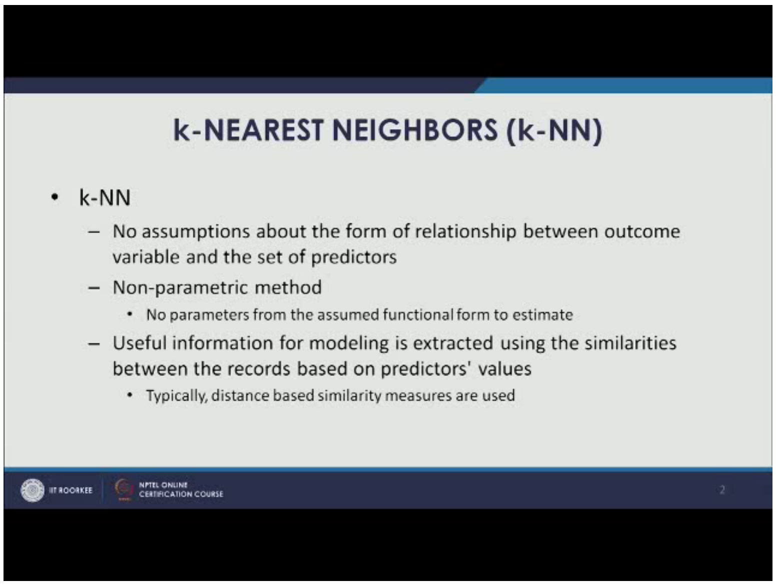
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture – 28
Machine Learning Technique K-Nearest Neighbors (K-NN)- Part I

Welcome to the course business analytics and data mining modeling using R. So, in the previous lecture, we concluded our discussion on multiple linear regression. So, in this particular lecture, we are going to start our discussion on the next technique K-nearest neighbors. So, this particular technique is more of a machine learning algorithm a data mining technique. The previous one the multiple linear regression was the statistical technique. So, let us start our discussion on k-nearest neighbors or k-NN.

So, as we have been discussing in previous lectures about the difference between statistical techniques and data mining techniques that in one specific difference being that in statistical techniques; we assume some form of relationship between the outcome variable of interest and the set of predictors; for example in linear regression multiple linear regression and other regression models typically, we assume that there is a linear relationship between the outcome variable of interest and set of vectors.

(Refer Slide Time: 01:16)



k-NEAREST NEIGHBORS (k-NN)

- k-NN
  - No assumptions about the form of relationship between outcome variable and the set of predictors
  - Non-parametric method
    - No parameters from the assumed functional form to estimate
  - Useful information for modeling is extracted using the similarities between the records based on predictors' values
    - Typically, distance based similarity measures are used

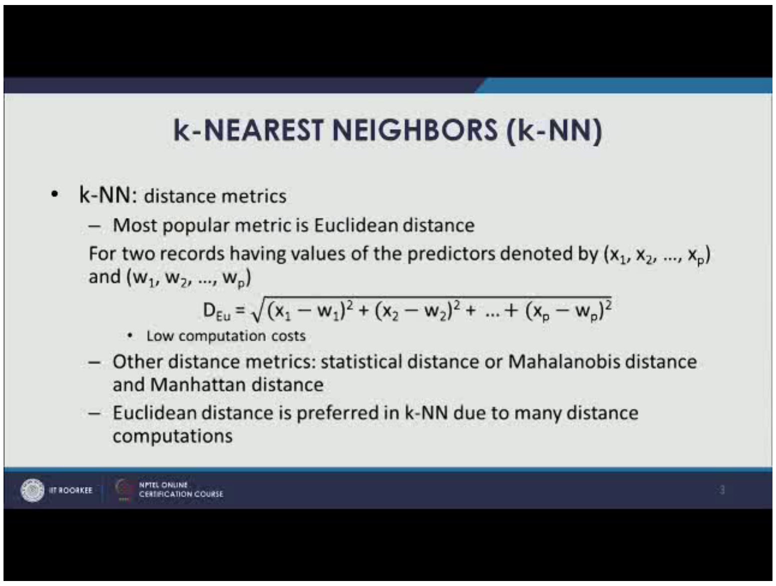
However in data mining techniques, as we have often mentioned that no such assumptions about the form of relationship between outcome variable of interest or set of predictors are made. So, therefore, first very first point in k-NN also that we have specified in this particular slide is that no junction about the form of relationship between outcome variable and the set of predictors.

Now, another important fact about k-NN and typically about other data mining techniques as well that nonparametric methods. So, k-NN is specifically nonparametric method. So, for example, in the multiple linear regression that we concluded in previous lectures we used to estimate the betas and the sigma value right. So, those we do not have such parameters to estimate. So, no parameters from the assumed functional form; so, those betas while coming from the linear relationships that we had assumed about the outcome variable and the predictors; so, those betas were required to estimate right. So, no such parameters especially from the; this functional form point on the functional form sense are there to be estimated right. So, this is being nonparametric method.

Then how do we build the model what the model is about right what we actually do in a k-NN modelling. So, what we actually do is we learn from the data. So, useful information from modeling is extracted using the similarities between the records based on predictors values. So, each row representing the records each row in the data set will represent a record and the different than the values are predictors for each record and for different records we can find out the similarities between those records and those similarities would become the basis of our modeling exercise.

So, typically how do we find out how do we measure the similarities between records. So, typically distance based similarity measures are used. So, let us move forward. So, as we said we learn from the data even though we learned from the data itself using statistical techniques as well, but we assume about some assume about the structure of the data or the relationship between variables right here we do not have such assumptions.

(Refer Slide Time: 04:32)



k-NEAREST NEIGHBORS (k-NN)

- k-NN: distance metrics
  - Most popular metric is Euclidean distance
    For two records having values of the predictors denoted by $(x_1, x_2, \ldots, x_p)$ and $(w_1, w_2, \ldots, w_p)$

    $$D_{Eu} = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2 + \ldots + (x_p - w_p)^2}$$

    - Low computation costs
  - Other distance metrics: statistical distance or Mahalanobis distance and Manhattan distance
  - Euclidean distance is preferred in k-NN due to many distance computations

IIT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE 3

So, let us discuss few more things for example, we talked about the similarity between records and specifically the distance based metrics distance based similarity metrics are used. So, one of the popular one; one of the popular metric that is used for distance is the Euclidean distance. So, for 2 records which are having values operators as denoted by x 1, x 2, x p, if there are p predictors and the second record being w 1, w 2 and w p; the distance between these 2 records as we all understand the Euclidean distance formula D E u is going to be square root of x 1 minus w 1 square plus x 2 minus w 2 square and plus up to x p minus w p is square. So, this is going to be the; this is the typical Euclidean distance formula that can actually use to compute the distance and therefore, the distance based similarity.

