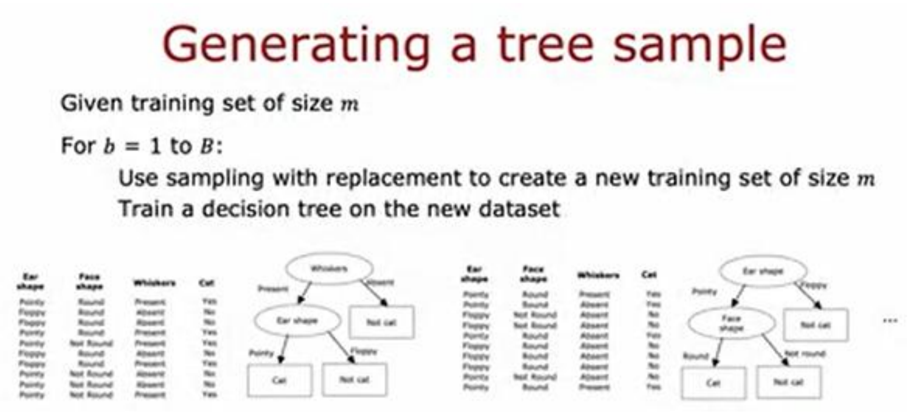
Random Forest Algorithm

Now that we have a way to use sampling with replacement to create new training sets that are a bit similar to but also quite different from the original training set. We're ready to build our first tree ensemble algorithm. In particular in this video, we'll talk about the random forest algorithm which is one powerful tree ensemble algorithm that works much better than using a single decision tree. Here's how we can generate an ensemble of trees.

If you are given a training set of size M, then for B equals 1 to capital b so we do this capital B times. You can use something with replacement to create a new training set of size M. So if you have 10 training examples, you will put the 10 training examples in that virtual bag and sample of replacement 10 times to generate a new training set with also 10 examples, and then you would train a decision tree on this data set.

So here's the data set I've generated using something with replacement. If you look carefully, you may notice that some of the training examples are repeated and that's okay. And if you train the decision on this data says you end up with this decision tree. And having done this once, we would then go and repeat this a second time. Use something with replacement to generate another training set of M or 10 training examples. This again looks a bit like the original training set but it's also a little bit different. You then train the decision tree on this new data set and you end up with a somewhat different decision tree.


Generating a tree sample

```
Given training set of size $m$
For $b = 1$ to $B$:
    Use sampling with replacement to create a new training set of size $m$
    Train a decision tree on the new dataset
```

Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Round	Present	Yes
Pointy	Not Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Present	Yes
Pointy	Not Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Not Round	Present	Yes

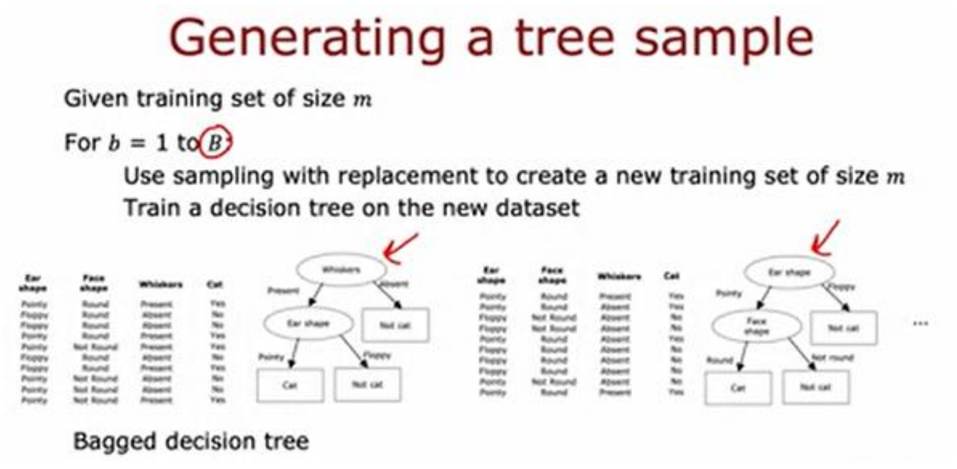
Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Pointy	Round	Absent	Yes
Floppy	Not Round	Absent	No
Floppy	Not Round	Absent	No
Pointy	Round	Absent	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Round	Present	Yes

And so on. And you can do this a total of capital B times. Typical choice of capital B the number of such trees you built might be around a 100 people recommend any value from Say 64, 128. And having built an ensemble of say 100 different trees, you would then when you're trying to make a prediction, get these trees all votes on the correct final prediction.

It turns out that setting capital B to be larger, never hurts performance, but beyond a certain point, you end up with diminishing returns and it doesn't actually get that much better when B is much larger than say 100 or so. And that's why I never use say 1000 trees that just slows down the computation significantly without meaningfully increasing the performance of the overall algorithm. Just to give this particular algorithm a name.

