Data Science

Decision Tree
&
Random Forest

Agenda

01 Introduction to Decision Tree

02 Finding the right split

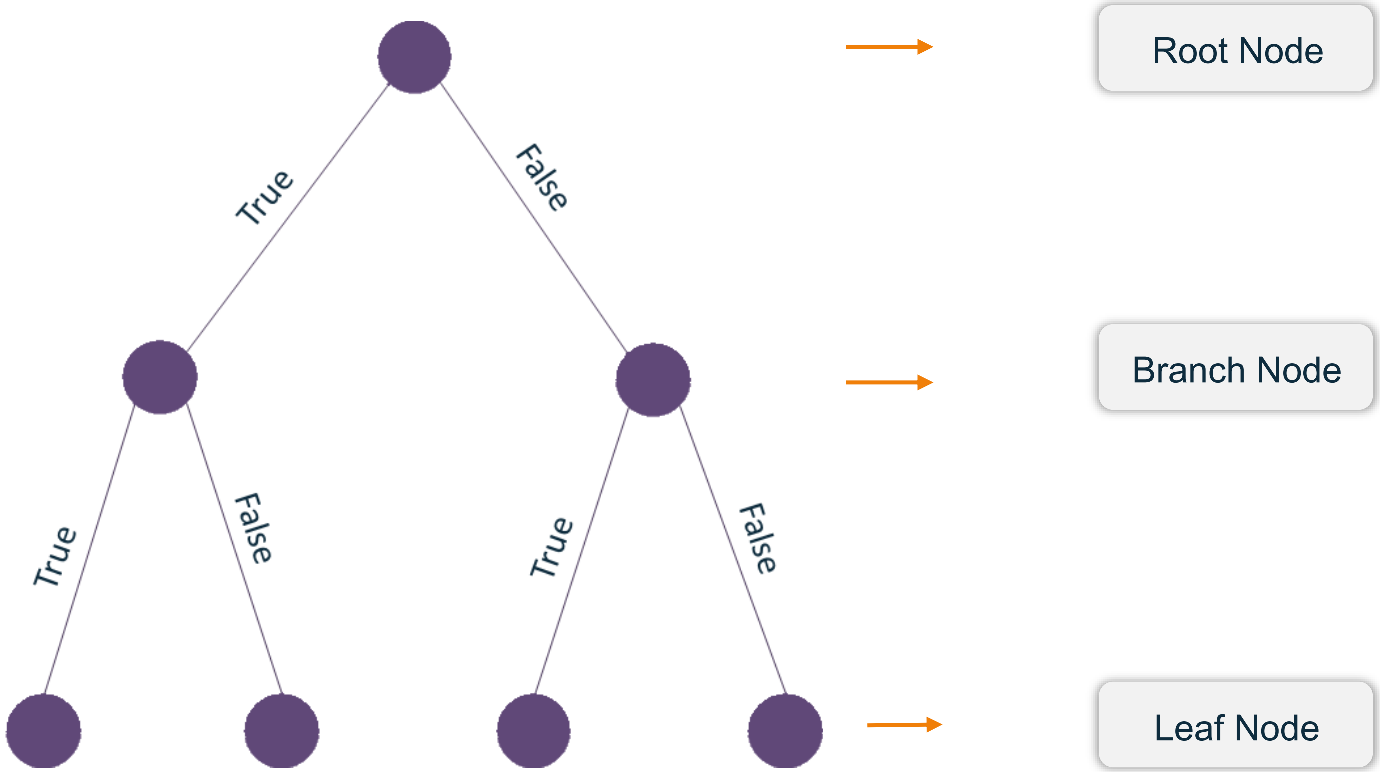
03 Ensemble of trees

04 Random Forest

Introduction to Decision Tree

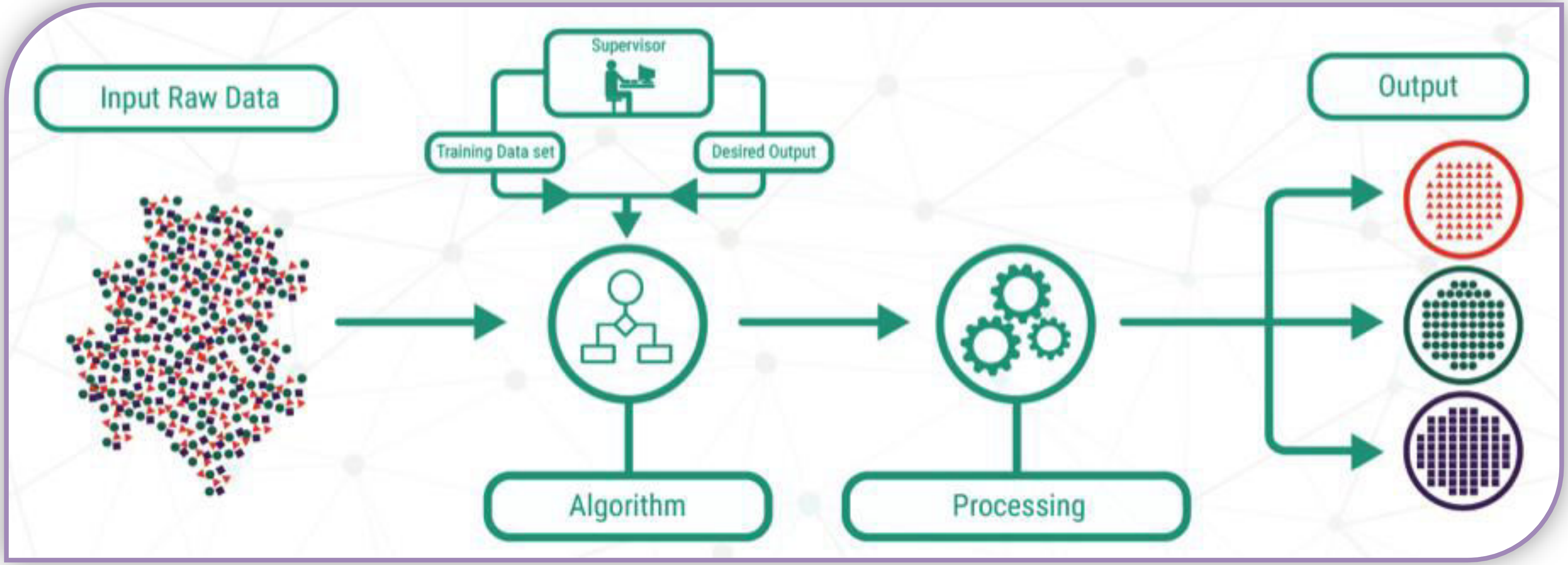
Introduction to Decision Tree

Decision tree is one of the most popular classification algorithms in use, currently, in Data Mining and Machine Learning!



Introduction to Decision Tree

Decision tree is a type of supervised learning algorithm that is mostly used in classification problems. It works for both categorical and continuous input and output variables



Terminologies

Root Node

Splitting

Decision Node

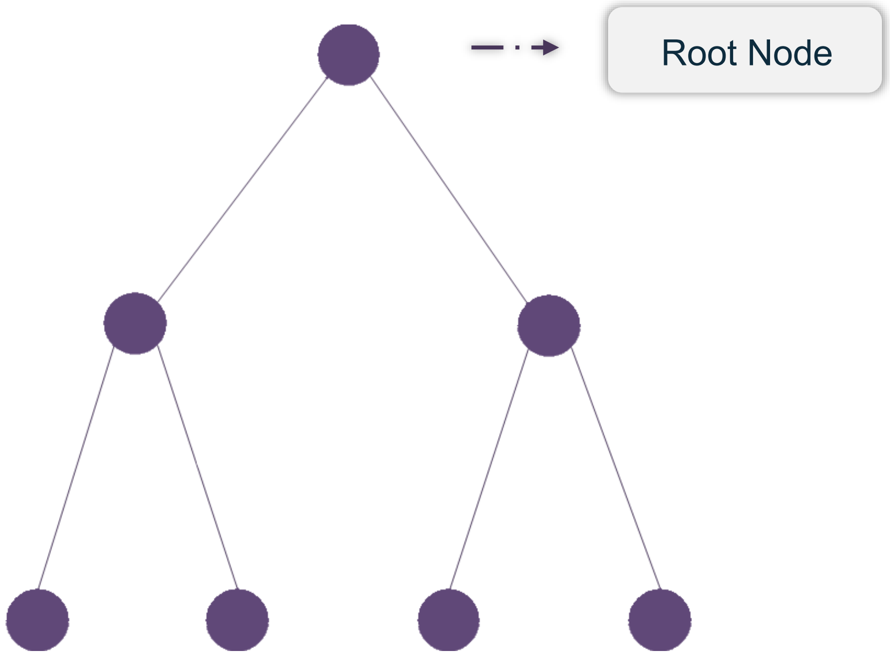
Leaf/Terminal Node

Pruning

Branch/Sub-tree

Parent and Child Node

It represents the entire population or sample, and this further gets divided into two or more homogeneous sets



Root Node

Splitting

Decision Node

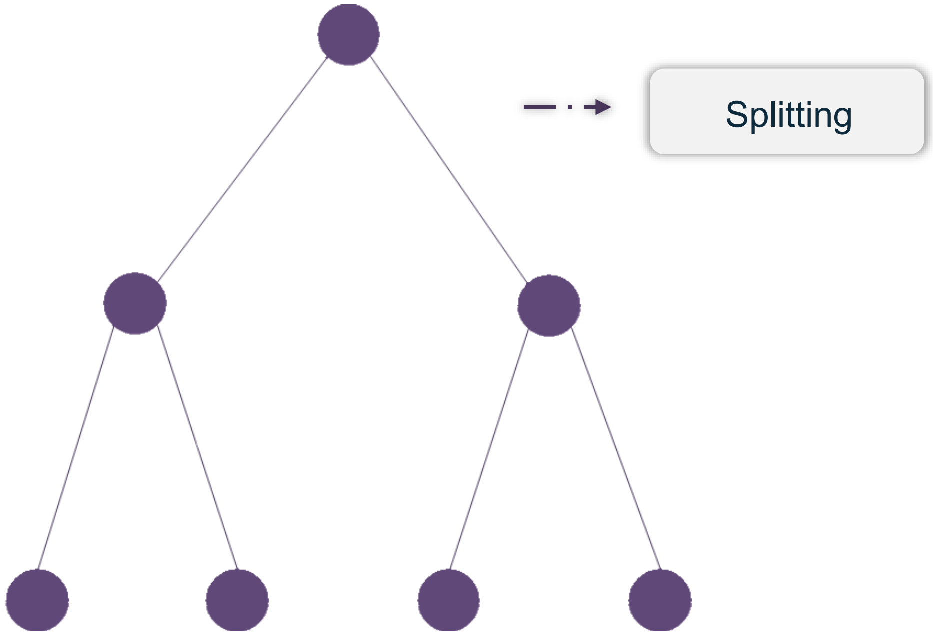
Leaf/Terminal Node

Pruning

Branch/Sub-tree

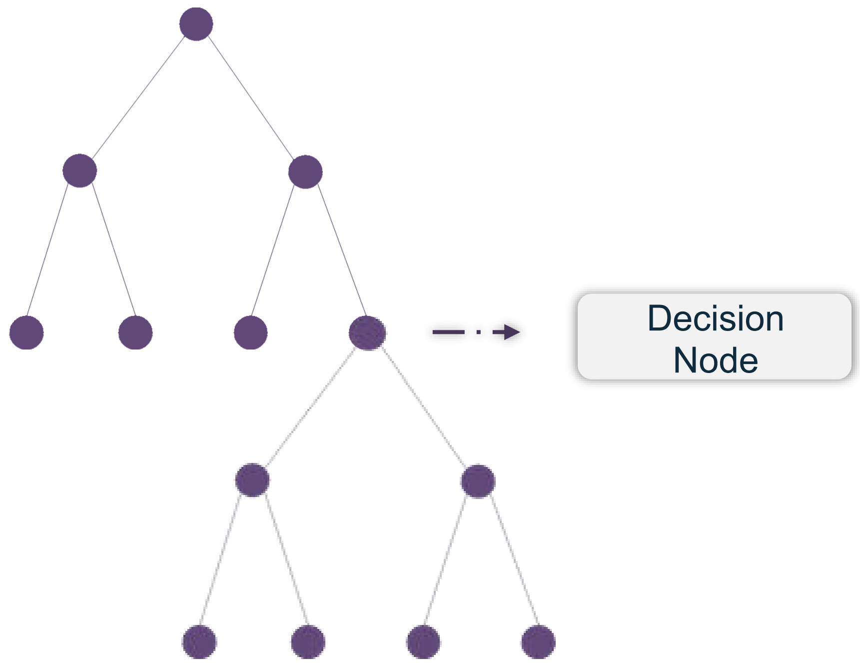
Parent and Child Node

Dividing a node into two or more sub-nodes



- Root Node
- Splitting
- **Decision Node**
- Leaf/Terminal Node
- Pruning
- Branch/Sub-tree
- Parent and Child Node

