In Class Assignment 6

Lydia Strebe

March 2, 2019

Predicting NBA Wins

Below we use 4 different methods to create models to predict the number of wins in a basketball season: forward selection, backwards selection, ridge, and LASSO. For each method, cross validation is used to choose the number of regressors. All variables are considered except Playoffs, SeasonEnd and Team. Each model is then tested to see how well it performs out of sample.

Forward Stepwise Selection

```
## Warning: package 'leaps' was built under R version 3.5.2
```

```
## Warning: package 'glmnet' was built under R version 3.5.2
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.5.2
```

```
## Loaded glmnet 2.0-16
```

```
NBA_train_full = read.csv("C:/Users/lydia/Documents/NBA_train.csv", header = TRUE)

#Disregard Playoffs, SeasonEnd and Team
NBA_train = NBA_train_full[,-c(1,2,3)]

NBA_test_full = read.csv("C:/Users/lydia/Documents/NBA_test.csv", header = TRUE)
NBA_test = NBA_test_full[,-c(1,2,3)]

#Create a predict function to work with regsubsets
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

#10-fold cross validation
k=10
ffolds=sample(1:k,nrow(NBA_train),replace=TRUE)
#Create a matrix to hold the cross validation error
f.cv.errors=matrix(0,k,13)
for(j in 1:k){
  fwd_fit=regsubsets(W~.,data=NBA_train[ffolds!=j,],nvmax=16,method='forward')
  for(i in 1:13){
    pred=predict(fwd_fit,NBA_train[ffolds==j,],id=i)
    f.cv.errors[j,i]=mean( (NBA_train$W[ffolds==j]-pred)^2) #calculate MSE and put into matrix
  }
}
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 3 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

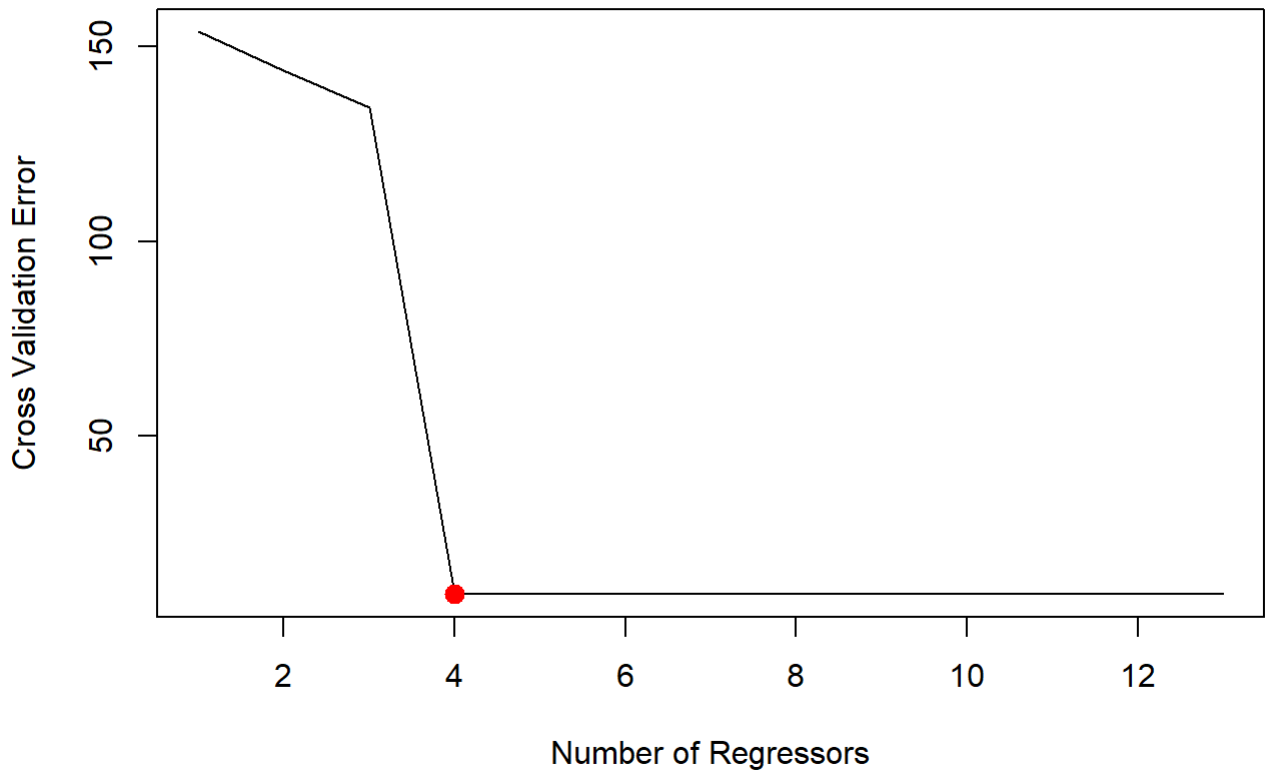
```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length
```

