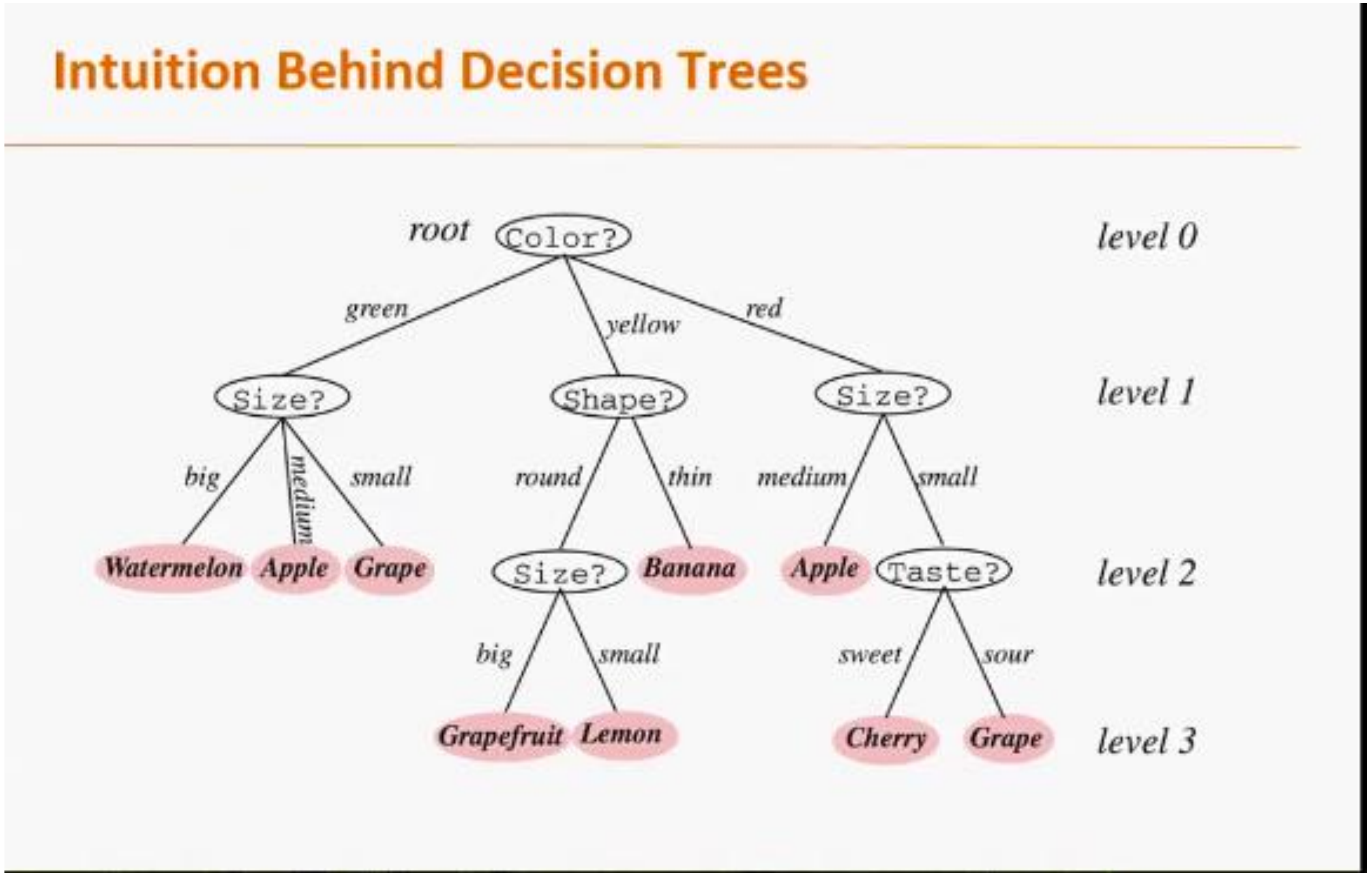
Decision Trees & Random Forests

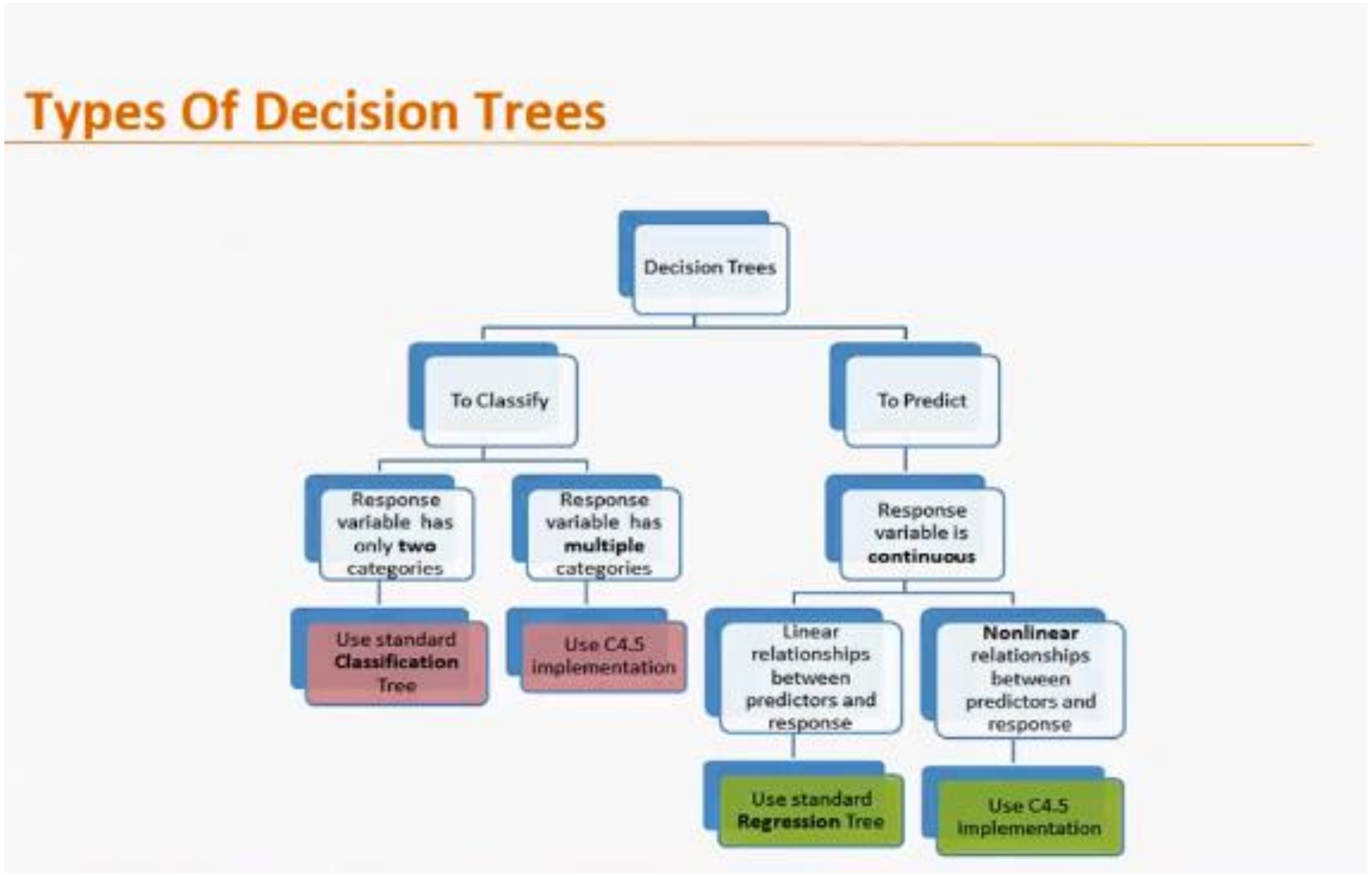
Decision Trees



Decision Trees

CART

Classification and Regression Trees


Types Of Decision Trees
Decision Trees
To Classify
To Predict
Response variable has only two categories
Response variable has multiple categories
Response variable is continuous
Use standard Classification Tree
Use C4.5 implementation
Linear relationships between predictors and response
Nonlinear relationships between predictors and response
Use standard Regression Tree
Use C4.5 implementation

Decision Trees

Types of Algorithms:

1) ID3 - Iterative Dichotomiser 3 using Entropy and Information Gain
2) Gini – Using Gini index
3) Chi – Square
4) Reduction in Variance

Decision Trees

Categorical Classification:

- Using the number of Yes or No

Numerical Variable

- Using the average of the adjacent data

Decision Trees

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

Decision Trees

If the subset says all are pure (All are yes) then we don’t need to split that further. We reached the leaf node. In the above example it is Overcast. If any of the nodes have combination of yes and no then we need to subset it and go further.

ID3 algorithm uses the Entropy and Information Gain to decide which node should be the current node or root node.

