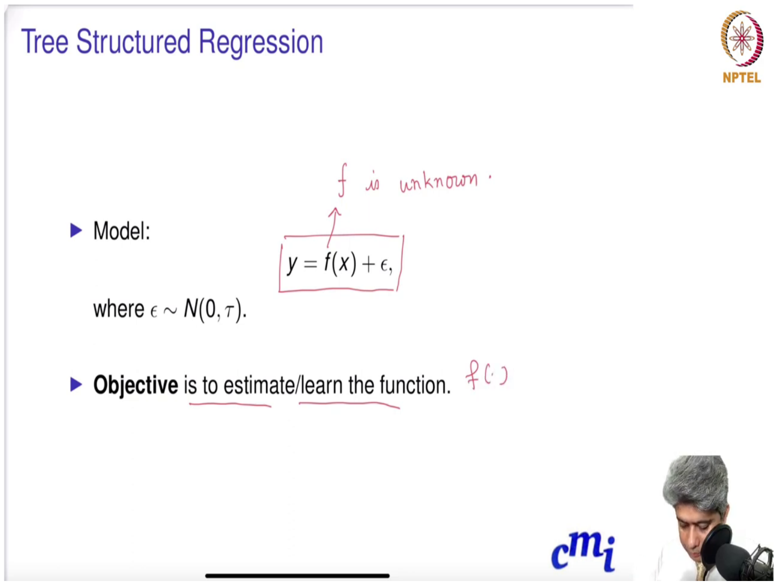
Predictive Analytics - Regression and Classification

Prof. Sourish Das
Department of Mathematics
Indian Institute of Technology, Madras

Lecture - 51
Tree Structured Regression

Hello all welcome to the lecture 16 part A. In this lecture, in this discussion we are going to discuss Tree Structured Regression.

(Refer Slide Time: 00:30)



Tree Structured Regression

- Model:

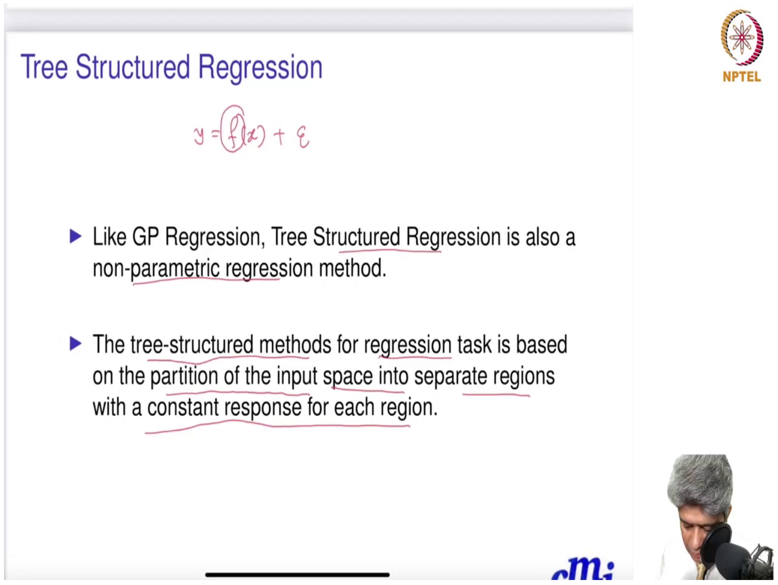
$$\boxed{y = f(x) + \epsilon,}$$

where $\epsilon \sim N(0, \tau)$.

- **Objective** is to estimate/learn the function. $f(\cdot)$

So, we will in a tree structured regression, we are going to have y as a function of x plus epsilon, where we do not know what is the functional form f, this functional form f is unknown ok. Now, obviously, our objective is to estimate or learn the function f ok. So, that is our objective.

(Refer Slide Time: 01:07)



Tree Structured Regression

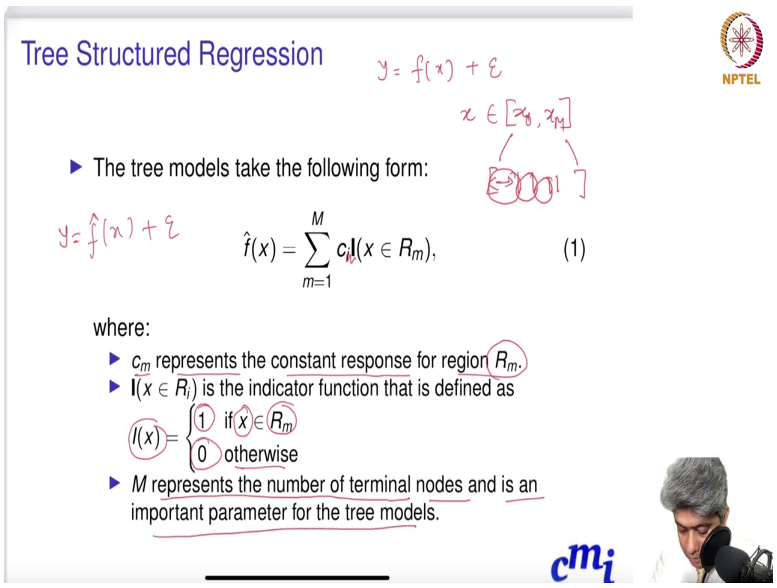
$y = f(x) + \varepsilon$

- Like GP Regression, Tree Structured Regression is also a non-parametric regression method.
- The tree-structured methods for regression task is based on the partition of the input space into separate regions with a constant response for each region.

And like GP regression, tree structured regression is also a non-parametric regression method ok. So, because remember that what is our model y is function of x plus epsilon. And GP regression also tries to estimate this f unknown f and tree structured regression also tries to estimate that f.

But the methodology is very different than GP regression. Let us see how tree structured regression works. The tree structured method for regression task is based on the partition of the input space into separate region and with constant response for each region. So, I will talk about it more about it.

(Refer Slide Time: 02:07)



Tree Structured Regression

- The tree models take the following form:

$$\hat{f}(x) = \sum_{m=1}^{M} c_i \mathbf{I}(x \in R_m), \qquad (1)$$

where:

- c_m represents the constant response for region R_m.
- $\mathbf{I}(x \in R_i)$ is the indicator function that is defined as

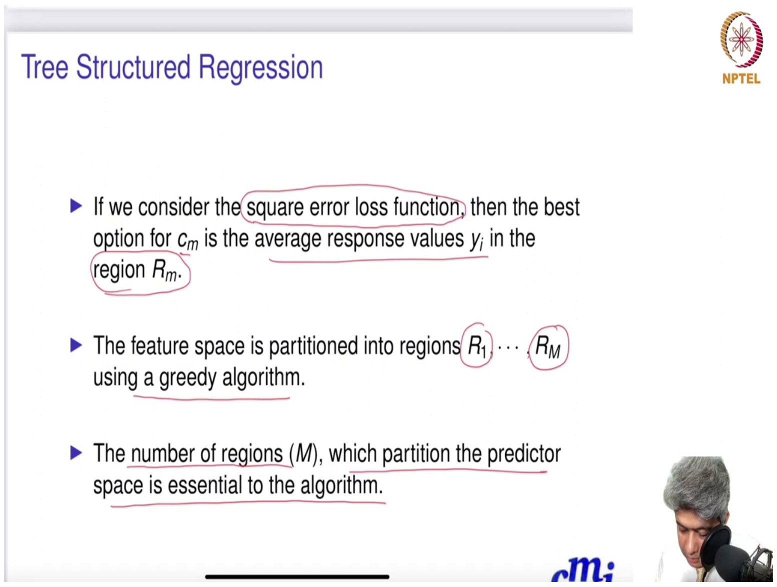
$$I(x) = \begin{cases} 1 & \text{if } x \in R_m \\ 0 & \text{otherwise} \end{cases}$$

- M represents the number of terminal nodes and is an important parameter for the tree models.

