Random Forest is one of the classification algorithm but can be used for both regression and classification.

Classification predictive modeling problems are different from regression predictive modeling problems.

* Classification is the task of predicting a discrete class label.
* Regression is the task of predicting a continuous quantity.

There is some overlap between the algorithms for classification and regression; for example:

* A classification algorithm may predict a continuous value, but the continuous value is in the form of a probability for a class label.
* A regression algorithm may predict a discrete value, but the discrete value in the form of an integer quantity.

**Random Forest is a ensemble classifier.**

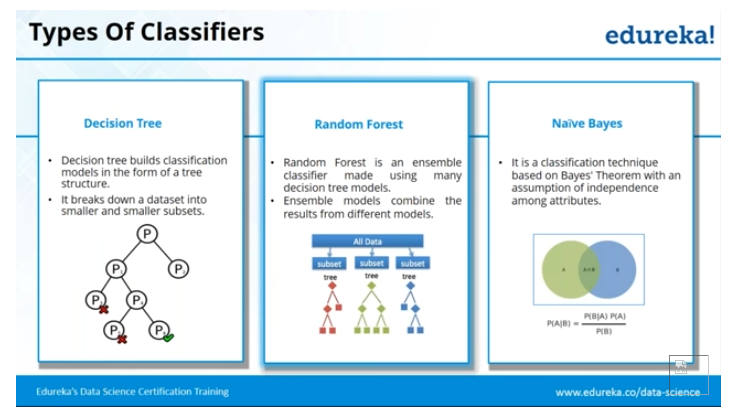
**Ensemble method** is a machine learning technique that combines several base models in order to produce one optimal predictive model.

The main principle behind the ensemble model is that a group of weak learners come together to form a strong learner, thus increasing the accuracy of the model.

**Random forest uses multiple forest trees for classification.**

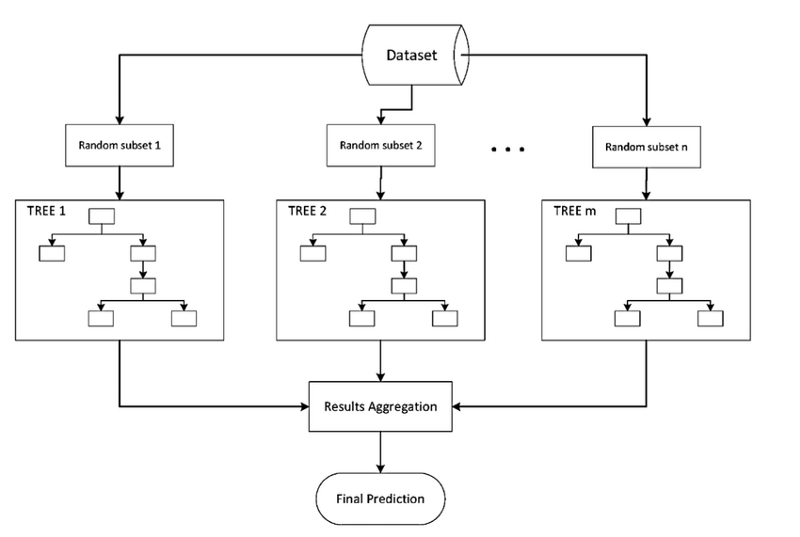
When making Decision Trees, there are several factors we must take into consideration: On what features do we make our decisions on? What is the threshold for classifying each question into a yes or no answer? In the first Decision Tree, what if we wanted to ask ourselves if we had friends to play with or not. If we have friends, we will play every time. If not, we might continue to ask ourselves questions about the weather. By adding an additional question, we hope to greater define the Yes and No classes.

This is where Ensemble Methods come in handy! Rather than just relying on one Decision Tree and hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at each split, and make a final predictor based on the aggregated results of the sampled Decision Trees.



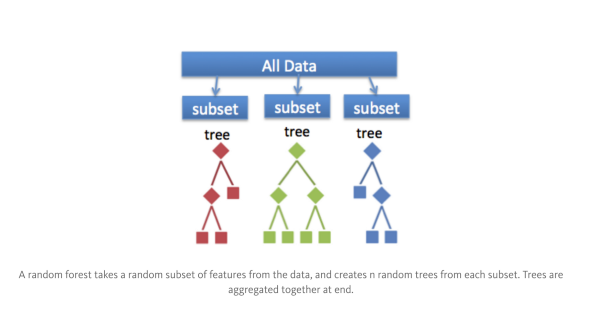
**Types of Ensemble Methods**

1. ***BAGG***ing, or ***B***ootstrap ***AGG***regating. **BAGG**ing gets its name because it combines ***B***ootstrapping and ***Agg***regation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor. The image below will help explain:



Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

2. **Random Forest** Models. Random Forest Models can be thought of as **BAGG**ing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor. Refer to the image for a better understanding.



