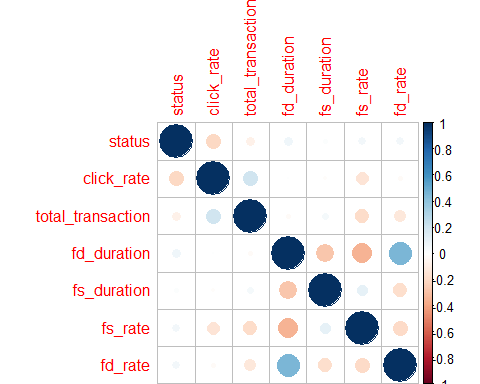
lift\_ks\_statistics.R

PM00450108

Mon Jun 05 20:05:26 2017

#lodaing the data :-  
data = read.csv("D:/prd\_data.csv")  
#running the model:-  
#lets see some correlation among data points  
library(corrplot)  
M = cor(data)  
corrplot(M)



# we now remove variables with high correlation  
dat1 = subset(data,select=-c(fd\_rate,fs\_rate))  
model = glm(status~.,data=dat1,family=binomial(logit))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# #lets check model summary()  
summary(model)

##   
## Call:  
## glm(formula = status ~ ., family = binomial(logit), data = dat1)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -8.4904 -1.1935 0.8611 1.0402 3.0378   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 4.437e-01 1.151e-01 3.856 0.000115 \*\*\*  
## click\_rate -2.742e-01 2.029e-02 -13.514 < 2e-16 \*\*\*  
## total\_transaction -1.684e-04 3.667e-05 -4.592 4.39e-06 \*\*\*  
## fd\_duration 1.050e+00 1.081e-01 9.715 < 2e-16 \*\*\*  
## fs\_duration 8.705e-01 1.029e-01 8.457 < 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: 7584.6 on 5484 degrees of freedom  
## Residual deviance: 7141.6 on 5480 degrees of freedom  
## AIC: 7151.6  
##   
## Number of Fisher Scoring iterations: 5

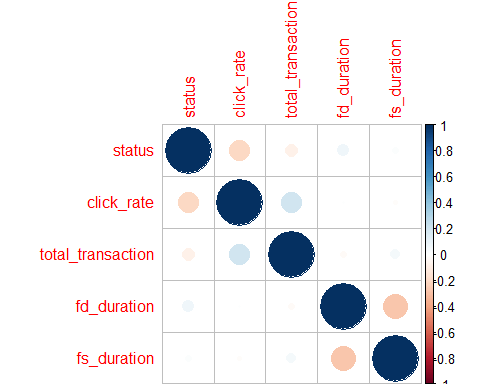
table(dat1$status)

##   
## 0 1   
## 2580 2905

M1 = cor(dat1)  
corrplot(M1)  
  
#predictions:-  
data$prob = predict(model,data=dat1,type="response")  
data$pred = ifelse(data$prob>0.5,1,0)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2



#Metrics:-  
#1.confusion Matrix:-  
confusionMatrix(data$pred,data$status)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1129 594  
## 1 1451 2311  
##   
## Accuracy : 0.6272   
## 95% CI : (0.6142, 0.64)  
## No Information Rate : 0.5296   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2375   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.4376   
## Specificity : 0.7955   
## Pos Pred Value : 0.6553   
## Neg Pred Value : 0.6143   
## Prevalence : 0.4704   
## Detection Rate : 0.2058   
## Detection Prevalence : 0.3141   
## Balanced Accuracy : 0.6166   
##   
## 'Positive' Class : 0   
##

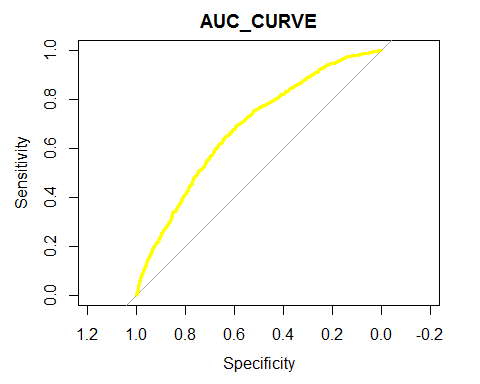
#2 AUC / ROC:-  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

plot(roc(data$status,data$prob, direction="<"),  
 col="yellow", lwd=3, main="AUC\_CURVE")



roc(data$status,data$prob, direction="<")

##   
## Call:  
## roc.default(response = data$status, predictor = data$prob, direction = "<")  
##   
## Data: data$prob in 2580 controls (data$status 0) < 2905 cases (data$status 1).  
## Area under the curve: 0.6819

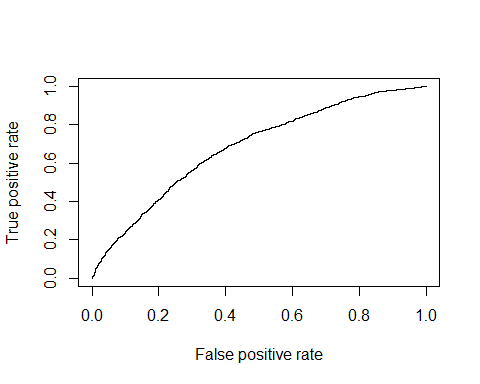
#AUC\_CODE VERSION\_2:-  
library('ROCR')

