K2 Analytics

Building Skills, Building Individuals

Performing Logistic Regression Using R

Rajesh Jakhotia

2 Mar 2017

About K2 Analytics

At K2 Analytics, we believe that skill development is very important for the growth of an individual, which in turn leads to the growth of Society & Industry and ultimately the Nation as a whole. For this it is important that access to knowledge and skill development trainings should be made available easily and economically to every individual.

Our Vision: *"To be the preferred partner for training and skill development"*

Our Mission: *"To provide training and skill development training to individuals, make them skilled & industry ready and create a pool of skilled resources readily available for the industry"*

*We have chosen Business Intelligence and Analytics as our focus area. With this endeavour we make this presentation on "**Logistic Regression**" accessible to all those who wish to learn this technique using R. We hope it is of help to you. For any feedback / suggestion feel free to write back to us at ar.jakhotia@k2analytics.co.in*

You can also write to us for job opportunities on analytics on our email ar.jakhotia@k2analytics.co.in

Welcome to Logistic Regression using R!!!

Assumption

- Note in this presentation we are making the assumption that you, the learner, is aware of the logistic regression technique
- This presentation does not aim to explain the science behind the Logistic Regression; and Maximum Likelihood Estimate technique which is used in logistic regression to compute the beta estimates for variables
- This presentation is designed to help the learner perform Logistic Regression in R and get various statistical outputs like variable estimates, p-value significance, chi-square, concordance, etc and s/he is aware of the interpretation of these statistics

Content

- Understanding Natural Log and Exp
- Logistic Regression Equation
- Creating Dataset to perform Logistic Regression
- Getting Summary Statistics
- Performing Basic Transformations
- Missing Value Imputation
- Computing Information Value
- Visualizations
- Running Logistic Regression
- Getting Various Modeling Statistics Like Rank Ordering, KS, HL GoF, Concordance, Gini

Understanding Natural Log and Exp

Logistic Regression Equation

e (mathematical constant)

The number e is an important mathematical constant that is the base of the natural logarithm. It is approximately equal to 2.71828,[1] and is the limit of $(1 + 1/n)^n$ as n approaches infinity, an expression that arises in the study of compound interest. It can also be calculated as the sum of the infinite series[2]

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \cdots$$

The constant can be defined in many ways; for example, e is the unique real number such that the value of the derivative (slope of the tangent line) of the function $f(x) = e^x$ at the point $x = 0$ is equal to 1.[3] The function e^x so defined is called the exponential function, and its inverse is the natural logarithm, or logarithm to base e. The natural logarithm of a positive number k can also be defined directly as the area under the curve $y = 1/x$ between $x = 1$ and $x = k$, in which case, e is the number whose natural logarithm is 1. There are also more alternative characterizations.

http://en.wikipedia.org/wiki/E_(mathematical_constant)

Natural Logarithm

The **natural logarithm** of a number is its logarithm to the base *e*, where *e* is an irrational and transcendental constant approximately equal to 2.718 281 828. The natural logarithm of *x* is generally written as ln *x*, $\log_e x$, or sometimes, if the base *e* is implicit, simply log *x*.[1] Parentheses are sometimes added for clarity, giving ln(*x*), $\log_e(x)$ or log(*x*). This is done in particular when the argument to the logarithm is not a single symbol, to prevent ambiguity.

The natural logarithm of *x* is the power to which *e* would have to be raised to equal *x*. For example, ln(7.5) is 2.0149..., because $e^{2.0149...}=7.5$. The natural log of *e* itself, ln(*e*), is 1, because $e^1 = e$, while the natural logarithm of 1, ln(1), is 0, since $e^0 = 1$.

http://en.wikipedia.org/wiki/Natural_logarithm

Logistic Regression Equation

The simple logistic model is based on a linear relationship between the natural logarithm (ln) of the odds of an event and a numerical independent variable. The form of this relationship is as follows:

$$L = ln(o) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X + \epsilon,$$

where Y is binary and represent the event of interest (response),coded as 0/1 for failure/success,

p is the proportion of successes,

o is the odds of the event,

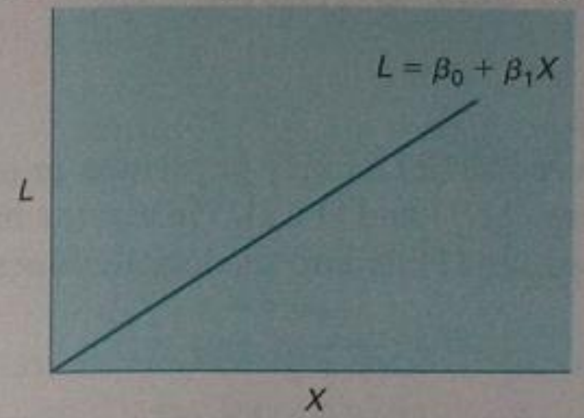
L is the ln(odds of event),

X is the independent variable,

β_0 *and* β_1 are the Y-intercept and the slope, respectively, and

ϵ is the random error.

