**Tree models**

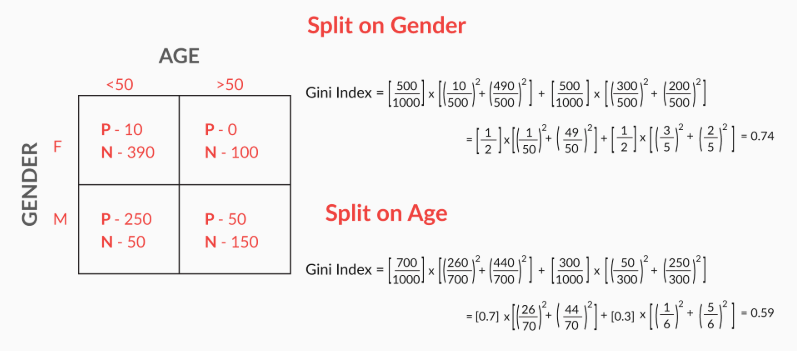
Regression with Decision Trees  
Leaves in classification contain labels. **Feedback :**In classification, the target variable is discrete. Hence, each data point in a leaf has an associated class label.

Leaves in regression contain models. **Feedback :**True. Each leaf in regression contains a model that is used to make predictions.

Session 2:

* Gini index
* Homogeneity measures
* Entropy and information Gain
* Splitting by R-Squared

**Gini Index**

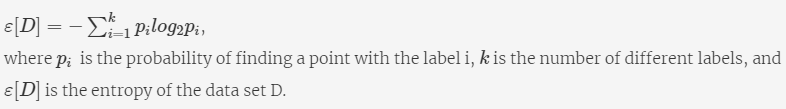


Gini is a measure of entropy,

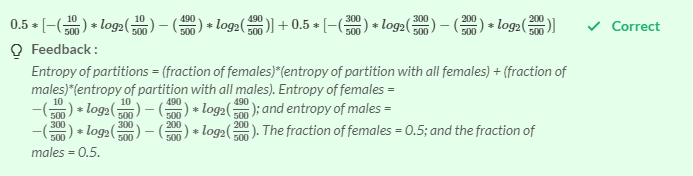
* Gini becomes 1 when pi is 1

Entropy is a measure of chaos

* Entropy becomes zero when pi is 1



Entropy



<https://www.slideshare.net/21_venkat/decision-tree-53154033>

So, the following steps are involved in decision tree construction:

1. A decision tree first decides on an attribute to split on.
2. To select this attribute, it measures the homogeneity of the nodes before and after the split.
3. There are various ways in which you can measure the homogeneity.
4. You have the Gini index and information gain for classification; you also have R2 for the regression, to measure the homogeneity.
5. The attribute that results in a maximum homogeneous data set is then selected for splitting.
6. Then, this whole cycle is repeated till you get a sufficiently homogeneous data set.

**Bias**: Bias describes how well a model matches the training set. A model with high bias won’t match the data set closely, while a model with low bias will match the data set very closely. Bias comes from models that are overly simple and fail to capture the trends present in the data set.

**Variance**: Variance describes how much a model changes when you train it using different portions of your data set. A model with high variance will have the flexibility to match any data set that’s provided to it, potentially resulting in dramatically different models each time. Variance comes from models that are highly complex, employing a significant number of features.

Though there are various ways to truncate or prune trees, the DecisionTreeClassifier function in sklearn provides the following hyperparameters which you can control:

