Machine Learning for Soil and Crop Management
Professor Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology Kharagpur
Lecture – 58
Digital Soil Mapping with Categorical Variables(contd.)

Welcome friends to this third lecture of week 12. And in this week, we are talking about digital soil mapping with categorical variables, this is lecture number 58. And in this lecture we are going to talk about how to execute this C 5 regression model as well as then another model that is random forest model for categorical prediction in the digital soil mapping.

In our previous lecture we have seen how to use how to develop this confusion matrix and how to calculate the user's accuracy producer's accuracy and how to get the how to use the goofcat function to get all these values and also the overall accuracy. And also we have seen how to deal with the multinomial logistic regression model, how to model the Terrons or soil classes using the multinomial logistic regression model.

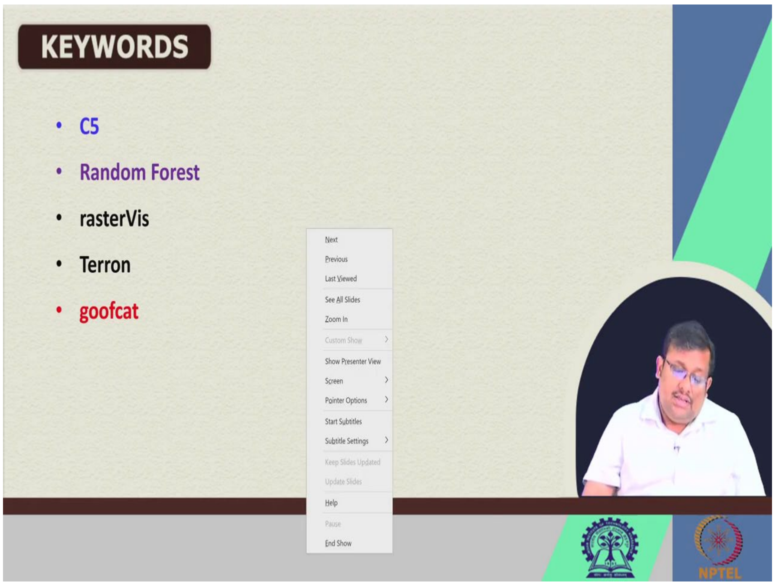
(Refer Slide Time: 1:28)

CONCEPTS COVERED

- C5 decision tree
- Random Forest classification model

In this lecture, we are going to focus on this following concept. So, we are going to use this C5 five decision tree and then random forest classification model.

(Refer Slide Time: 1:39)

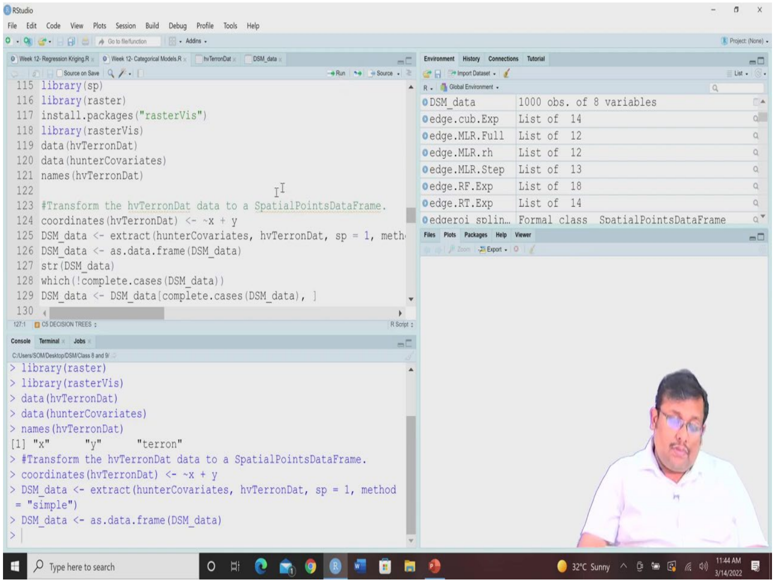


KEYWORDS

- C5
- Random Forest
- rasterVis
- Terron
- goofcat

Now, these are the keywords which are going to encounter in this lecture, C5, then random forest, rasterVis, Terron and goofcat. So, the C5 is a decision tree almost works like cubist model and let us see how it operates using how we can execute this C5 model in R remember the C50 package is there which we are need to install for this C5 Model C5 decision trees.

(Refer Slide Time: 2:22)

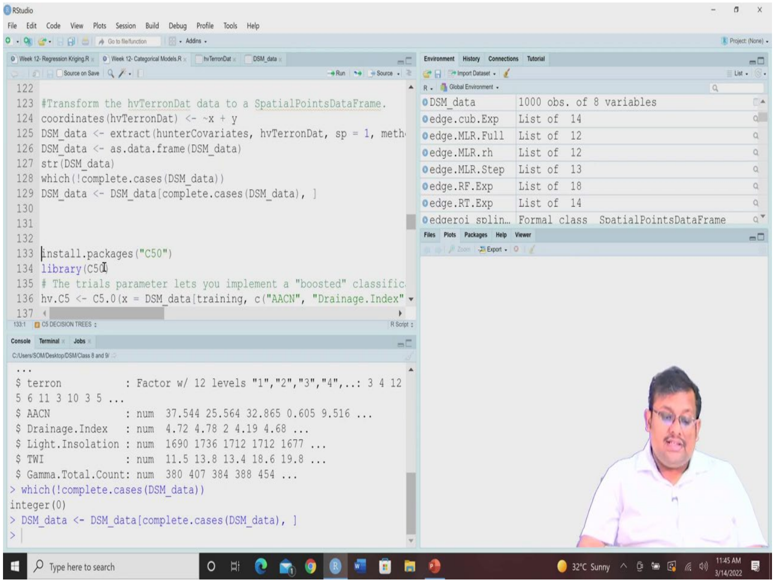


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Week 12- Regression Kriging.R
Week 12- Categorical Models.R
hvTerronDat
DSM_data
Source on Save
Run
Source
127:1
C5 DECISION TREES
R Script
Console Terminal Jobs
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
Project: (None)
Environment History Connections Tutorial
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
Type here to search
32°C Sunny
11:44 AM
3/14/2022

```
library(sp)
library(raster)
install.packages("rasterVis")
library(rasterVis)
data(hvTerronDat)
data(hunterCovariates)
names(hvTerronDat)

#Transform the hvTerronDat data to a SpatialPointsDataFrame.
coordinates(hvTerronDat) <- ~x + y
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, meth
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
which(!complete.cases(DSM_data))
DSM_data <- DSM_data[complete.cases(DSM_data), ]
```

```
> library(raster)
> library(rasterVis)
> data(hvTerronDat)
> data(hunterCovariates)
> names(hvTerronDat)
[1] "x"      "y"      "terron"
> #Transform the hvTerronDat data to a SpatialPointsDataFrame.
> coordinates(hvTerronDat) <- ~x + y
> DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, method
 = "simple")
> DSM_data <- as.data.frame(DSM_data)
> 
```

```
DSM_data          1000 obs. of 8 variables
edge.cub.Exp      List of  14
edge.MLR.Full     List of  12
edge.MLR.rh       List of  12
edge.MLR.Step     List of  13
edge.RF.Exp       List of  18
edge.RT.Exp       List of  14
edgeroi_splin…    Formal class  SpatialPointsDataFrame
```

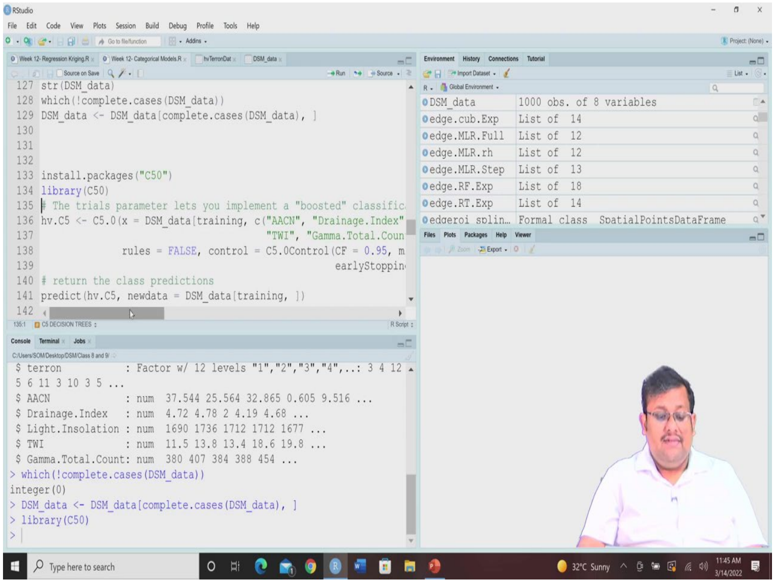


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Week 12- Regression Kriging.R
Week 12- Categorical Models.R
hvTerronDat
DSM_data
Environment History Connections Tutorial
Global Environment
Files Plots Packages Help Viewer
C5 DECISION TREES
R Script
Console Terminal Jobs
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
Type here to search
32°C Sunny
11:45 AM
3/14/2022

```r

#Transform the hvTerronDat data to a SpatialPointsDataFrame.
coordinates(hvTerronDat) <- ~x + y
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, meth
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
which(!complete.cases(DSM_data))
DSM_data <- DSM_data[complete.cases(DSM_data), ]



install.packages("C50")
library(C50)
# The trials parameter lets you implement a "boosted" classific
hv.C5 <- C5.0(x = DSM_data[training, c("AACN", "Drainage.Index"