Similar to BAGGing, bootstrapped subsamples are pulled from a larger dataset. A decision tree is formed on each subsample. HOWEVER, the decision tree is split on different features (in this diagram the features are represented by shapes).

## Simple Ensemble Techniques

In this section, we will look at a few simple but powerful techniques, namely:

1. Max Voting
2. Averaging
3. Weighted Averaging

### Max Voting

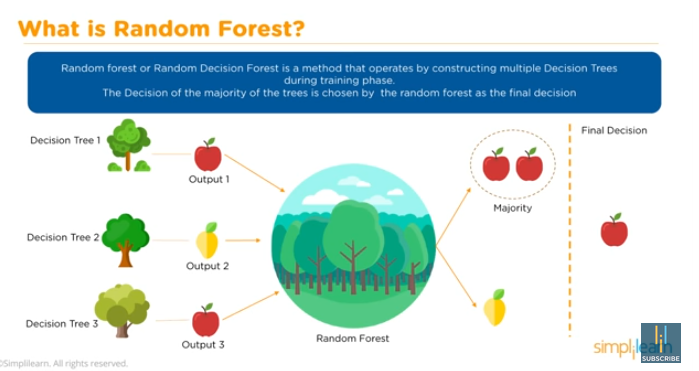
The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

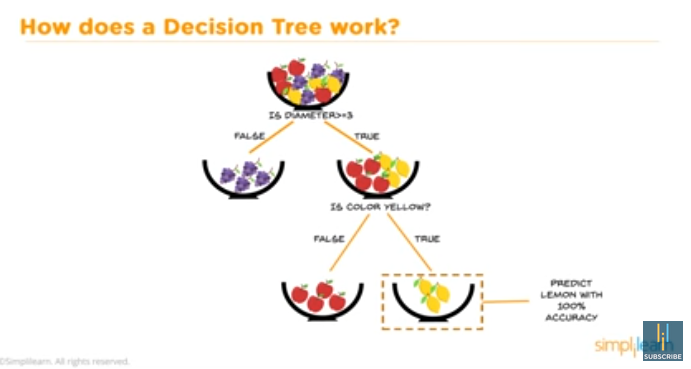
### Averaging

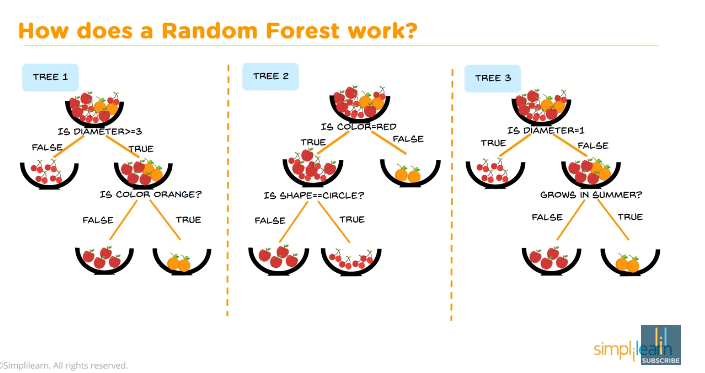
Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

