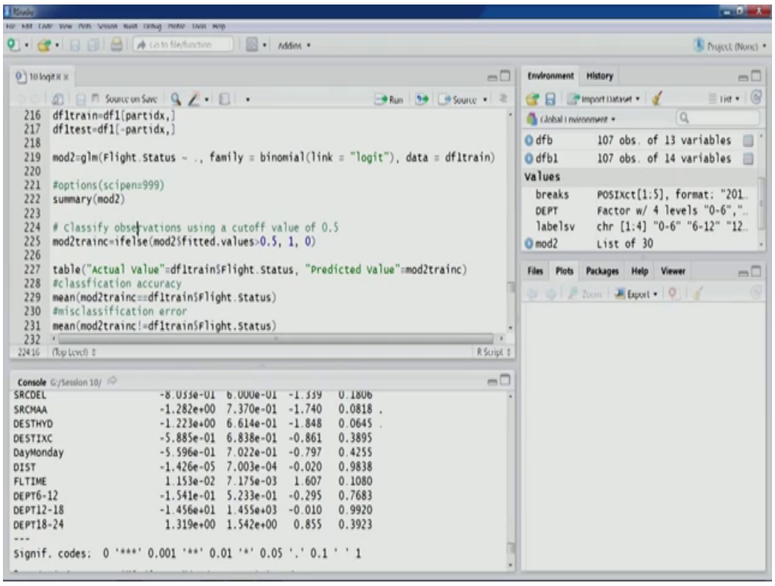
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture - 51
Logistic Regression-Part VI

Welcome to the course Business Analytics and Data Mining Modeling Using R. So, in previous few lectures we have been discussing logistic regression. So, in previous lecture we were doing an a modeling exercise or using a flight details data dataset.

So, let us go back to that exercise in R environment. So, we have been able to a build the model right and we understood we also discussed the results and different alternative models that could be created by dropping few variables and merging few categories.

(Refer Slide Time: 00:45)



```
dfltrain=df1[partidx,]
dfltest=df1[-partidx,]

mod2=glm(Flight.Status ~ ., family = binomial(link = "logit"), data = dfltrain)

#options(scipen=999)
summary(mod2)

# Classify observations using a cutoff value of 0.5
mod2trainc=ifelse(mod2$fitted.values>0.5, 1, 0)

table("Actual Value"=dfltrain$Flight.Status, "Predicted Value"=mod2trainc)
#classfication accuracy
mean(mod2trainc==dfltrain$Flight.Status)
#misclassification error
mean(mod2trainc!=dfltrain$Flight.Status)
```

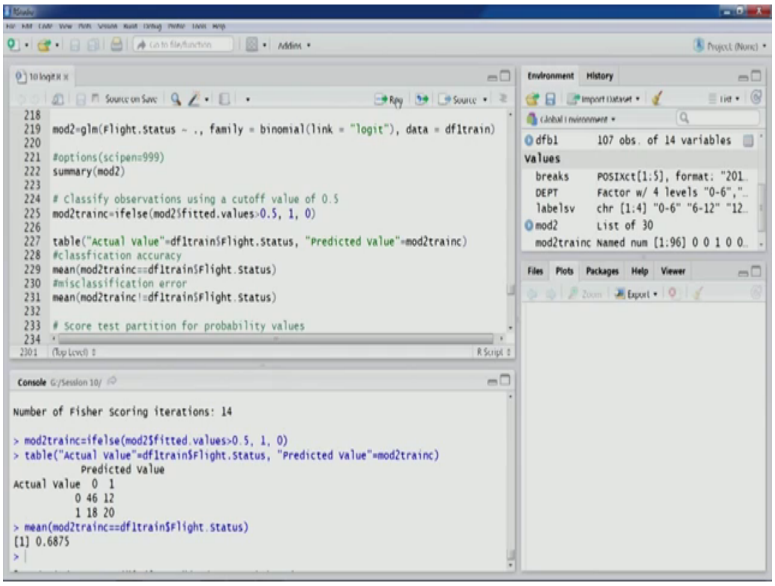
```
SRCDEL                -8.033e-01  6.000e-01  -1.339   0.1806
SRCMAA                -1.282e+00  7.370e-01  -1.740   0.0818 .
DESTHYD               -1.223e+00  6.614e-01  -1.848   0.0645 .
DESTIXC               -5.885e-01  6.838e-01  -0.861   0.3895
DayMonday             -5.596e-01  7.022e-01  -0.797   0.4255
DIST                  -1.426e-05  7.003e-04  -0.020   0.9838
FLTIME                 1.153e-02  7.175e-03   1.607   0.1080
DEPT6-12              -1.541e-01  5.233e-01  -0.295   0.7683
DEPT12-18             -1.456e+01  1.455e+03  -0.010   0.9920
DEPT18-24              1.319e+00  1.542e+00   0.855   0.3923
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
Values
  breaks   POSIXct[1:5], format: "201…
  DEPT     Factor w/ 4 levels "0-6",".…
  labelsv  chr [1:4] "0-6" "6-12" "12…
mod2       List of 30
```

So, we would be doing that, but before that we would like to check the performance of this model, look at some of the plots and discuss then. So, first we would like to check the performance of training partition itself. So, the we have fitted value with us. So, fitted values are nothing, but the probability estimated probabilities value, values for all the observation that we that we used in previous lecture.

So, same model result we are using again here in this particular lecture. So, first we will use these fitted values to classify observations. So, we can see if fitted value that is estimated probabilities value, if they get greater than 0.5 then the class is 1 another by 0; one means here delayed flight.

(Refer Slide Time: 01:42)



```
mod2=glm(Flight.Status ~ ., family = binomial(link = "logit"), data = df1train)

#options(scipen=999)
summary(mod2)

# Classify observations using a cutoff value of 0.5
mod2trainc=ifelse(mod2$fitted.values>0.5, 1, 0)

table("Actual Value"=df1train$Flight.Status, "Predicted Value"=mod2trainc)
#classfication accuracy
mean(mod2trainc==df1train$Flight.Status)
#misclassification error
mean(mod2trainc!=df1train$Flight.Status)

# Score test partition for probability values
```

```
Number of Fisher Scoring iterations: 14

> mod2trainc=ifelse(mod2$fitted.values>0.5, 1, 0)
> table("Actual Value"=df1train$Flight.Status, "Predicted Value"=mod2trainc)
            Predicted Value
Actual Value  0  1
           0 46 12
           1 18 20
> mean(mod2trainc==df1train$Flight.Status)
[1] 0.6875
>
```

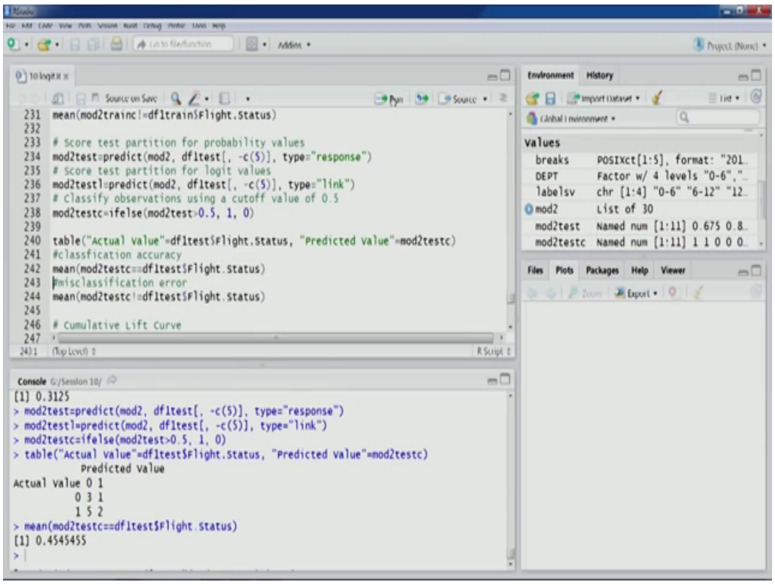
```
dfb1        107 obs. of 14 variables
Values
  breaks     POSIXct[1:5], format: "201…
  DEPT       Factor w/ 4 levels "0-6","…
  labelsv    chr [1:4] "0-6" "6-12" "12…
mod2        List of 30
  mod2trainc Named num [1:96] 0 0 1 0 0…
```

So, let us run this. So, we will have the classification and once that is there we can create our classification matrix, actual value is stored here flight status and the predicted values in, just now we have computed. So, let us look at the classification matrix. So, you can see in the diagonal element 46 plus 20. So, these are the observations correctly classified 46 and 20, and we can also see that off diagonal elements 18 and 12.

So, these are the incorrectly classified observations. So, let us look at the classification accuracy. So, this comes out to be 0.6875, so 68.75 percent accuracy; So, this is still good enough, for reason being we had very small sample size and even smaller sample size for training partition and then there are so many factor variables. So, therefore, there can be so many combinations of you know values which might not be modeled or which might not have enough number of observations to, you know have a good model. So, it still we got 68.75 percent actually accuracy. This is the error.

Now, we can look at performance of our model, on the 10 percent observation that we had left out in the test partition. So, let us look at the performance. So, first let us score the partition for probabilities value. So, we will have this. So, this is scored. Then let us also score the logic values and then let us classify these observations. So, once this is done, let us generate the classification matrix.