When a sub-node splits into further sub-nodes, then it is called a decision node



Root Node

Splitting

Decision Node

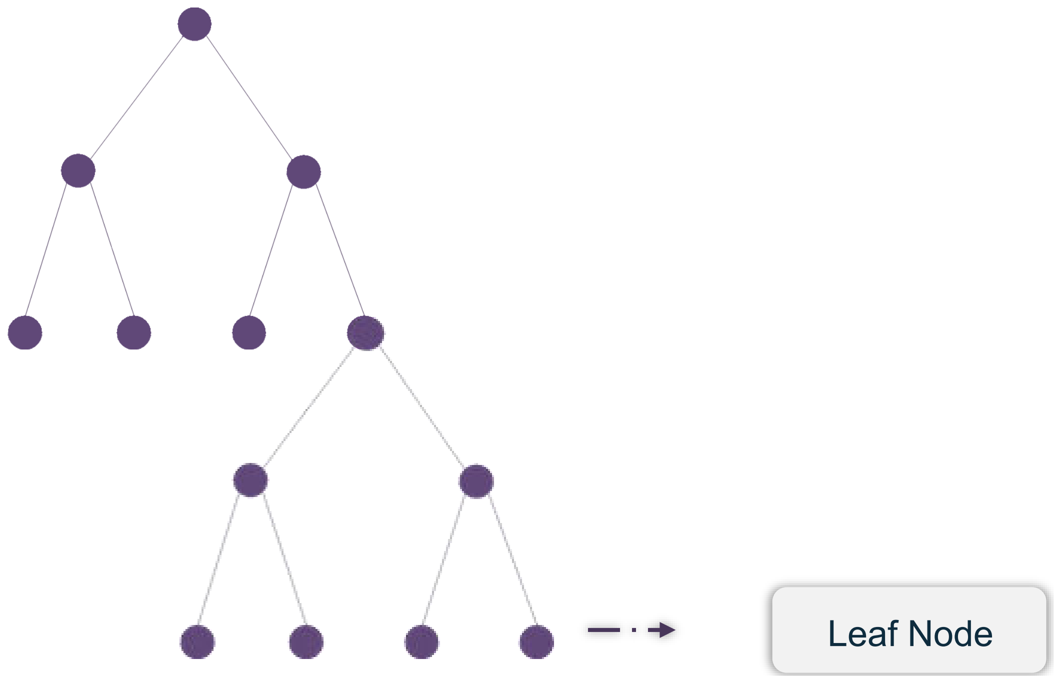
Leaf/Terminal Node

Pruning

Branch/Sub-tree

Parent and Child Node

Nodes which do not split further are called leaf or terminal nodes



Root Node

Splitting

Decision Node

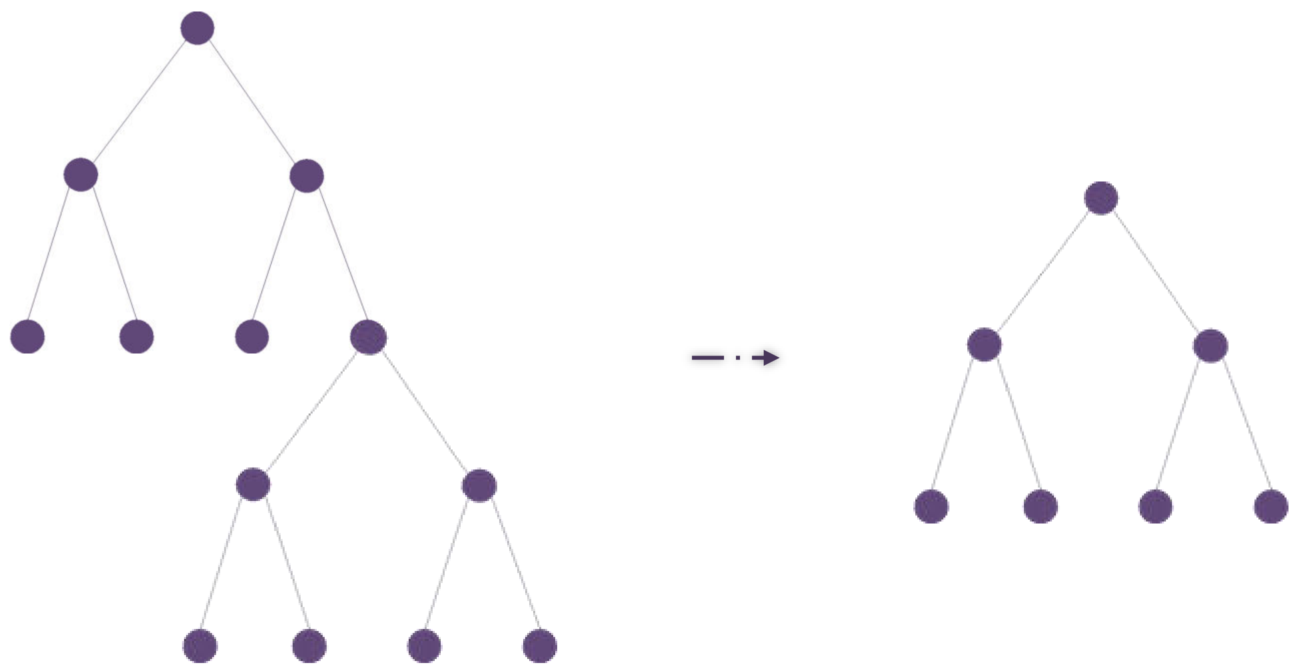
Leaf/Terminal Node

Pruning

Branch/Sub-tree

Parent and Child Node

When we remove sub-nodes of a decision node, this process is called pruning. In other words, it is the opposite process of splitting



Root Node

Splitting

Decision Node

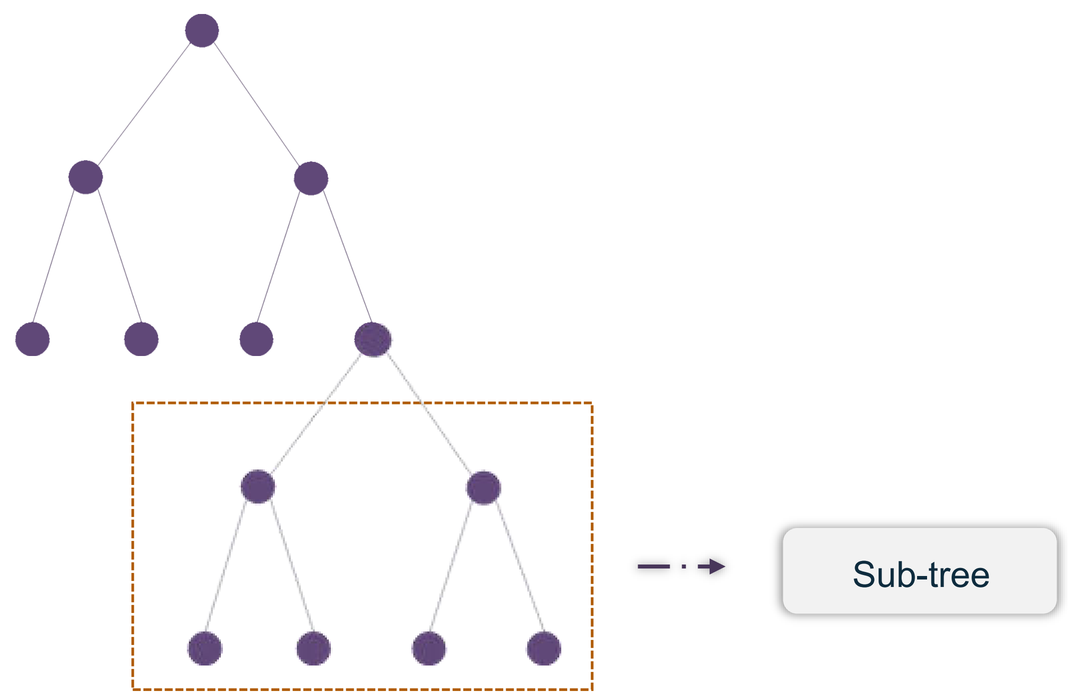
Leaf/Terminal Node

Pruning

Branch/Sub-tree

Parent and Child Node

A subsection of the entire tree is called a branch or sub-tree



Root Node

Splitting

Decision Node

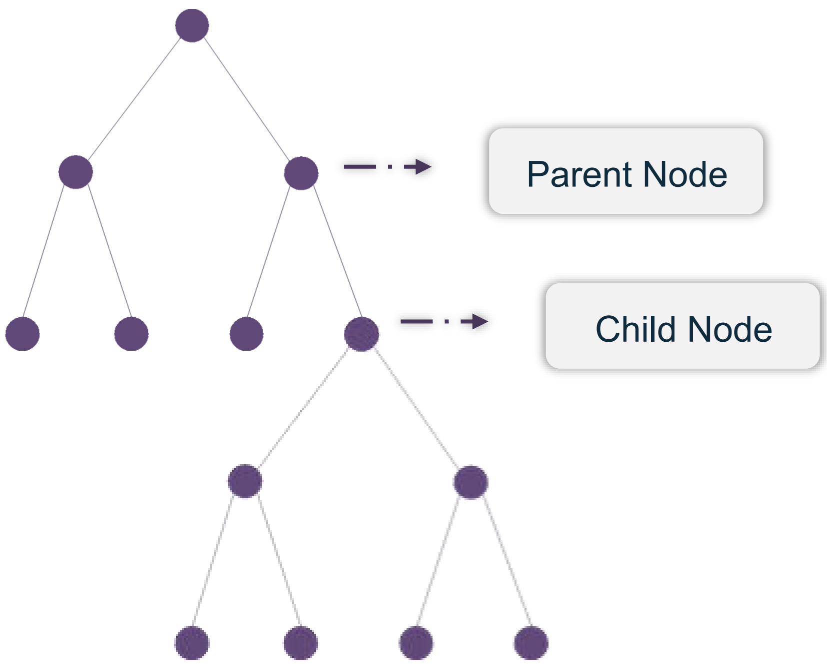
Leaf/Terminal Node

Pruning

Branch/Sub-tree

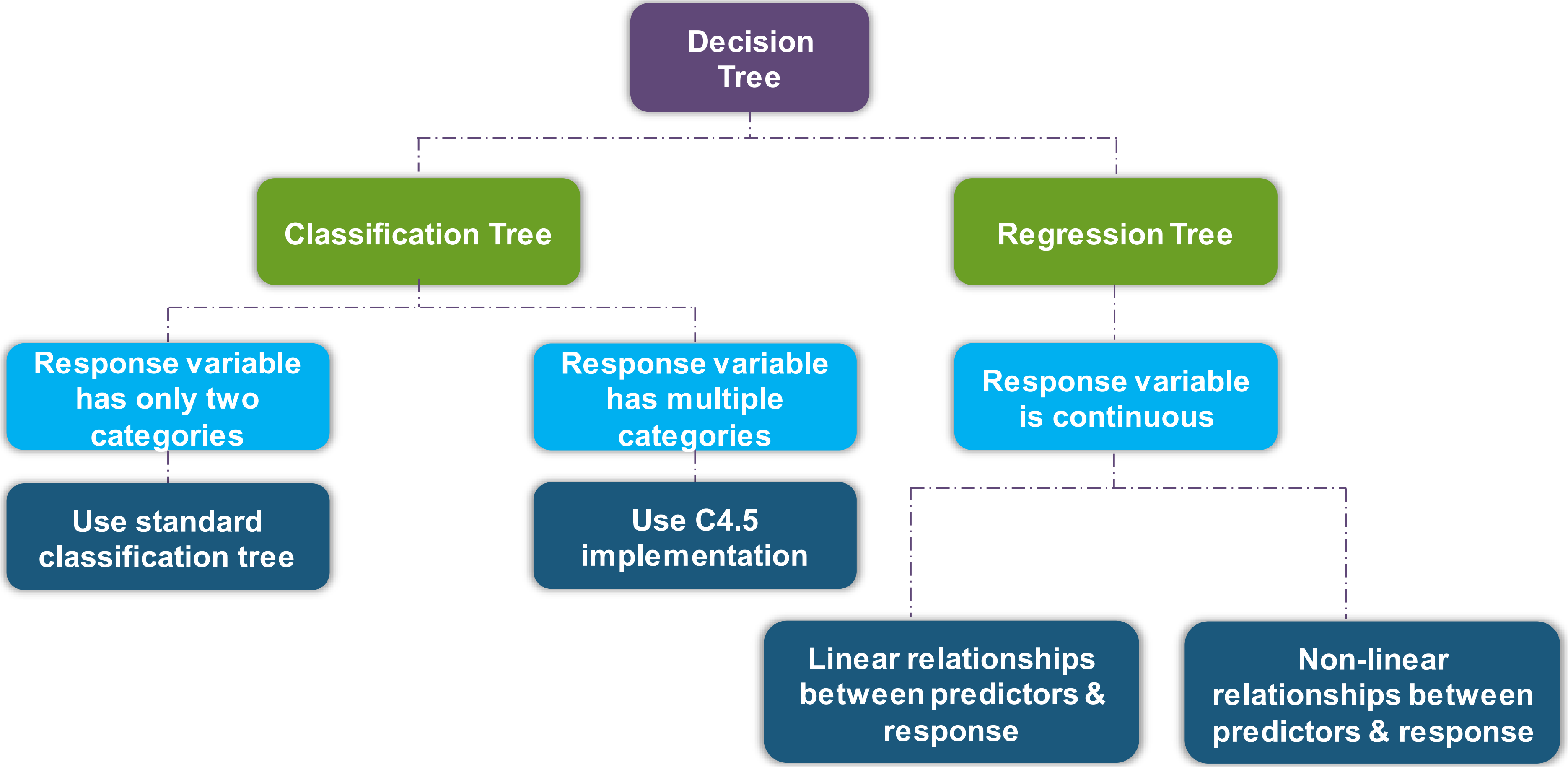
Parent and Child Node

A node which is divided into sub-nodes is called the parent node of sub-nodes, whereas sub-nodes are the children of the parent node



Regression vs. Classification Trees

Regression vs. Classification Trees



Where to Split?

Where to Split?

The decision of making strategic splits heavily affects a tree's accuracy

Decision criteria are different for classification and regression trees

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes

Where to Split?

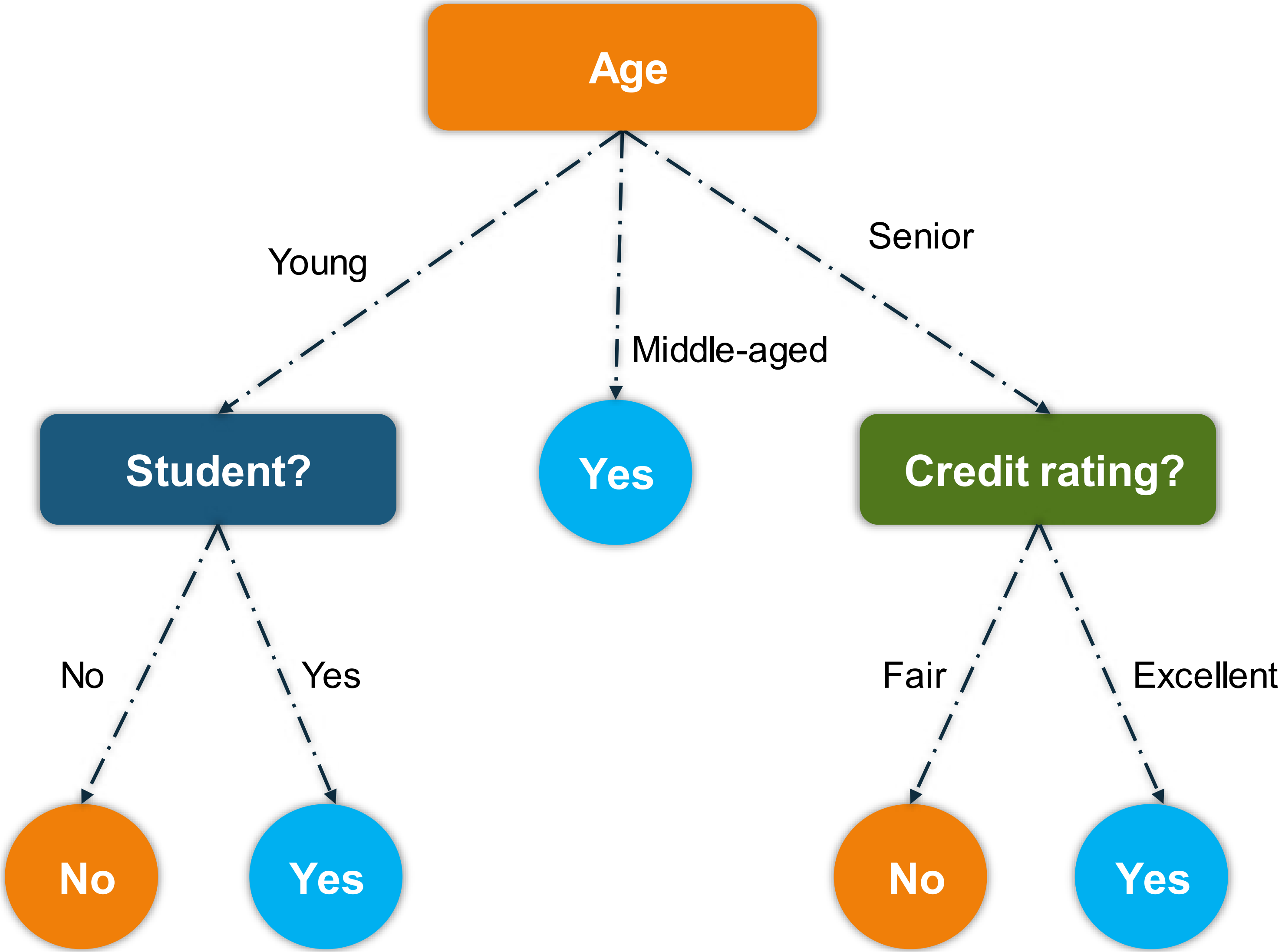
1

The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable.