1. **criterion (Gini/IG or entropy):** It defines the function to measure the quality of a split. Sklearn supports “gini” criteria for Gini Index & “entropy” for Information Gain. By default, it takes the value “gini”.
2. **max\_features:** It defines the no. of features to consider when looking for the best split. We can input integer, float, string & None value.
   1. If an integer is inputted then it considers that value as max features at each split.
   2. If float value is taken then it shows the percentage of features at each split.
   3. If “auto” or “sqrt” is taken then max\_features=sqrt(n\_features).
   4. If “log2” is taken then max\_features= log2(n\_features).
   5. If None, then max\_features=n\_features. By default, it takes “None” value.
3. **max\_depth:** The max\_depth parameter denotes maximum depth of the tree. It can take any integer value or None. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples. By default, it takes “None” value.
4. **min\_samples\_split:** This tells above the minimum no. of samples required to split an internal node. If an integer value is taken then consider min\_samples\_split as the minimum no. If float, then it shows percentage. By default, it takes “2” value.
5. **min\_samples\_leaf:** The minimum number of samples required to be at a leaf node. If an integer value is taken then consider - -min\_samples\_leaf as the minimum no. If float, then it shows percentage. By default, it takes “1” value.

**Pruning**

If the accuracy after pruning on the unseen data decreases significantly, then

You shouldn’t prune the branch.

**Feedback :***You should prune only if the accuracy after pruning does not decrease.*

*min\_sample\_split  tells above the minimum no. of samples reqd. to split an internal node. If an integer value is taken then consider min\_samples\_split as the minimum no. If float, then it shows percentage. By default, it takes “2” value.*

*min\_sample\_leaf is the minimum number of samples required to be at a leaf node. If an integer value is taken then consider - -min\_samples\_leaf as the minimum no. If float, then it shows percentage. By default, it takes “1” value.*

X

Random Forest:

 Random forests use a technique known as **bagging**, which is an **ensemble** method. So before diving into random forests, let's first understand 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.**

**Diversity** ensures that the models serve **complementary** purposes, which means that the individual models make predictions **independent of each other**. The advantages of this are different depending on the type of ensemble.

Diversity ensures that even if some trees overfit, the other trees in the ensemble will neutralise the effect. The independence among the trees results in a **lower variance** of the ensemble compared to a single tree.

**Acceptability** implies that each model is at least **better than a random model**. This is a pretty lenient criterion for each model to be accepted into the ensemble, i.e. it has to be at least better than a random guesser.

If two models give the same answers on a random data, it will be totally coincidental

**Feedback :***Diversity represents independence, i.e. models are not correlated (and do not get influenced by) other models. This means that the answers (predictions) given by two models are independent of each other (you’ll study how this is achieved in a short while).*

**Bagging**

To understand random forests, you will first need to understand **bagging**, an ensemble method. Bagging stands for **bootstrapped aggregation.**It is a technique for choosing random samples of observations from a dataset. Each of these samples is then used to train each tree in the forest.

As you study bagging, keep in mind that it is just a **sampling technique** and is **not specific to random forests.**

**

**Bootstrapping** means creating bootstrap samples from a given data set. A bootstrap sample is created by sampling the given data set **uniformly** and **with replacement**. A bootstrap sample typically contains about 30-70% data from the data set. **Aggregation** implies combining the results of different models present in the ensemble

A random subset of observations is chosen every time a new tree is built in a forest.

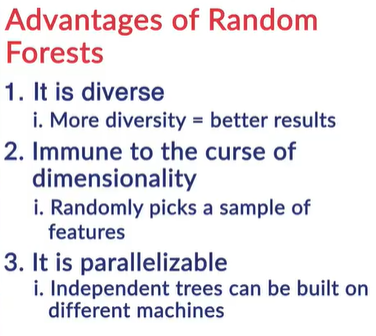
**Feedback :**A different random subset of observations is chosen, which is called the bootstrap sample, for each tree that is to be built in the forest. This is called bootstrapping.

A random subset of features is chosen every time a node is being split inside a tree.

**Feedback :**After the bootstrap sample is selected, tree building starts, and a random subset of features is selected at each node in order to split it. This is what makes random forests even better than a simple bagging algorithm.

In a random forest, a random sample of features is chosen at each node split, which does not happen in bagging.