The relationship between X and L is linear:



http://math.bu.edu/people/nkatenka/MA116/Week5Lecture3.pdf

K2 Analytics
Building Skills, Building Individuals

Creating the Dataset for Performing Logistic Regression

Getting Summary Statistics

Importing the Sample Data File

```
> ## Let us first create our dataset for modeling
> setwd("D:/K2Analytics")
>
> lr_ds1 <- read.table("Datafile/LR_DS1.csv",sep = ",", header = T)
> head(lr_ds1)
  Cust_ID Target Age Gender   Balance Occupation No_OF_CR_TXNS AGE_BKT
1      C1      0  30      M 160378.60        SAL             2   26-30
2      C2      0  43      M  26275.55       PROF            23   41-45
3      C3      0  53      M  33616.47        SAL            45     >50
4      C4      0  45      M   1881.37       PROF             3   41-45
5      C5      0  37      M   3274.37       PROF            33   36-40
6      C6      0  41      M 197632.53        SAL             6   41-45
>
> tail(lr_ds1)
      Cust_ID Target Age Gender   Balance Occupation No_OF_CR_TXNS AGE_BKT
19995  C19995      0  37      M 552566.63       PROF            12   36-40
19996  C19996      0  32      M 210660.74       PROF             1   31-35
19997  C19997      0  46      M 575880.87       PROF            12   46-50
19998  C19998      0  49      M 546795.59       PROF             0   46-50
19999  C19999      0  25      M 249809.73                      16     <25
20000  C20000      1  43      M  97100.48                       3   41-45
```

…so we have imported 20,000 records from the csv file

Importing data file containing customer scores…

```
> lr_ds2 <- read.fwf( "Datafile/LR_FWF.txt",
+ widths=c(6,3), header = T, sep = "|", strip.white = T)
>
> head(lr_ds2)
  Cust_ID SCR
1      C1 826
2      C2 270
3      C3 341
4      C4 284
5      C5 533
6      C6 253
>
> tail(lr_ds2)
      Cust_ID SCR
19995  C19995 879
19996  C19996 379
19997  C19997 711
19998  C19998 564
19999  C19999 362
20000  C20000 473
```

Note: The data file is in Fixed Width Format and as such we use the strip.white = T argument to remove any blank spaces

Importing data from xlsx file

```
> library(RODBC)
> con <- odbcConnectExcel2007("Datafile/LR_Xls.xlsx")
> lr_ds3 <- sqlFetch(con, "HoldingPeriod")
> close(con)
>
> head(lr_ds3)
  Cust_ID Holding_Period
1      C1              9
2      C2             23
3      C3              6
4      C4             16
5      C5             15
6      C6              2
>
> tail(lr_ds3)
      Cust_ID Holding_Period
18995  C18995             14
18996  C18996             22
18997  C18997              5
18998  C18998             26
18999  C18999              5
19000  C19000             18
```

Note: If you get error while importing then use the command given in next slide for importing the xlsx file

Alternate option for importing the xlsx file

```
> ###Importing excel file using XLConnect
> ##install.packages("XLConnect")
> library(XLConnect)

> wb <- loadWorkbook( "Datafile/LR_Xls.xlsx" )
> sh <- getSheets(wb)
> sh[1]
[1] "HoldingPeriod"
> lr_ds3 <- readWorksheet(wb, sheet = "HoldingPeriod",
+ startRow = 1, header = T)
>
> head(lr_ds3)
  Cust_ID Holding_Period
1      C1              9
2      C2             23
3      C3              6
4      C4             16
5      C5             15
6      C6              2
>
> tail(lr_ds3)
      Cust_ID Holding_Period
18995  C18995             14
18996  C18996             22
18997  C18997              5
18998  C18998             26
18999  C18999              5
19000  C19000             18
>
> cbind(nrow(lr_ds3), ncol(lr_ds3))
      [,1] [,2]
[1,] 19000    2
```

Merging the data sets

- Remember the merge syntax for various kinds of joins

```
> inner_join <- merge(x= , y= , by= )
> left_join <- merge(x= , y= , by= , all.x=T)
> right_join <- merge(x= , y= , by= , all.y=T)
> outer_join <- merge(x= , y= , by= , all=T)
> cross_join <- merge(x= , y= )
```