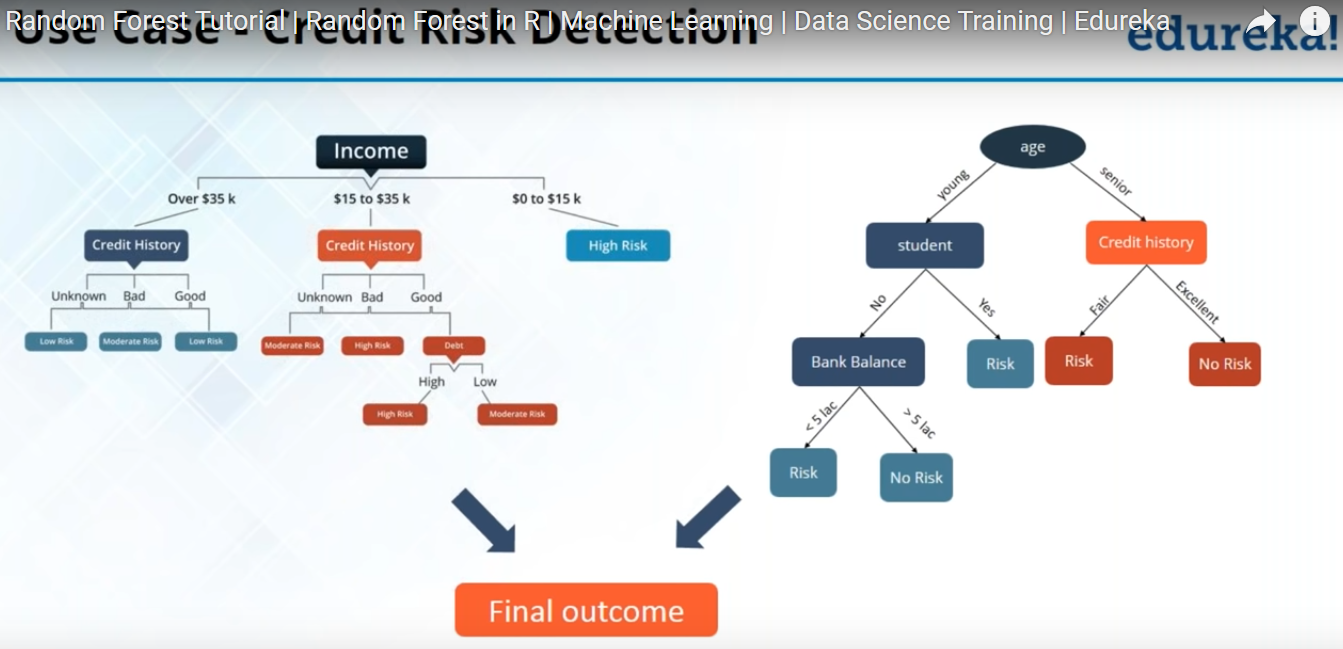
### Weighted Average

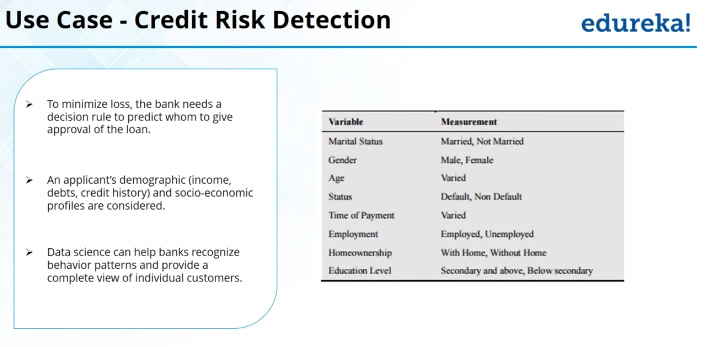
This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. For instance, if two of your colleagues are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.

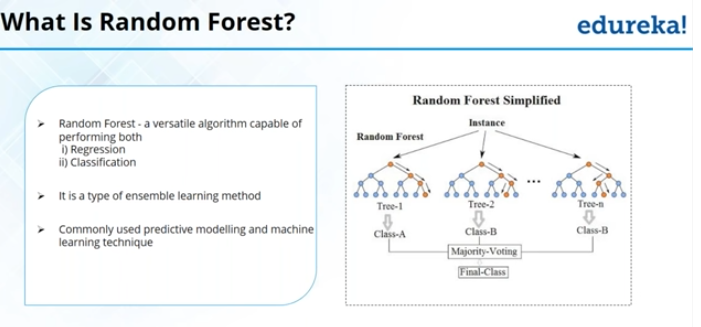
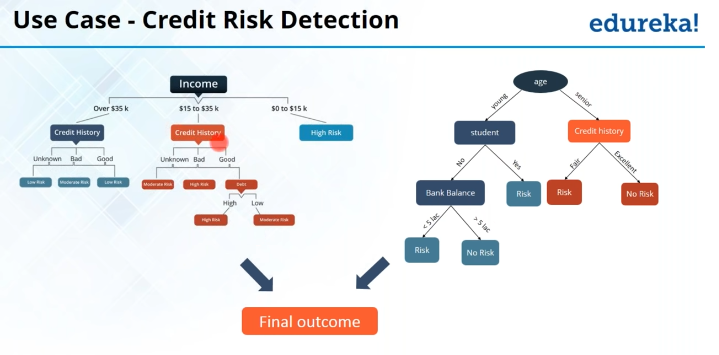