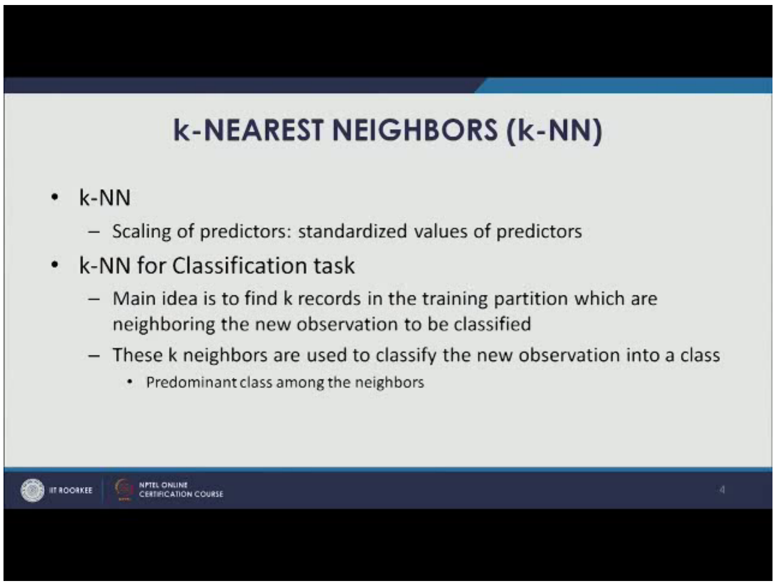
So, the main the main idea being that using the predictors information as the coordinates right for a particular record and therefore, the for 2 records to be considered as similar or considered to be closer if the distance is smaller distance of values predictors values between those 2 records are smaller than prop, then probably, we can say that those 2 records are similar or closer. So, using; so, these this particular distance based metric Euclidean distance metric is quite popular in k-NN the main reason being low computation cost as compared to other distance metric some of other distance metric matrix that could be used or statistical distance or Mahalanobis distance Manhattan distance. So, there are some other matrix also which can also be used.

So, some of the other matrix we would be covering in other discussions about you know discussing about other techniques other data mining techniques and statistical techniques. So, k-NN Euclidean distance is the most popular metric as we said especially for the reason being that low computation cost now we the main reason another region is that we need to compute more distances in k-NN and therefore, we would prefer a metric which is computationally less computationally intensive as expressed in this last line, in this particular slide Euclidean distance is preferred in k-NN due to many distance computations that we have to perform.

Now, another important aspect of k-NN is the scaling of predictors. So, as we talked about the distance metric that are going to be used. So, because of this the predictors set of predictors different predictors having different scales different units and because of that the easy because of just this scale some of the predictors might dominate the distance because if you look at the Euclidean distance formula it is just the way it is being computed the difference between predictors values of 2 points right. So, therefore, the particular difference of a particular difference of a you know of a predictor which is having higher scale might dominate the overall distance value and therefore, might dominate the results. So, therefore, scaling is very important in k-NN.

So, therefore, before we start our k-NN steps it is advisable to standardized values of predictors. So, how do we apply k-NN in a classification task? So, main idea is to find k records in the training partition. So, again here also we would be doing that partitioning training validation and test partitioning and the training partitions are records belonging to the training partition what then we used to find k records which are neighboring the new observation to be classified.

(Refer Slide Time: 09:05)



k-NEAREST NEIGHBORS (k-NN)

- k-NN
  - Scaling of predictors: standardized values of predictors
- k-NN for Classification task
  - Main idea is to find k records in the training partition which are neighboring the new observation to be classified
  - These k neighbors are used to classify the new observation into a class
    - Predominant class among the neighbors

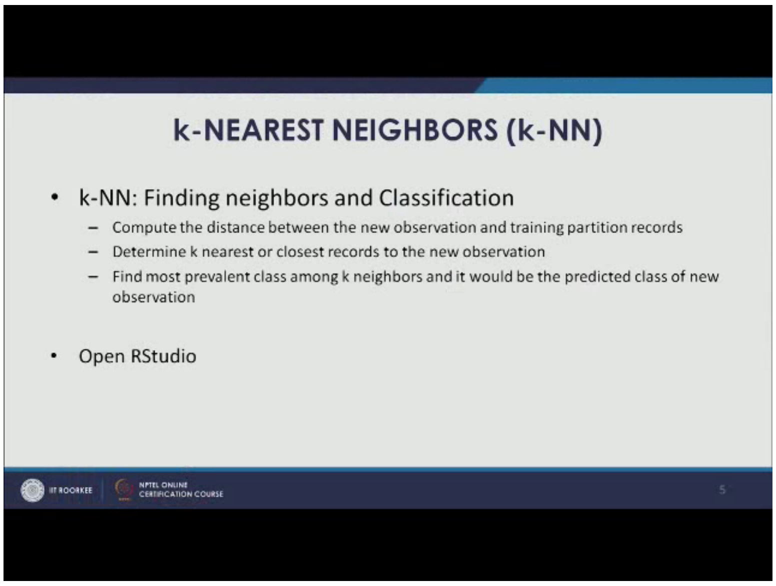
IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
4

So, the new observation could be in the validation partition or if we have the third partition the test partition then this new observation could be or it could be a completely new observation as well.

So, new observation could be in the validation partition or test partition and we need to find the k records from the training partition which are closer to this which are neighboring to this new observation closer to this observation or similar to this observation or neighboring this new observation now these k neighbors are then used to classify the new observation right into a class. So, typically the predominant class among the neighbors that is generally assigned as the class of new observation right. So, these are the 2 main steps for k-NN especially for the classification task.

So, let us move forward now the next important point would be finding neighbors and then performing this classification.

(Refer Slide Time: 10:18)



k-NEAREST NEIGHBORS (k-NN)

- k-NN: Finding neighbors and Classification
  - Compute the distance between the new observation and training partition records
  - Determine k nearest or closest records to the new observation
  - Find most prevalent class among k neighbors and it would be the predicted class of new observation
- Open RStudio

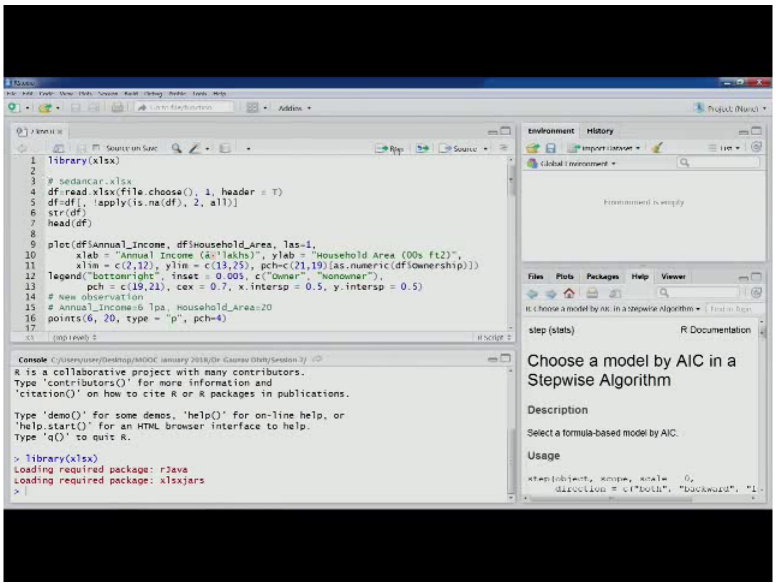
So, if we look at steps for this for the same first we need to compute the distance between the new observation and training partition records. So, that is why when we talk about why Euclidean distance metric would be preferred in is generally preferred in k-NN technique k-nearest neighbor technique the main reason being as expressed in the first point we need to compute the distance between the new observation that we want to classify the new record that we want to classify and training partition records.