- …. We require full outer join

Merging the datasets… contd

```
> lr_df <- merge(x=lr_ds1, y=lr_ds2, by = "Cust_ID")
>
> LR_DF <- merge(x=lr_df, y=lr_ds3, by="Cust_ID", all.x=T)
> cbind(nrow(LR_DF), ncol(LR_DF))
      [,1] [,2]
[1,] 20000   10
>
> head(LR_DF)
  Cust_ID Target Age Gender   Balance Occupation No_OF_CR_TXNS AGE_BKT SCR Holding_Period
1      C1      0  30      M 160378.60        SAL             2   26-30 826              9
2     C10      1  41      M  84370.59       PROF            14   41-45 843              9
3    C100      0  49      F  60849.26       PROF            49   46-50 328             26
4   C1000      0  49      M  10558.81        SAL            23   46-50 619             19
5  C10000      0  43      M  97100.48                        3   41-45 397              8
6  C10001      0  30      M 160378.60        SAL             2   26-30 781             11
```

After importing, merging, or any other data preparation it is a good practice to do eye-balling of the data and see if things are on expected lines

- We have successfully created our data set to begin the modeling related activities…
- Let us now get some summary statistics from data

Getting summary statistics

```
> summary(LR_DF)
    Cust_ID          Target               Age          Gender  
 C1      :     1   Min.   :0.0000   Min.   :21.0   F: 5525  
 C10     :     1   1st Qu.:0.0000   1st Qu.:30.0   M:14279  
 C100    :     1   Median :0.0000   Median :38.0   O:  196  
 C1000   :     1   Mean   :0.0444   Mean   :38.4            
 C10000  :     1   3rd Qu.:0.0000   3rd Qu.:47.0            
 C10001  :     1   Max.   :1.0000   Max.   :55.0            
 (Other):19994                                              
    Balance             Occupation    No_OF_CR_TXNS  
 Min.   :      0             :4640   Min.   : 0.00  
 1st Qu.:  23737   PROF      :5480   1st Qu.: 7.00  
 Median :  79756   SAL       :5908   Median :13.00  
 Mean   : 146181   SELF-EMP:3272   Mean   :16.62  
 3rd Qu.: 217311   SENP      : 700   3rd Qu.:21.00  
 Max.   :1246967                     Max.   :50.00  

  AGE_BKT         SCR          Holding_Period 
 <25   :1784   Min.   :100.0   Min.   : 0.0  
 >550  :3020   1st Qu.:333.0   1st Qu.: 7.0  
 26-30:3404   Median :560.0   Median :15.0  
 31-35:3488   Mean   :557.1   Mean   :15.2  
 36-40:2756   3rd Qu.:784.0   3rd Qu.:23.0  
 41-45:3016   Max.   :999.0   Max.   :31.0  
 46-50:2532                   NA's   :1000  
```

Read & interpret the summary statistics properly like:

1. Unique ID column should be unique and not have duplicate values
2. Min – Max range for continuous variables
3. Mean & Median values for continuous variables
4. Missing values
5. Distinct categories in categorical variables
6. Top categories

Getting percentile distribution

```
> quantile(LR_DF$Balance,
+ c(0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 1))
         1%          5%         10%         25%         50% 
   575.0709   3828.8695   7255.8710  23736.9150  79755.7450 
        75%         90%         95%         99%        100% 
217310.6325 392294.2230 516901.7505 723000.8420 1246966.7700 

> quantile(LR_DF$Age,
+ c(0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 1))
  1%   5%  10%  25%  50%  75%  90%  95%  99% 100% 
  21   24   26   30   38   47   52   54   55   55 
```

How do I get the percentile distribution for all numeric columns in one shot… rather than running the above syntax for each individual variable

Getting percentile distribution for all numeric variables

```
> apply(LR_DF[,sapply(LR_DF, is.numeric)],
+ 2, quantile,
+ probs=c(0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 1),
+ na.rm=T)
     Target Age      Balance No_OF_CR_TXNS SCR Holding_Period
1%        0  21     575.0709             0 108              0
5%        0  24    3828.8695             1 146              1
10%       0  26    7255.8710             3 194              2
25%       0  30   23736.9150             7 333              7
50%       0  38   79755.7450            13 560             15
75%       0  47  217310.6325            21 784             23
90%       0  52  392294.2230            38 916             28
95%       0  54  516901.7505            45 957             30
99%       1  55  723000.8420            49 992             31
100%      1  55 1246966.7700            50 999             31
```

