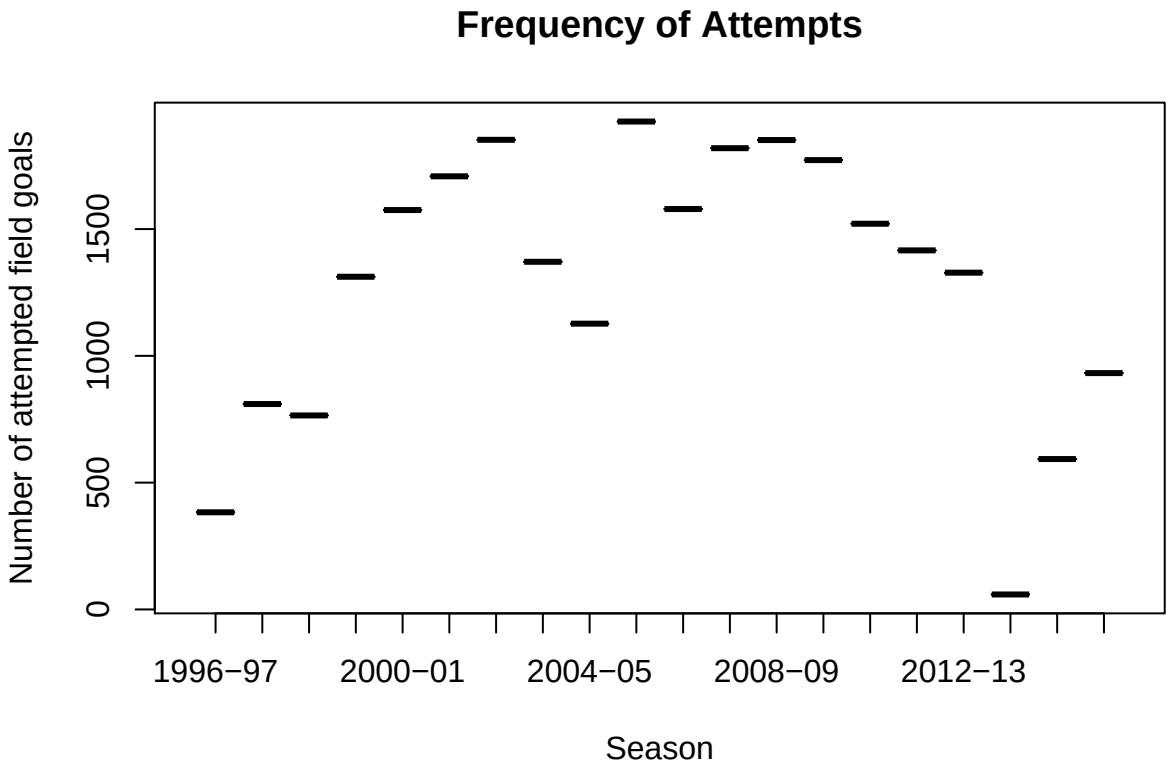
pstat 231

```
setwd("/Users/Shawn/Desktop/PSTAT231-Project")
set.seed(1)

# import data
library(data.table)
kobe = data.table(read.csv("kobe.csv"))
kobe = kobe[complete.cases(kobe),]

# randomly partition data into training set and test set
test.index = sample(seq_len(nrow(kobe)), size = floor(nrow(kobe)*0.1),replace = F)
train.index = setdiff(seq_len(nrow(kobe)),test.index)

shot.season=aggregate(shot_made_flag~season, kobe, length) # number of shots made per season
plot(shot.season,xlab="Season",ylab="Number of attempted field goals", main="Frequency of Attempts")
```




```
season.avg=aggregate(shot_made_flag~season,kobe,mean) # goal percentage per season
plot(season.avg,xlab="Season",ylab="Field goal percentage", main="Percentage of Shots Made")

# transform data to type date
kobe$game_date = as.Date(kobe$game_date, "%Y-%m-%d")

# latitude and longitude are obviously useless
# game_id, game_event_id,team_id,team_name also useless
kobe[,lon:=NULL]
kobe[,lat:=NULL]
```

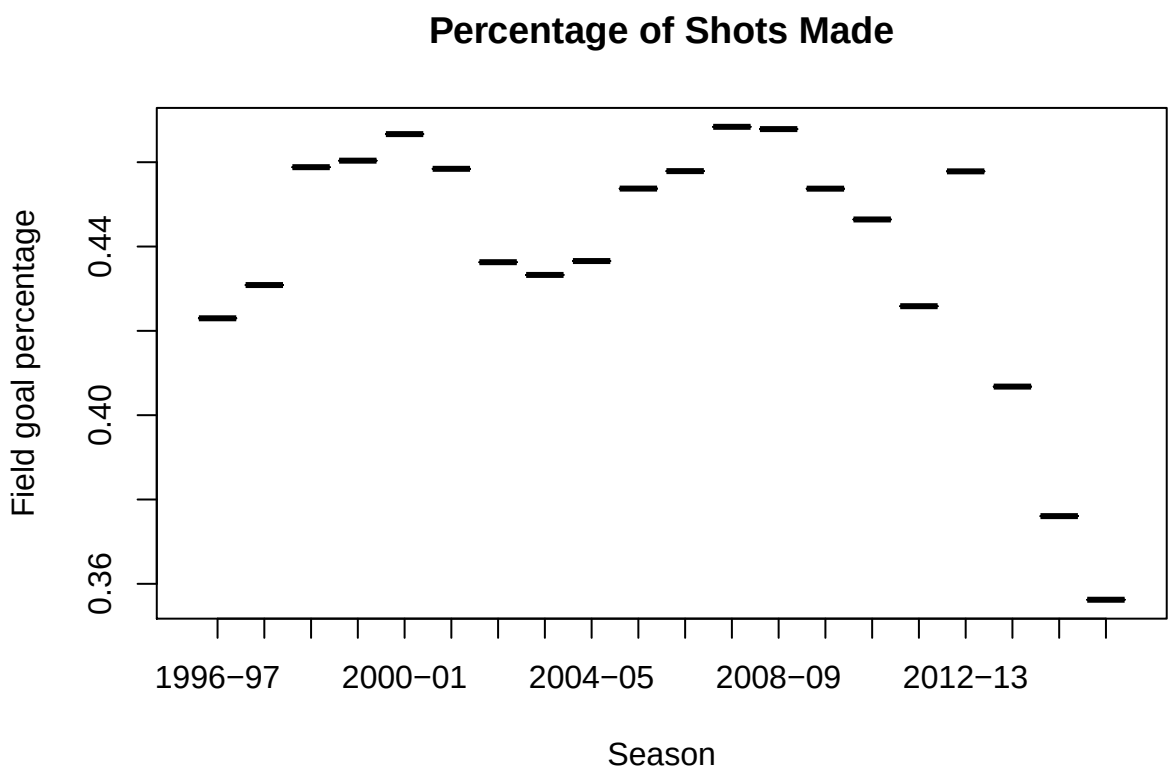
```
kobe[,game_id:=NULL]
kobe[,game_event_id:=NULL]
kobe[,team_id:=NULL]
kobe[,team_name:=NULL]

# combine minuetes remaining and seconds remaining
kobe[,seconds:=minutes_remaining*60+seconds_remaining,by=1:nrow(kobe)]
kobe[,minutes_remaining:=NULL]
kobe[,seconds_remaining:=NULL]

# replace matchup with binary variable "home"
kobe[,home:=1]
kobe[substr(matchup,5,5)=='@',home:=0]
kobe[,matchup:=NULL]

# some teams changed their names or locations
# we need combine old names and new names
kobe[opponent=="NOH",opponent:="NOP"]
kobe[opponent=="VAN",opponent:="MEM"]
kobe[opponent=="SEA",opponent:="OKC"]
kobe[opponent=="NJN",opponent:="BKN"]
library(rminer)
```

```
## Loading required package: kknn
```




```
kobe$opponent = delevels(kobe$opponent, c("NOH","VAN","SEA","NJN"), label = NULL)

# watch for the correlation between loc_x/loc_y with shot zone/shot distance
```

```
# library(corrplot)
# kobe.keep = kobe[,c("period","shot_distance","loc_x","loc_y","playoffs"),with=F]
# corr = corrplot(cor(kobe.keep))
# shot distance has a high correlation between loc_y
# kobe[,loc_y:=NULL]
# but it doesn't effect our model very much

# check if we need action_type
action.glm = glm(data = kobe, shot_made_flag~action_type)
# summary(action.glm) output too long -> hide
combine.glm = glm(data = kobe,shot_made_flag~combined_shot_type)