(Refer Slide Time: 03:29)



```
mean(mod2trainc!=df1train$Flight.Status)

# Score test partition for probability values
mod2test=predict(mod2, df1test[, -c(5)], type="response")
# Score test partition for logit values
mod2testl=predict(mod2, df1test[, -c(5)], type="link")
# Classify observations using a cutoff value of 0.5
mod2testc=ifelse(mod2test>0.5, 1, 0)

table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod2testc)
#classfication accuracy
mean(mod2testc==df1test$Flight.Status)
#misclassification error
mean(mod2testc!=df1test$Flight.Status)

# Cumulative Lift Curve
```

```
[1] 0.3125
> mod2test=predict(mod2, df1test[, -c(5)], type="response")
> mod2testl=predict(mod2, df1test[, -c(5)], type="link")
> mod2testc=ifelse(mod2test>0.5, 1, 0)
> table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod2testc)
            Predicted Value
Actual Value 0 1
           0 3 1
           1 5 2
> mean(mod2testc==df1test$Flight.Status)
[1] 0.4545455
> 
```

```
Values
breaks     POSIXct[1:5], format: "201…
DEPT       Factor w/ 4 levels "0-6","…
labelsv    chr [1:4] "0-6" "6-12" "12…
mod2       List of 30
mod2test   Named num [1:11] 0.675 0.8…
mod2testc  Named num [1:11] 1 1 0 0 0…
```

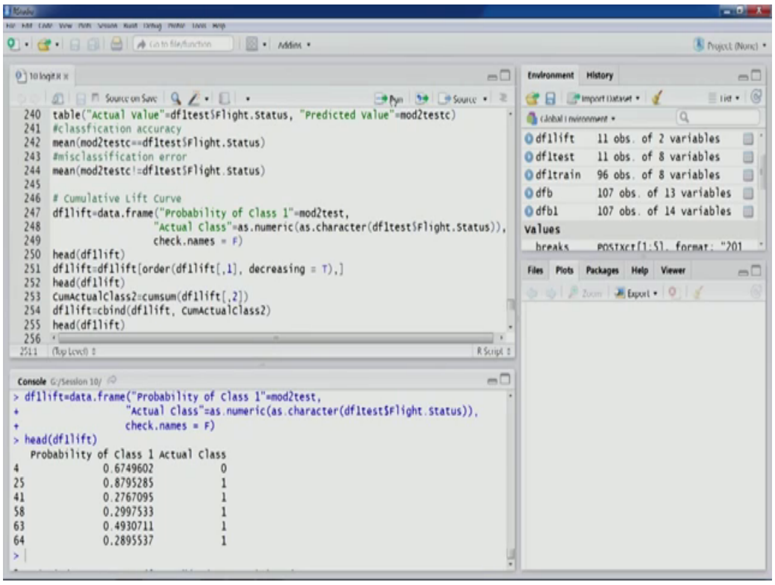
So, this is the classification matrix, we had just about 11 observations. So, 3 plus 2 5 observations have been correctly classified and the remaining observation 6, have been incorrectly classified. So, our model does not seem to be performing well on new data. So, just 45 percent; So, that is expected also from training partition to testing partition performance, might up logged on especially a specifically in this case, because we had very small sample size and you know too many factor variables right.

So, as we have talked about and small the initial lectures when we talk about factor variables, it is the number of categories that also play a role in terms of data, determining the minimum number of sample observation, sample points that we should have. So, you have too many factor variables and with, you know and quite a few number of categories and that puts more higher restriction on sample size. So, this is our error. So, error is much higher interest partition.

So, let us look at the lift curve in terms. So, we would like to understand, even though the model is not performing as well, as we would expect, because of the issues that we discussed sample size and others. Still in terms of identifying the most probable ones, most likely flights which are going to be delayed lift curve kind of give us some important information, how this particular model is going to help us with respect to an average case scenario.

So, as we have been doing for plotting cumulative lift curve; First we cleared the state of him, where we have first column we have probabilities values and then the second column we have typically the actual class. So, let us create this, let us look at first 6 observations.

(Refer Slide Time: 05:26)



```
table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod2testc)
#classfication accuracy
mean(mod2testc==df1test$Flight.Status)
#misclassification error
mean(mod2testc!=df1test$Flight.Status)

# Cumulative Lift Curve
df1lift=data.frame("Probability of Class 1"=mod2test,
                   "Actual Class"=as.numeric(as.character(df1test$Flight.Status)),
                   check.names = F)
head(df1lift)
df1lift=df1lift[order(df1lift[,1], decreasing = T),]
head(df1lift)
CumActualClass2=cumsum(df1lift[,2])
df1lift=cbind(df1lift, CumActualClass2)
head(df1lift)
```

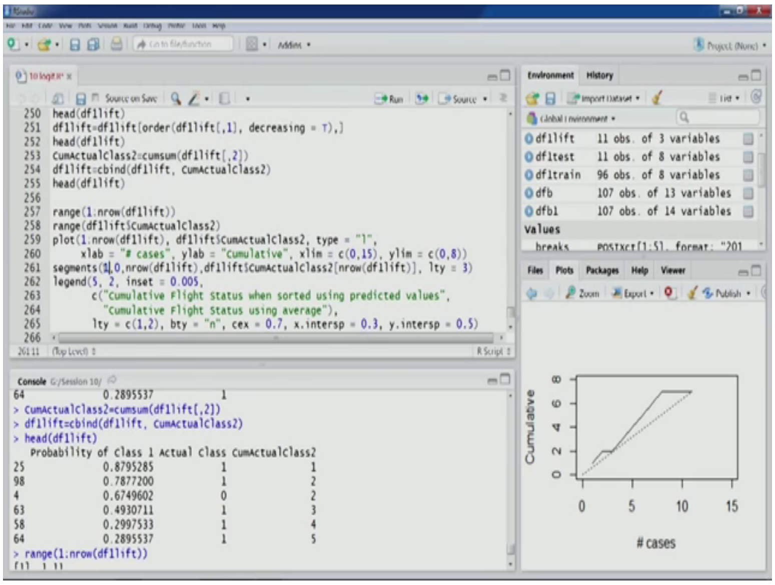
```
> df1lift=data.frame("Probability of Class 1"=mod2test,
+                    "Actual Class"=as.numeric(as.character(df1test$Flight.Status)),
+                    check.names = F)
> head(df1lift)
   Probability of Class 1 Actual Class
4               0.6749602            0
25              0.8795285            1
41              0.2767095            1
58              0.2997533            1
63              0.4930711            1
64              0.2895537            1
> 
```

```
df1lift     11 obs. of 2 variables
df1test     11 obs. of 8 variables
df1train    96 obs. of 8 variables
dfb         107 obs. of 13 variables
dfb1        107 obs. of 14 variables
Values
  breaks    POSIXct[1:5], format: "201
```

So, we can see that these are the values. So, this is for test partition, we have total 11 observations only and once this is done, we would like to sort this particular data frame with using actual class variable and decreasing order. So, the code is written there. Let us look at the sorted values. So, these are the sorted values. So, the data frame has been sorted with respect to probabilities values correction, with respect to probabilities will be descending, decreasing probabilities values descending order and.

Now, we will compute the cumulative class, and let us look at the cumulative class variable this data frame, you can see. So, this kind of exercise we have done before as well. Now let us create the cumulative lift curve, let us look at the range. So, this is the range. So, as I said we have just 11 observations and this is the range for cumulative actual class. So, out of 11 probably we have just 7 observations we. belonging to delete class category. So, let us create the plot.

(Refer Slide Time: 06:36)



```
head(df1lift)
df1lift=df1lift[order(df1lift[,1], decreasing = T),]
head(df1lift)
CumActualClass2=cumsum(df1lift[,2])
df1lift=cbind(df1lift, CumActualClass2)
head(df1lift)

range(1:nrow(df1lift))
range(df1lift$CumActualClass2)
plot(1:nrow(df1lift), df1lift$CumActualClass2, type = "l",
     xlab = "# cases", ylab = "Cumulative", xlim = c(0,15), ylim = c(0,8))
segments(1,0,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
legend(5, 2, inset = 0.005,
       c("Cumulative Flight Status when sorted using predicted values",
         "Cumulative Flight Status using average"),
       lty = c(1,2), bty = "n", cex = 0.7, x.intersp = 0.3, y.intersp = 0.5)
```

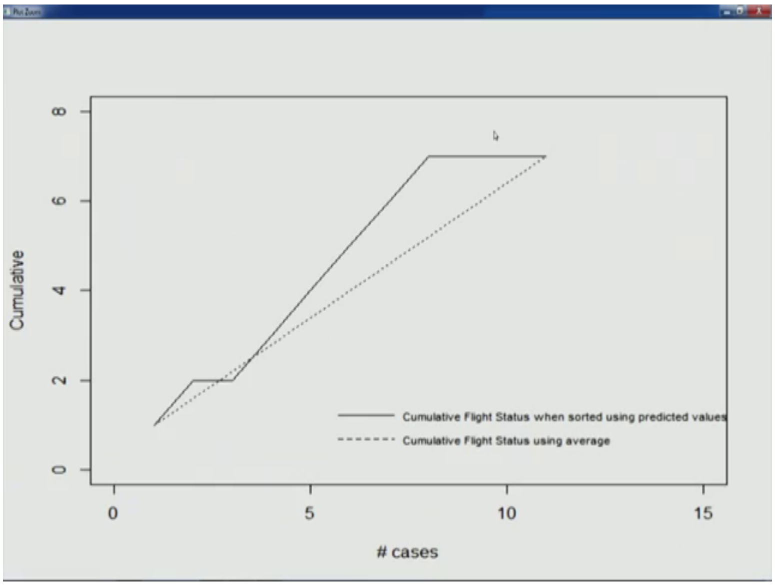
```
64                0.2895537            1
> CumActualClass2=cumsum(df1lift[,2])
> df1lift=cbind(df1lift, CumActualClass2)
> head(df1lift)
   Probability of Class 1 Actual Class CumActualClass2
25                0.8795285            1               1
98                0.7877200            1               2
4                 0.6749602            0               2
63                0.4930711            1               3
58                0.2997533            1               4
64                0.2895537            1               5
> range(1:nrow(df1lift))
[1]  1 11
```

```
df1lift     11 obs. of 3 variables
df1test     11 obs. of 8 variables
df1train    96 obs. of 8 variables
dfb         107 obs. of 13 variables
dfb1        107 obs. of 14 variables
Values
  breaks    POSIXct[1:5], format: "201
```

So, we need to correct our limits first, let us make it 8 and other things seem to be ok. So, now, let us run again. So, this is our plot, let us create the reference line. So, reference line can further be corrected here, as we can see first record cumulative actual class is 1. So, this reference line can actually be 1 and 1.

So, this would be much, you know corrected version, improved version of reference line. For this we will have to create the plot once again then reference line and then we can add the legend. So, this is the plot that we have.