Performing Basic Transformations

Capping & Flooring

Missing Value Imputation

Capping & Flooring

- Typically we floor and cap variables at P1 and P99 percentile respectively…. Plus based on business judgment
- Let us floor / cap the Balance variable
- Balance can be 0 as such we will not do anything on lower side… on higher side we will just cap the variable at P99

```
> LR_DF$BAL_CAP <-
+ ifelse(LR_DF$Balance > 723000, 723000, LR_DF$Balance)
>
> summary(LR_DF$BAL_CAP)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0   23740   79760  144900  217300  723000
> sd(LR_DF$BAL_CAP)
[1] 164604.1
>
> quantile(LR_DF$BAL_CAP,
+ c(0.01, 0.05, 0.1, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99, 1))
         1%          5%         10%         25%         50%
   575.0709   3828.8695   7255.8710  23736.9150  79755.7450
        75%         90%         95%         99%        100%
217310.6325 392294.2230 516901.7505 722926.2252 723000.0000
```

Decile Function

```
decile <- function(x){
  deciles <- vector(length=10)
  for (i in seq(0.1,1,.1)){
    deciles[i*10] <- quantile(x, i, na.rm=T)
  }
  return (
  ifelse(x<deciles[1], 1,
    ifelse(x<deciles[2], 2,
      ifelse(x<deciles[3], 3,
        ifelse(x<deciles[4], 4,
          ifelse(x<deciles[5], 5,
            ifelse(x<deciles[6], 6,
              ifelse(x<deciles[7], 7,
                ifelse(x<deciles[8], 8,
                  ifelse(x<deciles[9], 9, 10
  ))))))))))
}
```

Decile-wise Response Rate

```
> tmp <- LR_DF
> tmp$deciles <- decile(tmp$No_OF_CR_TXNS)
>
> library(data.table)
data.table 1.9.2  For help type: help("data.table")
> tmp_DT = data.table(tmp)
> RRate <- tmp_DT[, list(
+ min_hp = min(No_OF_CR_TXNS),
+ max_hp = max(No_OF_CR_TXNS),
+ avg_hp = mean(No_OF_CR_TXNS),
+ cnt = length(Target),
+ cnt_resp = sum(Target),
+ cnt_non_resp = sum(Target == 0)) ,
+ by=deciles][order(deciles)]
> RRate$rrate <- RRate$cnt_resp * 100 / RRate$cnt;
> RRate
    deciles min_hp max_hp    avg_hp  cnt cnt_resp cnt_non_resp     rrate
 1:       1      0      2  1.026804 1940       15         1925 0.7731959
 2:       2      3      5  3.989706 2040       36         2004 1.7647059
 3:       3      6      7  6.502110 1422       41         1381 2.8832630
 4:       4      8     10  9.026304 2319       70         2249 3.0185425
 5:       5     11     12 11.511029 1632       67         1565 4.1053922
 6:       6     13     15 13.993430 2131       92         2039 4.3172220
 7:       7     16     18 17.101942 2060       95         1965 4.6116505
 8:       8     19     26 21.337861 2356      124         2232 5.2631579
 9:       9     27     37 32.314583 1920      148         1772 7.7083333
10:      10     38     50 43.989908 2180      200         1980 9.1743119
```

Missing Value Imputation

- Imputation is the process of replacing missing data with substituted values
- In R, missing values are indicated by NA's
- Simple missing value imputation techniques
  - Mean imputation
  - Using information from related observations
  - Imputation based on logical reasons
  - Creating missing category

```
> LR_DF$HP_Imputed <- ifelse(is.na(LR_DF$Holding_Period),
+ 18, LR_DF$Holding_Period)
> summary(LR_DF$HP_Imputed)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    8.00   16.00   15.34   23.00   31.00
> summary(LR_DF$Holding_Period)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    0.0     7.0    15.0    15.2    23.0    31.0    1000
```