```

```
 ...
 $ terron           : Factor w/ 12 levels "1","2","3","4",..: 3 4 12
 5 6 11 3 10 3 5 ...
 $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
 $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
 $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
 $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
 $ Gamma.Total.Count: num  380 407 384 388 454 ...
> which(!complete.cases(DSM_data))
integer(0)
> DSM_data <- DSM_data[complete.cases(DSM_data), ]
> 
```

```
DSM_data       1000 obs. of 8 variables
edge.cub.Exp   List of  14
edge.MLR.Full  List of  12
edge.MLR.rh    List of  12
edge.MLR.Step  List of  13
edge.RF.Exp    List of  18
edge.RT.Exp    List of  14
edgeroi_splin… Formal class  SpatialPointsDataFrame
```

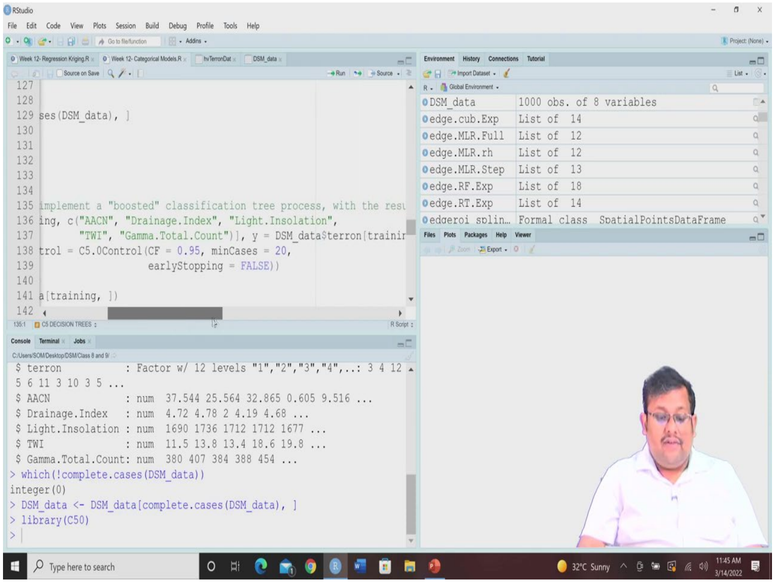


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Week 12- Regression Kriging.R
Week 12- Categorical Models.R
hvTerronDat
DSM_data
Environment History Connections Tutorial
Global Environment
Files Plots Packages Help Viewer
C5 DECISION TREES
R Script
Console Terminal Jobs
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
Type here to search
32°C Sunny
11:45 AM
3/14/2022

```r
str(DSM_data)
which(!complete.cases(DSM_data))
DSM_data <- DSM_data[complete.cases(DSM_data), ]



install.packages("C50")
library(C50)
# The trials parameter lets you implement a "boosted" classific
hv.C5 <- C5.0(x = DSM_data[training, c("AACN", "Drainage.Index"
                                       "TWI", "Gamma.Total.Coun
                 rules = FALSE, control = C5.0Control(CF = 0.95, m
                                                      earlyStoppin
# return the class predictions
predict(hv.C5, newdata = DSM_data[training, ])

```

```
 $ terron           : Factor w/ 12 levels "1","2","3","4",..: 3 4 12
 5 6 11 3 10 3 5 ...
 $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
 $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
 $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
 $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
 $ Gamma.Total.Count: num  380 407 384 388 454 ...
> which(!complete.cases(DSM_data))
integer(0)
> DSM_data <- DSM_data[complete.cases(DSM_data), ]
> library(C50)
> 
```

```
DSM_data       1000 obs. of 8 variables
edge.cub.Exp   List of  14
edge.MLR.Full  List of  12
edge.MLR.rh    List of  12
edge.MLR.Step  List of  13
edge.RF.Exp    List of  18
edge.RT.Exp    List of  14
edgeroi_splin… Formal class  SpatialPointsDataFrame
```

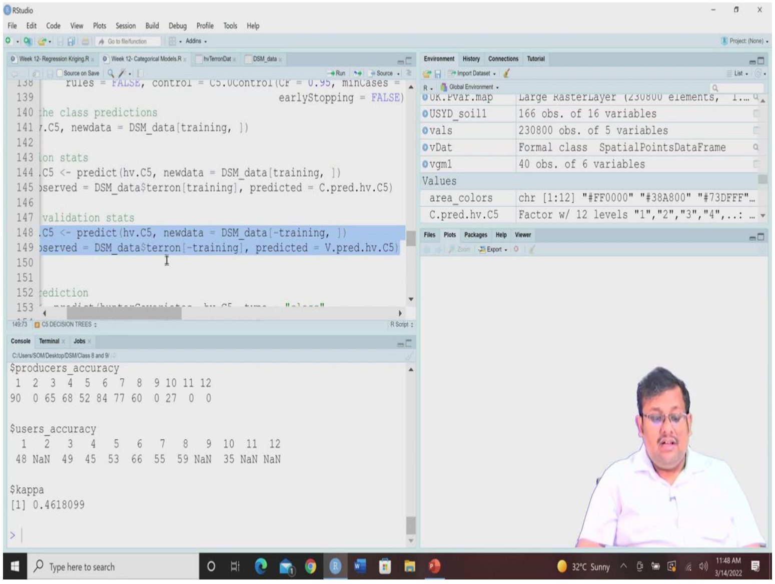


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Week 12- Regression Kriging.R
Week 12- Categorical Models.R
hvTerronDat
DSM_data
Environment History Connections Tutorial
Global Environment
Files Plots Packages Help Viewer
C5 DECISION TREES
R Script
Console Terminal Jobs
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
Type here to search
32°C Sunny
11:45 AM
3/14/2022

```r


ses(DSM_data), ]





implement a "boosted" classification tree process, with the resu
ing, c("AACN", "Drainage.Index", "Light.Insolation",
      "TWI", "Gamma.Total.Count")], y = DSM_data$terron[trainir
trol = C5.0Control(CF = 0.95, minCases = 20,
                   earlyStopping = FALSE))

a[training, ])

```

```
 $ terron           : Factor w/ 12 levels "1","2","3","4",..: 3 4 12
 5 6 11 3 10 3 5 ...
 $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
 $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
 $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
 $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
 $ Gamma.Total.Count: num  380 407 384 388 454 ...
> which(!complete.cases(DSM_data))
integer(0)
> DSM_data <- DSM_data[complete.cases(DSM_data), ]
> library(C50)
> 
```

```
DSM_data       1000 obs. of 8 variables
edge.cub.Exp   List of  14
edge.MLR.Full  List of  12
edge.MLR.rh    List of  12
edge.MLR.Step  List of  13
edge.RF.Exp    List of  18
edge.RT.Exp    List of  14
edgeroi_splin… Formal class  SpatialPointsDataFrame
```



```
    rules = FALSE, control = C5.0Control(CF = 0.95, minCases =
                                         earlyStopping = FALSE)
he class predictions
.C5, newdata = DSM_data[training, ])

on stats
.C5 <- predict(hv.C5, newdata = DSM_data[training, ])
served = DSM_data$terron[training], predicted = C.pred.hv.C5)

validation stats
.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
served = DSM_data$terron[-training], predicted = V.pred.hv.C5)


ediction
```

```
$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12 
90  0 65 68 52 84 77 60  0 27  0  0 

$users_accuracy
  1   2   3   4   5   6   7   8   9  10  11  12 
 48 NaN  49  45  53  66  55  59 NaN  35 NaN NaN 

$kappa
[1] 0.4618099

> 
```

```
UK.Pvar.map     Large RasterLayer (230800 elements, 1...
USYD_soil1      166 obs. of 16 variables
vals            230800 obs. of 5 variables
vDat            Formal class  SpatialPointsDataFrame
vgm1            40 obs. of 6 variables
Values
 area_colors    chr [1:12] "#FF0000" "#38A800" "#73DFFF"...
 C.pred.hv.C5   Factor w/ 12 levels "1","2","3","4",..: ...
```

So, let us run, let us call this C5, let us call this library ithir first and then we are calling the library sp then library rester we have already installed this rasterVis. So, I am going to just call this library rasterVis then just like previously we are going to call this hvTerronDat file.