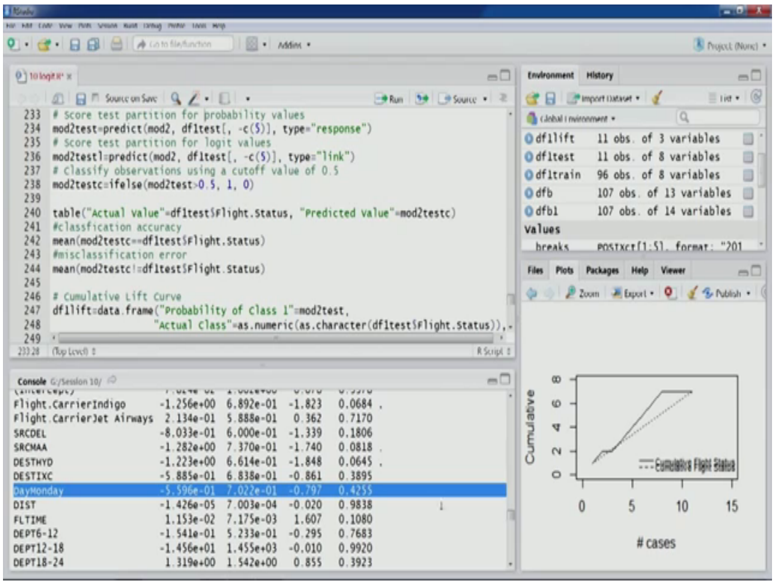
(Refer Slide Time: 07:28)



So, you can see that the lift curve goes below reference line somewhere here, but again it picks up so, but in this starting region you can see, you know for most part of this particular lift curve this solid line which is for our model remains above this reference line; that means, our model does a better job of identifying the most probable delayed flights with respect to the reference case; that is average case.

So, now what we have understood from this exercise and the results that we had discussed, we can go ahead and do a remodeling of this, this particular dataset, this particular problem, delayed fly, delayed flights prediction problem. So, let us look at the (Refer Time: 08:24) results once again.

(Refer Slide Time: 08:22)



```
# Score test partition for probability values
mod2test=predict(mod2, df1test[, -c(5)], type="response")
# Score test partition for logit values
mod2testl=predict(mod2, df1test[, -c(5)], type="link")
# Classify observations using a cutoff value of 0.5
mod2testc=ifelse(mod2test>0.5, 1, 0)

table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod2testc)
#classfication accuracy
mean(mod2testc==df1test$Flight.Status)
#misclassification error
mean(mod2testc!=df1test$Flight.Status)

# Cumulative Lift Curve
df1lift=data.frame("Probability of Class 1"=mod2test,
                   "Actual Class"=as.numeric(as.character(df1test$Flight.Status)),
```

```
Flight.CarrierIndigo      -1.256e+00  6.892e-01  -1.823   0.0684 .
Flight.CarrierJet Airways  2.134e-01  5.888e-01   0.362   0.7170
SRCDEL                    -8.033e-01  6.000e-01  -1.339   0.1806
SRCMAA                    -1.282e+00  7.370e-01  -1.740   0.0818 .
DESTHYD                   -1.223e+00  6.614e-01  -1.848   0.0645 .
DESTIXC                   -5.885e-01  6.838e-01  -0.861   0.3895
DayMonday                 -5.596e-01  7.022e-01  -0.797   0.4255
DIST                      -1.426e-05  7.003e-04  -0.020   0.9838
FLTIME                     1.153e-02  7.175e-03   1.607   0.1080
DEPT6-12                  -1.541e-01  5.233e-01  -0.295   0.7683
DEPT12-18                 -1.456e+01  1.455e+03  -0.010   0.9920
DEPT18-24                  1.319e+00  1.542e+00   0.855   0.3923
```

```
df1lift     11 obs. of 3 variables
df1test     11 obs. of 8 variables
df1train    96 obs. of 8 variables
dfb         107 obs. of 13 variables
dfb1        107 obs. of 14 variables
Values
  breaks    POSIXct[1:5], format: "201
```

So, as we can see that from in these results that three variables were found to be significant flight carrier. This is significant specifically that is indigo dummy variable and then source is also significant, one of the dummy variable is significant. Destination also one of the dummy variable is significant right.

Day is; however, you know this is not significant, but this could be of practical importance, reason being even though in our small data set that we have, we just have flights on 2 days Sunday and Monday; however, on working days the schedule might be more, schedule might be, traffic might be more, more flights might be running during working days.

So, therefore, and in comparison to weekends, so therefore, this particular variable a day is important information. So, because of its practical importance we would like to keep it, despite it being insignificant in our small data set and model distance does not seem to be significant it is highly in significant. So, probably it is not an important point. First anyway the flights you know the, the jets that are used they, they fly at a quite good speed, so distance is not a matter.

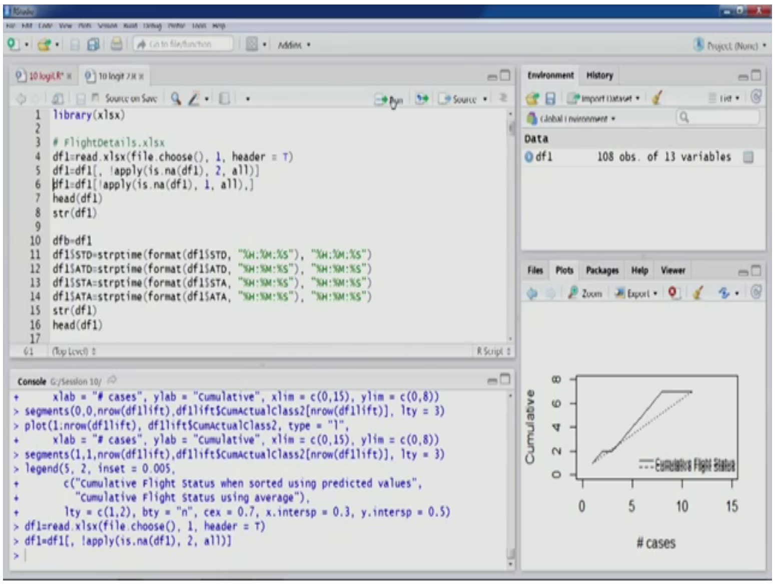
It is the operational for, factors which matter more. Flight time of course, can be important, because flight time with, if the flight is now, time is more then there are more chances for some factor playing a role, more like factor playing a role in terms of delaying the flight. So, flight time you would like to keep, and also you can see that p value is on the lower side. So, you know y is small margin, this has been ah; otherwise it has, it would have been significant at 90 percent confidence interval.

So, it has been left out by small margin, departure intervals also, because of the practical importance we would have, we would like to keep it, because flights, at what time the flights are originating or arriving. So, that can play an important role in terms of, whether the flight is going to be delayed or not; however, we would like to look at the categories, we would like to find out the categories, use the categories which would give us more, some improvement in our classification model.

So, from this, probably it seems that the departure, this one 18 to 24 category with respect to reference, you know if we look at the p value is smaller. So, probably we look to combine these two categories; the reference category and this departure 18 to 24 and the other two categories will group into 1 departure 6 to 12 and 12 to 18, we would like to group into 1. So, that we can call day and the other category 18 to 24 and then 24 to 6 that we would like to keep at the reference category 0 to 6, we would like to keep as night, part of night group.

So, two categories will be part of day group, two categories will be part of night group and we will like to see how it is performing in the model.

(Refer Slide Time: 12:08)



```
library(xlsx)

# FlightDetails.xlsx
df1=read.xlsx(file.choose(), 1, header = T)
df1=df1[, !apply(is.na(df1), 2, all)]
df1=df1[!apply(is.na(df1), 1, all),]
head(df1)
str(df1)

dfb=df1
df1$STD=strptime(format(df1$STD, "%H:%M:%S"), "%H:%M:%S")
df1$ATD=strptime(format(df1$ATD, "%H:%M:%S"), "%H:%M:%S")
df1$STA=strptime(format(df1$STA, "%H:%M:%S"), "%H:%M:%S")
df1$ATA=strptime(format(df1$ATA, "%H:%M:%S"), "%H:%M:%S")
str(df1)
head(df1)
```

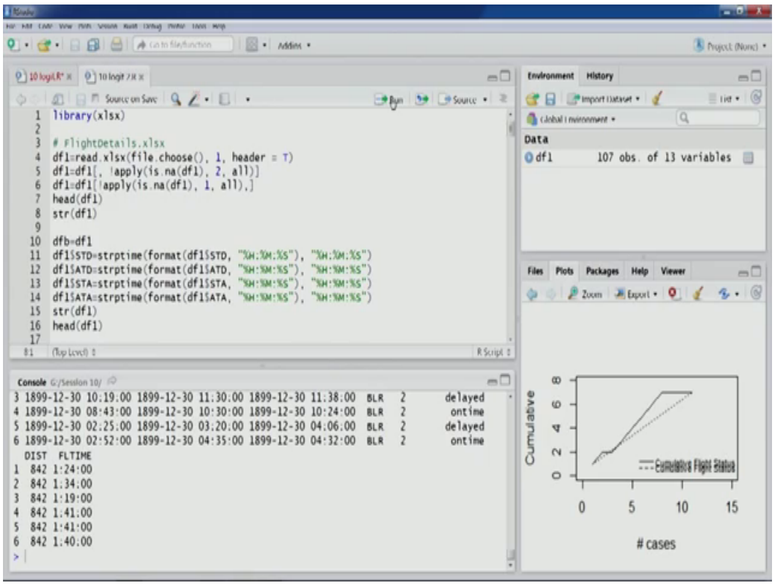
```
+      xlab = "# cases", ylab = "Cumulative", xlim = c(0,15), ylim = c(0,8))
> segments(0,0,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
> plot(1:nrow(df1lift), df1lift$CumActualClass2, type = "l",
+      xlab = "# cases", ylab = "Cumulative", xlim = c(0,15), ylim = c(0,8))
> segments(1,1,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
> legend(5, 2, inset = 0.005,
+        c("Cumulative Flight Status when sorted using predicted values",
+          "Cumulative Flight Status using average"),
+        lty = c(1,2), bty = "n", cex = 0.7, x.intersp = 0.3, y.intersp = 0.5)
> df1=read.xlsx(file.choose(), 1, header = T)
> df1=df1[, !apply(is.na(df1), 2, all)]
> |
```

```
df1        108 obs. of 13 variables
```

So, let us. So, let us open start our modeling exercise. So, before let us clear some of these, some of these variables and data frames from the environment section, because that might create some problem in this particular code. So, what we will do.