So, typically what happens that you have y equal to f of x plus epsilon right and x belongs to some space maybe you know some x 1 or x 0 to say x big m. And then what you do you partition it into different part the x ok. And then for each partition you come up with the estimate for y. And if so basically you want to say f of hat x plus epsilon and in each part you come up with a response a constant response. That is what it is doing.

So, c i now here c m is any of the Ith value here it should be m. c m represents the constant response for region m R m where I x is if the x belongs to R m then I will take it as 1 otherwise I will take it as 0. And m represents the number of terminal nodes and is an important parameter of the tree models we will talk about it.

(Refer Slide Time: 03:31)



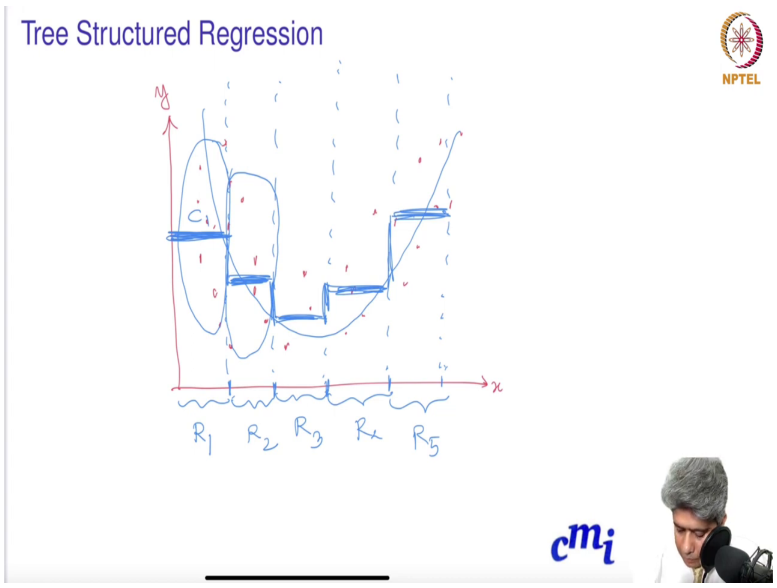
Tree Structured Regression

- If we consider the square error loss function, then the best option for c_m is the average response values y_i in the region R_m.
- The feature space is partitioned into regions $R_1, \cdots, R_M$ using a greedy algorithm.
- The number of regions (M), which partition the predictor space is essential to the algorithm.

c_{m_i}

If we consider square error loss function; if we consider square error loss function then the best option for c m is the average response values of y i in the region R m ok. The feature space is partitioned into regions R 1, R 2, R M using a greedy algorithm. The number of regions which partition the predictor space is essential to the algorithm ok.

(Refer Slide Time: 04:08)



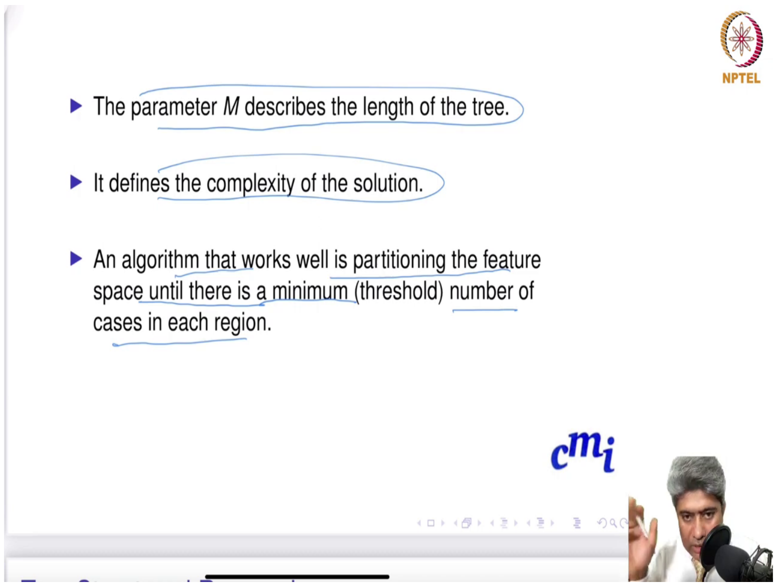
Tree Structured Regression

So, let me give you idea about how this tree structured algorithm works. So, suppose you have one x ok and one y just to understand this and suppose these are the points. So, some sort of you know. Now, what I will do? I will let me just take a different color maybe I take blue.

So, I just partition it into different partition and then here bunch of partition I will take ok here I am partitioning the x. And now in this partition these are the points what is the average of y of these points somewhere here ok. So, this is my C 1 in this partition. So, this is R 1 and this is C 1 this is R 2 in R 2 these are the points. So, and what is the average of these points of y this is my R 2 in this R 3 this is my R 3 in R 4 this is the step and then R 5 this is the value.

So, it is like sort of a create a step function. So, this is the function regression its some sort of non-linear structure it is get captured to true relationship might be like this. And it is getting a kind of a step function kind of behaviour is getting captured by this tree structured regression model ok.

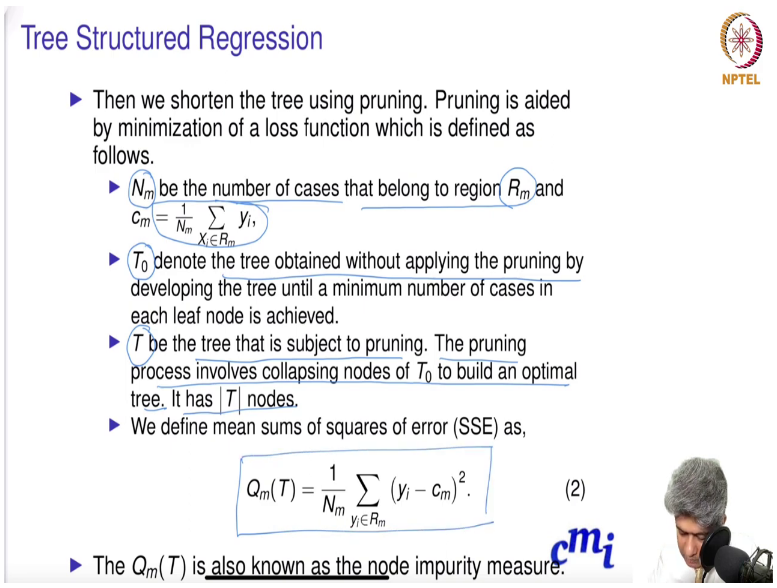
(Refer Slide Time: 06:40)


- The parameter *M* describes the length of the tree.
- It defines the complexity of the solution.
- An algorithm that works well is partitioning the feature space until there is a minimum (threshold) number of cases in each region.

So, here the parameter M describes the length of the tree this defines the complexity of the solutions because and it is very important. So, here I have taken M equal to 5. Now, obviously, if you choose a different M this will be like this would be more bigger or smaller or smoother. So, the smoothness will depend on value of M.

Now, if you choose too many m's then it could be too jittery or you know too smooth. So, this could be this is something you have to be careful about. An algorithm that works well is partitioning the feature space until there is a minimum threshold number of in each case of free case region ok.