Missing Value Treatment

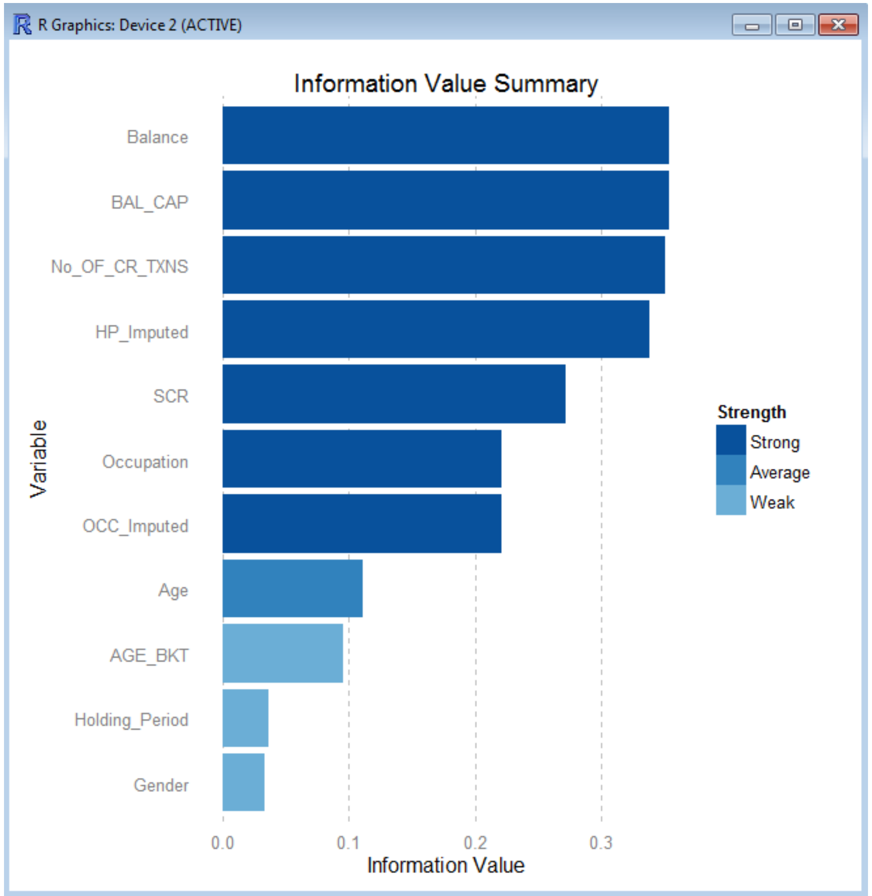
```
> ctab <- xtabs(~Target + Occupation, data = LR_DF)
> ctab
      Occupation
Target      PROF  SAL SELF-EMP SENP
     0 4432 5259 5749     2982  690
     1  208  221  159      290   10
> class(LR_DF$Occupation)
[1] "character"
> LR_DF$Occupation <- as.character(LR_DF$Occupation)
> LR_DF$OCC_Imputed <- ifelse(LR_DF$Occupation=="",
+ "MISSING", LR_DF$Occupation)
> table(LR_DF$OCC_Imputed)

 MISSING     PROF      SAL SELF-EMP     SENP
    4640     5480     5908     3272      700
```

Information Value

Visualization & Pattern Detection

Information Value

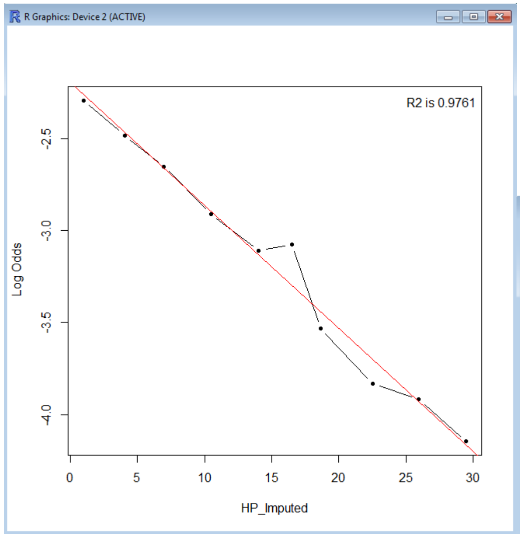
```
> ##install.packages("devtools")
> ##install_github("riv","tomasgreif")
> library(devtools)
> library(woe)
> iv.plot.summary(iv.mult(LR_DF[,!names(LR_DF) %in% c("Cust_ID")],
+ "Target",TRUE))
Information Value 0.11
Information Value 0.03
Information Value 0.35
Information Value 0.22
Information Value 0.35
Information Value 0.1
Information Value 0.27
Information Value 0.04
Information Value 0.35
Information Value 0.34
Information Value 0.22
```



