Random Forest

Dr. Dipak Kumar Mohanty
KIIT University, Bhubaneswr

Random Forest

Objective

Things you will learn at the end of this session:

- What is Classification?
- What is Random Forest?
- Why Random Forest?
- Introduction to Ensemble method
- Random Forest analogy
- How to use Random Forest?
- Hands-On

What is Classification?

"Classification is the process of **grouping things according to similar features** they share"

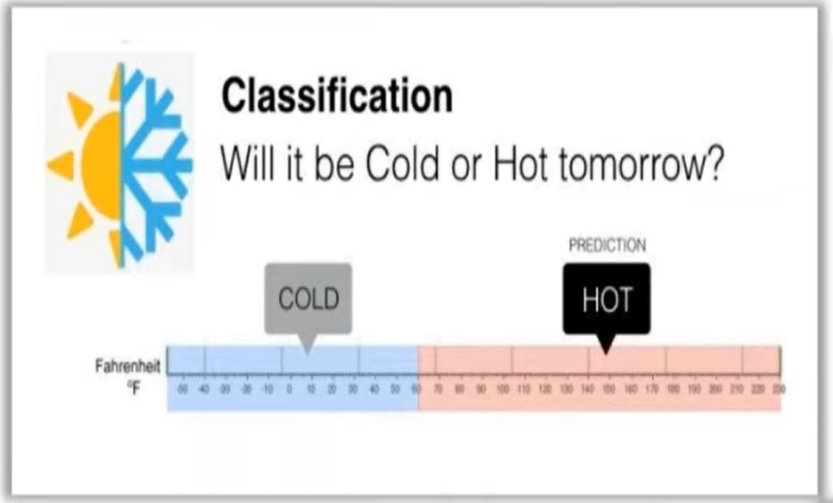
What is Classification?

"Classification is the process of **grouping things according to similar features** they share"

Class

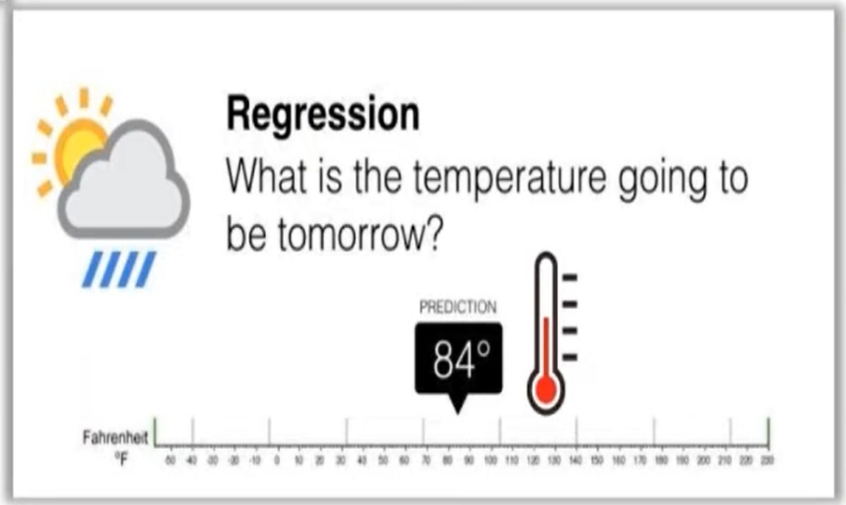
Supervised Learning

Classification vs Regression



Predicts categorical outcome

Predicts continuous outcome



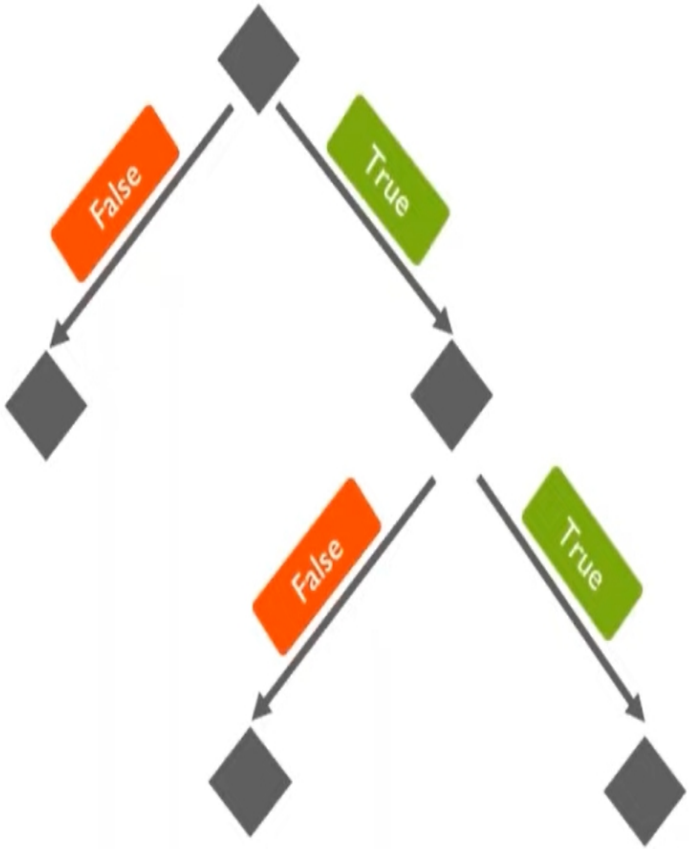
Examples of Classification Algorithm

- Logistic Regression
- Decision Tree
- Random Forest
- K- Nearest Neighbor
- Naïve Bayes
- Many More...

What is Random Forest?

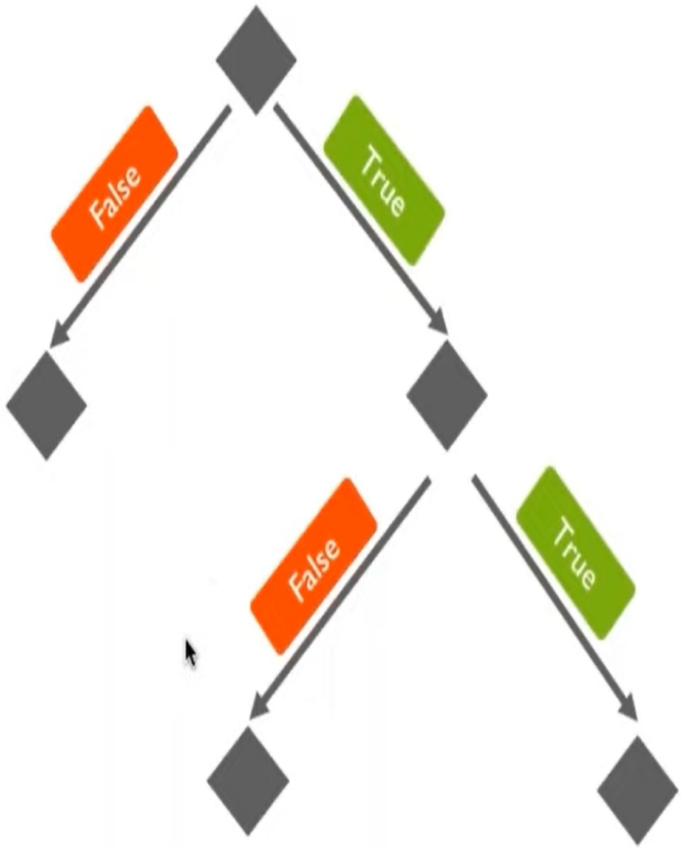
Random Forest is the most used supervised machine learning algorithm for classification and regression

Random Forest are made out of decision trees



Why Random Forest?

Decision Trees are easy to build and interpret.



Then **WHY** Random Forest?

Decision Trees have only **one aspect,**

therefore are **less accurate and inflexible**

when it comes to **classifying new samples**

Decision Trees also tend to **overfit**

Why Random Forest?

No overfitting

Use of multiple trees reduce the risk of overfitting

Training time is less

High accuracy

Runs efficiently on large database

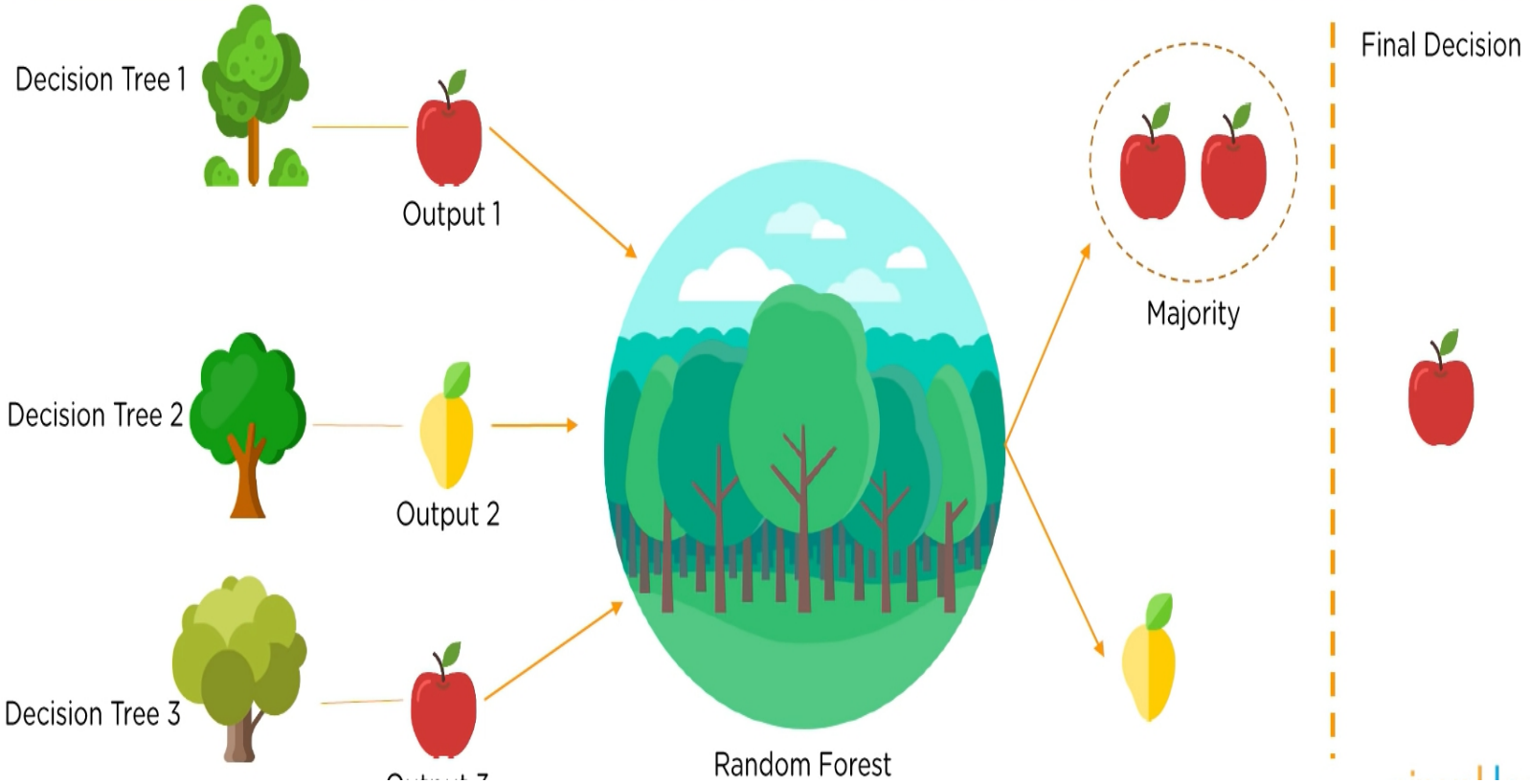
For large data, it produces highly accurate predictions

Estimates missing data

Random Forest can maintain accuracy when a large proportion of data is missing

What is Random Forest?

Random forest or Random Decision Forest is a method that operates by constructing multiple Decision Trees during training phase.
The Decision of the majority of the trees is chosen by the random forest as the final decision



Random Forest Analogy

Every friends gave suggestion by asking him few questions

Trip Suggestion I

Trip Suggestion II

Trip Suggestion III

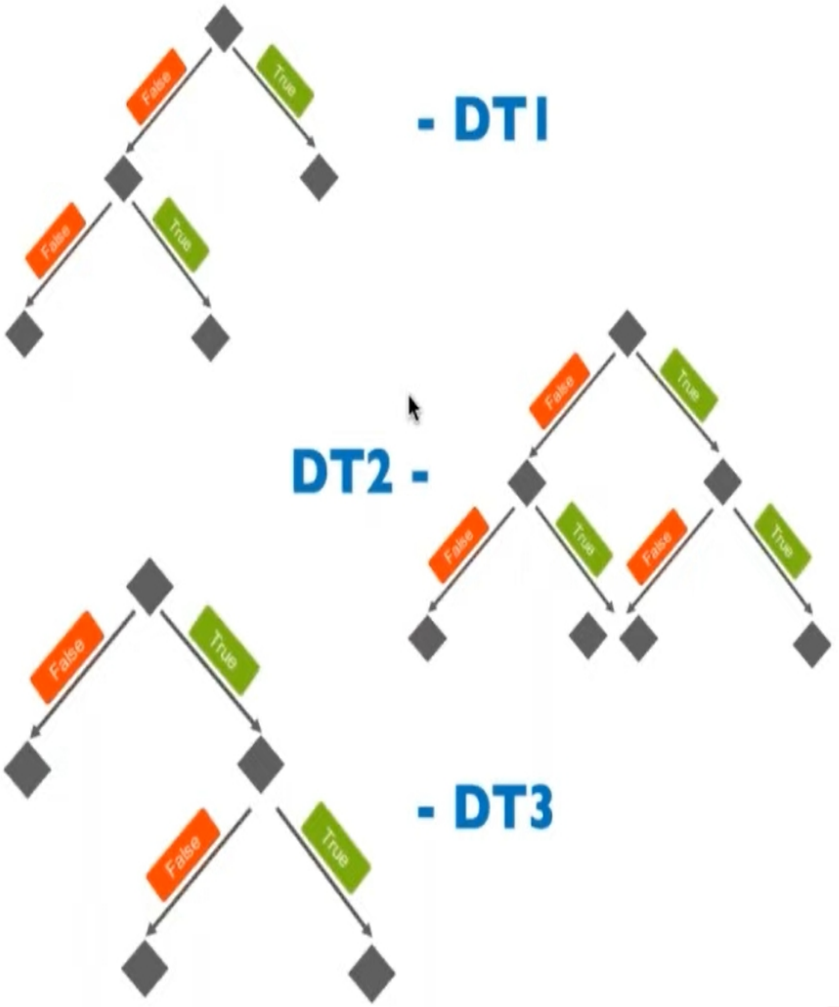
Random Forest Analogy

Later on, Chandler asked more of his friends to advise him. Once again, his friends asked him different questions to recommend about the places.

Now after talking to all of his friends he decide to visit the place with most number of votes. the above scenario is a typical example of **Random Forest Algorithm**.

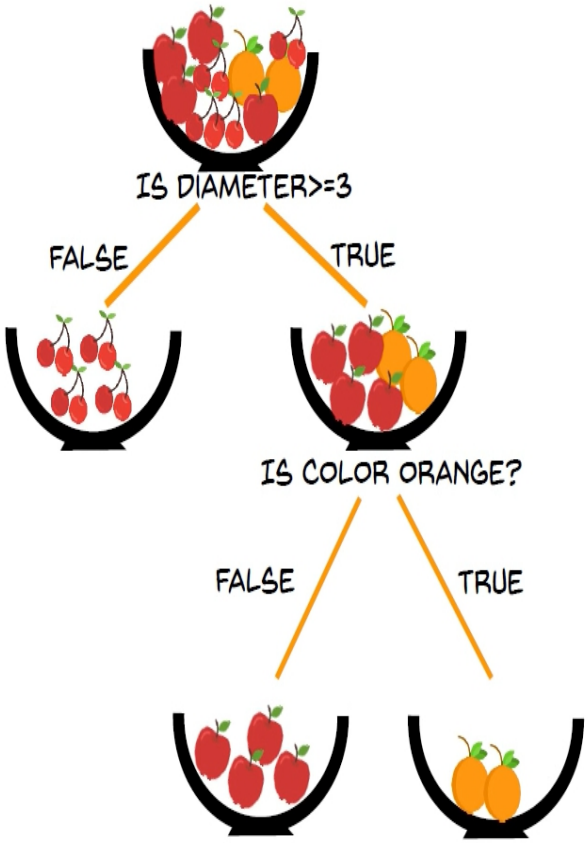
What is Random Forest?

Let's build a Random Forest...

