library(plyr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

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

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

library(stringr)  
library(ggplot2)  
library(usdm)

## Loading required package: sp

## Loading required package: raster

##   
## Attaching package: 'raster'

## The following object is masked from 'package:dplyr':  
##   
## select

library(corrplot)

## corrplot 0.84 loaded

library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:raster':  
##   
## mask, zoom

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:plyr':  
##   
## is.discrete, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

library(FSelector)  
library(ROSE)

## Loaded ROSE 0.0-3

library(caret)

##   
## Attaching package: 'caret'

## The following object is masked from 'package:survival':  
##   
## cluster

library(pROC)

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

##   
## Attaching package: 'pROC'

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

library(class)   
library(descr)

##   
## Attaching package: 'descr'

## The following objects are masked from 'package:raster':  
##   
## crosstab, freq

library(nortest)

setwd("C:/Users/Kevin.Phan/Desktop/Capstone")  
DF1 <- read.csv("DSI\_kickstarterscrape\_dataset.csv")  
DF2 <- read.csv("MasterKickstarter.csv")  
DF3 <- read.csv("ks-projects-201612.csv")  
DF4 <- read.csv("18k\_Projects.csv")

#DATA CLEANSING\*\*\*  
#Merging datasets based on left join.   
DF3 <- DF3[,c(1,8)]  
master\_data <- join(DF1,DF2, by = "ID", type = "left")  
master\_data <- join(master\_data,DF3, by = "ID", type = "left")  
master\_data <- master\_data[complete.cases(master\_data),]  
master\_data <- master\_data[,unique(names(master\_data))]  
duplicated(colnames(master\_data))

## [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [23] FALSE FALSE

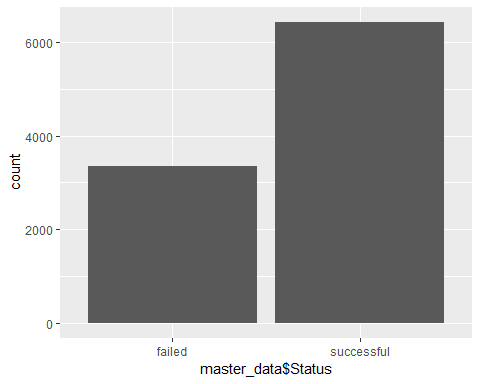
str(master\_data) #check for variable types. Change where meeded.

## 'data.frame': 10785 obs. of 24 variables:  
## $ ID : int 237090 610918 735377 1122138 1167521 1397716 1615245 1841780 2085232 2140106 ...  
## $ Name : Factor w/ 45754 levels "'2010' The Film",..: 16298 36646 38824 35252 34534 1226 19954 15325 42008 1229 ...  
## $ Main.Category : Factor w/ 14 levels "Art","Comics",..: 6 1 10 12 12 12 12 6 12 12 ...  
## $ Sub.Category : Factor w/ 51 levels "Animation","Art",..: 16 25 26 19 19 3 19 50 19 19 ...  
## $ Status : Factor w/ 5 levels "canceled","failed",..: 4 4 4 3 4 4 2 3 2 3 ...  
## $ Goal : num 6000 7500 20000 5000 1500 8750 2000 45000 12000 10000 ...  
## $ Pledged : int 6535 9836 20138 50 3511 8805 135 2726 50 1425 ...  
## $ Funded.Percentage : num 1.09 1.31 1.01 0.01 2.34 ...  
## $ Backers : int 100 255 115 1 114 139 5 68 2 38 ...  
## $ Funded.Date : Factor w/ 41068 levels "Fri, 01 Apr 2011 00:14:52 -0000",..: 20397 31165 2096 15546 7367 35605 29144 33926 5315 1403 ...  
## $ levels : int 13 10 21 1 8 12 4 9 4 9 ...  
## $ Updates : int 4 6 8 0 31 12 0 8 2 7 ...  
## $ Comments : int 0 5 12 0 7 12 0 5 0 1 ...  
## $ Duration : num 32.2 35.3 45 30 21 ...  
## $ City : Factor w/ 3211 levels "Aachen","Aalborg",..: 1626 506 114 1115 2789 2292 1312 26 2650 303 ...  
## $ Deadline : Factor w/ 2954 levels "1/01/10","1/01/11",..: 1507 500 827 2070 1248 226 2355 2094 1889 2233 ...  
## $ Created\_At : Factor w/ 2963 levels "1/01/10","1/01/11",..: 976 2888 383 1681 1032 721 1931 1517 1581 1648 ...  
## $ Staff\_Pick : logi FALSE TRUE FALSE FALSE FALSE FALSE ...  
## $ spotLight : logi TRUE TRUE TRUE TRUE TRUE TRUE ...  
## $ Pledge\_per\_person : int 65 38 175 50 30 63 27 58 25 36 ...  
## $ Population : int 3877129 2841952 23467 17106 324465 540513 773283 208414 103760 571281 ...  
## $ Backers\_as\_Prct\_of\_Pop : num 34.3517 14.2844 17.6887 0.0175 4.439 ...  
## $ Days\_spent\_making\_campign: int 32 5 7 15 3 0 4 8 14 9 ...  
## $ Launched : Factor w/ 295793 levels "100","1000","10000",..: 43623 29823 9013 52336 41166 34968 4154 48968 50887 50972 ...