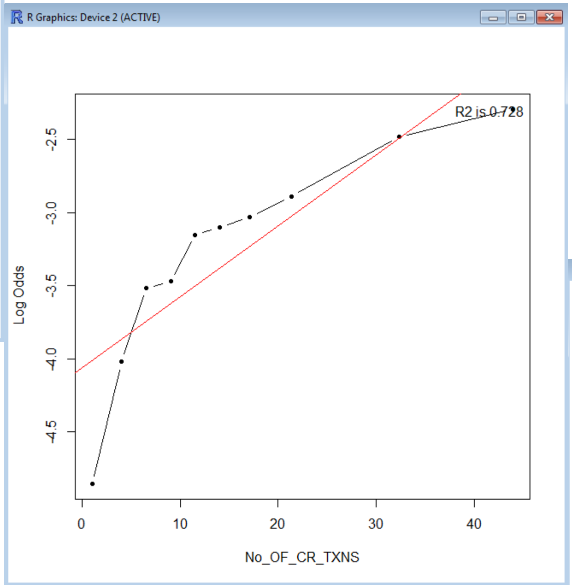
Visualization Code

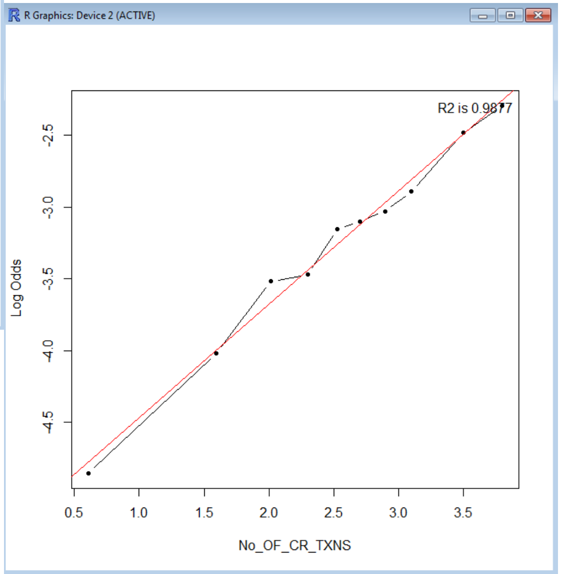
```
fn_visualize <- function(df, target, var, ln_trnfm=0)
{
tmp <- df[, c(var , target)]
head(tmp)
colnames(tmp)[1] = "Xvar"
if (ln_trnfm == 1){
tmp$Xvar = log(tmp$Xvar + 1)
}
tmp$deciles <- decile(tmp$Xvar)


tmp_DT = data.table(tmp)
RRate <- tmp_DT[, list(min_ = min(Xvar), max_ = max(Xvar), avg_ = mean(Xvar),
cnt = length(Target), cnt_responder = sum(Target), cnt_non_responder = sum(Target == 0)) ,
by=deciles][order(deciles)]
RRate$prob <- RRate$cnt_responder / RRate$cnt;
RRate$log_odds <- log(RRate$prob / (1 - RRate$prob))

plot(x=RRate$avg_, y=RRate$log_odds, type="b", pch = 20,
xlab=var, ylab=" Log Odds")

abline(fit <- lm(RRate$log_odds ~ RRate$avg_), col="red")

legend("topright", bty="n", legend=paste("R2 is",
format(summary(fit)$adj.r.squared, digits=4)))

}
```

Visualizations & variable transformation




```
> fn_visualize(LR_DF, "Target", "HP_Imputed")
> fn_visualize(LR_DF, "Target", "No_OF_CR_TXNS")
> fn_visualize(LR_DF, "Target", "No_OF_CR_TXNS", ln_trnfm = 1)
```






```
> LR_DF$No_OF_CR_TXNS_ln <- log(LR_DF$No_OF_CR_TXNS + 1)
```

K2 Analytics

Building Skills, Building Individuals

Model Development

Creating Development & Validation Samples

```
> mydata <- LR_DF
> mydata$random <- runif(nrow(mydata), 0, 1)
> mydata.dev <- mydata[which(mydata$random <= 0.7),]
> mydata.val <- mydata[which(mydata$random > 0.7),]
> nrow(mydata.dev)
[1] 14021
> nrow(mydata.val)
[1] 5979
```

Note: Though in the above step we have created Dev – Val sample… but in subsequent slides I am running the regression on the full base. In practice we should build the model on Development Sample and validate it on the Validation Sample

Modeling – Running Regression

```
> library(aod)
> library(ggplot2)
> mydata <- LR_DF
> mylogit <- glm(
+ Target ~ No_OF_CR_TXNS_ln + HP_Imputed,
+ data = mydata, family = "binomial"
+ )
> summary(mylogit)

Call:
glm(formula = Target ~ No_OF_CR_TXNS_ln + HP_Imputed, family = "binomial",
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6813  -0.3417  -0.2500  -0.1763   3.2298

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -4.376697   0.159697  -27.41   <2e-16 ***
No_OF_CR_TXNS_ln  0.771705   0.049860   15.48   <2e-16 ***
HP_Imputed       -0.065168   0.004195  -15.54   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7267.4  on 19999  degrees of freedom
Residual deviance: 6703.2  on 19997  degrees of freedom
AIC: 6709.2

Number of Fisher Scoring iterations: 6
```

Check VIF

```
> ##install.packages("car")
> library(car)
> vif(mylogit)
                          GVIF Df GVIF^(1/(2*Df))
No_OF_CR_TXNS_ln 1.005388  1        1.002691
HP_Imputed       1.000785  1        1.000392
AGE_BKT          1.006167  6        1.000512
```