2

Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Where to Split?



Common Algorithms

Few of the commonly used algorithms

1. ID3 (Iterative Dichotomiser)
2. C4.5
3. C5.0
4. CART (Classification and Regression Tree)
5. CHAID (Chi-squared Automatic Interaction Detector)

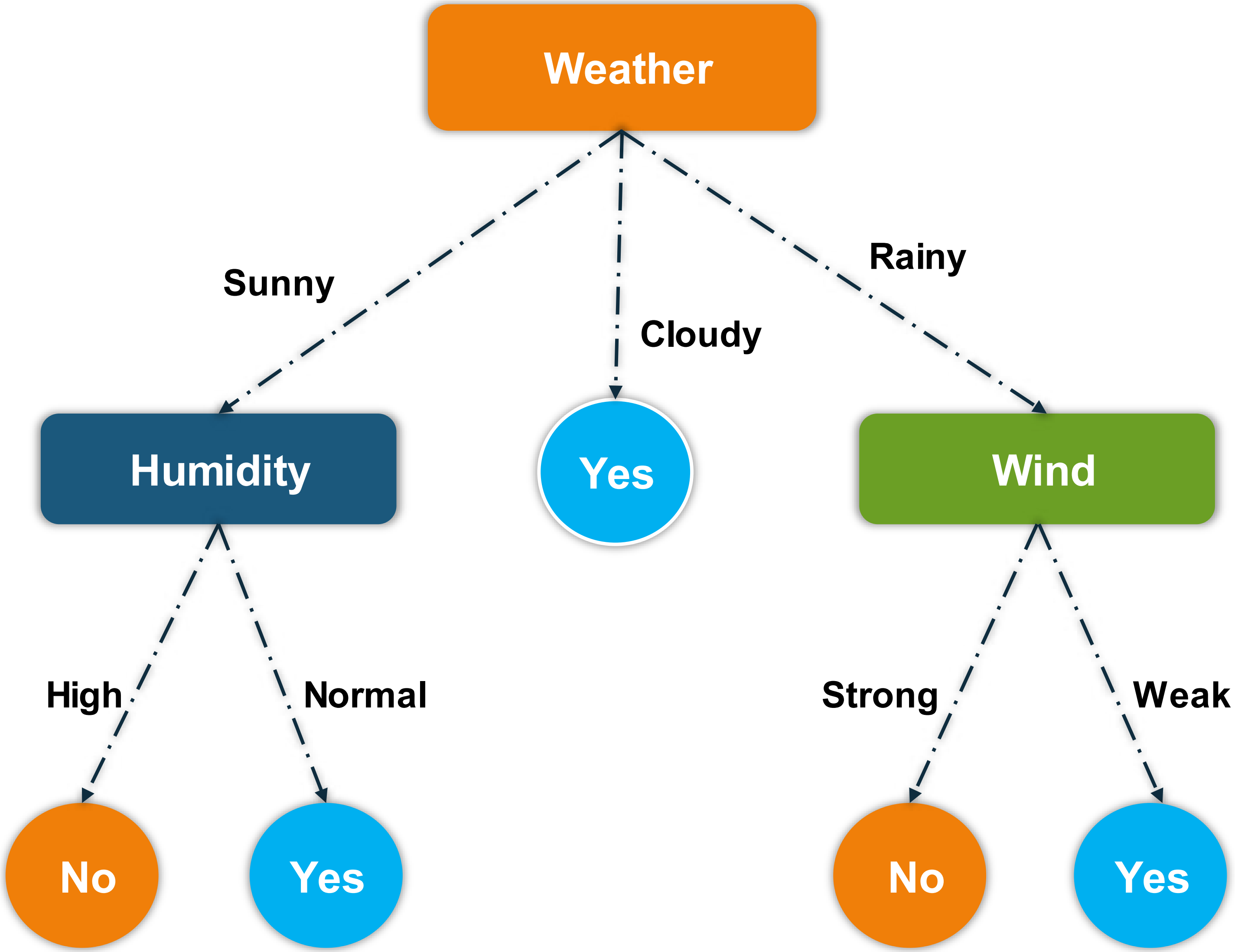
Algorithms (Example)

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Last 10 Days Observations

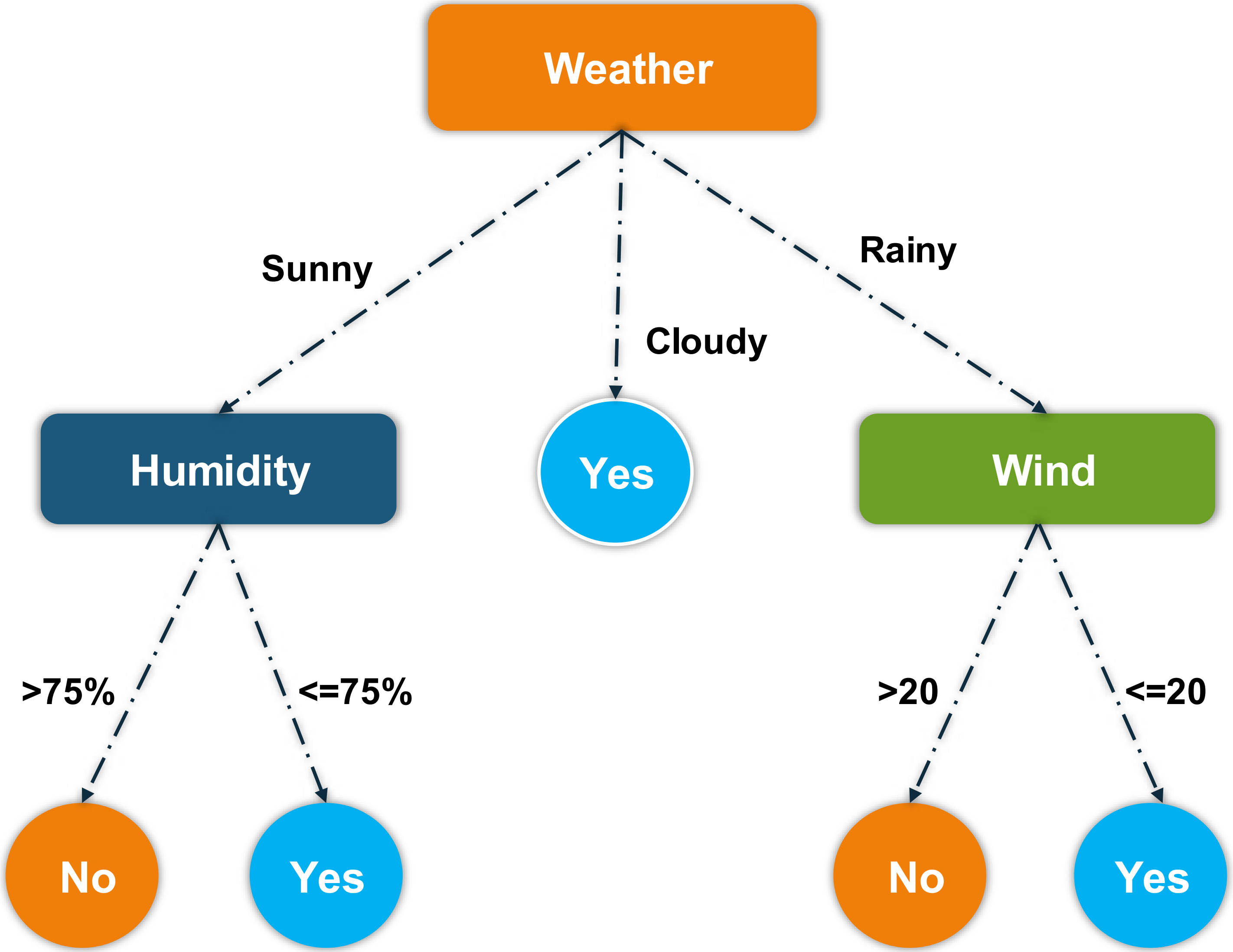
Algorithms (Example)

A decision tree for the concept "Play Badminton"



Algorithms (Example)

A decision tree for the concept "Play Badminton" (when attributes are continuous)



Decision Tree Algorithms

A general algorithm for a decision tree can be described -

Pick the **best attribute/feature**. The best attribute is the one which best splits or separates the data

Ask the relevant question. → Follow the answer path. → Go to step 1 until you arrive at the answer.

The best split is the one which separates two different labels into two sets

Gini Index

Gini Index

It works with the categorical target variable “Success” or “Failure”

It performs only binary splits

Higher the value of Gini, higher the homogeneity

CART (Classification and Regression Tree) uses Gini method to create binary splits

Gini Index

STEPS

Calculate Gini for sub-nodes, using the formula, sum of square of probability for success and failure ($p^2 + q^2$)

Calculate Gini for split using the weighted Gini score of each node of that split

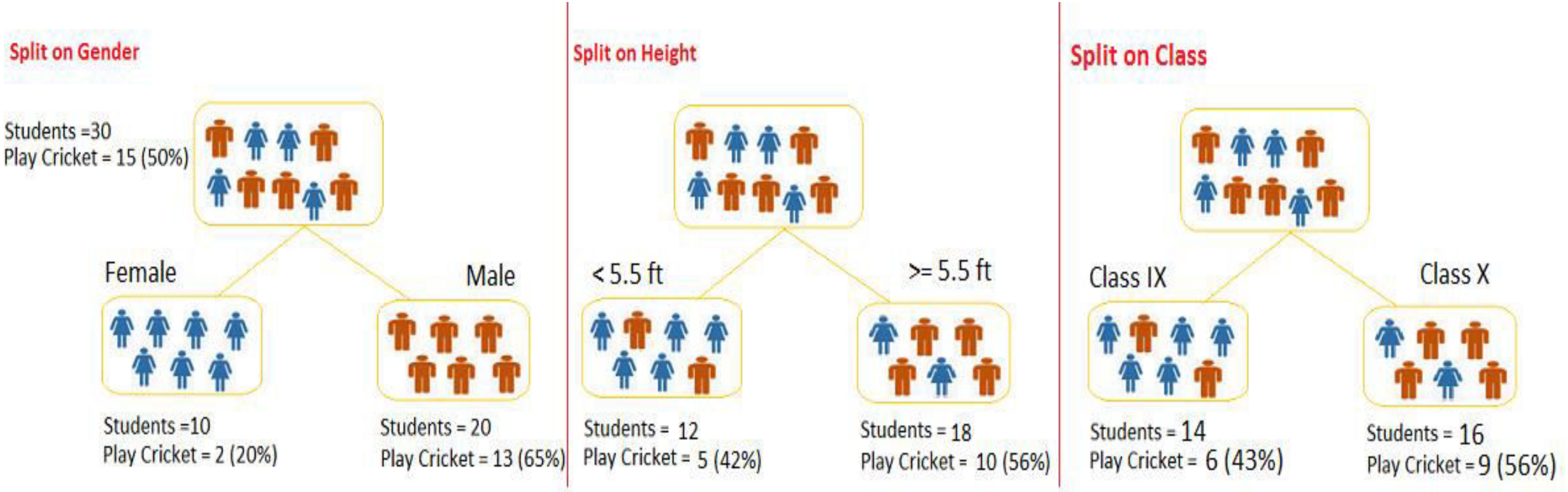
Gini Index Example

We have a sample of 30 students with three variables: Gender (Boy/Girl), Class (IX/X) and Height (5 to 6 ft)

15 out of these 30 play cricket in leisure time. Now, I want to create a model to predict who will play cricket during leisure period

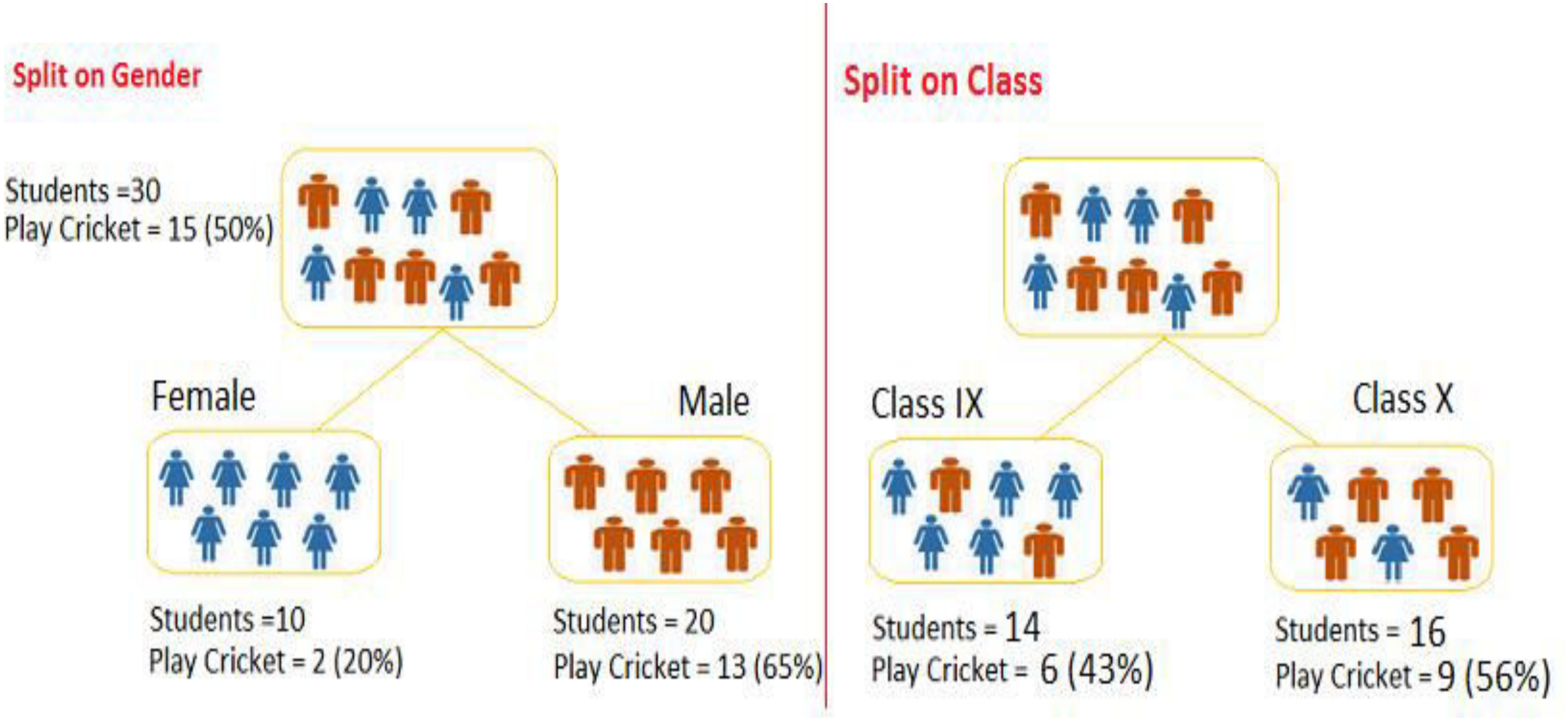
We need to segregate students who play cricket in their leisure time based on the highly significant input variable among all three