ID3 Algorithm will perform following tasks recursively

```
    Create root node for the tree
    If all examples are positive, return leaf node ‘positive’
    Else if all examples are negative, return leaf node ‘negative’
    Calculate the entropy of current state $H(S)$
    For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by $H(S, x)$
    Select the attribute which has maximum value of $IG(S, x)$
    Remove the attribute that offers highest IG from the set of attributes
    Repeat until we run out of all attributes, or the decision tree has all leaf nodes.
```

Decision Trees

Formula for Entropy calculation is:

$$\sum_{i=1}^{n} P(x).\,log2(x)$$

Formula for Information Gain is:

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Decision Trees

Yes	No	Total
9	5	14

$$Entropy(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$= 0.940$$

Decision Trees

Let’s start with ‘Wind’

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Wind = Weak	Wind = Strong	Total
8	6	14

Probability of weak = 8/14

Probability of strong = 6/14

Decision Trees

Let’s Now out of the 8 Weak examples, 6 of them were ‘Yes’ for Play Golf and 2 of them were ‘No’ for ‘Play Golf’.
So, we have, with ‘Wind’

$$Entropy(S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)$$
$$= 0.811$$

$$Entropy(S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$
$$= 1.000$$

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) * H(S_{weak}) - P(S_{strong}) * H(S_{strong})$$

$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$

$$= 0.048$$

Decision Trees

Calculating for other variables similarly we’ve got below result.

$$IG(S, Outlook) = 0.246$$

$$IG(S, Temperature) = 0.029$$

$$IG(S, Humidity) = 0.151$$

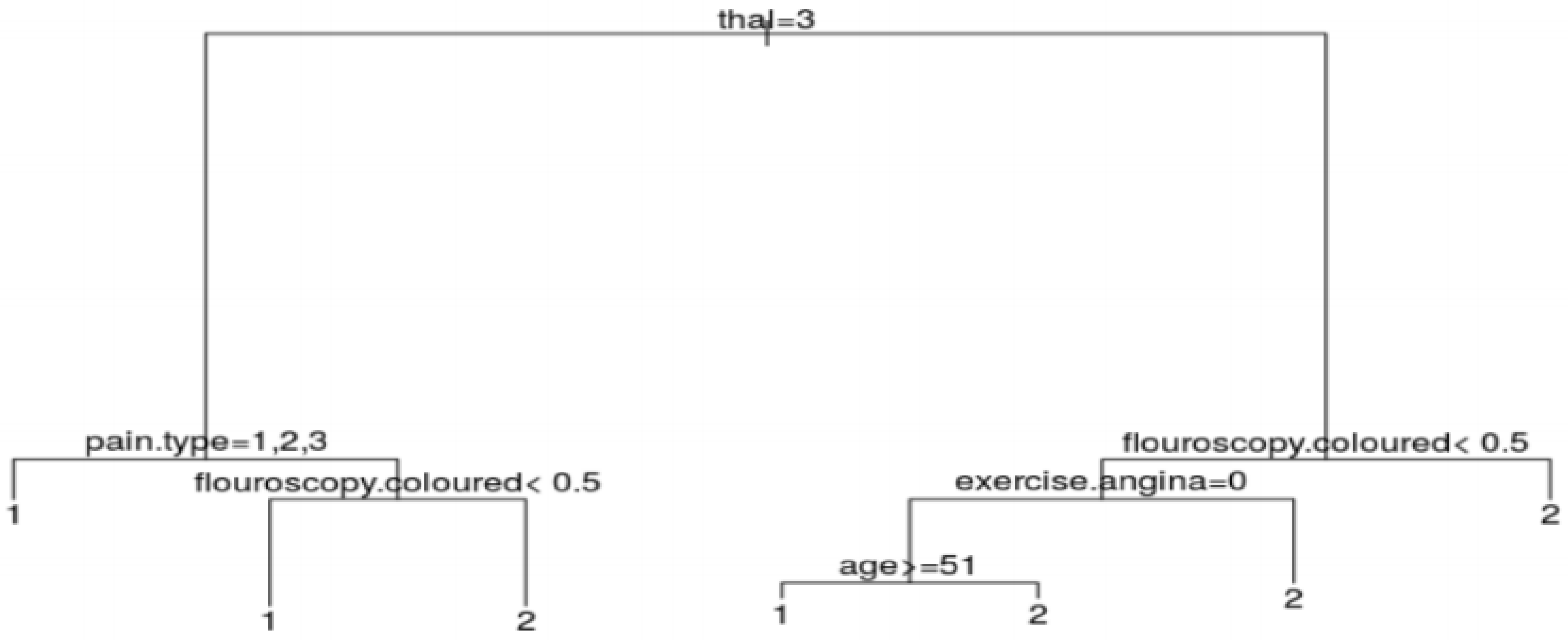
As we have IG value more for Outlook we will select Outlook as the root value