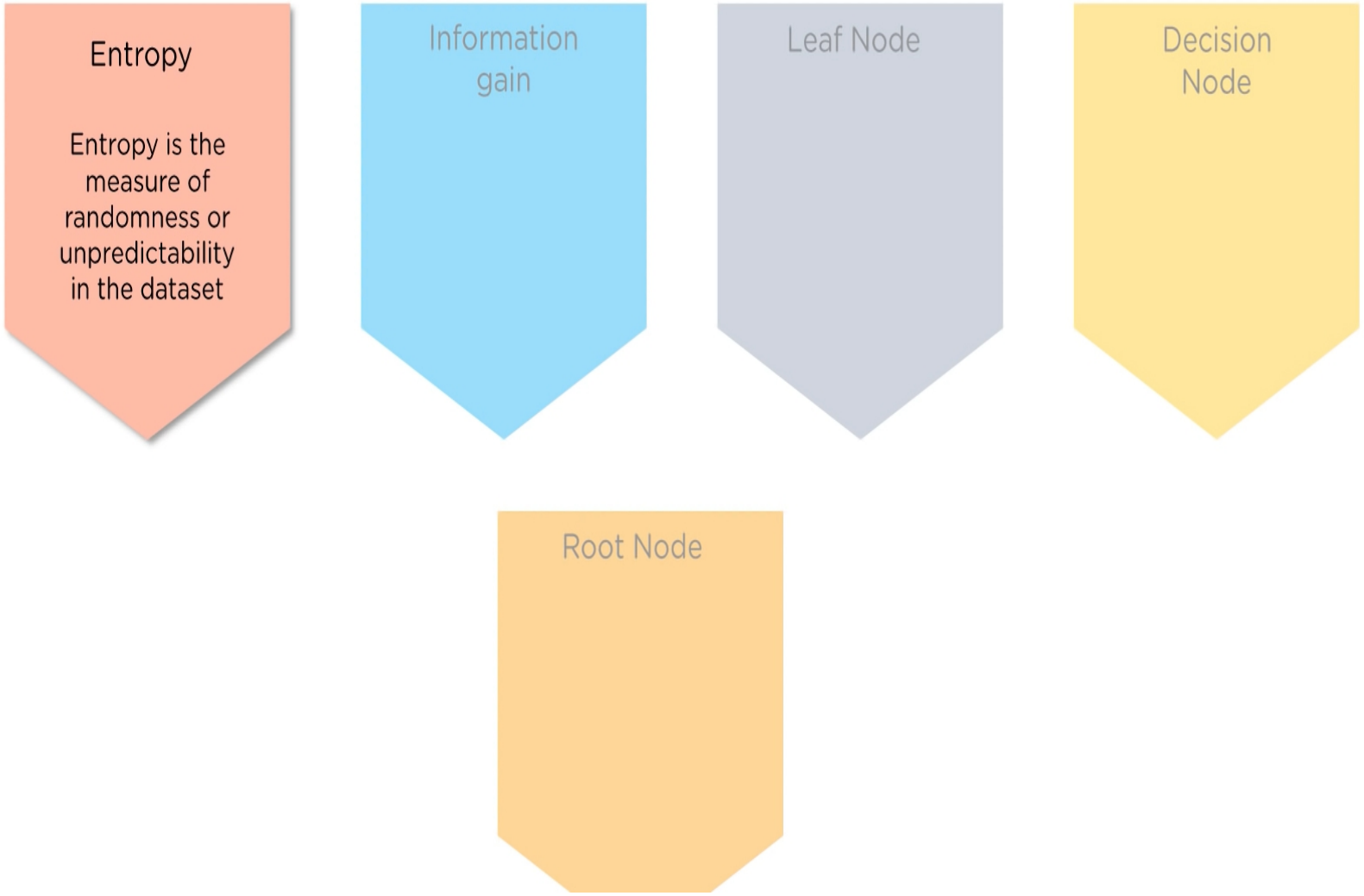


Decision Tree

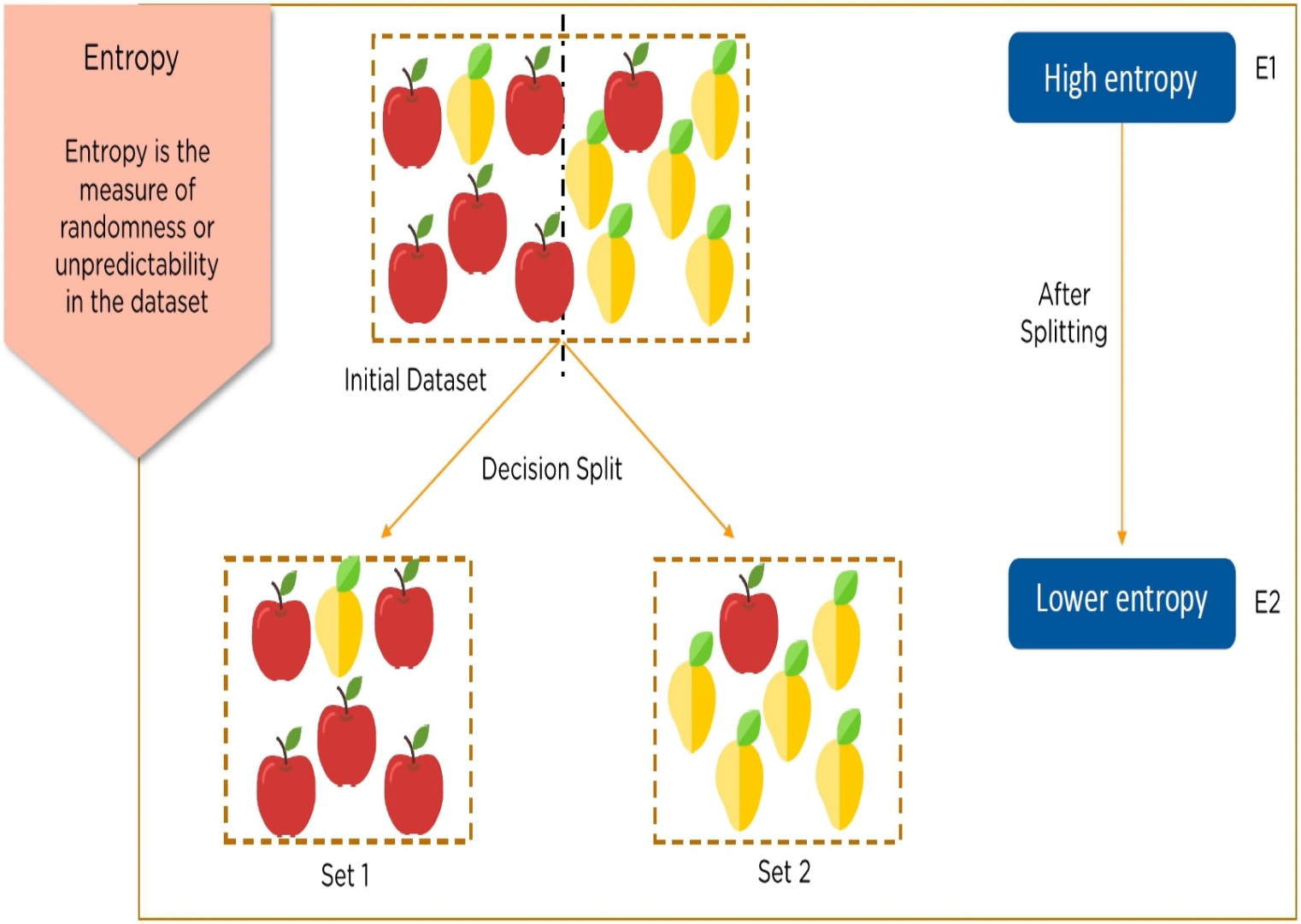
Decision Tree is a tree shaped diagram used to determine a course of action. Each branch of the tree represents a possible decision, occurrence or reaction