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

p <- predict(model, newdata=dat1, type="response")  
pr <- prediction(p, dat1$status)  
prf <- performance(pr, measure = "tpr", x.measure = "fpr")  
plot(prf)



auc <- performance(pr, measure = "auc")  
auc <- auc@y.values[[1]]  
auc

## [1] 0.6819149

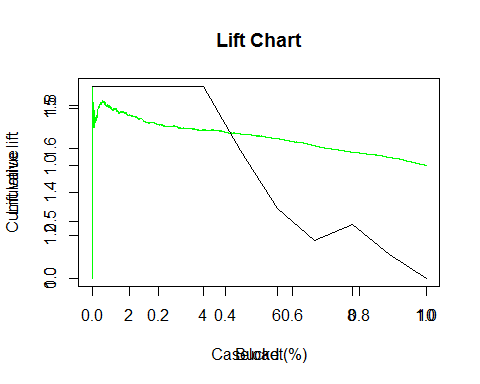
#Cocnrodance\_Discordance\_code:-  
  
###########################################################  
bruteforce<-function(model){  
 # Get all actual observations and their fitted values into a frame  
 fitted<-data.frame(cbind(model$y,model$fitted.values))  
 colnames(fitted)<-c('respvar','score')  
 # Subset only ones  
 ones<-fitted[fitted[,1]==1,]  
 # Subset only zeros  
 zeros<-fitted[fitted[,1]==0,]  
   
 # Initialise all the values  
 pairs\_tested<-0  
 conc<-0  
 disc<-0  
 ties<-0  
   
 # Get the values in a for-loop  
 for(i in 1:nrow(ones))  
 {  
 for(j in 1:nrow(zeros))  
 {  
 pairs\_tested<-pairs\_tested+1  
 if(ones[i,2]>zeros[j,2]) {conc<-conc+1}  
 else if(ones[i,2]==zeros[j,2]){ties<-ties+1}  
 else {disc<-disc+1}  
 }  
 }  
 # Calculate concordance, discordance and ties  
 concordance<-conc/pairs\_tested  
 discordance<-disc/pairs\_tested  
 ties\_perc<-ties/pairs\_tested  
 return(list("Concordance"=concordance,  
 "Discordance"=discordance,  
 "Tied"=ties\_perc,  
 "Pairs"=pairs\_tested))  
}  
  
  
# Let us see concordance values:-  
  
  
  
# LIFT CURVE:-  
#Now lets plot the lift chart for further insights in the data:-  
  
predtrain=prediction(data$prob,data$status)  
lifttrain=performance(predtrain, "lift", "rpp")  
plot(lifttrain, col="green", lty=1, xlab="Caseload (%)", add=FALSE,main="Lift Chart")  
par(new=TRUE)   
  
#Decile Table:-  
lift <- function(depvar, predcol, groups=10) {  
   
 library(dplyr)  
 if(is.factor(depvar)) depvar <- as.integer(as.character(depvar))  
 if(is.factor(predcol)) predcol <- as.integer(as.character(predcol))  
 helper = data.frame(cbind(depvar, predcol))  
 helper[,"bucket"] = ntile(-helper[,"predcol"], groups)  
 gaintable = helper %>% group\_by(bucket) %>%  
 summarise\_at(vars(depvar), funs(total = n(),  
 totalresp=sum(., na.rm = TRUE))) %>%  
 mutate(Cumresp = cumsum(totalresp),  
 Gain=Cumresp/sum(totalresp)\*100,  
 Cumlift=Gain/(bucket\*(100/groups)))  
 return(gaintable)  
}  
  
dt = lift(data$status , data$pred, groups = 10)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

graphics::plot(dt$bucket, dt$Cumlift, type="l", ylab="Cumulative lift", xlab="Bucket")



dt

## # A tibble: 10 × 6  
## bucket total totalresp Cumresp Gain Cumlift  
## <dbl> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 1 549 549 549 18.89845 1.889845  
## 2 2 548 548 1097 37.76248 1.888124  
## 3 3 549 549 1646 56.66093 1.888698  
## 4 4 548 548 2194 75.52496 1.888124  
## 5 5 549 117 2311 79.55250 1.591050  
## 6 6 548 0 2311 79.55250 1.325875  
## 7 7 549 78 2389 82.23752 1.174822  
## 8 8 548 516 2905 100.00000 1.250000  
## 9 9 549 0 2905 100.00000 1.111111  
## 10 10 548 0 2905 100.00000 1.000000

#KS\_STATISTICS:-  
KS <- function(pred,depvar){  
 require("ROCR")  
 p <- prediction(as.numeric(pred),depvar)  
 perf <- performance(p, "tpr", "fpr")  
 ks <- max(attr(perf, "y.values")[[1]] - (attr(perf, "x.values")[[1]]))  
 return(ks)  
}  
  
KS(data$prob,data$status)

## [1] 0.2819197

#Now lets see how far are predicted points from actual values:-  
  
  
  
  
  
  
  
#LIFT\_CHART OWN CODE:-  
#1 rankorder the prob in 1 table:-  
lf = subset(data,select=c(status,prob))  
library(dplyr)  
lf1=arrange(lf,desc(prob))  
  
library(gains)  
gains.cross <- gains(actual=data$status,  
 predicted=data$prob,  
 groups=5)  
#SOME MORE METRICS:-  
  
library("MLmetrics")

##   
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':  
##   
## Recall

#AUC CURVE:-  
AUC(data$pred,data$status)

## [1] 0.6165609

#COnfusion\_matrix:-  
ConfusionMatrix(data$pred,data$status)

## y\_pred  
## y\_true 0 1  
## 0 1129 1451  
## 1 594 2311

#F1\_score:-  
F1\_Score(data$pred,data$status)

## [1] 0.5247502

#Gini  
Gini(data$prob,data$status)

## [1] 0.3638301

#KS\_stats:-  
KS\_Stat(data$prob,data$status)

## [1] 28.19197

#Area\_under lift chart:-  
LiftAUC(data$prob,data$status)

## [1] 1.249584

#LOG\_LOSS/CROSS\_ENTROPY LOSS  
LogLoss(data$prob,data$status)

## [1] 0.6507349