Gini Index Example



Gini Index Example

We would split the population using two input variables, Gender and Class. Then, we would identify which split is producing more homogeneous sub-nodes using Gini index



Gini Index Example

Split on Gender

Calculate Gini for the sub-node, Female

(0.2)*(0.2) + (0.8)*(0.8) = 0.68

Calculate Gini for the sub-node, Male

(0.65)*(0.65) + (0.35)*(0.35) = 0.55

Calculate weighted Gini for Split on Gender

(10/30)*0.68 + (20/30)*0.55 = **0.59**

Gini Index Example

Split on Class

Calculate Gini for the sub-node, Class IX

((0.43)*(0.43) + (0.57)*(0.57)=0.51

Calculate Gini for the sub-node, Class X

(0.56)*(0.56) + (0.44)*(0.44)=0.51

Calculate weighted Gini for Split on Class

(14/30)*0.51 + (16/30)*0.51 = **0.51**

The Gini score for Split on Gender is higher than Split on Class; hence, the node split will take place on Gender

Information Gain

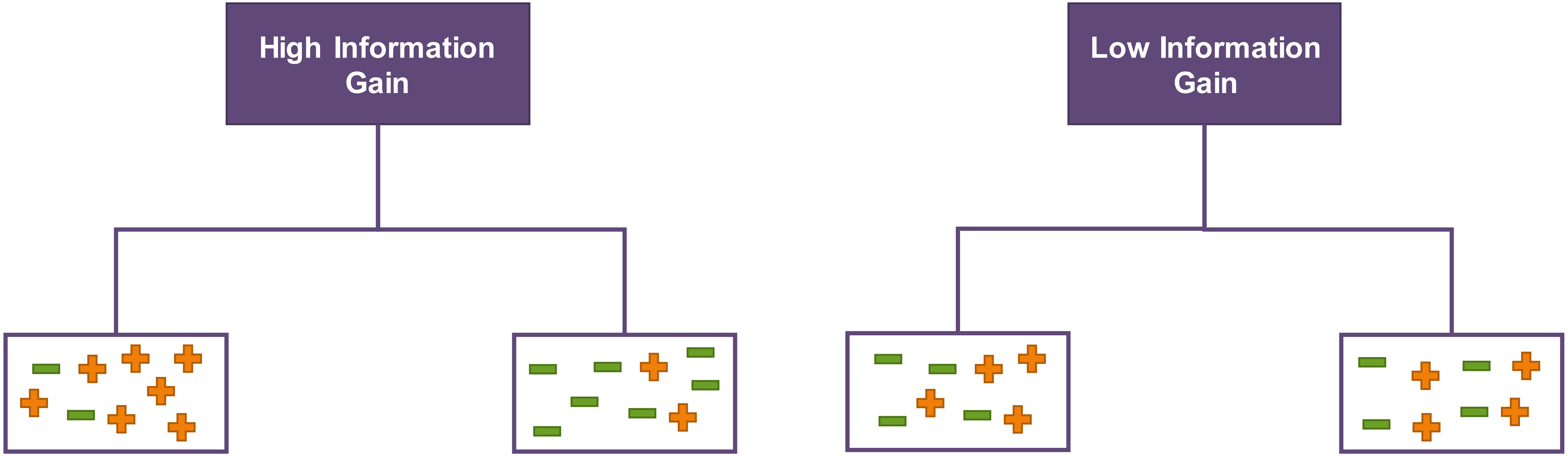
Information Gain

Information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification

$$InformationGain = Entropy(parent\_node) - [AverageEntropy(children)]$$

Constructing a decision tree is all about finding the attribute that returns the **highest information gain** (i.e., the most homogeneous branches)

Information Gain



Information Gain

Entropy measures the level of ***impurity*** in a group of examples

ID3 algorithm uses entropy to calculate the homogeneity of a sample

If the sample is completely homogeneous, the entropy is zero

Information Gain

Entropy measures the level of ***impurity*** in a group of examples

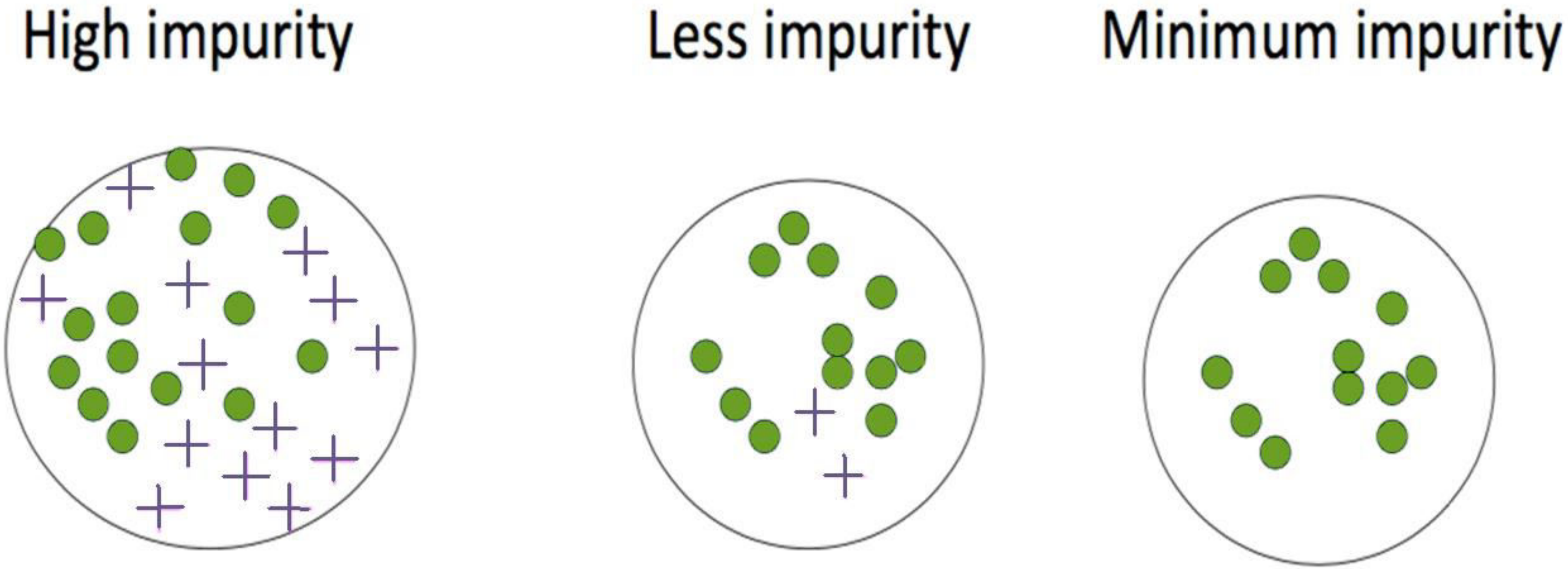
$$E = -p \log_2(p) - q \log_2(q).$$

Here, p and q is probability of success and failure, respectively, in that node

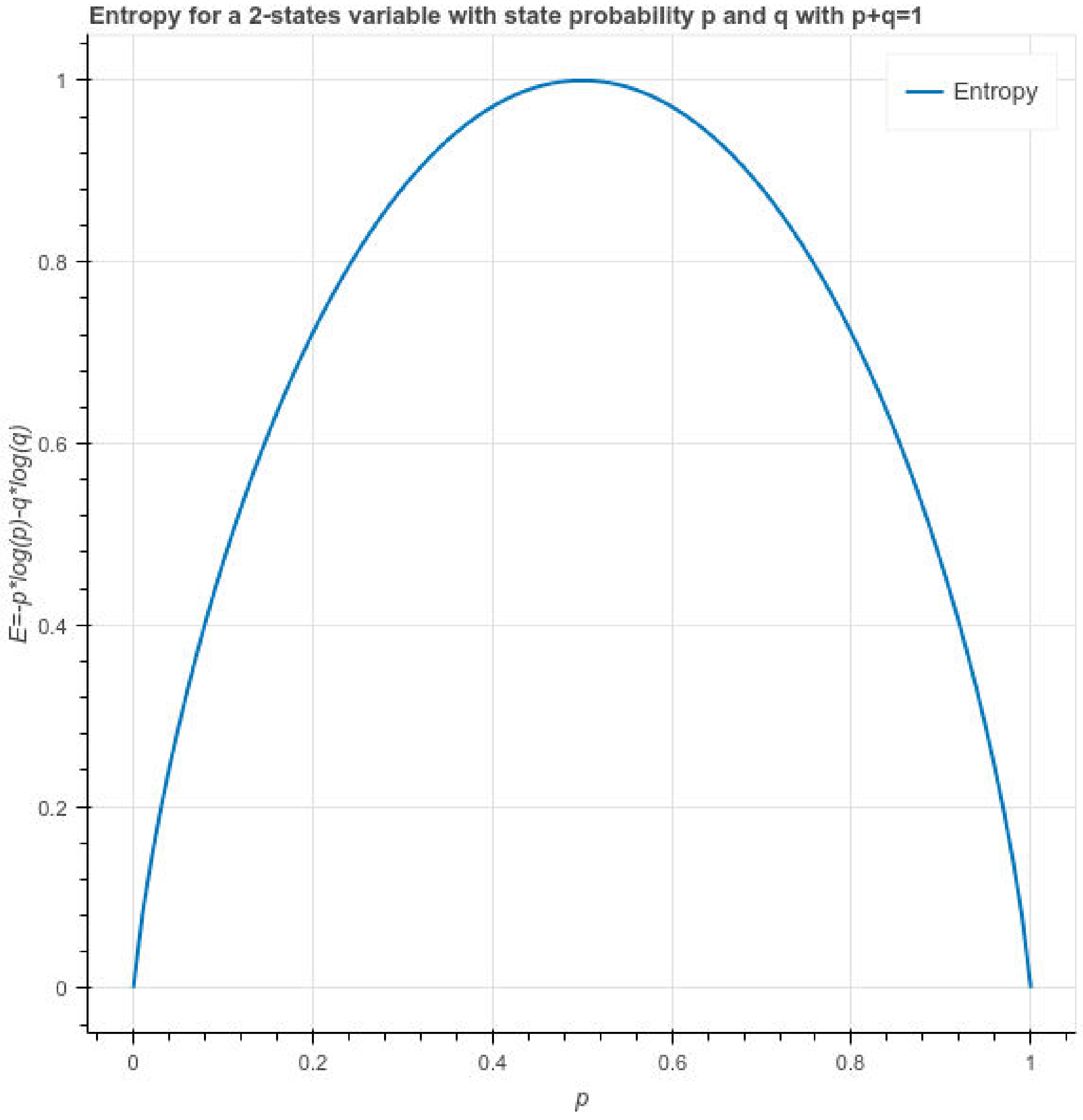
It chooses the split which has the lowest entropy compared to parent node and other splits. The lesser the entropy, the better it is

Information Gain

Entropy measures the level of ***impurity*** in a group of examples



Information Gain



Information Gain

Steps to calculate entropy for a split:

1. Calculate entropy of the parent node

Calculate entropy of each individual node of the split and calculate weighted average of all sub-nodes available in the split

We can derive information gain from entropy as **1 - Entropy**

Information Gain

Entropy for the parent node =

-(15/30) log2 (15/30) – (15/30) log2 (15/30) = **1**

Here, 1 shows that it is an impure node

Entropy for the Female node =

-(2/10) log2 (2/10) – (8/10) log2 (8/10) = 0.72

Entropy for the Male node =

(13/20) log2 (13/20) – (7/20) log2 (7/20) = 0.93

Entropy for split on Gender =

Weighted entropy of sub-nodes = (10/30)*0.72 + (20/30)*0.93 = 0.86

Information Gain

Entropy for the Class IX node =

(6/14) log2 (6/14) – (8/14) log2 (8/14) = 0.99

Entropy for the Class X node =

(9/16) log2 (9/16) – (7/16) log2 (7/16) = 0.99

Entropy for split on **Class** = (14/30)*0.99 + (16/30)*0.99 = **0.99**

Entropy for ***Split on Gender*** is the lowest among all, so the tree will split on *Gender*

