dodgers_predictions

```r
#install.packages("MASS")
library(MASS)
#install.packages("tree")
library(tree)
#install.packages("dplyr")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

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

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

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

```r
#install.packages("pls")
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
training.data=read.csv("train.csv")#load training data
testing.data=read.csv("test.csv")#load testing data for submissions(can not be used to test model, only



#summary(training.data) #what does that data look like?
#appears to have 8 variables that are not numeric:
# id (numeric but not useful since it is a unique ID number)
# gameID (numeric but not useful since it is a unique ID number)
# HTWins
# VT
# HT
# VTleague
# HWleague
# date (numeric but not useful since it is a number not actual date)

dim(training.data)
```

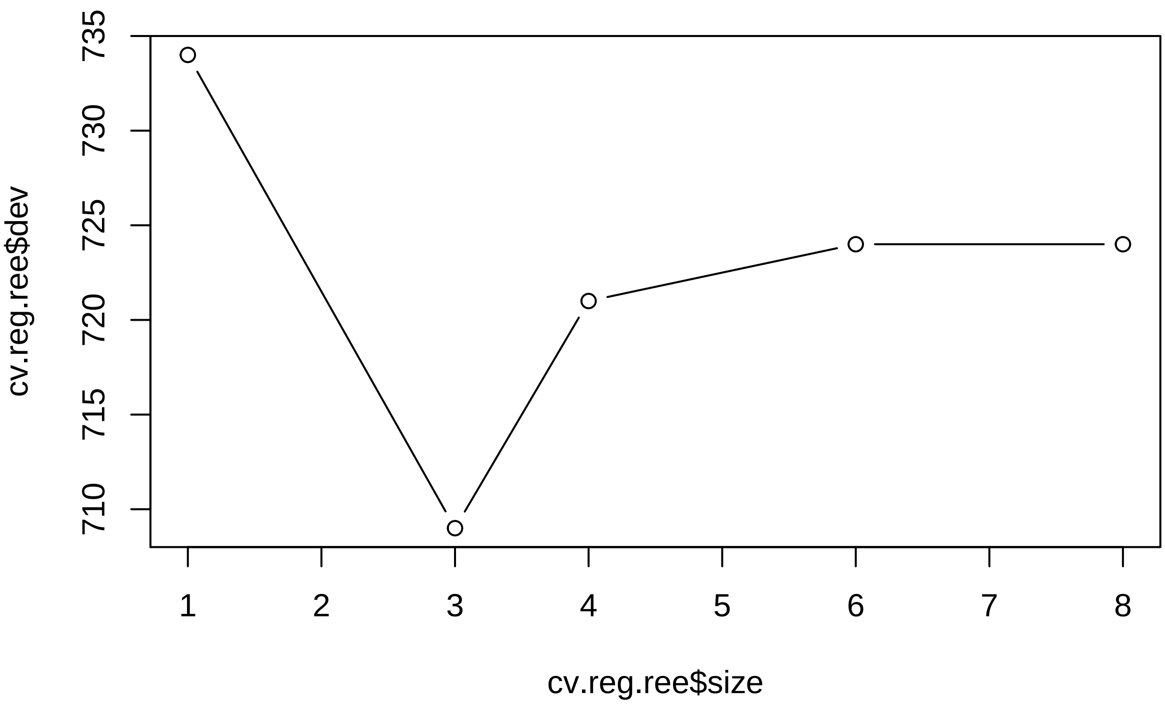
```
## [1] 9520  218
```

```r
set.seed(109837)
train=sample(1:nrow(training.data),0.8* nrow(training.data))#create our training and validation sets
```

```r
train.df=training.data[-train,]#train model on 80% of data
test.df=training.data[train,]

###FIRST MODEL: CART (classification and regression tree)

reg.tree=tree(HTWins~., data=train.df)
summary(reg.tree) # we see that the regression tree chose 3 variables to use: "VT.OS3.plmin" "VT.S1.plm
```

```
##
## Classification tree:
## tree(formula = HTWins ~ ., data = train.df)
## Variables actually used in tree construction:
## [1] "VT.S1.plmin"  "VT.OS3.plmin" "VT.OTA.ast"   "VT.S3.plmin"  "VT"
## Number of terminal nodes:  8
## Residual mean deviance:  1.156 = 2191 / 1896
## Misclassification error rate: 0.3157 = 601 / 1904
```

