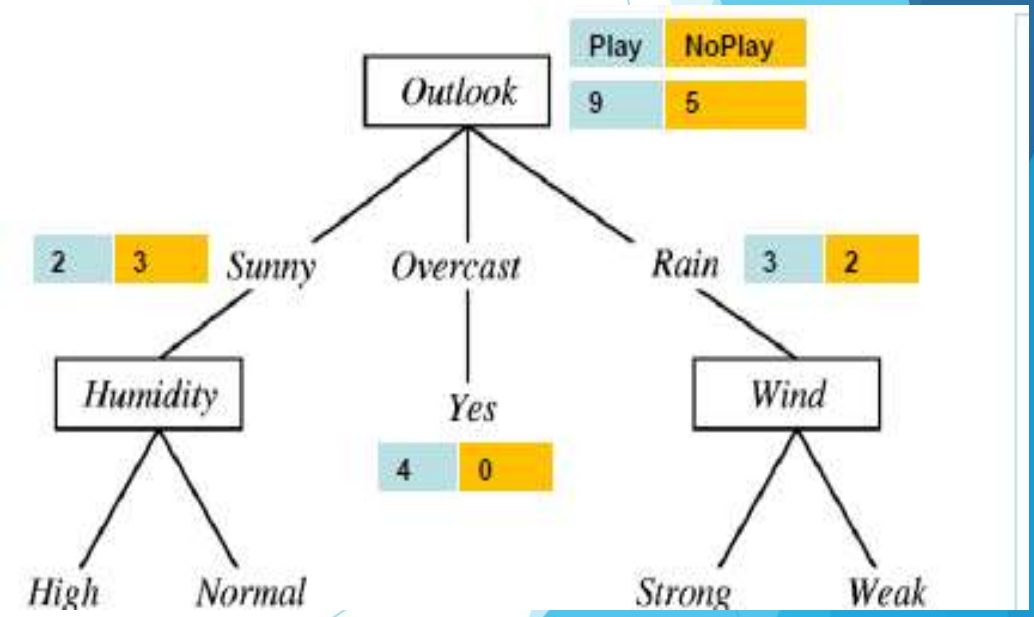
Construction of decision tree

Process Termination

To avoid complexity , we need to terminate splitting process before all of the leaf nodes reaches to perfect homogeneous.

Following criteria can be used to stop tree to reach the end :

1. We can decide the maximum depth of the tree
2. We can decide minimum number of samples in a leaf node
3. We can decide minimum number of samples in an internal node for splitting
4. We can decide maximum leaf nodes
5. We can decide minimum impurity decrease cutoff for a split.