```
#Take the avg cross validation error for each number of regressors
f.mean.cv.errors=colMeans(f.cv.errors)
#Plot the cross validation error vs. the number of regressors
plot(f.mean.cv.errors,type='l',xlab='Number of Regressors',ylab = 'Cross Validation Error')
#Pinpoint the number of regressors that results in the lowest cross validation error
f.mincv = which.min(f.mean.cv.errors)
points(f.mincv,f.mean.cv.errors[f.mincv], col="red",cex=2,pch=20)
```




```
f.reg.best=regsubsets(W~.,data=NBA_train, nvmax=16,method='forward')
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 3 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length
```

```
coef(f.reg.best,f.mincv)
```

```
##  (Intercept)          PTS       oppPTS          BLK          TOV
## 40.064337824  0.032329355 -0.032271271  0.004443628 -0.001086839
```

```
#Test the model out of sample
fwd_ypred = predict(f.reg.best,NBA_test,f.mincv)
sqrt(mean((fwd_ypred-NBA_test$W)^2))
```

```
## [1] 3.080311
```

Backward Stepwise Selection

```
#10-fold cross validation
bfolds=sample(1:k,nrow(NBA_train),replace=TRUE)
#Create a matrix to hold the cross validation error
b.cv.errors=matrix(0,k,13)
for(j in 1:k){
  fwd_fit=regsubsets(W~.,data=NBA_train[bfolds!=j,],nvmax=16,method='backward')
  for(i in 1:13){
    pred=predict(fwd_fit,NBA_train[bfolds==j,],id=i)
    b.cv.errors[j,i]=mean( (NBA_train$W[bfolds==j]-pred)^2) #calculate MSE and put into matrix
  }
}
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 3 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