```r
set.seed(1)
cv.reg.ree=cv.tree(reg.tree,FUN=prune.misclass) #prune the tree
plot(cv.reg.ree$size,cv.reg.ree$dev,type="b") #we see that 3 is best
```




```r
prune.reg.tree=prune.tree(reg.tree,best=3) #prune tree to 3
summary(prune.reg.tree) #we see the number of variables from pruning drops from 3 to 2, only "VT.OS3.pl
```

```
##
## Classification tree:
## snip.tree(tree = reg.tree, nodes = c(4L, 5L, 3L))
## Variables actually used in tree construction:
## [1] "VT.S1.plmin"  "VT.OS3.plmin"
## Number of terminal nodes:  3
## Residual mean deviance:  1.252 = 2379 / 1901
## Misclassification error rate: 0.3808 = 725 / 1904
```

```r
#After pruning the tree, we do not see any change in the misclassification error
predictions_regtree=predict(reg.tree,test.df )
predictions_regtree[predictions_regtree > 0.5]="Yes"
predictions_regtree[predictions_regtree < 0.5]="No"
mean(predictions_regtree!=test.df$HTWins)#error rate is the same
```

```
## [1] 0.5
```

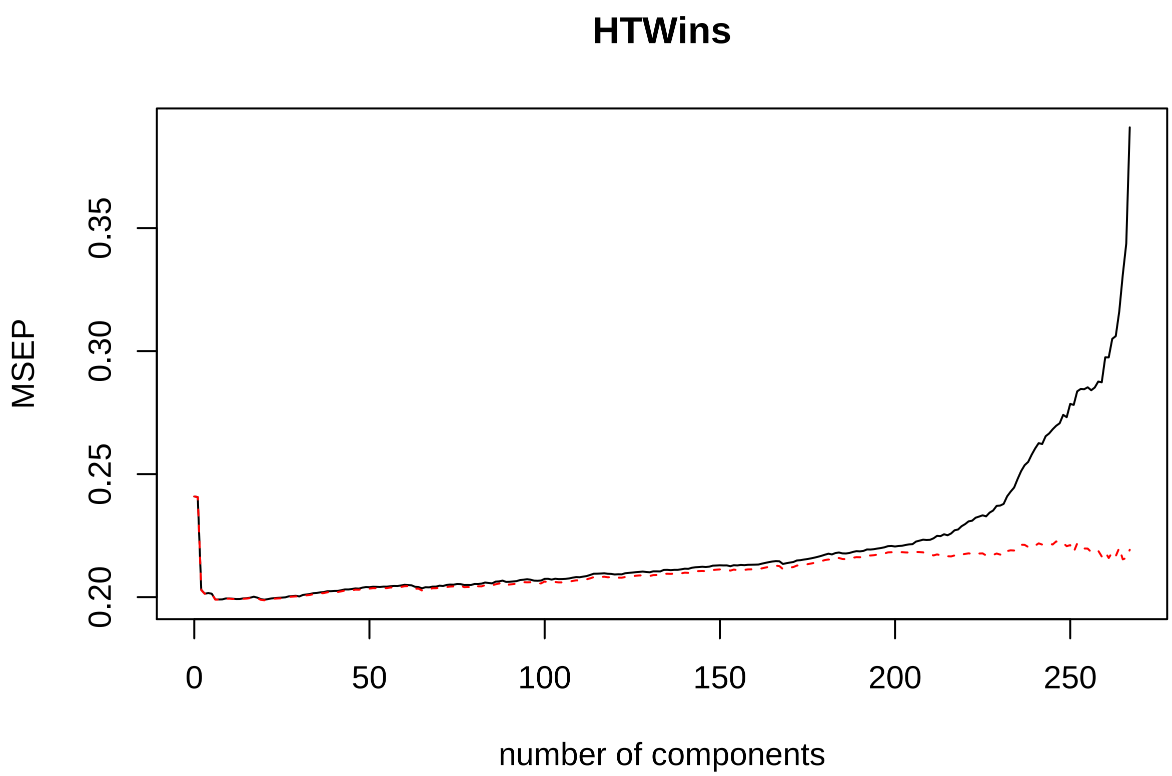
```r
predictions_pruneregtree=predict(prune.reg.tree,test.df )
predictions_pruneregtree[predictions_pruneregtree > 0.5]="Yes"
predictions_pruneregtree[predictions_pruneregtree < 0.5]="No"
mean(predictions_pruneregtree!=test.df$HTWins)#error rate is the same
```