Computing Probability

```
> ## Calculating the probabilities
> mydata$prob <- predict(mylogit, mydata, type="response")
>
> ## Creating Deciles for Rank Ordering Test
> mydata$deciles <- decile(mydata$prob)
```

Rank Ordering

```
#### Rank Ordering Table #####
library(data.table)
mydata.DT = data.table(mydata)  ## Converting the data frame to data table object

## Creating Aggregation and Group By similar to as in SQL
rank <- mydata.DT[, list(
min_prob = min(prob),
max_prob = max(prob),
cnt = length(Target),
cnt_resp = sum(Target),
cnt_non_resp = sum(Target == 0),
) ,
by=deciles]
[order(-deciles)]

rank$RRate <- rank$cnt_resp / rank$cnt                   ## computing response rate
rank$cum_resp <- cumsum(rank$cnt_resp)                   ## computing cum responders
rank$cum_non_resp <- cumsum(rank$cnt_non_resp) ## computing cum non-responders
rank$cum_rel_resp <- rank$cum_resp / sum(rank$cnt_resp);
rank$cum_rel_non_resp <- rank$cum_non_resp / sum(rank$cnt_non_resp);
rank$ks <- rank$cum_rel_resp - rank$cum_rel_non_resp; ## KS

rank    ## display Rank Ordering Table
```

Rank Ordering Table

```
> rank
    deciles    min_prob    max_prob  cnt cnt_resp cnt_non_resp       RRate cum_resp cum_non_resp cum_rel_resp cum_rel_non_resp         ks
 1:      10 0.096058166 0.207101408 2001      252         1749 0.125937031      252         1749    0.2837838       0.09151319 0.19227060
 2:       9 0.069152571 0.096050738 2004      177         1827 0.088323353      429         3576    0.4831081       0.18710758 0.29600053
 3:       8 0.053393656 0.069127423 1997      108         1889 0.054081122      537         5465    0.6047297       0.28594600 0.31878373
 4:       7 0.041658477 0.053354432 2017      100         1917 0.049578582      637         7382    0.7173423       0.38624948 0.33109287
 5:       6 0.033136869 0.041651957 1996       85         1911 0.042585170      722         9293    0.8130631       0.48623901 0.32682405
 6:       5 0.026523662 0.033111376 1987       48         1939 0.024157021      770        11232    0.8671171       0.58769360 0.27942352
 7:       4 0.020740034 0.026511226 2001       49         1952 0.024487756      819        13184    0.9222973       0.68982838 0.23246892
 8:       3 0.015116338 0.020552837 2005       35         1970 0.017456359      854        15154    0.9617117       0.79290498 0.16880673
 9:       2 0.009626664 0.015107064 1994       23         1971 0.011534604      877        17125    0.9876126       0.89603391 0.09157871
10:       1 0.001663944 0.009617002 1998       11         1987 0.005505506      888        19112    1.0000000       1.00000000 0.00000000
```

ROC Curve & KS

```
> ### Calculating ROC Curve and KS for the model
> ##install.packages("ROCR")
> library(ROCR)
> pred <- prediction(mydata$prob, mydata$Target)
> perf <- performance(pred, "tpr", "fpr")
> plot(perf)
> KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
> KS
[1] 0.3560835
```

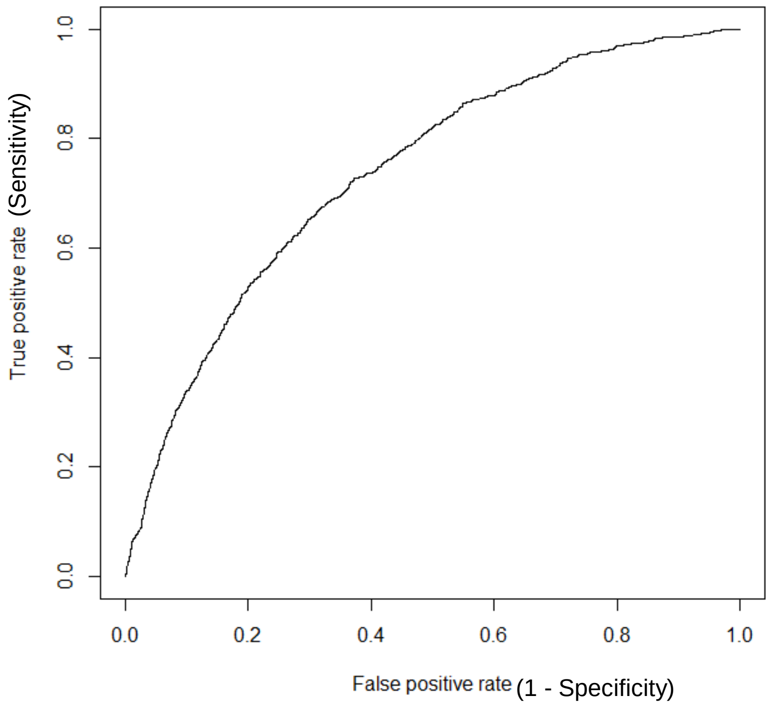
Classification Matrix		**Actual**	
		Y	N
Predictive	Y	a	c
	N	b	d