# summary(combine.glm) output too long -> hide

# not all action type are statistically significant
# we will replace action_type with combined_type
# if specific action_type is infrequent compared to others

freqTable = data.table(action_types = levels(kobe$action_type),frequency = as.vector(table(kobe$action_
freqTable =  freqTable[frequency>=50]
freqTable
```

```
##                       action_types frequency
##  1:            Alley Oop Dunk Shot        95
##  2:           Alley Oop Layup shot        67
##  3:              Driving Dunk Shot       257
##  4: Driving Finger Roll Layup Shot        59
##  5:       Driving Finger Roll Shot        68
##  6:             Driving Layup Shot      1628
##  7:     Driving Reverse Layup Shot        83
##  8:                       Dunk Shot       217
##  9:            Fadeaway Jump Shot       872
## 10:            Floating Jump shot        93
## 11:                      Hook Shot        73
## 12:                 Jump Bank Shot       289
## 13:                      Jump Shot     15836
## 14:                    Layup Shot      2154
## 15:              Pullup Jump shot       402
## 16:             Reverse Dunk Shot        61
## 17:            Reverse Layup Shot       333
## 18:             Running Jump Shot       779
## 19:            Running Layup Shot        51
## 20:                Slam Dunk Shot       334
## 21:           Step Back Jump shot       106
## 22:                       Tip Shot       151
## 23:          Turnaround Bank shot        58
## 24:      Turnaround Fadeaway shot       366
## 25:          Turnaround Jump Shot       891
##                       action_types frequency
```

```
kobe[,type:=combined_shot_type,by=1:nrow(kobe)]
kobe[action_type %in% freqTable$action_types,type:=action_type]

# now we have these levels in our new varible "type"
```

```r
#levels(kobe$type)

# delete action_type and combined_type
kobe[,action_type:=NULL]
kobe[,combined_shot_type:=NULL]
########################## data cleaning complete ################################

# use chi-square test to see if these "distance/location variable" are independent
tbl1 = table(kobe$shot_type,kobe$shot_zone_area)
chi1 = chisq.test(tbl1,correct = F)
chi1
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl1
## X-squared = 6337.9, df = 5, p-value < 2.2e-16
```

```r