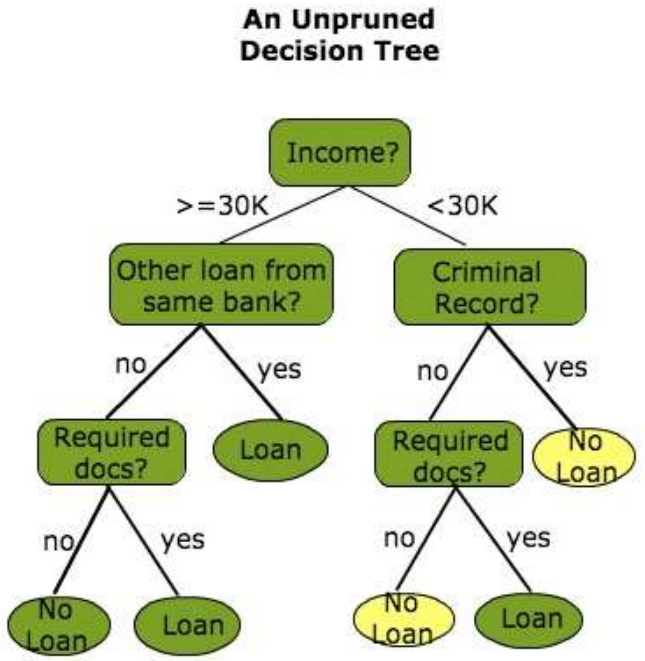


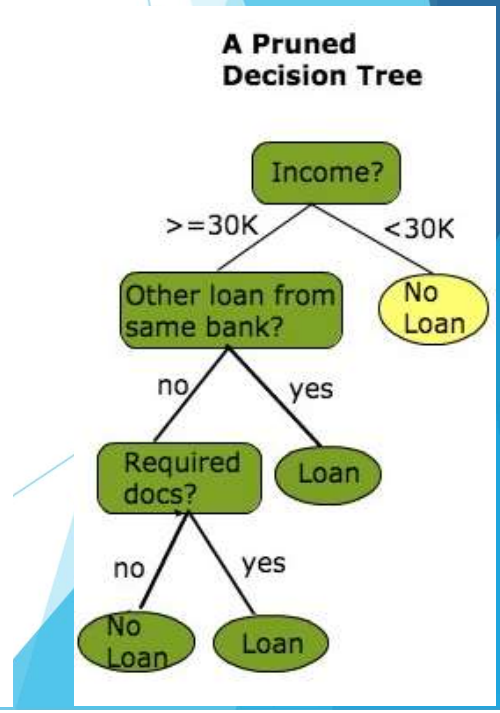
Construction of decision tree

Pruning

Stopping tree at early stage might cause overfitting and weak model.

To avoid complexity , We can also Allow the tree to overfit the data, and then post-**prune** the tree layer by layer
(cutting branches)





Construction of decision tree

Pruning

We can decide level of pruning (complexity of tree) by validating performance on validation data.

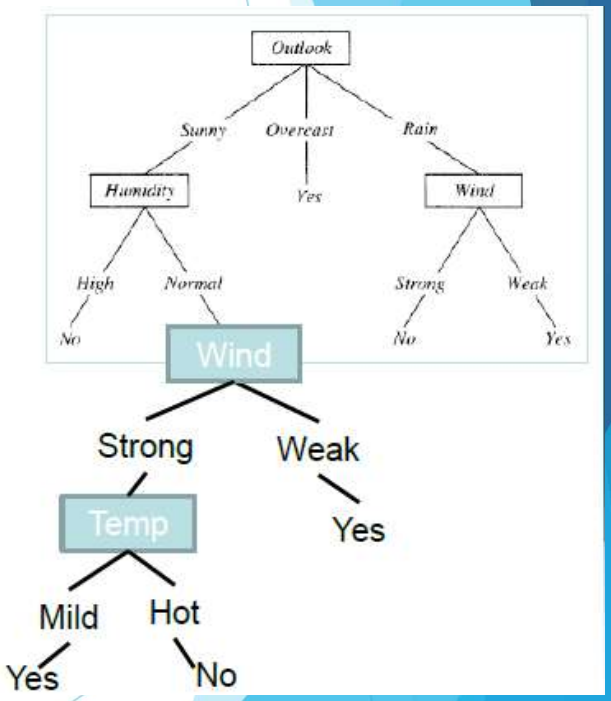
There are various methods of pruning available :

Bottom-Top Approach:

1. Minimal Cost-Complexity Pruning
2. Reduced Error Pruning
3. Minimum Error Pruning

Top-Bottom Approach:

1. pessimistic error pruning



Construction of decision tree

In case of continuous target variable

Method of choosing variable for splitting : Reduction in Variance

Reduction in Variance is a method for splitting the node used when the target variable is continuous, i.e., regression problems.

$$Variance = \frac{\sum (X - \mu)^2}{N}$$

If a node is entirely homogeneous, then the variance is zero.

Here are the steps to split a decision tree using reduction in variance:

1. For each split, individually calculate the variance of each child node
2. Calculate the variance of each split as the weighted average variance of child nodes
3. Select the split with the lowest variance
4. Perform steps 1-3 until completely homogeneous nodes are achieved

Performance Validation

Performance validation of Decision Tree

Customer	City	NoOfGamesPlayed	Channel	FavoriteGame	Actual	
1059	1	36	Favorite	Uniform	1	Training Data
1060	1	298	Favorite	Uniform	0	
1061	1	27	Uniform	Uniform	1	
1062	1	156	Favorite	Uniform	0	
1063	1	217	Favorite	Uniform	0	
1064	1	51	Favorite	Uniform	0	
1065	1	30	Favorite	Uniform	1	
1066	2	74	Favorite	Uniform	0	
1067	2	37	Favorite	Uniform	0	
1068	2	110	Favorite	Uniform	0	
1069	2	140	Favorite	Uniform	0	
1070	2	60	Favorite	Uniform	1	
1071	2	75	Favorite	Uniform	0	Test Data
1072	2	116	Favorite	Uniform	0	
1073	2	171	Favorite	Uniform	1	
1074	2	67	Favorite	Uniform	0	
1075	2	16	Favorite	Uniform	1	
1076	2	121	Uniform	Uniform	0	
1077	2	78	Favorite	Uniform	1	