And then just like previously, we are going to call the hunterCovariates and let us see the names of the hunter hvTerron Data. So, of course like previously we know there are three category, three fields one is x, then y and then Terron x is x and y are the coordinates and Terron is the soil landscape class based on certain characteristics.

So, here there are 12 classes named as 1 to 12 and we are going to predict their occurrence based on this C5 decision tree. Now, just like previously, we have to do the, we have to go through the same process right now it is in tabular format and we have to convert it to special points data frame.

So, we are going to use this coordinate function to specify this coordinate x y and then we are going to extract these hunterCovariates using the simple method. So, once we extract that, now, after it is extraction, we are going to again revert is back using the as dot data dot frame function.

So, initially it was in hvTerronDat that was in tabular format data frame format simple data frame format then for the sake of extraction for this have an intersection and extraction of the covariates with this hvTerronDat we have converted that to special points data frame after this extraction and combination of the covariates with these hvTerronDat now, we have again revert it back to the simple data frame. Because the simple data frame will allow for the, for model to run.

Now, we are going to see the structure of the data set. So, you can see here now, we are having 1000 observation of 8 variables x y Terron which is a factor of 12 levels from 1 to 12 then altitude of channel network then drainage index then light insulation twi and also the gamma total count.

So, from there this is the complete dataset from there also we want to remove the incomplete cases and keep only those complete cases for further execution of the model. Now, we have to install this package C50 for running the C5 decision tree. So, I have already installed that so, please go ahead and install this and then we are going to call this library C50.

Now, here this is how we go with this C5 decision tree simple function is C5 here our x is our training samples and our predictors are AACN drainage index, light insulation, Twi and gamma total count here y is also the training samples and then here you can see that to do control there is a control called C5 is 0.0 control and here you can see here minimum cases 20 and here one argument is there that is trials equal to 1.

So, this trials parameter gives you lets you implement a boosted classic classification tree process with the results aggregated as the end so, again, it is a boosted classification, tree based approach and ultimately it is aggregated at the end. So, here we are giving 1. Now, once we let us run this whole comment now, once we run the whole comment then the let us see the predicted values using the training samples.

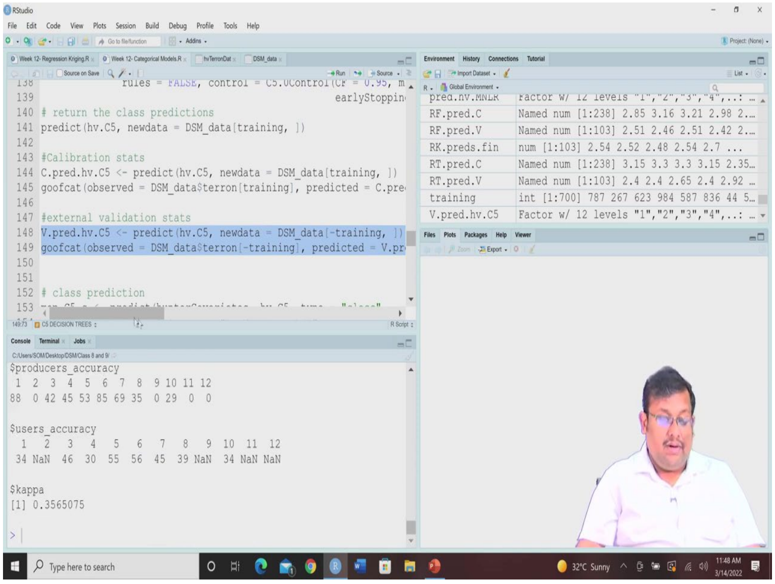
So, we are predicting the training samples. So, you can see how these 70 percent training samples will be predicted the first one will be predicted as 6 Terron the second one will be predicted as 8th Terron and the third one will be predicted as 4th Terron and so on. So, this is how this all the 70 percent of the data will be predicted using the C5 decision tree and once we have done that now the calibration steps.

So, here we are going to predict based on these training samples and finally, we are going to use this goofcat function. So, this is the calibration goofcat that is calibration goodness of fit. So, you can see here this is a confusion matrix for all the 12th Terrons overall accuracies 53 percent producer's accuracy you can see for all the Terron class user's accuracy and also the Kappa coefficient you can see. So, this is an internal validation or calibration statistics.

Now, how to do with the how to execute the validation strategies of course, minus training samples. And then again these goofcat function observed is our DSM data for this Terrons.

So, it is minus training and predicted values is v dot pred dot hv dot C5 which we have already created previously.

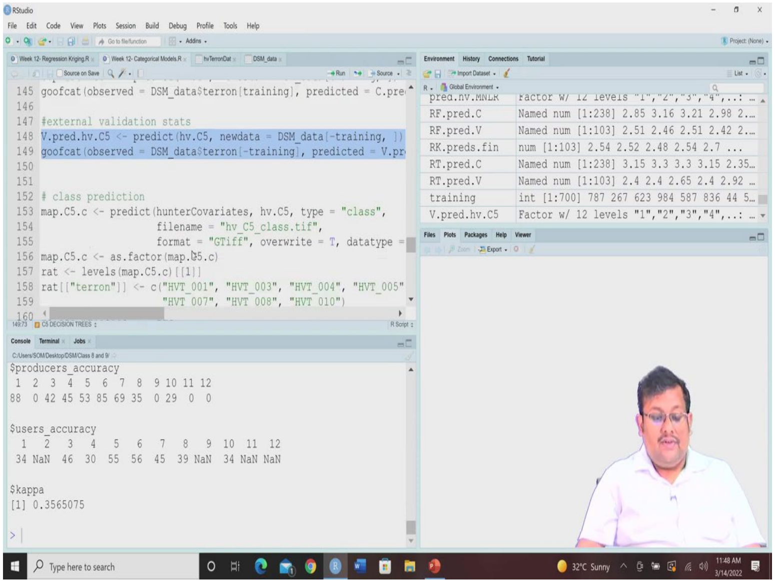
(Refer Slide Time: 9:30)



```
                    rules = FALSE, control = C5.0Control(CF = 0.95, m
                                                       earlyStoppin
# return the class predictions
predict(hv.C5, newdata = DSM_data[training, ])

#Calibration stats
C.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[training, ])
goofcat(observed = DSM_data$terron[training], predicted = C.pre

#external validation stats
V.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pr


# class prediction
```

```
$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
88  0 42 45 53 85 69 35  0 29  0  0

$users_accuracy
  1   2   3   4   5   6   7   8   9  10  11  12
 34 NaN  46  30  55  56  45  39 NaN  34 NaN NaN

$kappa
[1] 0.3565075
>
```

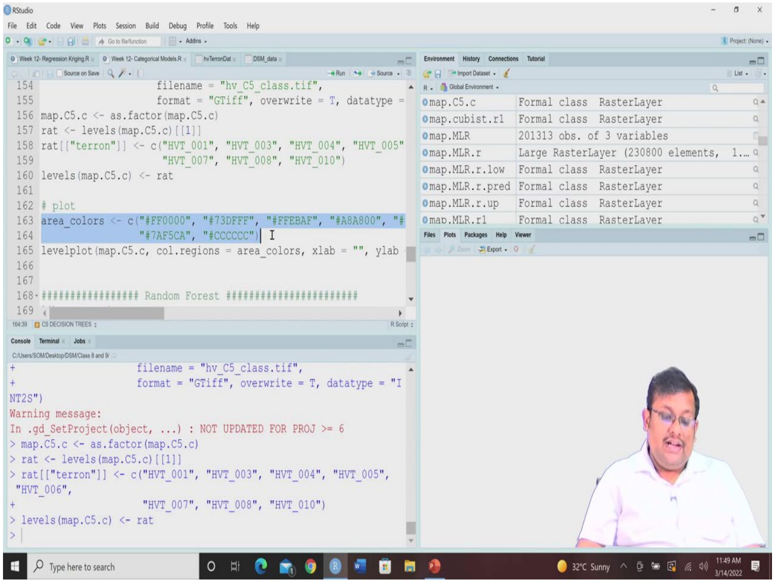
```
pred.hv.MNLR    Factor w/ 12 levels "1","2","3","4",..: …
RF.pred.C       Named num [1:238] 2.85 3.16 3.21 2.98 2.…
RF.pred.V       Named num [1:103] 2.51 2.46 2.51 2.42 2.…
RK.preds.fin    num [1:103] 2.54 2.52 2.48 2.54 2.7 ...
RT.pred.C       Named num [1:238] 3.15 3.3 3.3 3.15 2.35…
RT.pred.V       Named num [1:103] 2.4 2.4 2.65 2.4 2.92 …
training        int [1:700] 787 267 623 984 587 836 44 5…
V.pred.hv.C5    Factor w/ 12 levels "1","2","3","4",..: …
```



```
goofcat(observed = DSM_data$terron[training], predicted = C.pre

#external validation stats
V.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pr


# class prediction
map.C5.c <- predict(hunterCovariates, hv.C5, type = "class",
                    filename = "hv_C5_class.tif",
                    format = "GTiff", overwrite = T, datatype =
map.C5.c <- as.factor(map.C5.c)
rat <- levels(map.C5.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005"
                     "HVT_007", "HVT_008", "HVT_010")
```

```
$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
88  0 42 45 53 85 69 35  0 29  0  0

$users_accuracy
  1   2   3   4   5   6   7   8   9  10  11  12
 34 NaN  46  30  55  56  45  39 NaN  34 NaN NaN

$kappa
[1] 0.3565075
>
```