So, if we are; we have a large number of observations in our sample and therefore, large number of observation in our training partitions. So, the number of distance computations that we will have to perform would be would be much more in comparison to other scenarios therefore, other distance metrics that we talked about the statistical distance or also called Mahalanobis distance and we also mentioned one more name Manhattan distance. So, those metrics are not actually used then once the computation once the distance between the new observation and the training partition records once the distance has been computed, then we can go ahead with the point number 2 that is to determine K-nearest or closest records to the new observation.

So, out of all the records in the training partition and right. So, the distances have been computed for all of them. So, out of all those distance values we can find out the K-nearest or k closest records to the new observation now within these now this will also. So, the next step would also be that find most prevalent class among k neighbors. So, once k neighbors have been identified in a step number 2 we can find out the most prevalent class among k neighbors and once this is done the this particular class the class most prevalent the majority rule decision would take place here. So, the among all the nam all k neighbors the class which is more common which is the majority class among those k neighbors that would be the predicted class of new observation. So, that is the typical procedure typical steps that we apply in that we apply in k-NN modeling for classification tasks.

So, let us understand some of these concepts using an exercise in R; let us open R studio.

(Refer Slide Time: 13:10)



```r
library(xlsx)

# sedancar.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
str(df)
head(df)

plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
>
```

step {stats} R Documentation

Choose a model by AIC in a Stepwise Algorithm

Description

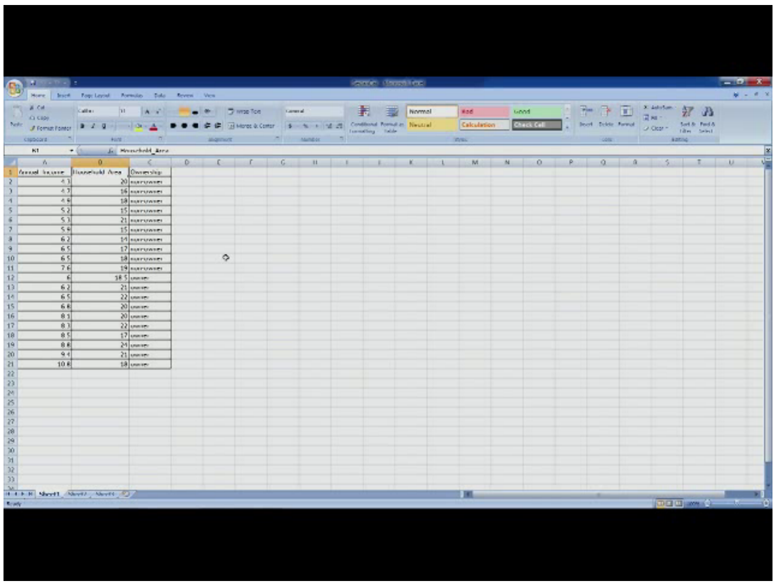
Select a formula-based model by AIC.

Usage

```
step(object, scope, scale = 0,
     direction = c("both", "backward", "
```

So, first as for as first step, as we generally do typically do that first we need to load this particle library x l s x. So, that we are able to import the data sets available in an excel file. So, the data set that we are going to use here is this sedan car dot x l s x file the sedan car data set.

(Refer Slide Time: 13:38)



Annual_Income	Household_Area	Ownership
4.3	20	nonowner
4.7	16	nonowner
4.8	18	nonowner
5.2	15	nonowner
5.3	21	nonowner
5.8	15	nonowner
6.2	14	nonowner
6.5	17	nonowner
6.5	18	nonowner
7.6	19	nonowner
6	18.5	owner
6.2	21	owner
6.5	22	owner
6.8	20	owner
8.1	20	owner
8.3	22	owner
8.5	17	owner
8.8	24	owner
9.4	21	owner
10.8	18	owner

So, let us look at the in the data set first. So, this is the one. So, this particular data set we have used in the in previous lectures as well. So, in this particular data set we have three variables mainly annual income and household area and ownership annual income and household area being the numeric numerical variable and ownership being the categorical variable.

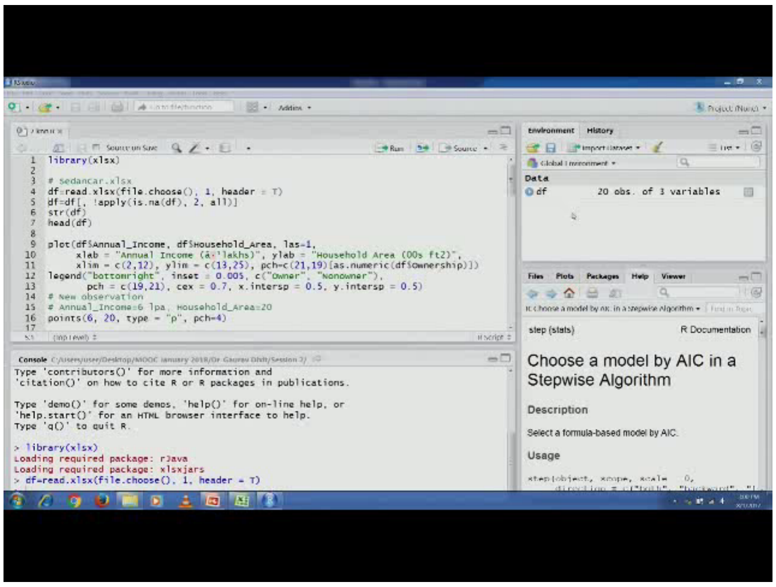
So, for our classification tasks that we shall be performing ownership would be the categorical outcome variable. So, we would be using the 2 variables annual income and household area to classify the ownership of a particular ownership of observation right. So, for different; 2 different classes are there non owner or owner. So, whether a particular household. So, a unit of analysis here is the household. So, for different households we have the information about those household as annual income of those households and the area of those households and then whether they own is whether the own sedan car or not.