So, this library is already loaded. So, we would like to again import the data set. Let us remove an a row and a columns and a rows for 6 observations a structure.

(Refer Slide Time: 12:47)



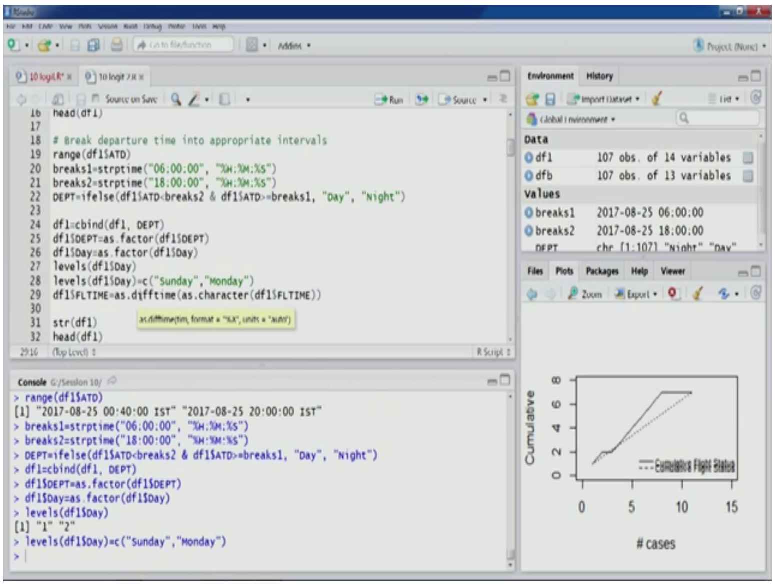
```
library(xlsx)

# FlightDetails.xlsx
df1=read.xlsx(file.choose(), 1, header = T)
df1=df1[, !apply(is.na(df1), 2, all)]
df1=df1[!apply(is.na(df1), 1, all),]
head(df1)
str(df1)

dfb=df1
df1$STD=strptime(format(df1$STD, "%H:%M:%S"), "%H:%M:%S")
df1$ATD=strptime(format(df1$ATD, "%H:%M:%S"), "%H:%M:%S")
df1$STA=strptime(format(df1$STA, "%H:%M:%S"), "%H:%M:%S")
df1$ATA=strptime(format(df1$ATA, "%H:%M:%S"), "%H:%M:%S")
str(df1)
head(df1)
```

```
3 1899-12-30 10:19:00 1899-12-30 11:30:00 1899-12-30 11:38:00  BLR   2        delayed
4 1899-12-30 08:43:00 1899-12-30 10:30:00 1899-12-30 10:24:00  BLR   2         ontime
5 1899-12-30 02:25:00 1899-12-30 03:20:00 1899-12-30 04:06:00  BLR   2        delayed
6 1899-12-30 02:52:00 1899-12-30 04:35:00 1899-12-30 04:32:00  BLR   2         ontime
  DIST  FLTIME
1  842 1:24:00
2  842 1:34:00
3  842 1:19:00
4  842 1:41:00
5  842 1:41:00
6  842 1:40:00
> |
```

```
df1        107 obs. of 13 variables
```

So, this we all have already gone through in previous lecture. Let us take a backup, let us change the time, so that it is an appropriate format for us to derive some variables; like we did in previous lecture. So, these are the observation. Now, here we do a certain variable transformation differently based on our learning from previous modeling.

(Refer Slide Time: 13:08)



```
head(df1)

# Break departure time into appropriate intervals
range(df1$ATD)
breaks1=strptime("06:00:00", "%H:%M:%S")
breaks2=strptime("18:00:00", "%H:%M:%S")
DEPT=ifelse(df1$ATD<breaks2 & df1$ATD>=breaks1, "Day", "Night")

df1=cbind(df1, DEPT)
df1$DEPT=as.factor(df1$DEPT)
df1$Day=as.factor(df1$Day)
levels(df1$Day)
levels(df1$Day)=c("Sunday","Monday")
df1$FLTIME=as.difftime(as.character(df1$FLTIME))

str(df1)
head(df1)
```

```
> range(df1$ATD)
[1] "2017-08-25 00:40:00 IST" "2017-08-25 20:00:00 IST"
> breaks1=strptime("06:00:00", "%H:%M:%S")
> breaks2=strptime("18:00:00", "%H:%M:%S")
> DEPT=ifelse(df1$ATD<breaks2 & df1$ATD>=breaks1, "Day", "Night")
> df1=cbind(df1, DEPT)
> df1$DEPT=as.factor(df1$DEPT)
> df1$Day=as.factor(df1$Day)
> levels(df1$Day)
[1] "1" "2"
> levels(df1$Day)=c("Sunday","Monday")
> |
```

```
Data
df1        107 obs. of 14 variables
dfb        107 obs. of 13 variables
Values
breaks1    2017-08-25 06:00:00
breaks2    2017-08-25 18:00:00
DEPT       chr [1:107] "Night" "Day"
```

So, as you can see that range is of course, going to remain same, but now the grouping that we are going to do, the categories the groups that we are going to create for departure time interval are different. Now, we would like to as, as we discussed we would like to create these two groups day and night. So, all the all the flights, which fly from, which fly from 6 to 18 hours, so that would be categorized as day, that would be grouped as a day and remaining flights would be grouped as night.

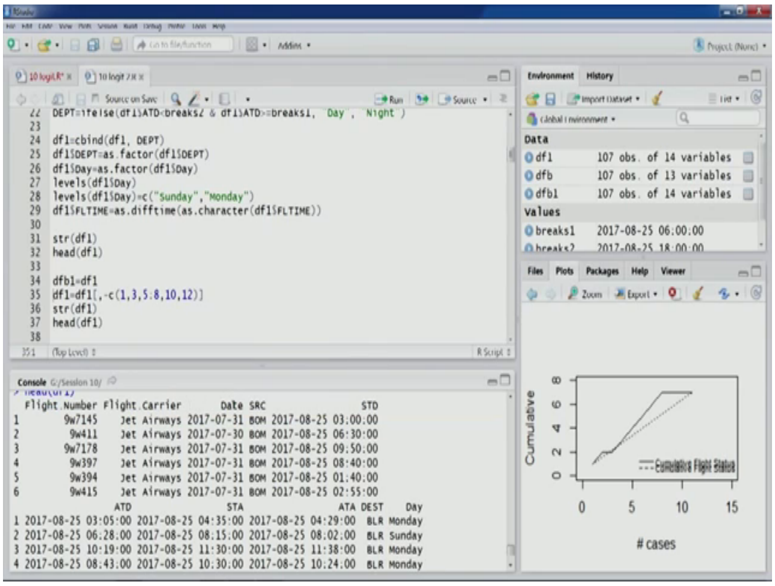
So, we can have few other changes, so we do not have to should not be restricting ourselves to this particular time 6 and 18 and this can be changed depending on how useful it in terms of operating delayed flights. So, we have to keep working on this these in categories and how they are created. So, for this model we will go with these hours; 6 to 18 and then remaining as another group.

So, we are using if else to perform this categorization. So, once it is done, we will you can see in the environment section and character vector has been created, for all observation, specifying whether it is a night flight or day flight. So, let us add this to our data frame. Now you would like to convert it into a factor variable, so let us do that. They, we would like to as we discussed, we would like to include this in our model though it was found to be insignificant, because of the practical importance.

So, let us convert it into factor variable as well. The labels as we did in previous lecture, labels for day would also we would like to change from 1 and 2 to Sunday and Monday, and flight time as we discussed this is also of practical importance. So, therefore, we would like to have it in our model and as we did in previous lecture we use as dot diff diff time to convert the particular values for this variable into a suitable format. So, these are the values. So, some of the convergence transformation we have proper performed.

Let us look at the first 6 observations, you can see flight time appropriately mention we still have some of the variables that we would, we will be taking substance.

(Refer Slide Time: 15:36)



```
DEPT=ifelse(df1$ATD<breaks2 & df1$ATD>=breaks1, "Day", "Night")

df1=cbind(df1, DEPT)
df1$DEPT=as.factor(df1$DEPT)
df1$Day=as.factor(df1$Day)
levels(df1$Day)
levels(df1$Day)=c("Sunday","Monday")
df1$FLTIME=as.difftime(as.character(df1$FLTIME))

str(df1)
head(df1)

dfb1=df1
df1=df1[,-c(1,3,5:8,10,12)]
str(df1)
head(df1)
```

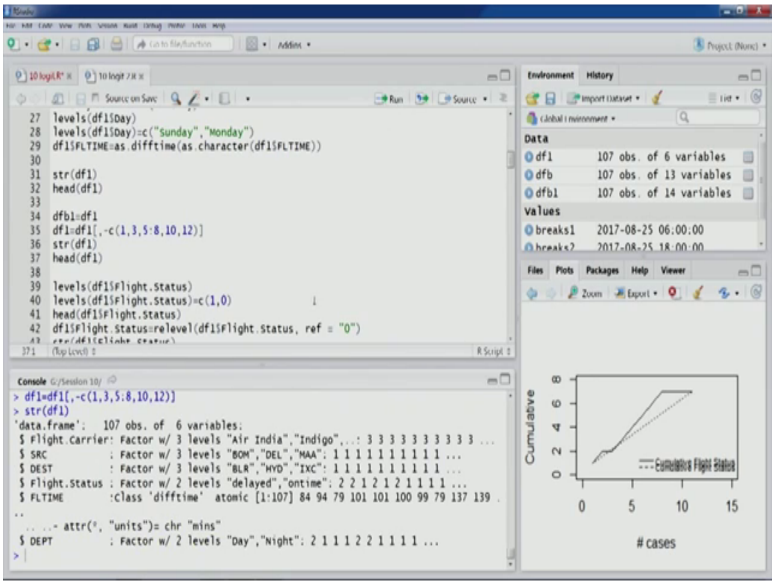
```
> head(df1)
  Flight.Number Flight.Carrier       Date SRC                 STD
1        9w7145    Jet Airways 2017-07-31 BOM 2017-08-25 03:00:00
2         9w411    Jet Airways 2017-07-30 BOM 2017-08-25 06:30:00
3        9w7178    Jet Airways 2017-07-31 BOM 2017-08-25 09:50:00
4         9w397    Jet Airways 2017-07-31 BOM 2017-08-25 08:40:00
5         9w394    Jet Airways 2017-07-31 BOM 2017-08-25 01:40:00
6         9w415    Jet Airways 2017-07-31 BOM 2017-08-25 02:55:00
                  ATD                 STA                 ATA DEST    Day
1 2017-08-25 03:05:00 2017-08-25 04:35:00 2017-08-25 04:29:00  BLR Monday
2 2017-08-25 06:28:00 2017-08-25 08:15:00 2017-08-25 08:02:00  BLR Sunday
3 2017-08-25 10:19:00 2017-08-25 11:30:00 2017-08-25 11:38:00  BLR Monday
4 2017-08-25 08:43:00 2017-08-25 10:30:00 2017-08-25 10:24:00  BLR Monday
```

```
Data
df1       107 obs. of 14 variables
dfb       107 obs. of 13 variables
dfb1      107 obs. of 14 variables
Values
breaks1   2017-08-25 06:00:00
breaks2   2017-08-25 18:00:00
```

So, before that let us take a backup of this data frame. Now, you would see that we are getting rid of column number 1; that is flight number and column number 3 that is state. So, we do not want it, then 5 to 8 that is scheduled time of departure, actual time of departure, scheduled time of arrival, actual time of arrival like the previous models. So, we get rid-off these variables as well.