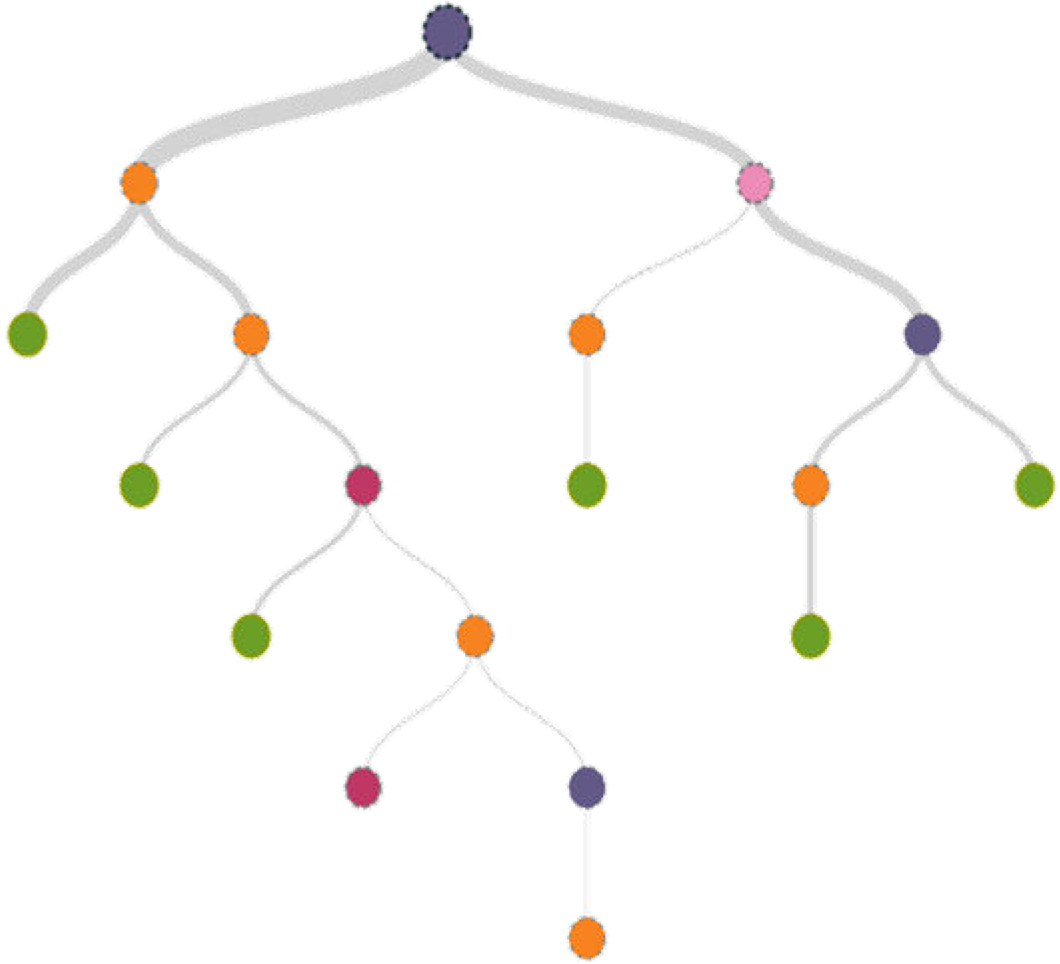
Regression Tree

Regression Tree

CART can handle both classification and regression tasks

The splitting criterion for CART is **MSE** (mean squared error)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$



Regression Tree

Suppose we are doing a binary tree. The algorithm first will pick a value and split the data into two subset. For each subset, it will calculate the MSE for each set separately

$$MSE(node) = \frac{1}{n_i} \sum_{data_i} (\hat{Y}_i - Y_i)^2$$

$$MSE(tree) = \frac{1}{n} \sum_{i=1}^{2} \sum_{data_i} (\hat{Y}_i - Y_i)^2$$

The tree chooses a value with the smallest MSE value to split the tree. The Yi <hat> for each subset is just the mean value with subset

Regression Tree

If the relationship between dependent and independent variables is well approximated by a linear model, linear regression will outperform the tree-based model

If there is a high nonlinearity and complex relationship between dependent and independent variables, a tree model will outperform the linear regression method

If you need to build a model which is easy to explain to people, a decision tree model will always do better than a linear model. Decision tree models are even simpler to interpret than linear regression!

Decision Tree in R

Problem Statement

Building a decision tree model on top of the customer_churn dataset

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7590-VHVEG	Female	0	Yes	No	1	No	No phone service
5575-GNVDE	Male	0	No	No	34	Yes	No
3668-QPYBK	Male	0	No	No	2	Yes	No
7795-CFOCW	Male	0	No	No	45	No	No phone service
9237-HQITU	Female	0	No	No	2	Yes	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes
6713-OKOMC	Female	0	No	No	10	No	No phone service
7892-POOKP	Female	0	Yes	No	28	Yes	Yes
6388-TABGU	Male	0	No	Yes	62	Yes	No

Tasks to be Performed

1. Build a classification model where the dependent variable is “churn” and the independent variable is “tenure” and find out the accuracy of the model built

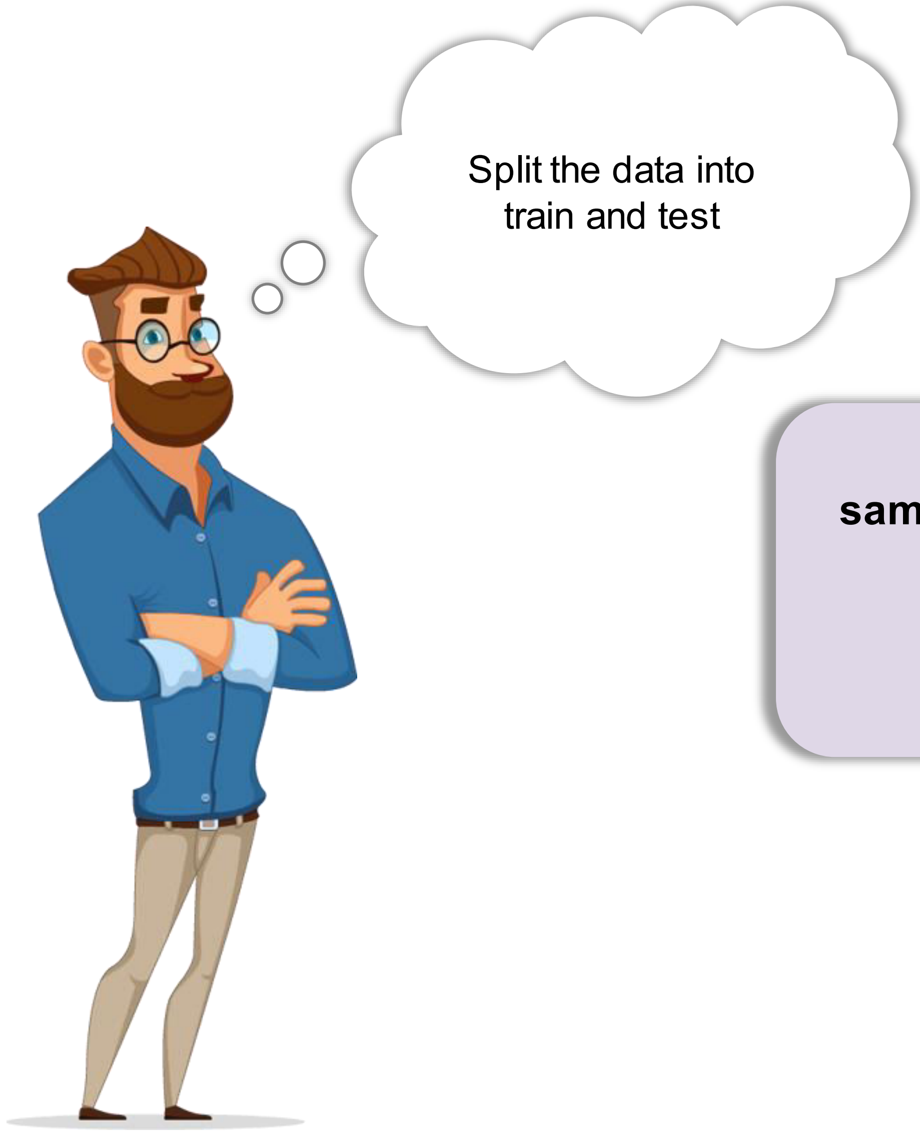
2. Build a classification model where the dependent variable is “churn” and the independent variables are “tenure” and “Monthly_Charges” and find out the accuracy of the model built

3. Build a classification model where the dependent variable is “churn” and the independent variables are “tenure”, ”MonthlyCharges”, ”Contract” and “TechSupport” and find out the accuracy of the model built

Decision Tree in R


```
customer_churn<-read.csv("C:/Users/INTELLIPAAT/Desktop/customer_churn.csv")
```

Decision Tree in R




```
sample.split(customer_churn$Churn,SplitRatio  = 0.65)-> split_tag
subset(customer_churn,  split_tag==T)->train
subset(customer_churn,  split_tag==F)->test
```

Decision Tree in R

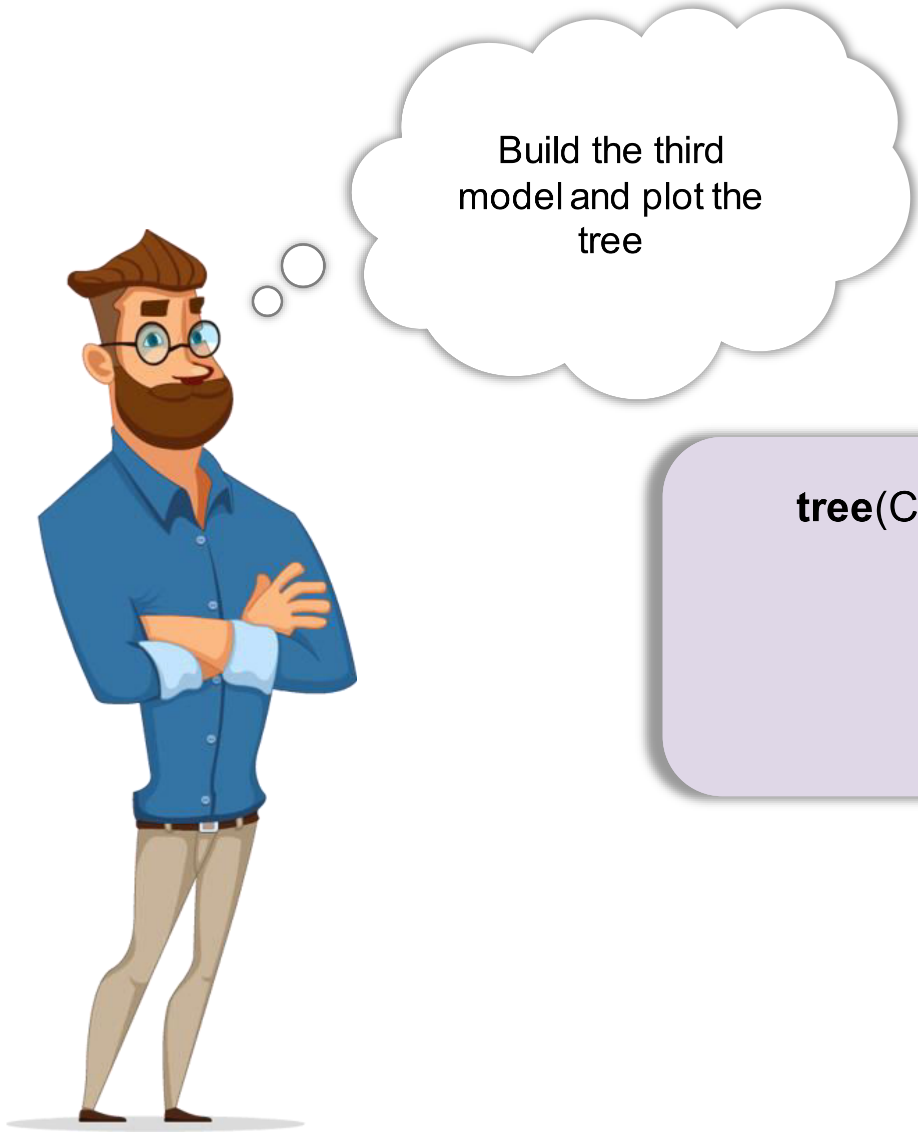

```
library(tree)
tree(Churn~tenure,  data=train)->
mod_tree1
```

```
plot(mod_tree1)
text(mod_tree1)
```

Decision Tree in R

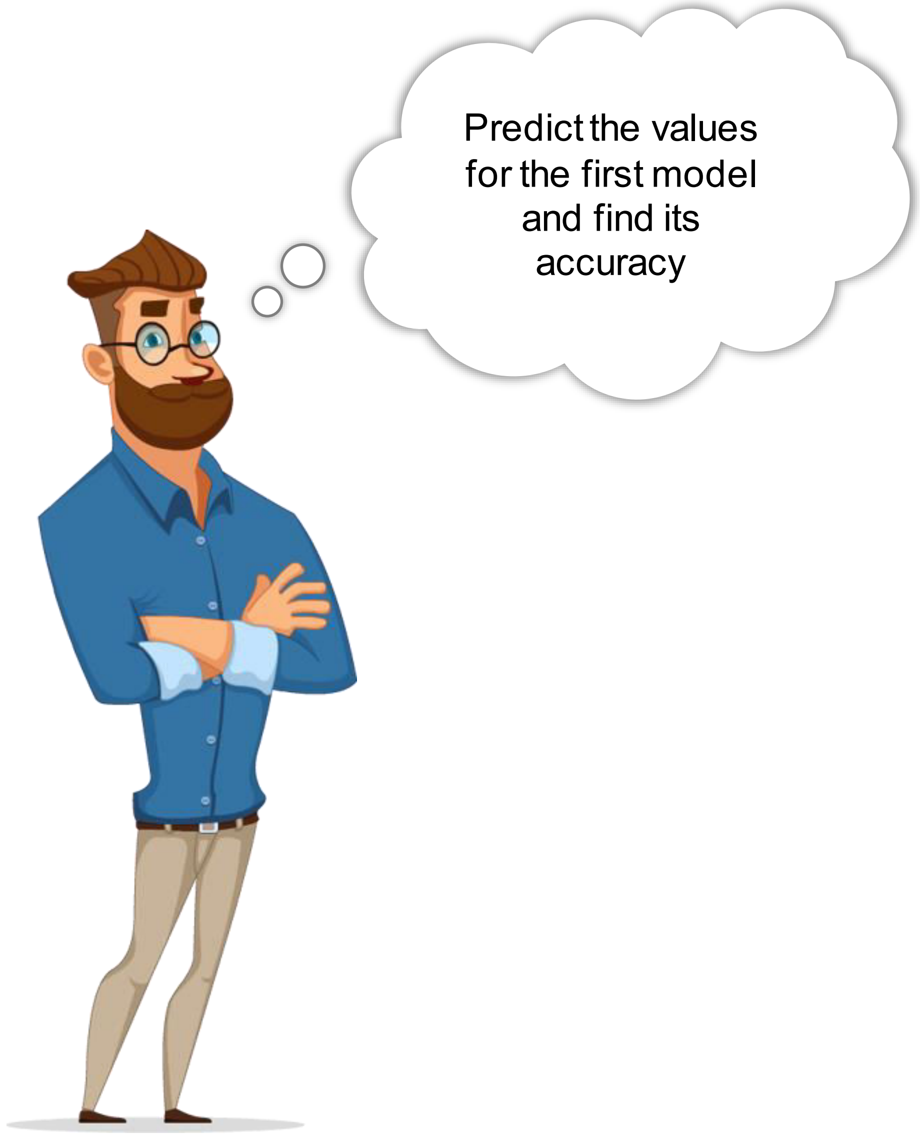

```
tree(Churn~tenure+MonthlyCharges,  data=train)->
mod_tree2

plot(mod_tree2)
text(mod_tree2)
```

Decision Tree in R




```
tree(Churn~tenure+MonthlyCharges+Contract+TechSupport,  data=train)->
mod_tree3
plot(mod_tree3)
text(mod_tree3)
```

Decision Tree in R



predict(mod_tree1,newdata=test,type="class")->result1

table(test**$**Churn, result1)