So, typically we would expect that the households which are having higher annual income they are expected to own a sedan car similarly household which have bigger which have larger household area they are also expected to own a sedan car because you would need more space in your household to park to make the parking arrangement of your car. So, these 2 variables numerical variables are going to be used to classify or to build our model k-NN model specifically classifier model.

So, let us import this data set. So, the function once we have loaded the library this x l s x the read dot x l s x function would be available for us. So, we can call this function and as we have been doing in the previous lecture the file dot choose function would allow us to browse this particular excel file; other ways of importing or mentioning this particular file that we have discussed in the supplementary lectures as well that you can mention the absolute path of the file name or you can change your working directory as using the set WDA function and then once that is done and if your file the excel file is has been copied or stored in that working directory then you can use the name of that particular file that is sedan car x l s x in the first argument and it would be easily imported.

So, different ways of importing a particular file. So, let us execute this code.

(Refer Slide Time: 16:47)



```r
library(xlsx)

# sedancar.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
str(df)
head(df)

plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â‚¹lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)
```

```
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> df=read.xlsx(file.choose(), 1, header = T)
```

```
Data
df          20 obs. of 3 variables
```

step (stats) R Documentation

Choose a model by AIC in a Stepwise Algorithm

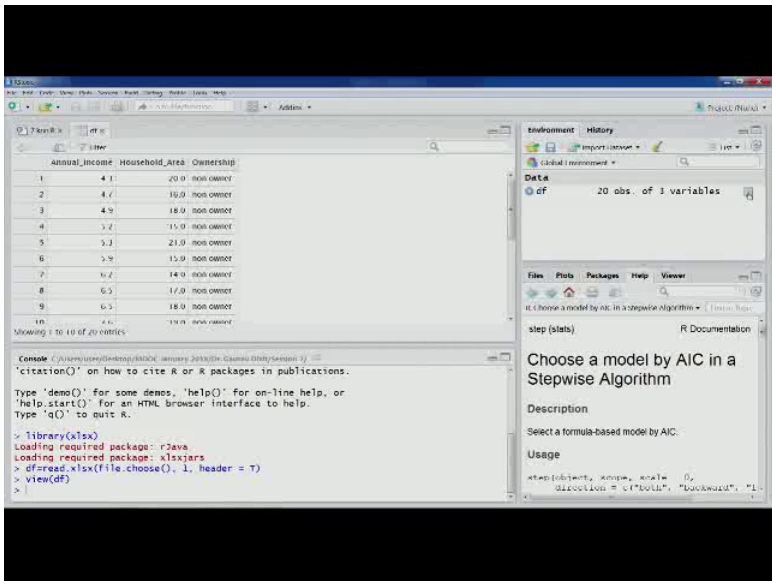
Description

Select a formula-based model by AIC.

Usage

And you can see in the environment section we have imported that is data set. So, this has been stored in. So, we have used the data frame d f and then twenty observation of three variables if you want to confirm the data that we had in the excel file you can just click on this icon.

(Refer Slide Time: 17:04)



	Annual_Income	Household_Area	Ownership
1	4.1	20.0	non owner
2	4.7	16.0	non owner
3	4.9	18.0	non owner
4	5.2	15.0	non owner
5	5.3	21.0	non owner
6	5.9	15.0	non owner
7	6.2	14.0	non owner
8	6.5	17.0	non owner
9	6.5	18.0	non owner

Showing 1 to 10 of 20 entries

```
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> df=read.xlsx(file.choose(), 1, header = T)
> view(df)
> 
```

```
Data
df        20 obs. of 3 variables
```

step {stats}   R Documentation

Choose a model by AIC in a Stepwise Algorithm

Description

Select a formula-based model by AIC.

Usage

This is small icon that is that appears in the in the environment section and then the data sub section this is small worksheet kind of icon. So, you can immediately see this that the annual income household area immediately you would be able to you do you; this particular file and this particular information.

You see another important aspect of about this R environment or R studio environment is a is that once we import a particular dataset, it would be loaded into memory and therefore, immediately whenever we click on this particular small icon or whenever we want to access the full data or few observation or few rows the output is produced quite quickly in comparison to when we open the excel file, it takes a bit more processing and takes a bit more time for especially for larger excel file to load and therefore, that being the main difference. So, once we import the data set is loaded into memory and therefore, it can be easily accessed in a much faster or quicker way.

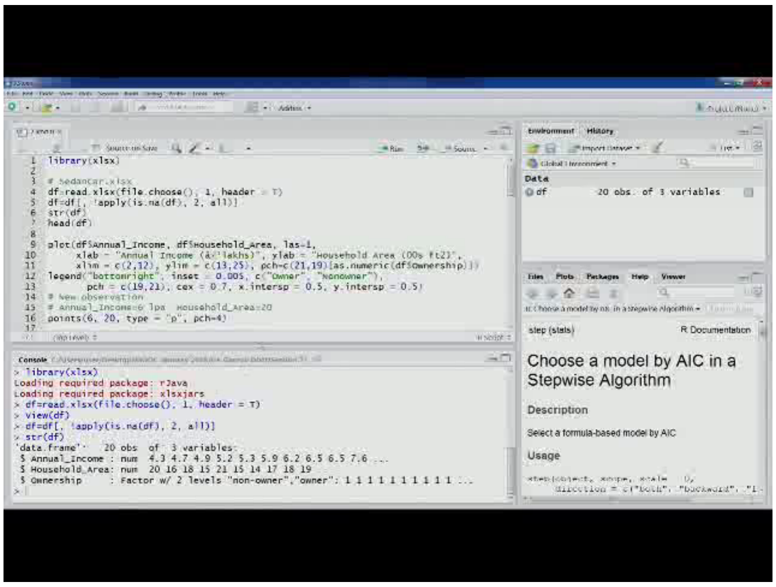
So, the next line is about deleting the removing the na columns. So, once we have data set in excel files and if we happen to delete some of the irrelevant columns in those excel files. So, those deleted columns would be picked up as variables na variables; variables having na values in R environment. So, therefore, if we had such columns and then the imported data set will have something like twenty observation of five variables if we had 2 columns which were deleted irrelevant columns which were deleted in the excel file.

So, to get rid of those situations we have this next line where as you can see in the column value.

So, for the data frame as we have talked about in the supplementary lectures that the first one is in the brackets within the brackets first value is for the rows and the second value is for the columns within the second you know for the second value we are applying this apply function where we are first we are applying this is dot na function whether a particular and then the second argument is the margin 2 so; that means, that the function that is all specified in the third argument is going to be applied column wise on different variables of this particular data frame right. So, if data frame is having na values. So, that would be appropriately written and then those columns would be appropriately subsetted using these brackets.

So, let us get rid of these any columns now let us look at the structure of this particular data set.