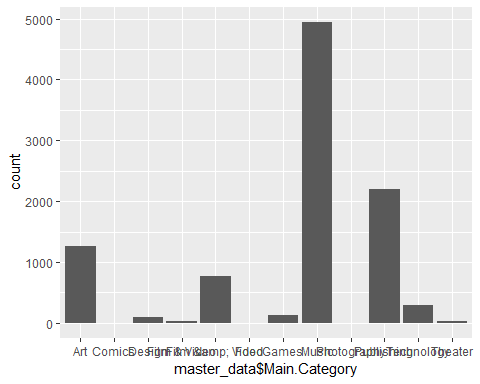
master\_data$Deadline <- as.Date(as.character(master\_data$Deadline),"%m/%d/%y")  
master\_data$Launched <- substr(master\_data$Launched,1,10)  
master\_data$Launched <- as.Date(as.character(master\_data$Launched),"%Y-%m-%d")  
master\_data$Created\_At <- as.Date(as.character(master\_data$Created\_At),"%m/%d/%y")  
master\_data <- master\_data[,c(1:16,24,17:23)]  
master\_data$Name <- as.character(master\_data$Name)  
master\_data <- master\_data[master\_data$Status != "live",]  
master\_data <- master\_data[master\_data$Status != "canceled",]  
master\_data$Status <- factor(master\_data$Status)#We will not deal with live and cancelled in this analysis.

numericAtt <- vector("numeric",10L)  
for (i in 1:ncol(master\_data)) {  
 if (class(master\_data[,i]) == "numeric" || class(master\_data[,i]) == "integer" ){numericAtt[i] = i}  
 else {numericAtt[i] = NA}  
}  
numericAtt <- numericAtt[!is.na(numericAtt)][-1]

#We will now visualize imbalanaces in Status and see Category Frequencies  
ggplot(data.frame(master\_data$Status),aes(x=master\_data$Status)) + geom\_bar()



ggplot(data.frame(master\_data$Main.Category),aes(x=master\_data$Main.Category)) + geom\_bar()

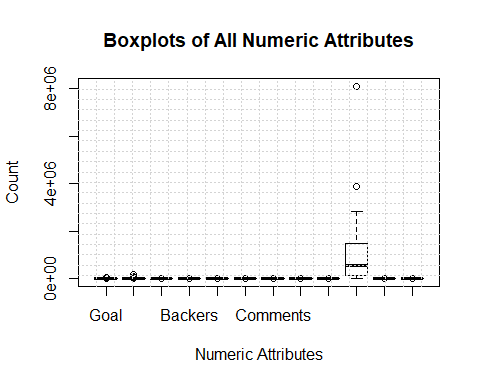


#We see there is way more successful then failed. and category of music is alot more than others. We will fix this in the next step  
#There are few major outliers that we can afford to remove. We will use mahalanobis distance.Multivariate

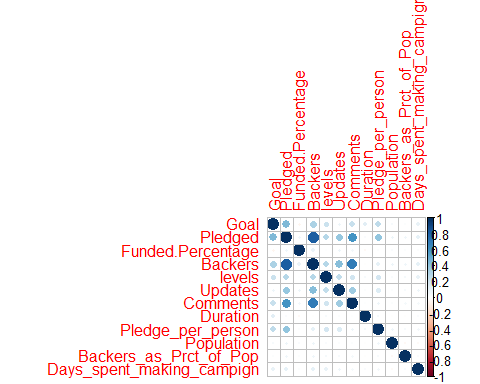
MD <- mahalanobis(master\_data[,c(numericAtt)], colMeans(master\_data[,c(numericAtt)]),cov(master\_data[,c(numericAtt)]),tol=1e-20)  
master\_data$MD <- round(MD,3)  
master\_data$Outlier\_Mahalanobis <- "No"  
master\_data$Outlier\_Mahalanobis[master\_data$MD > 12] <- "Yes" #Threshold i did chose was 9.   
master\_data <- master\_data[master\_data$Outlier\_Mahalanobis == "No",]  
attach(master\_data)

## The following object is masked \_by\_ .GlobalEnv:  
##   
## MD

boxplot(master\_data[numericAtt], xlab = "Numeric Attributes", ylab = "Count", main = "Boxplots of All Numeric Attributes")  
grid(20,20, col = "lightgray", lty = "dotted",lwd = par("lwd"), equilogs = TRUE) #population has outliers does it matter



Correlations <- cor(master\_data[,numericAtt])  
corrplot(Correlations) #We see there are a few highly correlated variables. We will use feature selection.