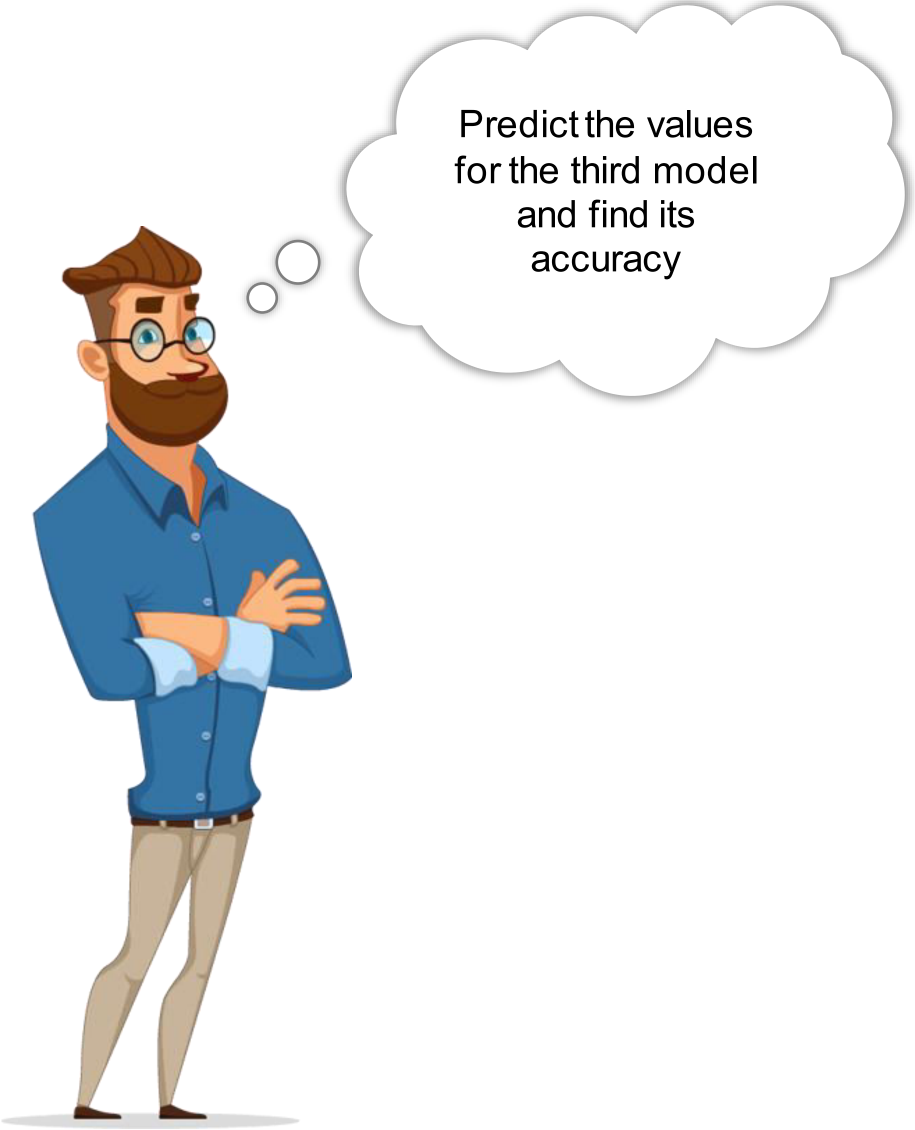
Decision Tree in R

Predict the values for the second model and find its accuracy

```
predict(mod_tree2,newdata=test,type="class")->result2
```

```
table(test$Churn, result2)
```

Decision Tree in R



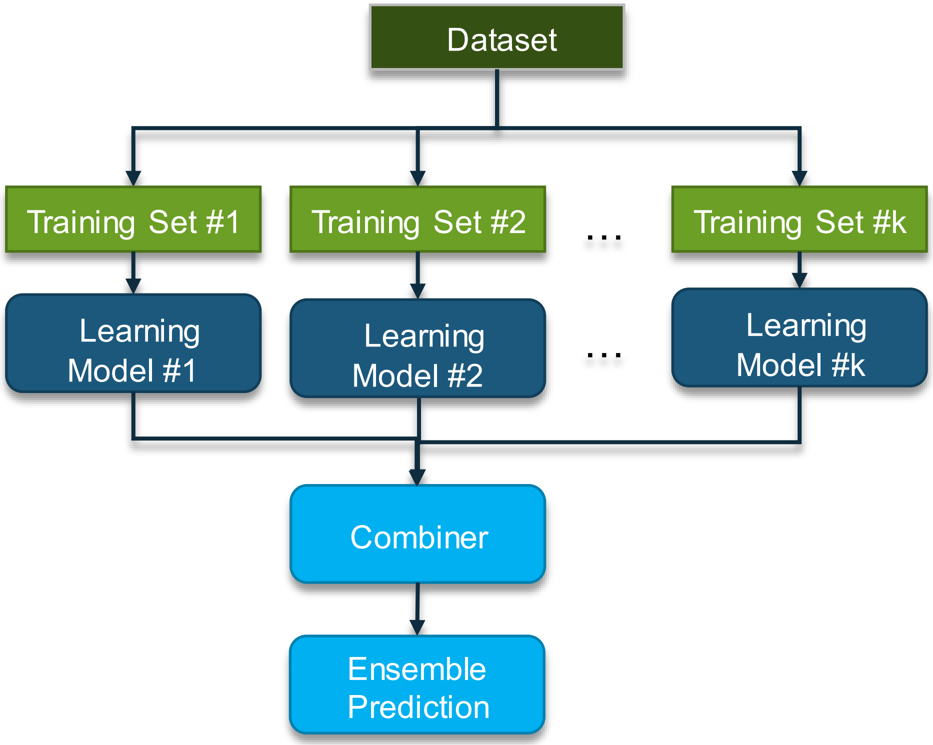

```
predict(mod_tree3,newdata=test,type="class")->result3
```

```
table(test$Churn, result3)
```

Ensemble Models

Ensemble Models

Ensemble methods involve a group of predictive models to achieve a better accuracy and model stability. They are known to impart supreme boost to tree-based models



Bias–Variance Trade-off

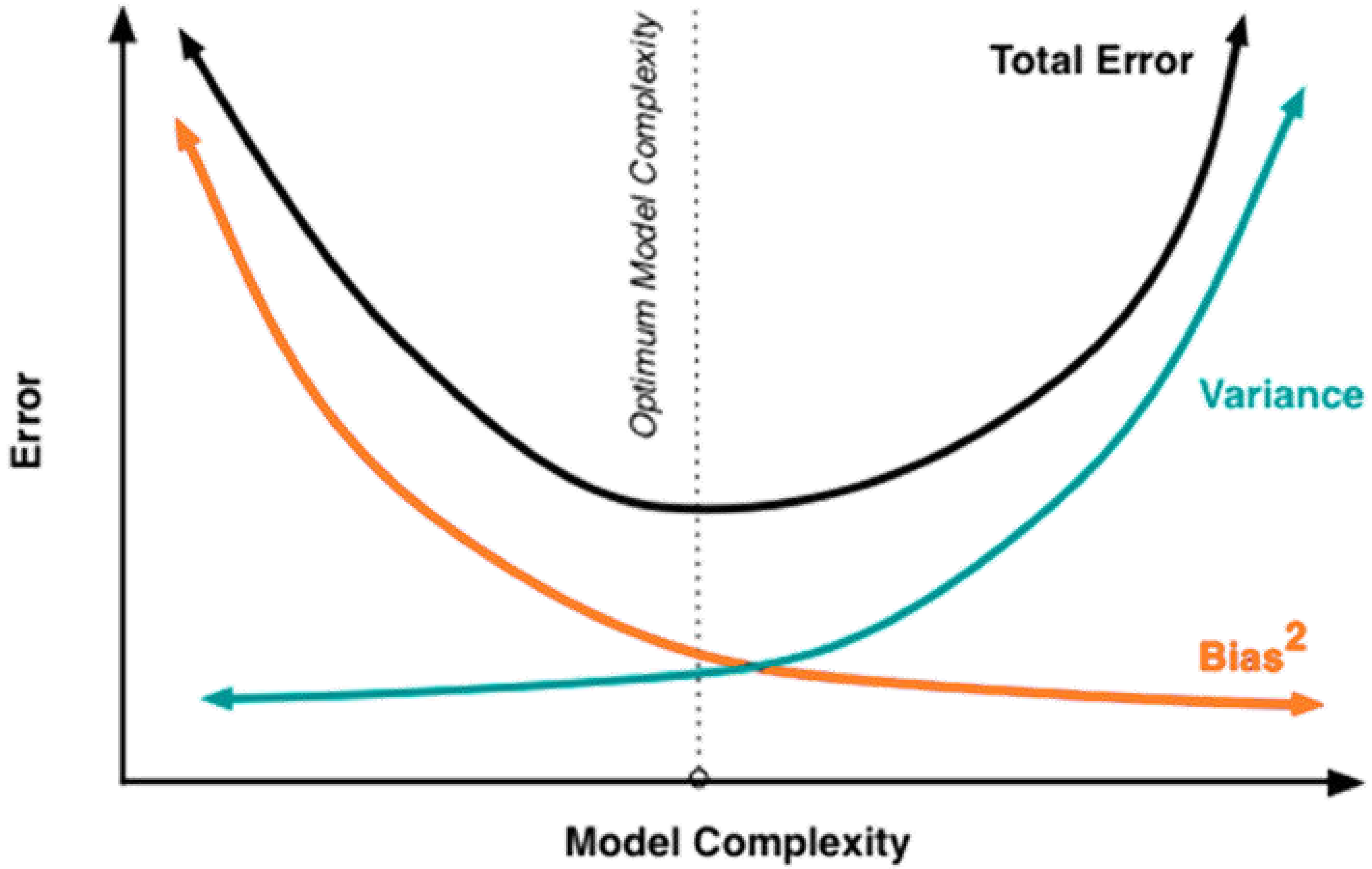
Bias–Variance Trade-off

Bias is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting)

Variance is an error from sensitivity to small fluctuations in the training set

High variance can cause an algorithm to model random noise in the training data, rather than the intended outputs (overfitting)

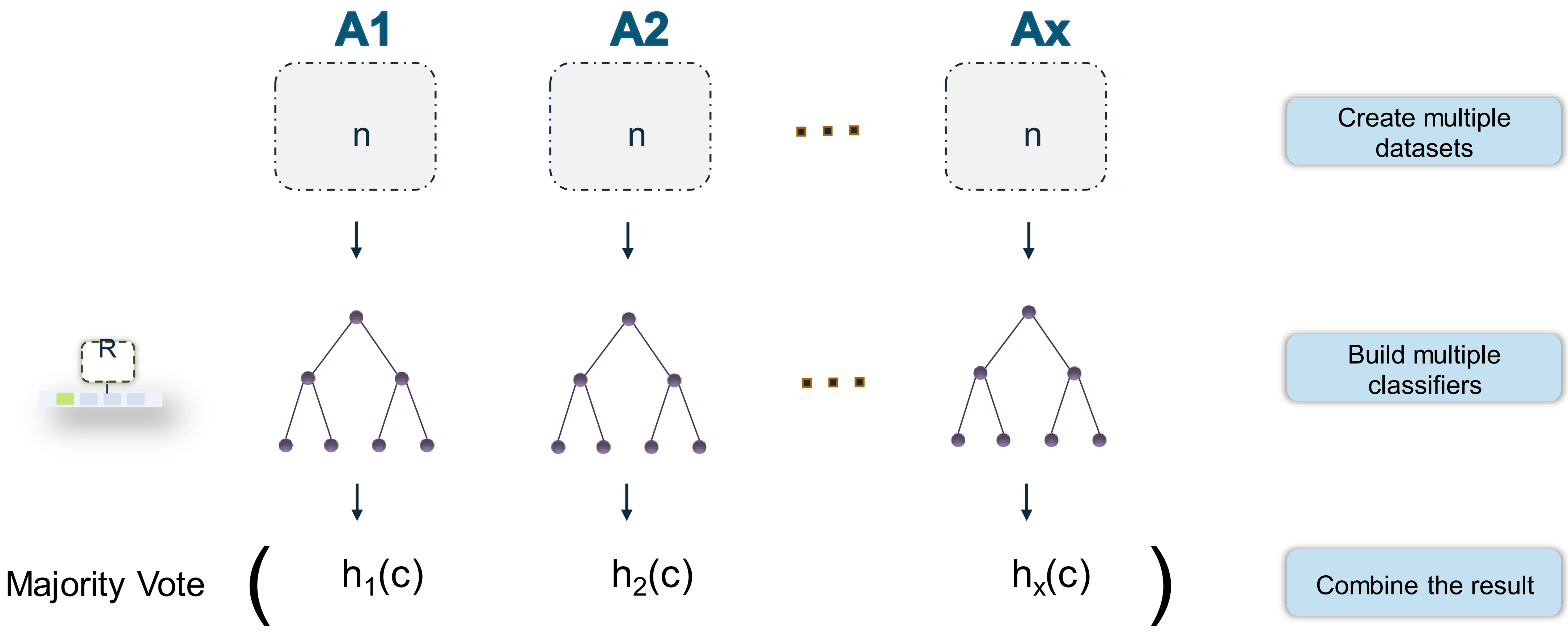
Bias–Variance Trade-off



Bagging

Bagging

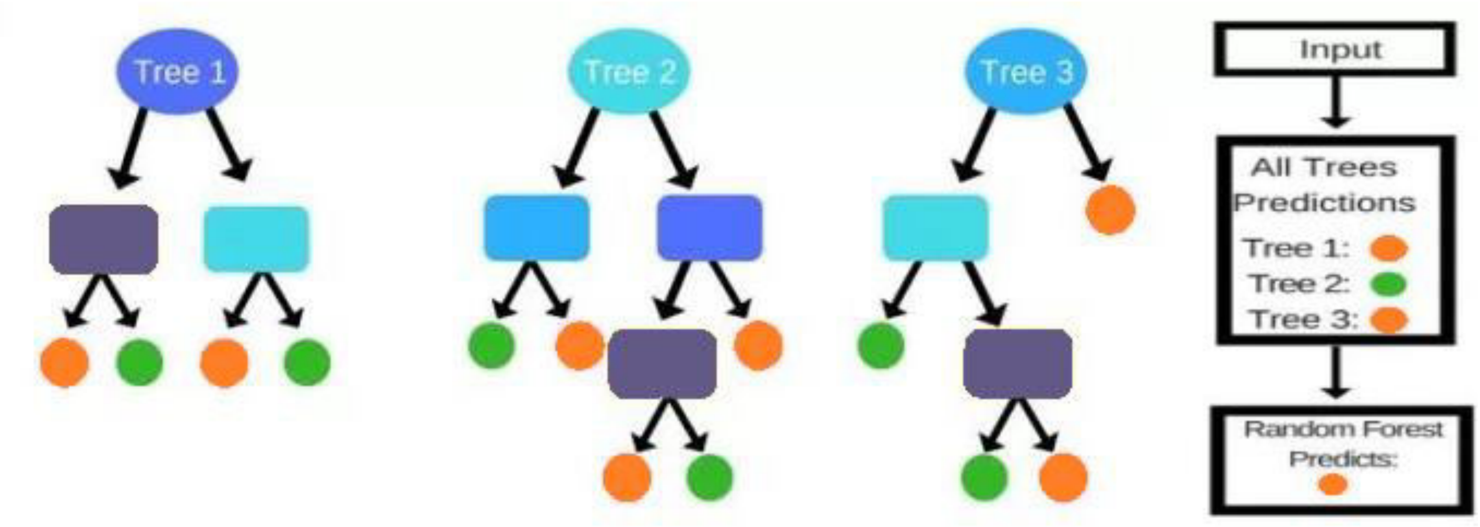
Bagging is a technique used to reduce the variance of our predictions by combining the result of multiple classifiers modeled on different sub-samples of the same dataset



Random Forest

Random Forest

Random Forest is a versatile machine learning method capable of performing both regression and classification tasks. It is a type of ensemble learning method, where a group of weak models combine to form a powerful model



Random Forest – How Does it Work?