(Refer Slide Time: 20:10)



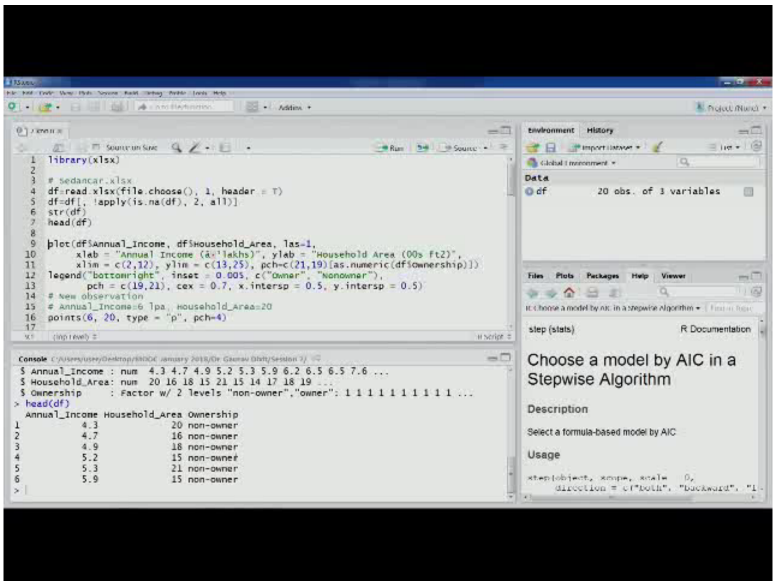
```r
library(xlsx)

# SedanCar.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
str(df)
head(df)

plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â‚¹lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa  Household_Area=20
points(6, 20, type = "p", pch=4)
```

```
> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> df=read.xlsx(file.choose(), 1, header = T)
> View(df)
> df=df[, !apply(is.na(df), 2, all)]
> str(df)
'data.frame':   20 obs. of  3 variables:
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
>
```

Environment History
Data
df   20 obs. of 3 variables
step {stats}   R Documentation
Choose a model by AIC in a Stepwise Algorithm
Description
Select a formula-based model by AIC
Usage

So, as you can see using the str function the structure function, we get to know about these three variables and we will come and householder area 2 numerical variables then the ownership the cat categorical variable or factor variable as it is called in R environment with 2 labels non owner and owner.

(Refer Slide Time: 20:32)

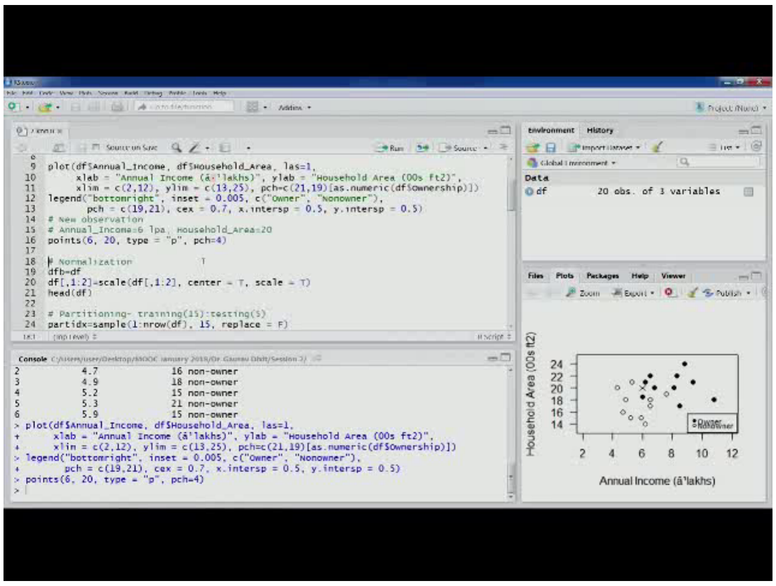


```
library(xlsx)

# Sedancar.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
str(df)
head(df)

plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)
```

```
 $ Annual_Income : num  4.3 4.7 4.9 5.2 5.3 5.9 6.2 6.5 6.5 7.6 ...
 $ Household_Area: num  20 16 18 15 21 15 14 17 18 19 ...
 $ Ownership     : Factor w/ 2 levels "non-owner","owner": 1 1 1 1 1 1 1 1 1 1 ...
> head(df)
  Annual_Income Household_Area Ownership
1           4.3             20 non-owner
2           4.7             16 non-owner
3           4.9             18 non-owner
4           5.2             15 non-owner
5           5.3             21 non-owner
6           5.9             15 non-owner
>
```

Now, let us look at the first 6 observations. So, this is how it is now one of the first few things that we do in R is in our modeling exercise that we plot a we generate a scatter plot between the predictors or important variables of the model. So, in this particular ca case we have a 2 variables and income and household area and you would see that ranges have been appropriately specified as we have been using this particular data set in previous lectures as well.

(Refer Slide Time: 21:14)



```
plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)

# Normalization
dfb=df
df[,1:2]=scale(df[,1:2], center = T, scale = T)
head(df)

# Partitioning- training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
```

```
2           4.7             16 non-owner
3           4.9             18 non-owner
4           5.2             15 non-owner
5           5.3             21 non-owner
6           5.9             15 non-owner
> plot(df$Annual_Income, df$Household_Area, las=1,
+      xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
+      xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
> legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
+        pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
> points(6, 20, type = "p", pch=4)
>
```

So, let us create the generate this scatter plot let us also generate the legend for the same. So, these are the points. So, this different-different characters are being used to denote the owner or non owner class among these points.

So, on x axis we have annual income and on y axis we have household area. So, we would like to classify these observation into belonging owner or non owner category as we can see this is particular small data set all very already we can see a clear class separation between these points; however, we would like to do same the classification task using a modeling.

So, for example, if we have a new observation let us say annual income 6 l p a. This is the new observation and will a particular household having annual income as 6 l p a and household area as 20,000 the square feet right 22,000 square feet because we have the household area in the hundreds of square feet. So, this observation if you would like to lets also plot this observation using the point function. So, point function can actually be used to plot points or add points in an existing plot the current plot.

So, you can see the type is the argument that we are specifying. So, its p right; so, on the plotting character is also been selected differently different plotting characterise has been selected. So, this point would be added in into the existing plot the idea; idea is that to understand that we want to classify this is the new observation. So, this particular observation we want to apply K-NN and you would like to classify it into one of the 2 classes owner or non owner.