```
## [1] 0.5
```

```r
###Let's try PCR instead

#set the training set to binary for HTWins

pcr.train=train.df
pcr.train$HTWins=as.character(pcr.train$HTWins)
pcr.train$HTWins[pcr.train$HTWins == "Yes"]<-1
pcr.train$HTWins[pcr.train$HTWins == "No"]<-0
pcr.train$HTWins=as.numeric(pcr.train$HTWins)



pcr.train=na.omit(pcr.train)#make sure no NA values in the data

train.pcr=pcr(HTWins~., data=pcr.train, validation="CV",scale=TRUE)
MSE_pcr=(MSEP(train.pcr))
#MSE_pcr$val[1,1,]
which.min(MSE_pcr$val[1,1,])# we see that with 7 comps, the MSE is the lowest
```

```
## 20 comps
##       21
```

```r
#summary(train.pcr)
validationplot(train.pcr, val.type = "MSEP")
```



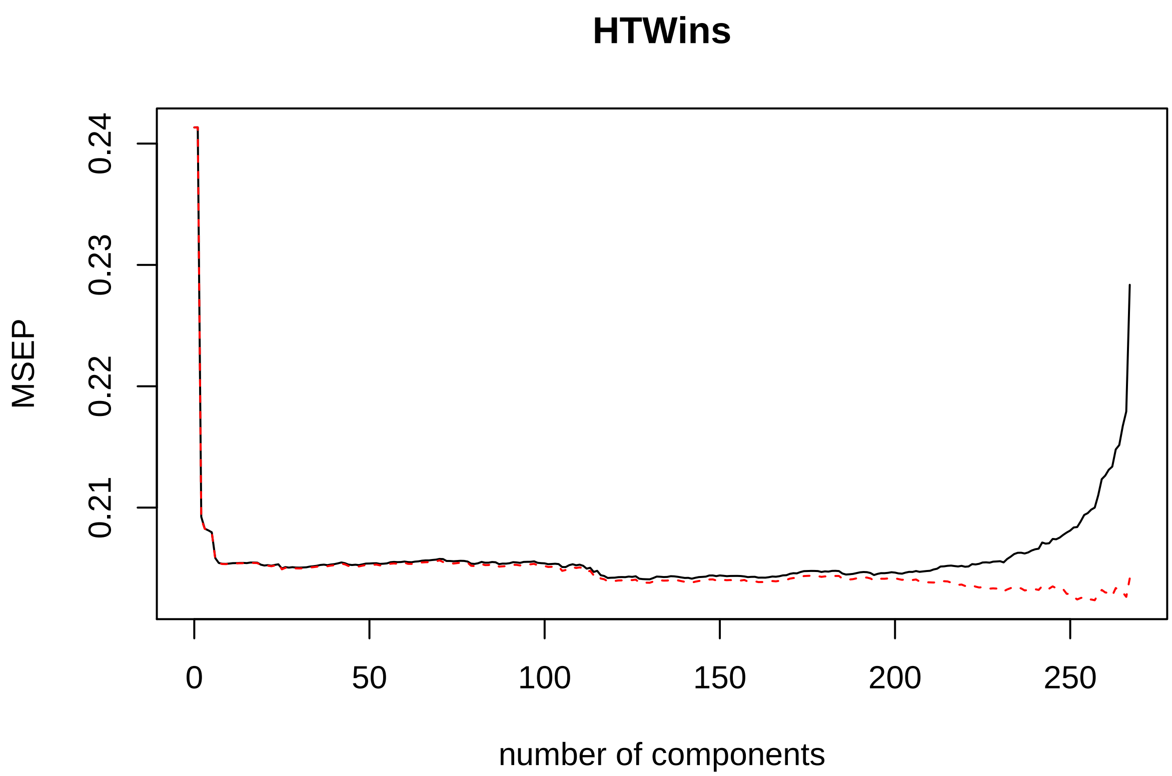