**Feedback :***Bagging includes the creation of different bootstrap samples for different models, and aggregating the results of the models. Random forests use this technique along with randomly selecting features at each node while splitting it.*



The models will not be diverse enough

**Feedback :***If a few variables are prominent, a large number of trees will have them as important nodes, and they will look similar. Similar trees violate the condition of diversity.*

**Bootstrapping** means creating bootstrap samples from a given data set. A bootstrap sample is created by sampling the given data set **uniformly** and **with replacement**. A bootstrap sample typically contains about 30-70% data from the data set. **Aggregation** implies combining the results of different models present in the ensemble.

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.

## Advantages of Random Forests

**Diversity** arises because you create each tree with a **subset of the attributes/features/variables**, i.e. you don’t consider all the attributes while making each tree. The choice of the attributes considered for each tree is **random**. This ensures that the trees are independent of each other.

**Stability arises** because the answers given by a large number of trees average out. A random forest has a **lower model variance** than an ordinary individual tree.

**Immunity to the curse of dimensionality**: Since each tree does not consider all the features, the feature space (the number of features a model has to consider) reduces. This makes the algorithm immune to the curse of dimensionality. A large feature space causes computational and complexity issues.

**Parallelizability**: You need a number of trees to make a forest. Since two trees are independently built on different data and attributes, they can be built separately. This implies that you can make full use of your multi-core CPU to build random forests. Suppose there are 4 cores and 100 trees to be built; each core can build 25 trees to make a forest.

**Testing /training** data and the **OOB or out-of-bag error**: You always want to avoid violating the fundamental tenet of learning: “not testing a model on what it has been trained on”. While building individual trees, you choose a **random** subset of the observations to train it. If you have 10,000 observations, each tree may only be made from 7000 (70%) randomly chosen observations. OOB is the mean prediction error on each training sample xᵢ, using only the trees that do not have xᵢ in their bootstrap sample. If you think about it, this is very similar to a **cross-validation error**. In a CV error, you can measure the performance on the subset of data the model hasn’t seen before.

In fact, it has been proven that using an OOB estimate is as accurate as using a test data set of a size equal to the training set.

Thus, the OOB error completely omits the need for set-aside test data (though you can still work with test data like you have been doing, at the cost of eating into the training data).

Bootstrapping implies that each tree in a RF is built on randomly chosen observations

**Feedback :**The word ‘random’ in random forests refers to the random choice of bootstrapped observations

Random choice of attributes while splitting at nodes ensures diversity in a random forest

**Feedback :**Random choice of attributes ensures that the prominent features do not appear in every tree, thus ensuring diversity.

Are typically unstable

**Feedback :***If you have only one tree, you have to rely on the decision it makes. The decision a single tree makes (on unseen data) depend highly on the training data since trees are unstable. In a forest, even if a few trees are unstable, averaging out their decisions ensures that you are not making mistakes because of a few trees’ unstable behaviour.*

False

**Feedback :***While it is well known that random forests are better, in terms of accuracy, than a single decision tree, it cannot be said that they are better than every possible decision tree. It is just more difficult to build a decision tree that is better than a random forest. In fact, there may be several trees that provide better predictions on unseen data.*

A larger number of trees will result in a lower variance of the ensemble.

**Feedback :***Variance means how much a model (ensemble here) changes with changes in the training data. If a large number of trees is at work, then even if some of them show a high instability (extreme variation in the trees and their predictions), the ensemble as a whole will reduce the variance by averaging out the results of each tree.*

In the last segment, you learnt that the **OOB error** is almost as good as the **cross-validation error**. The final prediction is the **aggregation** of all the predictions of individual decision trees. Remember that each tree in a random forest is only trained on a **random subset** of the training set, which is called a **bootstrapped sample**. This means that for each sample (observation), there are several trees that did not include that sample, and to these trees, this sample is **unseen**. Let’s understand this better.   
Training set

**Feedback :***Only the training set is used while calculating the OOB error, which is why it gives a good idea of model performance on the unseen data without using a test set.*

All the observations of the training set are used to calculate the OOB error.