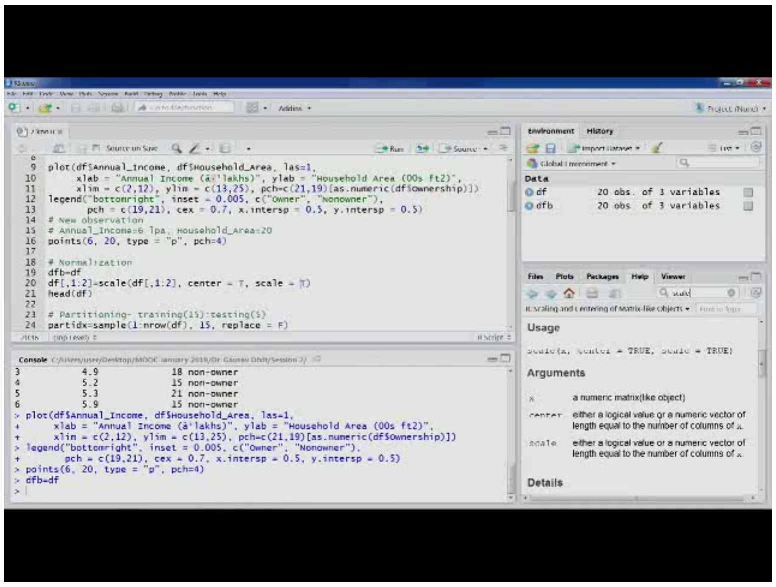
Now, as we have discussed that scaling is an important aspect of modeling using K-NN because these scales different scales that could be there for different predictors and that could actually. So, some of those predictors having higher scale could actually dominate the distance metric that we are going to use the Euclidean distance metric that we use. So, therefore, they can dominate the results and then influence the results and the we would end of these end up with the meaningless classifications. So, let us before proceeding further; let us normalize the values standardize the values. So, let us take a backup of this particular data set.

Now, as you can see we have selected just 2 columns for annual income and household area column one and 2 and these being the numerical variables. So, we the scaling has is to be applied on only these 2 variables. So, you can see we are using a scale function. So, scale function can be used to apply different kind of a scaling.

So, for example, in this particular case the first argument is the variables as subsetted using this from this data frame alright. So, we would be applying a scaling on these variables 2 variables and then we would see the second argument is about centralization of these values. So, which has been specified as true then the scaling of these values which has been specified as true.

So, centralization and scaling both would be performed using this particular function.

(Refer Slide Time: 25:05)



```r
plot(df$Annual_Income, df$Household_Area, las=1,
     xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)

# Normalization
dfb=df
df[,1:2]=scale(df[,1:2], center = T, scale = T)
head(df)

# Partitioning- training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
```

```
3          4.9             18 non-owner
4          5.2             15 non-owner
5          5.3             21 non-owner
6          5.9             15 non-owner
> plot(df$Annual_Income, df$Household_Area, las=1,
+      xlab = "Annual Income (â'lakhs)", ylab = "Household Area (00s ft2)",
+      xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
> legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
+        pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
> points(6, 20, type = "p", pch=4)
> dfb=df
>
```

```
Data
df      20 obs. of 3 variables
dfb     20 obs  of 3 variables
```

Scaling and Centering of Matrix-like Objects

Usage

```r
scale(x, center = TRUE, scale = TRUE)
```

Arguments

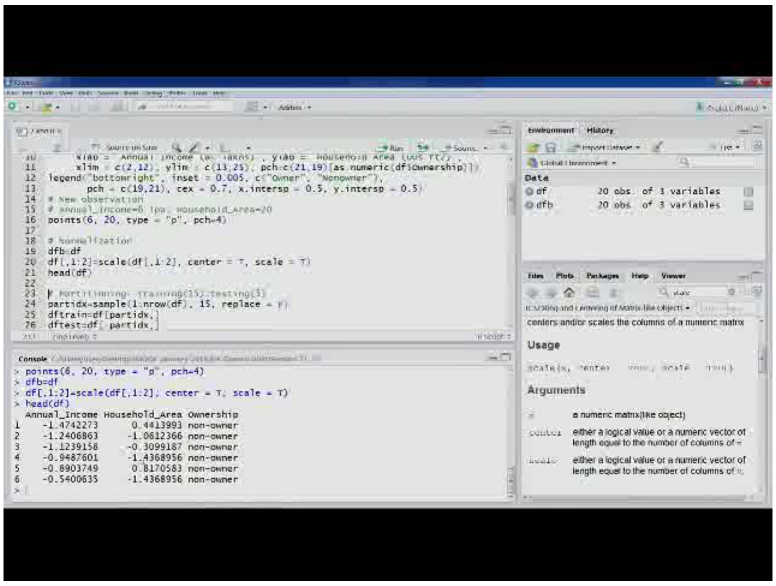
x	a numeric matrix(like object)
center	either a logical value or a numeric vector of length equal to the number of columns of x.
scale	either a logical value or a numeric vector of length equal to the number of columns of x.

Details

If you are understand and finding out more information on this particular function you can type scale in the help section and you look at more information on this particular very function you can see scale is used for scaling and centering of matrix like objects a scale is generate the function and whose default method centers and all scales the columns of a numeric matrix right you can see center either a logical value or numeric vector of length equal to the number of columns of x.

So, different types of standardization or normalization scaling could actually be performed using this plot form function in the coming lectures we would be trying different variations of the same. So, let us perform this scaling here let us do this code.

(Refer Slide Time: 25:58)



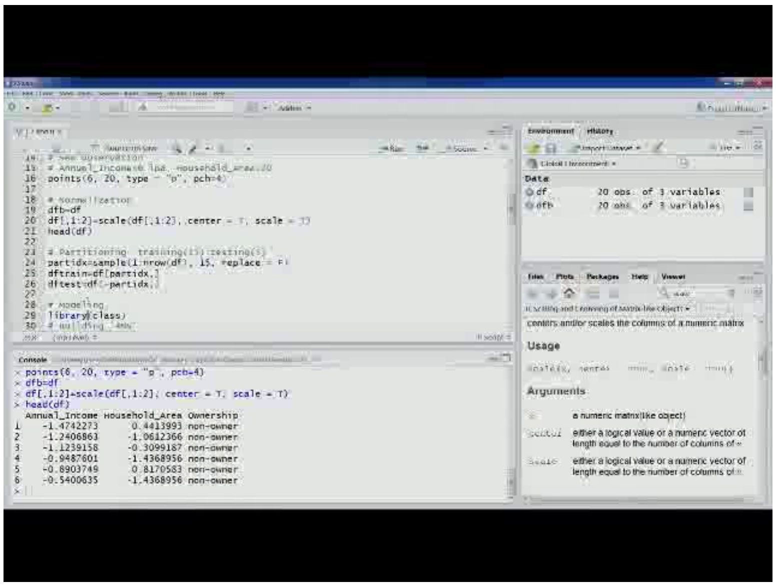
```r
     xlab = "Annual Income (in lakhs)", ylab = "Household Area (00s ft2)",
     xlim = c(2,12), ylim = c(13,25), pch=c(21,19)[as.numeric(df$Ownership)])
legend("bottomright", inset = 0.005, c("Owner", "Nonowner"),
       pch = c(19,21), cex = 0.7, x.intersp = 0.5, y.intersp = 0.5)
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)

# Normalization
dfb=df
df[,1:2]=scale(df[,1:2], center = T, scale = T)
head(df)

# Partitioning: training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]
```

```
> points(6, 20, type = "p", pch=4)
> dfb=df
> df[,1:2]=scale(df[,1:2], center = T, scale = T)
> head(df)
  Annual_Income Household_Area Ownership
1    -1.4742273      0.4413993 non-owner
2    -1.2406863     -1.0612366 non-owner
3    -1.1239158     -0.3099187 non-owner
4    -0.9487601     -1.4368956 non-owner
5    -0.8903749      0.8170583 non-owner
6    -0.5400635     -1.4368956 non-owner
>
```

```
df     20 obs. of 3 variables
dfb    20 obs. of 3 variables
```

Now, let us look at the first 6 observation as you can see after the scaling has been applied the values have changed right. So, these values have change. So, earlier the scale was different now the scaling has been done so that to eliminate the influence of the same.