The algorithm creates random subsets with random values from the complete dataset

From each subset, it creates a decision tree. Each tree is built from a sample

So, it creates multiple decision trees and then merge the results

Random Forest - How Does it Work?

Sampling is done on the training dataset. Every time, a new sample is chosen to build the tree. This introduction of randomness increases the bias and reduces the variances of the model

This prevents the overfitting of the model which is a serious concern in the case of decision trees

This yields much better performing generalized models

Random Forest – How Does it Work?

There are 2 levels of randomness:

At Row Level	• Each decision tree gets a random sample of the training data
At Column Level	• Each decision tree gets a random sample of columns. Not all trees get the same number of same column

Random Forest in R

Problem Statement

Building a Random Forest model on top of the customer_churn dataset

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7590-VHVEG	Female	0	Yes	No	1	No	No phone service
5575-GNVDE	Male	0	No	No	34	Yes	No
3668-QPYBK	Male	0	No	No	2	Yes	No
7795-CFOCW	Male	0	No	No	45	No	No phone service
9237-HQITU	Female	0	No	No	2	Yes	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes
6713-OKOMC	Female	0	No	No	10	No	No phone service
7892-POOKP	Female	0	Yes	No	28	Yes	Yes
6388-TABGU	Male	0	No	Yes	62	Yes	No

Tasks to be Performed

1. Build a classification model where the dependent variable is “churn” and the independent variables are “tenure”, “MonthlyCharges”, “gender”, “InternetService” and “Contract”. Number of trees are 100 and number of variables available for split are **3**

2. Build a classification model where the dependent variable is “churn” and the independent variables are “tenure”, “MonthlyCharges”, “gender”, “InternetService” and “Contract”. Number of trees are 100 and number of variables available for split are **4**

3. Build a classification model where the dependent variable is “churn” and the independent variables are “tenure”, “MonthlyCharges”, “gender”, “InternetService” and “Contract”. Number of trees are 100 and number of variables available for split are **5**

Decision Tree in R

Split the data into
train and test

```
sample.split(customer_churn$Churn,SplitRatio  = 0.65)-> split_tag
subset(customer_churn,  split_tag==T)->train
subset(customer_churn,  split_tag==F)->test
```

Decision Tree in R


```
library(randomForest)
```

```
randomForest(Churn~MonthlyCharges+tenure+gender+InternetService+Contract,
data=train, mtry=3,ntree=100)-> mod_forest1
```

Decision Tree in R

Find the importance of independent variables and plot it

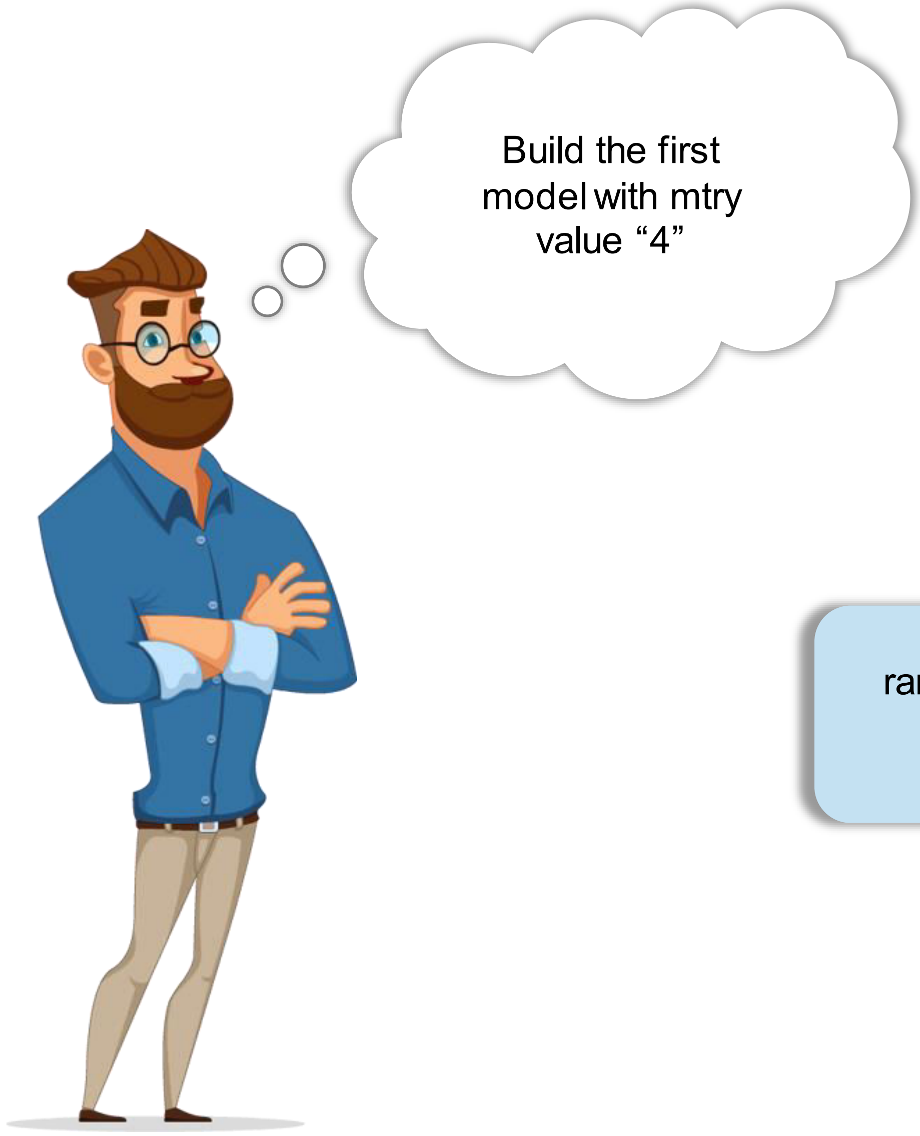
```
importance(mod_forest1)

varImpPlot(mod_forest1)
```

Decision Tree in R


```
predict(mod_forest1,newdata=test,type="class")->result_forest
```

```
table(test$Churn, result_forest)
```

Decision Tree in R



library(randomForest)

randomForest(Churn~MonthlyCharges+tenure+gender+InternetService+Contract, data=train, mtry=4,ntree=100)-> mod_forest2

Decision Tree in R

Find the importance of independent variables and plot it

```
importance(mod_forest2)

varImpPlot(mod_forest2)
```

Decision Tree in R


```
predict(mod_forest2,newdata=test,type="class")->result_forest2
```

```
table(test$Churn, result_forest2)
```

Decision Tree in R

Build the first model with mtry value “5”

```
library(randomForest)
```

```
randomForest(Churn~MonthlyCharges+tenure+gender+InternetService+Contract,
data=train, mtry=5,ntree=100)-> mod_forest3
```

Decision Tree in R


```
importance(mod_forest3)

varImpPlot(mod_forest3)
```

Decision Tree in R


```
predict(mod_forest3,newdata=test,type="class")->result_forest3
```

```
table(test$Churn, result_forest3)
```

Demo on “Carseats” Dataset

Problem Statement

Building a decision tree model on top of the “Carseats” dataset

Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban
9.50	138	73	11	276	120	Bad	42	17	Yes
11.22	111	48	16	260	83	Good	65	10	Yes
10.06	113	35	10	269	80	Medium	59	12	Yes
7.40	117	100	4	466	97	Medium	55	14	Yes
4.15	141	64	3	340	128	Bad	38	13	Yes
10.81	124	113	13	501	72	Bad	78	16	No
6.63	115	105	0	45	108	Medium	71	15	Yes
11.85	136	81	15	425	120	Good	67	10	Yes
6.54	132	110	0	108	124	Medium	76	10	No
4.69	132	113	0	131	124	Medium	76	17	No
9.01	121	78	9	150	100	Bad	26	10	No

Task 1

Task 1


```
library(ISLR)
```

```
data(package="ISLR")
carseats<-Carseats
```

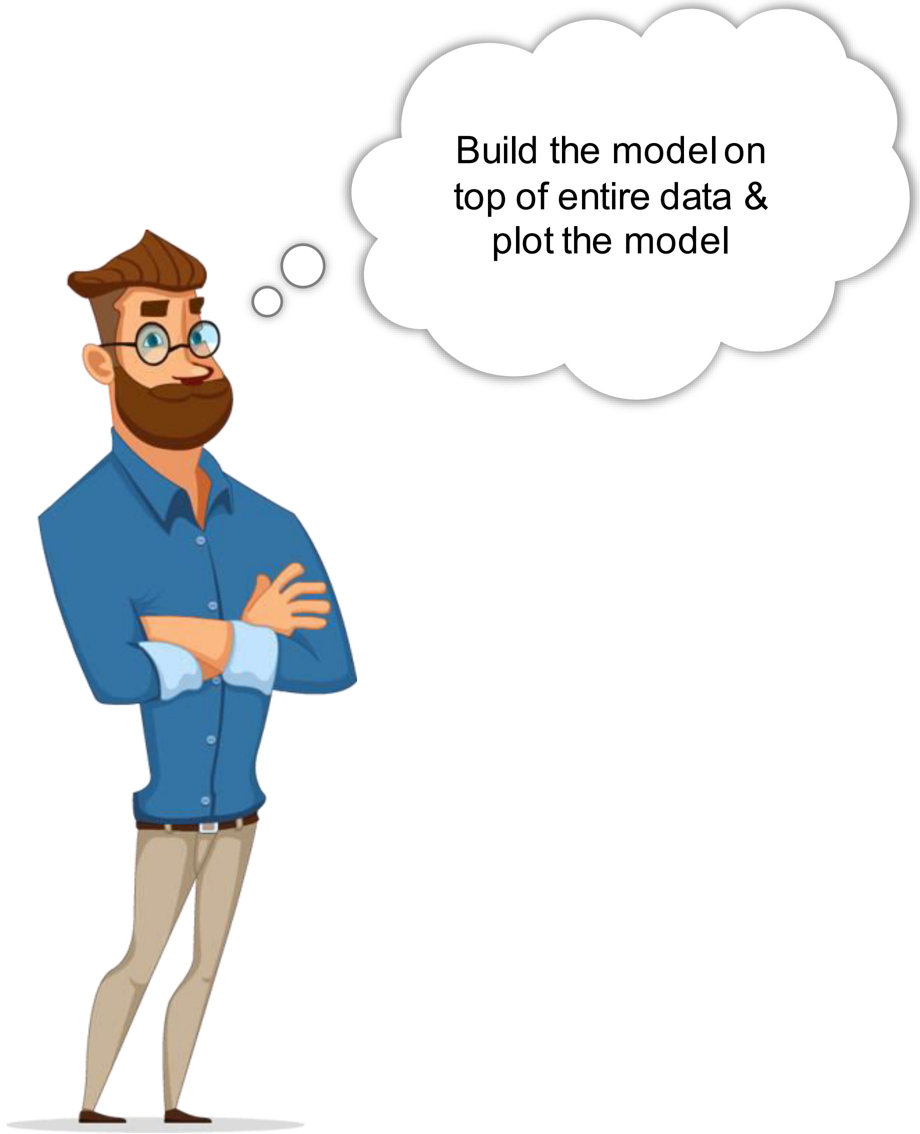
Task 1

Add a new variable ‘High’ to carseats dataset which has label ‘No’ if value in Sales column <=8

```
High = ifelse(carseats$Sales<=8, "No", "Yes")
```

```
carseats = data.frame(carseats, High)
```

Task 1




```
library(tree)
tree.carseats = tree(High~.-Sales, data=carseats)
summary(tree.carseats)
```

```
plot(tree.carseats)
text(tree.carseats,pretty = 0)
```

Task 2

Task 2


```
library(caTools)
set.seed(101)
sample.split(carseats$High,SplitRatio = .65) -> split_tag
subset(carseats,split_tag==T) -> train
subset(carseats,split_tag==F) -> test
```

```
tree.carseats = tree(High~.-Sales, train)
plot(tree.carseats)
text(tree.carseats, pretty=0)
```

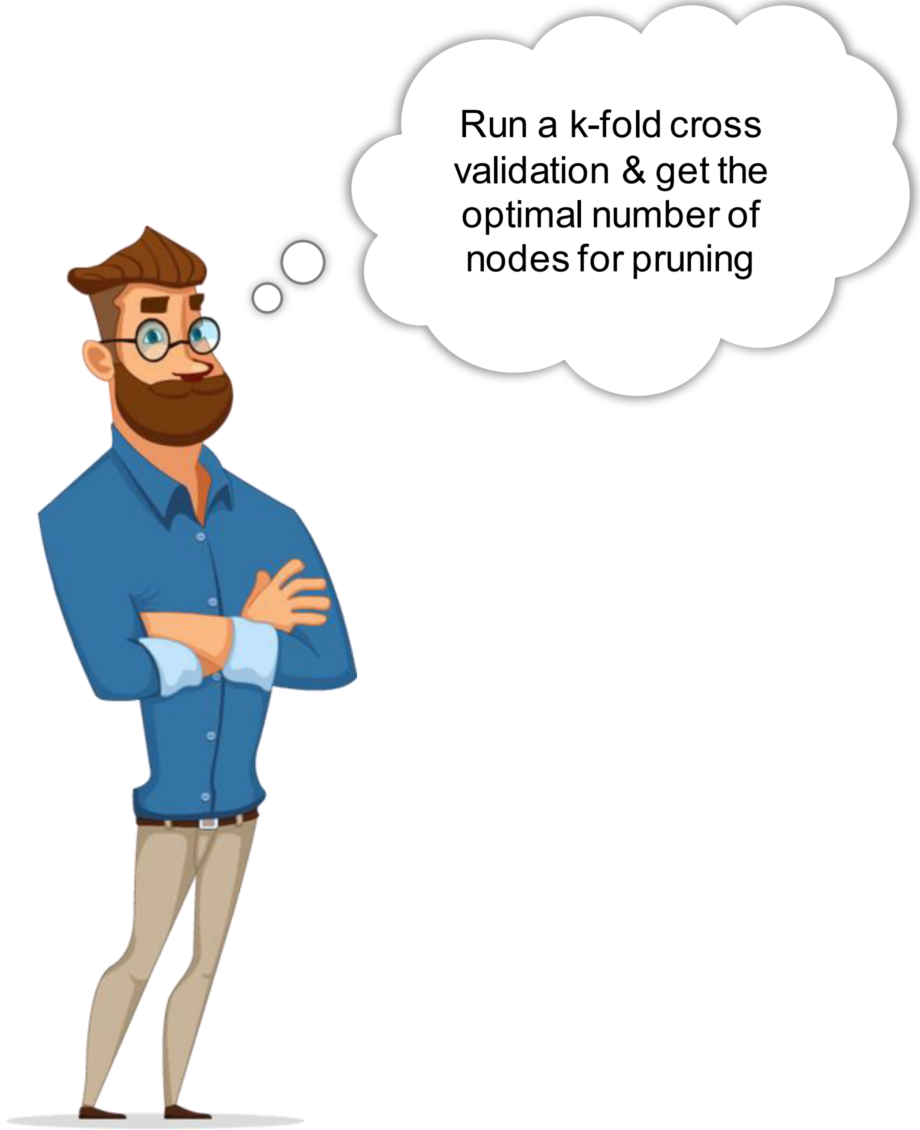
Task 2


```
tree.pred = predict(tree.carseats, test, type="class")
```

```
table(test$High,tree.pred)
```