Decision Trees

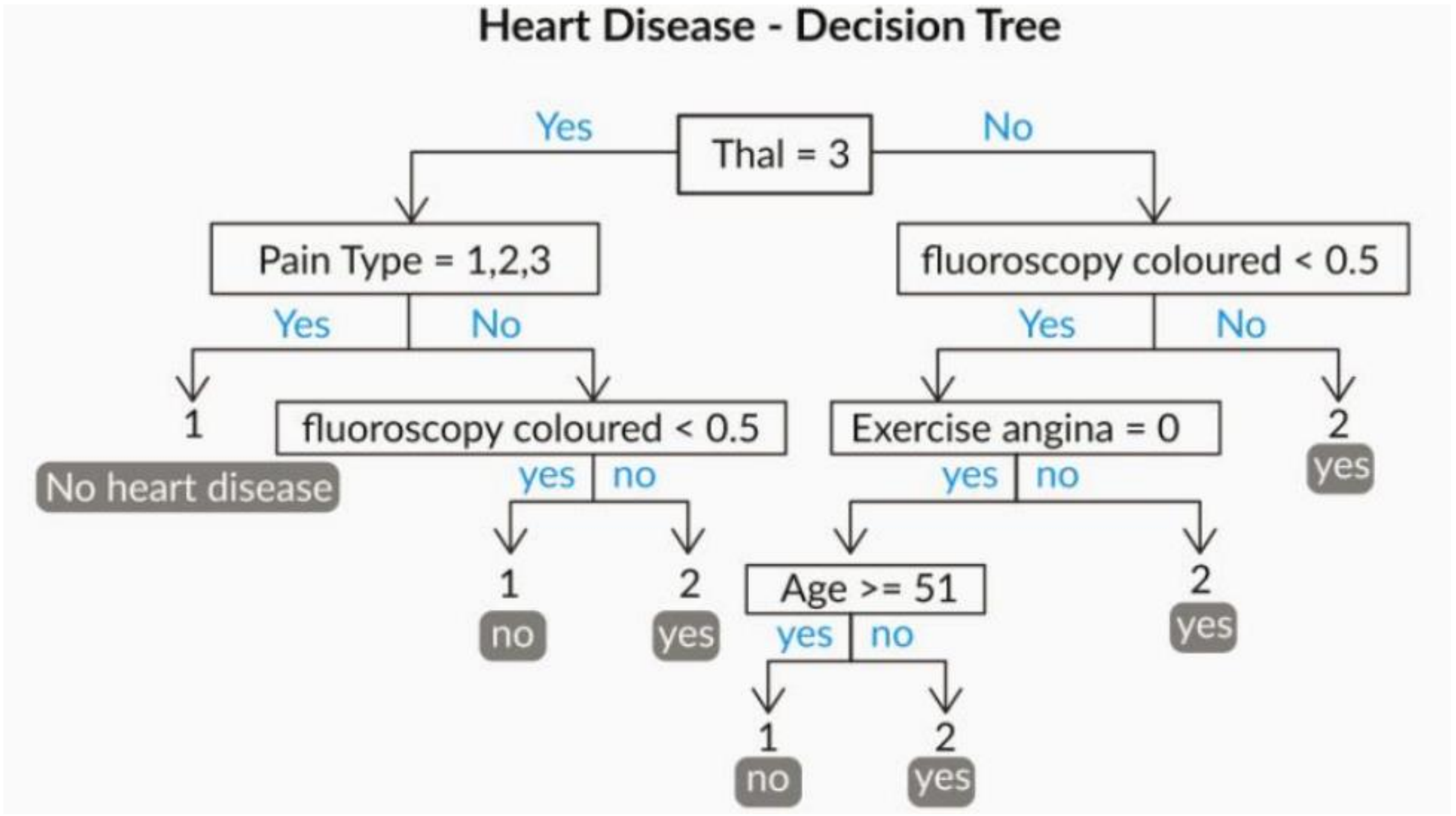
Decision Trees naturally represent the way we make decisions. Think of a machine learning model as a decision making engine that takes a decision on any given input object (data point). Imagine a doctor making a decision (the diagnosis) on whether a patient is suffering from a particular condition given the patient data, an insurance company making a decision on whether claims on a particular insurance policy needs to be paid out or not given the policy and the claim data, a company deciding on which role an applicant seeking a position in the company is eligible to apply for, based on the past track record and other details of the applicant, etc.. Solutions to each of these can be thought of as machine learning models trying to mimic the human decision making.