Then the next one is tenth variable; that is, the next one is tenth variable, so that is day, that is day. Also as we as we saw that even though this is of practical importance, we are not including this on our model, because this was insignificant. So, we will see how would what the results would be, once we excluded. So, this is also gone and then we have distance, so distance also you are trying to get rid-off. So, these variables will get rid-off.

(Refer Slide Time: 16:57)



```
levels(df1$Day)
levels(df1$Day)=c("Sunday","Monday")
df1$FLTIME=as.difftime(as.character(df1$FLTIME))

str(df1)
head(df1)

dfb1=df1
df1=df1[,-c(1,3,5:8,10,12)]
str(df1)
head(df1)

levels(df1$Flight.Status)
levels(df1$Flight.Status)=c(1,0)
head(df1$Flight.Status)
df1$Flight.Status=relevel(df1$Flight.Status, ref = "0")
```

```
> df1=df1[,-c(1,3,5:8,10,12)]
> str(df1)
'data.frame':   107 obs. of  6 variables:
 $ Flight.Carrier: Factor w/ 3 levels "Air India","Indigo",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ SRC           : Factor w/ 3 levels "BOM","DEL","MAA": 1 1 1 1 1 1 1 1 1 1 ...
 $ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
 $ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
 $ FLTIME        :Class 'difftime'  atomic [1:107] 84 94 79 101 101 100 99 79 137 139 .
..
  .. ..- attr(*, "units")= chr "mins"
 $ DEPT          : Factor w/ 2 levels "Day","Night": 2 1 1 1 2 2 1 1 1 1 ...
>
```

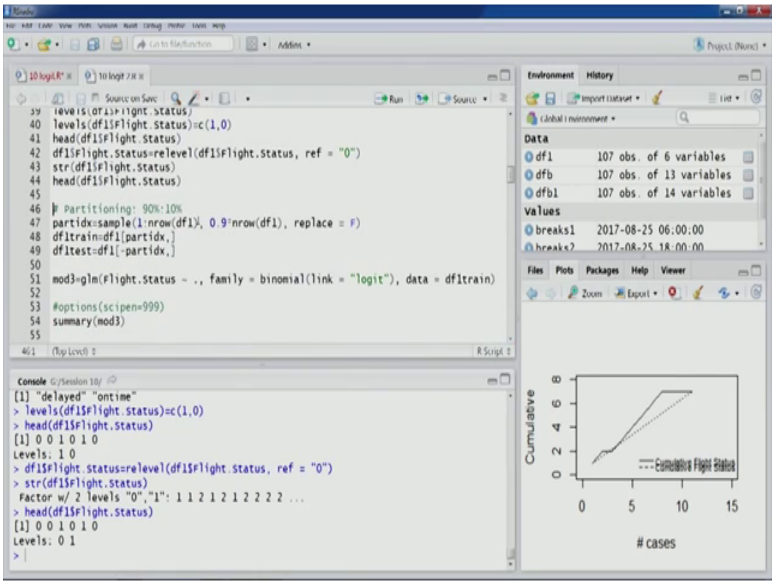
Data
df1 107 obs. of 6 variables
dfb 107 obs. of 13 variables
dfb1 107 obs. of 14 variables
Values
breaks1 2017-08-25 06:00:00
breaks2 2017-08-25 18:00:00

So, now, these are the remaining variables, we have flight carrier ah; three levels, we have source, destination three levels. Flight status we will have to correct the labels as we did in previous modeling, previous lecture. Flight time is and departure. Now we have just two levels day and night. So, let us look at the first 6 observations. So, these are the observations.

Now, let us work on the outcome variable. So, we will go through the same piece of code that we did in previous lecture, we would like to change the labels delayed and on time. So, as we said, because we generally create cumulative lift curve and for that we require this particular variable into 0 and 1 format Americ code format, so that we are able to later on do certain computations.

So, we would like to change this you can see the red is 1 and on time is 0 and now we would like to reliable it, so that the reference category is 0, so let us execute this. So, now, the reference category is 0 as you can see, first 6 observations also you can see.

(Refer Slide Time: 18:13)



```
levels(df1$Flight.Status)
levels(df1$Flight.Status)=c(1,0)
head(df1$Flight.Status)
df1$Flight.Status=relevel(df1$Flight.Status, ref = "0")
str(df1$Flight.Status)
head(df1$Flight.Status)

# Partitioning: 90%:10%
partidx=sample(1:nrow(df1), 0.9*nrow(df1), replace = F)
df1train=df1[partidx,]
df1test=df1[-partidx,]

mod3=glm(Flight.Status ~ ., family = binomial(link = "logit"), data = df1train)

#options(scipen=999)
summary(mod3)
```

```
[1] "delayed" "ontime"
> levels(df1$Flight.Status)=c(1,0)
> head(df1$Flight.Status)
[1] 0 0 1 0 1 0
Levels: 1 0
> df1$Flight.Status=relevel(df1$Flight.Status, ref = "0")
> str(df1$Flight.Status)
 Factor w/ 2 levels "0","1": 1 1 2 1 2 1 2 2 2 2 ...
> head(df1$Flight.Status)
[1] 0 0 1 0 1 0
Levels: 0 1
> |
```

```
Data
df1        107 obs. of 6 variables
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
Values
breaks1    2017-08-25 06:00:00
breaks2    2017-08-25 18:00:00
```

Now, partitioning is same 90 percent for any partition, because of this smaller sample size. So, again in this modeling exercise also we would like to follow the same, so a 90 percent and 10 percent for the testing partition. So, once this is done, let us create the model.

So, model all the arguments remain same, so there is hardly any change, with respect to model let us look at the results. Yes. So, now, if we look at the results, now the level of significance has gone up right. So, you can see the flight time, this is now significant at two star level; that means, it is significant at 99 percent confidence interval; that is a flight time because.

So, this could be, because we have this smaller sample size and this time the observations that have been selected for that to be part of training partition that will also have their influence, because of this for R sample size and you can see flight time is significant at 99 percent confidence level all right. So, this is one change.

So, in the last modeling, this was you know this was just left out by a small margin, to achieve the significance at 90 percent confidence interval line; so, another important source. So, in the previous modeling we saw that madras dummy variable for source, so that was significant at 90 percent confidence interval. Now, this time it is very different at 95 percent confidence interval; 1 star level significance.

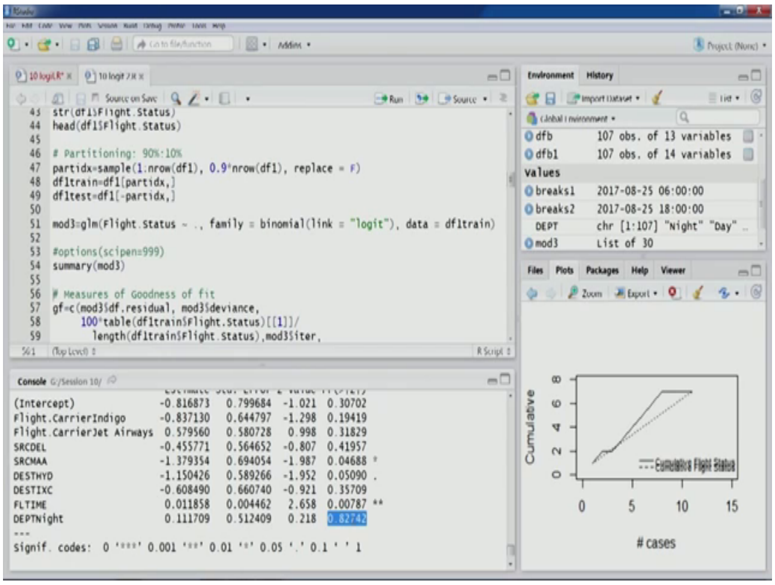
Now, we can see that destination, the same dummy variable which was significant last time also it is significant at 90 percent confidence interval, so there is no change in confidence interval for this one. Now, there is one change we can see that in this time our flight carrier, the indigo dummy variable which was significant at 90 percent contrast and our in previous model.

Now, this has missed by a bit more margin for 90 percent level significance; however, because of the practical importance and even if some of these variables are insignificant, we would like to use them in our you know modeling and in our predicting and in our predictions right, yes scoring new observations.

So, if we look at the last variable that is departure interval night, in this case this does not seem to be the significant right. So, it seems that either we will have to work further on departure time intervals to understand if there is any difference between two groups of intervals; however, we have tried in the us modeling we had created 4 groups and in this particular modeling, we have created two groups. So, this also underscores the importance of descriptive analysis.

So, probably we should focused more on this particular grouping in our pivot table descriptive analysis, to find out which groups, if there are any groups which can be created with respect to prediction of delayed flights, you know classification or delayed flights.