True Positive Rate = True Positive / Total Positive

= a / (a + b)

Specificity = True Negative / Total Negative

= d / (c + d)



Checking Chi Sq - Goodness of Fit

```
######### GOODNESS OF FIT ############################
hosmerlem <-
 function (y, yhat, g = 10)
 {
    cutyhat <- cut(yhat, breaks = quantile(yhat, probs = seq(0,
       1, 1/g)), include.lowest = T)
    obs <- xtabs(cbind(1 - y, y) ~ cutyhat)
    expect <- xtabs(cbind(1 - yhat, yhat) ~ cutyhat)
    chisq <- sum((obs - expect)^2/expect)
    P <- 1 - pchisq(chisq, g - 2)
    c("X^2" = chisq, Df = g - 2, "P(>Chi)" = P)
 }
```

```
> hl_gof = hosmerlem(mydata$Target, mydata$prob )
> hl_gof
      X^2        Df   P(>Chi)
9.3333181 8.0000000 0.3149625
```

Chi-Sq Calculation

```
## install.packages("sqldf")

library(sqldf)
sqldf ("select deciles, count(1) as cnt,
sum (Target) as Obs_Resp, count (Target == 0) as Obs_Non_Resp,
sum (prob) as Exp_Resp, sum (1 - prob) as Exp_Non_Resp
from mydata
group by deciles
order by deciles desc")
```

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

O = the frequencies observed

E = the frequencies expected

$\sum$ *= the 'sum of'*

deciles	cnt	Obs_Resp	Obs_Non_Resp	Exp_Resp	Exp_Non_Resp
10	2001	252	2001	258.01295	1742.987
9	2004	177	2004	162.10475	1841.895
8	1997	108	1997	120.98726	1876.013
7	2017	100	2017	94.61875	1922.381
6	1996	85	1996	73.87535	1922.125
5	1987	48	1987	59.13513	1927.865
4	2001	49	2001	46.62922	1954.371
3	2005	35	2005	35.51307	1969.487
2	1994	23	1994	24.67565	1969.324
1	1998	11	1998	12.44787	1985.552

Gini Index

```
> library(ineq)
> gini = ineq(mydata$prob, type="Gini")
> gini
[1] 0.4509204
```

The Gini coefficient measures the inequality among values of a frequency distribution (for example levels of income). A Gini coefficient of zero expresses perfect equality, where all values are the same (for example, where everyone has the same income). A Gini coefficient of one (or 100%) expresses maximal inequality among values (for example where only one person has all the income).

Code for Calculating Concordance

```
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE***#
concordance=function(y, yhat)
{
Con_Dis_Data = cbind(y, yhat)
ones = Con_Dis_Data[Con_Dis_Data[,1] == 1,]
zeros = Con_Dis_Data[Con_Dis_Data[,1] == 0,]
conc=matrix(0, dim(zeros)[1], dim(ones)[1])
disc=matrix(0, dim(zeros)[1], dim(ones)[1])
ties=matrix(0, dim(zeros)[1], dim(ones)[1])
for (j in 1:dim(zeros)[1])
{
for (i in 1:dim(ones)[1])
{
if (ones[i,2]>zeros[j,2])
{conc[j,i]=1}
else if (ones[i,2]<zeros[j,2])
{disc[j,i]=1}
else if (ones[i,2]==zeros[j,2])
{ties[j,i]=1}
}
}
Pairs=dim(zeros)[1]*dim(ones)[1]
PercentConcordance=(sum(conc)/Pairs)*100
PercentDiscordance=(sum(disc)/Pairs)*100
PercentTied=(sum(ties)/Pairs)*100
return(list("Percent Concordance"=PercentConcordance,"Percent Discordance"=PercentDiscordance,"Percent Tied"=PercentTied,"Pairs"=Pairs))
}
#***FUNCTION TO CALCULATE CONCORDANCE AND DISCORDANCE ENDS***#
```

Concordance

```
> concordance_output = concordance(mydata$Target, mydata$prob)
> concordance_output
$`Percent Concordance`
[1] 73.79235

$`Percent Discordance`
[1] 26.19606

$`Percent Tied`
[1] 0.01159005

$Pairs
[1] 16971456
```

K2 Analytics
Building Skills, Building Individuals

Thank you