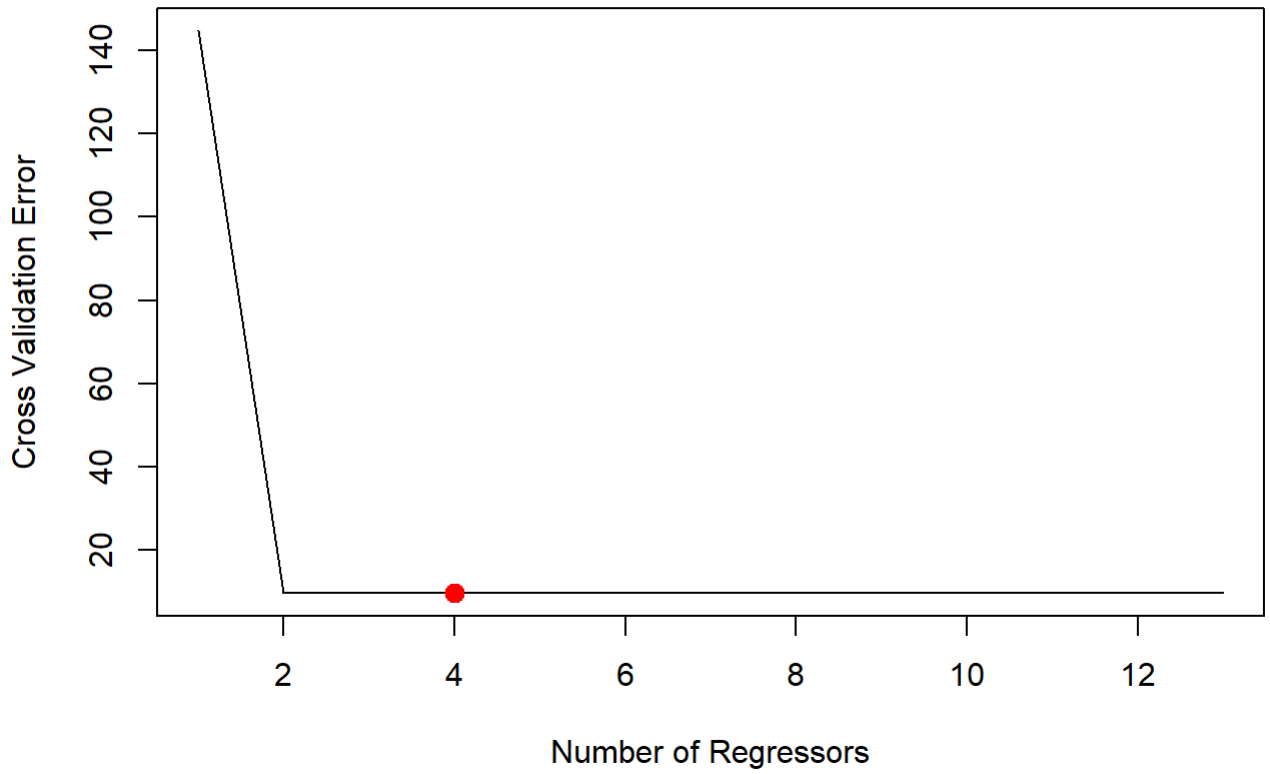
```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length

## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: 3 linear dependencies
## found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length
```

```
#Take the avg cross validation error for each number of regressors
b.mean.cv.errors=colMeans(b.cv.errors)
#Plot the cross validation error vs. the number of regressors
plot(b.mean.cv.errors,type='l',xlab='Number of Regressors',ylab = 'Cross Validation Error')
#Pinpoint the number of regressors that results in the lowest cross validation error
b.mincv = which.min(b.mean.cv.errors)
points(b.mincv,b.mean.cv.errors[b.mincv], col="red",cex=2,pch=20)
```




```
b.reg.best=regsubsets(W~.,data=NBA_train, nvmax=16,method='forward')
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 3 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to
## replace is not a multiple of replacement length
```

```
coef(b.reg.best,b.mincv)
```

```
##  (Intercept)          PTS       oppPTS          BLK          TOV
## 40.064337824  0.032329355 -0.032271271  0.004443628 -0.001086839
```

```
#Test the model out of sample
back_ypred = predict(b.reg.best,NBA_test,b.mincv)
sqrt(mean((back_ypred-NBA_test$W)^2))
```

```
## [1] 3.080311
```