Decision Trees

Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed." The Heart dataset Figure 2) consists of data about various cardiac parameters along with an indicator column that says whether the person has a heart disease or not.



Decision Trees



Decision Trees

So the decision trees can go back and tell you the factors leading to a given decision. a decision tree gives you the exact reason, i.e. either 'Thal is 3, the pain type is neither 1, nor 2, nor 3, and the coloured fluoroscopy is greater than or equal to 0.5', or 'Thal is not equal to 3, and either of the three tests, shown in the right half of the tree, failed'.

Consider the heart disease decision tree again. Given that a patient is diagnosed with heart disease, you can easily trace your way back to the multiple tests that would have led to this diagnosis. One such case could be where the patient doesn't have thal = 3, and coloured fluoroscopy is greater than or equal to 0.5.

In other words, each decision is reached via a path that can be expressed as a series of 'if' conditions satisfied together, i.e., if 'thal' is not equal to 3, and if coloured fluoroscopy is greater than or equal to 0.5, then the patient has heart disease. Final decisions in the form of class labels are stored in leaves.

Decision Trees

Gini Index

Gini Index uses the probability of finding a data point with one label as an indicator for homogeneity — if the dataset is completely homogeneous, then the probability of finding a datapoint with one of the labels is 1 and the probability of finding a data point with the other label is zero.

An empirical estimate of the probability pi of finding a data point with label i (assuming the target attribute can take say k distinct values) is just the ratio of the number of data points with label i to the total number of data points. It must be that ∫ pi = 1 k i=1 . For binary classification problems the probabilities for the two classes become p and (1 − p). Gini Index is then defined as:

$$\sum_{i=1}^{n} P \ ({p_i}^2 + (1 - p_i)^2)$$

Decision Trees

Note that the Gini index is maximum when Pi = 1 for exactly one of the classes and all others are zero. So higher the Gini index higher the homogeneity. In a Gini based decision tree algorithm, we therefore find the split that maximizes the weighted sum (weighted by the size of the partition) of the Gini indices of the two partitions created by the split. For the example in Figure 6:

Split on gender: the two partitions will have 10/500 and 300/500 as the probabilities of finding a football player respectively. Each partition is half the total population:

Decision Trees

$$Gini = \frac{1}{2}\left(\left(\frac{1}{50}\right)^2 + \left(\frac{49}{50}\right)^2\right) + \frac{1}{2}\left(\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2\right) = 0.7404$$

Split on Age: the two partitions will have 260/700 and 50/250 as the probabilities, and 700 and 300 as the sizes respectively, giving us a Gini index of:

$$Gini = 0.7\left(\left(\frac{26}{70}\right)^2 + \left(\frac{44}{70}\right)^2\right) + 0.3\left(\left(\frac{1}{5}\right)^2 + \left(\frac{4}{5}\right)^2\right) = 0.5771$$

Therefore we would first need to split on the gender — this split gives a higher GINI index for the partitions. Gini index can only be used on classification problems where the target attribute is categorical.

Decision Trees

Advantages

- Predictions made by a decision tree are easily interpretable.
- A decision tree does not assume anything specific about the nature of the attributes in a data set. It can seamlessly handle all kinds of data — numeric, categorical, strings, Boolean, and so on.
- Decision trees often give us an idea of the relative importance of the explanatory attributes that are used for prediction.

Disadvantages

- Decision trees tend to overfit the data. If allowed to grow with no check on its complexity, a tree will keep splitting till it has correctly classified (or rather, mugged up) all the data points in the training set.
- Decision trees tend to be very unstable, which is an implication of overfitting. A few changes in the data can change a tree considerably.