tbl2 = table(kobe$shot_type,kobe$shot_zone_basic)
chi2 = chisq.test(tbl2,correct = F)
chi2
```

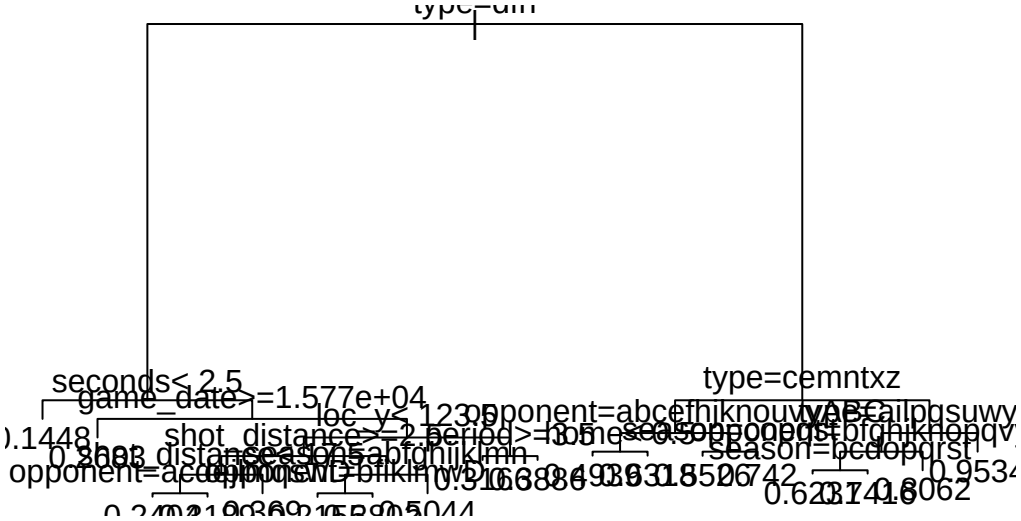
```
##
##  Pearson's Chi-squared test
##
## data:  tbl2
## X-squared = 25296, df = 6, p-value < 2.2e-16
```

```r
tbl3 = table(kobe$shot_type,kobe$shot_zone_area)
chi3 = chisq.test(tbl3,correct = F)
chi3
```

```
##
##  Pearson's Chi-squared test
##
## data:  tbl3
## X-squared = 6337.9, df = 5, p-value < 2.2e-16
```

```r
# as expected they are highly dependent
# we will eventually drop them in the following procedure

# decision tree
library(rpart)
orig.tree = rpart(data = kobe,formula = shot_made_flag~.-shot_id,na.action = NULL,control=rpart.control
plot(orig.tree)
text(orig.tree)
```



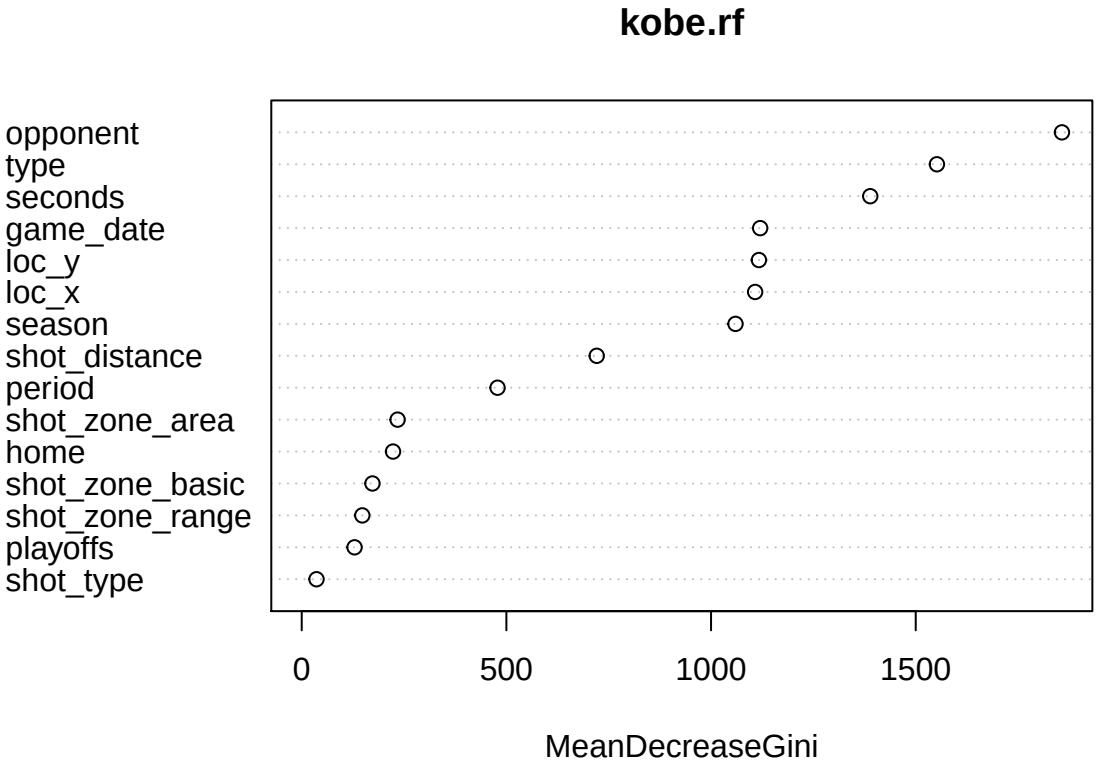

```
# interesting fact: Kobe shot poorly at the last 2.5 seconds

# random forrest
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
kobe.rf = randomForest(formula=as.factor(shot_made_flag)~.-shot_id,na.action=NULL,data = kobe,ntree=500)
imptplot = varImpPlot(kobe.rf)
```

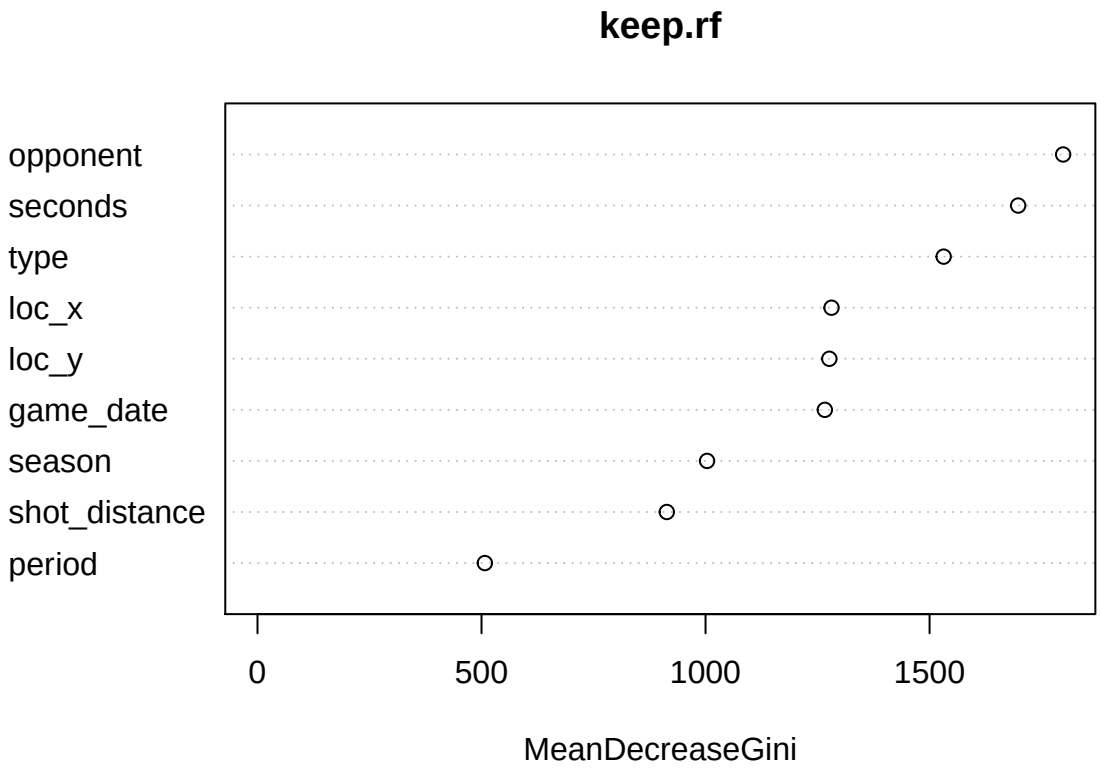


```
imptplot = as.data.table(imptplot,keep.rownames = T)
impt.sort = setorder(imptplot, cols = "MeanDecreaseGini")

# Important variables:
# opponent, type, seconds, loc_x, game_date, season, shot_distance, period
# 1,2,3,5,6,7,12,13,14,15,17  index of variable
col.keep = c(impt.sort$rn[7:15],"shot_made_flag","shot_id")
kobe.keep = subset(kobe,select = col.keep)

# actuall prediction
test = kobe.keep[test.index,]
train = kobe.keep[train.index,]

# redo random forrest on training data
keep.rf = randomForest(formula=as.factor(shot_made_flag)~.-shot_id,na.action=NULL,data = train,ntree=500
imptplot = varImpPlot(keep.rf)
```




```
#imptplot



# random forrest prediction on training data
rand.train = predict(keep.rf,train,type="class")
rand.train.table = table(rand.train,train$shot_made_flag)
rf.train.error = (rand.train.table[3] + rand.train.table[2])/nrow(train)
rf.train.error
```

```
## [1] 0.0003026634
```

```r
# random forrest prediction on test data
rand.test = predict(keep.rf,test,type="class")
rand.conti.table = table(rand.test,test$shot_made_flag)
rf.error.rate = (rand.conti.table[3] + rand.conti.table[2])/nrow(test)
rf.error.rate
```

```
## [1] 0.3479953
```

```r
glm.fit = glm(data = train, shot_made_flag~.-shot_id,family = binomial)
glm.fit
```

```
