Challenger Logistic Regression

Lei Deng, ti7597, STAT6620

May 18, 2017

## Step 1 - Colleting data

Read the data to the R

# load the necessary library  
library(Amelia)

## Loading required package: Rcpp

## ##   
## ## Amelia II: Multiple Imputation  
## ## (Version 1.7.4, built: 2015-12-05)  
## ## Copyright (C) 2005-2017 James Honaker, Gary King and Matthew Blackwell  
## ## Refer to http://gking.harvard.edu/amelia/ for more information  
## ##

library(ROCR)

## Loading required package: gplots

##   
## Attaching package: 'gplots'

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

launch <- read.csv("challenger.csv")

## Step 2 - Exploring and preparing the data

# examine the launch data  
str(launch)

## 'data.frame': 23 obs. of 4 variables:  
## $ distress\_ct : int 0 1 0 0 0 0 0 0 1 1 ...  
## $ temperature : int 66 70 69 68 67 72 73 70 57 63 ...  
## $ field\_check\_pressure: int 50 50 50 50 50 50 100 100 200 200 ...  
## $ flight\_num : int 1 2 3 4 5 6 7 8 9 10 ...

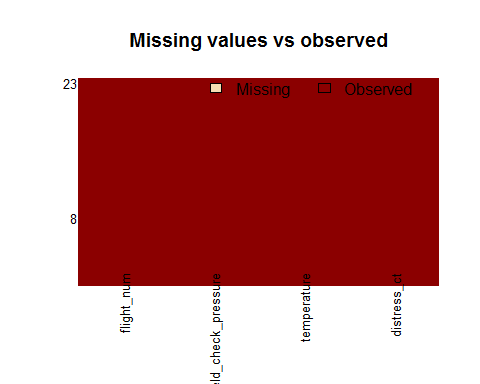
# First recode the distress\_ct variable into 0 and 1, making 1 to represent at least one failure during a launch.  
  
launch$distress\_ct = ifelse(launch$distress\_ct<1,0,1)  
launch$distress\_ct

## [1] 0 1 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1

# set up trainning and test data sets  
  
indx = sample(1:nrow(launch), as.integer(0.9\*nrow(launch)))  
indx

## [1] 23 5 7 9 1 3 11 12 20 17 10 19 4 16 18 14 15 22 8 2

launch\_train = launch[indx,]  
launch\_test = launch[-indx,]  
  
launch\_train\_labels = launch[indx,4]  
launch\_test\_labels = launch[-indx,4]   
  
# Check if there are any missing values  
  
missmap(launch, main = "Missing values vs observed")



# number of missing values in each column  
  
sapply(launch,function(x) sum(is.na(x)))

## distress\_ct temperature field\_check\_pressure   
## 0 0 0   
## flight\_num   
## 0

# number of unique values in each column  
  
sapply(launch, function(x) length(unique(x)))

## distress\_ct temperature field\_check\_pressure   
## 2 16 3   
## flight\_num   
## 23

## Step3 - Training a model on the data

# fit the logistic regression model, with all predictor variables  
  
model <- glm(distress\_ct ~.,family=binomial(link='logit'),data=launch\_train)  
model

##   
## Call: glm(formula = distress\_ct ~ ., family = binomial(link = "logit"),   
## data = launch\_train)  
##   
## Coefficients:  
## (Intercept) temperature field\_check\_pressure   
## 26.98027 -0.43380 0.03994   
## flight\_num   
## -0.38808   
##   
## Degrees of Freedom: 19 Total (i.e. Null); 16 Residual  
## Null Deviance: 24.43   
## Residual Deviance: 10.5 AIC: 18.5

summary(model)

##   
## Call:  
## glm(formula = distress\_ct ~ ., family = binomial(link = "logit"),   
## data = launch\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2129 -0.4790 -0.1546 0.0497 2.1323   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 26.98027 16.95322 1.591 0.1115   
## temperature -0.43380 0.25262 -1.717 0.0859 .  
## field\_check\_pressure 0.03994 0.02806 1.423 0.1546   
## flight\_num -0.38808 0.30755 -1.262 0.2070   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 24.435 on 19 degrees of freedom  
## Residual deviance: 10.496 on 16 degrees of freedom  
## AIC: 18.496  
##   
## Number of Fisher Scoring iterations: 7

anova(model, test="Chisq")

## Analysis of Deviance Table  
##   
## Model: binomial, link: logit  
##   
## Response: distress\_ct  
##   
## Terms added sequentially (first to last)  
##   
##   
## Df Deviance Resid. Df Resid. Dev Pr(>Chi)   
## NULL 19 24.435   
## temperature 1 11.0288 18 13.406 0.0008971 \*\*\*  
## field\_check\_pressure 1 0.7503 17 12.655 0.3863782   
## flight\_num 1 2.1598 16 10.496 0.1416596   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

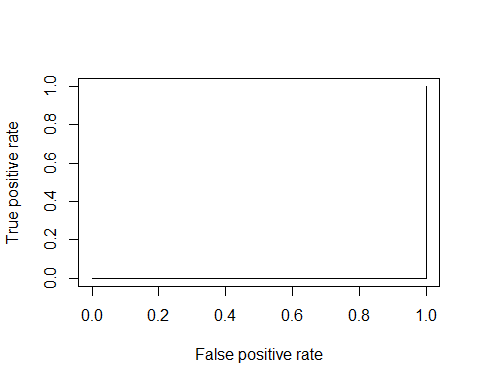
## step4 - Evaluating model performance

# check Accuracy  
  
fitted.results <- predict(model,newdata=launch\_test,type='response')  
fitted.results <- ifelse(fitted.results > 0.5,1,0)  
  
misClasificError <- mean(fitted.results != launch\_test$distress\_ct)  
print(paste('Accuracy',1-misClasificError))

## [1] "Accuracy 0.333333333333333"

## step5 - Improving model performance

# Because this data set is so small, it is possible that the test data set  
# does not contain both 0 and 1 values. If this happens the code will not  
# run. And since the test data set is so small the ROC is not useful here  
# but the code is provided.  
  
p <- predict(model, newdata=launch\_test, type="response")  
pr <- prediction(p, launch\_test$distress\_ct)  
prf <- performance(pr, measure = "tpr", x.measure = "fpr")  
plot(prf)



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

## [1] 0