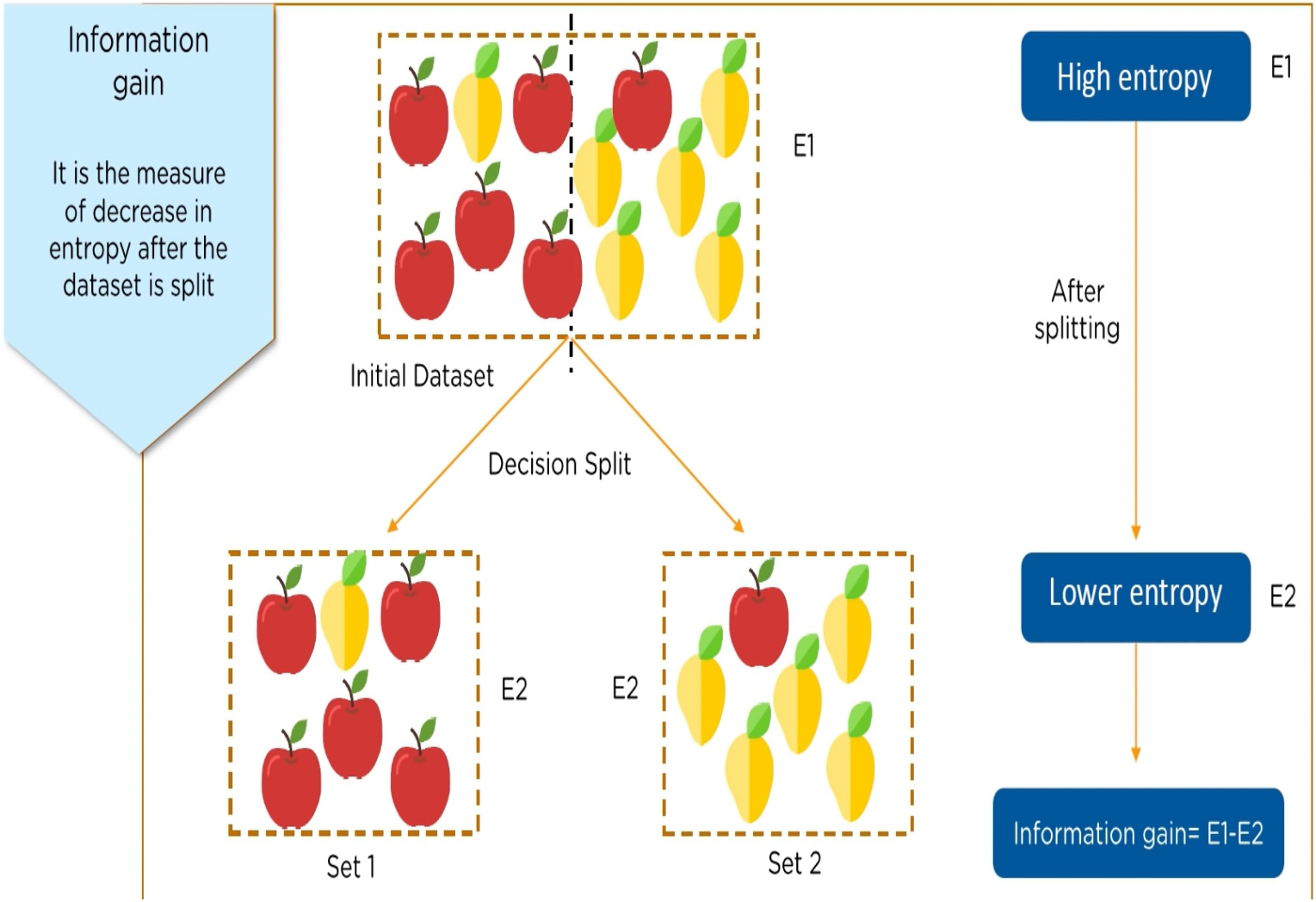
Decision Tree- Important Terms



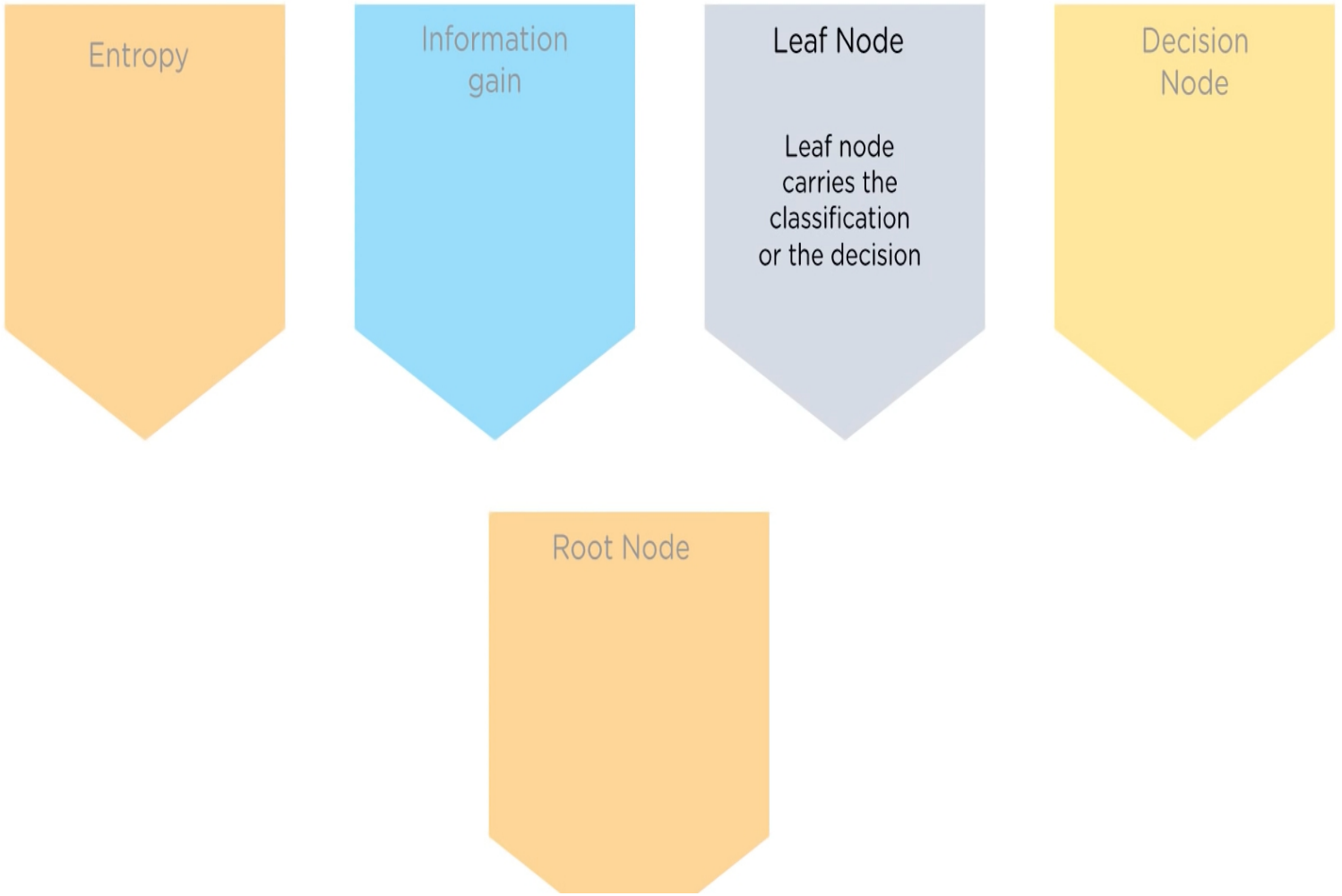
Decision Tree - Important Terms



Decision Tree - Important Terms



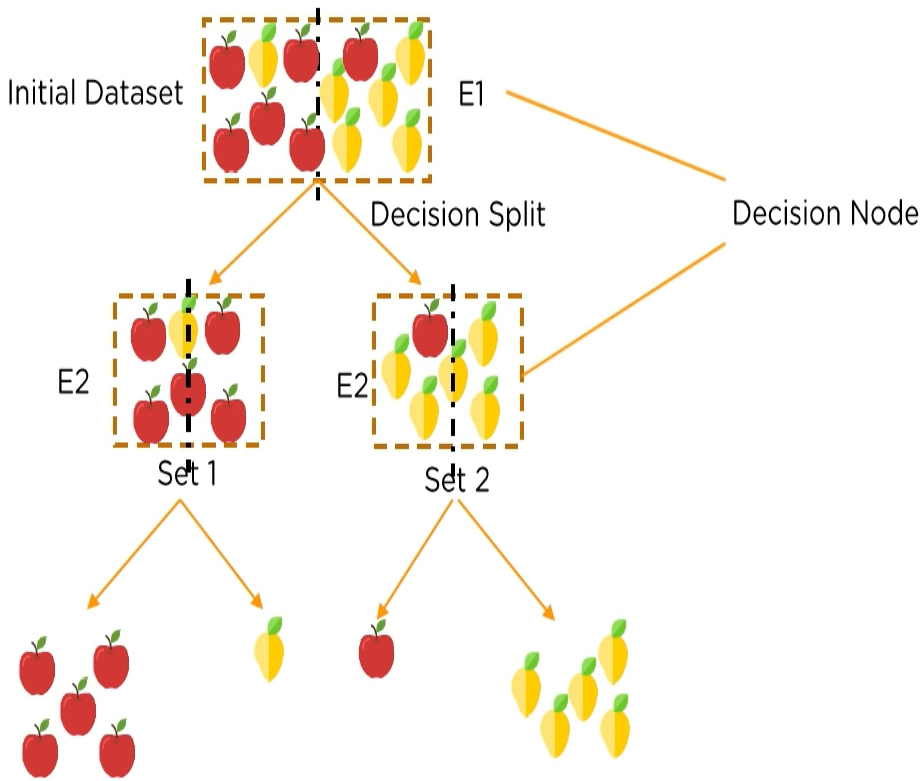
Decision Tree - Important Terms



Decision Tree - Important Terms

Decision Node

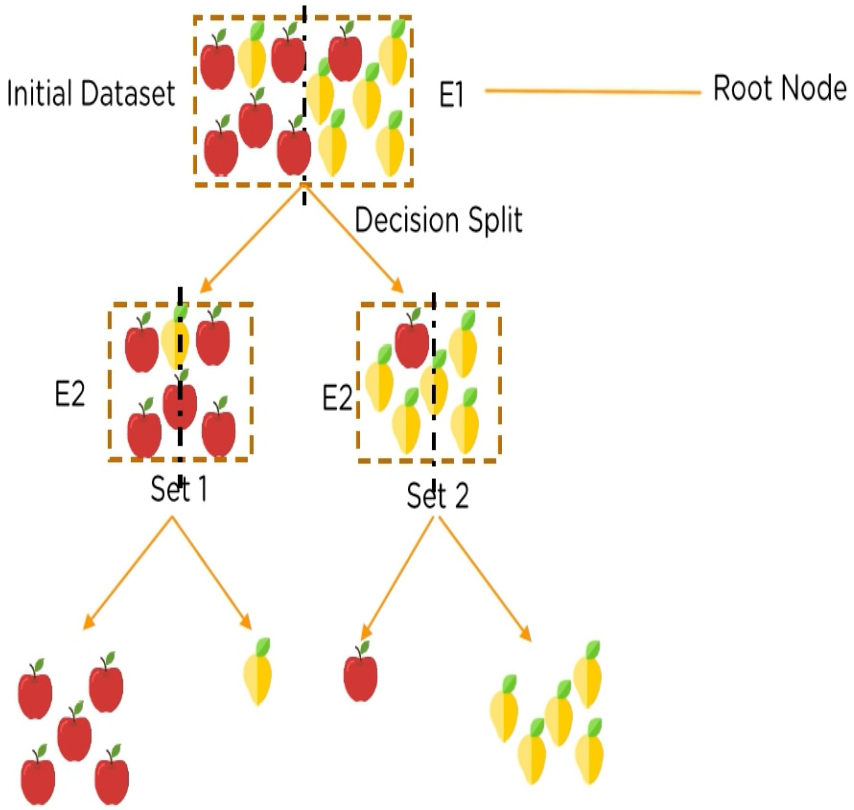
Decision node has two or more branches



Decision Tree - Important Terms

Root Node

The top most Decision node is known as the Root node



How does a Decision Tree work?

PROBLEM STATEMENT

TO CLASSIFY THE DIFFERENT TYPES OF FRUITS IN THE BOWL BASED ON DIFFERENT FEATURES

THE DATASET(BOWL) IS LOOKING QUITE MESSY AND THE ENTROPY IS HIGH IN THIS CASE

TRAINING DATASET

COLOR	DIAMETER	LABEL
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES

How does a Decision Tree work?

HOW TO SPLIT THE DATA

WE HAVE TO FRAME THE CONDITIONS THAT SPLIT THE DATA IN SUCH A WAY THAT THE INFORMATION GAIN IS THE HIGHEST

NOTE

GAIN IS THE MEASURE OF DECREASE IN ENTROPY AFTER SPLITTING

How does a Decision Tree work?

NOW WE WILL TRY TO CHOOSE A CONDITION THAT GIVES US THE HIGHEST GAIN

WE WILL DO THAT BY SPLITTING THE DATA USING EACH CONDITION AND CHECKING THE GAIN THAT WE GET OUT THEM.

How does a Decision Tree work?

CONDITIONS

COLOR== PURPLE?

DIAMETER=3

COLOR== YELLOW?

COLOR== RED?

DIAMETER=1

TRAINING DATASET

COLOR	DIAMETER	LABEL
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES

How does a Decision Tree work?

CONDITIONS

COLOR== PURPLE?

DIAMETER=3

COLOR== YELLOW?

COLOR== RED?

DIAMETER=1

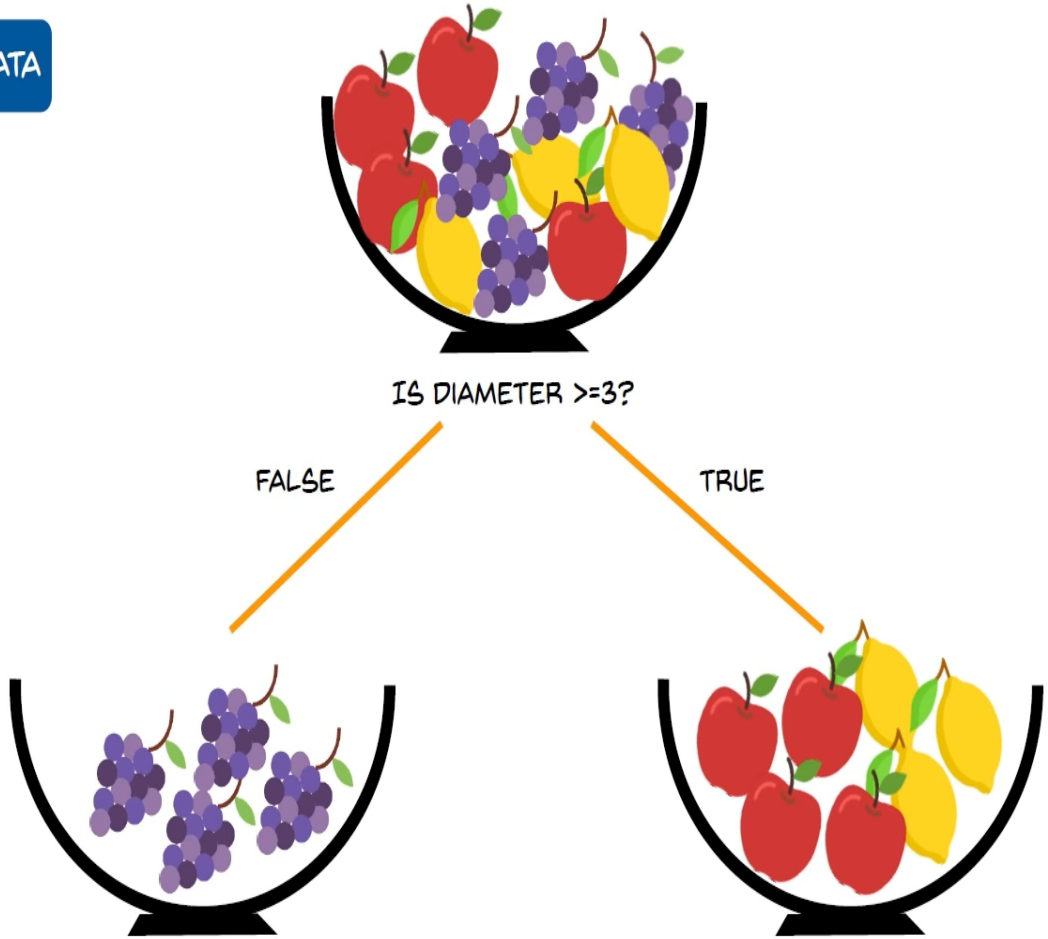
LET'S SAY THIS CONDITION GIVES US THE MAXIMUM GAIN

TRAINING DATASET

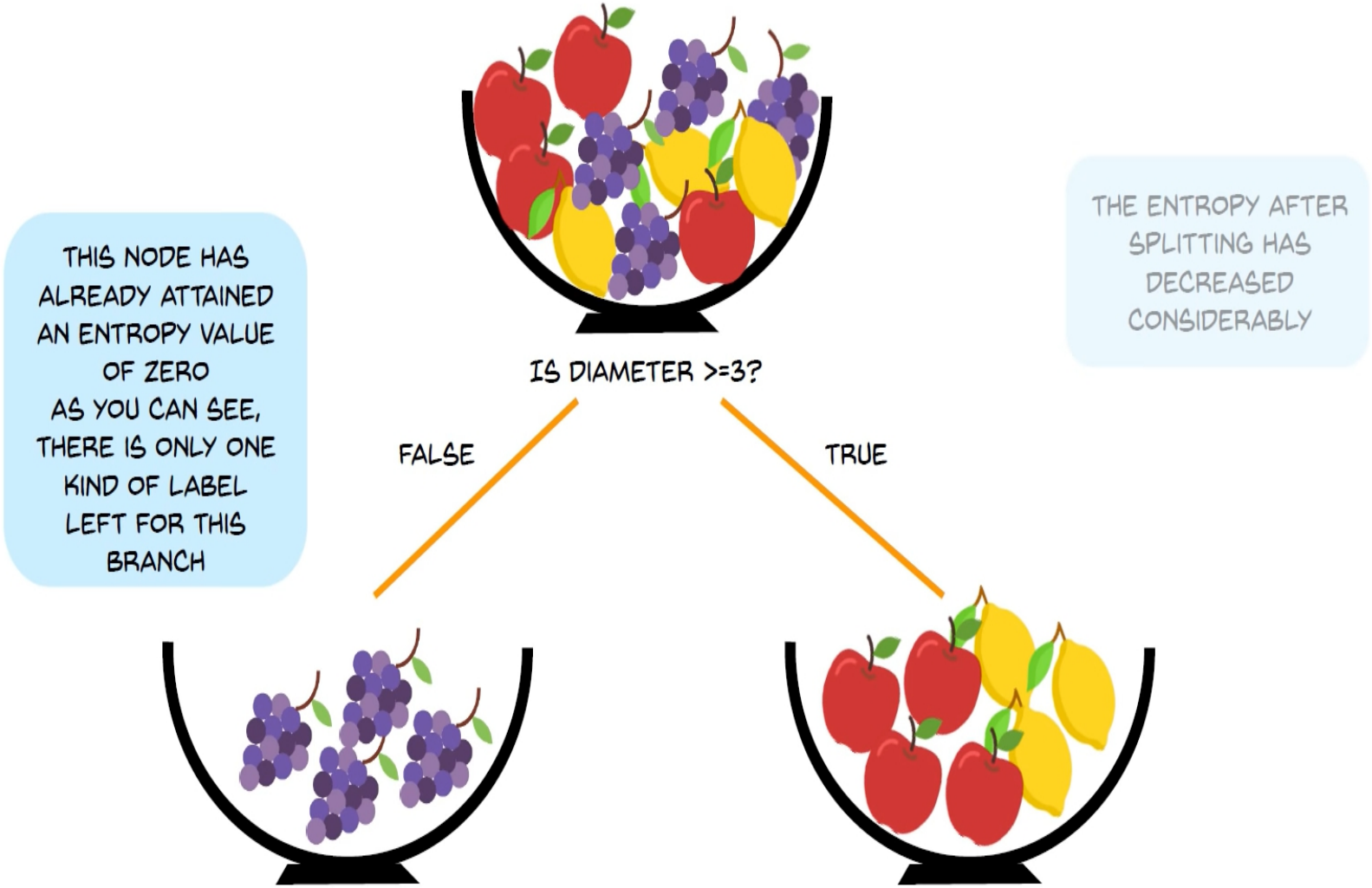
COLOR	DIAMETER	LABEL
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES
RED	3	APPLE
YELLOW	3	LEMON
PURPLE	1	GRAPES

How does a Decision Tree work?

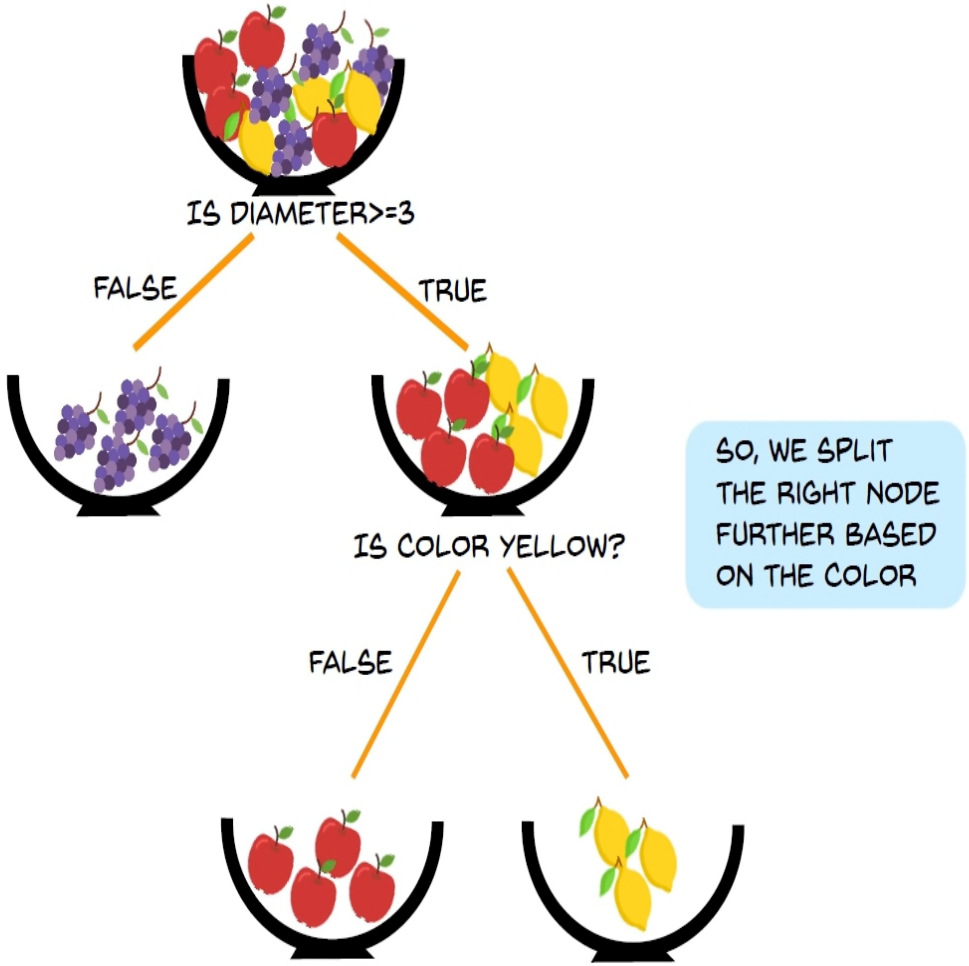
WE SPLIT THE DATA



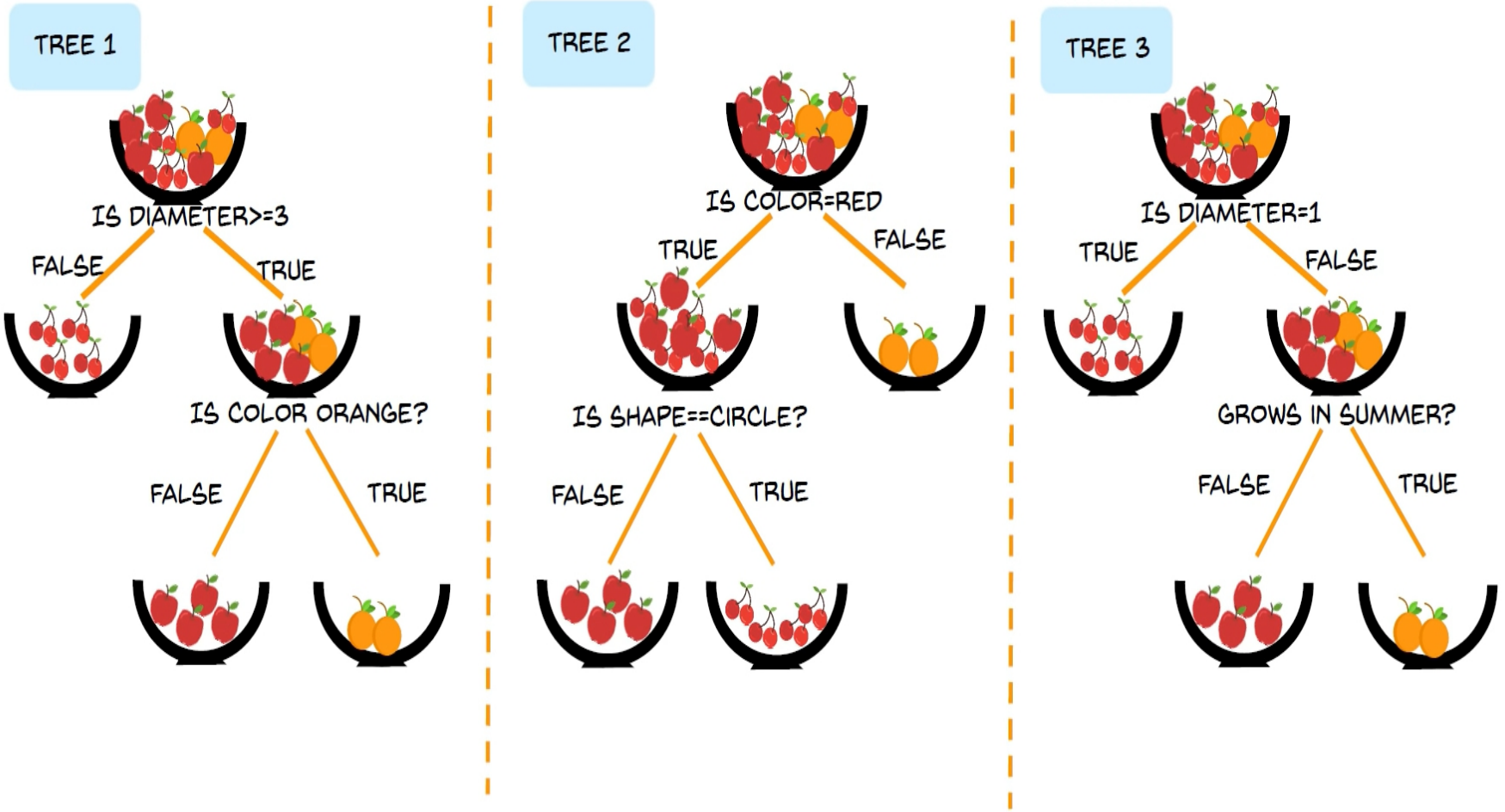
How does a Decision Tree work?