```
#summary(train.pcr)
pred.pcr=predict(train.pcr, test.df, ncomp=7)
pred.pcr[(pred.pcr > 0.5)]<-"Yes"
pred.pcr[(pred.pcr < 0.5)]<-"No"
mean(pred.pcr!=test.df$HTWins) #we see with 7 components, we have a MSE of 0.324 which is much lower th
```

```
## [1] 0.3266807
```

```
#let's see what the MSE is for the entire training data set
pcr.train=training.data
pcr.train$HTWins=as.character(pcr.train$HTWins)
pcr.train$HTWins[pcr.train$HTWins == "Yes"]<-1
pcr.train$HTWins[pcr.train$HTWins == "No"]<-0
pcr.train$HTWins=as.numeric(pcr.train$HTWins)

train.pcr=pcr(HTWins~., data=pcr.train, validation="CV",scale=TRUE)
MSE_pcr=(MSEP(train.pcr))
#MSE_pcr$val[1,1,]
which.min(MSE_pcr$val[1,1,])
```

```
## 130 comps
##       131
```

```
# we see that with 118 comps, the MSE is the lowest, however that is a significantly larger amount of c
#summary(train.pcr)
#we see that with 7 comps, the RMSEP is 0.4533 whereas for 118 comps, it is 0.4523. This is a differenc
validationplot(train.pcr, val.type = "MSEP")
```




```
pred.pcr=predict(train.pcr, training.data, ncomp=118)
pred.pcr[(pred.pcr > 0.5)]<-"Yes"
pred.pcr[(pred.pcr < 0.5)]<-"No"
mean(pred.pcr!=training.data$HTWins)#the misclassification rate is 0.3093 for PCR with 118 components.
```

```
## [1] 0.3093487
```

```
#Our previous PCR with only 7 components had a misclassification rate of 0.324(on the testing data fram
#Let's see how our 7component PCR performs on the entire training data set.
pred.pcr=predict(train.pcr, training.data, ncomp=7)
pred.pcr[(pred.pcr > 0.5)]<-"Yes"
pred.pcr[(pred.pcr < 0.5)]<-"No"
mean(pred.pcr!=training.data$HTWins)
```

```
## [1] 0.3211134
```

```
#the misclassification error is 0.32111 ; this is not too different from out 118 component model. To av
#let's check the model with the entire training data set now, not just the 80% training data set
MSE_pcr=(MSEP(train.pcr))
#MSE_pcr$val[1,1,]
which.min(MSE_pcr$val[1,1,])#says that 118 components has the lowest MSE, however let's look at 7 compo
```

```
## 130 comps
##       131
```

```
#we see that at 7 components, the MSE_pcr$val is 0.2054814 and 118 components is 0.2048. This is a very
pred.pcr=predict(train.pcr, training.data, ncomp=9)
pred.pcr[(pred.pcr > 0.5)]<-"Yes"
pred.pcr[(pred.pcr < 0.5)]<-"No"
mean(pred.pcr!=training.data$HTWins)
```

```
## [1] 0.3203782
```

```
#we see that 9 components has a lower misclassification error than 7 components. 9 components may actua


#from the Kaggle compentition, we know that the 9 component model performed the best out of the 3 model

#let's finalize our submission and predict based on our testing.data

pred.pcr=predict(train.pcr, testing.data, ncomp=118)
pred.pcr[(pred.pcr > 0.5)]<-"Yes"
pred.pcr[(pred.pcr < 0.5)]<-"No"

testing.submission.pcr= data.frame("id"=testing.data$id, "HTWins"=pred.pcr)
testing.submission.pcr2=data.frame("id"=testing.submission.pcr$id, "HTWins"=testing.submission.pcr$HTWi
write.csv(testing.submission.pcr2,"test_predictions.pcr_fulldata_118comps.csv",row.names = FALSE)

#FROM KAGGLE RESULTS:

#From the Kaggle results, we see that the performance of our model with 9 comps
#correctly classified the Win or Loss 68.446% of the time. For 7 components,
#it correctly classfied 67.597% of the time. For 118 components, it classified
#correctly 67.111% of the time. We see that 9 component model is the best.



#FUTURE:
#From the training data set, we see there is a variable date. In the future
#I would love to explore how and if time affects the scores. From the Kaggle testing data
#we see that the data set used to test our models on Kaggle are actually observations
#in the future. It would be interesting to conduct some time series analysis on
#some of the variables.
```