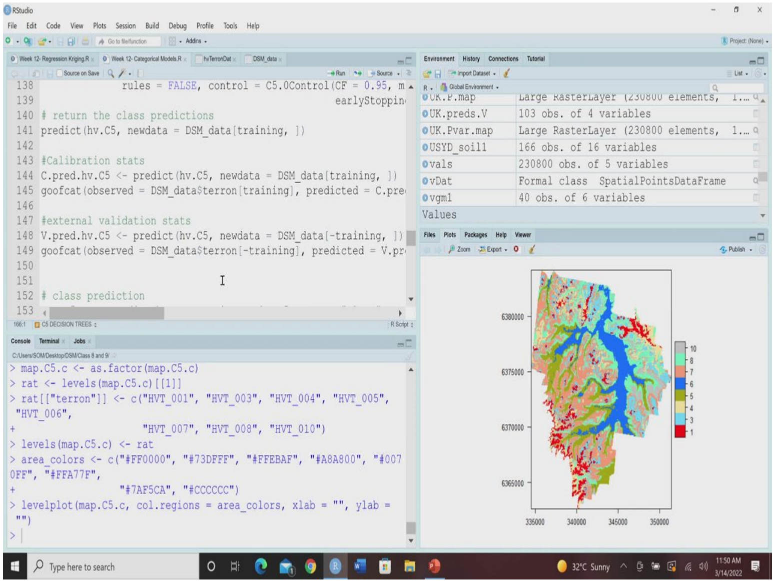
```
pred.hv.MNLR    Factor w/ 12 levels "1","2","3","4",..: …
RF.pred.C       Named num [1:238] 2.85 3.16 3.21 2.98 2.…
RF.pred.V       Named num [1:103] 2.51 2.46 2.51 2.42 2.…
RK.preds.fin    num [1:103] 2.54 2.52 2.48 2.54 2.7 ...
RT.pred.C       Named num [1:238] 3.15 3.3 3.3 3.15 2.35…
RT.pred.V       Named num [1:103] 2.4 2.4 2.65 2.4 2.92 …
training        int [1:700] 787 267 623 984 587 836 44 5…
V.pred.hv.C5    Factor w/ 12 levels "1","2","3","4",..: …
```



```r
                        filename = "hv_C5_class.tif",
                        format = "GTiff", overwrite = T, datatype =
map.C5.c <- as.factor(map.C5.c)
rat <- levels(map.C5.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005"
                         "HVT_007", "HVT_008", "HVT_010")
levels(map.C5.c) <- rat

# plot
area_colors <- c("#FF0000", "#73DFFF", "#FFEBAF", "#A8A800", "#
                    "#7AF5CA", "#CCCCCC")
levelplot(map.C5.c, col.regions = area_colors, xlab = "", ylab


################# Random Forest ######################
```

```
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
+                        filename = "hv_C5_class.tif",
+                        format = "GTiff", overwrite = T, datatype = "I
NT2S")
Warning message:
In .gd_SetProject(object, ...) : NOT UPDATED FOR PROJ >= 6
> map.C5.c <- as.factor(map.C5.c)
> rat <- levels(map.C5.c)[[1]]
> rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005",
 "HVT_006",
+                        "HVT_007", "HVT_008", "HVT_010")
> levels(map.C5.c) <- rat
> 
```

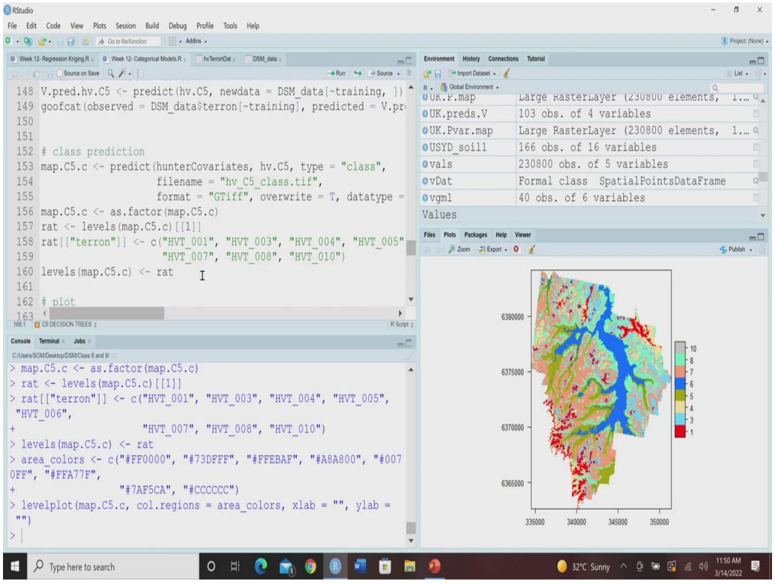
map.C5.c	Formal class RasterLayer
map.cubist.r1	Formal class RasterLayer
map.MLR	201313 obs. of 3 variables
map.MLR.r	Large RasterLayer (230800 elements, 1.…
map.MLR.r.low	Formal class RasterLayer
map.MLR.r.pred	Formal class RasterLayer
map.MLR.r.up	Formal class RasterLayer
map.MLR.r1	Formal class RasterLayer



```r
                   rules = FALSE, control = C5.0Control(CF = 0.95, m
                                                   earlyStoppin
# return the class predictions
predict(hv.C5, newdata = DSM_data[training, ])

#Calibration stats
C.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[training, ])
goofcat(observed = DSM_data$terron[training], predicted = C.pre

#external validation stats
V.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pr


# class prediction
```

```
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
> map.C5.c <- as.factor(map.C5.c)
> rat <- levels(map.C5.c)[[1]]
> rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005",
 "HVT_006",
+                        "HVT_007", "HVT_008", "HVT_010")
> levels(map.C5.c) <- rat
> area_colors <- c("#FF0000", "#73DFFF", "#FFEBAF", "#A8A800", "#007
0FF", "#FFA77F",
+                    "#7AF5CA", "#CCCCCC")
> levelplot(map.C5.c, col.regions = area_colors, xlab = "", ylab =
 "")
> 
```