##
## Call:  glm(formula = shot_made_flag ~ . - shot_id, family = binomial,
##     data = train)
##
## Coefficients:
##                         (Intercept)                               period
##                           5.8284628                           -0.0527548
##                        shot_distance                       season1997-98
##                           -0.0015696                           -0.1191371
##                        season1998-99                       season1999-00
##                           0.4271879                            0.5414256
##                        season2000-01                       season2001-02
##                           0.7455897                            0.8483033
##                        season2002-03                       season2003-04
##                           0.9587302                            1.0708419
##                        season2004-05                       season2005-06
##                           1.2443476                            1.5719843
##                        season2006-07                       season2007-08
##                           1.7487687                            1.9209886
##                        season2008-09                       season2009-10
##                           2.0032560                            2.1113901
##                        season2010-11                       season2011-12
##                           2.2544538                            2.3882475
##                        season2012-13                       season2013-14
##                           2.5636406                            2.3533549
##                        season2014-15                       season2015-16
##                           2.6269049                            2.6135204
##                               loc_x                                loc_y
##                           0.0001919                            0.0004710
##                           game_date                              seconds
##                          -0.0004533                            0.0003214
##                            typeDunk                       typeHook Shot
##                           1.2635352                           -1.1315420
##                       typeJump Shot                            typeLayup
##                          -2.0704859                           -0.7557226