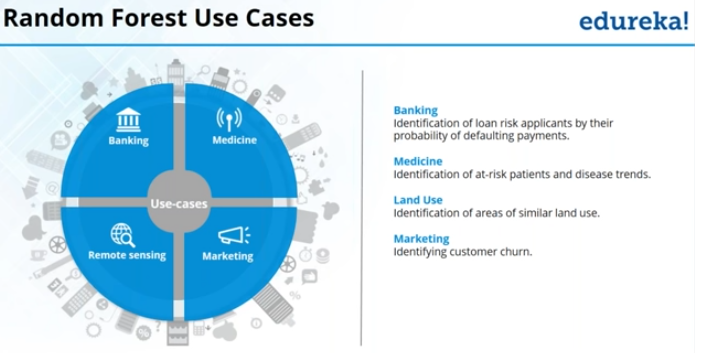


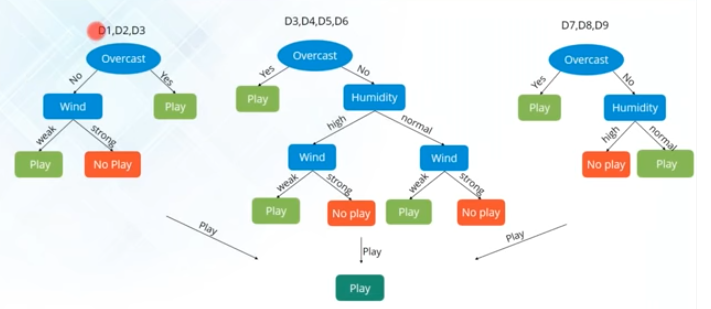
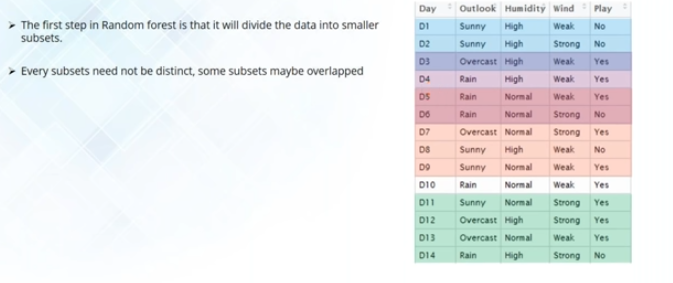
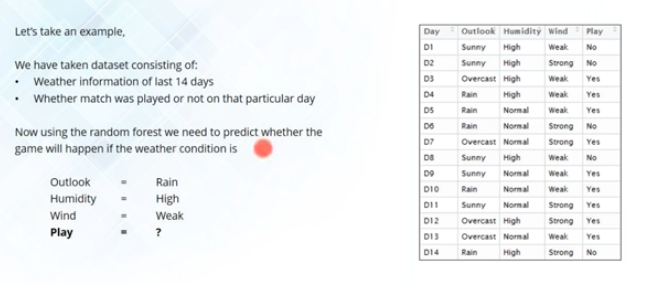
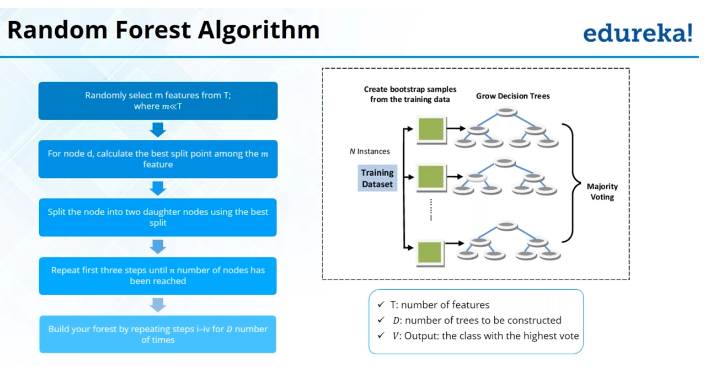