Task 3

Task 3



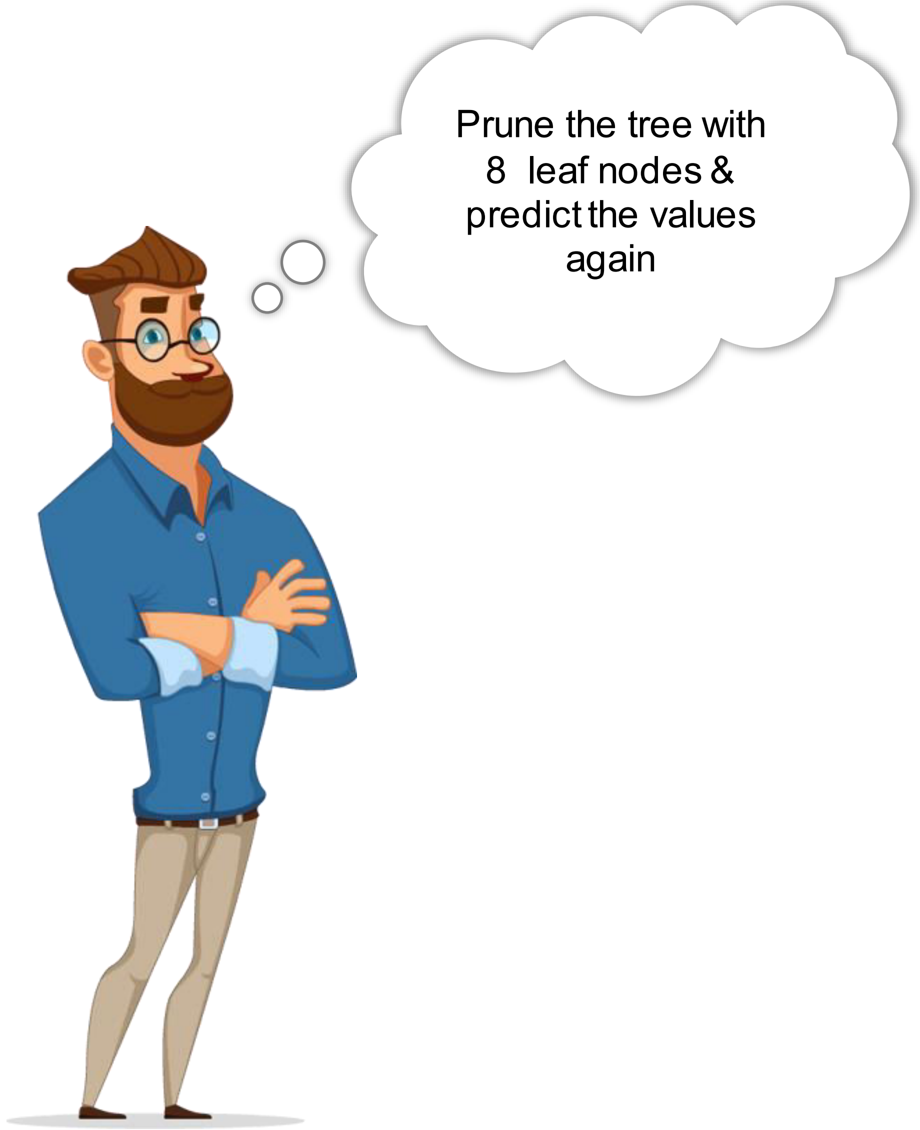

```
cv.carseats = cv.tree(tree.carseats, FUN = prune.misclass)
cv.carseats
plot(cv.carseats)
```

```
prune.carseats = prune.misclass(tree.carseats, best = 13)
plot(prune.carseats)
text(prune.carseats, pretty=0)
```

Task 3


```
tree.pred = predict(prune.carseats, test, type="class")
table(test$High,tree.pred)
```

Task 3




```
prune.carseats = prune.misclass(tree.carseats, best = 8)
plot(prune.carseats)
text(prune.carseats, pretty=0)
```

```
tree.pred = predict(prune.carseats, test, type="class")
table(test$High,tree.pred)
```

Demo on “iris” Dataset

Problem Statement

Building a decision tree model on top of the "iris" dataset

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa

Task 1

Task 1


```
library(party)
```

```
library(caret)
split_tag <- createDataPartition(iris$Species, p = 0.65, list = F)
iris[split_tag,] -> train
iris[-split_tag,] -> test
```

Task 1


```
mytree <- ctree(Species~., train)
plot(mytree)
```

```
predict(mytree,test,type="response") -> mypred
table(test$Species,mypred)
```

Task 2

Task 2

Build the 2^{nd} model
& predict the
values

```
mytree2 <- ctree(Species~Petal.Width+Petal.Length, train)
plot(mytree2)
```

```
predict(mytree2,test,type="response") -> mypred
table(test$Species,mypred)
```

Demo on “Boston” Dataset

Problem Statement

Building a decision tree model on top of the “Boston” dataset

crim	zn	indus	chas	nox	rm	age	dis	rad	tax
0.00632	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296
0.02731	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242
0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242
0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222
0.06905	0.0	2.18	0	0.4580	7.147	54.2	6.0622	3	222
0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222
0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311
0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311
0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311

Task 1

Task 1

Load the 'rpart' package & divide the data into train & test sets

```
library(rpart)
```

```
split_tag <- createDataPartition(Boston$medv, p = 0.65, list = F)
Boston[split_tag,] -> train
Boston[-split_tag,] -> test
```

Task 1

Build the 1st model
& plot the tree

```
my_tree <- rpart(medv~., train)
library(rpart.plot)
rpart.plot(my_tree)
```

Task 1

Predict the values
& find the Root
Mean Square Error

```r
predict(my_tree,newdata = test) -> predict_tree
cbind(Actual=test$medv,Predicted=predict_tree) -> final_data
as.data.frame(final_data) -> final_data
```

```r
(final_data$Actual - final_data$Predicted)  -> error
cbind(final_data,error) -> final_data
sqrt(mean((final_data$error)^2))
```

Task 2

Task 2


```
my_tree <- rpart(medv~rm+lstat+crim+nox, train)
library(rpart.plot)
rpart.plot(my_tree)
```

Task 2


```
predict(my_tree,newdata = test) -> predict_tree
cbind(Actual=test$medv,Predicted=predict_tree) -> final_data
as.data.frame(final_data) -> final_data
```

```
(final_data$Actual - final_data$Predicted)  -> error
cbind(final_data,error) -> final_data
sqrt(mean((final_data$error)^2))
```

Random Forest Demo on “CTG” Dataset

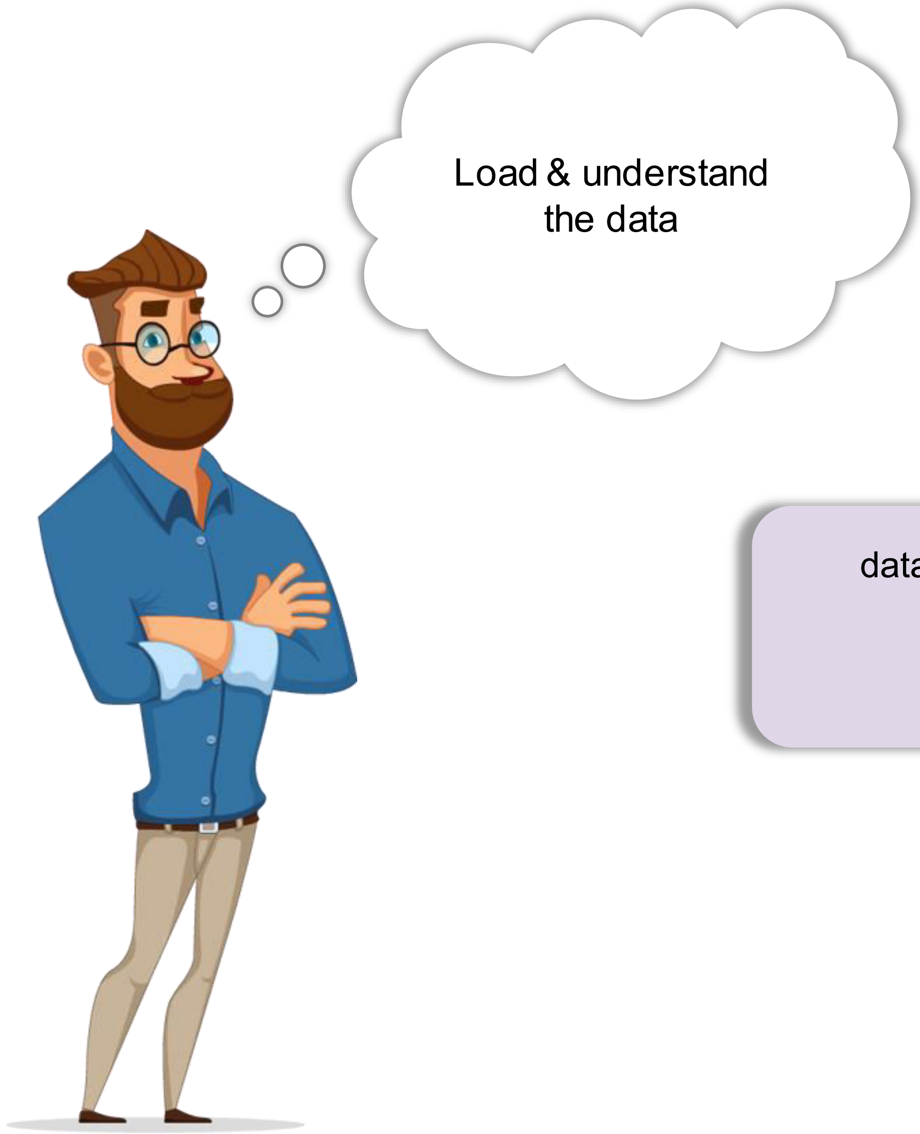
Problem Statement

Building a Random Forest model on top of the "CTG" dataset

LB	AC	FM	UC	DL	DS	DP	ASTV	MSTV	ALTV	MLTV	Width
120	0.000000000	0.000000000	0.000000000	0.000000000	0	0.000000000	73	0.5	43	2.4	64
132	0.006379585	0.000000000	0.006379585	0.003189793	0	0.000000000	17	2.1	0	10.4	130
133	0.003322259	0.000000000	0.008305648	0.003322259	0	0.000000000	16	2.1	0	13.4	130
134	0.002560819	0.000000000	0.007682458	0.002560819	0	0.000000000	16	2.4	0	23.0	117
132	0.006514658	0.000000000	0.008143322	0.000000000	0	0.000000000	16	2.4	0	19.9	117
134	0.001049318	0.000000000	0.010493179	0.009443861	0	0.002098636	26	5.9	0	0.0	150
134	0.001402525	0.000000000	0.012622721	0.008415147	0	0.002805049	29	6.3	0	0.0	150
122	0.000000000	0.000000000	0.000000000	0.000000000	0	0.000000000	83	0.5	6	15.6	68
122	0.000000000	0.000000000	0.001517451	0.000000000	0	0.000000000	84	0.5	5	13.6	68
122	0.000000000	0.000000000	0.002967359	0.000000000	0	0.000000000	86	0.3	6	10.6	68

Task 1

Task 1




```
data <- read.csv("C:/Users/Intellipaat-Ashwin/Desktop/CTG.csv", header = TRUE)
str(data)
data$NSP <- as.factor(data$NSP)
table(data$NSP)
```

Task 1


```
set.seed(123)
split_tag <- createDataPartition(data$NSP, p = 0.65, list = F)
data[split_tag,] -> train
data[-split_tag,] -> test
```

Task 1


```
library(randomForest)
set.seed(222)
rf<-randomForest(NSP~.,data=train)
rf
```

```
predict(rf,test) -> p1
table(test$NSP,p1)
```

Task 2

Task 2

Tune the model for optimal ‘mtry’ value

```
set.seed(100)
tuneRF(train[,-22], train[,22],
       stepFactor = 0.5,
       plot = TRUE,
       ntreeTry = 300,
       trace = TRUE,
       improve = 0.05)
```

Task 2

Build the model & predict the valules

```
set.seed(222)
rf <- randomForest(NSP~., data=train,ntree = 300,mtry = 8)
rf
```

```
predict(rf,test) -> p2
table(test$NSP,p2)
```

Task 2

Have a look at the variable importance

```
varImpPlot(rf,sort = T, n.var = 10,main = "Top 10 - Variable Importance")

importance(rf)
```

Quiz

Quiz

Q 1. Which of the following is/are true about bagging trees?

1. In bagging trees, individual trees are independent of each other
2. Bagging is the method for improving the performance by aggregating the results of weak learners

A) 1

B) 2

C) 1 and 2

D) None of these

Quiz

Q 2. In Random Forest, you can generate hundreds of trees (say, T1, T2, … Tn) and then aggregate the results of these tree. Which of the following are true about an individual (Tk) tree in Random Forest?

1. Individual tree is built on a subset of the features
2. Individual tree is built on all the features
3. Individual tree is built on a subset of observations
4. Individual tree is built on full set of observations

A) 1 and 3

B) 1 and 4

C) 2 and 3

D) 2 and 4

Quiz

Q 3. The data scientists at “BigMart Inc” have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product based on these attributes and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store during a defined period.

Which learning problem does this belong to?

A. Supervised learning

B. Unsupervised learning

C. None

Thank You

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor