Machine Learning for Soil and Crop Management
Professor Somsubhra Chakraborty
Agricultural and Food Engineering Department
Indian Institute of Technology Kharagpur
Lecture – 57
Digital Soil Mapping with Categorical Variables (Contd.)

Welcome friends to this second lecture of last week that is week 12 of NNPTEL online certification course of machine learning for soil and crop management. And in this week we are trying to discuss the digital soil mapping with categorical variables. In our first lecture, we have already discussed the hybrid approach of regression Kriging.

And in this lecture, we are going to see how to calculate several how to see different types of accuracy measures from a confusion matrix and we are going to first develop a confusion matrix and then based on that confusion matrix we are going to calculate several model accuracy parameters.

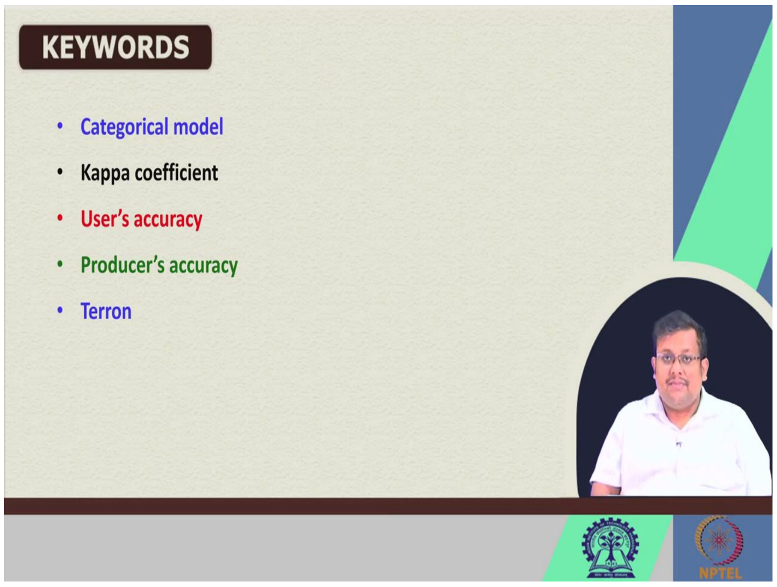
(Refer Slide Time: 1:13)

CONCEPTS COVERED

- Categorical Modelling in DSM
- Kappa coefficient and other accuracy measures
- Multinomial logistic model

And so, these are the concepts which we are going to discuss first of all we are going to discuss the categorical modeling in DSM and then we are going to discuss the Kappa coefficient and other accuracy measures and then we are going to discuss the multinomial logistic model.

(Refer Slide Time: 1:29)

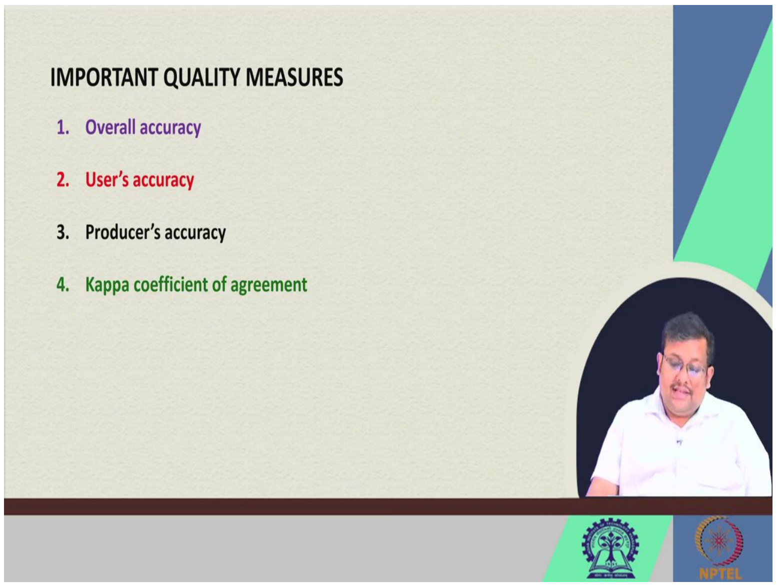


KEYWORDS

- Categorical model
- Kappa coefficient
- User's accuracy
- Producer's accuracy
- Terron

So, these are the keywords categorical model, then Kappa coefficient, user accuracy, producer's accuracy and Terron. These are some of the keywords for this lecture.

(Refer Slide Time: 1:42)

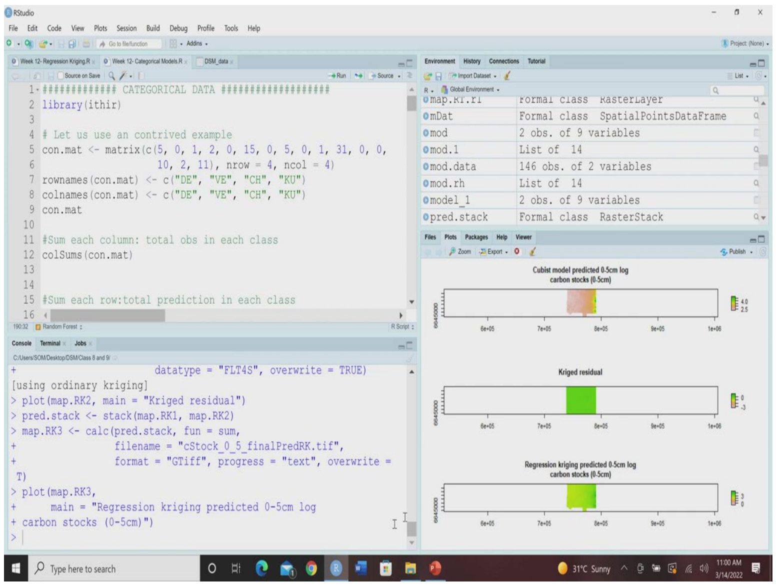


IMPORTANT QUALITY MEASURES

1. Overall accuracy
2. User's accuracy
3. Producer's accuracy
4. Kappa coefficient of agreement

And also some important quality measures we are going to see in this lecture on the overall accuracy, user's accuracy, producer's accuracy, Kappa coefficient of agreement okay guys.

(Refer Slide Time: 2:05)

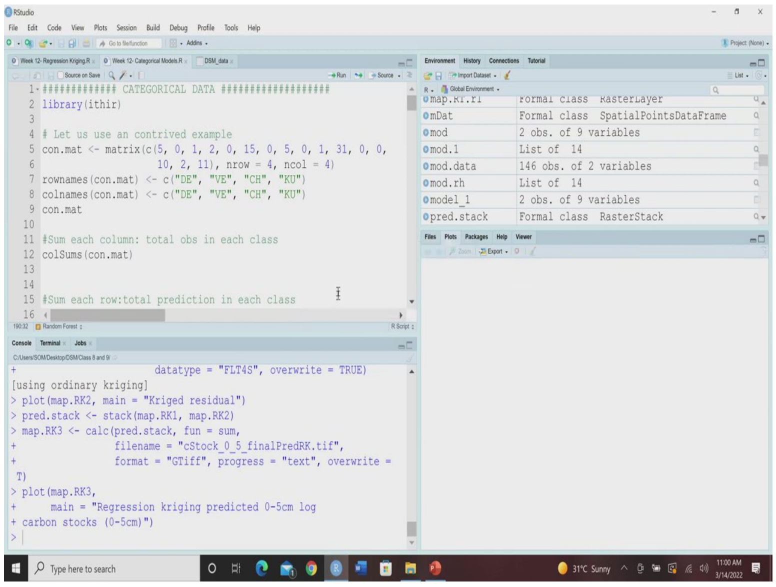


```
############# CATEGORICAL DATA ##################
library(ithir)

# Let us use an contrived example
con.mat <- matrix(c(5, 0, 1, 2, 0, 15, 0, 5, 0, 1, 31, 0, 0,
                    10, 2, 11), nrow = 4, ncol = 4)
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
```

```
map.RK1       Formal class  RasterLayer
mDat          Formal class  SpatialPointsDataFrame
mod           2 obs. of 9 variables
mod.1         List of  14
mod.data      146 obs. of 2 variables
mod.rh        List of  14
model_1       2 obs. of 9 variables
pred.stack    Formal class  RasterStack
```

```
+                      datatype = "FLT4S", overwrite = TRUE)
[using ordinary kriging]
> plot(map.RK2, main = "Kriged residual")
> pred.stack <- stack(map.RK1, map.RK2)
> map.RK3 <- calc(pred.stack, fun = sum,
+                 filename = "cStock_0_5_finalPredRK.tif",
+                 format = "GTiff", progress = "text", overwrite =
 T)
> plot(map.RK3,
+      main = "Regression kriging predicted 0-5cm log
+ carbon stocks (0-5cm)")
>
```

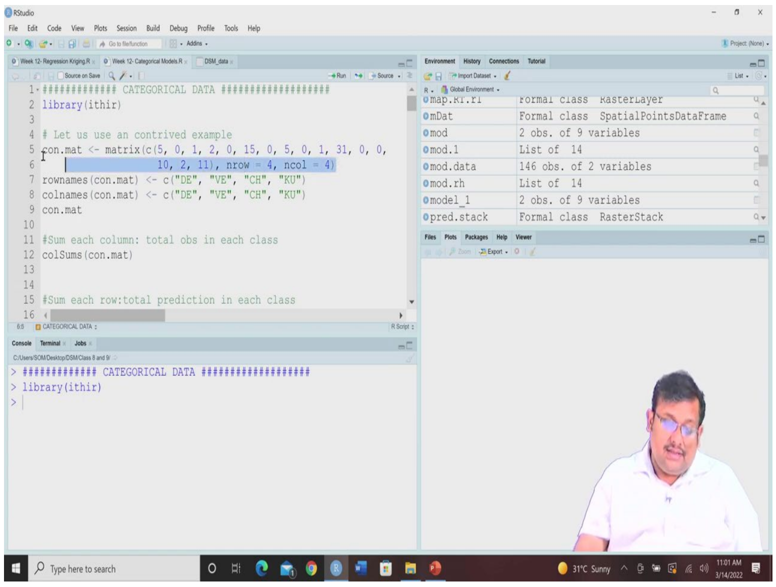
Cubist model predicted 0-5cm log carbon stocks (0-5cm)
Kriged residual
Regression kriging predicted 0-5cm log carbon stocks (0-5cm)



```
############# CATEGORICAL DATA ##################
library(ithir)

# Let us use an contrived example
con.mat <- matrix(c(5, 0, 1, 2, 0, 15, 0, 5, 0, 1, 31, 0, 0,
                    10, 2, 11), nrow = 4, ncol = 4)
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
```

```
map.RK1       Formal class  RasterLayer
mDat          Formal class  SpatialPointsDataFrame
mod           2 obs. of 9 variables
mod.1         List of  14
mod.data      146 obs. of 2 variables
mod.rh        List of  14
model_1       2 obs. of 9 variables
pred.stack    Formal class  RasterStack
```

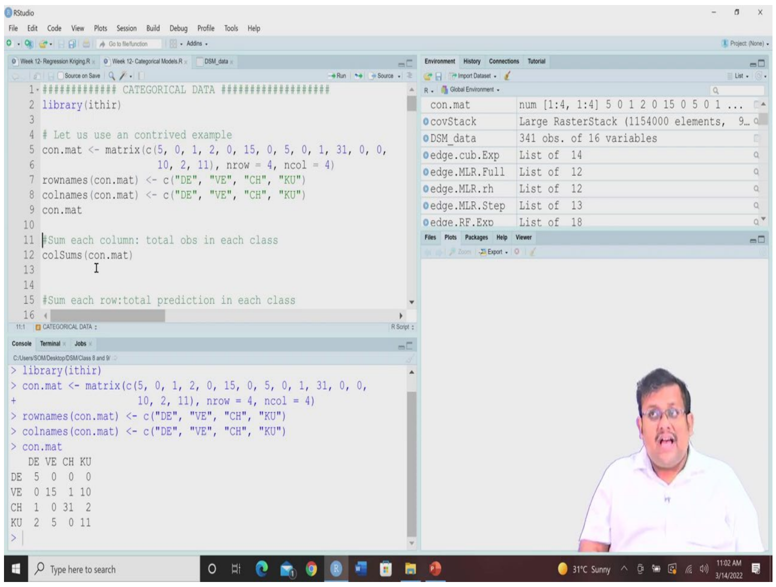
```
+                      datatype = "FLT4S", overwrite = TRUE)
[using ordinary kriging]
> plot(map.RK2, main = "Kriged residual")
> pred.stack <- stack(map.RK1, map.RK2)
> map.RK3 <- calc(pred.stack, fun = sum,
+                 filename = "cStock_0_5_finalPredRK.tif",
+                 format = "GTiff", progress = "text", overwrite =
 T)
> plot(map.RK3,
+      main = "Regression kriging predicted 0-5cm log
+ carbon stocks (0-5cm)")
>
```



```
############# CATEGORICAL DATA ##################
library(ithir)

# Let us use an contrived example
con.mat <- matrix(c(5, 0, 1, 2, 0, 15, 0, 5, 0, 1, 31, 0, 0,
                    10, 2, 11), nrow = 4, ncol = 4)
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
```

```
map.RK1       Formal class  RasterLayer
mDat          Formal class  SpatialPointsDataFrame
mod           2 obs. of 9 variables
mod.1         List of  14
mod.data      146 obs. of 2 variables
mod.rh        List of  14
model_1       2 obs. of 9 variables
pred.stack    Formal class  RasterStack
```

```
> ############# CATEGORICAL DATA ##################
> library(ithir)
>
```



```r
############# CATEGORICAL DATA ##################
library(ithir)

# Let us use an contrived example
con.mat <- matrix(c(5, 0, 1, 2, 0, 15, 0, 5, 0, 1, 31, 0, 0,
                    10, 2, 11), nrow = 4, ncol = 4)
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
```

```
> library(ithir)
> con.mat <- matrix(c(5, 0, 1, 2, 0, 15, 0, 5, 0, 1, 31, 0, 0,
+                    10, 2, 11), nrow = 4, ncol = 4)
> rownames(con.mat) <- c("DE", "VE", "CH", "KU")
> colnames(con.mat) <- c("DE", "VE", "CH", "KU")
> con.mat
   DE VE CH KU
DE  5  0  0  0
VE  0 15  1 10
CH  1  0 31  2
KU  2  5  0 11
>
```

```
con.mat        num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack       Large RasterStack (1154000 elements,  9…
DSM_data       341 obs. of 16 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
```

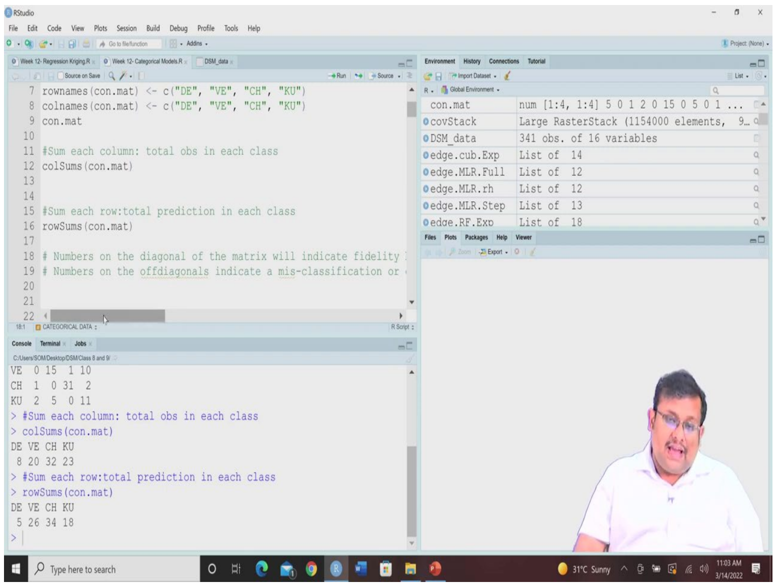
So, let me just go ahead and discuss we are going to discuss this categorical model. But before going to discuss the categorical model, let us again call this library ithir. Let us call this library ithir and then for this lecture I am going to develop in a contrived example. Where we are going to develop a confusion matrix using some values with for number of rows 4 and number and 4 number of columns.

And we are going to give some names of those columns and some and also we are going to see how these coefficient matrix looks like. So, for this contrived example, we are going to use this matrix function to develop this confusion matrix once we have developed this confusion matrix, let us assume that we are giving the name that these four are the different names for different classes and of course, the column name is also same.

So, let us see that this confusion matrix you can see here, this is the confusion matrix of course, in this confusion matrix, these diagonal elements like in case of DE. So, here you can see 5, 15, 31, 11 so, all these 5, 15, 31, 11 are the so, the diagonal components of this confusion matrix, it is showing the correctly classified samples.

So, here along these columns these are the observed classes and these are the predicted classes in the row, in the row we can see the predicted classes in the columns we can see the observed classes. So, if we see that what is the total accuracy of classification accuracy. So, for getting the total classification accuracy, we are going to use the, for calculating the total classification accuracy it is important to calculate the sums of all the columns. So, for that or in other words total observation in each class.

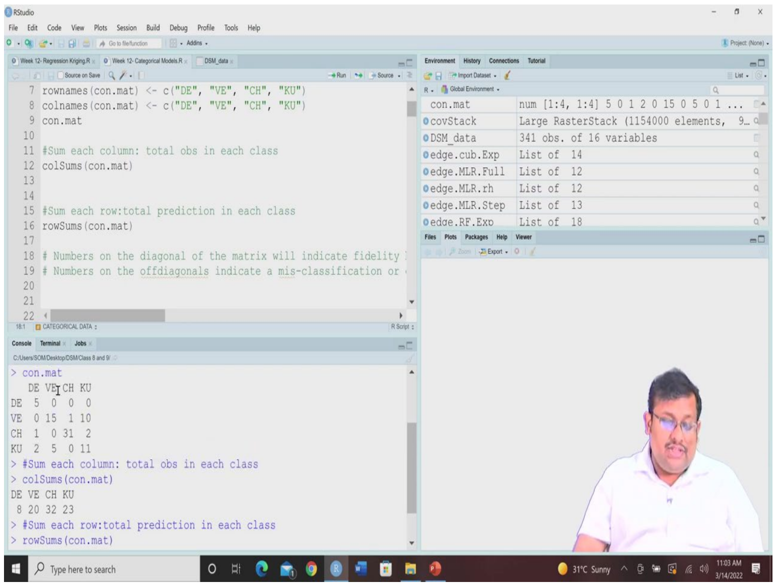
(Refer Slide Time: 4:38)



```
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
rowSums(con.mat)

# Numbers on the diagonal of the matrix will indicate fidelity
# Numbers on the offdiagonals indicate a mis-classification or
```

```
con.mat         num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack        Large RasterStack (1154000 elements, 9…
DSM_data        341 obs. of 16 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
```

```
VE  0 15  1 10
CH  1  0 31  2
KU  2  5  0 11
> #Sum each column: total obs in each class
> colSums(con.mat)
DE VE CH KU
 8 20 32 23
> #Sum each row:total prediction in each class
> rowSums(con.mat)
DE VE CH KU
 5 26 34 18
>
```



```
rownames(con.mat) <- c("DE", "VE", "CH", "KU")
colnames(con.mat) <- c("DE", "VE", "CH", "KU")
con.mat

#Sum each column: total obs in each class
colSums(con.mat)


#Sum each row:total prediction in each class
rowSums(con.mat)

# Numbers on the diagonal of the matrix will indicate fidelity
# Numbers on the offdiagonals indicate a mis-classification or
```

```
> con.mat
   DE VE CH KU
DE  5  0  0  0
VE  0 15  1 10
CH  1  0 31  2
KU  2  5  0 11
> #Sum each column: total obs in each class
> colSums(con.mat)
DE VE CH KU
 8 20 32 23
> #Sum each row:total prediction in each class
> rowSums(con.mat)
```

So, we can use these col sums function from this confusion matrix and let us see that for DE it is 8. So, these are the number of samples within each class these are observed values. Now, also at the same time we can also use these row sums function to get the total prediction in each of these classes.

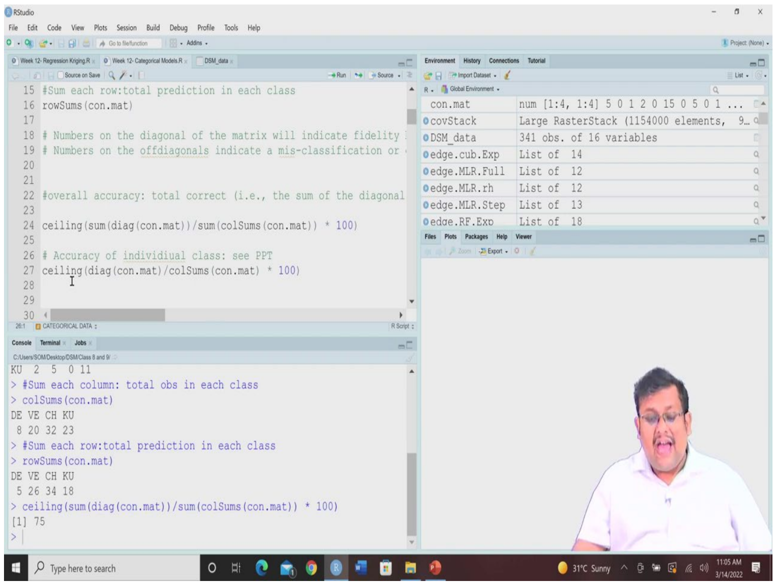
So, you can see the total prediction in DE is 5, for VE is 26, for CH is 34, for KU it is 18. Now, as I have said previously that the numbers on the diagonal of the matrix will indicate the fidelity between the observed and the subsequent prediction and numbers on the off diagonals are shown as the misclassification.

So, of course, you can see in this contract examples or in this confusion matrix, you can see here this 5 shows the fidelity between the DE and DE, 15 shows the fidelity between VE and

VE. So, observed and predicted class 31 shows the, fidelity between observed CH and the predicted CH and 11 shows the fidelity between observed KU and predicted KU.

And all other values like 1, 2 these are the off diagonal points and shows the misclassification within this D E class these 5 is the misclassification within the VE, this is a misclassification within the 1 is the misclassification within the CH and these 12 will be the misclassified samples within the clear KU class.

(Refer Slide Time: 6:54)



```
#Sum each row:total prediction in each class
rowSums(con.mat)

# Numbers on the diagonal of the matrix will indicate fidelity
# Numbers on the offdiagonals indicate a mis-classification or


#overall accuracy: total correct (i.e., the sum of the diagonal

ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)

# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)
```

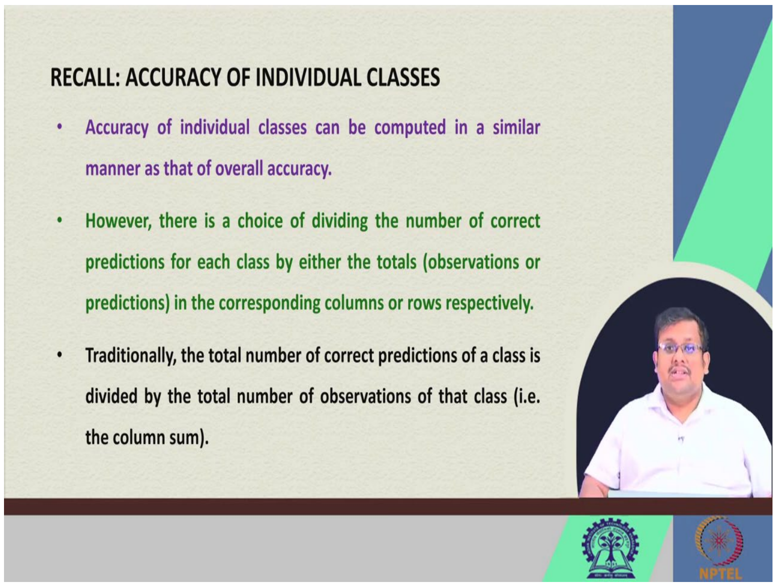
```
KU  2  5  0 11
> #Sum each column: total obs in each class
> colSums(con.mat)
DE VE CH KU
 8 20 32 23
> #Sum each row:total prediction in each class
> rowSums(con.mat)
DE VE CH KU
 5 26 34 18
> ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)
[1] 75
>
```

```
con.mat         num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack        Large RasterStack (1154000 elements,  9…
DSM_data        341 obs. of 16 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
```

Now, to calculate the overall accuracy, we are going to use this formula that is total correct by total number of observations. So, total correct the sum of the diagonals and total number of observation you can take the sum of the columns sums.

So, if we go and use this for this we are going to use this ceiling function. So, ceiling function some of the diagonal of confusion matrix by some of the column sense of the confusion matrix multiply by 100. So, you can see that this shows the overall accuracy that is 75 percent.

(Refer Slide Time: 7:36)



RECALL: ACCURACY OF INDIVIDUAL CLASSES

- Accuracy of individual classes can be computed in a similar manner as that of overall accuracy.
- However, there is a choice of dividing the number of correct predictions for each class by either the totals (observations or predictions) in the corresponding columns or rows respectively.
- Traditionally, the total number of correct predictions of a class is divided by the total number of observations of that class (i.e. the column sum).

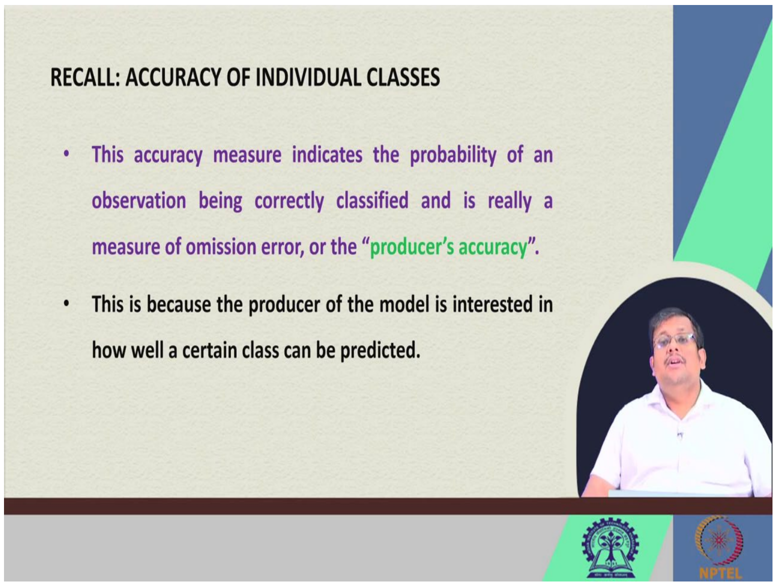
Now, if we are concerned about the accuracy of the individual class, then we have to calculate these accuracy of the individual class in two ways. One is using user's accuracy and other is producer's accuracy.

Now, if you go back to our PPT and see what are the user's accuracy and producer's accuracy you have to recall that accuracy of individual classes can be computed in a similar manner as that of overall accuracy. However, there is a choice of dividing the number of correct predictions for each class by either the total observation or the predictions.

So, you can either take the division between the number of correct prediction by the number of total prediction within that class or number of observation within that class in the corresponding columns or rows respectively.

Now, so, if we again we can take, we can calculate that, by for individual class we can divide the number of correct prediction by the number of observation or by the number of prediction in the corresponding columns and rows respectively. And traditionally, the total number of correct prediction of a class is divided by the total number of observation in the class that is the column sum it is a traditional way.

(Refer Slide Time: 8:46)



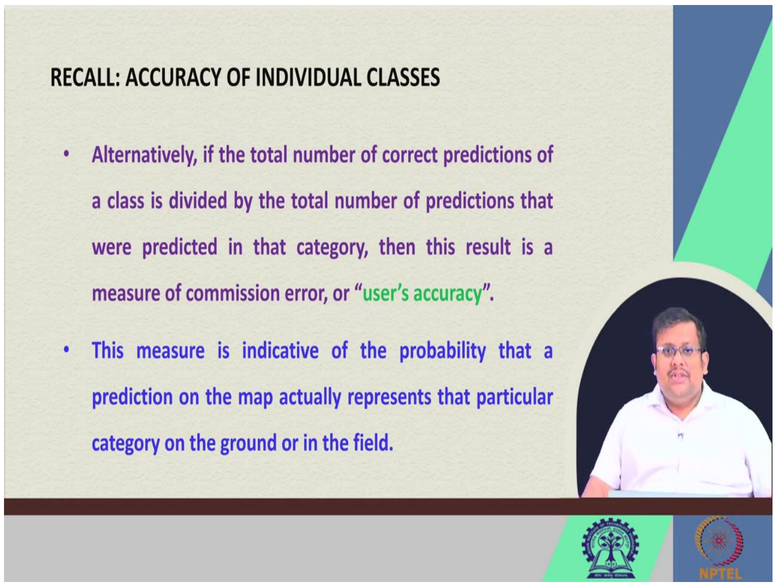
RECALL: ACCURACY OF INDIVIDUAL CLASSES

- This accuracy measure indicates the probability of an observation being correctly classified and is really a measure of omission error, or the "producer's accuracy".
- This is because the producer of the model is interested in how well a certain class can be predicted.

Now, these accuracy measures when we calculate the number of samples divided by the column sum, then these accuracy measure indicate the probability of an observation being correctly classified and is really a measure of omission errors or it is also known as the producer's accuracy.

This is because the producer of the model is interested in how well a certain class can be predicted. So, this is called the producer's accuracy and also calculate the omission error.

(Refer Slide Time: 9:21)



RECALL: ACCURACY OF INDIVIDUAL CLASSES

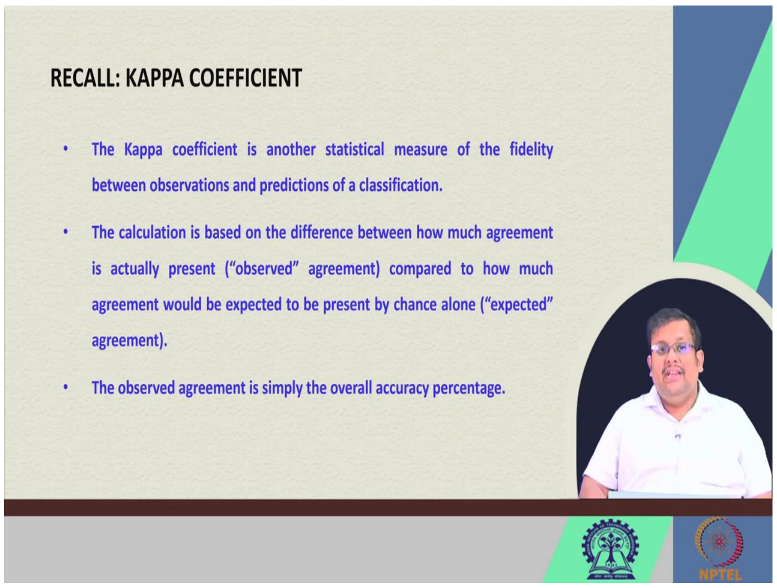
- Alternatively, if the total number of correct predictions of a class is divided by the total number of predictions that were predicted in that category, then this result is a measure of commission error, or "user's accuracy".
- This measure is indicative of the probability that a prediction on the map actually represents that particular category on the ground or in the field.

Now, I told you there is another way of calculating the accuracy of the individual class by dividing the number of correct prediction by the samples within the individual, individual row so, if the total number of correct predictions of the class is divided by the total number of prediction that were predicted in that category, then this result is also known as the user's accuracy and it basically measures the commission error.

Now, this measure is indicative of the probability that a prediction on the map actually represents that particular category on the ground in the field or in the field. So, these are the two types of accuracy measures for individual classes. One is called the user's accuracy producer's accuracy another is called the user's accuracy.

Again, producer's accuracy measures the omission error whereas the user's accuracy measures the commission error so, these are the differences for producer accuracy, we have to divide the number of correctly classified samples with the individual column sum, but in case of the user's accuracy, we have to divide the number of correctly classified sample by the correct by the row sum. So, this is the difference between producer's accuracy and user's accuracy.

(Refer Slide Time: 10:48)



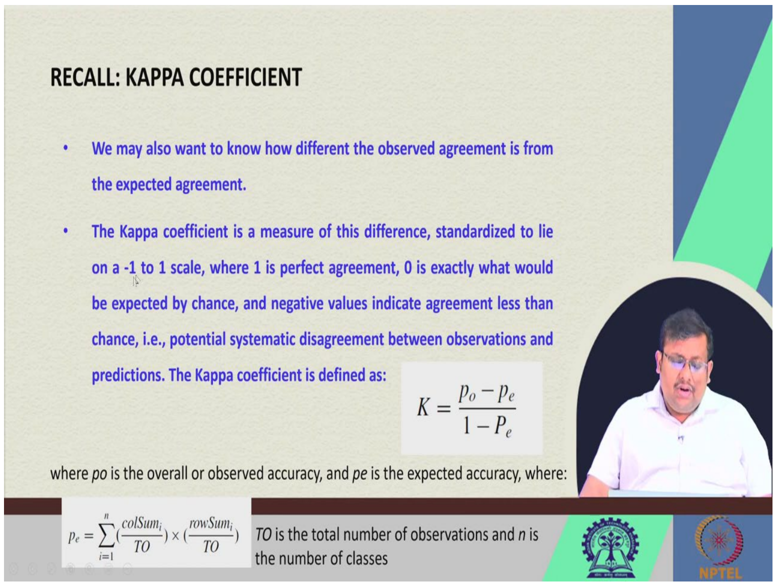
RECALL: KAPPA COEFFICIENT

- The Kappa coefficient is another statistical measure of the fidelity between observations and predictions of a classification.
- The calculation is based on the difference between how much agreement is actually present ("observed" agreement) compared to how much agreement would be expected to be present by chance alone ("expected" agreement).
- The observed agreement is simply the overall accuracy percentage.

Now, also, the Kappa coefficient is another the Kappa coefficient is another statistical measures or the, of the fidelity between the observation and prediction of a classification. And the calculation is based on the difference between how much agreement is actually present that is observed agreement compared to how much agreement would be expected to be present by chance alone that is expected agreement.

So, observed agreement by expected agreement gives you the Kappa coefficient. So, observe agreement is simply remember that in this Kappa coefficient calculation this observed agreement is showing basically the overall accuracy percentage.

(Refer Slide Time: 11:27)



RECALL: KAPPA COEFFICIENT

- We may also want to know how different the observed agreement is from the expected agreement.
- The Kappa coefficient is a measure of this difference, standardized to lie on a -1 to 1 scale, where 1 is perfect agreement, 0 is exactly what would be expected by chance, and negative values indicate agreement less than chance, i.e., potential systematic disagreement between observations and predictions. The Kappa coefficient is defined as:

$$K = \frac{p_o - p_e}{1 - P_e}$$

where *po* is the overall or observed accuracy, and *pe* is the expected accuracy, where:

$$p_e = \sum_{i=1}^{n} \left(\frac{colSum_i}{TO}\right) \times \left(\frac{rowSum_i}{TO}\right)$$

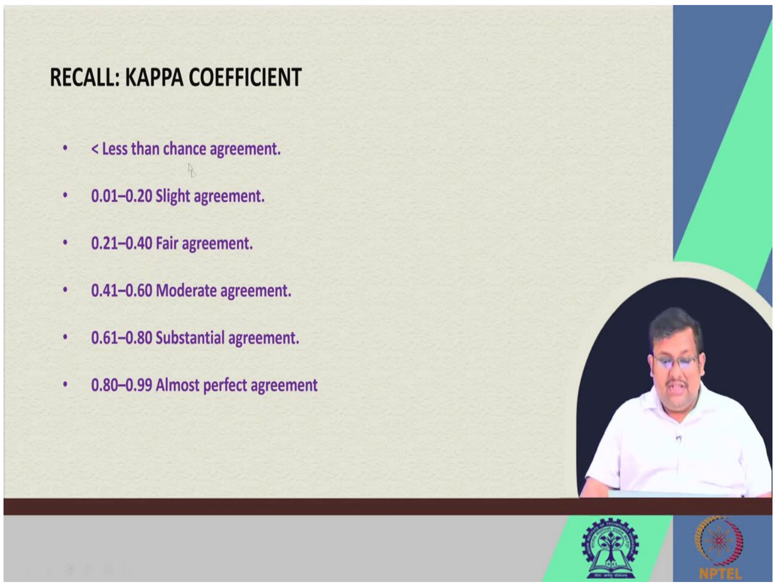
TO is the total number of observations and *n* is the number of classes

So, we may also want to know how different these observed agreement is from the expected agreement. So, this Kappa coefficient is a measure of this difference and standardized to a scale of minus 1 to plus 1. So, Kappa coefficient can take any value from minus 1 to plus 1 whereas these plus 1 is showing the perfect agreement 0 is exactly what would be expected by chance and negative agreement that is negative values indicate agreement less than the chance.

So, that is potential systematic disagreement between the observation and then prediction. So, the Kappa coefficient can be calculated by using this formula where this k equal to p 0 minus pe and 1 minus pe.

So, p 0 is the overall or observed accuracy, we know that we have already calculated that and p is the expected accuracy where this pe can be calculated by using this formula that is T O stands for Total number of observation, n is the number of classes and this is the individual column sum and individual rows sum from 1 to n. So, this is how you calculate this Kappa coefficient.

(Refer Slide Time: 12:43)

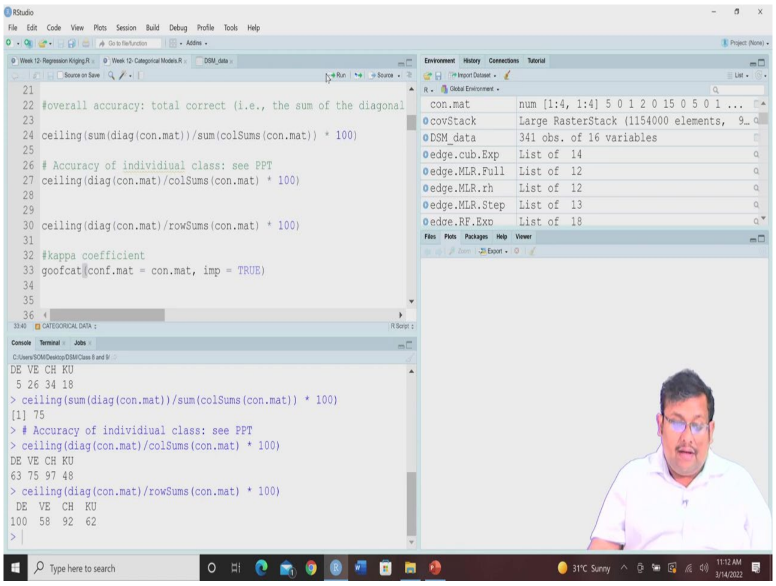


RECALL: KAPPA COEFFICIENT

- < Less than chance agreement.
- 0.01–0.20 Slight agreement.
- 0.21–0.40 Fair agreement.
- 0.41–0.60 Moderate agreement.
- 0.61–0.80 Substantial agreement.
- 0.80–0.99 Almost perfect agreement

And of course, this Kappa coefficient can be classified based on its value, when it is less than 0 of course this shows the less than chance agreement then 0.01 to 0.20 that shows the slight agreement. It is 0.21 to 0.40 fair agreement, 0.41 to 0.60 moderate agreement and then 0.61 to 0.80 substantial agreement and 0.80 to 0.99 almost perfect agreement. So, this is how you classify the values of the Kappa coefficient.

(Refer Slide Time: 13:25)



```
#overall accuracy: total correct (i.e., the sum of the diagonal

ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)

# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)


ceiling(diag(con.mat)/rowSums(con.mat) * 100)

#kappa coefficient
goofcat(conf.mat = con.mat, imp = TRUE)
```

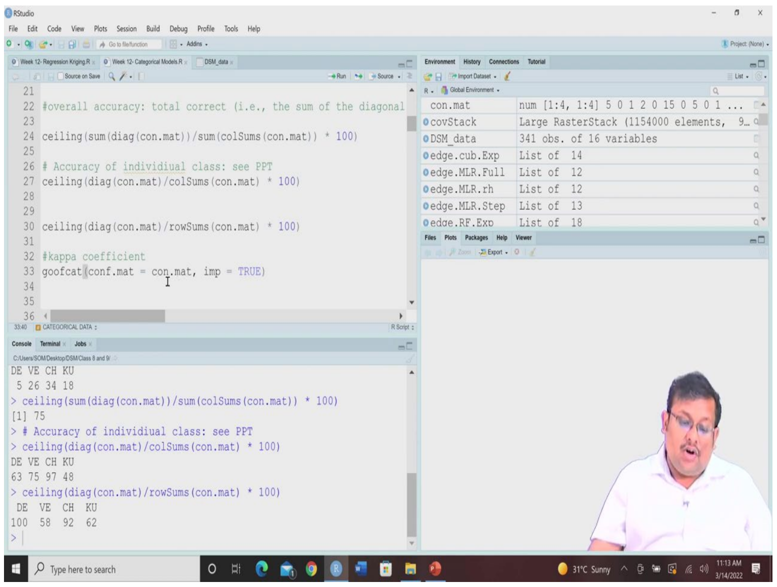
```
DE VE CH KU
 5 26 34 18
> ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)
[1] 75
> # Accuracy of individiual class: see PPT
> ceiling(diag(con.mat)/colSums(con.mat) * 100)
DE VE CH KU
63 75 97 48
> ceiling(diag(con.mat)/rowSums(con.mat) * 100)
 DE  VE  CH  KU
100  58  92  62
>
```

```
con.mat        num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack       Large RasterStack (1154000 elements,  9…
DSM_data       341 obs. of 16 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
```

Now, let us go back to our R and see how we can calculate these different components. So, if we want to see the column sum, so, this is the, we want to see the producer’s accuracy. So, you can see that confusion matrix we are going to sum the diagonal elements of the confusion matrix by the, and we divide it by the column sums multiplied by 100. Of course, this will give you the producers accuracy, and then if you are going to get the user's accuracy, it will be simply dividing by the sum of the rows individual rows.

Now, if we want to get the Kappa coefficient remember in R instead of using simple goof function, we are going to use this goofcat function, goofcat function is goodness of fit of categorical model. So, using this goofcat function here you can get the results. So, we are going to use this goofcat function and let us see how this Kappa coefficient can be calculated.

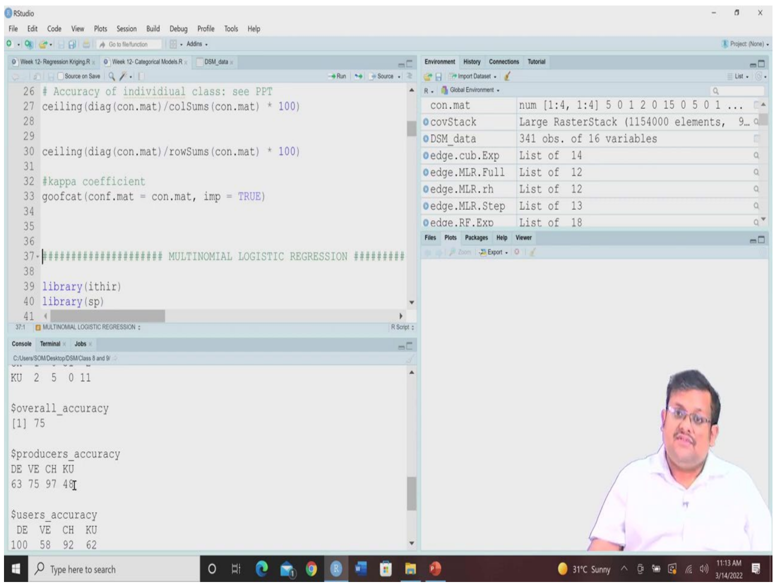
(Refer Slide Time: 14:42)



```
#overall accuracy: total correct (i.e., the sum of the diagonal

ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)

# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)


ceiling(diag(con.mat)/rowSums(con.mat) * 100)

#kappa coefficient
goofcat(conf.mat = con.mat, imp = TRUE)
```

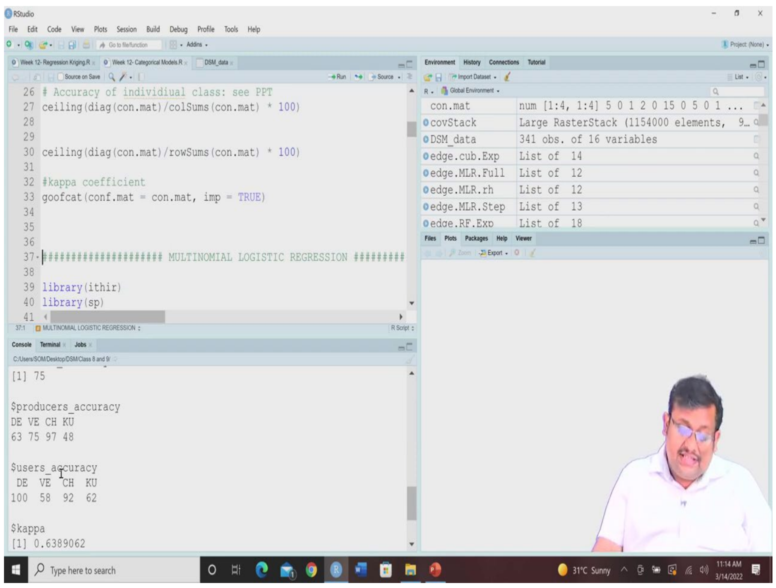
```
DE VE CH KU
 5 26 34 18
> ceiling(sum(diag(con.mat))/sum(colSums(con.mat)) * 100)
[1] 75
> # Accuracy of individiual class: see PPT
> ceiling(diag(con.mat)/colSums(con.mat) * 100)
DE VE CH KU
63 75 97 48
> ceiling(diag(con.mat)/rowSums(con.mat) * 100)
 DE  VE  CH  KU
100  58  92  62
>
```

```
con.mat          num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack         Large RasterStack (1154000 elements, 9…
DSM_data         341 obs. of 16 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
```



```
# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)


ceiling(diag(con.mat)/rowSums(con.mat) * 100)

#kappa coefficient
goofcat(conf.mat = con.mat, imp = TRUE)



##################### MULTINOMIAL LOGISTIC REGRESSION #########

library(ithir)
library(sp)
```

```
KU  2  5  0 11

$overall_accuracy
[1] 75

$producers_accuracy
DE VE CH KU
63 75 97 48

$users_accuracy
 DE  VE  CH  KU
100  58  92  62
```



```
# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)


ceiling(diag(con.mat)/rowSums(con.mat) * 100)

#kappa coefficient
goofcat(conf.mat = con.mat, imp = TRUE)



#################### MULTINOMIAL LOGISTIC REGRESSION #########

library(ithir)
library(sp)
```

```
con.mat        num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack       Large RasterStack (1154000 elements, 9…
DSM_data       341 obs. of 16 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
```

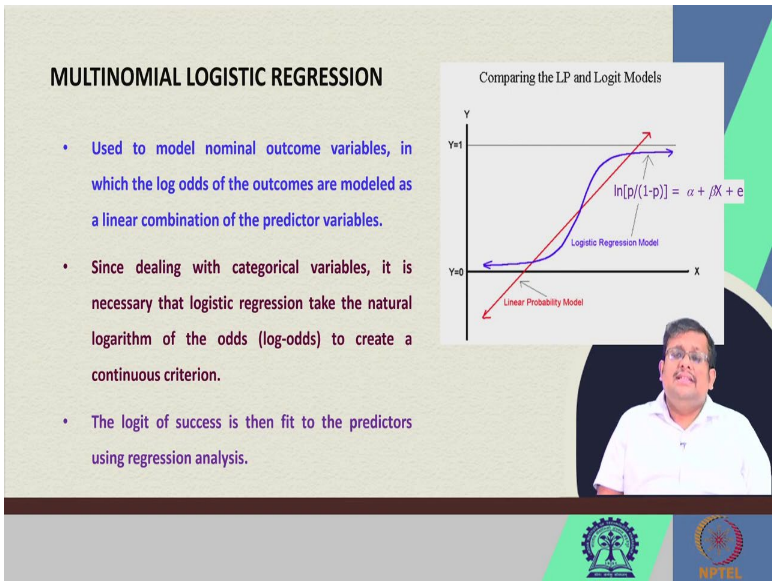
```
[1] 75

$producers_accuracy
DE VE CH KU
63 75 97 48

$users_accuracy
 DE  VE  CH  KU
100  58  92  62

$kappa
[1] 0.6389062
```

So, we have already know how to about this confusion matrix. Now, we are going to show you how to calculate this goofcat based on this confusion matrix. So, here we are going to use this confusion matrix argument and here we are giving this con dot mat is our confusion matrix.

Let us see just if we run this thing then you will see that how this confusion matrix will give you the results. So, this is the confusion metric overall accuracy we already know and then producer's accuracy for individual classes you can see for 63 percent for DE, 75 percent for VE, 97 percent for CH, and 48 percent for KU.

And case of user's accuracy 100 percent for DE, then 58 percent for VE, then 92 percent for CH, and then 62 percent for KU. So, and also finally, the Kappa coefficient is 0.63. So, this is how, for any confusion matrix using the goofcat function in R you can calculate not only the user accuracy producer's accuracy, you can also calculate the Kappa coefficient and based on the value of the Kappa coefficient, you can conclude whether that is accurate. How much agreement between the observed category and the expected category you are getting.

(Refer Slide Time: 16:20)



MULTINOMIAL LOGISTIC REGRESSION

- Used to model nominal outcome variables, in which the log odds of the outcomes are modeled as a linear combination of the predictor variables.
- Since dealing with categorical variables, it is necessary that logistic regression take the natural logarithm of the odds (log-odds) to create a continuous criterion.
- The logit of success is then fit to the predictors using regression analysis.

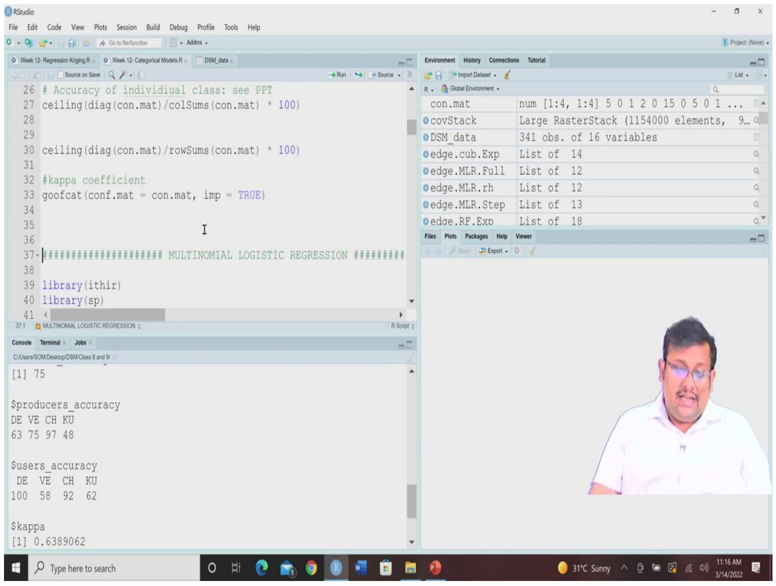
Now, guys, let us move ahead and let us discuss one actual categorical model in our multi in which we use in digital soil mapping that is multinomial logistic regression. Now, multinomial logistic regression generally we use it for modeling the nominal outcome. So, it is a kind of a logistic regression.

And we generally use it for modeling that nominal outcome variable that is categorical variables in which the log of odds of the outcomes are modeled as a linear combination of the predictor variables. Now, the logistic regression we have already discussed the logistic regression.

Now, in case of logistic regression instead of the linear probability model we generally use the logistic regression model to keep our probability between these 0 to 1 threshold levels and then we fit the linear regression based on the log of odds or so, this is the logistic model. Now, since dealing with the categorical variables it is necessary that logistic regression takes the natural logarithm of the core, of the odds that is log of odds to create a continuous criterion.

And this logit of success is then fit to the predictors using the regression analysis, we already have covered this thing in our regression discussion and in our classification discussion previously in week 4. So, we already know that this is the logistic regression.

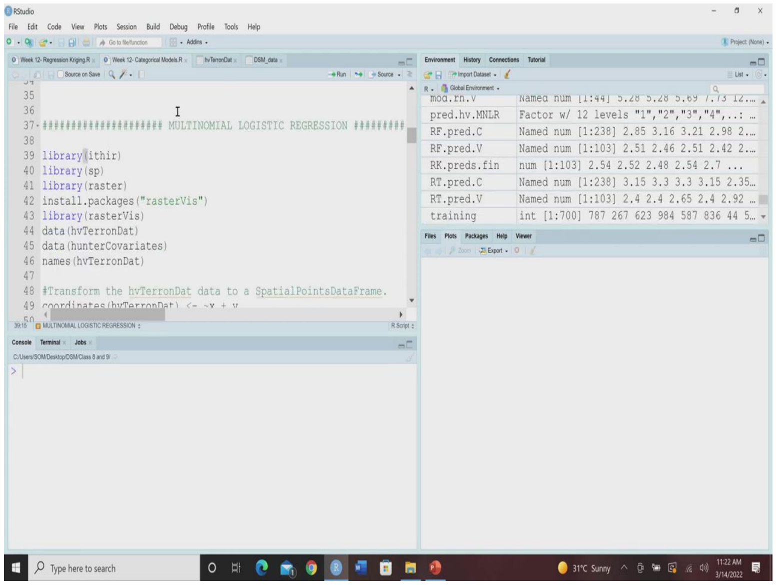
(Refer Slide Time: 18:11)



```
# Accuracy of individiual class: see PPT
ceiling(diag(con.mat)/colSums(con.mat) * 100)


ceiling(diag(con.mat)/rowSums(con.mat) * 100)

#kappa coefficient
goofcat(conf.mat = con.mat, imp = TRUE)



#################### MULTINOMIAL LOGISTIC REGRESSION ########

library(ithir)
library(sp)
```

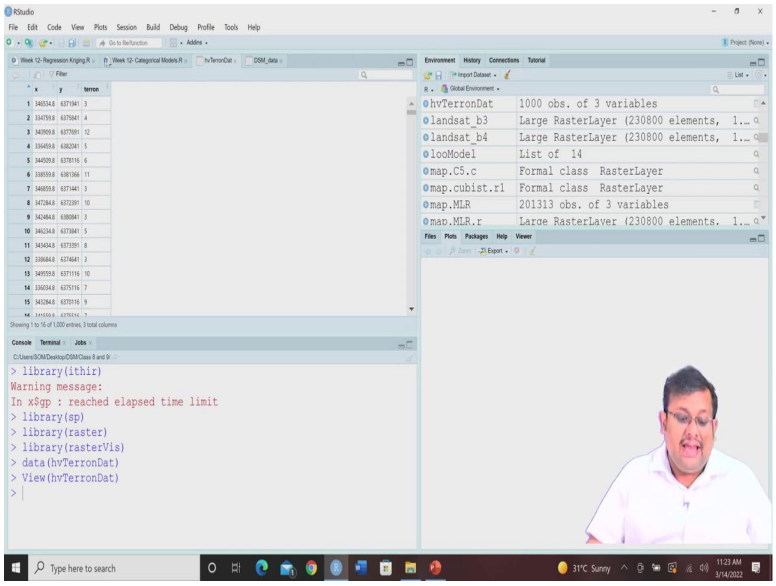
```
[1] 75

$producers_accuracy
DE VE CH KU
63 75 97 48

$users_accuracy
 DE  VE  CH  KU
100  58  92  62

$kappa
[1] 0.6389062
```

```
con.mat         num [1:4, 1:4] 5 0 1 2 0 15 0 5 0 1 ...
covStack        Large RasterStack (1154000 elements, 9…
DSM_data        341 obs. of 16 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
```



```


#################### MULTINOMIAL LOGISTIC REGRESSION ########

library(ithir)
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
```

```
mod.rh.V        Named num [1:44] 5.28 5.28 5.69 7.73 12.…
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…
```

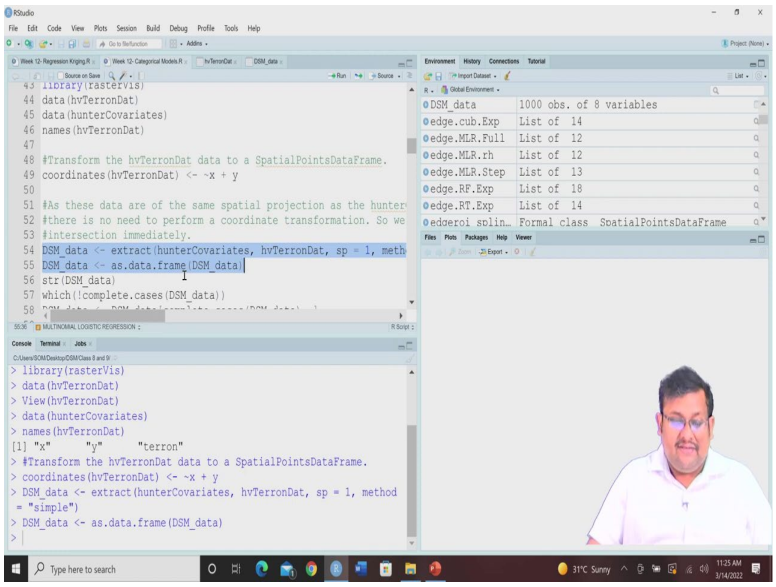


	x	y	terron
1	3465348	6371941	3
2	3347598	6375841	4
3	3409098	6377691	12
4	3364598	6382041	5
5	3445098	6378116	6
6	3385598	6381366	11
7	3468598	6371441	3
8	3472848	6372391	10
9	3424848	6380841	3
10	3462348	6373841	5
11	3434348	6373391	8
12	3386848	6374641	3
13	3495598	6371116	10
14	3360348	6375116	7
15	3432848	6370116	9

Showing 1 to 16 of 1,000 entries, 3 total columns

```
> library(ithir)
Warning message:
In x$gp : reached elapsed time limit
> library(sp)
> library(raster)
> library(rasterVis)
> data(hvTerronDat)
> View(hvTerronDat)
>
```

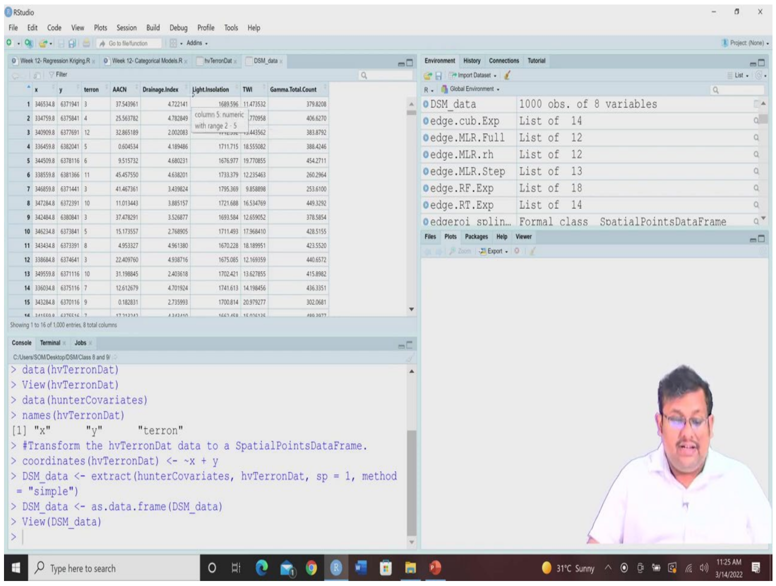
```
hvTerronDat     1000 obs. of 3 variables
landsat_b3      Large RasterLayer (230800 elements, 1.…
landsat_b4      Large RasterLayer (230800 elements, 1.…
looModel        List of  14
map.C5.c        Formal class  RasterLayer
map.cubist.r1   Formal class  RasterLayer
map.MLR         201313 obs. of 3 variables
map.MLR.r       Large RasterLayer (230800 elements, 1.…
```



```r
library(rasterVis)
data(hvTerronDat)
data(hunterCovariates)
names(hvTerronDat)

#Transform the hvTerronDat data to a SpatialPointsDataFrame.
coordinates(hvTerronDat) <- ~x + y

#As these data are of the same spatial projection as the hunter
#there is no need to perform a coordinate transformation. So we
#intersection immediately.
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, meth
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
which(!complete.cases(DSM_data))
```

```
> library(rasterVis)
> data(hvTerronDat)
> View(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
```

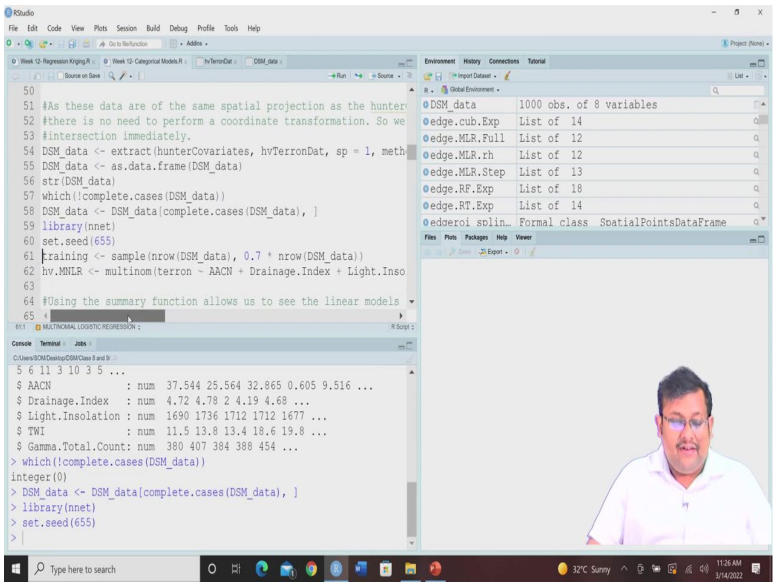


	x	y	terron	AACN	Drainage.Index	Light.Insolation	TWI	Gamma.Total.Count
1	346534.8	6371941	3	37.543961	4.722141	1689.596	11.473532	379.8208
2	334759.8	6375841	4	25.563782	4.782849	[illegible]	[illegible]	406.6270
3	340909.8	6377691	12	32.865189	2.002083	[illegible]	[illegible]	383.8792
4	336459.8	6382041	5	0.604534	4.189486	1711.715	18.555082	388.4246
5	344509.8	6378116	6	9.515732	4.680231	1676.977	19.770855	454.2711
6	338559.8	6381366	11	45.457550	4.638201	1733.379	12.235463	260.2964
7	346859.8	6371441	3	41.467361	3.439824	1795.369	9.858898	253.6100
8	347284.8	6372391	10	11.013443	3.885157	1721.688	16.534769	449.3292
9	342484.8	6380841	3	37.478291	3.526877	1693.584	12.659052	378.5854
10	346234.8	6373841	5	15.173557	2.768905	1711.493	17.968410	428.5155
11	343434.8	6373391	8	4.953327	4.961380	1670.228	18.189951	423.5520
12	338684.8	6374641	3	22.409760	4.938716	1675.085	12.169359	440.6572
13	349559.8	6371116	10	31.198845	2.403618	1702.421	13.627855	415.8982
14	336034.8	6375116	7	12.612679	4.701924	1741.613	14.198456	436.3351
15	343284.8	6370116	9	0.182831	2.735993	1700.814	20.979277	302.0681

column 5: numeric with range 2 - 5

Showing 1 to 16 of 1,000 entries, 8 total columns

```
> data(hvTerronDat)
> View(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)
> View(DSM_data)
> 
```



```r

#As these data are of the same spatial projection as the hunter
#there is no need to perform a coordinate transformation. So we
#intersection immediately.
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(nnet)
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
hv.MNLR <- multinom(terron ~ AACN + Drainage.Index + Light.Inso

#Using the summary function allows us to see the linear models
```

```
 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(nnet)
> set.seed(655)
> 
```

Now, we are going to see how we can execute this multinomial logistic regression using in DSM using R. So, you can see guys we are going to use this multinomial logistic regression, let us see how to go with this and so, we are going to use this ithir package and then let us call this library ithir. And then let us call this library raster.

And we are going to install these rasterVis package for visualization for raster visualization. So, we are going to call this library rasterVis and then hvTerronDat. So, this is the data which we are going to use so, this hvTerronDat if you click on it, you will see there are 1000s observation with the three fields x, y and Terron.

So, Terron is basically the class of soil based on certain characteristics. And there are 12 Terron classes from 1 to 12 and they are named accordingly. So, our idea our, the objective of this exercise is to use this multinomial logistic regression to predict the multinomial logistic regression to predict these Terron classes using the covariance data.

Now, what are these covariance data, the covariance data is hunterCovariates for the same location. So, we are going to download these hunterCovariates, hunterCovariates is of course, you can say it is a large raster stack object and then let us see the names of the hvTerronDat. Of course, there are three names x y and Terron you know that.

Now right now this Terron is in a tabular format, so we need to convert it to the special points dataframe you know that. So, for that we are going to use the coordinate function. So, we are instructing R that please understand that these x and y are the 2 coordinates.

Now, you can see it will be changed to hvTerronDat has changed to former class of special points data frame. Now, as they as these data are of the same special projection of the hunterCovariates we do not know we do not need any kind of coordinate transformation. So, we can do direct intersection we know how to intercept the data.

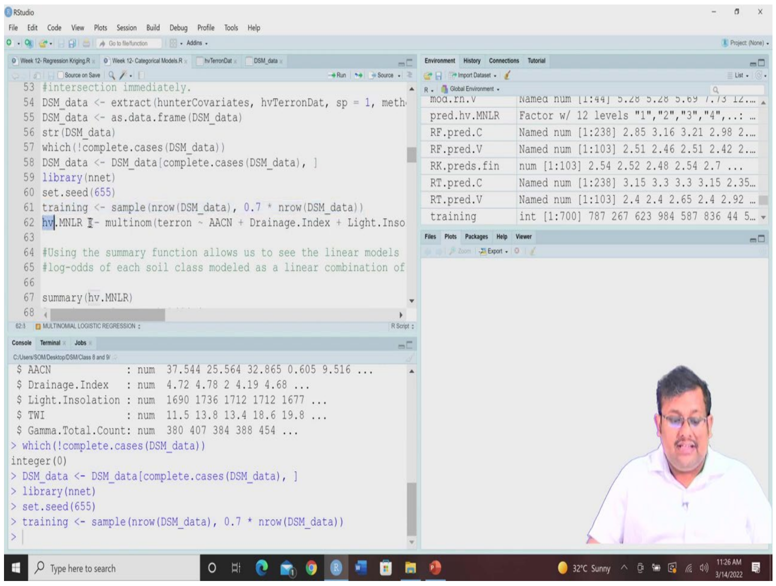
So, for that we are going to extract function and then we are going to see we once we have extracted that now, we converted them back to simple data frame because we need to run some models and models can be only run in simple data frame. So, now, you can see DSM data similarly 1000 observation of 8 variables. What are these 8 variables?

So, you can see x y Terron and then 1, 2, 3, 4, 5 different covariates altitude of channel network, drainage index, light installation, Terron witness index, and gamma total count these are the 5 covariates which are there in the hunter covariates file. So, we have extracted them for these locations.

Now, when data see the structure, so, the structure just like we have seen 1000 observation 8 variables and you can see here altitude of channel network, then drainage index, light installation, Twi, gamma total count and so on so forth. So, now, we want to keep only the complete cases and we want to just keep that for further modeling exercise for this multinomial logistic regression we are going to use this library nnet.

So, this nnet we are going to use. So, these nnet is there in the carrot package which we have already installed previously. So, I am going to call this library nnet, then we are going to set the seat 655 and then we are going to again just like previously we are going to select the 70 percent of the data as the training data set.

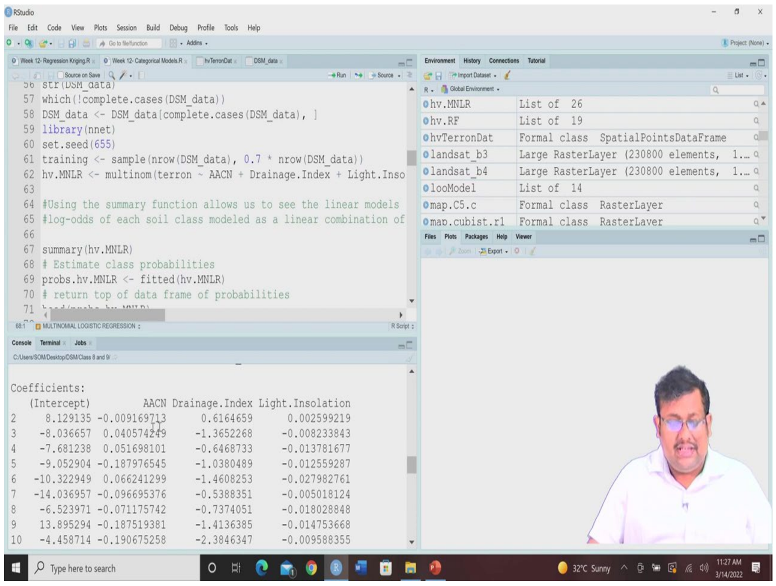
(Refer Slide Time: 22:16)



```
#intersection immediately.
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(nnet)
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
hv.MNLR <- multinom(terron ~ AACN + Drainage.Index + Light.Inso

#Using the summary function allows us to see the linear models
#log-odds of each soil class modeled as a linear combination of

summary(hv.MNLR)
```

```
mod.rn.V       Named num [1:44] 5.28 5.28 5.69 7.73 12....
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...
```

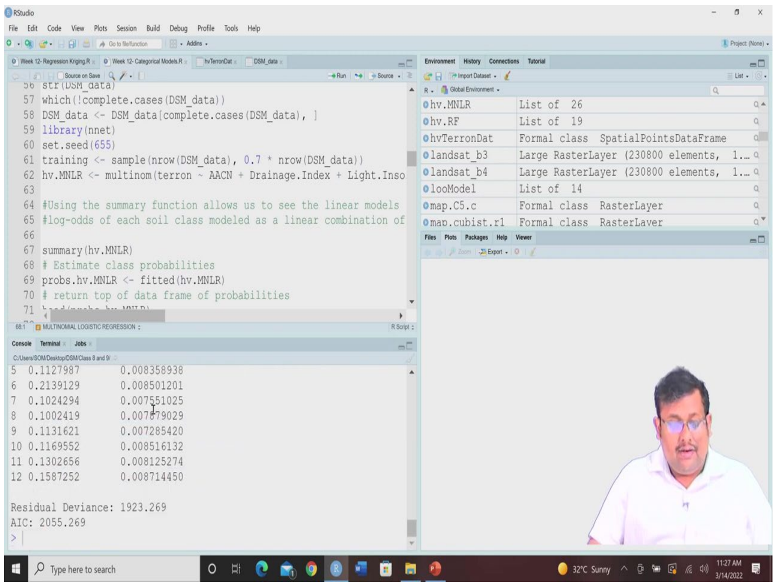
```
 $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
 $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
 $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
 $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
 $ Gamma.Total.Count: num  380 407 384 388 454 ...
> which(!complete.cases(DSM_data))
integer(0)
> DSM_data <- DSM_data[complete.cases(DSM_data), ]
> library(nnet)
> set.seed(655)
> training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
> 
```



```
str(DSM_data)
which(!complete.cases(DSM_data))
DSM_data <- DSM_data[complete.cases(DSM_data), ]
library(nnet)
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
hv.MNLR <- multinom(terron ~ AACN + Drainage.Index + Light.Inso

#Using the summary function allows us to see the linear models
#log-odds of each soil class modeled as a linear combination of

summary(hv.MNLR)
# Estimate class probabilities
probs.hv.MNLR <- fitted(hv.MNLR)
# return top of data frame of probabilities
```

```
hv.MNLR         List of  26
hv.RF           List of  19
hvTerronDat     Formal class  SpatialPointsDataFrame
landsat_b3      Large RasterLayer (230800 elements,  1...
landsat_b4      Large RasterLayer (230800 elements,  1...
looModel        List of  14
map.C5.c        Formal class  RasterLayer
map.cubist.r1   Formal class  RasterLayer
```

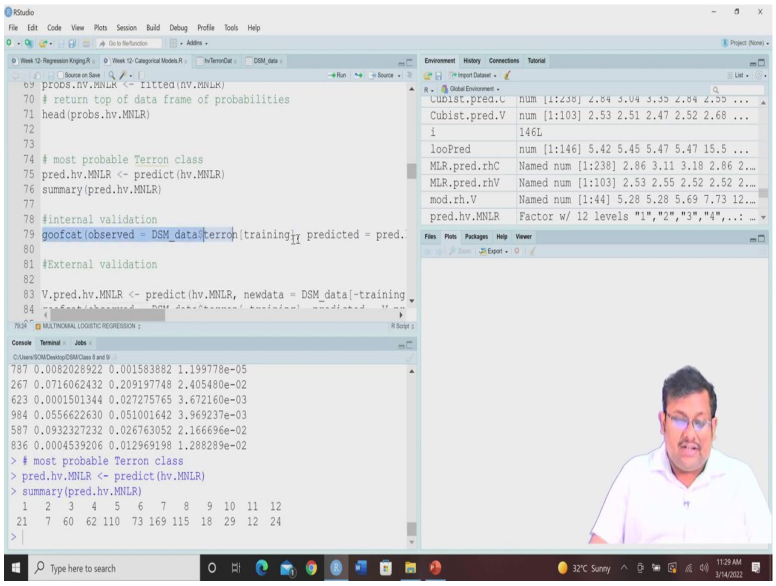
```
Coefficients:
   (Intercept)         AACN Drainage.Index Light.Insolation
2     8.129135 -0.009169713      0.6164659      0.002599219
3    -8.036657  0.040574249     -1.3652268     -0.008233843
4    -7.681238  0.051698101     -0.6468733     -0.013781677
5    -9.052904 -0.187976545     -1.0380489     -0.012559287
6   -10.322949  0.066241299     -1.4608253     -0.027982761
7   -14.036957 -0.096695376     -0.5388351     -0.005018124
8    -6.523971 -0.071175742     -0.7374051     -0.018028848
9    13.895294 -0.187519381     -1.4136385     -0.014753668
10   -4.458714 -0.190675258     -2.3846347     -0.009588355
```



```r
str(DSM_data)
which(!complete.cases(DSM_data))
DSM_data <- DSM_data[complete.cases(DSM_data), ]
library(nnet)
set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
hv.MNLR <- multinom(terron ~ AACN + Drainage.Index + Light.Inso

#Using the summary function allows us to see the linear models
#log-odds of each soil class modeled as a linear combination of

summary(hv.MNLR)
# Estimate class probabilities
probs.hv.MNLR <- fitted(hv.MNLR)
# return top of data frame of probabilities
```

```
hv.MNLR        List of  26
hv.RF          List of  19
hvTerronDat    Formal class  SpatialPointsDataFrame
landsat_b3     Large RasterLayer (230800 elements,  1.…
landsat_b4     Large RasterLayer (230800 elements,  1.…
looModel       List of  14
map.C5.c       Formal class  RasterLayer
map.cubist.r1  Formal class  RasterLayer
```

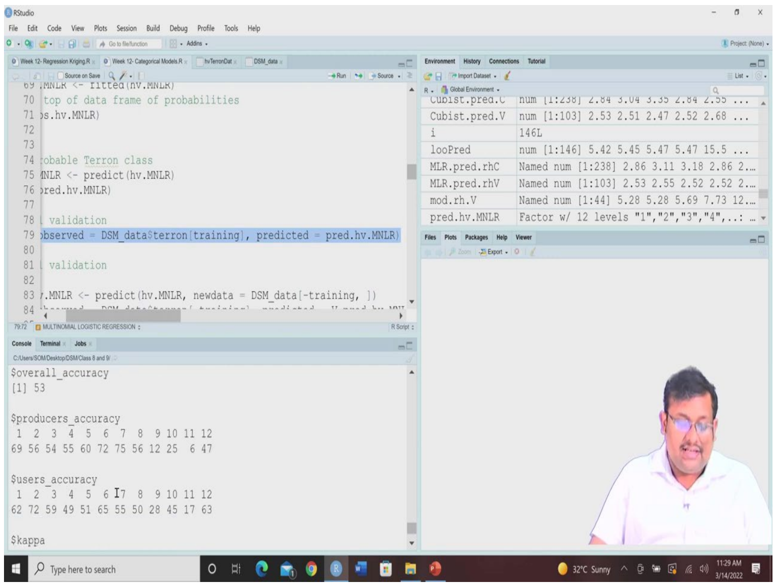
```
5  0.1127987        0.008358938
6  0.2139129        0.008501201
7  0.1024294        0.007551025
8  0.1002419        0.007679029
9  0.1131621        0.007285420
10 0.1169552        0.008516132
11 0.1302656        0.008125274
12 0.1587252        0.008714450

Residual Deviance: 1923.269
AIC: 2055.269
>
```



```r
probs.hv.MNLR <- fitted(hv.MNLR)
# return top of data frame of probabilities
head(probs.hv.MNLR)


# most probable Terron class
pred.hv.MNLR <- predict(hv.MNLR)
summary(pred.hv.MNLR)

#internal validation
goofcat(observed = DSM_data$terron[training], predicted = pred.

#External validation

V.pred.hv.MNLR <- predict(hv.MNLR, newdata = DSM_data[-training
```

```
Cubist.pred.V  num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
i              146L
looPred        num [1:146] 5.42 5.45 5.47 5.47 15.5 ...
MLR.pred.rhC   Named num [1:238] 2.86 3.11 3.18 2.86 2.…
MLR.pred.rhV   Named num [1:103] 2.53 2.55 2.52 2.52 2.…
mod.rh.V       Named num [1:44] 5.28 5.28 5.69 7.73 12.…
pred.hv.MNLR   Factor w/ 12 levels "1","2","3","4",..: …
```

```
787 0.0082028922 0.001583882 1.199778e-05
267 0.0716062432 0.209197748 2.405480e-02
623 0.0001501344 0.027275765 3.672160e-03
984 0.0556622630 0.051001642 3.969237e-03
587 0.0932327232 0.026763052 2.166696e-02
836 0.0004539206 0.012969198 1.288289e-02
> # most probable Terron class
> pred.hv.MNLR <- predict(hv.MNLR)
> summary(pred.hv.MNLR)
  1   2   3   4   5   6   7   8   9  10  11  12
 21   7  60  62 110  73 169 115  18  29  12  24
>
```



```r
.MNLR <- fitted(hv.MNLR)
top of data frame of probabilities
s.hv.MNLR)


obable Terron class
NLR <- predict(hv.MNLR)
red.hv.MNLR)

 validation
bserved = DSM_data$terron[training], predicted = pred.hv.MNLR)

 validation

.MNLR <- predict(hv.MNLR, newdata = DSM_data[-training, ])
```

```
Cubist.pred.V  num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
i              146L
looPred        num [1:146] 5.42 5.45 5.47 5.47 15.5 ...
MLR.pred.rhC   Named num [1:238] 2.86 3.11 3.18 2.86 2.…
MLR.pred.rhV   Named num [1:103] 2.53 2.55 2.52 2.52 2.…
mod.rh.V       Named num [1:44] 5.28 5.28 5.69 7.73 12.…
pred.hv.MNLR   Factor w/ 12 levels "1","2","3","4",..: …
```

```
$overall_accuracy
[1] 53

$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
69 56 54 55 60 72 75 56 12 25  6 47

$users_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
62 72 59 49 51 65 55 50 28 45 17 63

$kappa
```

Now, we are going to fit the multinomial logistic regression using our target is Terron and our predictors are AACN and all these 5 covariates and then we are going to use this our covariates as predictors. So, you can see using the summary function, you can allows us to see the linear models on the each of these Terron class which are the results of the log odds of the each soil class model as a linear combination of the covariates and log odds of each class model as a linear combination of the covariates.

So, if you click on the summary status of you can see here for there are 12 models and for these 12 models you can see for each of these model, they have been predicted using a linear regression, their coefficients and standard errors and all these things are there. So, once we know this so, log of odds for individual classes can be represented as a linear regression, and we have seen these values of the coefficients.

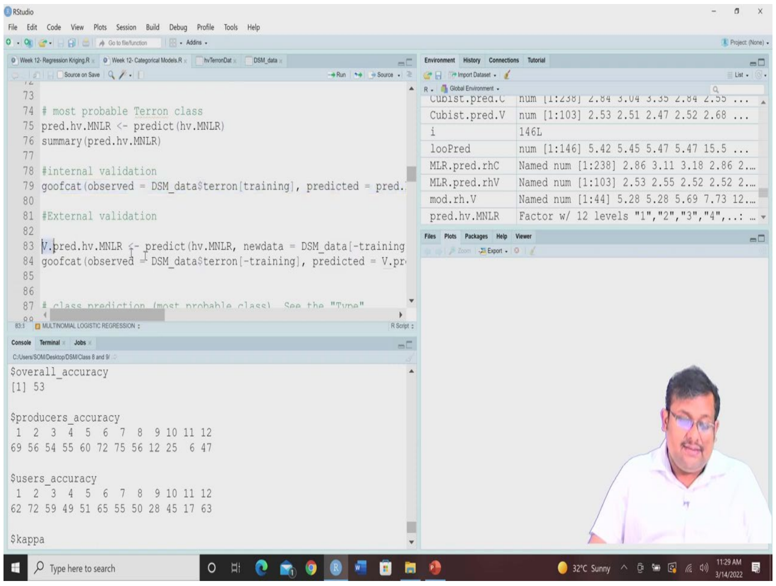
Now, once we have developed this model multinomial logistic regression Now, let us consider let us calculate the probabilities class probabilities. So, we are going to get these class probabilities and then once we calculate this class probabilities that is the probability of the class then let us see the first 6 or 8 on that top of data frame a poor probability.

So, you can see top of the data frame of the probabilities for 6 first for the 12 classes, you can see here the results and from there you can get the most probable Terron class also. So, for getting the most probable Terron class we are going to run this thing and then we are going to see the summary of the most probable Terron class.

So, the Terron class 1 appears 21 times, Terron class 2 appears 7 times and so on so forth. Now, once we have done that let us see the internal validation internal validation means calibration again we are going to use this goofcat function using this goofcat function our observed and predicted values we just fit it.

Now, you can see these goofcat function will show this is the confusion matrix and this is the overall accuracy, producer's accuracy and user's accuracy. So, you can get the values for each of these variables each of these classes each of these Terron classes using this goofcat function.

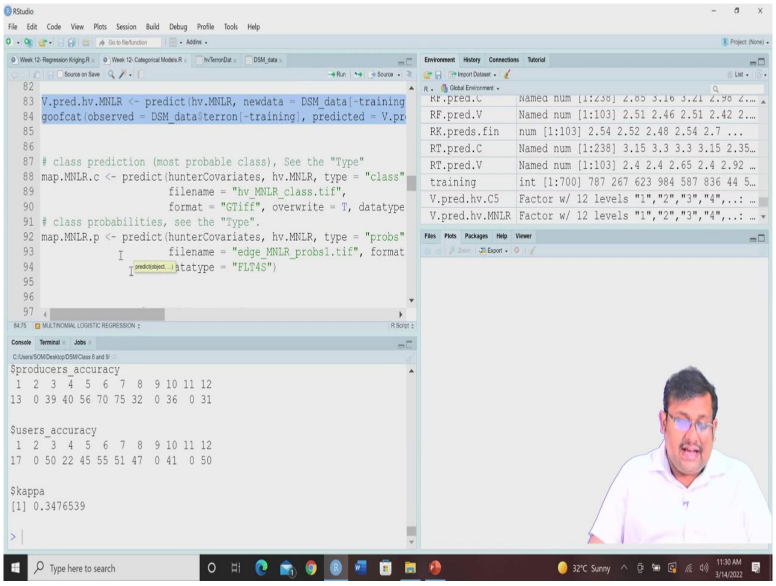
(Refer Slide Time: 25:20)



```r

# most probable Terron class
pred.hv.MNLR <- predict(hv.MNLR)
summary(pred.hv.MNLR)

#internal validation
goofcat(observed = DSM_data$terron[training], predicted = pred.

#External validation

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


# class prediction (most probable class), See the "Type"
```

```
$overall_accuracy
[1] 53

$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
69 56 54 55 60 72 75 56 12 25  6 47

$users_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
62 72 59 49 51 65 55 50 28 45 17 63

$kappa
```

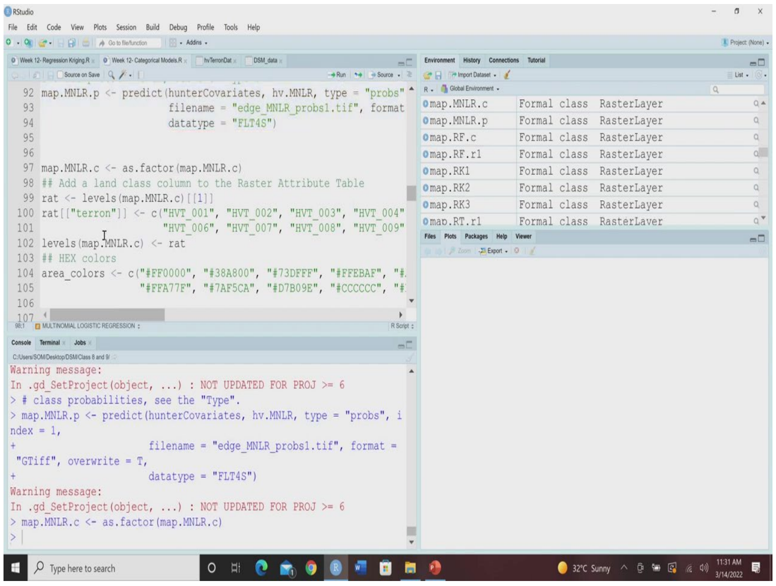
```
Cubist.pred.C     num [1:238] 2.84 3.04 3.35 2.84 2.55 ...
Cubist.pred.V     num [1:103] 2.53 2.51 2.47 2.52 2.68 ...
i                 146L
looPred           num [1:146] 5.42 5.45 5.47 5.47 15.5 ...
MLR.pred.rhC      Named num [1:238] 2.86 3.11 3.18 2.86 2....
MLR.pred.rhV      Named num [1:103] 2.53 2.55 2.52 2.52 2....
mod.rh.V          Named num [1:44] 5.28 5.28 5.69 7.73 12....
pred.hv.MNLR      Factor w/ 12 levels "1","2","3","4",..: ...
```



```r

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


# class prediction (most probable class), See the "Type"
map.MNLR.c <- predict(hunterCovariates, hv.MNLR, type = "class"
                      filename = "hv_MNLR_class.tif",
                      format = "GTiff", overwrite = T, datatype
# class probabilities, see the "Type".
map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs"
                      filename = "edge_MNLR_probs1.tif", format
                      datatype = "FLT4S")
```

```
$producers_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
13  0 39 40 56 70 75 32  0 36  0 31

$users_accuracy
 1  2  3  4  5  6  7  8  9 10 11 12
17  0 50 22 45 55 51 47  0 41  0 50

$kappa
[1] 0.3476539

>
```

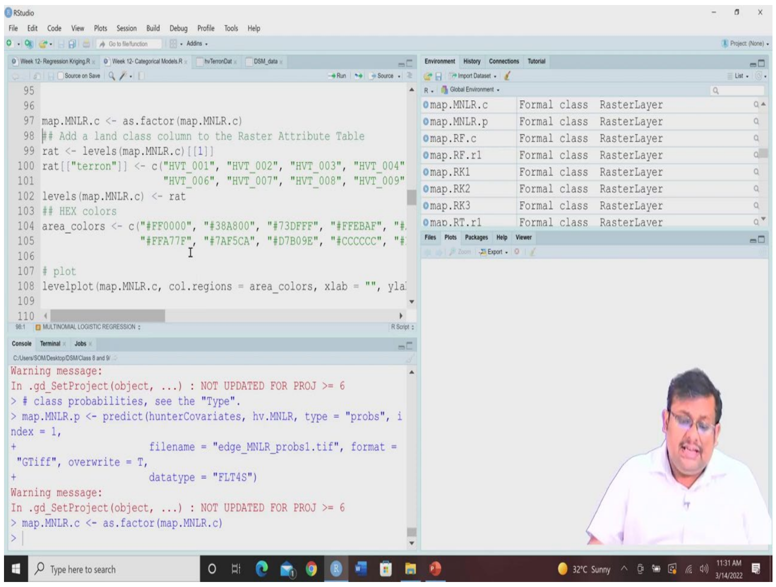
```
RF.pred.C         Named num [1:238] 2.85 3.16 3.21 2.98 2....
RF.pred.V         Named num [1:103] 2.51 2.46 2.51 2.42 2....
RK.preds.fin      num [1:103] 2.54 2.52 2.48 2.54 2.7 ...
RT.pred.C         Named num [1:238] 3.15 3.3 3.3 3.15 2.35...
RT.pred.V         Named num [1:103] 2.4 2.4 2.65 2.4 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",..: ...
V.pred.hv.MNLR    Factor w/ 12 levels "1","2","3","4",..: ...
```



```r
map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs"
                      filename = "edge_MNLR_probs1.tif", format
                      datatype = "FLT4S")


map.MNLR.c <- as.factor(map.MNLR.c)
## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004"
                     "HVT_006", "HVT_007", "HVT_008", "HVT_009"
levels(map.MNLR.c) <- rat
## HEX colors
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#.
                 "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#
```

```
Warning message:
In .gd_SetProject(object, ...) : NOT UPDATED FOR PROJ >= 6
> # class probabilities, see the "Type".
> map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs", i
ndex = 1,
+                       filename = "edge_MNLR_probs1.tif", format =
 "GTiff", overwrite = T,
+                       datatype = "FLT4S")
Warning message:
In .gd_SetProject(object, ...) : NOT UPDATED FOR PROJ >= 6
> map.MNLR.c <- as.factor(map.MNLR.c)
>
```

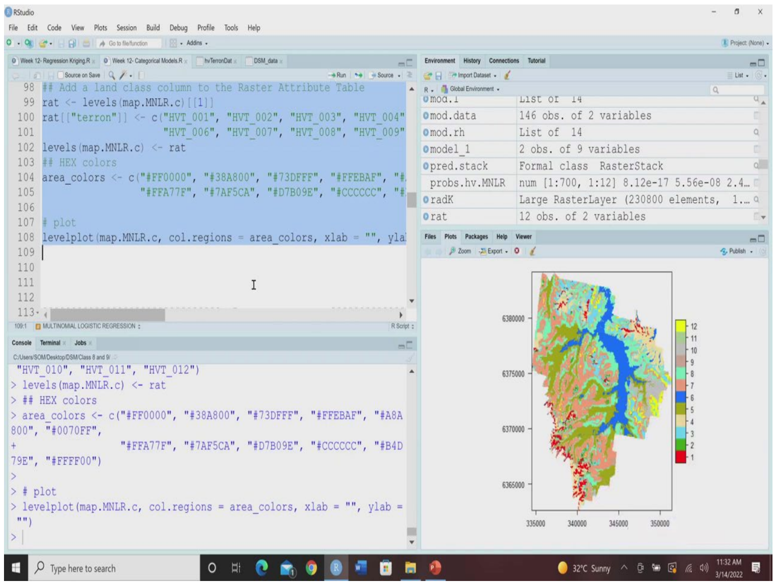
```
map.MNLR.c        Formal class  RasterLayer
map.MNLR.p        Formal class  RasterLayer
map.RF.c          Formal class  RasterLayer
map.RF.r1         Formal class  RasterLayer
map.RK1           Formal class  RasterLayer
map.RK2           Formal class  RasterLayer
map.RK3           Formal class  RasterLayer
map.RT.r1         Formal class  RasterLayer
```



```r
map.MNLR.c <- as.factor(map.MNLR.c)
## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004"
                     "HVT_006", "HVT_007", "HVT_008", "HVT_009"
levels(map.MNLR.c) <- rat
## HEX colors
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#
                 "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#

# plot
levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", yla
```

```
Warning message:
In .gd_SetProject(object, ...) : NOT UPDATED FOR PROJ >= 6
> # class probabilities, see the "Type".
> map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs", i
ndex = 1,
+                       filename = "edge_MNLR_probs1.tif", format =
 "GTiff", overwrite = T,
+                       datatype = "FLT4S")
Warning message:
In .gd_SetProject(object, ...) : NOT UPDATED FOR PROJ >= 6
> map.MNLR.c <- as.factor(map.MNLR.c)
> 
```

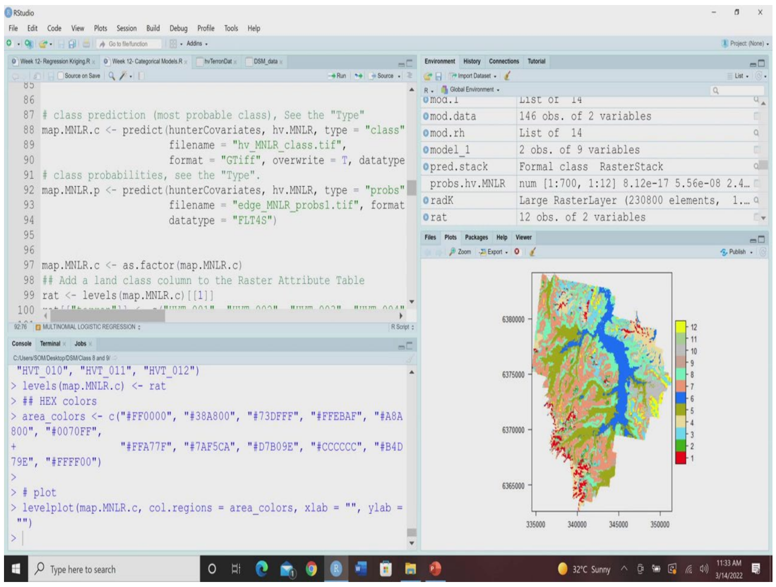
```
map.MNLR.c   Formal class  RasterLayer
map.MNLR.p   Formal class  RasterLayer
map.RF.c     Formal class  RasterLayer
map.RF.r1    Formal class  RasterLayer
map.RK1      Formal class  RasterLayer
map.RK2      Formal class  RasterLayer
map.RK3      Formal class  RasterLayer
map.RT.r1    Formal class  RasterLayer
```



```r
## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004"
                     "HVT_006", "HVT_007", "HVT_008", "HVT_009"
levels(map.MNLR.c) <- rat
## HEX colors
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#
                 "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#

# plot
levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", yla
```

```
 "HVT_010", "HVT_011", "HVT_012")
> levels(map.MNLR.c) <- rat
> ## HEX colors
> area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#A8A
800", "#0070FF",
+                  "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#B4D
79E", "#FFFF00")
>
> # plot
> levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", ylab =
 "")
> 
```

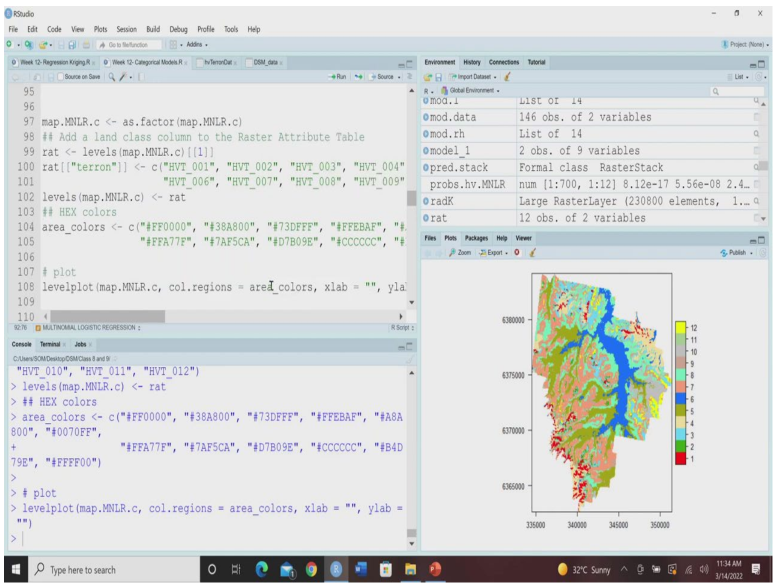
```
mod.l            List of  14
mod.data         146 obs. of 2 variables
mod.rh           List of  14
model_1          2 obs. of 9 variables
pred.stack       Formal class  RasterStack
probs.hv.MNLR    num [1:700, 1:12] 8.12e-17 5.56e-08 2.4...
radK             Large RasterLayer (230800 elements,  1...
rat              12 obs. of 2 variables
```



```r
# class prediction (most probable class), See the "Type"
map.MNLR.c <- predict(hunterCovariates, hv.MNLR, type = "class"
                      filename = "hv_MNLR_class.tif",
                      format = "GTiff", overwrite = T, datatype
# class probabilities, see the "Type".
map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs"
                      filename = "edge_MNLR_probs1.tif", format
                      datatype = "FLT4S")


map.MNLR.c <- as.factor(map.MNLR.c)
## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
```

```
 "HVT_010", "HVT_011", "HVT_012")
> levels(map.MNLR.c) <- rat
> ## HEX colors
> area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#A8A
800", "#0070FF",
+                  "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#B4D
79E", "#FFFF00")
>
> # plot
> levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", ylab =
 "")
> 
```

```
mod.l            List of  14
mod.data         146 obs. of 2 variables
mod.rh           List of  14
model_1          2 obs. of 9 variables
pred.stack       Formal class  RasterStack
probs.hv.MNLR    num [1:700, 1:12] 8.12e-17 5.56e-08 2.4...
radK             Large RasterLayer (230800 elements,  1...
rat              12 obs. of 2 variables
```



```
map.MNLR.c <- as.factor(map.MNLR.c)
## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004"
                     "HVT_006", "HVT_007", "HVT_008", "HVT_009"
levels(map.MNLR.c) <- rat
## HEX colors
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#
                 "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#

# plot
levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", yla
```

```
 "HVT_010", "HVT_011", "HVT_012")
> levels(map.MNLR.c) <- rat
> ## HEX colors
> area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#A8A
800", "#0070FF",
+                  "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#B4D
79E", "#FFFF00")
>
> # plot
> levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", ylab =
 "")
> |
```

```
mod.1          List of  14
mod.data       146 obs. of 2 variables
mod.rh         List of  14
model_1        2 obs. of 9 variables
pred.stack     Formal class  RasterStack
probs.hv.MNLR  num [1:700, 1:12] 8.12e-17 5.56e-08 2.4…
radK           Large RasterLayer (230800 elements,  1.…
rat            12 obs. of 2 variables
```

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
Source on Save
Run
Source
MULTINOMIAL LOGISTIC REGRESSION
R Script
Console Terminal Jobs
C:/Users/SOM/Desktop/DSM/Class 8 and 9/
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer
Zoom Export Publish
6380000
6375000
6370000
6365000
335000
340000
345000
350000
12
11
10
9
8
7
6
5
4
3
2
1
Type here to search
32°C Sunny
11:34 AM
3/14/2022

Now, let us go for the external validation. Similarly, using the validation data minus training dataset, you are going to get these results. So, here you can see that some of the classes are showing the 0 accuracy. So, that means, our model was not accurate to accurately predicted the probabilities for these classes specifically for Terron class 2 Terron class 9.

So, this is the inference from that Kappa coefficient is 0.34 and then let us see the class we can plot we can map the probabilities as well as most probable class. So, we can plot it the most probable class or we can map it. So, for my class a raster layer has been generated.

Now, once we have done that, there is a new thing that is you can use the add a land class column to the raster attribute table. So, here you can see that we are using a raster attribute table we can add a land class column that is called raster attribute table in the raster attribute table we are giving the different colors for each of these classes.

So, these colors are nothing but the hex colors, I request you to go ahead and see the hex color what are the hex color codes and how these hex color codes are important. So, here this followings script is very much important for producing the color map for showing this Terron and classes.

So, here for each of these Terron for 0 for, 0 0 1 to 0 0 0 0 2 0 1 2. So, for each of these Terron we have assigned a particular color using these hex color codes, and now we are going to see the plot. So, if we run it, let us see how it looks like. So, this is the multinomial logistic regression predicted classes.

So, you can see these are the Terrons 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and this is how we can see that the most probable class we can clearly see here. So, also here you can see when we are showing the we want to see the class probabilities. So, here in the first instance we are getting the most probable class in the second instance we are getting the class probability.

So, using here you can see we are using the type of argument we are using the probabilities and here you can see we have we can index any classes suppose, we want to see the probability of Terron class 1. So, we are giving these index 1 so, we can change this number from 1 to 12 and then we can see the probability of that class in the total area of interest.

So, here using these two sets of comments you can either produce the most probable class at the same time you can produce the map of class probabilities also. So, here in this map we are getting the most probable map most probable class. So, in this maps it is showing what are the most probable class in these different zones.

And here we are using these hex color codes and these hex color codes have been added to these individual classes. And you can go ahead and see the hex color codes and how it has been assigned you can gather more information and this is how we produce the map using this categorical model.

(Refer Slide Time: 29:51)

REFERENCE

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

Okay guys. So, I hope that you have you have got some good knowledge and this is again, this book should be read for more details about this multinomial logistic regression based mapping of soil properties. So, I Hope that you have gathered some useful knowledge. And let us meet in our next lecture, where we will see how to utilize the C 5 regression model as well as the random forest for categorical mapping in the digital soil mapping. So, thank you guys, let us meet in our next lecture.