This specific instance creation of tree ensemble is sometimes also called a bagged decision tree. And that refers to putting your training examples in that virtual bag. And that's why also we use the let us lower case B an uppercase B here because that stands for bag. There's one modification to this album that will actually make it work even much better and that changes this algorithm the bagged decision tree into the random forest algorithm.

The key idea is that even with this sampling with replacement procedure sometimes you end up with always using the same split at the root node and very similar splits near the root note. That didn't happen in this particular example where a small change the trainings that resulted in a different split at the root note. But for other training sets it's not uncommon that for many or even all capital B training sets, you end up with the same choice of feature at the root node and at a few of the nodes near the root node.



Generating a tree sample

Given training set of size m

```
For $b = 1$ to $B$
    Use sampling with replacement to create a new training set of size $m$
    Train a decision tree on the new dataset
```

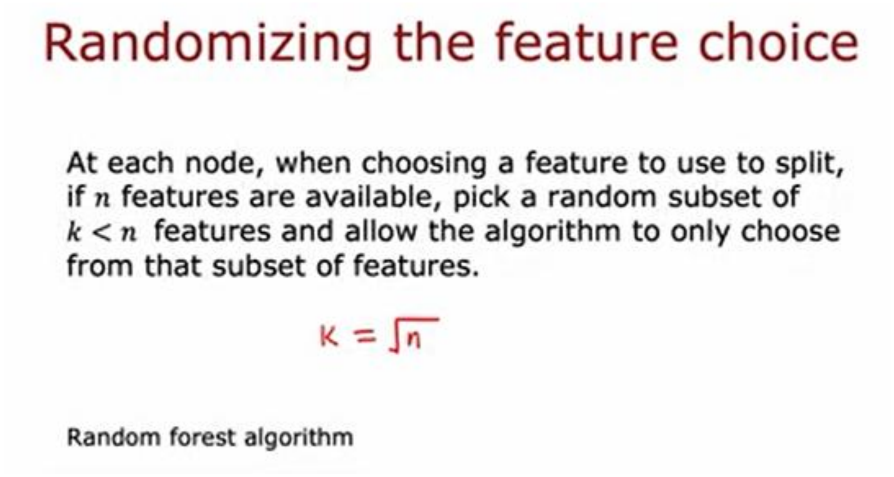
Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Round	Present	Yes
Pointy	Not Round	Present	Yes
Floppy	Round	Absent	No
Floppy	Round	Present	Yes
Pointy	Not Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Not Round	Present	Yes

Ear shape	Face shape	Whiskers	Cat
Pointy	Round	Present	Yes
Pointy	Round	Absent	Yes
Floppy	Not Round	Absent	No
Floppy	Not Round	Absent	No
Pointy	Round	Absent	Yes
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Floppy	Round	Absent	No
Pointy	Not Round	Absent	No
Pointy	Round	Present	Yes

Bagged decision tree

So there's one modification to the algorithm to further try to randomize the feature choice at each node that can cause the set of trees and you learn to become more different from each other. So when you vote them, you end up with an even more accurate prediction. The way this is typically done is at every note when choosing a feature to use to split if end features are available.

So in our example we had three features available rather than picking from all end features, we will instead pick a random subset of K less than N features. And allow the algorithm to choose only from that subset of K features. So in other words, you would pick K features as the allowed features and then out of those K features choose the one with the highest information gain as the choice of feature to use the split. When N is large, say n is Dozens or 10's or hundreds even.



Randomizing the feature choice

At each node, when choosing a feature to use to split, if n features are available, pick a random subset of $k < n$ features and allow the algorithm to only choose from that subset of features.

$K = \sqrt{n}$

Random forest algorithm

A typical choice for the value of K would be to choose it to be square root of N, In our example we have only three features and this technique tends to be used more for larger problems with a larger number of features. And will just further change the algorithm you end up with the random Forest algorithm which will work typically much better and becomes much more robust than just a single decision tree.

One way to think about why this is more robust to than a single decision tree is the sampling with replacement procedure causes the algorithm to explore a lot of small changes to the data already and it's training different decision trees and is averaging over all of those changes to the data that the sampling with replacement procedure causes.

And so this means that any little change further to the training set makes it less likely to have a huge impact on the overall output of the overall random forest algorithm. Because it's already explored and it's averaging over a lot of small changes to the training set. Before wrapping up this video there's just one more thought I want to share with you Which is, where does a machine learning engineer go camping? In a random forest.

All right. Go and tell that joke to your friends. I hope you enjoy it. The random forest is an effective algorithm and I hope you better use it in your work. Beyond the random forest It turns out there's one other algorithm that works even better. Which is a boosted decision tree. In the next video, let's talk about a boosted decision tree algorithm called XG boost