Modelling on Training Data

ML Model

Apply on Test Data

Prediction

Customer	City	NoOfGamesPlayed	Channel	FavoriteGame	Actual	Prediction	
1071	2	75	Favorite	Uniform	0	1	Test Data
1072	2	116	Favorite	Uniform	0	1	
1073	2	171	Favorite	Uniform	1	0	
1074	2	67	Favorite	Uniform	0	1	
1075	2	16	Favorite	Uniform	1	1	
1076	2	121	Uniform	Uniform	0	0	
1077	2	78	Favorite	Uniform	1	1	

Model Evaluation Metrics

The various metrics used to evaluate the results of the prediction are :

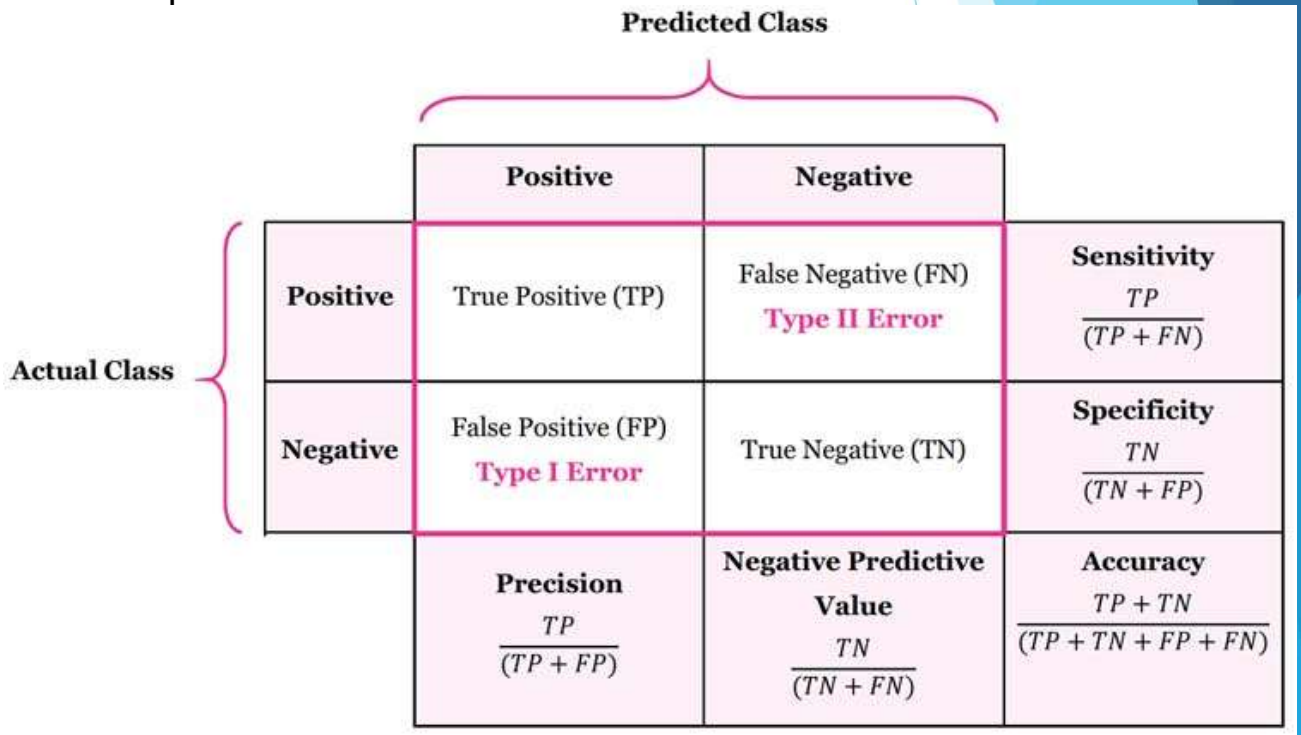
1.Accuracy

2.Precision

3.Recall

4.Area Under ROC Curve

5.MAPE/ MAD/ RMSE (For Regressors)



		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

DataDNA

Hyper parameter Tuning

criterion*{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter*{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split *int or float, default=2*
The minimum number of samples required to split an internal node:

Hyper Parameter Tuning

max_features ***int, float or {“auto”, “sqrt”, “log2”}, default=None***

The number of features to consider when looking for the best split:

max_leaf_nodes int, default=None

Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease ***float, default=0.0***

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

min_samples_leaf int or float, default=1
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression

DataDNA

Model Inference