(Refer Slide Time: 21:33)



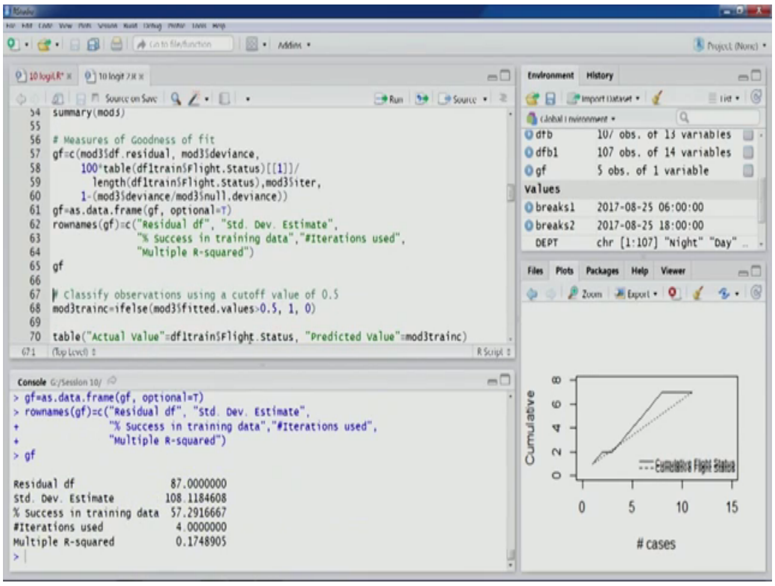
```
str(df1$Flight.Status)
head(df1$Flight.Status)

# Partitioning: 90%:10%
partidx=sample(1:nrow(df1), 0.9*nrow(df1), replace = F)
df1train=df1[partidx,]
df1test=df1[-partidx,]

mod3=glm(Flight.Status ~ ., family = binomial(link = "logit"), data = df1train)

#options(scipen=999)
summary(mod3)

# Measures of Goodness of fit
gf=c(mod3$df.residual, mod3$deviance,
     100*table(df1train$Flight.Status)[[1]]/
       length(df1train$Flight.Status),mod3$iter,
```

```
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -0.816873   0.799684  -1.021  0.30702
Flight.CarrierIndigo    -0.837130   0.644797  -1.298  0.19419
Flight.CarrierJet Airways 0.579560  0.580728   0.998  0.31829
SRCDEL                  -0.455771   0.564652  -0.807  0.41957
SRCMAA                  -1.379354   0.694054  -1.987  0.04688 *
DESTHYD                 -1.150426   0.589266  -1.952  0.05090 .
DESTIXC                 -0.608490   0.660740  -0.921  0.35709
FLTIME                   0.011858   0.004462   2.658  0.00787 **
DEPTNight                0.111709   0.512409   0.218  0.82742
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
Values
breaks1    2017-08-25 06:00:00
breaks2    2017-08-25 18:00:00
DEPT       chr [1:107] "Night" "Day" ...
mod3       List of 30
```

So, another variable this, this destination airport we have already discussed other variables. So, now, let us look at the performance of some of this model. So, majors of goodness of it can also be computed. So, discussion on this we will do later so; however, if you are interested in seeing the values. So, these are the some of the values that could be interesting for a different discussion.

(Refer Slide Time: 22:14)



```
summary(mod3)

# Measures of Goodness of fit
gf=c(mod3$df.residual, mod3$deviance,
     100*table(df1train$Flight.Status)[[1]]/
       length(df1train$Flight.Status),mod3$iter,
     1-(mod3$deviance/mod3$null.deviance))
gf=as.data.frame(gf, optional=T)
rownames(gf)=c("Residual df", "Std. Dev. Estimate",
               "% Success in training data","#Iterations used",
               "Multiple R-squared")
gf

# Classify observations using a cutoff value of 0.5
mod3trainc=ifelse(mod3$fitted.values>0.5, 1, 0)

table("Actual Value"=df1train$Flight.Status, "Predicted Value"=mod3trainc)
```

```
> gf=as.data.frame(gf, optional=T)
> rownames(gf)=c("Residual df", "Std. Dev. Estimate",
+               "% Success in training data","#Iterations used",
+               "Multiple R-squared")
> gf

Residual df                 87.0000000
Std. Dev. Estimate         108.1184608
% Success in training data  57.2916667
#Iterations used             4.0000000
Multiple R-squared           0.1748905
> 
```

```
dfb       107 obs. of 13 variables
dfb1      107 obs. of 14 variables
gf        5 obs. of 1 variable
Values
breaks1   2017-08-25 06:00:00
breaks2   2017-08-25 18:00:00
DEPT      chr [1:107] "Night" "Day" ...
```

So, these are some of the observation multiple R squared is quite low for this model 17 percent. So, that is another reason for low performance of the models in previous exercise and maybe in this exercise as well. So, let us look at the performance. So, from the low R squared value we can say one thing that, probably we need to think about more variables, which can increase your certain you know R square, which can gauge R squared to by certain margin.

So, let us look at the performance of training partition itself. So, mod fitted values we have. So, these are estimated probabilities value, let us classify the training partition observations. Let us look at the classification matrix. So, you can see 44 plus 24 these are the observation correctly classified, 17 and 11, so these are the observation incorrectly classified. So, despite low R squared value and many variables mean insignificant; however, because they are practical, because most of the variables of, are of practical importance.

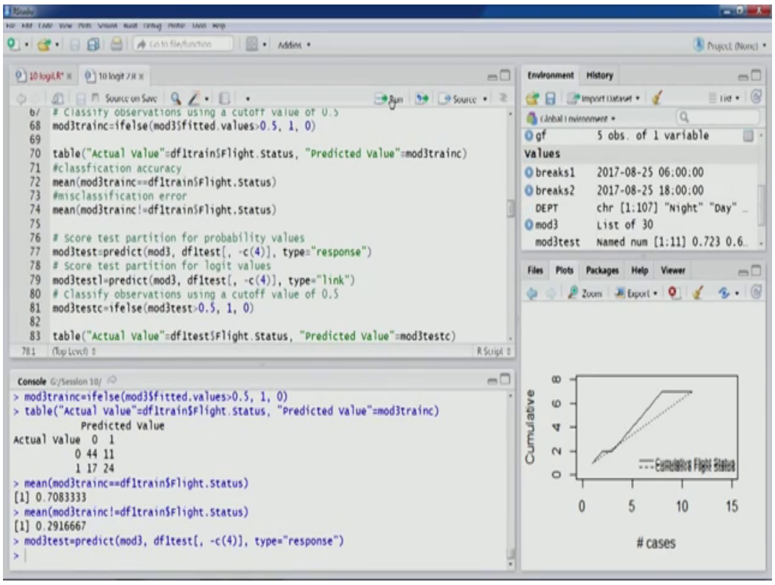
We can see that the performance for any partition is 70 percent 70.8. If you remember the performance of previous model that we had that was around 68 percent this is around 70 percent more than 70 percent. So, there is 2 percent increase after certain transformation of a certain modeling that we did this time.

So, you would see that is this particular modeling exercise, these two models that we have created and looked at the performance underscore some important points. For example, when we talk about data mining modeling, when we talk about prediction tasks classification tasks, even if some variables are not insignificant it you know, but if they are of practical importance that can help us in predicting a new observations that is quite clear from the performance numbers as well.

However, however we can always work on certain, i know a certain variable is specifically factor variables and the transformation regrouping that can be done and improve our performance further.

So, error is despite a small sample size, so we got this much performance. Now, let us look at the performance on tests partitions, we had, we have the still 11 variables i think, like the last time yes 11 variables in the test partition. So, let us see the performance. The outcome variable we would like to leave out, so let us look at the 1 2 3 4; yes, so correctly specified, so it was the fourth column. So, we would like to exclude it for the clarity.

(Refer Slide Time: 25:11)



```
# Classify observations using a cutoff value of 0.5
mod3trainc=ifelse(mod3$fitted.values>0.5, 1, 0)

table("Actual Value"=df1train$Flight.Status, "Predicted Value"=mod3trainc)
#classfication accuracy
mean(mod3trainc==df1train$Flight.Status)
#misclassification error
mean(mod3trainc!=df1train$Flight.Status)

# Score test partition for probability values
mod3test=predict(mod3, df1test[, -c(4)], type="response")
# Score test partition for logit values
mod3testl=predict(mod3, df1test[, -c(4)], type="link")
# Classify observations using a cutoff value of 0.5
mod3testc=ifelse(mod3test>0.5, 1, 0)

table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod3testc)
```

```
> mod3trainc=ifelse(mod3$fitted.values>0.5, 1, 0)
> table("Actual Value"=df1train$Flight.Status, "Predicted Value"=mod3trainc)
            Predicted Value
Actual Value  0  1
           0 44 11
           1 17 24
> mean(mod3trainc==df1train$Flight.Status)
[1] 0.7083333
> mean(mod3trainc!=df1train$Flight.Status)
[1] 0.2916667
> mod3test=predict(mod3, df1test[, -c(4)], type="response")
> 
```

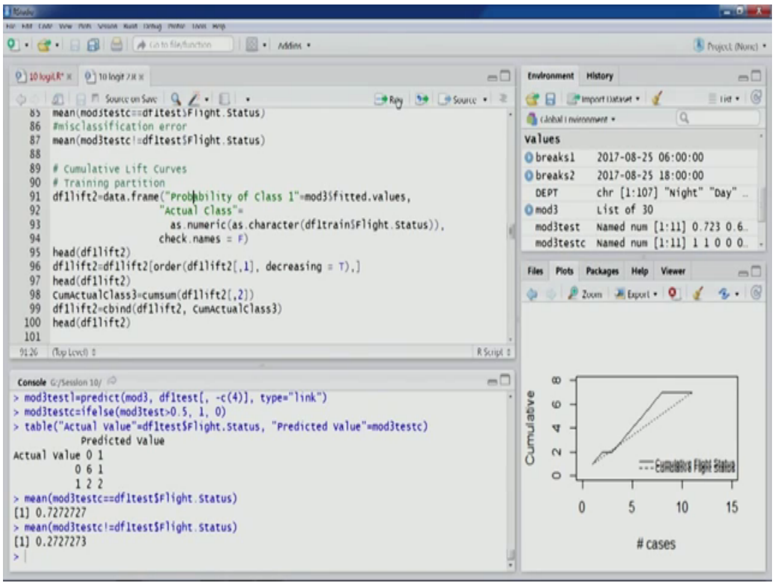
```
gf          5 obs. of 1 variable
Values
breaks1     2017-08-25 06:00:00
breaks2     2017-08-25 18:00:00
DEPT        chr [1:107] "Night" "Day" ...
mod3        List of 30
mod3test    Named num [1:11] 0.723 0.6...
```

So, probabilities values were estimated, logistic values are also extracted and let us classify the observations. Let us look at the classification matrix for test partition. So, we can see that 6 plus 2 or the observation correctly classified and 2 plus 1 that are the, these are the observation incorrectly classified.