How does a Decision Tree work?



How does a Random Forest work?



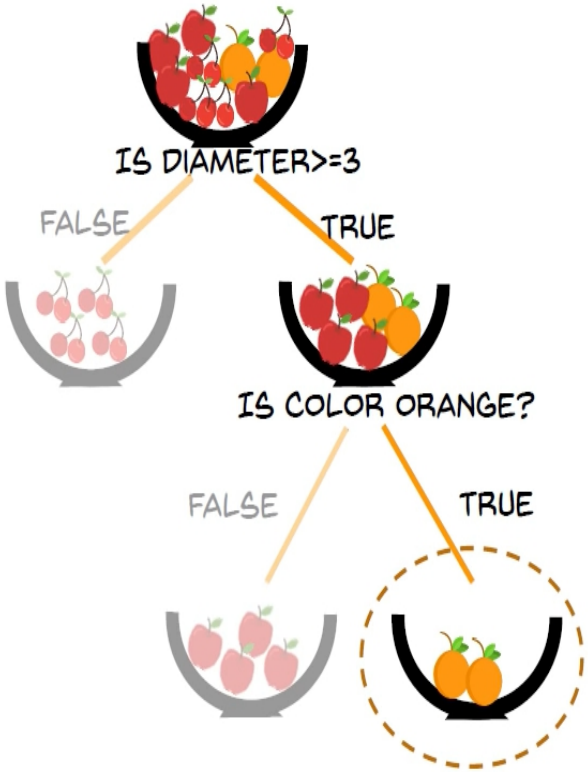
How does a Random Forest work?

NOW LETS TRY TO CLASSIFY THIS FRUIT

How does a Random Forest work?

TREE 1 CLASSIFIES IT AS AN ORANGE

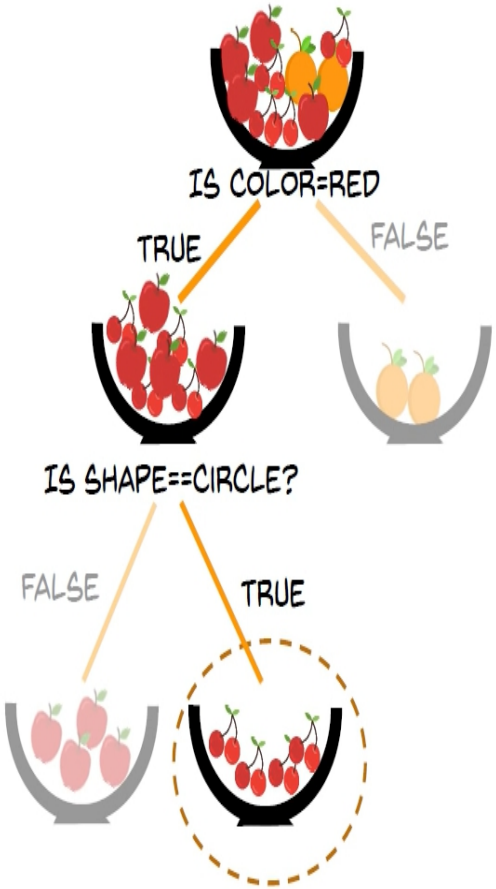
DIAMETER = 3
COLOUR = ORANGE
GROWS IN SUMMER = YES
SHAPE = CIRCLE



How does a Random Forest work?

TREE 2 CLASSIFIES IT AS CHERRIES

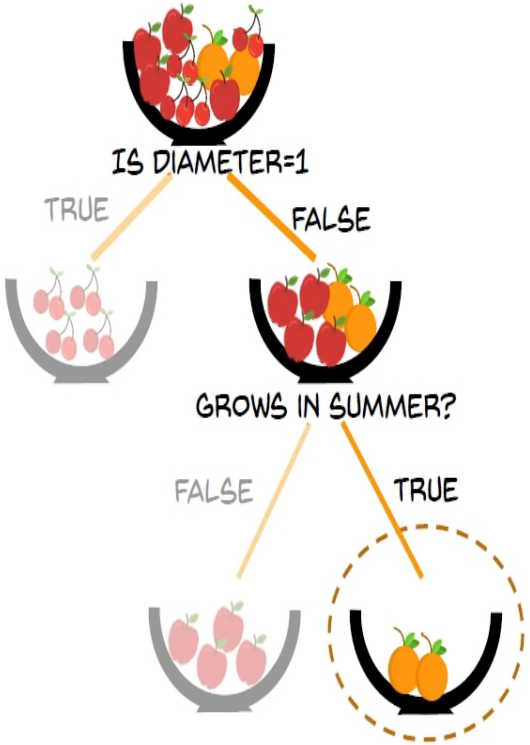
DIAMETER = 3
COLOUR = ORANGE
GROWS IN SUMMER = YES
SHAPE = CIRCLE



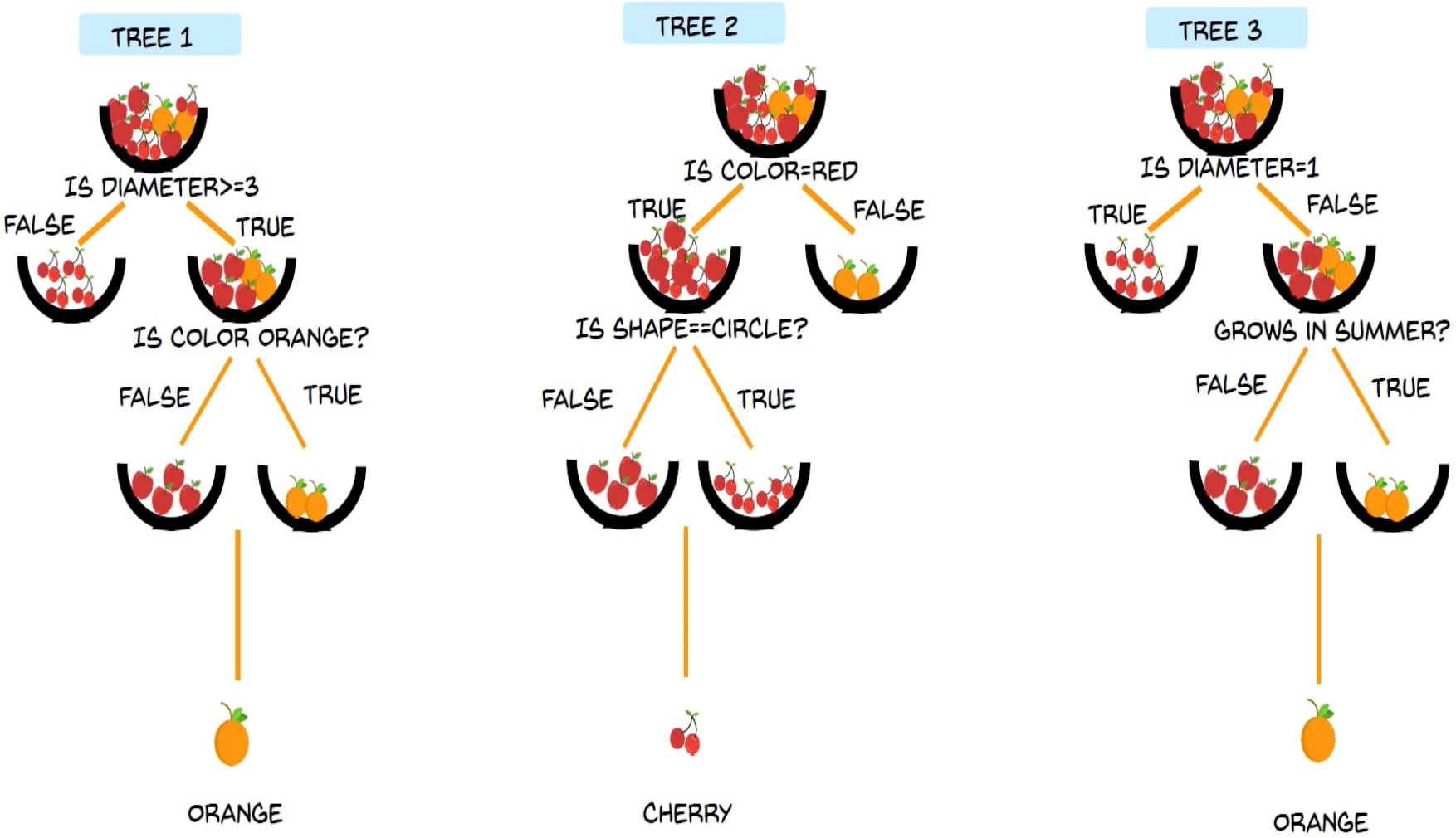
How does a Random Forest work?

TREE 3 CLASSIFIES IT AS ORANGE

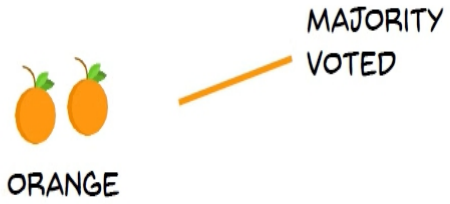
DIAMETER = 3
COLOUR = ORANGE
GROWS IN SUMMER = YES
SHAPE = CIRCLE



How does a Random Forest work?



How does a Random Forest work?



CHERRY

How does a Random Forest work?

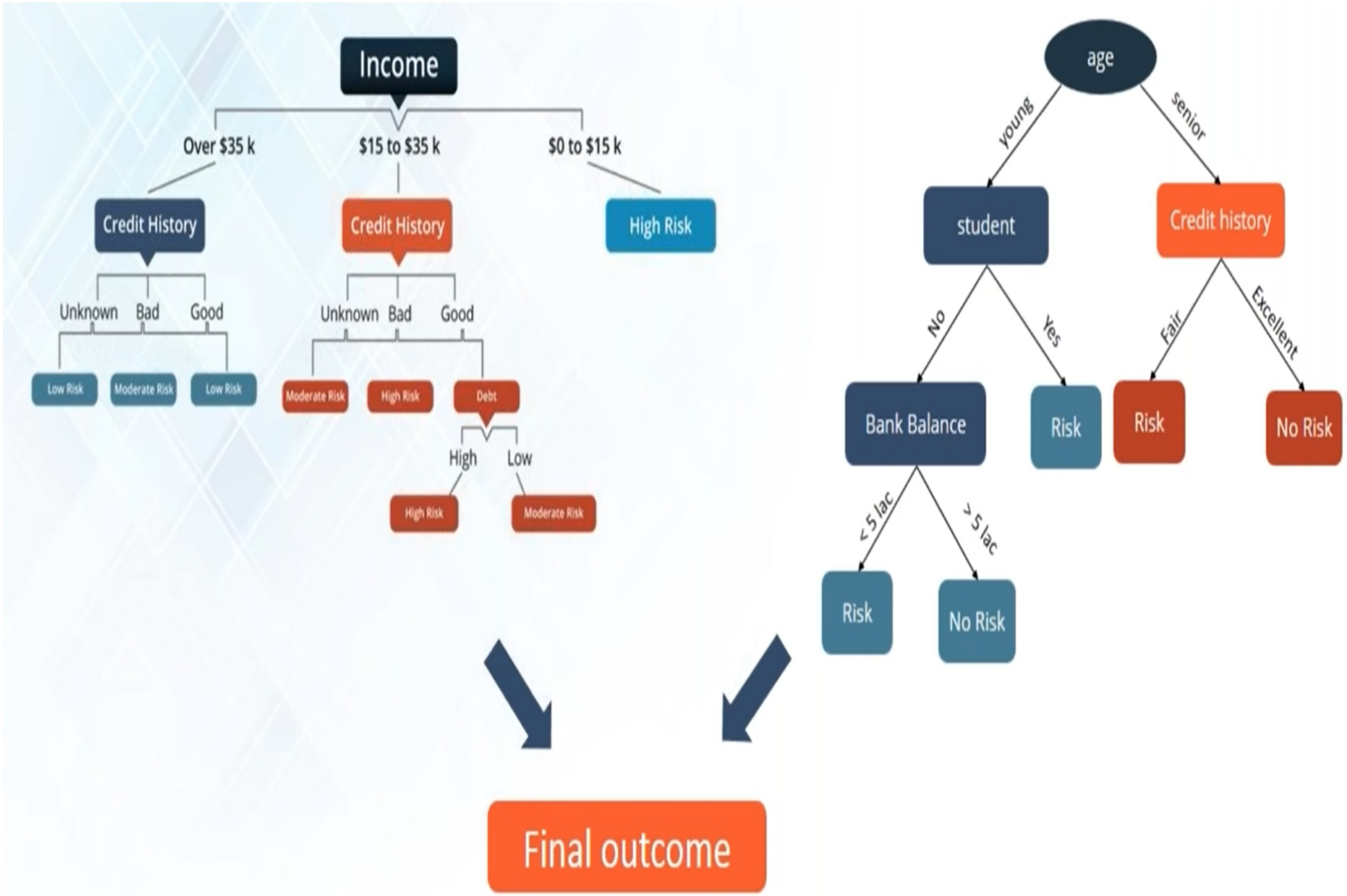
SO THE FRUIT IS CLASSIFIED AS AN ORANGE

Use Case- Credit Risk Detection

- To minimize loss, the bank needs a decision rule to predict whom to give approval of the loan.
- An applicant's demographic (income, debts, credit history) and socio-economic profiles are considered.
- Data science can help banks recognize behavior patterns and provide a complete view of individual customers.