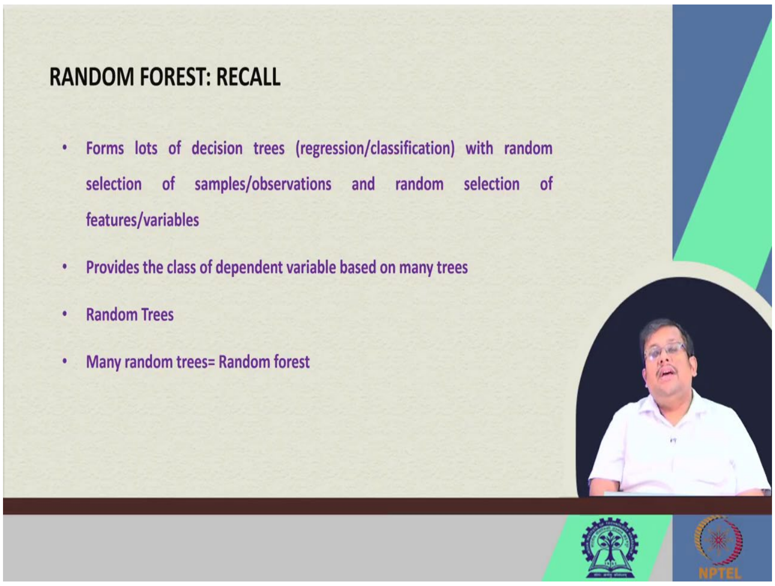
UK.P.map	Large RasterLayer (230800 elements, 1.…
UK.preds.V	103 obs. of 4 variables
UK.Pvar.map	Large RasterLayer (230800 elements, 1.…
USYD_soil1	166 obs. of 16 variables
vals	230800 obs. of 5 variables
vDat	Formal class SpatialPointsDataFrame
vgm1	40 obs. of 6 variables
Values	



```r
V.pred.hv.C5 <- predict(hv.C5, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pr


# class prediction
map.C5.c <- predict(hunterCovariates, hv.C5, type = "class",
                        filename = "hv_C5_class.tif",
                        format = "GTiff", overwrite = T, datatype =
map.C5.c <- as.factor(map.C5.c)
rat <- levels(map.C5.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005"
                         "HVT_007", "HVT_008", "HVT_010")
levels(map.C5.c) <- rat

# plot
```

```
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
> map.C5.c <- as.factor(map.C5.c)
> rat <- levels(map.C5.c)[[1]]
> rat[["terron"]] <- c("HVT_001", "HVT_003", "HVT_004", "HVT_005",
 "HVT_006",
+                        "HVT_007", "HVT_008", "HVT_010")
> levels(map.C5.c) <- rat
> area_colors <- c("#FF0000", "#73DFFF", "#FFEBAF", "#A8A800", "#007
0FF", "#FFA77F",
+                    "#7AF5CA", "#CCCCCC")
> levelplot(map.C5.c, col.regions = area_colors, xlab = "", ylab =
 "")
> 
```

UK.P.map	Large RasterLayer (230800 elements, 1.…
UK.preds.V	103 obs. of 4 variables
UK.Pvar.map	Large RasterLayer (230800 elements, 1.…
USYD_soil1	166 obs. of 16 variables
vals	230800 obs. of 5 variables
vDat	Formal class SpatialPointsDataFrame
vgm1	40 obs. of 6 variables
Values	



RANDOM FOREST: RECALL

- Forms lots of decision trees (regression/classification) with random selection of samples/observations and random selection of features/variables
- Provides the class of dependent variable based on many trees
- Random Trees
- Many random trees= Random forest

So, let us run it and you can see here again the confusion matrix for the validation samples overall accuracy producer's accuracy user's accuracy and Kappa coefficient. Now, class prediction. So, for this class prediction, we are going to use this predict function and the type is class and let us first go for it.

And then once you do that, then again this raster attribute table we can add these different names or the legions for these different hunter valley Terron classes. So, once we do that, then again we can add these color coding just like we have seen in case of C5 just like we have seen in case of multinomial logistic regression.

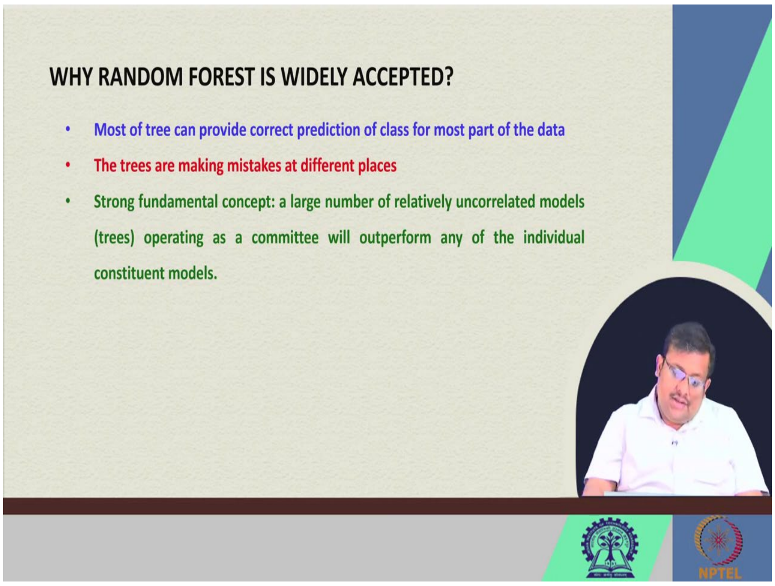
So, we can go ahead and give these hex color codes for this class Terron class and then we are going to use this Levelplot function just like previously to develop these, this plot of map so, you can see here some of the classes are not predicted like class 2 and then class 9 are not predicted. Here is only 1, 3, 4, 5, 6, 7, 8 and also class 11, 12 are not predicted. So, that shows that this is how we can go and produce the plots or produce the map using the C5 decision tree.

So, the logic of workflow, the logic of workflow is basically same as we have seen in case of multinomial logistic regression model also. Here just the individual model parameters and model functions are different, but the rest of the steps are almost same and ultimately showing the results. So, this is how guys we can get the map by using this C5 decision tree model. Okay guys.

So, you know that now we are going to discuss another classification algorithm that is random forest of course, random forests can be used for both regression and classification you know from your week 3 knowledge that random forest develops lots of decision trees either regression or classification with random selection of the samples or observation and random selection of features or variables.

And of course, it gives us the provides a class of dependent variable based on many trees and all these trees are develop randomly how we will see and then that is why these individual trees are known as random tree and when there are many random tree we call it random forest. So, we know that.

(Refer Slide Time: 12:50)



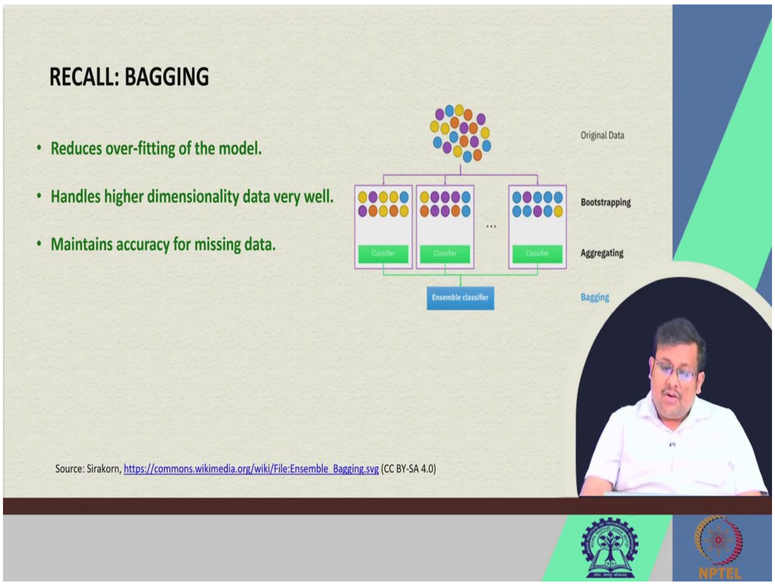
WHY RANDOM FOREST IS WIDELY ACCEPTED?

- Most of tree can provide correct prediction of class for most part of the data
- The trees are making mistakes at different places
- Strong fundamental concept: a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Now, we also know that why random forest is widely accepted? The random forest is widely accepted that most of the tree can provide correct prediction of class for most part of the data and the trees are making mistakes at different places and strong fundamental concept is there that is a large number of relatively uncorrelated models or trees operating as a committee will outperform any of the individual constituents model.

So, here we are not based our we are not basing our final decision based on a single learner we are basically for predicting using multiple learners and use it, and then and then we are making the decision based on the either averaging of the results or by taking the majority of the vote in case of classification, we are taking the majority of the vote and in case of regression, we take the average of their predictions. So, this is why this random forest is widely accepted.

(Refer Slide Time: 14:02)



RECALL: BAGGING

- Reduces over-fitting of the model.
- Handles higher dimensionality data very well.
- Maintains accuracy for missing data.

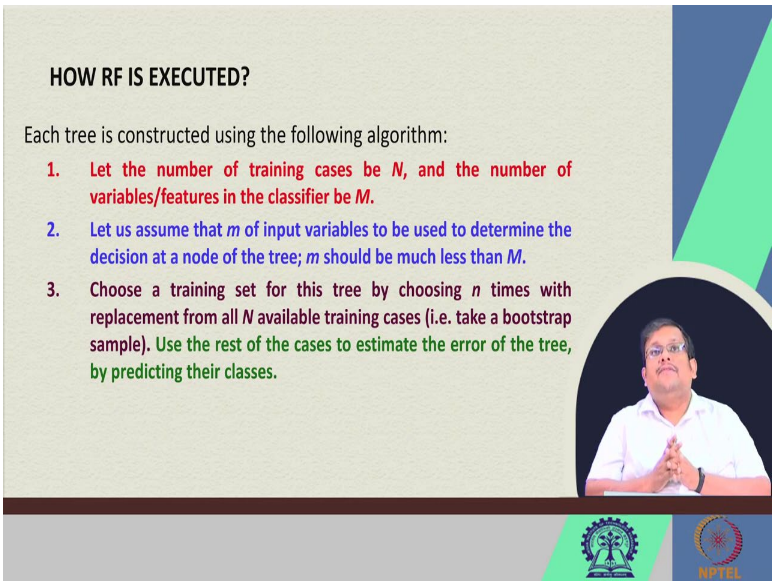
Source: Sirakorn, https://commons.wikimedia.org/wiki/File:Ensemble_Bagging.svg (CC BY-SA 4.0)

Now, you also know what is bagging? So, if there is a original data, and we take the samples you can see here from this data set if we take the bootstrap sample by without replacement sorry, if we bootstrapping is with replacement, I am sorry guys. So, using the original data if we take the sample with replacement, so, these will be bootstrap samples.

And when we develop the classifier individual classifier based on each bootstrap samples so, that is called aggregation. And finally, ultimately the ensemble classifier will based will be based on these individual classifiers. So, and then maximum voting so, that is why it is known as bootstrap aggregation or bagging.

So, why do we take the bagging? Because bagging is helpful for reducing the overfitting of the model and it handles higher dimensionally higher dimensional data very well. And also it maintains the accuracy for the missing data. So, that is why we go with the bagging.

(Refer Slide Time: 15:22)



HOW RF IS EXECUTED?

Each tree is constructed using the following algorithm:

1. Let the number of training cases be *N*, and the number of variables/features in the classifier be *M*.
2. Let us assume that *m* of input variables to be used to determine the decision at a node of the tree; *m* should be much less than *M*.
3. Choose a training set for this tree by choosing *n* times with replacement from all *N* available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.

So, let us see how these random forest is being made. So, this random forest in case of random forest the individual tree is being developed somewhat differently than that of the decision tree in case of decision tree we remember that all the samples and all the variables with all its values are considered and the algorithm tries to find the optimum variable and their value combination to get the splitting criteria in each of these node.

So, but in case of random forest, the idea is somewhat different. So, here let us assume that that there are capital N number of samples or observations and there are capital M number of features or variables. So, let us assume that the small m of input variables to be used to determine the decision at a node of this tree and m should be much less than n.

Now, let us assume that there are maybe 100 of features, but from these 100 of features we are not going to use all the 100 features, we are going to use a small subset which is denoted by small m. So, this small m is also known as m try in R hyper parameter and you can see that there are different ways to calculate these m try different ways to assume this m try value.

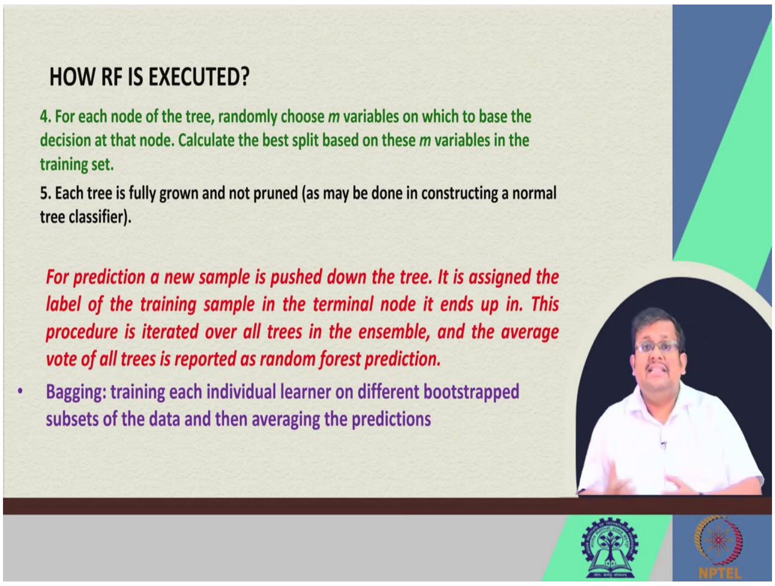
In case of regression generally capital M by 3 can be a value of m try. But in case of classification problem people take square root of capital M as an m try value. There are other instances also where people are taking other values of m try also. So, what we understand that there is total number of observation is capital N total number of features is capital M from there we are taking small m which is also known as m try for as a splitting criteria in each of these tree.

So, choose now let us choose a training set for this tree suppose we are growing the first tree. So, let us choose a training set for this tree by choosing n times with replacement from all the n available training cases. So, basically we are taking the bootstrap sample by replacement.

So, we use the rest of the cases to estimate the error of the tree by predicting their classes. So, these are known as out of bag we are going to discuss what is out of bag in coming slides. So, what do we do we have N number capital N number of samples, we have capital M number of features from there we are selecting a bootstrap sample and then using that bootstrap sample we are going to fit a tree and those samples which are left out we are going to use this for predicting the error from this tree.

So, they are known as the outer bag samples and for developing this tree for splitting criteria, we are going to use the small m number of features which are way less than that of capital M. There are a couple of criteria choose at this is known as m try. So, we can calculate the m try by taking the square root of m in case of classification problem. In case of regression problem we take capital M by tree as the value of m try.

(Refer Slide Time: 18:47)



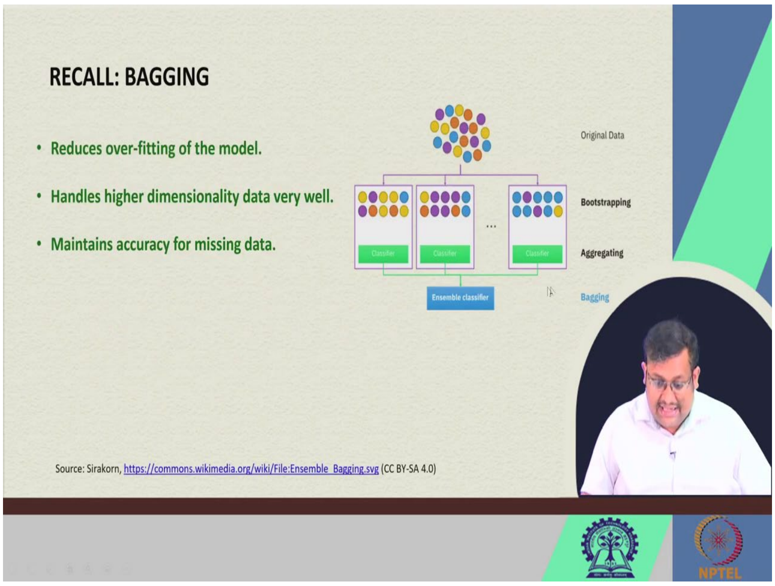
HOW RF IS EXECUTED?

4. For each node of the tree, randomly choose *m* variables on which to base the decision at that node. Calculate the best split based on these *m* variables in the training set.

5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

- Bagging: training each individual learner on different bootstrapped subsets of the data and then averaging the predictions



RECALL: BAGGING

- Reduces over-fitting of the model.
- Handles higher dimensionality data very well.
- Maintains accuracy for missing data.

Source: Sirakorn, https://commons.wikimedia.org/wiki/File:Ensemble_Bagging.svg (CC BY-SA 4.0)

So, for each node of the tree randomly choose the small m variables which is m try on which to base the decision at the, at that node and calculate the base split based on these m variables in the training set. Now, each tree will fully grown and not pruned as may be done in case of constructing a normal tree classifier.

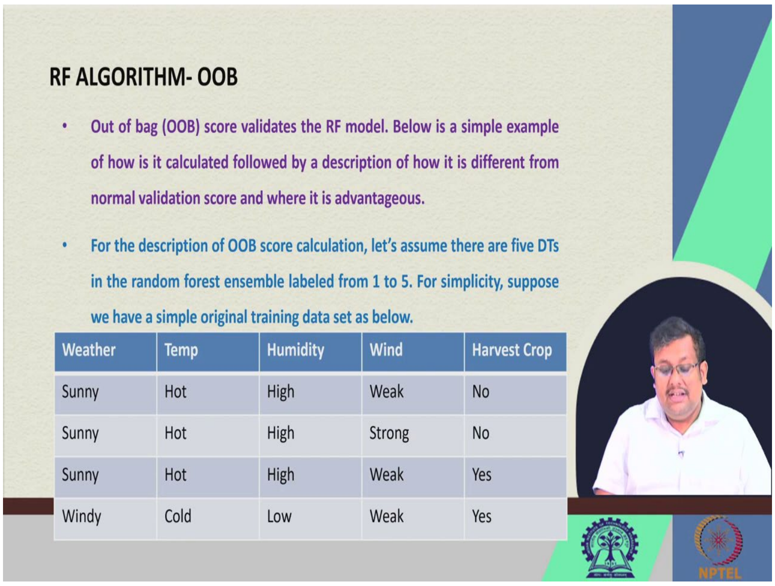
So, here, we generally want to grow the full tree without pruning it. So, far predicting a new sample so, the tree is built very good. Now the next step, what is the next step? So, for a predicting a new sample, that sample is pushed down the tree it is assigned to assign the level of the training sample in the terminal node it ends up in.