(Refer Slide Time: 07:33)



Tree Structured Regression

- Then we shorten the tree using pruning. Pruning is aided by minimization of a loss function which is defined as follows.
  - N_m be the number of cases that belong to region R_m and $c_m = \frac{1}{N_m} \sum_{X_i \in R_m} y_i$,
  - T_0 denote the tree obtained without applying the pruning by developing the tree until a minimum number of cases in each leaf node is achieved.
  - T be the tree that is subject to pruning. The pruning process involves collapsing nodes of T_0 to build an optimal tree. It has $|T|$ nodes.
  - We define mean sums of squares of error (SSE) as,

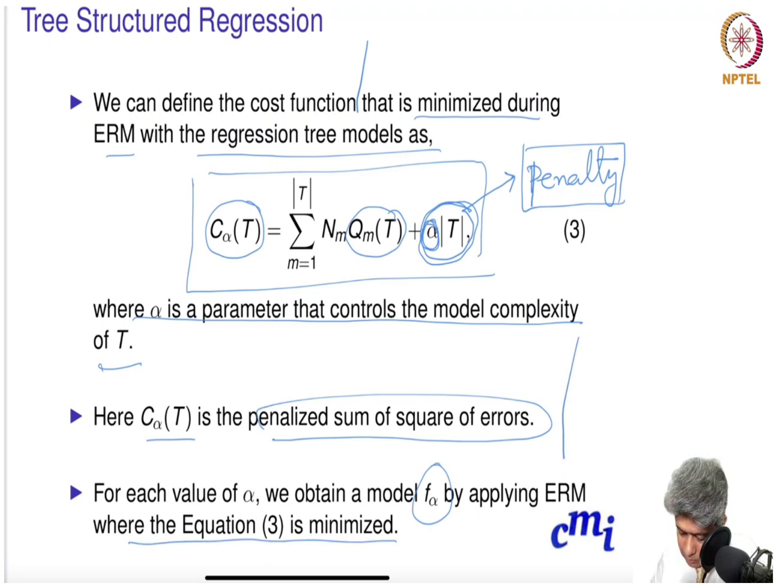
$$\boxed{Q_m(T) = \frac{1}{N_m} \sum_{y_i \in R_m} (y_i - c_m)^2.} \qquad (2)$$

- The $Q_m(T)$ is also known as the node impurity measure.

So, then we shorten the tree using pruning. So, we do some pruning of the tree as well you have to do typical pruning. Pruning is aided by minimization of a loss function which is defined as follows ok. So, how we define? So, N m is the true number of on the number of cases that belong to region R m and c m is the average of the average of the responses in that region.

Suppose T 0 denote the tree obtained without applying any pruning ok. Without applying any pruning it just grows the allows you to grow the tree as much as possible with a small small bean and that is perfectly fine and then T be the tree that subject to pruning. Then you do some pruning that you can now I need to reduce the number of branches. The pruning process involves collapsing nodes T 0 to build optimal tree it has mod T nodes. So, we define mean sum of squares of error as Q T m 1 by N m summation y i minus c m square whole square.

(Refer Slide Time: 08:50)



Tree Structured Regression

- We can define the cost function that is minimized during ERM with the regression tree models as,

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \underbrace{\alpha|T|}_{\text{Penalty}}, \qquad (3)$$

where α is a parameter that controls the model complexity of T.

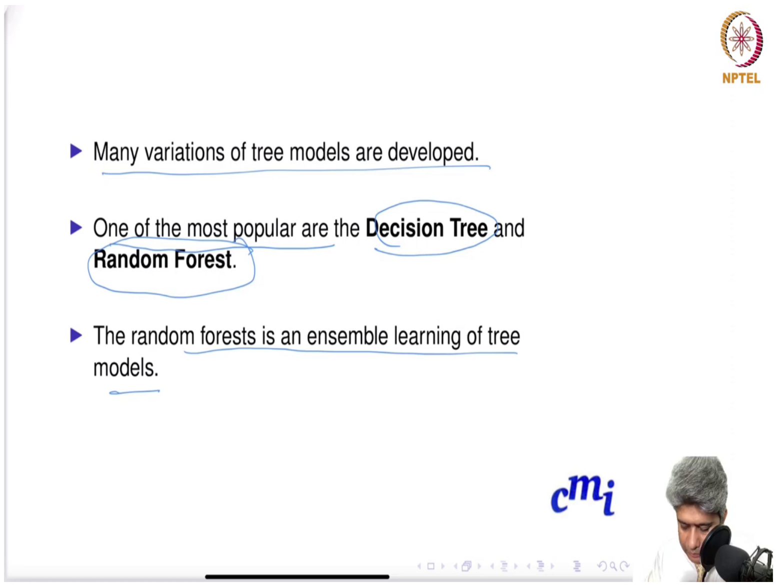
- Here $C_\alpha(T)$ is the penalized sum of square of errors.
- For each value of α, we obtain a model f_α by applying ERM where the Equation (3) is minimized.

Q m T is also known as the node impurity measure ok. So, we can define the cost function that is minimized during ERM with the regression tree models as like this ok. So, now you have to be careful here, what is Q m? T Q m T is the essentially sum of squares of errors squares of errors sums of squares of errors right. So, this is the error you square it sums it.

So, it is sort of a mean squared error in the mth region ok. Now, what it is doing is essentially multiply it by N m, so this cancels. So, what you left with just summation y i c m square then taking of this sum over all the regions and then it is putting a penalty this is a penalty ok penalty, so this is a penalty. So, alpha mod T is a penalty.

So, where alpha is the parameter controls the model complexity of the T that how much model complexity you will allow. So, alpha is the tuning parameter and mod T is the L 1 penalty where C alpha T is the penalized sum of squares of error. So, C alpha T is the penalized sum of squares of error ok. For each value of alpha you obtain a f of alpha by applying ERM and where equation is the minimized.

(Refer Slide Time: 10:37)



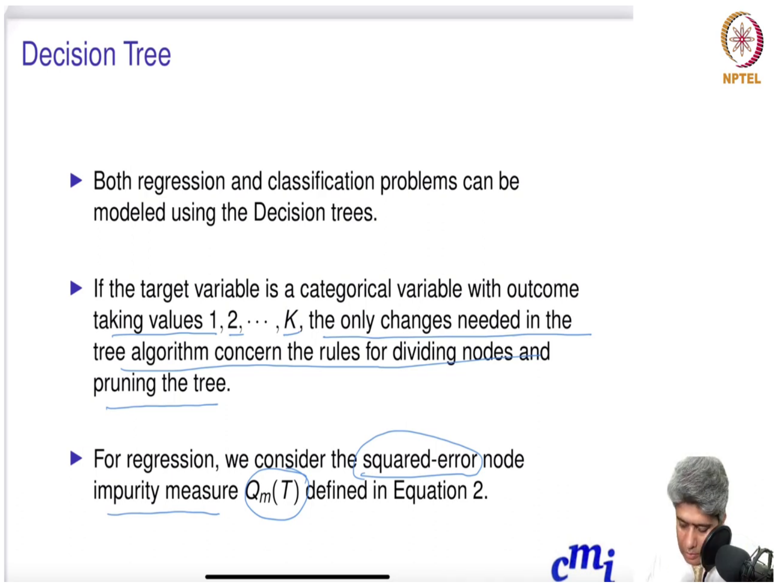
- Many variations of tree models are developed.
- One of the most popular are the **Decision Tree** and **Random Forest**.
- The random forests is an ensemble learning of tree models.

So, many variations of the tree models are developed; one of the popular model is decision tree and random forest. Now, (Refer Time: 10:46) you can be little bit confused that typically in a machine learning literature decision tree or random forest that typically develop for classification problem. Yes, you are right decision tree and random forest are generally developed for classification binary classification or multi-class classification problems.

But interestingly decision tree and random forest can also be applied for regression as well. So, the way I just showed you. So, in that case instead of using entropy or instead of using the likelihood of binary class classification you have to use the sum of squares of error. Like obviously, in each node whatever the sum of squares of error or each region what is the sum of squares of error you have to appropriately calculate and then optimize it and come up with the optimal tree.

Of course, if you put penalized penalty then you will get a pruning pruned optimal tree regression in that way. So, the random forest is an ensemble learning of tree models.