Variable	Measurement
Marital Status	Married, Not Married
Gender	Male, Female
Age	Varied
Status	Default, Non Default
Time of Payment	Varied
Employment	Employed, Unemployed
Homeownership	With Home, Without Home
Education Level	Secondary and above, Below secondary

Here we have 2 decision trees



Random Forest Algorithm

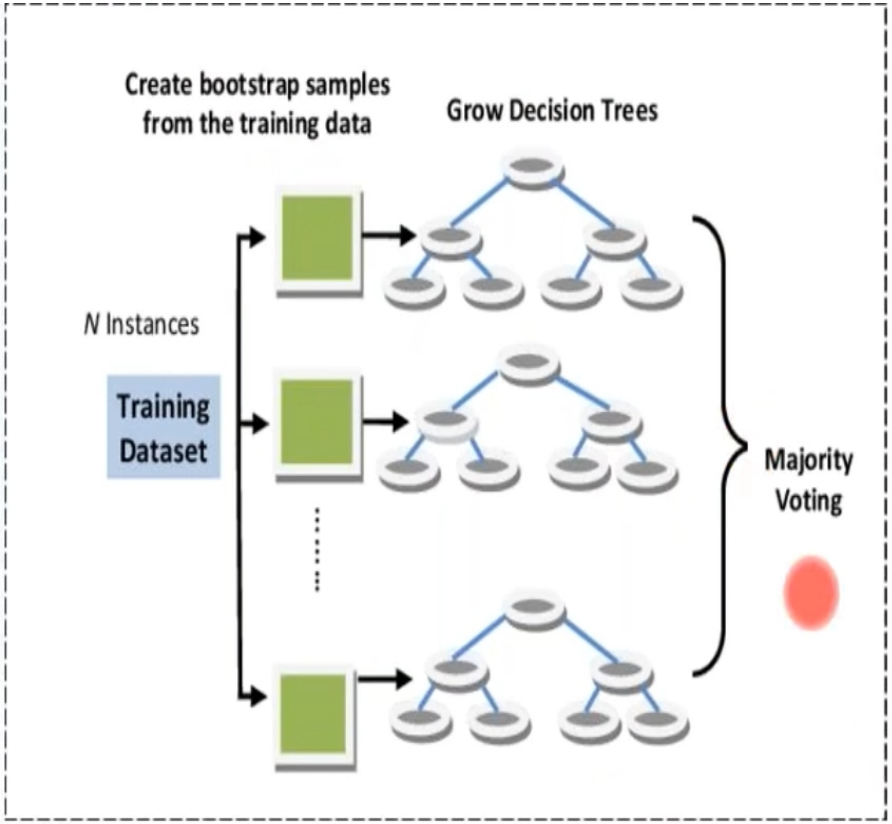
Randomly select m features from T;
where $m \ll T$

For node d, calculate the best split point among the m feature

Split the node into two daughter nodes using the best split

Repeat first three steps until n number of nodes has been reached

Build your forest by repeating steps i–iv for D number of times



- ✓ T: number of features
- ✓ D: number of trees to be constructed
- ✓ V: Output: the class with the highest vote

Example:-

Let's take an example,

We have taken dataset consisting of:

- Weather information of last 14 days
- Whether match was played or not on that particular day

Now using the random forest we need to predict whether the game will happen if the weather condition is

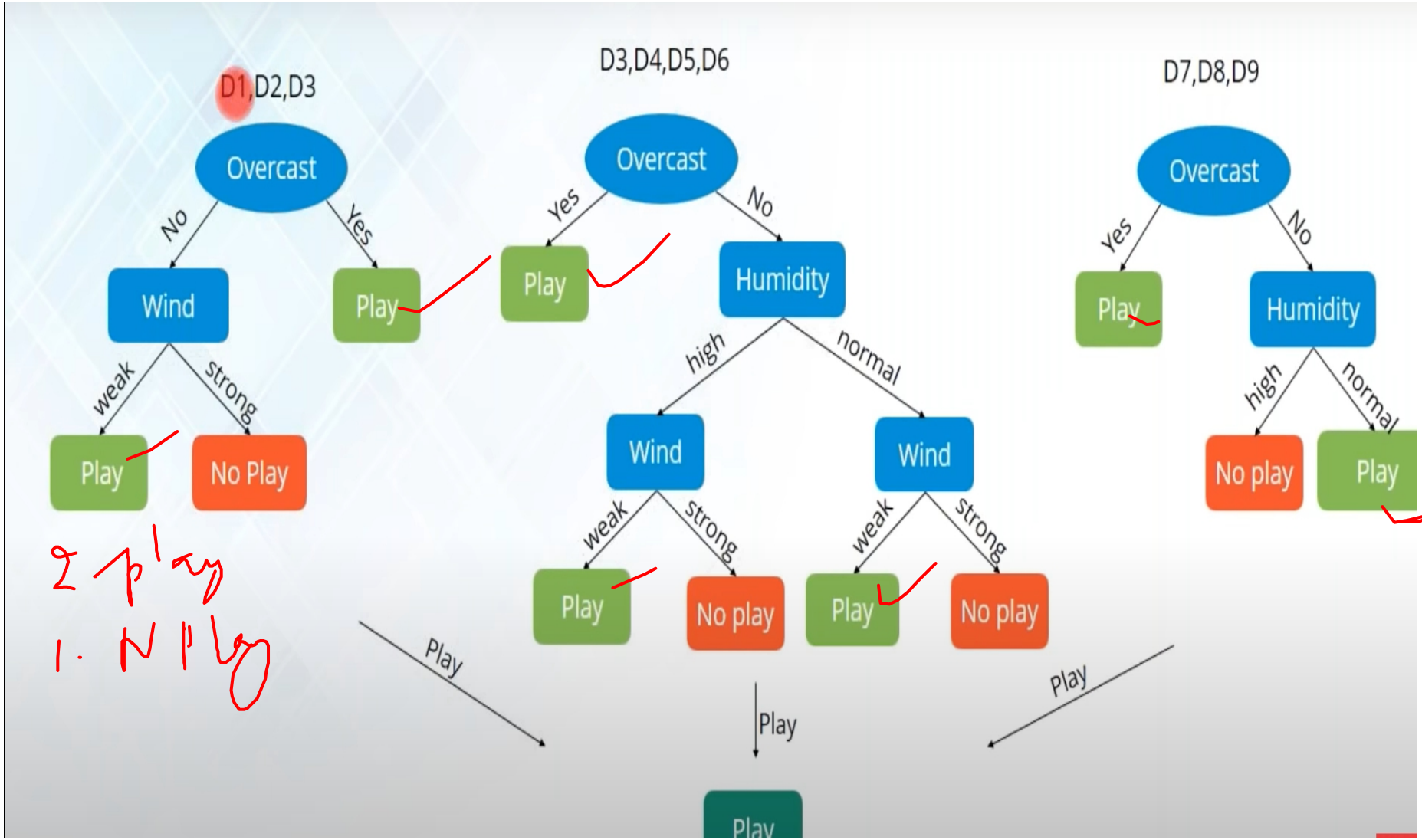
Outlook	=	Rain
Humidity	=	High
Wind	=	Weak
Play	=	**?**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

- The first step in Random forest is that it will divide the data into smaller subsets.
- Every subsets need not be distinct, some subsets maybe overlapped

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Here, let we have 3 subsets (DTs)



Use Case:- Diabetes Prediction

- Data Acquisition
- Divide dataset
- Implement model
- Visualize
- Model Validation

Doctor gets the following data from the medical history of the patient.

glucose_conc	blood_pressure	skin_fold_thickness	X2.Hour_serum_insulin	BMI	Diabetes_pedigree_fn	Age	Is_Diabetic
148	72	35	0	33.6	0.627	50	YES
85	66	29	0	26.6	0.351	31	NO
183	64	0	0	23.3	0.672	32	YES
89	66	23	94	28.1	0.167	21	NO
137	40	35	168	43.1	2.288	33	YES
116	74	0	0	25.6	0.201	30	NO
78	50	32	88	31.0	0.248	26	YES
115	0	0	0	35.3	0.134	29	NO
197	70	45	543	30.5	0.158	53	YES
125	96	0	0	0.0	0.232	54	YES

- Data Acquisition
- **Divide dataset**
- Implement model
- Visualize
- Model Validation

We will divide our entire dataset into two subsets as:

- Training dataset -> to train the model
- Testing dataset -> to validate and make predictions

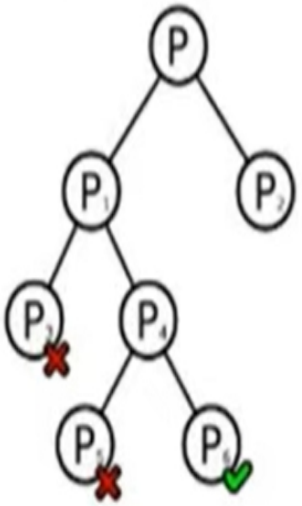
```
#load the data into R
diabet<-read.csv("Diabetes.csv")
View(diabet)

#divide the data into Training and Testing datasets
#instead of using library caTools and splitting the data, we can split data by
set.seed(3)
id<-sample(2,nrow(diabet),prob = c(0.7,0.3),replace = TRUE)
diabet_train<-diabet[id==1,]
diabet_test<-diabet[id==2,]
```

Types of Classifiers

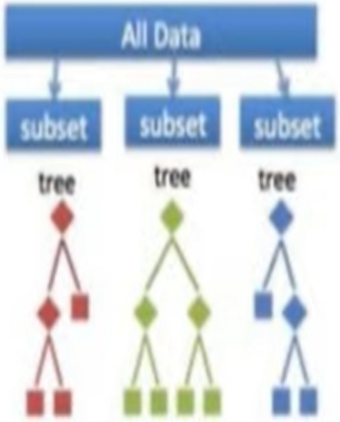
Decision Tree

- Decision tree builds classification models in the form of a tree structure.
- It breaks down a dataset into smaller and smaller subsets.



Random Forest

- Random Forest is an ensemble classifier made using many decision tree models.
- Ensemble models combine the results from different models.



Naïve Bayes

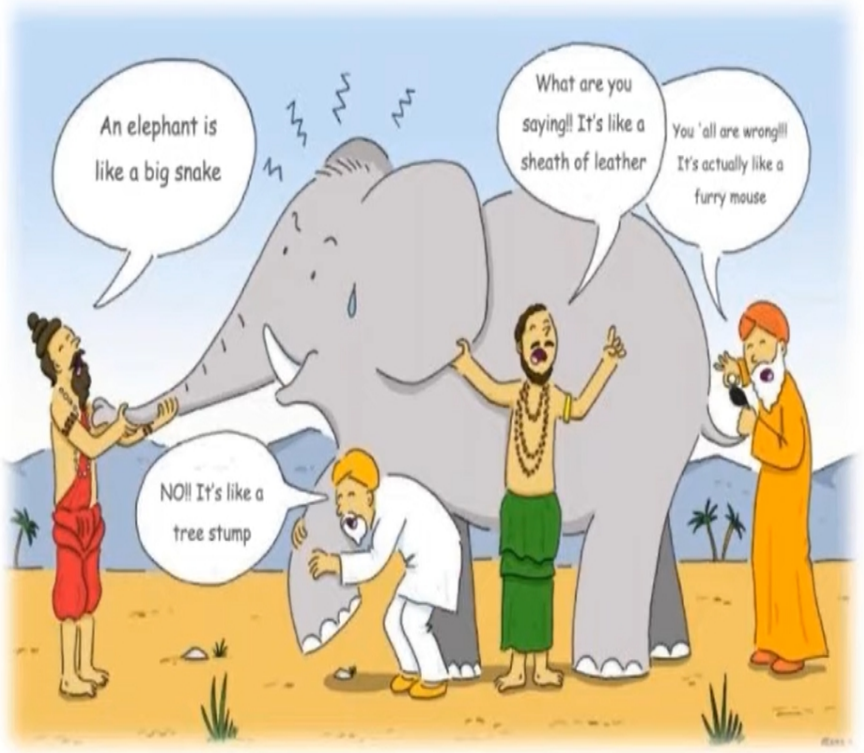
- It is a classification technique based on Bayes' Theorem with an assumption of independence among attributes.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Terminology Alert - What is Ensemble Learning?

Ensemble Learning

Random forest uses Ensemble learning method in which the predictions are based on the **combined results** *of various individual models*



Fable of blind men and an elephant

Terminology Alert !!

Introduction to Ensemble techniques

- What are they?
  - Create multiple predictors for the same problem and make predictions by aggregating the predictions resulting from the predictors.
  - Different versions of the same model vs Multiple Classifier Systems
- Why do this?
  - Superior predictive capacity
  - Flexibility of not being constrained rigidity of the base predictor
- What works?
  - Better results even if the base predictor is poor.
  - Diversity among predictors.

Common Ensemble Techniques

- Bagging (Bootstrap Aggregating)
- Boosting
- Stacking and choosing

- Random Forests
  - Essentially use Bagging on Trees (bagged trees)
  - Random Subspace method
  - Random split selection

Ensemble Method: Terminology Alert - Bagging

Bagging

Bootstrapping the data and using its aggregate to make a decision is known as **Bagging.**

In other words, **Bagging is** training a bunch of individual models **parallelly**, and each model is trained by a random subset of the data.

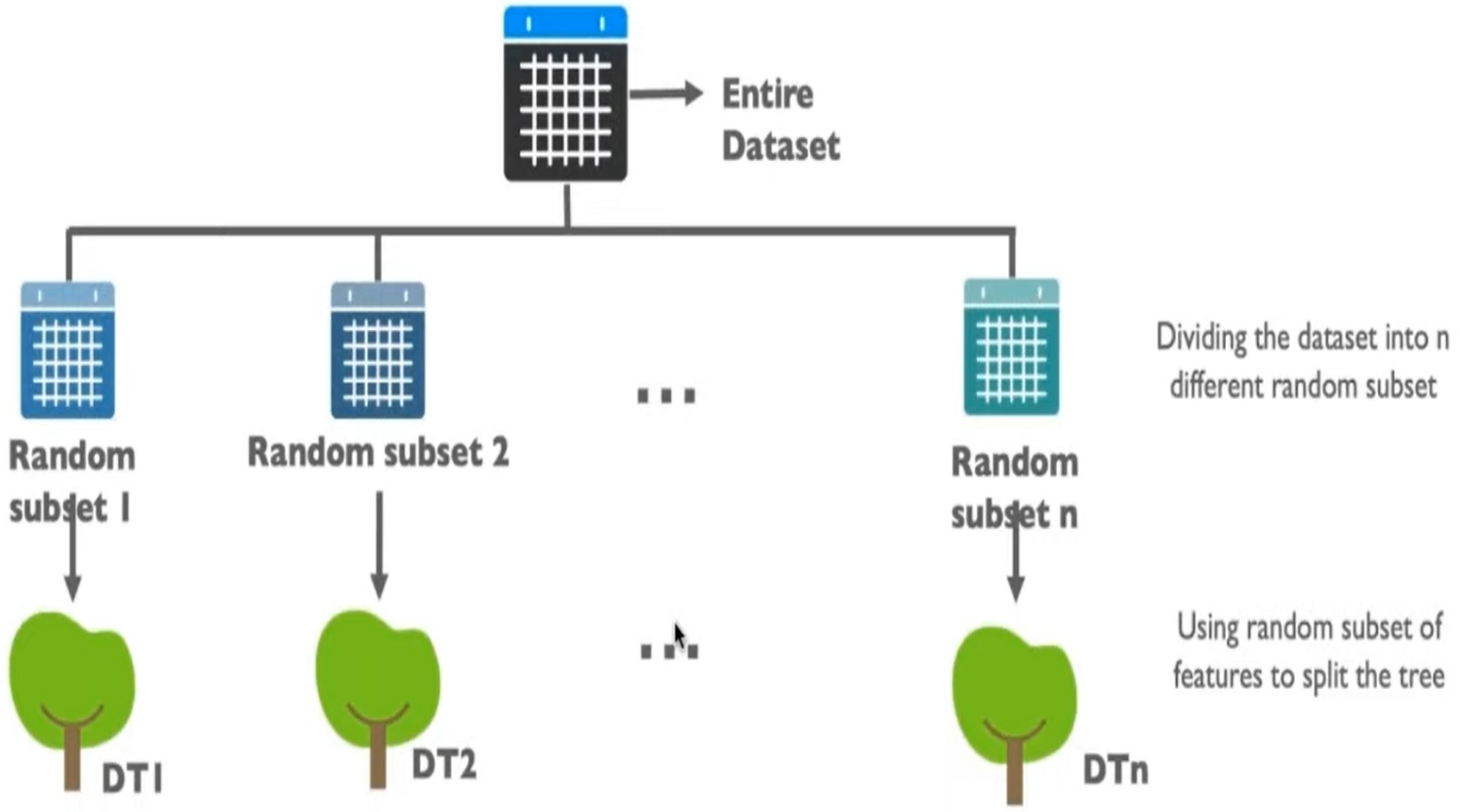
Bootstrapped Dataset

Chest Pain	Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	169	YES

Vote Count

Heart Disease	
YES	NO
95	5

Ensemble Method - Bagging



Bagging

Ensemble Method: Terminology Alert - Boosting

Boosting

Boosting is training a bunch of individual models in a **sequential** way. Each individual model learns from mistakes made by the previous model.