So, the model seems do, it seems to be doing a good job this time; however, this as i would like to point out; however, this is because of the way partitioning might happen, the observation that going to training and the remaining observation that are left as part of test partition. So, because of this, there are going to be certain swing performance.

So, therefore, what we require is, larger sample size and within that we require larger number of observations to be part of training partition and therefore, you know check the performance on test partition. You can see in this particular case the performance of model is better in this partition which is not always expected. So, in this case it is performing well.

(Refer Slide Time: 26:24)



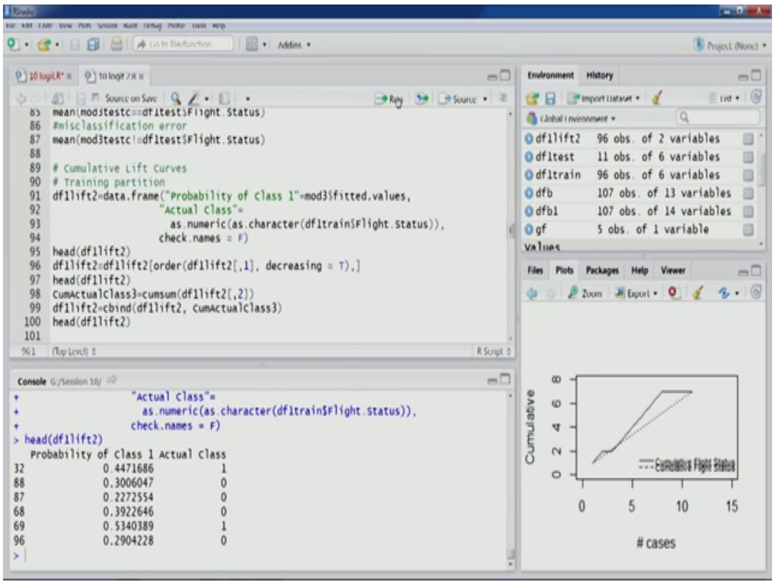
```
mean(mod3testc==df1test$Flight.Status)
#misclassification error
mean(mod3testc!=df1test$Flight.Status)

# Cumulative Lift Curves
# Training partition
df1lift2=data.frame("Probability of Class 1"=mod3$fitted.values,
                    "Actual Class"=
                      as.numeric(as.character(df1train$Flight.Status)),
                    check.names = F)
head(df1lift2)
df1lift2=df1lift2[order(df1lift2[,1], decreasing = T),]
head(df1lift2)
CumActualClass3=cumsum(df1lift2[,2])
df1lift2=cbind(df1lift2, CumActualClass3)
head(df1lift2)
```

```
> mod3testl=predict(mod3, df1test[, -c(4)], type="link")
> mod3testc=ifelse(mod3test>0.5, 1, 0)
> table("Actual Value"=df1test$Flight.Status, "Predicted Value"=mod3testc)
            Predicted Value
Actual Value 0 1
           0 6 1
           1 2 2
> mean(mod3testc==df1test$Flight.Status)
[1] 0.7272727
> mean(mod3testc!=df1test$Flight.Status)
[1] 0.2727273
>
```

```
Values
breaks1     2017-08-25 06:00:00
breaks2     2017-08-25 18:00:00
DEPT        chr [1:107] "Night" "Day" …
mod3        List of 30
mod3test    Named num [1:11] 0.723 0.6…
mod3testc   Named num [1:11] 1 1 0 0 0…
```

Now, what we will do? We will also look at the lift curve this time. So, let us see how our model is doing in terms of identifying the most probable delayed flights. So, as we did you know in previous exercises as well, let us create a data frame of probabilities values and actual class. So, this is for training partition. So, let us create this.

(Refer Slide Time: 26:45)



```
mean(mod3testc==df1test$Flight.Status)
#misclassification error
mean(mod3testc!=df1test$Flight.Status)

# Cumulative Lift Curves
# Training partition
df1lift2=data.frame("Probability of Class 1"=mod3$fitted.values,
                    "Actual Class"=
                      as.numeric(as.character(df1train$Flight.Status)),
                    check.names = F)
head(df1lift2)
df1lift2=df1lift2[order(df1lift2[,1], decreasing = T),]
head(df1lift2)
CumActualClass3=cumsum(df1lift2[,2])
df1lift2=cbind(df1lift2, CumActualClass3)
head(df1lift2)
```

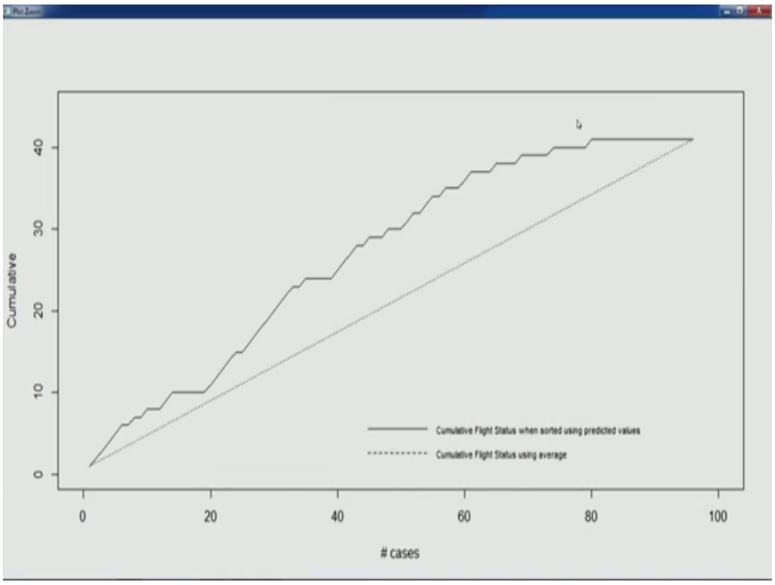
```
+                   "Actual Class"=
+                     as.numeric(as.character(df1train$Flight.Status)),
+                   check.names = F)
> head(df1lift2)
   Probability of Class 1 Actual Class
32              0.4471686            1
88              0.3006047            0
87              0.2272554            0
68              0.3922646            0
69              0.5340389            1
96              0.2904228            0
> |
```

```
df1lift2    96 obs. of 2 variables
df1test     11 obs. of 6 variables
df1train    96 obs. of 6 variables
dfb         107 obs. of 13 variables
dfb1        107 obs. of 14 variables
gf          5 obs. of 1 variable
```

So, these are the observations. Probabilities values and actual corresponding, actual class for those observation, let us order them sort them with respect to probabilities values decreasing order you can see. First observation is highest having highest probability value and the corresponding actual class. Now, let us compute the cumulative actual class numbers that column, let us add it, let us look at the observations. So, once we have this information, we let us look at the range and create.

Let us look at the range let us create now create the cumulative lift curve. So, we can see x limit and y limit, they seem to be 1 to 96 and that is covered and y limit also 1 to 45 1 this is where we have 0 to 45, so this is covered. So, we can go ahead and create our plots. For the reference category in this case this would be 1 and 1. So, we will get better reference line this time. So, this is our reference line and let us legend as well. So, let us look at the cumulative lift curve.

(Refer Slide Time: 27:53)

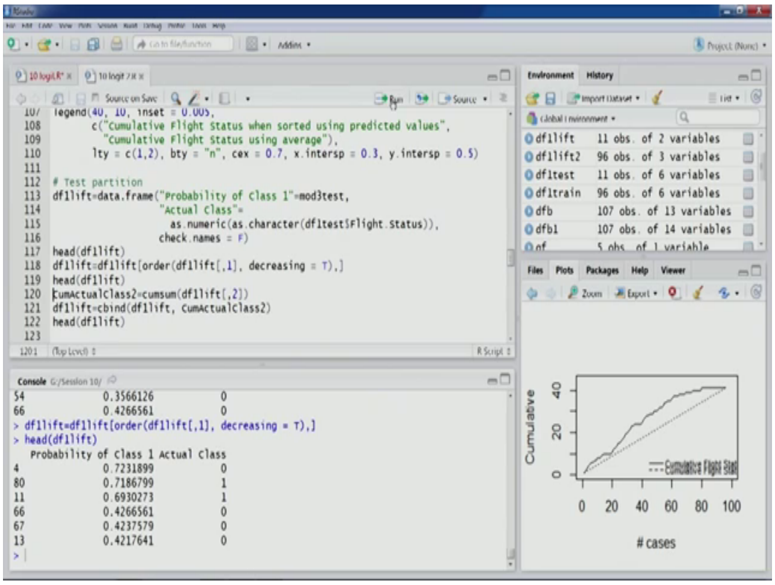


So, as we can see that in majority part of the reference line, the model curve remains above the reference line and. So, that indicates that usefulness of the model. So, as we have talked about that sometimes our model, you know might be giving, you know might not be giving much improved performance in comparison to reference case, typically because of the sample size problems; however, even with those smaller sample size, in terms of identifying the most probable ones, the model does a better job.

So, you can see that in terms of identifying most probable ones the model always remains above this and does a better job. So, as we discussed in the previous lecture, we are always looking to, we are always interested in top left corner of this particular curve right. So, you would like to identify as many observations correctly as possible; so, probably somewhere here.

Let us create cumulative left curve for our test partition. So, first let us create this data frame, probabilities values an actual class. First 6 observation, let us sort them out decreasing order or probabilities values. So, we can see here the same.