(Refer Slide Time: 12:05)

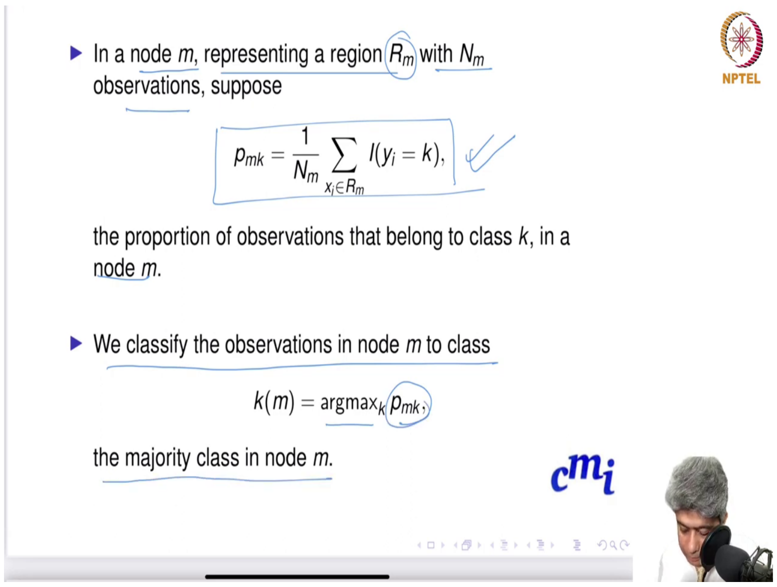


Decision Tree

- Both regression and classification problems can be modeled using the Decision trees.
- If the target variable is a categorical variable with outcome taking values $1, 2, \cdots, K$, the only changes needed in the tree algorithm concern the rules for dividing nodes and pruning the tree.
- For regression, we consider the squared-error node impurity measure $Q_m(T)$ defined in Equation 2.

In decision tree both regression and classification problem can be modelled using decision tree. If the target variables is a categorical variable with outcome values taking values say 1, 2, 3 up to K, the only changes needed in the tree algorithm concern rules dividing the node and pruning the tree ok. So, for regression we consider the squared error node impurity measure Q m T. Basically, this is sum of squares of error at each node or at each region that is it ok.

(Refer Slide Time: 12:51)



- In a node m, representing a region R_m with N_m observations, suppose

$$\boxed{p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),}$$

the proportion of observations that belong to class k, in a node m.

- We classify the observations in node m to class

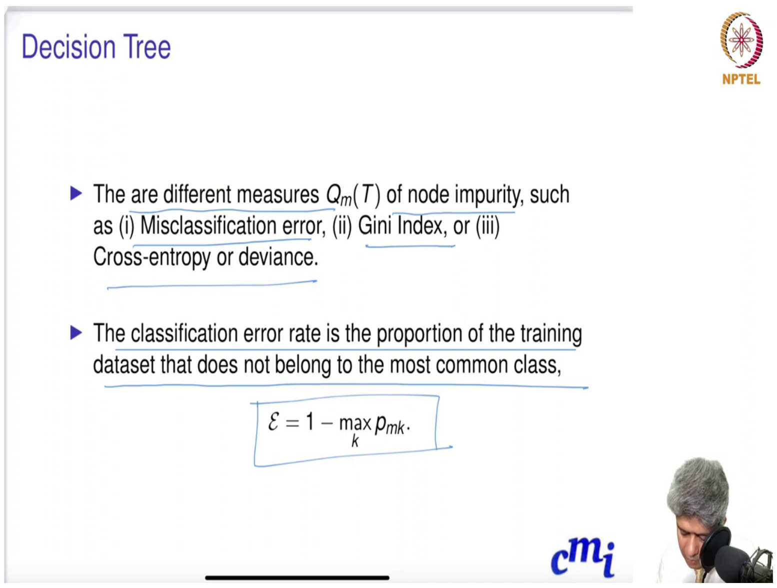
$$k(m) = \text{argmax}_k \, p_{mk},$$

the majority class in node m.

So, in decision tree in a node m representing region R m with N m observation. So, pmk is simply average of samples in the that node. The proportion of observation that belong to class k in a node m. So, as I said if you go back to the let us go back to the. So, here it is, so this is my region this is one region in that region how many points are there these points you just take the average of those points on y and that is your that is what exactly it is talking about ;that is what exactly this guy is talking about ok.

And we classify the observation of node m classes, but basically R you take argmax over k p mk the majority class in the node m.

(Refer Slide Time: 13:54)



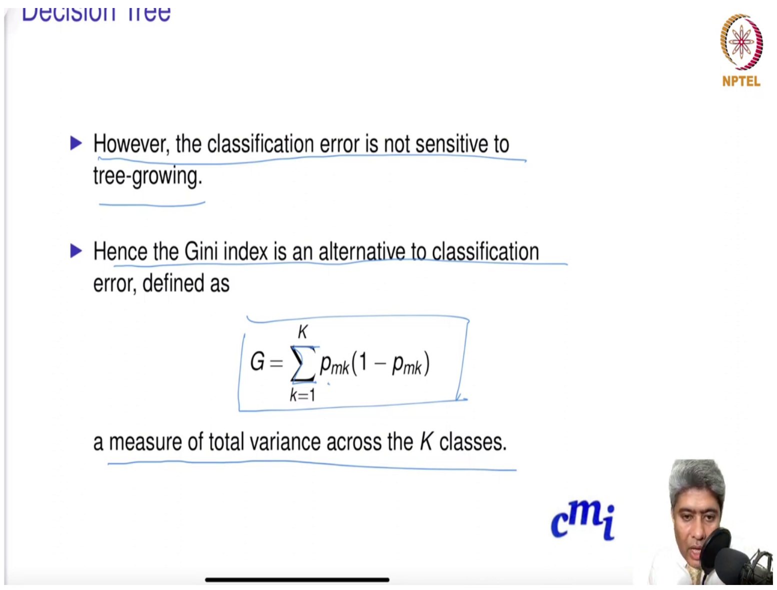
Decision Tree

- The are different measures $Q_m(T)$ of node impurity, such as (i) Misclassification error, (ii) Gini Index, or (iii) Cross-entropy or deviance.
- The classification error rate is the proportion of the training dataset that does not belong to the most common class,

$$\boxed{\mathcal{E} = 1 - \max_k p_{mk}.}$$

In decision there are different measures of Q m T of node impurity such as misclassification error, Gini index, cross entropy or deviance. The classification error rate is the proportion of the training dataset that does not belong to the common class. So, epsilon equal to 1 minus max of pmk.

(Refer Slide Time: 14:17)



Decision Tree

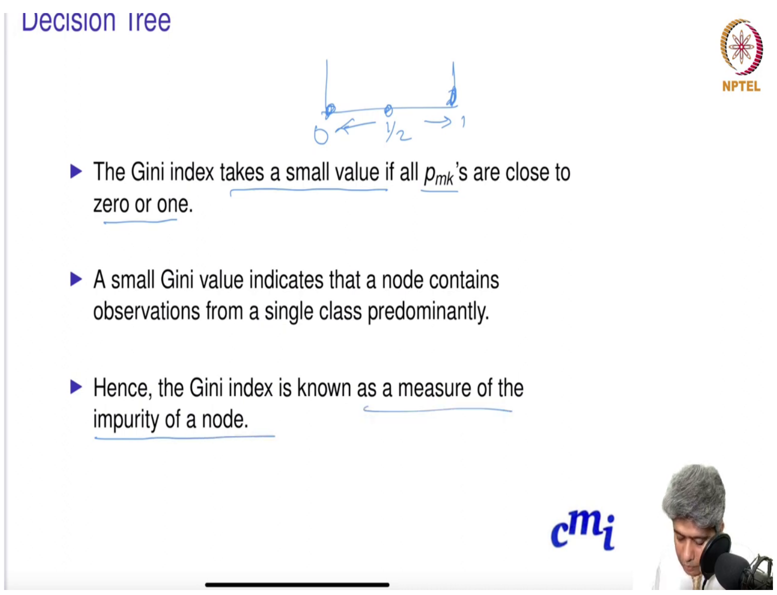
- However, the classification error is not sensitive to tree-growing.
- Hence the Gini index is an alternative to classification error, defined as

$$\boxed{G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})}$$

a measure of total variance across the K classes.