So, this procedure is iterated over all the trees in the ensemble and the average vote of all the trees is reported as a random forest prediction. So, what happens we have a sample, we have a validation sample. So, we put that validation sample in each of these individual trees and it will end up to one of these terminal node and we will take the value of the terminal node and after we get suppose there are 1000 trees.

So, for these 1000 trees, we will get 1000 outcomes. From there, we will take for a classification problem, we will take the majority of the vote, and we will assign that label to the final as a final prediction for that sample. So, this is how this prediction is being done. Now, what is bagging I have already told you it is training each individual learner on different bootstrap subset of the data and then averaging the predictions.

So, if you go back and see that we are basically call it a bagging because we are taking different bootstrap samples and we are averaging these learners to get our final decision. So, that is called the bootstrap aggregation or bagging.

(Refer Slide Time: 20:38)



RF ALGORITHM- OOB

- Out of bag (OOB) score validates the RF model. Below is a simple example of how is it calculated followed by a description of how it is different from normal validation score and where it is advantageous.
- For the description of OOB score calculation, let's assume there are five DTs in the random forest ensemble labeled from 1 to 5. For simplicity, suppose we have a simple original training data set as below.

Weather	Temp	Humidity	Wind	Harvest Crop
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

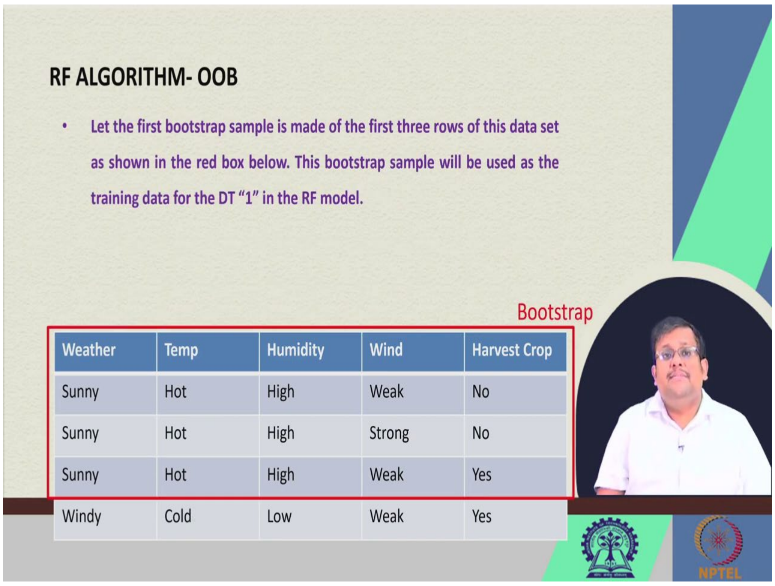
So, let us take a very simple example of this out of bag prediction. So, the out of bag score validates the random forest model you will make frequently encountered this term that is out of bag or OOB. So, below is the simple example of how it is calculated followed by a description of how it is different from normal validation square where it is advantageous.

So, here you can see this is a table where we have 5 observation and we can see there are 4 variables, whether it is a sunny or windy temperature and then humanity and then wind and then based on that whether you are going to harvest your crops or not. So, if the weather is sunny then it is temperature is hot then humid, humid climate, high humidity, and weak wind then will not harvest the crop.

If it is sunny, then hot then high then strong wind, then we will get we will know we will know we will not harvest the crop. If it is sunny, hot temperature, high humidity, weak wind we will harvest the crop. If it is windy then cold and it is low humidity and weak wind of course we will harvest the crop.

So, for the description of this OOB score where calculation let us assume there are 5 decision trees in the random forest labeled from 1 to 5. So, decision tree 1 decision tree 2 decision tree 3 decision tree 4 decision tree 5.

(Refer Slide Time: 22:07)



RF ALGORITHM- OOB

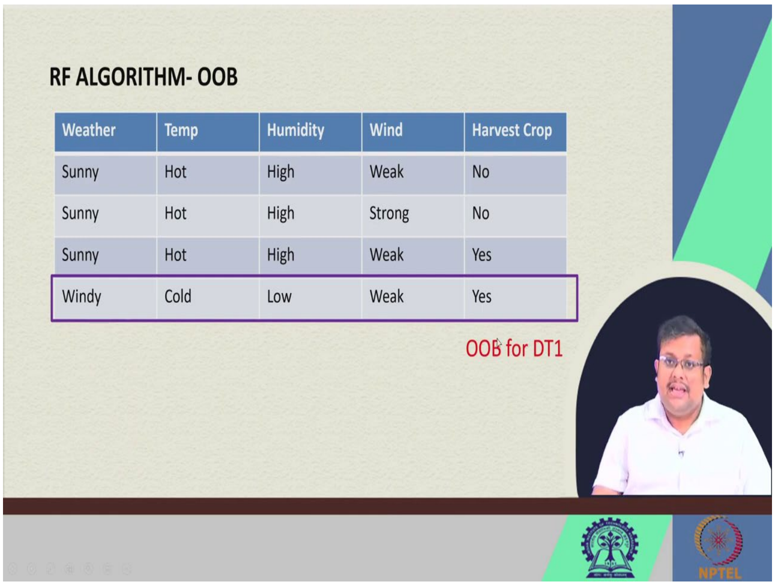
- Let the first bootstrap sample is made of the first three rows of this data set as shown in the red box below. This bootstrap sample will be used as the training data for the DT "1" in the RF model.

Bootstrap

Weather	Temp	Humidity	Wind	Harvest Crop
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

For simplicity, suppose we have a simple original training set. So, these are the 4 observation we can see. Let us assume that the first bootstrap sample is made up the first 3 rows. So, these first 3 rows will make the first bootstrap sample. So, let us assume that the first tree will be built on this bootstrap sample. And the decision tree will 1 will be based on this bootstrap sample.

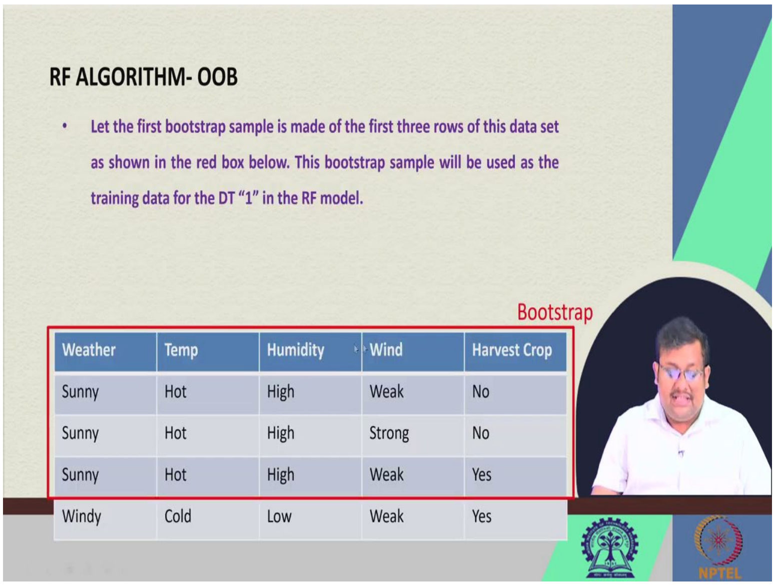
(Refer Slide Time: 22:25)



RF ALGORITHM- OOB

Weather	Temp	Humidity	Wind	Harvest Crop
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

OOB for DT1



RF ALGORITHM- OOB

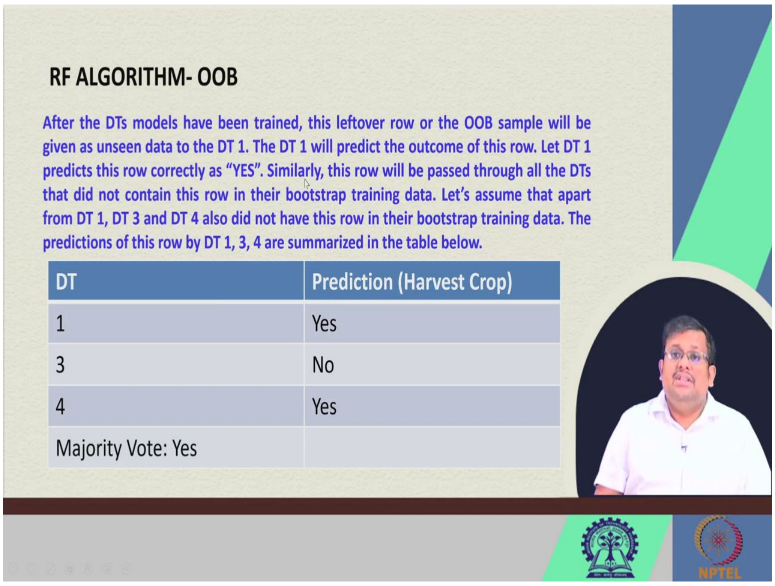
- Let the first bootstrap sample is made of the first three rows of this data set as shown in the red box below. This bootstrap sample will be used as the training data for the DT "1" in the RF model.