(Refer Slide Time: 26:20)



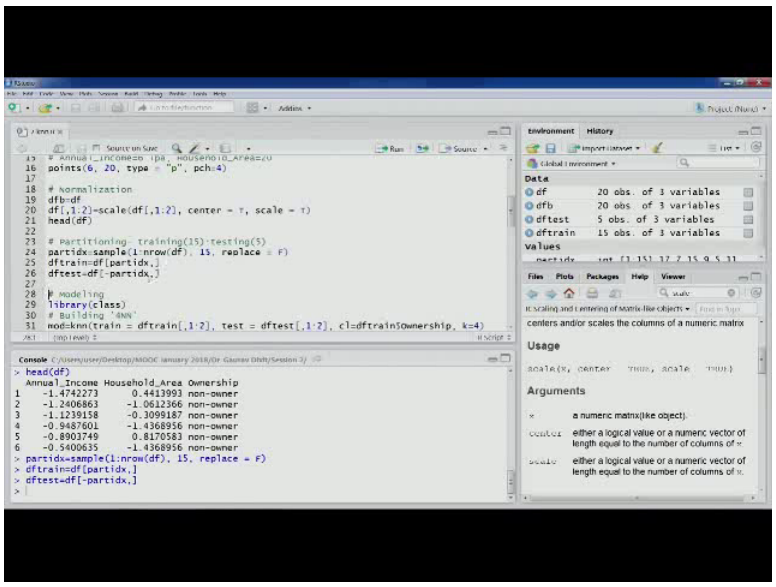
```r
# New observation
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)

# Normalization
dfb=df
df[,1:2]=scale(df[,1:2], center = T, scale = T)
head(df)

# Partitioning: training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]

# Modeling
library(class)
# building kNN
```

```
> points(6, 20, type = "p", pch=4)
> dfb=df
> df[,1:2]=scale(df[,1:2], center = T, scale = T)
> head(df)
  Annual_Income Household_Area Ownership
1    -1.4742273      0.4413993 non-owner
2    -1.2406863     -1.0612366 non-owner
3    -1.1239158     -0.3099187 non-owner
4    -0.9487601     -1.4368956 non-owner
5    -0.8903749      0.8170583 non-owner
6    -0.5400635     -1.4368956 non-owner
>
```

Now, let us do the partitioning. So, the data set that we have is mainly for the illustration purpose and we are having very small data set. So, therefore, the results might not be that stable therefore, it is generally recommended to have a larger data set and the data mining modeling is actually applied on largest data set the process is suitable for the larger data set where we can create you know partitions having enough number of observation.

So, that the results could be stable right and could be reliable. So, in this case because this is for the illustration purpose we are going to partition this particular data set we have twenty observations twenty values and of 15 values are going to be used in the training partition and the five values five records are going to be used in the test partitions. So, partitioning let us do the partitioning here and then lets clear the training partition using the index that we have just generated.

(Refer Slide Time: 27:27)



```
# Annual_Income=6 lpa, Household_Area=20
points(6, 20, type = "p", pch=4)

# Normalization
dfb=df
df[,1:2]=scale(df[,1:2], center = T, scale = T)
head(df)

# Partitioning- training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]

# Modeling
library(class)
# Building '4NN'
mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl=dftrain$Ownership, k=4)
```

```
> head(df)
  Annual_Income Household_Area Ownership
1    -1.4742273      0.4413993 non-owner
2    -1.2406863     -1.0612366 non-owner
3    -1.1239158     -0.3099187 non-owner
4    -0.9487601     -1.4368956 non-owner
5    -0.8903749      0.8170583 non-owner
6    -0.5400635     -1.4368956 non-owner
> partidx=sample(1:nrow(df), 15, replace = F)
> dftrain=df[partidx,]
> dftest=df[-partidx,]
>
```

```
Data
df        20 obs. of 3 variables
dfb       20 obs. of 3 variables
dftest    5 obs. of 3 variables
dftrain   15 obs. of 3 variables
Values
```

Scaling and Centering of Matrix-like Objects

centers and/or scales the columns of a numeric matrix

Usage

```
scale(x, center = TRUE, scale = TRUE)
```

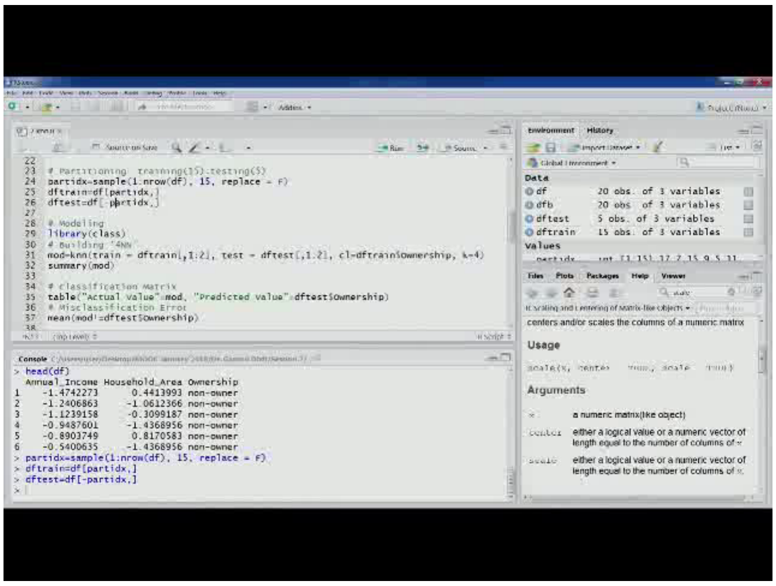
Arguments

x	a numeric matrix(like object).
center	either a logical value or a numeric vector of length equal to the number of columns of x.
scale	either a logical value or a numeric vector of length equal to the number of columns of x.

So, let us create this in the test partition as we have been talking about in the previous lecture also sample function can be used to create the indices can actually allow you to select the random and indices belonging to different observations different observation in your sample in this case you can see that second argument is specified as 15 because we want to draw 15 indexes we want to randomly select 15 indices to create the partition right the placement is also this is this process is done without replacement.