So, however, the classification error is the is not sensitive to tree growing. Hence Gini index is an alternative classification error ok Gini index is an alternative classification error and this is the measure pmk into 1 minus pmk and sum over k a measure of total variance across the k classes. How much variability you see across classes? Ok.

(Refer Slide Time: 14:48)

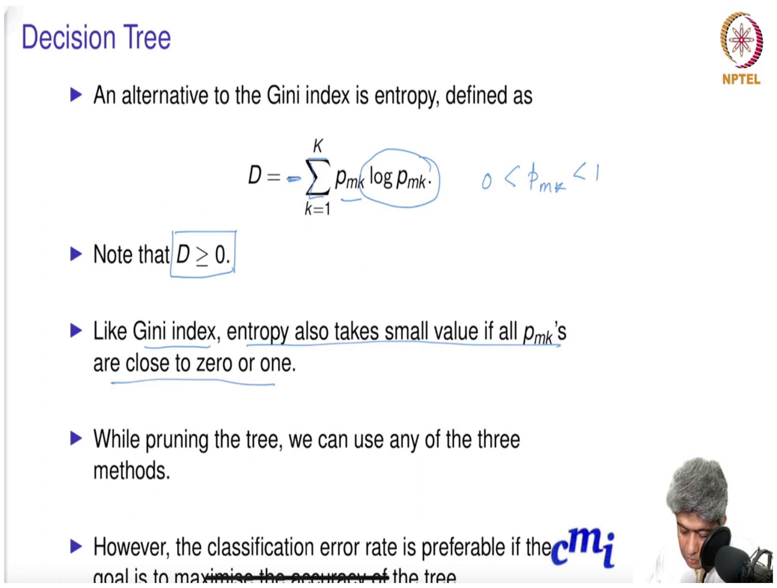


Decision Tree

- The Gini index takes a small value if all p_{mk}'s are close to zero or one.
- A small Gini value indicates that a node contains observations from a single class predominantly.
- Hence, the Gini index is known as a measure of the impurity of a node.

So, Gini index takes small value if all p k's are close to 0 or 1. So, if all the values are close to zero or close to one, then Gini index will be very small and if they are close to half then Gini index will be very high. So, you would like to all the values to be either close to 0 or 1. So, Gini you will try to reduce the Gini index. Gini index is known as the measure of impurity of a node.

(Refer Slide Time: 15:22)



Decision Tree

- An alternative to the Gini index is entropy, defined as

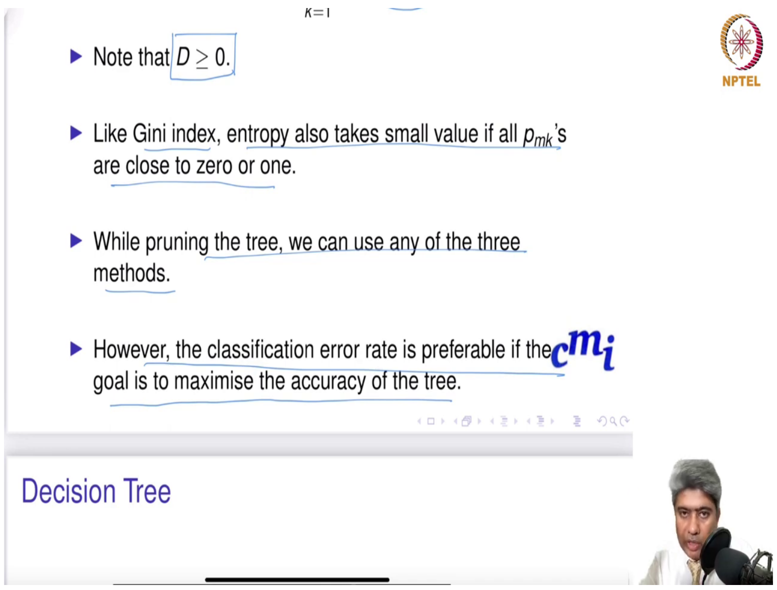
$$D = -\sum_{k=1}^{K} p_{mk} \log p_{mk}. \qquad 0 < p_{mk} < 1$$

- Note that $\boxed{D \geq 0.}$
- Like Gini index, entropy also takes small value if all p_{mk}'s are close to zero or one.
- While pruning the tree, we can use any of the three methods.
- However, the classification error rate is preferable if the goal is to ~~maximise the accuracy of~~ the tree

So, an alternative to the Gini index is the entropy and which is basically p pmk times log of pmk take the summation and take the minus ok. So, and d has to be d will be automatically greater than equal to 0 and its kind of obvious because pmk will be always between 0 and 1 typically for a classification problem and that will make the log pmk negative and pmk is a proportion.

So, negative positive so, the whole thing always D greater than equal to 0. Like Gini index entropy also takes small values if pmk's are close to 0 or 1.

(Refer Slide Time: 16:09)

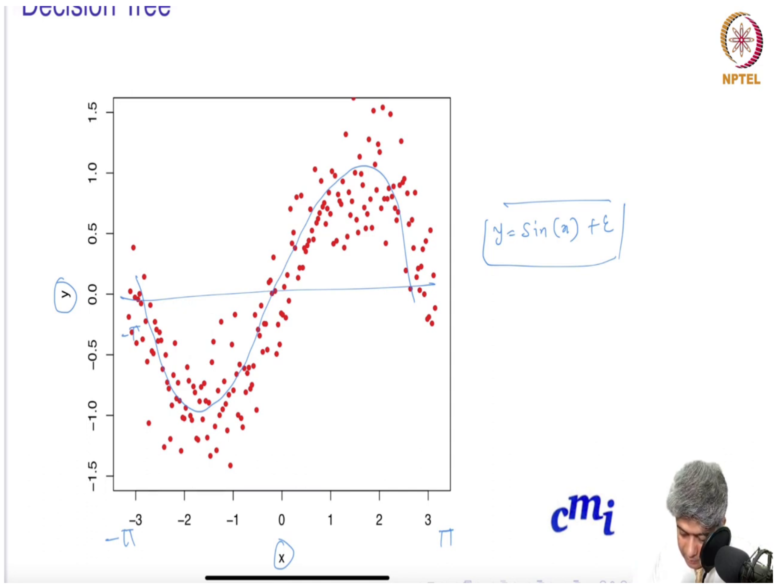


- Note that $\boxed{D \geq 0}$.
- Like Gini index, entropy also takes small value if all p_{mk}'s are close to zero or one.
- While pruning the tree, we can use any of the three methods.
- However, the classification error rate is preferable if the goal is to maximise the accuracy of the tree.