Bootstrap

Weather	Temp	Humidity	Wind	Harvest Crop
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

So, the left out sample will be this 4th observation. So, this is known as the out of bag sample. So, this is the bootstrap sample and this is the out of bag sample. So, basically, we will validate the decision tree 1 using our bootstrap sample. So, you see here for the sake of simplicity, we are showing one bootstrap sample but assume but remember that there could be more than 1 bootstrap sample also.

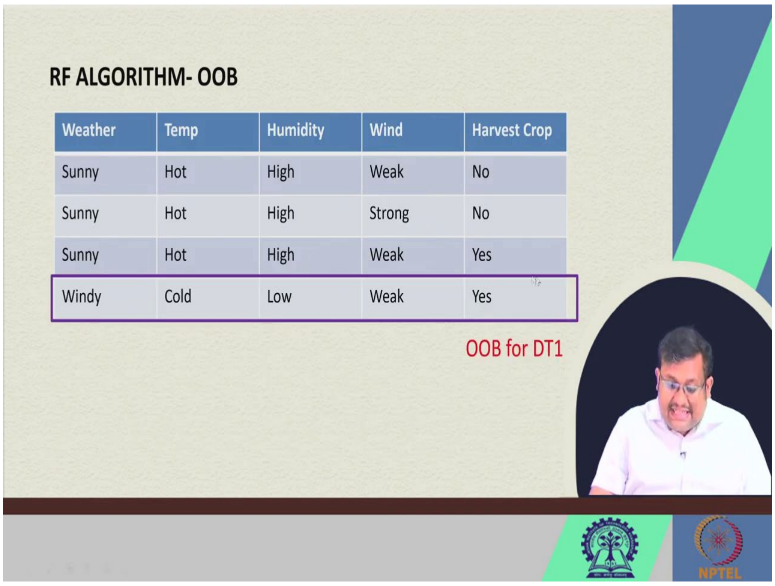
(Refer Slide Time: 22:55)



RF ALGORITHM- OOB

After the DTs models have been trained, this leftover row or the OOB sample will be given as unseen data to the DT 1. The DT 1 will predict the outcome of this row. Let DT 1 predicts this row correctly as "YES". Similarly, this row will be passed through all the DTs that did not contain this row in their bootstrap training data. Let's assume that apart from DT 1, DT 3 and DT 4 also did not have this row in their bootstrap training data. The predictions of this row by DT 1, 3, 4 are summarized in the table below.

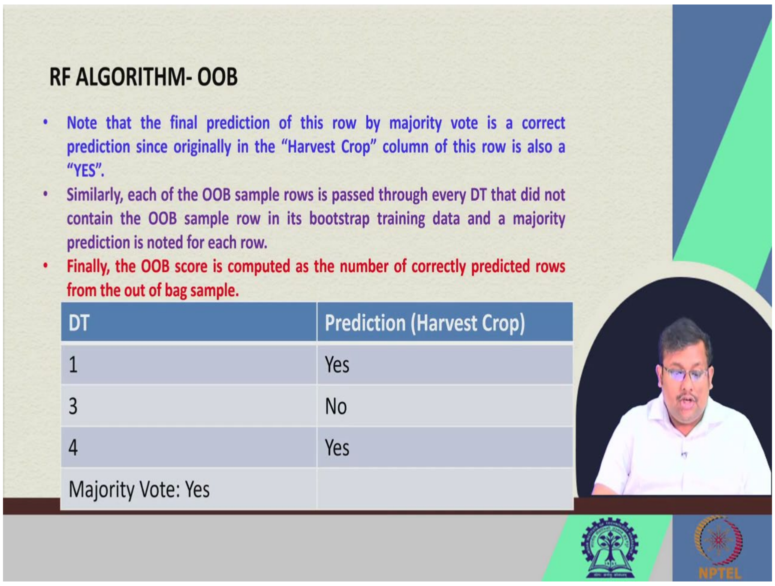
DT	Prediction (Harvest Crop)
1	Yes
3	No
4	Yes
Majority Vote: Yes	



RF ALGORITHM- OOB

Weather	Temp	Humidity	Wind	Harvest Crop
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

OOB for DT1



RF ALGORITHM- OOB

- Note that the final prediction of this row by majority vote is a correct prediction since originally in the "Harvest Crop" column of this row is also a "YES".
- Similarly, each of the OOB sample rows is passed through every DT that did not contain the OOB sample row in its bootstrap training data and a majority prediction is noted for each row.
- Finally, the OOB score is computed as the number of correctly predicted rows from the out of bag sample.

DT	Prediction (Harvest Crop)
1	Yes
3	No
4	Yes
Majority Vote: Yes	

So, anyway based on this bootstrap sample, so, we get the result from the decision tree 1. So, after all the decision tree models have been trained, this leftover row of the bootstrap samples or out of sorry out of the bag samples will be given an unseen data to the decision tree 1. So, let us assume this decision tree 1 will predict the outcome of this row let us assume that it is yes will harvest this crop based on this out of bag sample.

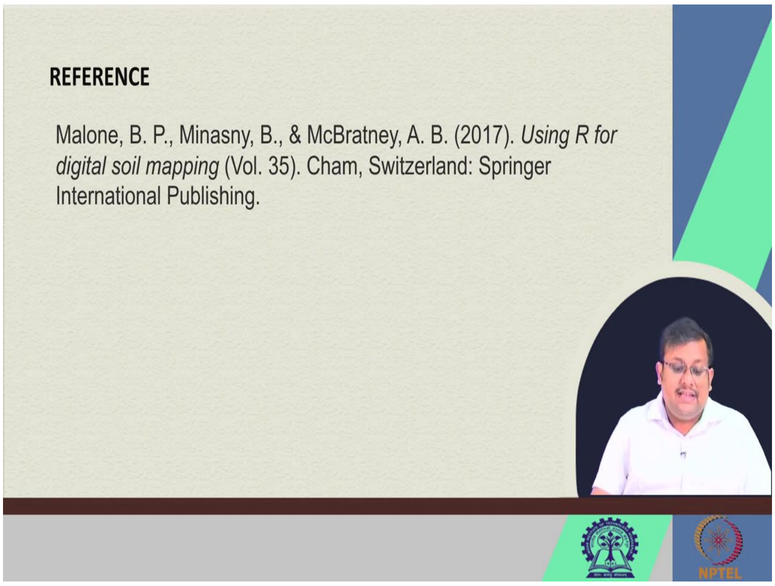
So, based on this out of bag sample suppose decision tree is predicting yes, we will harvest the crop. Similarly, this row will be passed through all the decision trees where it will be considered as an out of bag. So, this 4th row let us assume that this fourth row may be considered as out of bag for decision tree 1 decision tree 3 decision tree 4.

So, in these 3 decision trees we have kept this 4th observation as an out of bag sample. And for each of these 3 decision trees we are getting the result as yes no and yes. Now, you see the majority of the vote is yes. So, that means we are going to give the final prediction as yes.

So, this is how we calculate the result based on this out of bag score. So, you can see here the final prediction of this row by majority of the vote is a correct prediction since originally in the harvest crop column of this row it is also yes. So, if you go back and see the originally the bootstrap sample also yes, we are going to harvest crops so, the prediction is also correct.

Similarly, each of these out of bag sample rows is passed through these every decision tree that did not contain these out of bag sample though in its bootstrap training data and a majority predictions is noted for each row. And finally, the out of bag score is computed as the number of correctly predicted rows for the out of bag sample. So, this is how we calculate this random forest specifically classification.

(Refer Slide Time: 25:12)



REFERENCE

Malone, B. P., Minasny, B., & McBratney, A. B. (2017). *Using R for digital soil mapping* (Vol. 35). Cham, Switzerland: Springer International Publishing.

So, guys, let us wrap up our lecture here, I have used the same reference as previously. And in our next lecture we are going to see how we can use R for executing the random forest based classification of the categorical variables. So, please stay tuned and let us meet in our 4th lecture to discuss this further and see how to execute this random forest classification using R for digital soil mapping. Thank you guys. Let us meet in our next lecture.