#Remove NA  
#Dealing with imbalance   
table(master\_data$Status)

##   
## failed successful   
## 2961 5375

Balanced\_Data <- ovun.sample(Status ~ ., data = master\_data, method = "both",p = 0.5)$data #Utilizes both over and under sampling. \*\* DEBUG ROSE\*\*  
table(Balanced\_Data$Status)

##   
## successful failed   
## 4148 4170

#Removing Collinear variables  
vifselection <- vif(master\_data[,numericAtt])

#Test for nomality  
lapply(Balanced\_Data[,numericAtt], ad.test) #Anderson Darling Test

## $Goal  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 703.07, p-value < 2.2e-16  
##   
##   
## $Pledged  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 1056.9, p-value < 2.2e-16  
##   
##   
## $Funded.Percentage  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 2041.4, p-value < 2.2e-16  
##   
##   
## $Backers  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 844.44, p-value < 2.2e-16  
##   
##   
## $levels  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 60.476, p-value < 2.2e-16  
##   
##   
## $Updates  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 586.92, p-value < 2.2e-16  
##   
##   
## $Comments  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 1507.2, p-value < 2.2e-16  
##   
##   
## $Duration  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 408.11, p-value < 2.2e-16  
##   
##   
## $Pledge\_per\_person  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 294.46, p-value < 2.2e-16  
##   
##   
## $Population  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 1094.3, p-value < 2.2e-16  
##   
##   
## $Backers\_as\_Prct\_of\_Pop  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 438.15, p-value < 2.2e-16  
##   
##   
## $Days\_spent\_making\_campign  
##   
## Anderson-Darling normality test  
##   
## data: X[[i]]  
## A = 836.76, p-value < 2.2e-16

lapply(Balanced\_Data[1:5000,numericAtt], shapiro.test) # Shapiro-Wilk Test

## $Goal  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.68243, p-value < 2.2e-16  
##   
##   
## $Pledged  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.56506, p-value < 2.2e-16  
##   
##   
## $Funded.Percentage  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.050617, p-value < 2.2e-16  
##   
##   
## $Backers  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.65065, p-value < 2.2e-16  
##   
##   
## $levels  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.97288, p-value < 2.2e-16  
##   
##   
## $Updates  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.85079, p-value < 2.2e-16  
##   
##   
## $Comments  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.44298, p-value < 2.2e-16  
##   
##   
## $Duration  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.8889, p-value < 2.2e-16  
##   
##   
## $Pledge\_per\_person  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.85427, p-value < 2.2e-16  
##   
##   
## $Population  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.65143, p-value < 2.2e-16  
##   
##   
## $Backers\_as\_Prct\_of\_Pop  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.85238, p-value < 2.2e-16  
##   
##   
## $Days\_spent\_making\_campign  
##   
## Shapiro-Wilk normality test  
##   
## data: X[[i]]  
## W = 0.68817, p-value < 2.2e-16

#Both tests have p values under 0.05. Therefore, we can reject the null hypothesis of normality.

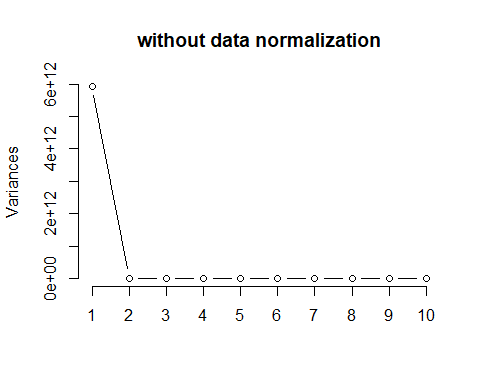
#further Feature Selection via information gain  
weights <- information.gain(Status~., master\_data[,numericAtt])  
print(weights)

## attr\_importance  
## Goal 0.027141097  
## Pledged 0.330716358  
## Funded.Percentage 0.650610125  
## Backers 0.334269882  
## levels 0.014698774  
## Updates 0.146257237  
## Comments 0.105057166  
## Duration 0.011166285  
## Pledge\_per\_person 0.086821714  
## Population 0.005590419  
## Backers\_as\_Prct\_of\_Pop 0.012903117  
## Days\_spent\_making\_campign 0.002115807

subset <- cutoff.k(weights, 4) #mean is 0.13  
f <- as.simple.formula(subset, "Status")  
print(f)

## Status ~ Funded.Percentage + Backers + Pledged + Updates  
## <environment: 0x17105ca8>

#Feature PCA  
PCA <- prcomp(master\_data[,numericAtt], scale = FALSE, center= FALSE)  
plot(PCA, type = "l", main = 'without data normalization')



#Seems only one principal component is all it needs. Better to go with info gain with all the vairables above the average value.