Random Forest

You are familiar with decision trees, now it’s time to learn about Random Forests, which is a collection of decision trees. The great thing about random forests is that - they almost always outperform a decision tree in terms of accuracy

Ensembles

An ensemble means a group of things viewed as a whole rather than individually. In ensembles, a collection of models is used to make predictions, rather than individual models. Arguably, the most popular in the family of ensemble models is the random forest: an ensemble made by the combination of a large number of decision trees. Random forests use a technique known as **bagging**, which is an **ensemble** method

Random Forest

For an ensemble to work, each model of the ensemble should comply with the following conditions:

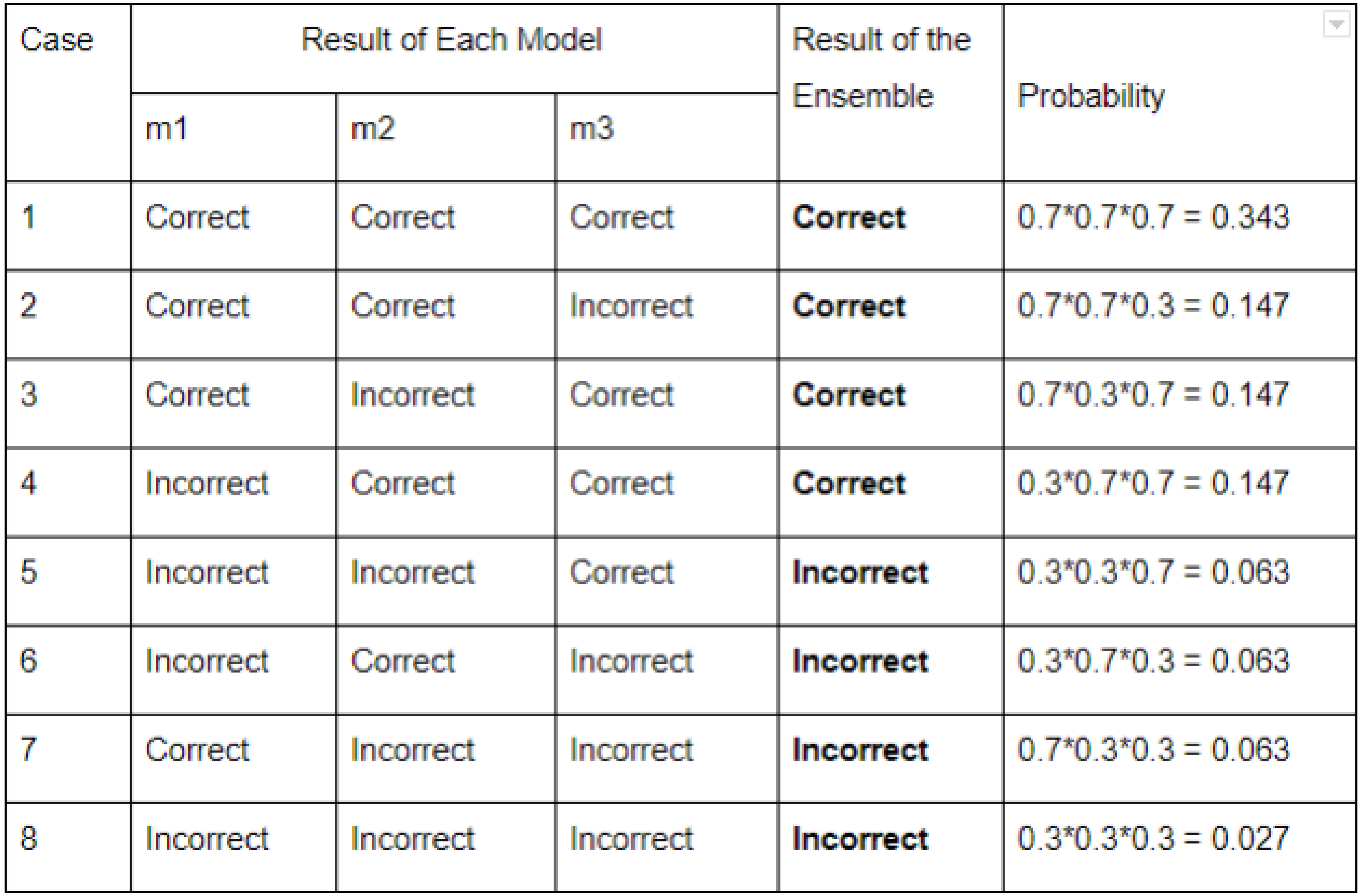
1. Each model should be diverse. Diversity ensures that the models serve complementary purposes, which means that the individual models make predictions independent of each other.

2. Each model should be acceptable. Acceptability implies that each model is at least better than a random model.

Consider a binary classification problem where the response variable is either 0 or 1. You have an ensemble of three models, where each model has an accuracy of 0.7 i.e. it is correct 70% of the times. The following table shows all the possible cases that can occur while classifying a test data point as 1 or 0. The column to the extreme right shows the probability of each case.