**Feedback :***Recall that all the observations of the training set are used to calculate the OOB error.*

There are two critical steps in building a random forest:

1. Bagging, or choosing observations in a bootstrapped manner and creating individual trees
2. Random subsetting of features while creating each tree and finding the right splits

Building an ensemble requires the following steps:

1) Take a random sample of observations, say j= 40% of the total and

2) Build S trees by finding all the splits within each tree from a subsetted feature space. The time required is thus proportional to the number of trees S and the time required to build each tree.

The time required to build each tree is proportional to the average number of levels of a tree and the average time required to find the split at each level. To find the right split, only a subset of features f is considered.

Further, the expected number of levels in a tree is given by log(n) where n is the number of observations being considered to build the tree.

S, M and n

**Feedback :***The time required will obviously depend on S. While building each of the S trees, time is spent in creating the levels of trees and time required to find splits among f features. Levels of trees are given by log(n). Finding the right split depends on both n observations and f features because homogeneity will be measured for all f features and n observations.*

Time taken to build Random forest -   
sqrt(M).n.j

**Feedback :***Each split is made by comparing the homogeneity across j= 40% of the n observations. Thus, it has to depend on j and n (more the observations, more the time required to compare homogeneity). The time required to find a split also depends upon the number of features being considered which is sqrt(M).*

**The Effect of max\_features**

You saw that there is an optimal value of max\_features - at very **low values**, the component trees are**too simple to learn anything useful**, while at extremely high values, the component trees **become similar to each other**(and violate the 'diversity' criterion).

**The Effect of n\_estimators**

Also, when you observe the plot of n\_estimators and training and test accuracies, you will see that the as you increase the value of n\_estimators, both the training test accuracies gradually increase. More importantly, the model does not overfit even when its complexity is increasing. This is an important benefit of random forests - you can increase the number of trees as much you like without worrying about overfitting (if your computational resources allow).

There is one more ensemble method: '**Boosting'**. Boosting is also a very popular method in online machine learning competitions, such as Kaggle. The basic idea is to **combine a lot of weak learners to get a strong learner**. **Weak learner** refers to a model that is slightly better than a model that predicts at random. Weak learners are built sequentially on top of each other, giving a **boost** to the overall performance of the model.

**Random Forests**

Which two things are ‘random’ about random forests?

**Subset on which each tree of the random forest is built**

**Feedback :**

Bootstrapping – each tree is built from a random subset. Random split – while growing a tree, the features considered for a split at a node is sampled at random.

**Features examined for determining the best split at a node**

**Feedback :**

Bootstrapping – each tree is built from a random subset. Random split – while growing a tree, the features considered for a split at a node is sampled at random.

**OOB Error**

OOB error in a random forest is similar in spirit to

**Cross validation error**

**Feedback :**

*Exactly like in cross validation, each tree is built on a (different) subset of the data and is tested on the remaining data. The error on data that trees were not trained on is the OOB error.*

**Random Forest**

Large forests (i.e. RF with a large number of decision trees) are likely to be more generalizable than a single decision tree because

**The random forest takes the average of the predictions from all trees, so it does not overfit.**

**Feedback :**

*Decision trees in large numbers together classify a point more accurately than an individual tree.*

**Time Taken to build a Random Forest**

If f is the total number of features and n is the number of data points in the training data. A crude estimate of the time taken to build a random forest is proportional to the number of trees times

sqrt(f)\*n\*log(n)

**Random Forest**

Which of the following information is not true with respect to Random Forest?

**The training process individual tree of the Random forest is the same as training a decision tree.**

**Feedback :**

*This is not true. Training process of each tree in Random forest is same as decision tree except with the difference that at each node in the tree only a random selection of features is used for the split in that node.*

.