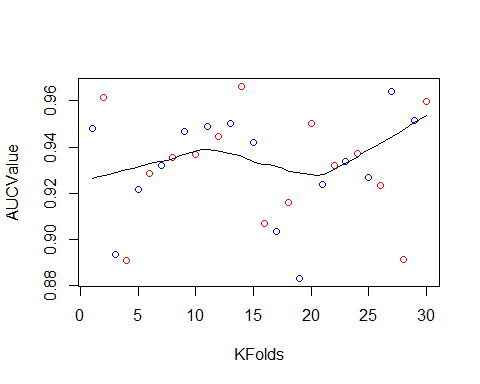
#Logistic Regression using cross fold for partitioning  
#Created a new dataset for the logisitc regression. I created a loop to test the model under different amounts of trianing data folds.   
Balanced\_DataLOG <- Balanced\_Data %>% mutate\_if(is.numeric, scale) #Scaling Balanced Data Set  
Balanced\_DataLOG<-Balanced\_DataLOG[sample(nrow(Balanced\_DataLOG)),]  
folds <- cut(seq(1,nrow(Balanced\_DataLOG)),breaks=100,labels=FALSE)  
AUCValue <- vector("numeric",10L)  
#Perform 10 fold cross validation on our first model: Logistic Regression  
for(i in 1:30){  
 #Segement your data by fold using the which() function   
 testIndexes <- which(folds==i,arr.ind=TRUE)  
 testData <- Balanced\_DataLOG[testIndexes, ]  
 trainData <- Balanced\_DataLOG[-testIndexes, ]  
 Mod <- glm(Status ~ Pledged + Backers + Updates + Comments, family = "binomial", data = trainData)  
 pred <- predict(Mod,testData, type = "response")  
 RocVal <- roc(testData$Status,pred)  
 AUCValue[i] <- auc(RocVal)  
}

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

AUCValue

## [1] 0.9481481 0.9616725 0.8933002 0.8907563 0.9215116 0.9287749 0.9318182  
## [8] 0.9356725 0.9468085 0.9366279 0.9488837 0.9447293 0.9503546 0.9662791  
## [15] 0.9418182 0.9069767 0.9036643 0.9160522 0.8831395 0.9500581 0.9236597  
## [22] 0.9318182 0.9336075 0.9370748 0.9268293 0.9233449 0.9639480 0.8912791  
## [29] 0.9515723 0.9597506

KFolds <- c(1:30)  
scatter.smooth(KFolds,AUCValue, col = c("Blue","red")) #We seethe same pattern. I say about 4 folds is optimal.



#Logistic Regression using optimal Percentage Split  
set.seed(100)  
smp\_size <- floor(0.75 \* nrow(Balanced\_Data))  
train\_ind <- sample(seq\_len(nrow(Balanced\_DataLOG)), size = smp\_size)  
train <- Balanced\_DataLOG[train\_ind, ]  
test <- Balanced\_DataLOG[-train\_ind, ]  
Mod <- glm(Status ~ Pledged + Backers + Updates + Comments + Pledge\_per\_person, family = "binomial", data = train)

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

predPercSplit <- predict(Mod,test, type = "response")  
auc(roc(test$Status,predPercSplit)) #0.922 roc and auc give different auc vals?

## Area under the curve: 0.9284

#8 fold cv is better and so we will use this.

#KNN  
#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
#Standardize  
Balanced\_DataKNN <- Balanced\_Data %>% mutate\_if(is.numeric, scale)  
Balanced\_DataKNN <- Balanced\_Data[,c(5,numericAtt)]  
Balanced\_DataKNN$Status <- as.numeric(Balanced\_DataKNN$Status) #converted successful and failed to 1 and 2  
Balanced\_DataKNN <- Balanced\_DataKNN[sample(1:nrow(Balanced\_Data)), ]  
  
for (i in seq(.1,1,.1)) {  
  
 index = createDataPartition(Balanced\_DataKNN$Status, p = i, list = F )  
 train = Balanced\_DataKNN[index,]  
 test = Balanced\_DataKNN[-index,]  
 Balanced\_Train\_labels <- Balanced\_DataKNN[1:nrow(train),1]  
 Balanced\_Test\_labels <- Balanced\_DataKNN[(nrow(train)+1):nrow(Balanced\_DataKNN),1]  
   
 KnnPredictions <- knn(train = train, test = test,cl = Balanced\_Train\_labels, k=10)  
  
}