If we can prove that the probability of more than half the models making a wrong prediction is lesser than that of any of the individual models, we know that the ensemble is a better choice than any of the individual models.

Random Forest



Case	Result of Each Model			Result of the Ensemble	Probability
	m1	m2	m3		
1	Correct	Correct	Correct	**Correct**	0.7*0.7*0.7 = 0.343
2	Correct	Correct	Incorrect	**Correct**	0.7*0.7*0.3 = 0.147
3	Correct	Incorrect	Correct	**Correct**	0.7*0.3*0.7 = 0.147
4	Incorrect	Correct	Correct	**Correct**	0.3*0.7*0.7 = 0.147
5	Incorrect	Incorrect	Correct	**Incorrect**	0.3*0.3*0.7 = 0.063
6	Incorrect	Correct	Incorrect	**Incorrect**	0.3*0.7*0.3 = 0.063
7	Correct	Incorrect	Incorrect	**Incorrect**	0.7*0.3*0.3 = 0.063
8	Incorrect	Incorrect	Incorrect	**Incorrect**	0.3*0.3*0.3 = 0.027

Random Forest

In the table, there are 4 cases each where the decision of the final model (ensemble) is either correct or wrong. Let’s assume that the probability of the ensemble being correct is p, and the probability of the ensemble being wrong is q.

For the data in the table, p and q can be calculated as follows: p = 0.343 + 0.147 + 0.147 + 0.147 = 0.784 q = 0.027 + 0.063 + 0.063 + 0.063 = 0.216 = 1 – p

You can see how an ensemble of just three model gives a boost to the accuracy from 70% to 78.4%. In general, the more the number of models, the higher the accuracy of an ensemble is

Random Forest

Creating a Random Forest

Random forests are created using a special ensemble method called bagging. Bagging stands for Bootstrap Aggregation. Bootstrapping means creating bootstrap samples from a given data set.

In the bagging type of ensembles, random forests are by far the most successful. They are essentially ensembles of a **number of decision trees**. You create a large number of models (say, 100 decision trees), each one on a **different bootstrap sample**, from the training set. To get the result, you **aggregate** the decisions taken by all the trees in the ensemble

Random Forest

Creating a Random Forest

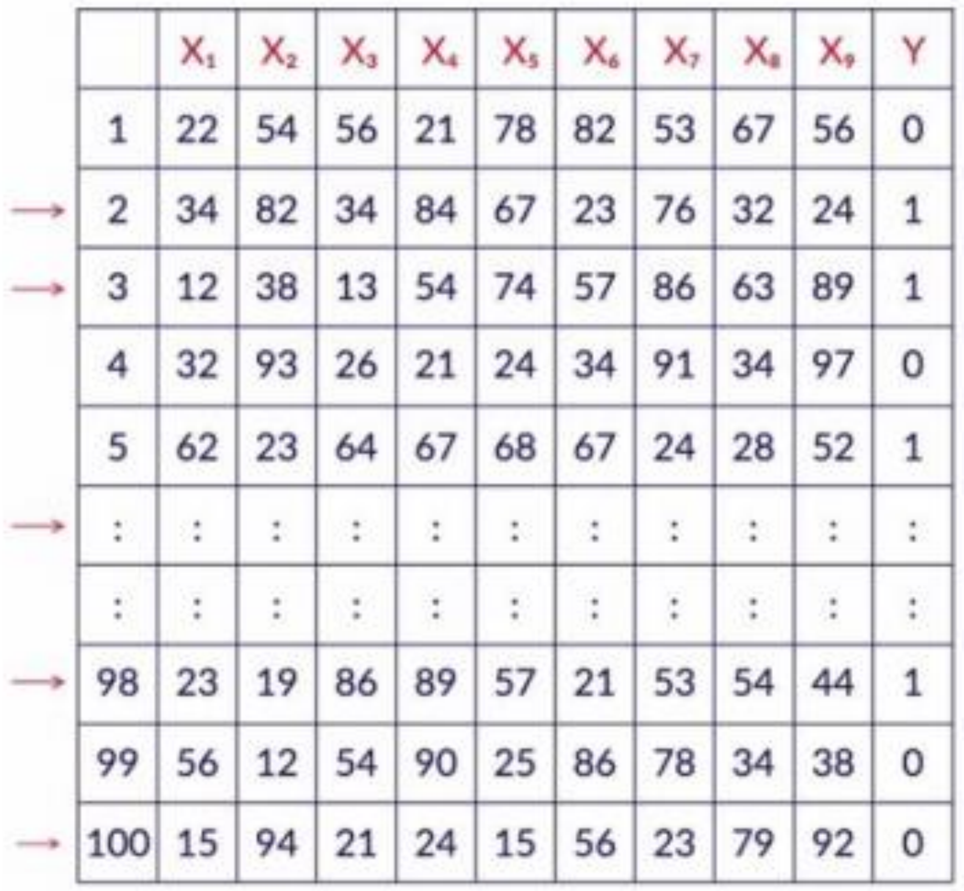
You learnt that a random forest selects a random sample of data points (bootstrap sample) to build each tree, and a random sample of features while splitting a node. Randomly selecting features ensures that each tree is **diverse**.