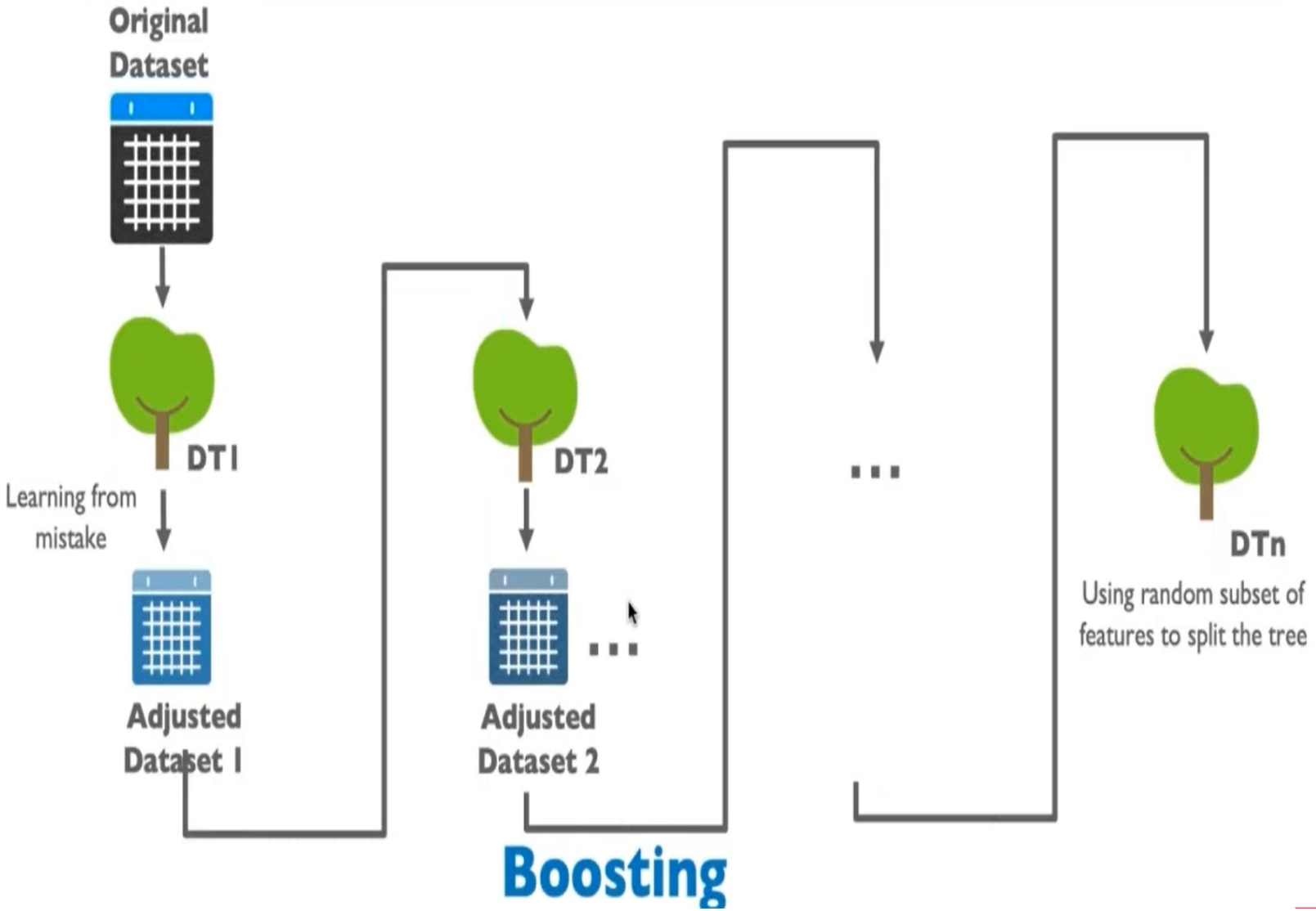
Bootstrapped Dataset

Chest Pain	Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	169	YES

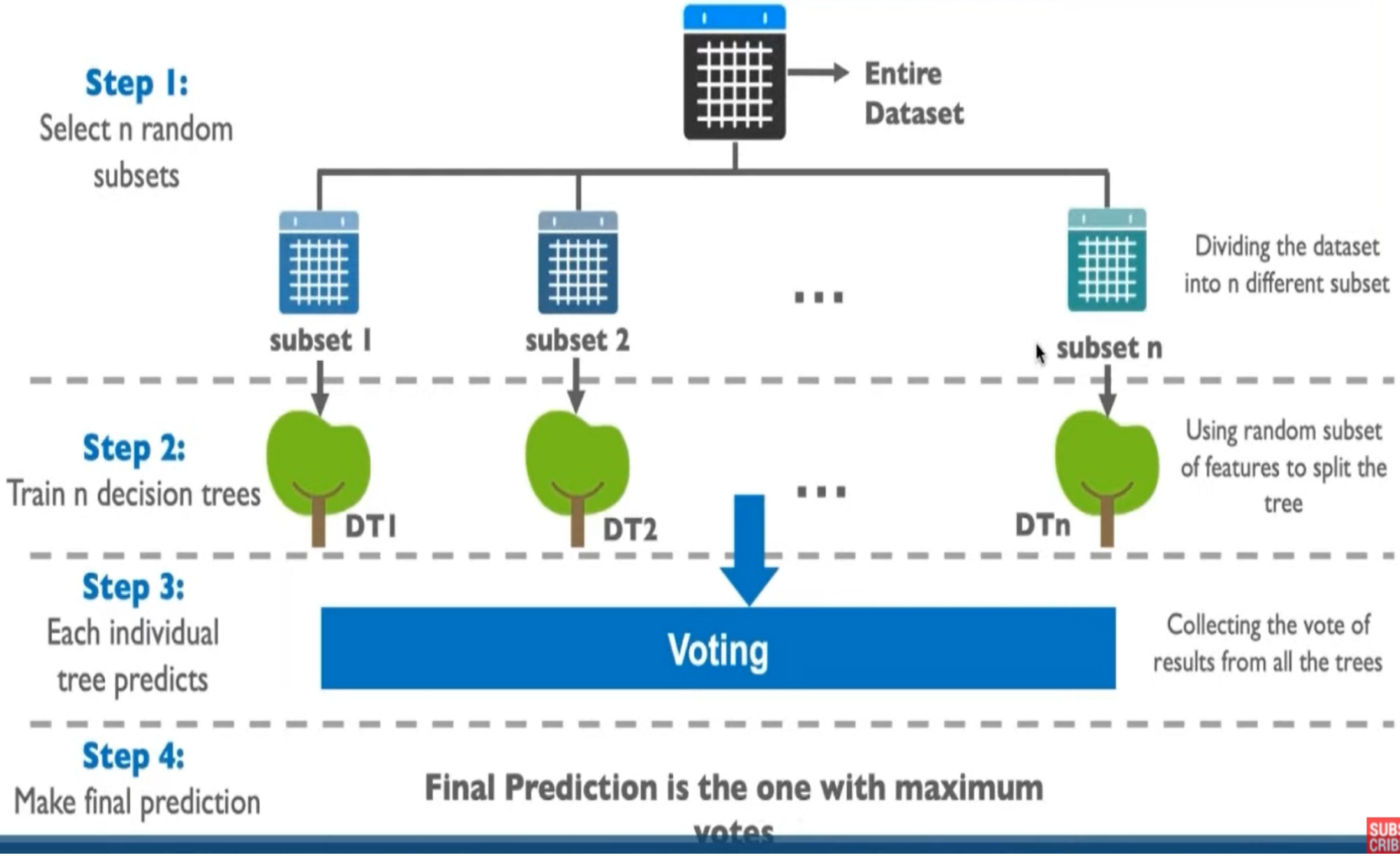
Vote Count

Heart Disease	
YES	NO
95	5

Ensemble Method - Boosting



Creating a Random Forest: Steps Involved



Step 1: Create a Bootstrap Dataset

Bootstrapped Dataset

The **bootstrap** method is a resampling technique used to estimate statistics on a population by sampling a **dataset** with replacement. It can be used to estimate summary statistics such as the mean or standard deviation.

The **bootstrap** dataset (same size as original) is created by randomly selecting samples from the original dataset.

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

This is our sample dataset...

NOTE: You can pick the same sample more than once

Step 1: Create a Bootstrap Dataset

Original Dataset

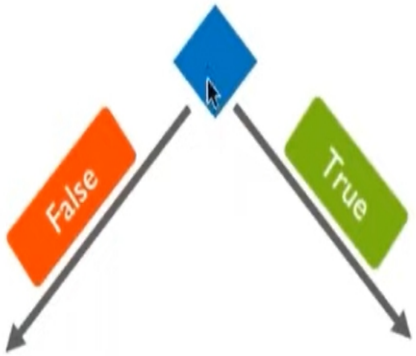
Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

Creating a bootstrapped dataset with randomly selected sample from the original dataset

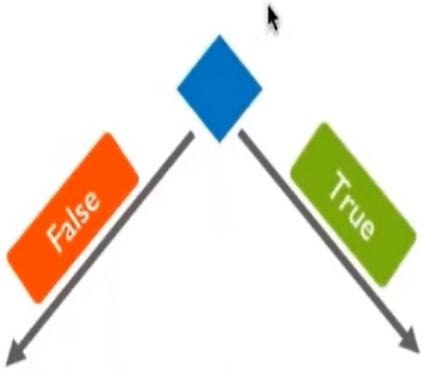
Step 2: Creating Decision Tree: Bootstrapped Dataset



Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	85	No

Step 2: Creating Decision Tree: Bootstrapped Dataset



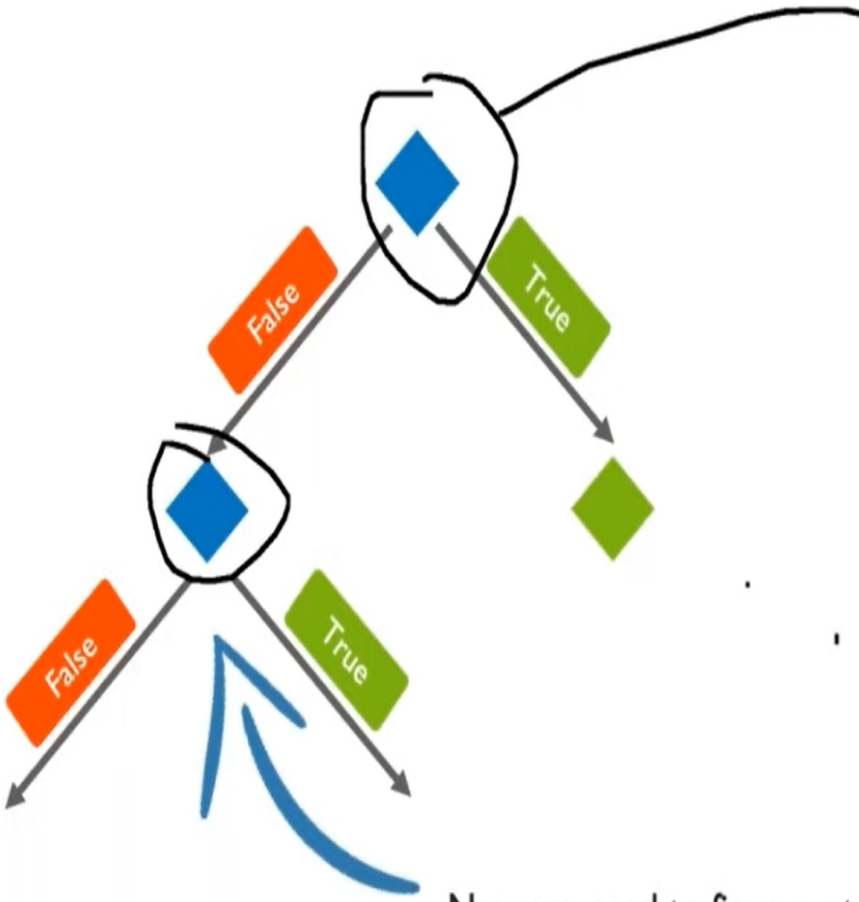
Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

Create a decision tree (eg: Root node - *Overweight*) using the bootstrapped dataset. Use only a random subset of variables/column at each step

Instead of considering all the 4 variables to figure out how to split the root node. **In this example**, consider only 2 *variables* at each step

Step 2: Creating Decision Tree: Bootstrapped Dataset



Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

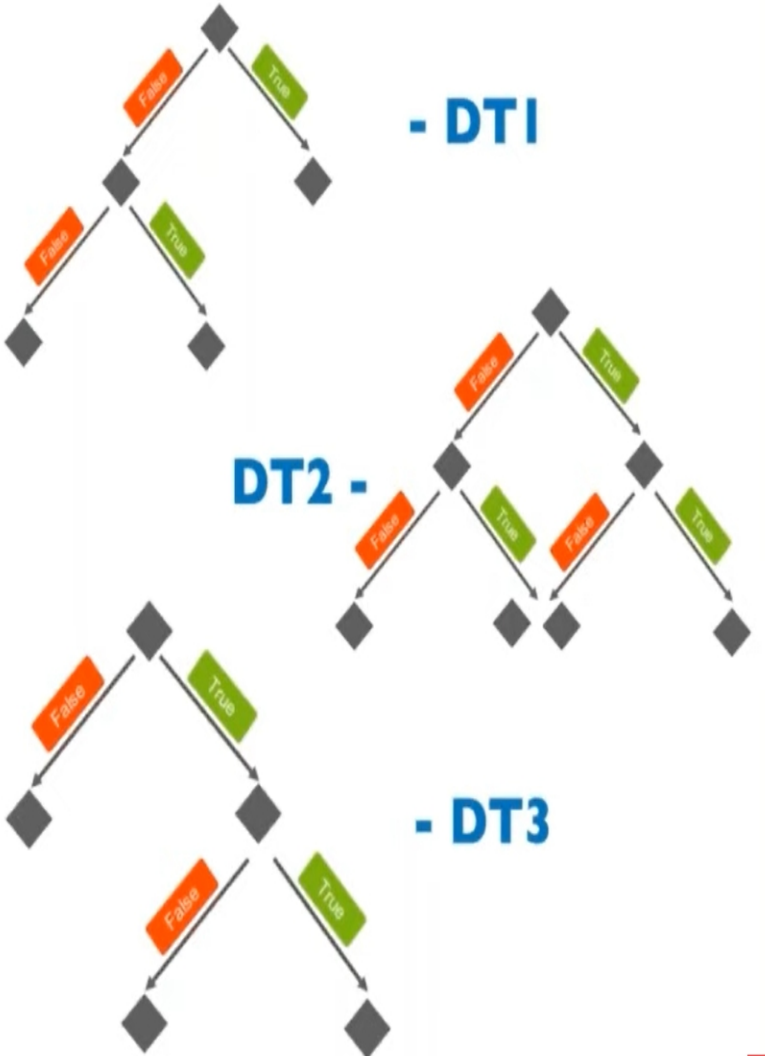
The decision tree is ready using the random subset of variables at each step

Now we need to figure out, how to split samples at this node

Step 2: Creating Decision Tree: Bootstrapped Dataset

Original Dataset

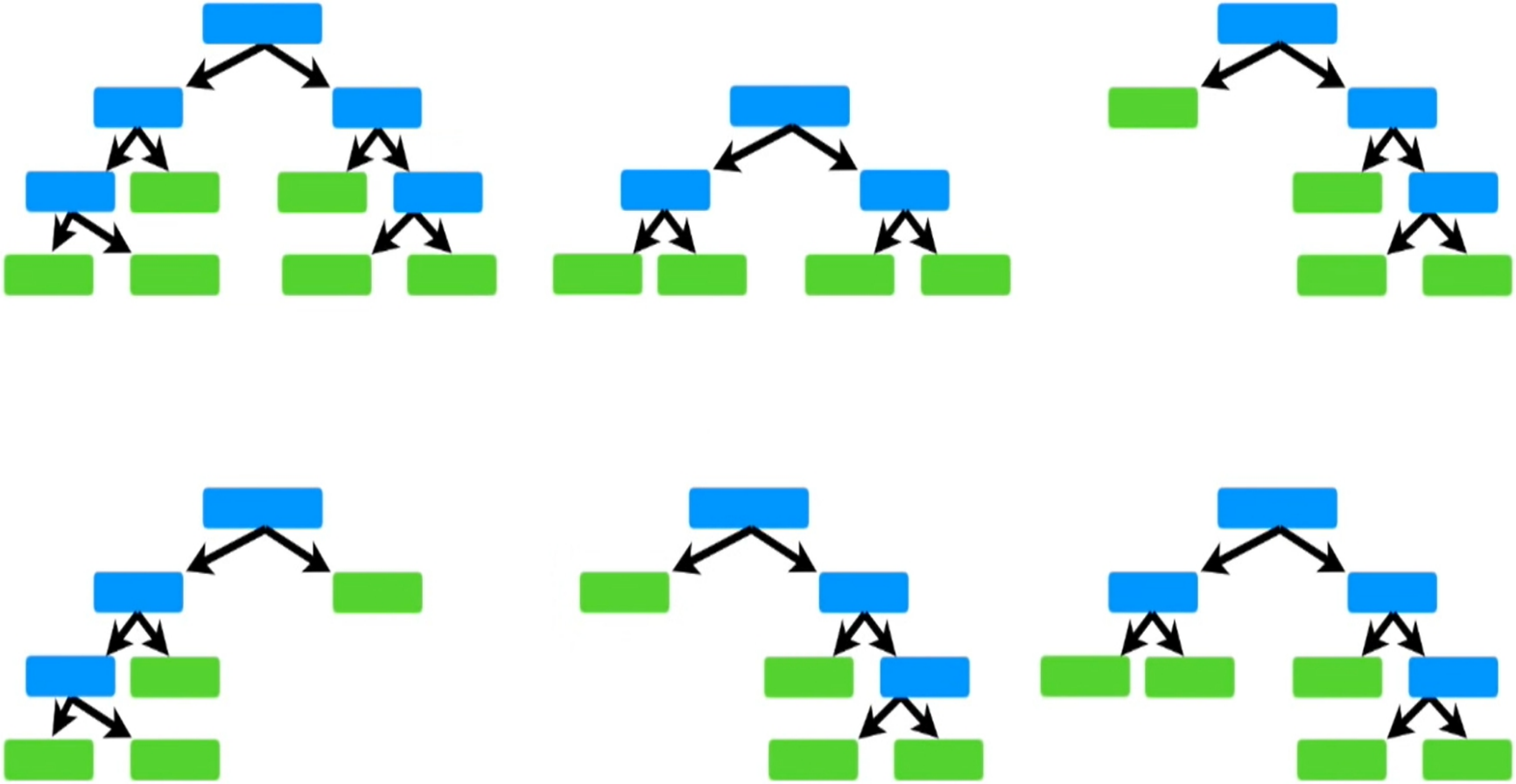
Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes



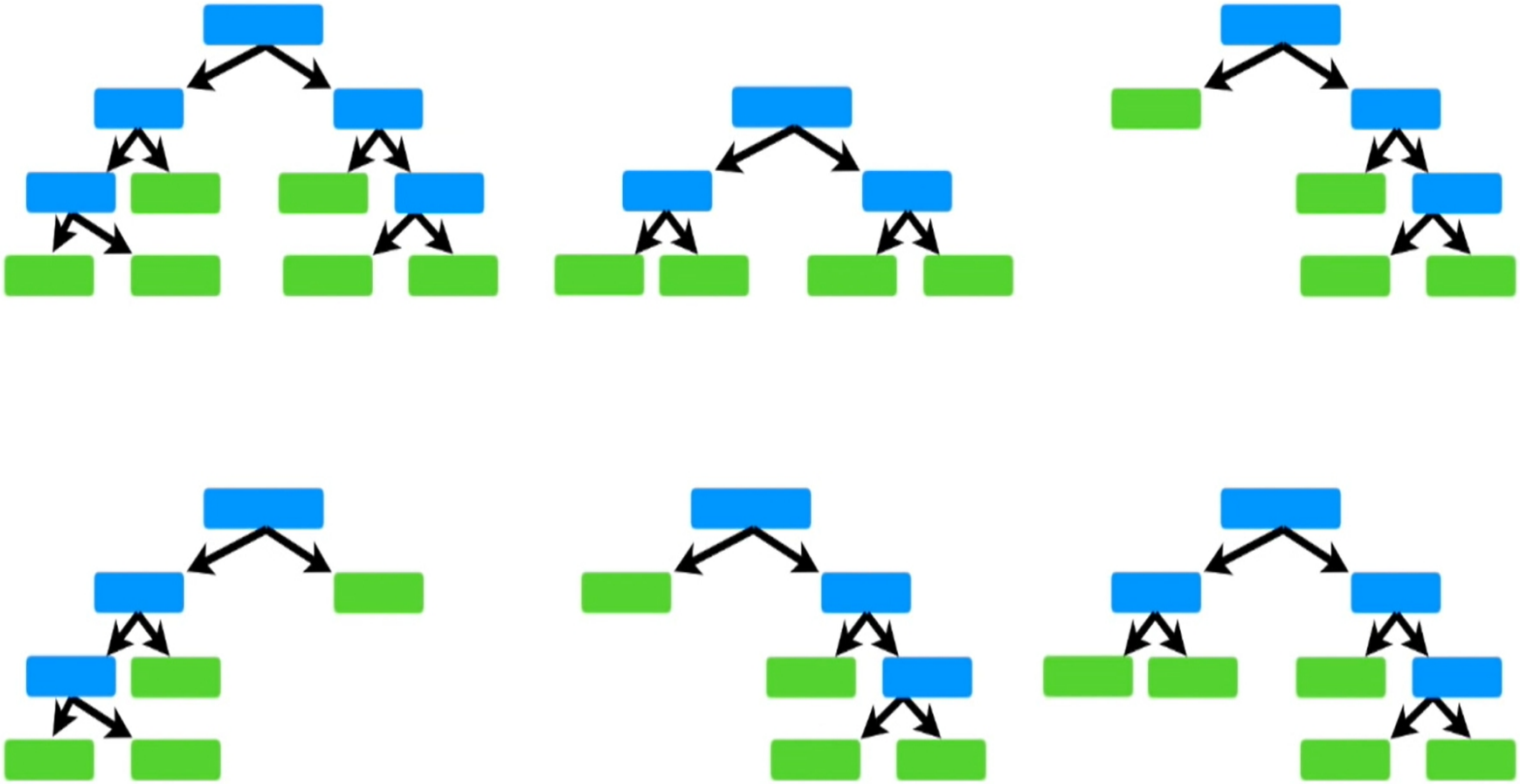
Recursively splitting sample at other nodes

Get back to Step 1 and repeat: Build new bootstrapped dataset and rebuild decision trees considering subset of variables at each step (ideally you have to repeat this step 100's of time)

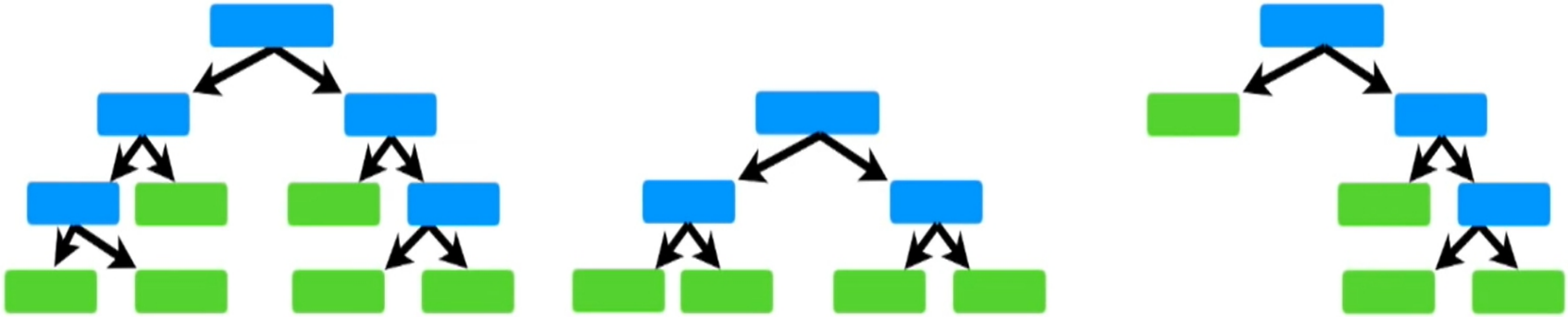
Now go back to Step 1 and repeat: Make a new bootstrapped dataset and build a tree considering a subset of variables at each step.



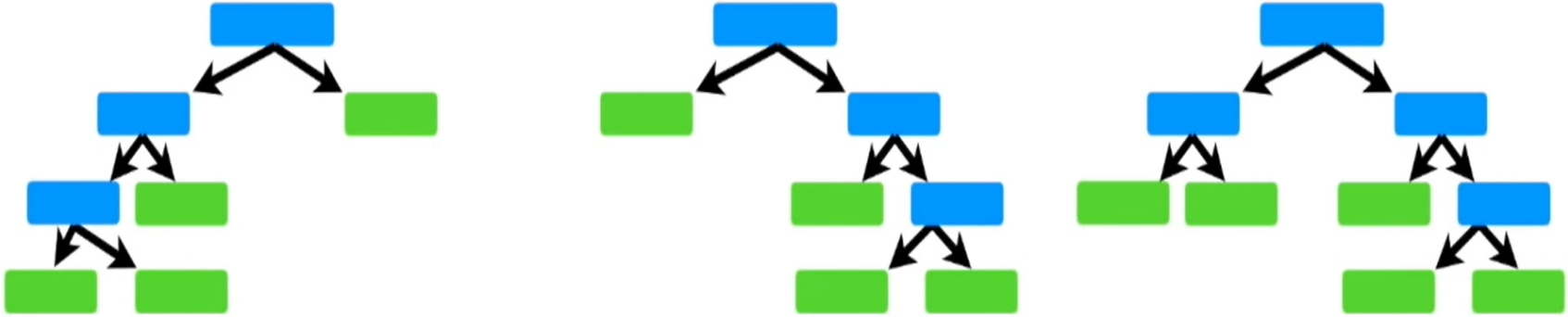
Ideally, you'd do this 100's of times, but we only have space to show 6… but you get the idea.



Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees.



The variety is what makes random forests more effective than individual decision trees.



Step 3 & 4: Counting the Votes for Predicting

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	No	75	?

Predict if the patient has diabetes or not

Vote Count

Diabetes	
YES	NO
95	5

In this case, '**YES**' received most of the votes,
→ **patient has heart disease**

Let's say we created 100 decision tree and this is the overall vote count of the predictions from all the decision tree. Next thing is to find out the option which received most votes

How Good is Your Model: Bootstrapped Dataset

Original Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

Bootstrapped Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	Yes	No	75	No

Remember! this entry (almost 1/3rd of the original dataset) was not included in the bootstrapped dataset because of the duplicate entry

How Good is Your Model: Out-of-bag Dataset

Original Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No
Yes	Yes	Yes	100	Yes
Yes	Yes	No	75	No
Yes	No	Yes	110	Yes

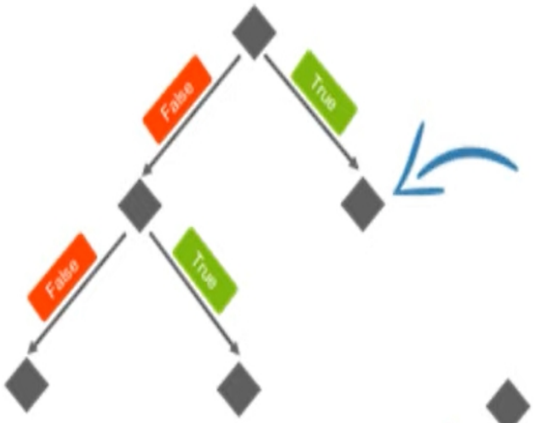
Out-of-Bag Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	No	Yes	110	Yes

This is the entry that did not end up in the bootstrapped dataset and the collection of such entries as a dataset is known as **" Out of Bag Dataset"**

NOTE: Just in case, the original dataset were larger, we would have more than 1 entry over

How Good is Your Model: Out-of-bag Dataset



Predicted
Diabetes: **YES**

Out-of-Bag Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	No	Yes	110	Yes

Run this Out-of-Bag sample data through all the trees that were built without it

Predicted
Diabetes: **NO**

. . .

Predicted
Diabetes: **NO**

How Good is Your Model: Out-of-bag Dataset

Classification of Out-of-Bag Dataset

Diabetes	
YES	NO
1	4

Diabetes	
YES	NO
4	0

Diabetes	
YES	NO
3	1

Out-of-Bag Dataset

Family History	High BP	Overweight	Weight (kg)	Diabetes
No	No	No	65	No

Next we do the same for all of the other Out-of-Bag samples for all the trees, and finally the accuracy of random forest can be measured by the proportion of Out-of-Bag samples that were correctly classified by the algorithm

Classification of the Out-Of-Bag sample	
Yes	No
1	3

Classification of the Out-Of-Bag sample	
Yes	No
4	0

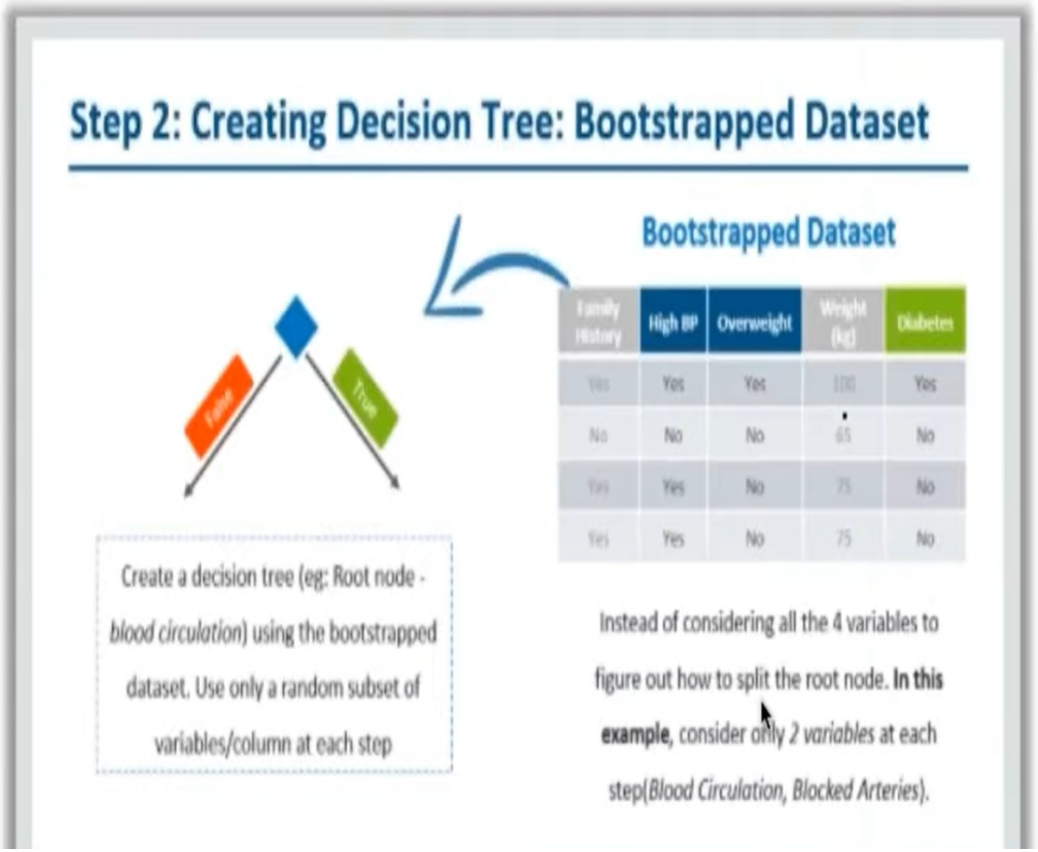
Classification of the Out-Of-Bag sample	
Yes	No
3	1

etc… etc… etc…

Ultimately, we can measure how accurate our random forest is by the proportion of Out-Of-Bag samples that were correctly classified by the Random Forest.

The proportion of Out-Of-Bag samples that were *incorrectly* classified is the "**Out-Of-Bag Error**"

Bootstrapped Dataset: 2 Variables



Remember when we built our first tree, we only used 2 variables to make decision at each step?

Bootstrapped Dataset: 2 Variables – 3 Variables

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

2 Variables

Family History	High BP	Overweight	Weight (kg)	Diabetes
Yes	Yes	Yes	100	Yes
No	No	No	65	No
Yes	Yes	No	75	No
Yes	Yes	No	75	No

3 Variables

Compare the Out-of-Bag error for a random forest built using 2 variables vs 3 variables and select the most accurate random forest

```
 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1
 0 0 0 0]
```