##                        typeTip Shot                  typeDriving Dunk Shot
##                          -1.8845773                            2.4962455
##                      typeLayup Shot                typeRunning Jump Shot
##                          -1.8132605                           -0.2829470
##                typeReverse Dunk Shot                   typeSlam Dunk Shot
##                           0.9827202                            2.6080218
##               typeDriving Layup Shot              typeTurnaround Jump Shot
```

```
##                          -0.2438981                          -0.9501220
##              typeReverse Layup Shot              typeAlley Oop Dunk Shot
##                          -0.8093886                           1.5788317
##                       typeDunk Shot            typeAlley Oop Layup shot
##                          -0.1486913                          -0.6407424
##       typeDriving Finger Roll Shot              typeRunning Layup Shot
##                           0.4752757                          -0.4514729
##              typeFadeaway Jump Shot                  typeJump Bank Shot
##                          -1.0053035                          -0.0771615
## typeDriving Finger Roll Layup Shot                typePullup Jump shot
##                           0.4737361                          -0.1734717
##       typeTurnaround Fadeaway shot      typeDriving Reverse Layup Shot
##                          -0.8135983                          -0.0557145
##            typeStep Back Jump shot            typeTurnaround Bank shot
##                          -0.5799245                          -0.0064698
##             typeFloating Jump shot                         opponentBKN
##                          -0.3082572                          -0.2482801
##                         opponentBOS                         opponentCHA
##                          -0.0489221                           0.1193880
##                         opponentCHI                         opponentCLE
##                           0.0175555                          -0.0299388
##                         opponentDAL                         opponentDEN
##                           0.0858121                           0.0880921
##                         opponentDET                         opponentGSW
##                           