Suppose you want to build a random forest of 10 decision trees. First, you will create 10 bootstrap samples from the data and then, **train** each tree on a **different bootstrap sample**. Finally, while predicting a test case, each tree will make a prediction, and the final prediction will be the **majority score** of all these predictions.

Random Forest

Random forests is an ensemble of many decision trees. A random forest is created in the following way:

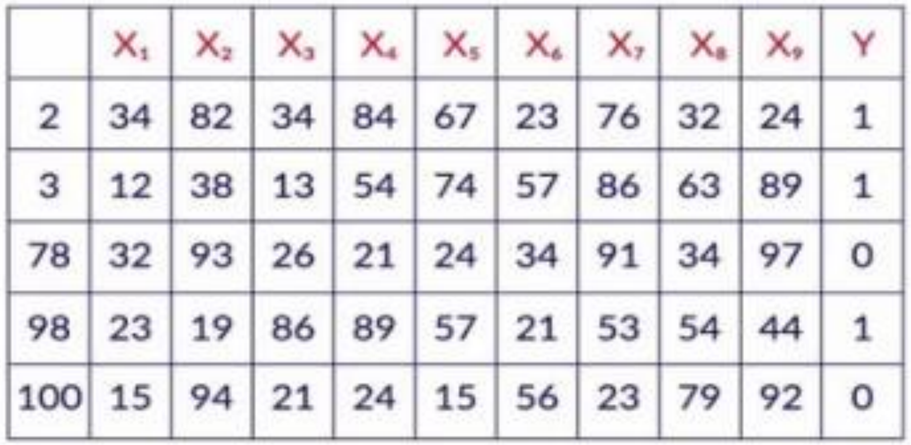
1. Create a **bootstrap sample** from the training set.



		X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
	1	22	54	56	21	78	82	53	67	56	0
→	2	34	82	34	84	67	23	76	32	24	1
→	3	12	38	13	54	74	57	86	63	89	1
	4	32	93	26	21	24	34	91	34	97	0
	5	62	23	64	67	68	67	24	28	52	1
→	:	:	:	:	:	:	:	:	:	:	:
	:	:	:	:	:	:	:	:	:	:	:
→	98	23	19	86	89	57	21	53	54	44	1
	99	56	12	54	90	25	86	78	34	38	0
→	100	15	94	21	24	15	56	23	79	92	0

Figure 2 - Training set

Random Forest



	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
2	34	82	34	84	67	23	76	32	24	1
3	12	38	13	54	74	57	86	63	89	1
78	32	93	26	21	24	34	91	34	97	0
98	23	19	86	89	57	21	53	54	44	1
100	15	94	21	24	15	56	23	79	92	0

2. Now construct a decision tree using the bootstrap sample. While splitting a node of the tree, only consider a random subset of features. Every time a node has to split, a different random subset of features will be considered.

3. Repeat the steps 1 and 2 n times, to construct n trees in the forest. Remember each tree is constructed independently, so it is possible to construct each tree in parallel.

4. While predicting a test case, each tree predicts individually, and the final prediction is given by the majority vote of all the trees.

Random Forest

There are several advantages of a random forest:

1. A random forest is more stable than any single decision tree because the results get averaged out; it is not affected by the instability and bias of an individual tree.

2. A random forest is immune to the curse of dimensionality since only a subset of features is used to split a node.
3. You can parallelize the training of a forest since each tree is constructed independently

Random Forest

Time taken to build a forest

To construct a forest of S trees, on a dataset which has M features and N observations, the time taken will depends on the following factors:

1. The number of trees. The time is directly proportional to the number of trees. But this time can be reduced by creating the trees in parallel.
2. The size of bootstrap sample. Generally the size of a bootstrap sample is 30-70% of N. The smaller the size the faster it takes to create a forest.
3. The size of subset of features while splitting a node. Generally this is taken as $\surd M$ in classification and M/3 in regression.