Decision Tree

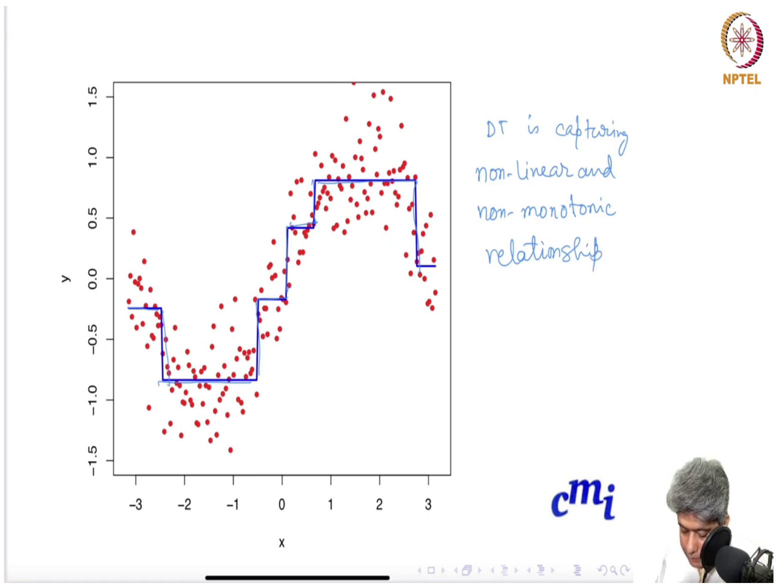
So, while pruning the tree we can use any of the three methods. However, classification error rate is preferable if goal is to maximize the accuracy for the tree ok.

(Refer Slide Time: 16:22)



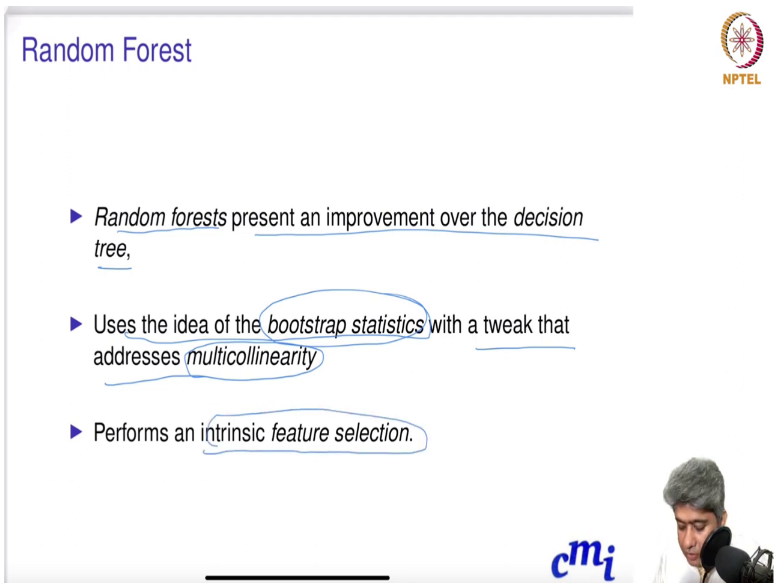
So, suppose here I have taken a example ok. So, here is the x and here is the y I have taken y equal to sort of sin x plus epsilon this is the model I have considered. So, you can see that you know between minus pi here it is minus pi and pi its sort of kind of goes in and some with some epsilon ok.

(Refer Slide Time: 16:54)



Now, if you apply decision tree; decision tree is effectively nicely capturing the relationship non-linear non monotonic relationship. So, decision tree is capturing non-linear and non-monotonic monotonic relationship ok.

(Refer Slide Time: 17:37)

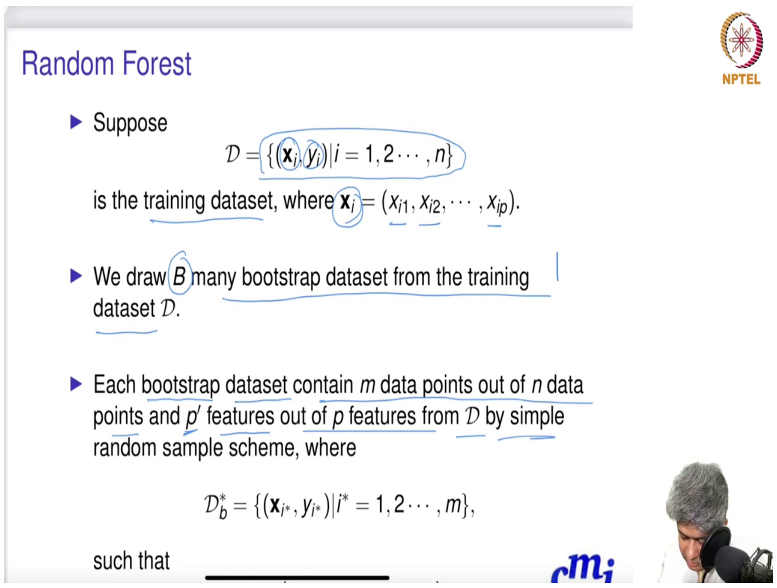

Random Forest

- *Random forests* present an improvement over the *decision tree*,
- Uses the idea of the *bootstrap statistics* with a tweak that addresses *multicollinearity*
- Performs an intrinsic *feature selection.*

So, the random forest presents an improvement over the decision tree. So, it uses the big uses the idea of bootstrap it uses the idea of bootstrap statistics. Remember we discussed bootstrap statistics. So, random forest uses the bootstrap statistics with the tweak that addresses the multicollinearity it is a beautiful algorithm. I think I have seen many algorithm, but random forest is one of the algorithm which is very close to my heart and is beautiful algorithm.

Let us see what it is doing ok. So, and it also so, first of all the way the algorithm is structured it automatically take care of the multicollinearity issue also it performs an intrinsic feature selection. Automatically it does a feature selection. So, beautiful algorithm let us see what it does.

(Refer Slide Time: 18:39)

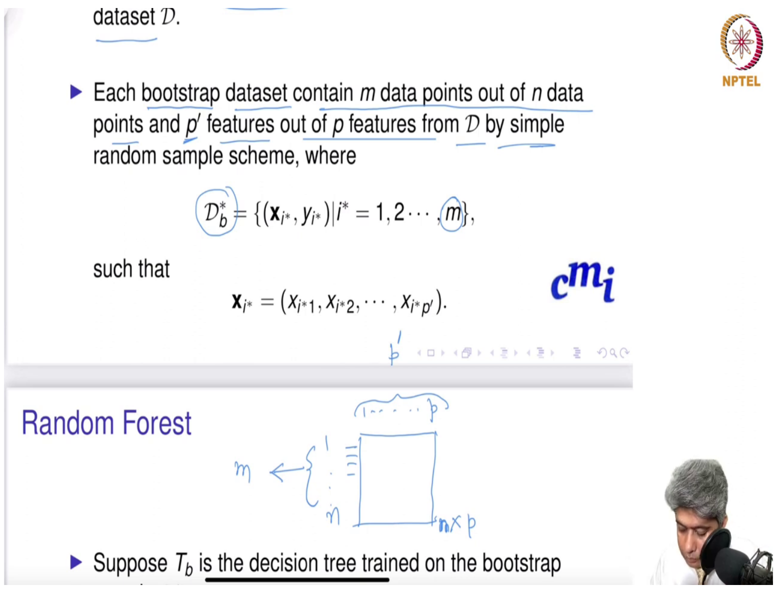


Random Forest

- Suppose
$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, 2 \cdots, n\}$$
is the training dataset, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$.
- We draw B many bootstrap dataset from the training dataset $\mathcal{D}$.
- Each bootstrap dataset contain m data points out of n data points and p' features out of p features from $\mathcal{D}$ by simple random sample scheme, where
$$\mathcal{D}_b^* = \{(\mathbf{x}_{i^*}, y_{i^*}) | i^* = 1, 2 \cdots, m\},$$
such that

So, suppose D is this is my data set some bunch of x i's and some y i is the trailing data set where x i is x i1, x i2, x ip. We draw capital B many bootstrap data set as training data set D. Each bootstrap data set contain m data points out of n data points p dash features out of p features ok from the D of m samples. So, what it does? Let me just try to explain you what is it doing.

(Refer Slide Time: 00:00)


dataset $\mathcal{D}$.