0.0566639                          -0.1029760
##                         opponentHOU                         opponentIND
##                          -0.0868537                          -0.0368277
##                         opponentLAC                         opponentMEM
##                           0.0498924                           0.0169809
##                         opponentMIA                         opponentMIL
##                           0.0649980                           0.0800539
##                         opponentMIN                         opponentNOP
##                           0.0740057                           0.1021115
##                         opponentNYK                         opponentOKC
##                           0.2404772                          -0.0069060
##                         opponentORL                         opponentPHI
##                           0.0115125                           0.0102791
##                         opponentPHX                         opponentPOR
##                           0.1550477                           0.0264415
##                         opponentSAC                         opponentSAS
##                           0.1563376                           0.0488850
##                         opponentTOR                         opponentUTA
##                          -0.0607065                           0.0761159
##                         opponentWAS
##                           0.0250403
##
## Degrees of Freedom: 23127 Total (i.e. Null);  23047 Residual
## Null Deviance:       31800
## Residual Deviance: 28230     AIC: 28400
```

```
# logistic regresssion prediciton on training data
glm.probs.train = predict(glm.fit,train,type="response")
glm.pred.train = rep("Shot fail",nrow(train))
glm.pred.train[glm.probs.train>0.5]="Shot Made"
```

```
glm.conti.table.train = table(glm.pred.train,train$shot_made_flag)
glm.error.rate.train = (glm.conti.table.train[3] + glm.conti.table.train[2])/nrow(train)
glm.error.rate.train
```

```
## [1] 0.3196126
```

```
# logistic regression prediction on testing data
glm.probs = predict(glm.fit,test,type="response")
glm.pred=rep("Shot fail",nrow(test))
glm.pred[glm.probs>0.5]="Shot Made"
glm.conti.table = table(glm.pred,test$shot_made_flag)
glm.error.rate = (glm.conti.table[3] + glm.conti.table[2])/nrow(test)
glm.error.rate
```

```
## [1] 0.3164656
```