(Refer Slide Time: 29:14)



```
legend(40, 10, inset = 0.005,
       c("Cumulative Flight Status when sorted using predicted values",
         "Cumulative Flight Status using average"),
       lty = c(1,2), bty = "n", cex = 0.7, x.intersp = 0.3, y.intersp = 0.5)

# Test partition
df1lift=data.frame("Probability of Class 1"=mod3test,
                   "Actual Class"=
                     as.numeric(as.character(df1test$Flight.Status)),
                   check.names = F)
head(df1lift)
df1lift=df1lift[order(df1lift[,1], decreasing = T),]
head(df1lift)
CumActualClass2=cumsum(df1lift[,2])
df1lift=cbind(df1lift, CumActualClass2)
head(df1lift)
```

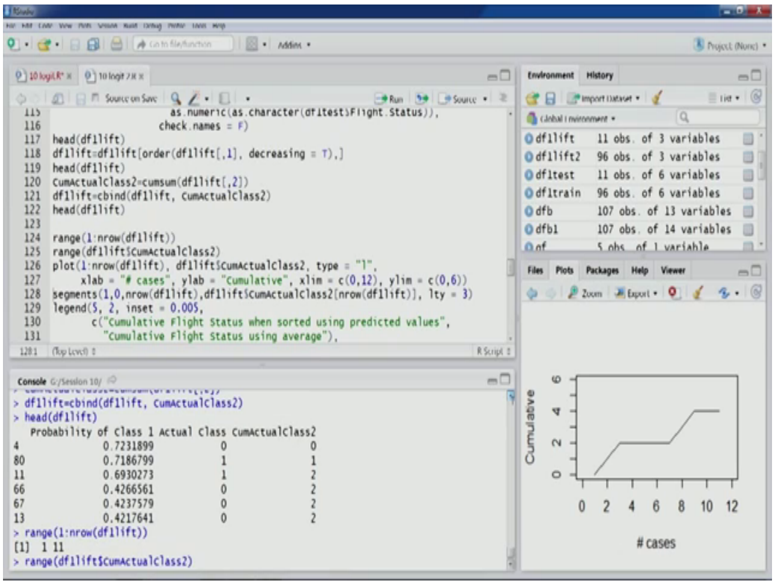
```
54           0.3566126       0
66           0.4266561       0
> df1lift=df1lift[order(df1lift[,1], decreasing = T),]
> head(df1lift)
   Probability of Class 1 Actual Class
4            0.7231899       0
80           0.7186799       1
11           0.6930273       1
66           0.4266561       0
67           0.4237579       0
13           0.4217641       0
> 
```

```
df1lift    11 obs. of 2 variables
df1lift2   96 obs. of 3 variables
df1test    11 obs. of 6 variables
df1train   96 obs. of 6 variables
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
```

Let us compute the cumulative actual class values. As you can see now in this case first reference line we can see, it should be one record and 0 value. So, that has to be reflected there. Let us look at the range again for the test partition.

So, a range for x limit is appropriate for y, we have to change that is also appropriate. So, let us create this curve. So, this is the curve. So, because the very few observation that we have in test partition.

(Refer Slide Time: 29:55)



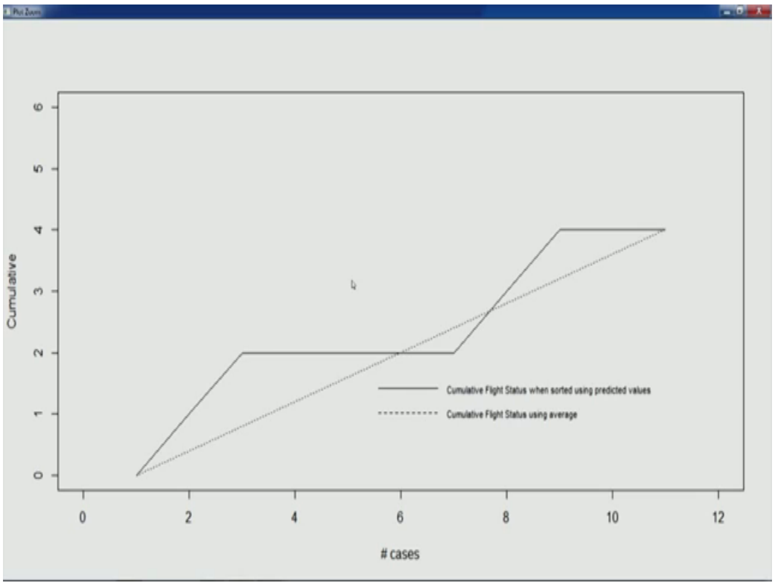
```
                     as.numeric(as.character(df1test$Flight.Status)),
                   check.names = F)
head(df1lift)
df1lift=df1lift[order(df1lift[,1], decreasing = T),]
head(df1lift)
CumActualClass2=cumsum(df1lift[,2])
df1lift=cbind(df1lift, CumActualClass2)
head(df1lift)

range(1:nrow(df1lift))
range(df1lift$CumActualClass2)
plot(1:nrow(df1lift), df1lift$CumActualClass2, type = "l",
     xlab = "# cases", ylab = "Cumulative", xlim = c(0,12), ylim = c(0,6))
segments(1,0,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
legend(5, 2, inset = 0.005,
       c("Cumulative Flight Status when sorted using predicted values",
         "Cumulative Flight Status using average"),
```

```
> df1lift=cbind(df1lift, CumActualClass2)
> head(df1lift)
   Probability of Class 1 Actual Class CumActualClass2
4            0.7231899       0               0
80           0.7186799       1               1
11           0.6930273       1               2
66           0.4266561       0               2
67           0.4237579       0               2
13           0.4217641       0               2
> range(1:nrow(df1lift))
[1]  1 11
> range(df1lift$CumActualClass2)
```

```
df1lift    11 obs. of 3 variables
df1lift2   96 obs. of 3 variables
df1test    11 obs. of 6 variables
df1train   96 obs. of 6 variables
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
```

So, that could be curve, might sometimes be touching the reference line or even going below reference line first coordinate, it should be 1 and 0. So, that is fine. So, we can create our reference line; that is created allegiant.

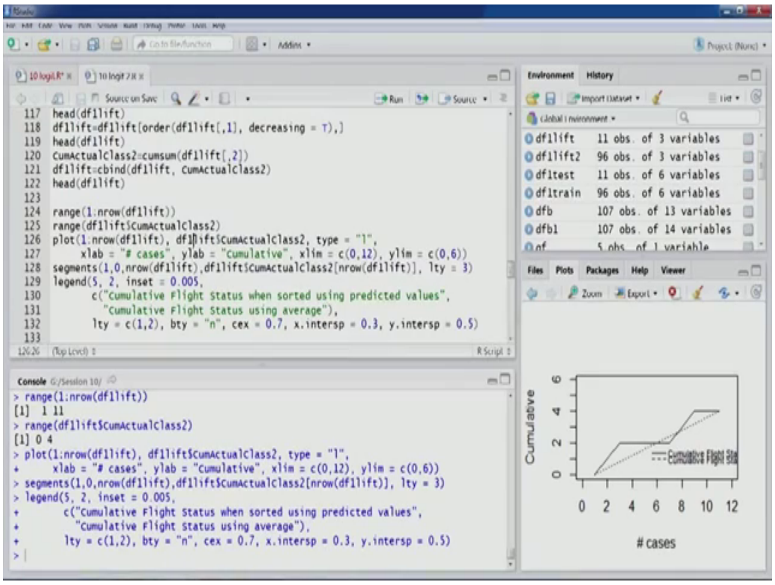
(Refer Slide Time: 30:13)



So, as we can see, because few observations, you know, therefore, you know some part of this particular plot we can see that model is going below the reference line; however, initial part of the plot we can see, the model is clearly above the average scale. This is for very small, just 11 observations in test partition. So, again here we can say usefulness of the model is that, we can easily identify, it is does a much better job of identifying the most probable ones.

So, with this till now in logistic regression modeling we have done two exercises; one was using promotion offers data set and then this one was using this flight detailed data set. Some of the important points like goodness of fit majors, because logistic regression is a classical statistical technique, and it is quite popular as we talked about in, you know statistical modeling.

(Refer Slide Time: 31:07)



```
head(df1lift)
df1lift=df1lift[order(df1lift[,1], decreasing = T),]
head(df1lift)
CumActualClass2=cumsum(df1lift[,2])
df1lift=cbind(df1lift, CumActualClass2)
head(df1lift)

range(1:nrow(df1lift))
range(df1lift$CumActualClass2)
plot(1:nrow(df1lift), df1lift$CumActualClass2, type = "l",
     xlab = "# cases", ylab = "Cumulative", xlim = c(0,12), ylim = c(0,6))
segments(1,0,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
legend(5, 2, inset = 0.005,
       c("Cumulative Flight Status when sorted using predicted values",
         "Cumulative Flight Status using average"),
       lty = c(1,2), bty = "n", cex = 0.7, x.intersp = 0.3, y.intersp = 0.5)
```

```
> range(1:nrow(df1lift))
[1]  1 11
> range(df1lift$CumActualClass2)
[1] 0 4
> plot(1:nrow(df1lift), df1lift$CumActualClass2, type = "l",
+      xlab = "# cases", ylab = "Cumulative", xlim = c(0,12), ylim = c(0,6))
> segments(1,0,nrow(df1lift),df1lift$CumActualClass2[nrow(df1lift)], lty = 3)
> legend(5, 2, inset = 0.005,
+        c("Cumulative Flight Status when sorted using predicted values",
+          "Cumulative Flight Status using average"),
+        lty = c(1,2), bty = "n", cex = 0.7, x.intersp = 0.3, y.intersp = 0.5)
> |
```

```
df1lift    11 obs. of 3 variables
df1lift2   96 obs. of 3 variables
df1test    11 obs. of 6 variables
df1train   96 obs. of 6 variables
dfb        107 obs. of 13 variables
dfb1       107 obs. of 14 variables
of         5 obs. of 1 variable
```

So, in the next lecture, we will also discuss some of those aspects, some of the aspects with respect to the statistical modeling will, will develop will, will build a few more models, which could be more useful for a classical setting. We will also try to understand the difference between logistic regression and linear regression, where we look at why linear regression is not suitable when the variable, whenever outcome variable is categorical in nature. So, we will discuss those points as well.

We will also one particular task that we, that we discussed in our previous logistic relations lectures, that elastic modeling can also be used for profiling in terms of understanding the similarities and differences between two groups. So, how it can help, how logistic relation model can help us in profiling tasks that also we would like to understand. So, some of these things we would like to do and, we will be doing in next lecture. So, at this point we will stop.

Thank you.