- Each bootstrap dataset contain m data points out of n data points and p' features out of p features from $\mathcal{D}$ by simple random sample scheme, where

$$\mathcal{D}_b^* = \{(\mathbf{x}_{i^*}, y_{i^*}) | i^* = 1, 2 \cdots, m\},$$

such that

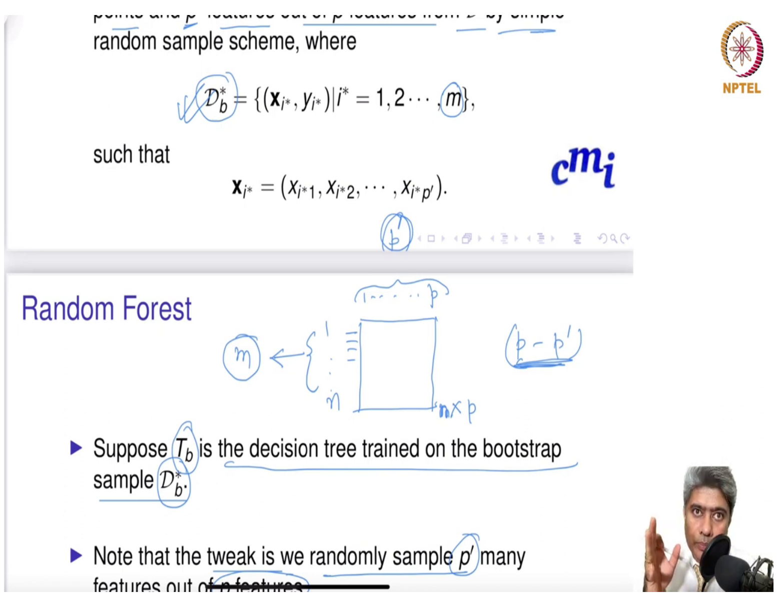
$$\mathbf{x}_{i^*} = (x_{i^*1}, x_{i^*2}, \cdots, x_{i^*p'}).$$

Random Forest

- Suppose T_b is the decision tree trained on the bootstrap

So, so first of all here you have a data set let me try to tell you ok. The data set has n samples and p features. So, 1, 2, 3, 4 up to n samples and p feature 1 to p ok n goes p column. So, first what it does? Each bootstrap data set. So, I am. So, the this is my bootstrap data, so it samples m sample out of n. So, from here it draws m samples, but at the same time its randomly not take all the features its randomly take m features p dash many features not p all p features it randomly takes p many feature p dash many features ok.

(Refer Slide Time: 20:32)


random sample scheme, where

$$\mathcal{D}_b^* = \{(\mathbf{x}_{i^*}, y_{i^*}) | i^* = 1, 2 \cdots, m\},$$

such that

$$\mathbf{x}_{i^*} = (x_{i^*1}, x_{i^*2}, \cdots, x_{i^*p'}).$$

Random Forest

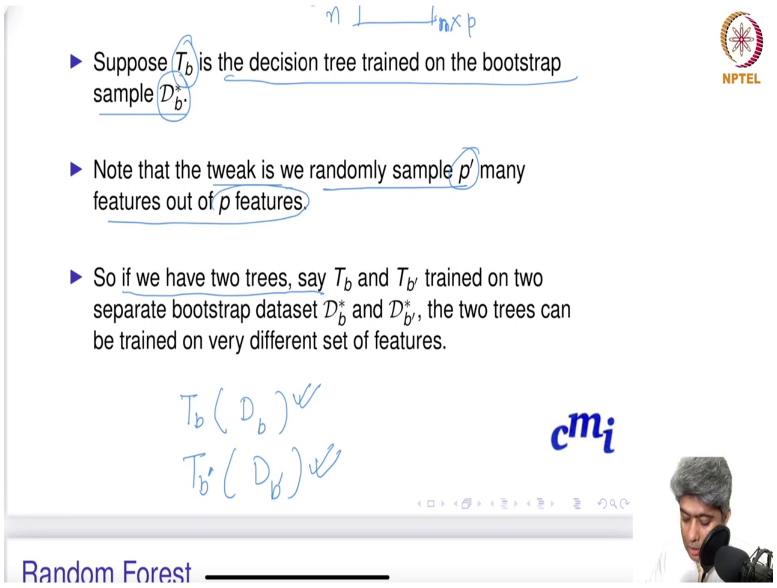
- Suppose T_b is the decision tree trained on the bootstrap sample $\mathcal{D}_b^*$.
- Note that the tweak is we randomly sample p' many features out of p features

And so suppose T b is the decision tree trained on the bootstrap sample D b star ok. Now, note that the tweak is we randomly sample p dash many features out of p features. So, these p dash out of p many features we are randomly sampling. So, m is also randomly sampling and p dash is also. So, the number of rows I am sampling randomly number of columns also I am sampling randomly ok.

And that is a beautiful trick that I found because when you are drawing p many phi, p dash many features; that means, you are dropping p minus p dash many features from the data set or from an any model that will be developed on using this data set. In though that in this data set p minus p dash feature will not be present p minus p dash many feature will not be present that is an interesting feature that I found.

And as a result what will happen? So, that means, you are dropping those data sets or sorry those features though you are dropping those features and you are building a sub model a nested model. So, this is a interesting feature.

(Refer Slide Time: 22:06)



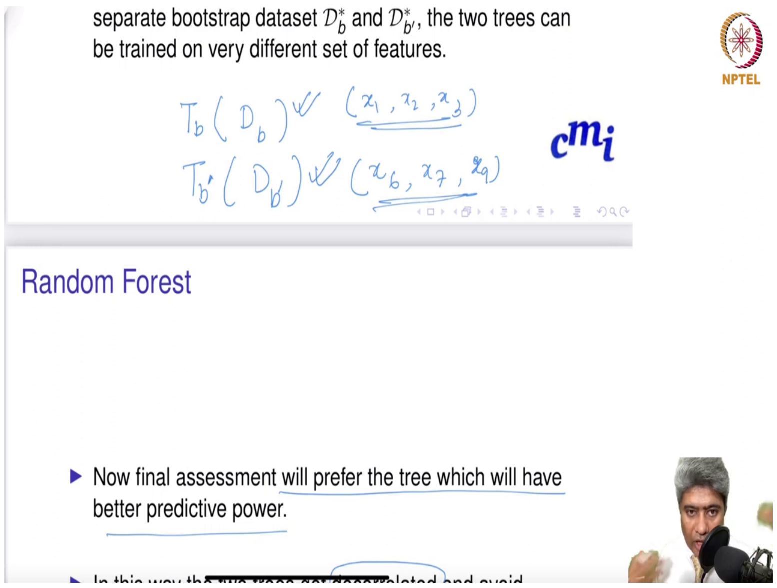
- Suppose T_b is the decision tree trained on the bootstrap sample $\mathcal{D}_b^*$.
- Note that the tweak is we randomly sample p' many features out of p features.
- So if we have two trees, say T_b and $T_{b'}$ trained on two separate bootstrap dataset $\mathcal{D}_b^*$ and $\mathcal{D}_{b'}^*$, the two trees can be trained on very different set of features.

$T_b(D_b)$ ✓✓

$T_{b'}(D_{b'})$ ✓✓

So, the that is the tweak that they are doing. So, if we have two trees one is based on T b another is T b dash ok. So, b is and that is based on data set D b and this is based on D b dash 2 T be trained on a very. So, this tree can be based trained on a very different data set a very different set of features. It could be trained on say feature number 1, 2, 3 and this would be trained on completely different feature number 9, 10, 11. So, they could be very different model.

(Refer Slide Time: 22:53)



separate bootstrap dataset $\mathcal{D}_b^*$ and $\mathcal{D}_{b'}^*$, the two trees can be trained on very different set of features.