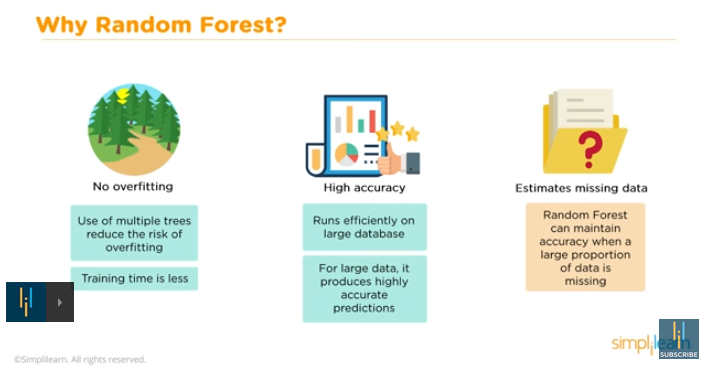


### Use Cases:

The random forest algorithm is used in a lot of different fields, like Banking, Stock Market, Medicine and E-Commerce. In Banking it is used for example to detect customers who will use the bank’s services more frequently than others and repay their debt in time. In this domain it is also used to detect fraud customers who want to scam the bank. In finance, it is used to determine a stock’s behaviour in the future. In the healthcare domain it is used to identify the correct combination of components in medicine and to analyze a patient’s medical history to identify diseases. And lastly, in E-commerce random forest is used to determine whether a customer will actually like the product or not







### Advantages and Disadvantages:

Like I already mentioned, an advantage of random forest is that it **can be used for both regression and classification** tasks and that it’s easy to view the relative importance it assigns to the input features.

**Random Forest is also considered as a very handy and easy to use algorithm, because it’s default hyperparameters often produce a good prediction result**. The number of hyperparameters is also not that high and they are straightforward to understand.

One of the big problems in machine learning is overfitting, but most of the time this won’t happen that easy to a random forest classifier**. That’s because if there are enough trees in the forest, the classifier won’t overfit the model.**

The main limitation of Random Forest is that a large number of trees **can make the algorithm to slow and ineffective for real-time predictions**. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

**And of course Random Forest is a predictive modeling tool** and not a descriptive tool. That means, if you are looking for a description of the relationships in your data, other approaches would be preferred.

**Random Forest adds additional randomness to the model,** while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

**Deep“ decision trees might suffer from overfitting.** Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets.

**Example:**

**Asking friend which movie to watch or where to go for vacation**

### Important Hyperparameters:

* 1. **Firstly,** there is the **„n\_estimators“** hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.
  2. Another important hyperparameter is **„max\_features“**, which is the maximum number of features Random Forest considers to split a node. Sklearn provides several options, described in their [documentation](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html).
  3. The last important hyper-parameter we will talk about in terms of speed, is **„min\_sample\_leaf “**. This determines, like its name already says, the minimum number of leafs that are required to split an internal node.
  4. The **„n\_jobs“** hyperparameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of “-1” means that there is no limit.
  5. **„random\_state“** makes the model’s output replicable. The model will always produce the same results when it has a definite value of random\_state and if it has been given the same hyperparameters and the same training data.
  6. Lastly, there is the **„oob\_score“** (also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples. It is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

### Proximities

These are one of the most useful tools in random forests. The proximities originally formed a NxN matrix. After a tree is grown, put all of the data, both training and oob, down the tree. If cases k and n are in the same terminal node increase their proximity by one. At the end, normalize the proximities by dividing by the number of trees.

**Missing value replacement for the training set**

Random forests has two ways of replacing missing values. The [first way](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_manual.htm#l7) is fast. If the mth variable is not categorical, the method computes the median of all values of this variable in class j, then it uses this value to replace all missing values of the mth variable in class j. If the mth variable is categorical, the replacement is the most frequent non-missing value in class j. These replacement values are called fills.

The [second way](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_manual.htm#l7) of replacing missing values is computationally more expensive but has given better performance than the first, even with large amounts of missing data. It replaces missing values only in the training set. It begins by doing a rough and inaccurate filling in of the missing values. Then it does a forest run and computes proximities.

If x(m,n) is a missing continuous value, estimate its fill as an average over the non-missing values of the mth variables weighted by the proximities between the nth case and the non-missing value case. If it is a missing categorical variable, replace it by the most frequent non-missing value where frequency is weighted by proximity.