In [98]:
```
print(y_test)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

Feature Scaling

In [99]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [ ]:
```
print(X_train[0:10])
```

In [ ]:
```
print(X_test[0:5])
```

Training the Random Forest Classification model on the Training set

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

Predicting a new result

Random Forests: Missing data and clustering

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

We've got data for 4 patients...

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

However, for patient #4, we've got some missing data.

Random Forests consider 2 types of missing data...

1) Missing data in the original dataset used to create the random forest.

2) Missing data in a new sample that you want to categorize.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

New Sample

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	???	

Random Forests consider 2 types of missing data...

1) Missing data in the original dataset used to create the random forest.

We'll start with this one...

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	???	???	No

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

The general idea for dealing with missing data in this context is to make an initial guess that could be bad, then gradually refine the guess until it is (hopefully) a good guess.

No is our initial guess as it is 2:3

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

"**No**" is the most common value for Blocked arteries - it occurs in 2 out of 3 samples.

Our guess for weight is median value: 180

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	???	No

Since weight is numeric, our initial guess will be the median value.

Correction: In median

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	???	No

Since **weight** is numeric, our initial guess will be the median value of the patients that ***did not*** have **heart disease**.

Correction: Instead of 180 it will be 167.5 in all remaing slides

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

In this case, the median value is **167.5**.

Our new data set after filling the missing values:

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we want to refine these guesses.

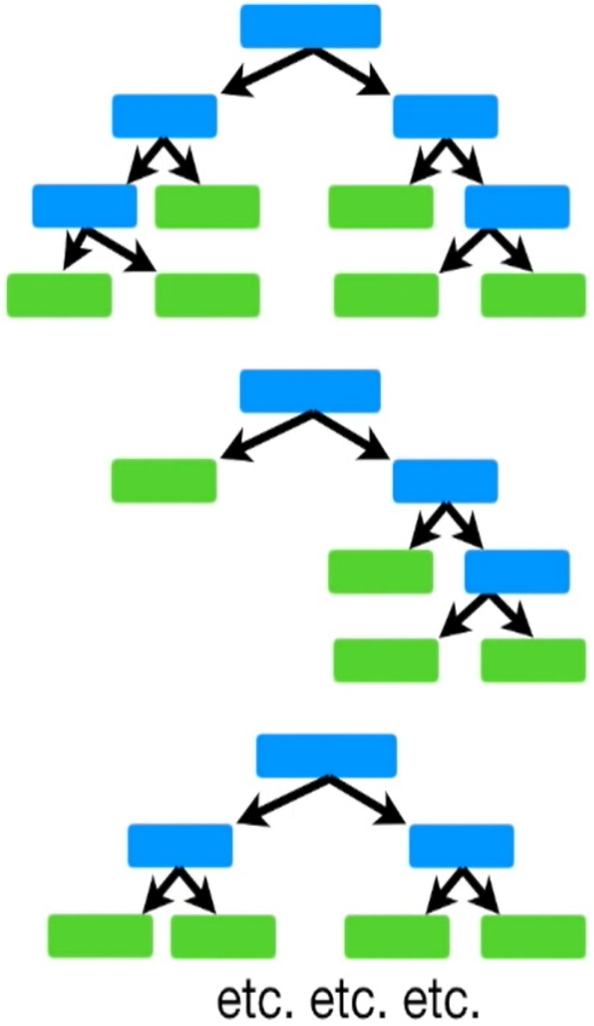
We do this by first determining which samples are similar to the one with missing data.

So let's talk about how to determine similarity...

Filled-in Missing Values

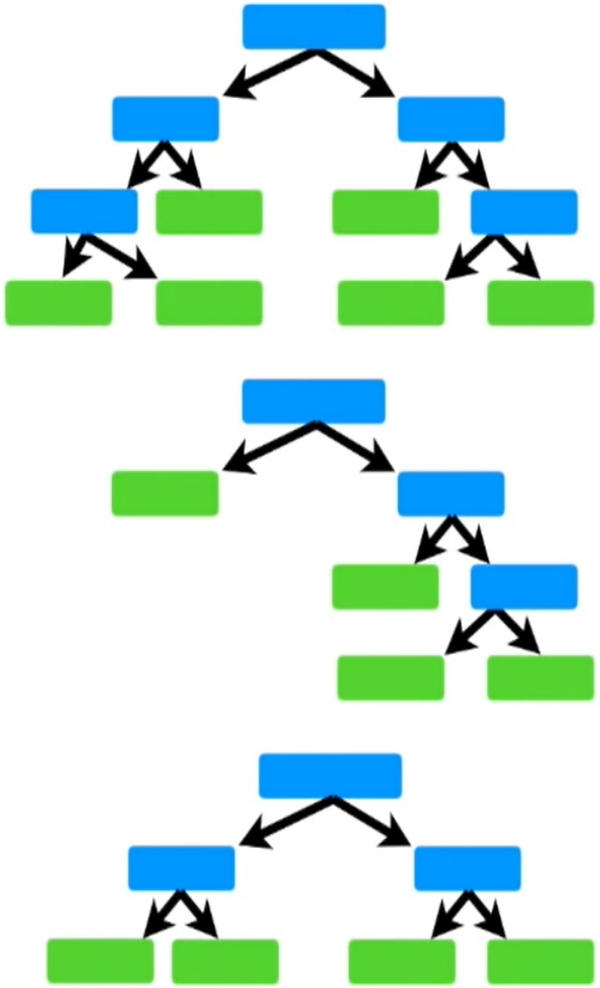
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Step 1: Build a random forest…



etc. etc. etc.

Step 2: Run all of the data down all of the trees.



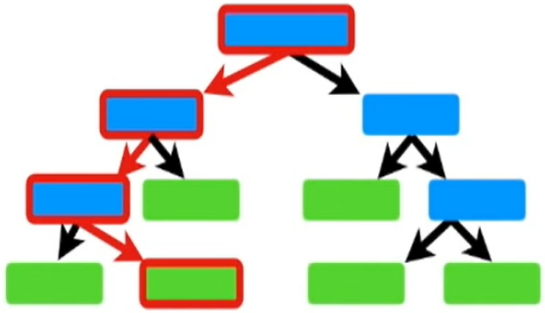
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Sample 3 and 4 are similar as they ended up at same leaf node.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



Notice that Sample 3 and Sample 4 both ended up at the same leaf node.

The proximity matrix has a row and has a column from each Sample

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

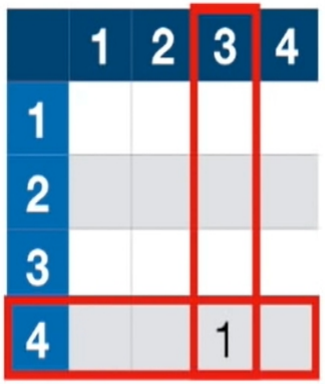
We keep track of similar samples using a "Proximity Matrix"

	1	2	3	4
1				
2				
3				
4				

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Because sample 3…

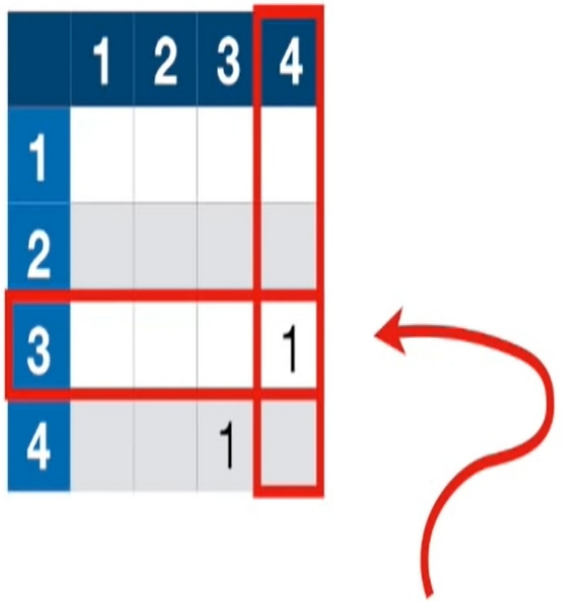


…and sample 4 ended up in the same leaf node…

…we put a 1 here.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No



We also put a 1 here, since this position also represents samples 3 and 4.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

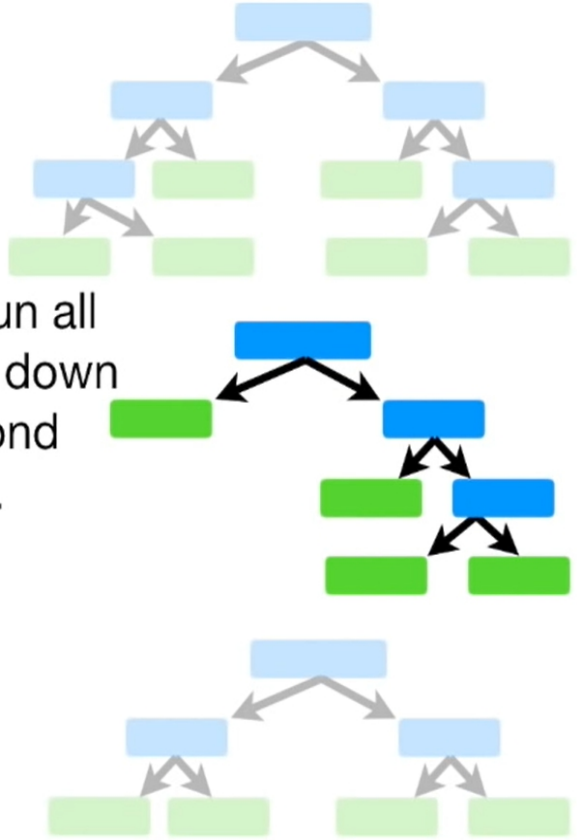
Because no other pair of samples ended in the same leaf node, our proximity matrix looks like this after running the samples down the first tree.

	1	2	3	4
1				
2				
3				1
4			1	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

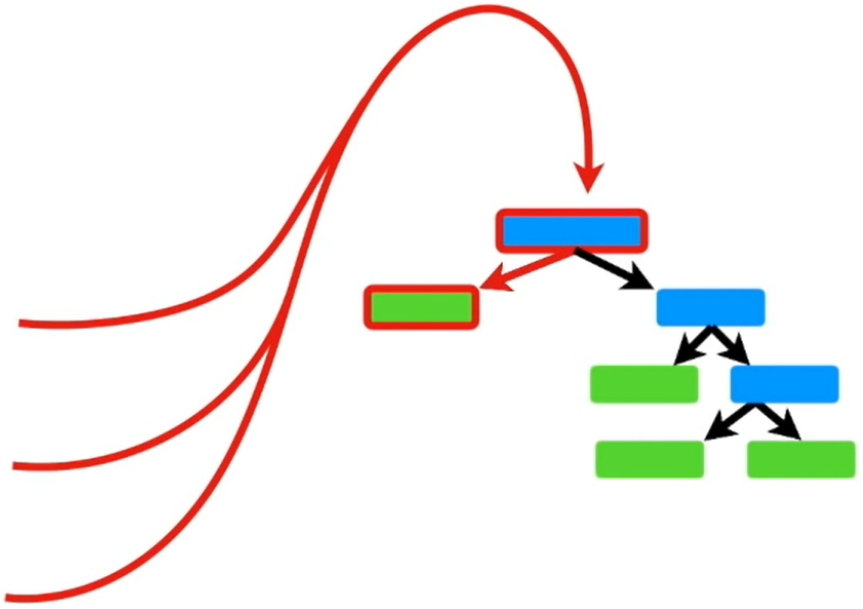
Now we run all of the data down the second tree…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

NOTE: Samples 2, 3 and 4 all ended up in the same leaf node.



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

…after the second tree, we add 1 to any pair of samples that ended up in the same leaf node.

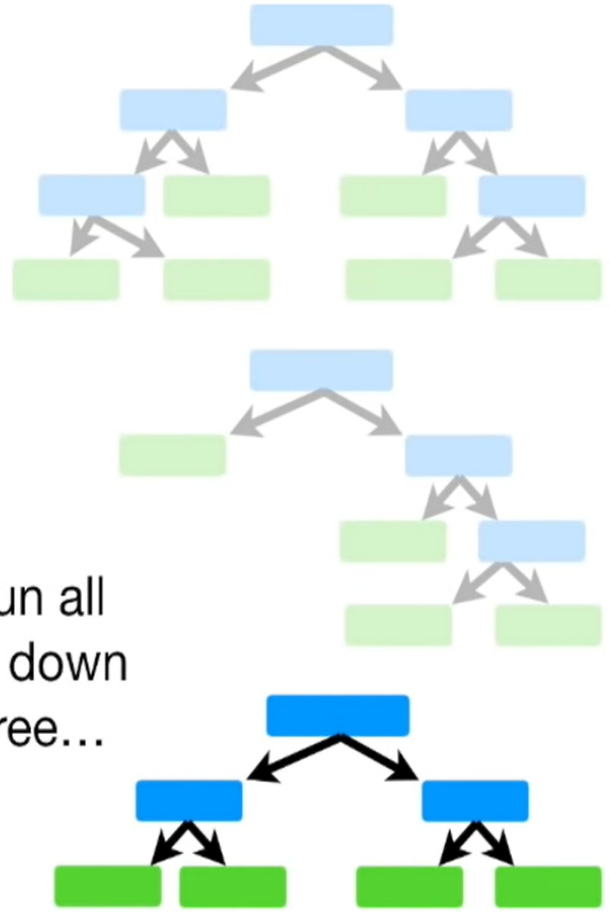
	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

Samples 3 and 4 ended up in the same node together again…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**180**	No

Now we run all of the data down the third tree…



Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**167.5**	No

Ultimately, we run the data down all the trees and the proximity matrix fills in.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**No**	**167.5**	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Then we divide each proximity value by the total number of trees. In this example, assume we had 10 trees.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Now we use the proximity values for sample 4 to make better guesses about the missing data.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

For Blocked Arteries, we calculate the weighted frequency of "Yes" and "No, using proximity values as the weights.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Yes = 1/3

No = 2/3

The frequency of "No".

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" =

$$\frac{\text{Proximity of "Yes"}}{\text{All Proximities}}$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

Yes = $\frac{1}{3}$ x The weight for **"Yes"**

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" = 0.1

The proximity value for Sample 2 (the only one with "Yes")

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

The weighted frequency for **"Yes"** is...

Yes = $\frac{1}{3}$ x The weight for **"Yes"**

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "Yes" = $\frac{0.1}{0.1 + 0.1 + 0.8}$

Divided by the sum of the proximities for Sample 4.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"Yes"** is…

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

$$\text{Yes} = \frac{1}{3} \times 0.1$$

$$\text{The weight for "Yes"} = \frac{0.1}{0.1 + 0.1 + 0.8} = \frac{0.1}{1} = 0.1$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

The weighted frequency for "**Yes**".

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"No"** is…

Yes = $\frac{1}{3}$ x 0.1 = 0.03

No = $\frac{2}{3}$ x The weight for **"No"**

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Samples 1 and 3 both have "No"…

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**???**	???	No

The weighted frequency for **"No"** is…

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 = 0.6$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weight for "No" $= \frac{0.1 + 0.8}{0.1 + 0.1 + 0.8} = \frac{0.9}{1} = 0.9$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	???	No

The weighted frequency for **"No"** is…

Yes = 1/3

No = 2/3

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 = 0.6$$

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

"No" has a way higher weighted frequency, so we'll go with it.

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

For weight, we use the proximities to calculate a weighted average.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

Weighted average = 125 x Sample 1's weighted average weight…

$$\frac{0.1}{0.1 + 0.1 + 0.8}$$

Divided by the sum of the proximities.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = 125 x 0.1

$$\frac{0.1}{0.1 + 0.1 + 0.8} = \frac{0.1}{1} = 0.1$$

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1)

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

…the weighted value for 180…

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

…the weighted value for 210…

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

= 198.5

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	**198.5**	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

The weighted average weight!

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	**NO**	**198.5**	No

Now that we've revised our guesses a little bit, we do the whole thing over again…

We build a random forest, run the data through the trees, recalculate the proximities and recalculate the missing values.

We do this 6 or 7 times until the missing values converge (i.e. no longer change each time we recalculate).

Let me show you something super cool
we can do with the proximity matrix!!!

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

This is the proximity matrix before we divided each value by 10, the number of trees in the pretend random forest.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Just for the sake of easy math, imagine if Samples 3 and 4 ended up in the same leaf node in all 10 trees.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

Now we have 10 here and here…

After dividing by 10 (the number of trees in the forest), we see that the largest number in the proximity matrix is 1.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

1 in the proximity matrix means the samples are as close as close can be.

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

That means…
1 - the proximity values
= distance

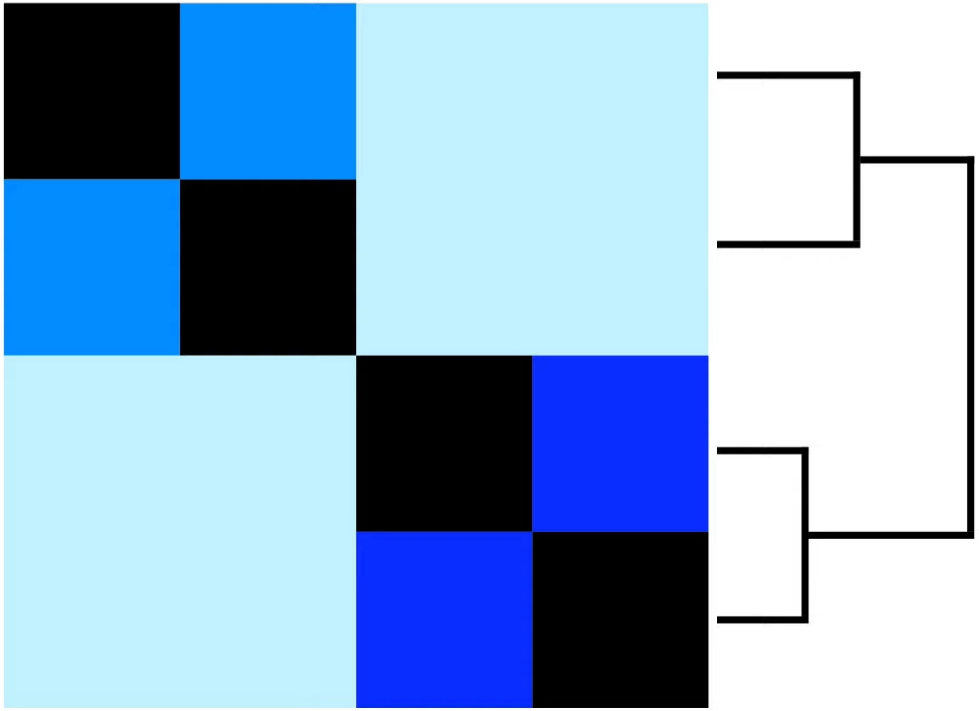
	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Close as can be = no distance between.

Sample 1 Sample 2 Sample 3 Sample 4



This is a distance matrix…

…and that means we can draw a heatmap with it!!!

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Thank You