So, the first partition that is the training partition; so, all these 15 indices and the appropriate sub setting of the data frame d f is performed in the second partition that test partition the remaining five observation of the 20 total that would be left for the testing partition now once this partitioning has been done.

(Refer Slide Time: 28:28)



```
# Partitioning  training(15):testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]

# Modeling
library(class)
# Building '4NN'
mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl=dftrain$Ownership, k=4)
summary(mod)

# Classification Matrix
table("Actual Value"=mod, "Predicted Value"=dftest$Ownership)
# Misclassification Error
mean(mod!=dftest$Ownership)
```

```
> head(df)
  Annual_Income Household_Area Ownership
1    -1.4742273      0.4413993 non-owner
2    -1.2406863     -1.0612366 non-owner
3    -1.1239158     -0.3099187 non-owner
4    -0.9487601     -1.4368956 non-owner
5    -0.8903749      0.8170583 non-owner
6    -0.5400635     -1.4368956 non-owner
> partidx=sample(1:nrow(df), 15, replace = F)
> dftrain=df[partidx,]
> dftest=df[-partidx,]
>
```

```
Data
df        20 obs. of 3 variables
dfb       20 obs. of 3 variables
dftest    5 obs. of 3 variables
dftrain   15 obs. of 3 variables
Values
partidx   int [1:15] 17 7 15 9 5 11
```

Scaling and Centering of Matrix-like Objects

centers and/or scales the columns of a numeric matrix

Usage

```
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	a numeric matrix(like object)
center	either a logical value or a numeric vector of length equal to the number of columns of x
scale	either a logical value or a numeric vector of length equal to the number of columns of x.

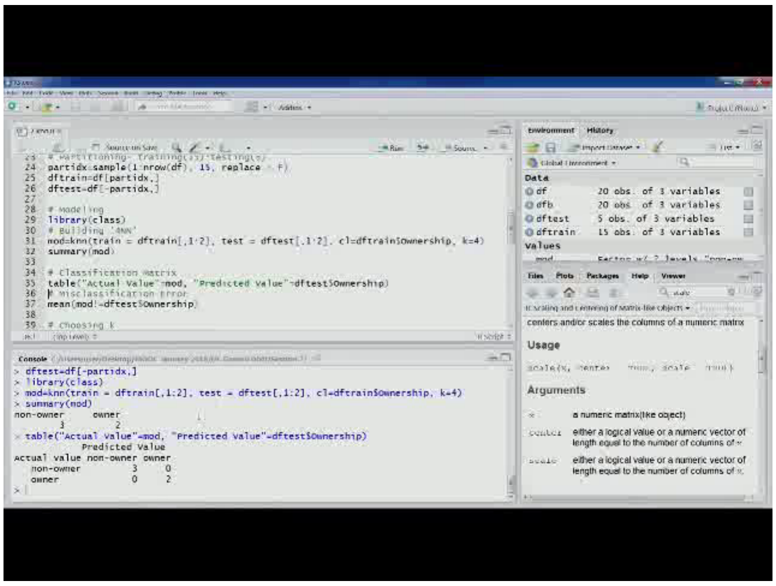
We can start with our modeling process. So, as you can see it library class is actually used to perform the k-NN modelling. So, let us load this particular library and we will do. So, there is another important aspect of k-NN is the value of k. Till now, we have been using k as our reference, but k actually signify the number of neighbors number of neighboring values that are going to be used to perform computations, right.

So, we will start with an example of 4 n n; that means, value k value of k is being taken as four, but later on we will see how the value appropriate value of k optimal value of k can be selected right. So, let us apply lets execute this code. So, the next function that we are going to use for k-NN modeling is the k-NN function.

So, where in the first argument is that the training data set the training partition? So, that we have appropriately specified second argument is the test partition. So, that we have again specified you can see only the numerical variables are being used and then because this is a classification task you can see the third argument that is CL. So, there we are specifying the different classification different you know using the third variable in the training partition that is ownership. So, from this using this argument the classification would be used by the k-NN model.

Now, we have also specified the value of k s 4. So, let us execute this record and we would see.

(Refer Slide Time: 30:12)



```
# Partitioning- Training(15)-Testing(5)
partidx=sample(1:nrow(df), 15, replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]

# Modeling
library(class)
# Building '4NN'
mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl=dftrain$Ownership, k=4)
summary(mod)

# Classification matrix
table("Actual Value"=mod, "Predicted Value"=dftest$Ownership)
# misclassification error
mean(mod!=dftest$Ownership)

# Choosing k
```

```
> dftest=df[-partidx,]
> library(class)
> mod=knn(train = dftrain[,1:2], test = dftest[,1:2], cl=dftrain$Ownership, k=4)
> summary(mod)
non-owner     owner
        3         2
> table("Actual Value"=mod, "Predicted Value"=dftest$Ownership)
            Predicted Value
Actual value non-owner owner
   non-owner         3     0
   owner             0     2
>
```

```
Data
df         20 obs. of 3 variables
dfb        20 obs. of 3 variables
dftest     5 obs. of 3 variables
dftrain    15 obs. of 3 variables
Values
mod        Factor w/ 2 levels "non-own
```

centers and/or scales the columns of a numeric matrix

Usage

`scale(x, center = TRUE, scale = TRUE)`

Arguments

x	a numeric matrix(like object)
center	either a logical value or a numeric vector of length equal to the number of columns of x
scale	either a logical value or a numeric vector of length equal to the number of columns of x.

Let us execute on the summary function and in summary you would see that out of out of five observations that we had in the test partition right. So, three have been classified as non owner and 2 have been classified as owner here we are interested in looking at the classification matrix. So, we can use this table function.

So, in the table function first argument is always about the actual values and then the second argument is for the predicted values right. So, let us use this. So, you can see all the values have been the mod the model that we build and the classification that we have got as we can see from the classification matrix all three owners have been collects all three non owners have been correctly classified as non owners and 2 owners have been correctly classified as owner.

So, if we look at the misclassification error. So, this we can compute using the mean function where we are equating the actual values with the predicted values if anything is not equal. So, that would be counted and an average would be taken right. So, that will get the misclassification error. So, we would see that misclassification error in this case is 0; so, as because this was a small data set. So, the results that we have they depend on the partition that we do the observation that are selected in the training partition and the observation that are left over for the testing partition.

So, in this particular case we got the 100 percent accuracy and 0 percent error, but if you repeat the exercise if you again do the partitioning because this being a small data set the results might change significantly; however, as we have discussed in the previous lectures a larger data set is generally recommended. So, that our results remain stable this even if we repeat the exercise.

So, I will stop here and in the next session we will start our discussion on how to how to select a select a suitable or a appropriate and optimal value of k for the k-NN modeling.

Thank you.