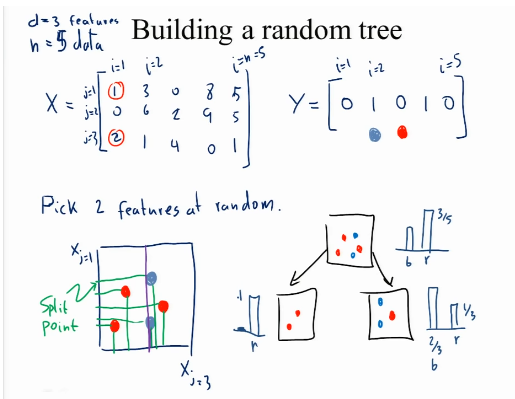
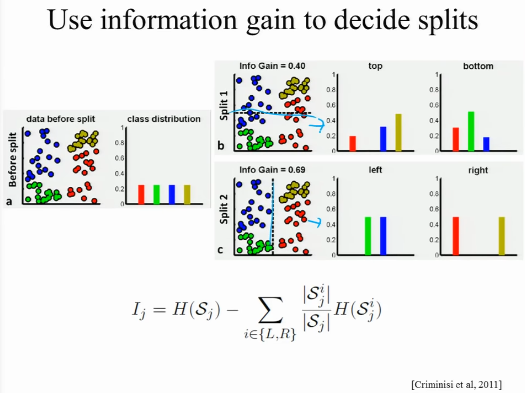
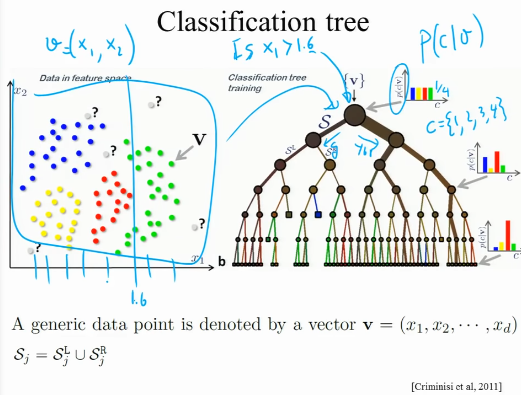
### Outliers

Outliers are generally defined as cases that are removed from the main body of the data. Translate this as: outliers are cases whose proximities to all other cases in the data are generally small. A useful revision is to define outliers relative to their class. Thus, an outlier in class j is a case whose proximities to all other class j cases are small

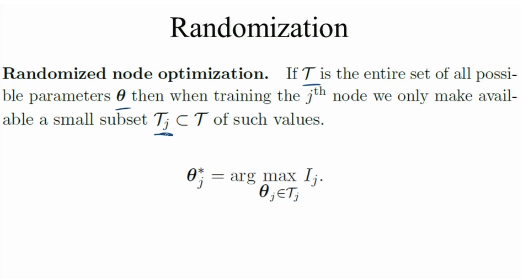
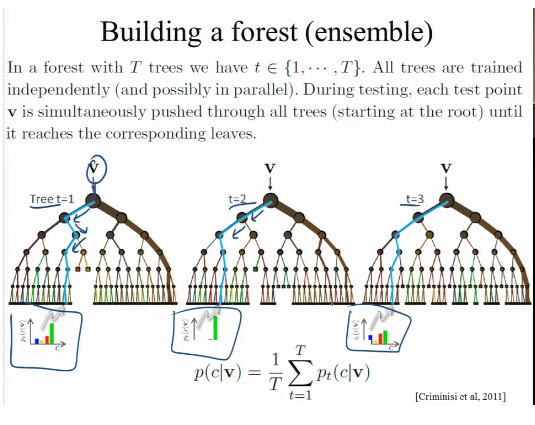
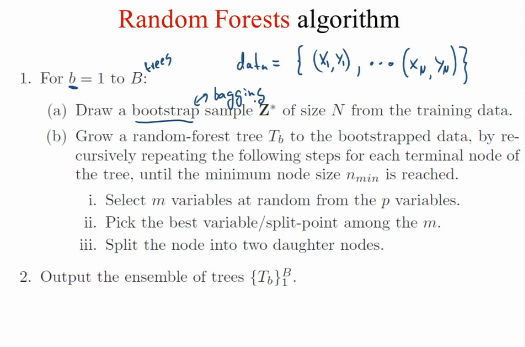
**Accuracy of Random forest:**

1. **accuracy\_score** method,
2. Confusion matrix
3. OOB error rate/OOB percentage of variance explained: This can be viewed when you print the random forest model object.
4. Perform a cross validation/out-of-sample forecast. You may want to look at rfcv function in randomForest package.

**From USC:**

****

**Different tree can be trained at different machines and then aggregation can be taken for final results.**

****

**This is high variance classifier due to high randomness.**

**Little bias** because all the tree has different input data set so giving different results. Each tree is correct because having little information of data they give different result.

**High variance:** Each tree is different from all other trees. So high variance