$T_b(D_b)$ ✓✓ (x_1, x_2, x_3)

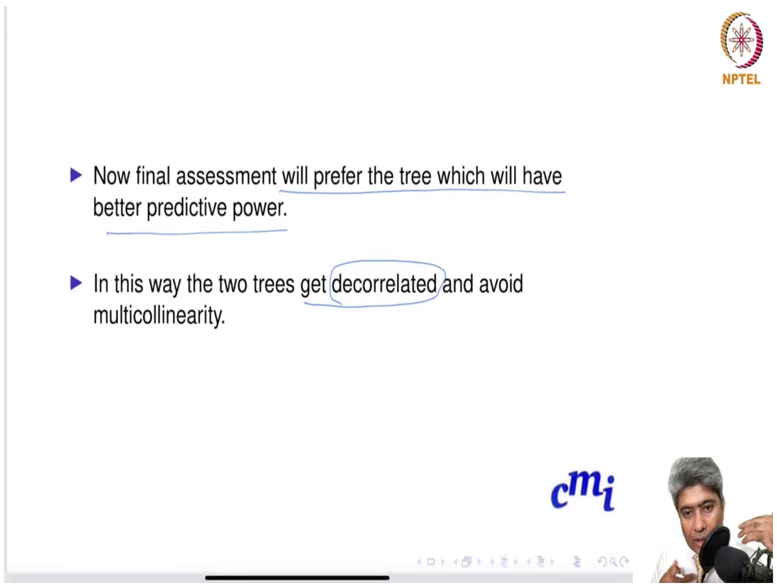
$T_{b'}(D_{b'})$ ✓✓ (x_6, x_7, x_9)

Random Forest

- Now final assessment will prefer the tree which will have better predictive power.

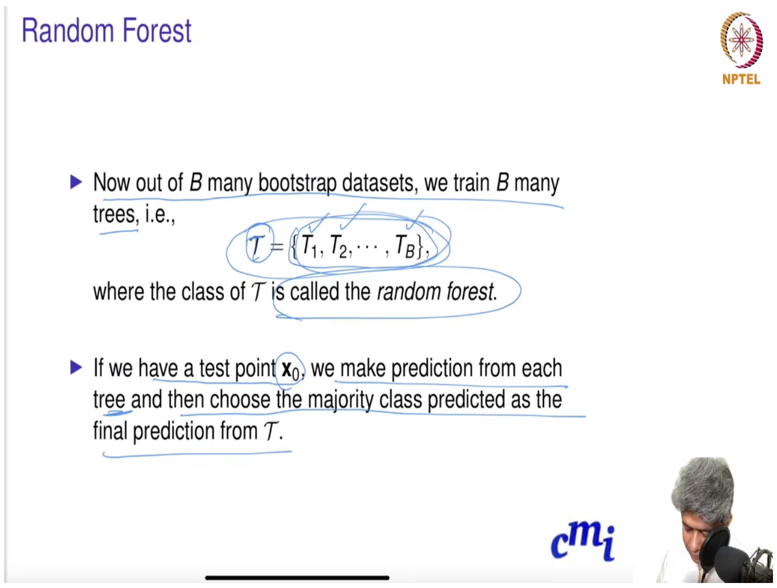
And then from these models you are taking the votes and you are taking the predictions. In the final assessment you will prefer the tree which will have a better predictive power. In this way you the two trees get decorrelated because here you may be you are taking the feature x 1, x 2 and x 3 whereas, in here you are building the model 3 based on x 6, x 7, x 9 I am just throwing a number. So, but we can see these set of features and these set of features are completely different.

(Refer Slide Time: 23:36)


- Now final assessment will prefer the tree which will have better predictive power.
- In this way the two trees get decorrelated and avoid multicollinearity.

So, as a result final prediction in the final thing the features get decorrelated and avoid the multicollinearity. So, this is a interesting feature I found.

(Refer Slide Time: 23:49)



Random Forest

- Now out of B many bootstrap datasets, we train B many trees, i.e.,

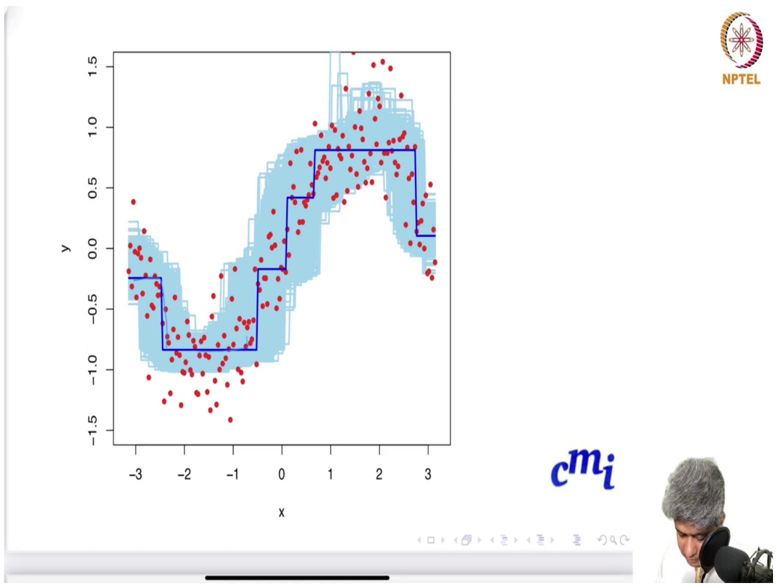
$$\mathcal{T} = \{T_1, T_2, \cdots, T_B\},$$

where the class of $\mathcal{T}$ is called the *random forest.*

- If we have a test point $\mathbf{x}_0$, we make prediction from each tree and then choose the majority class predicted as the final prediction from $\mathcal{T}$.

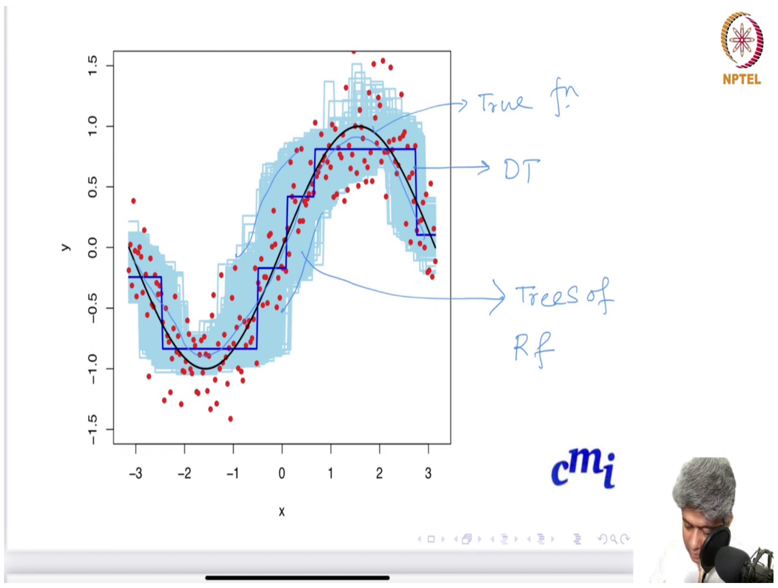
So, now out of B many bootstrap data set we train B many trees ok. So, now I have T 1 tree, T 2 tree. So, T B many trees where the class T this T is called random forest because this T is now class of all these trees. So, now the class T is called random forest. So, if we have test point x naught we make prediction from each tree and then choose the majority class predicted as the final prediction.

(Refer Slide Time: 24:27)



So, here is a example of random forest and here is a example of decision tree kind of plotted over there.

(Refer Slide Time: 24:42)



And then this is the true effectively true function the black one is the true function this is the true function. This blue line is the decision tree and all these sky blue sky blues are trees from random trees of random forest ok. So, thank you very much, I hope you enjoyed this video and in the next video we will do the hands on.

Thank you see, you in the next video.