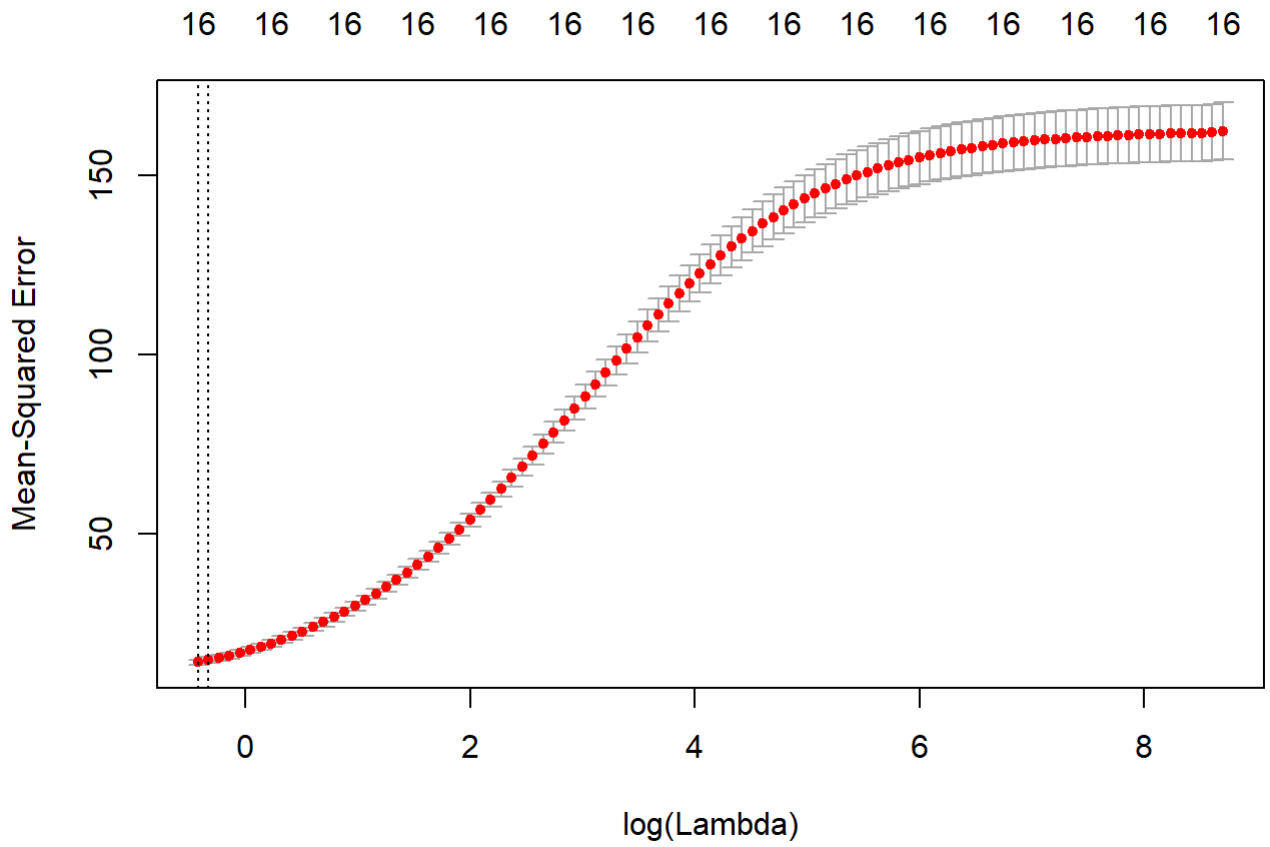
Ridge Regression

```
#Create matrix of the data. Throw away the first column.
x=model.matrix(W~.,NBA_train)[,-1]
y=NBA_train$W

#Create lambdas
grid=10^seq(10,-2,length=100)

ridge.mod=glmnet(x,y,alpha=0,lambda=grid)

#Cross validation
cv.out=cv.glmnet(x,y,alpha=0)
#Plot MSE as lambda increases
plot(cv.out)
```




```
#Use the best lambda up to one standard deviation
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 0.6580633
```

```
#Create linear model
whole_data_ridge=glmnet(x,y,alpha=0,lambda=bestlam)
coef(whole_data_ridge)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) 16.9655063398
## PTS          0.0116414585
## oppPTS      -0.0208323712
## FG           0.0189012731
## FGA         -0.0065656656
## X2P          0.0059128689
## X2PA        -0.0017664951
## X3P          0.0098403344
## X3PA         0.0008665429
## FT           0.0095217142
## FTA          0.0021817565
## ORB          0.0068993019
## DRB          0.0160775839
## AST          0.0076037515
## STL          0.0156609027
## BLK          0.0054206620
## TOV         -0.0145626299
```

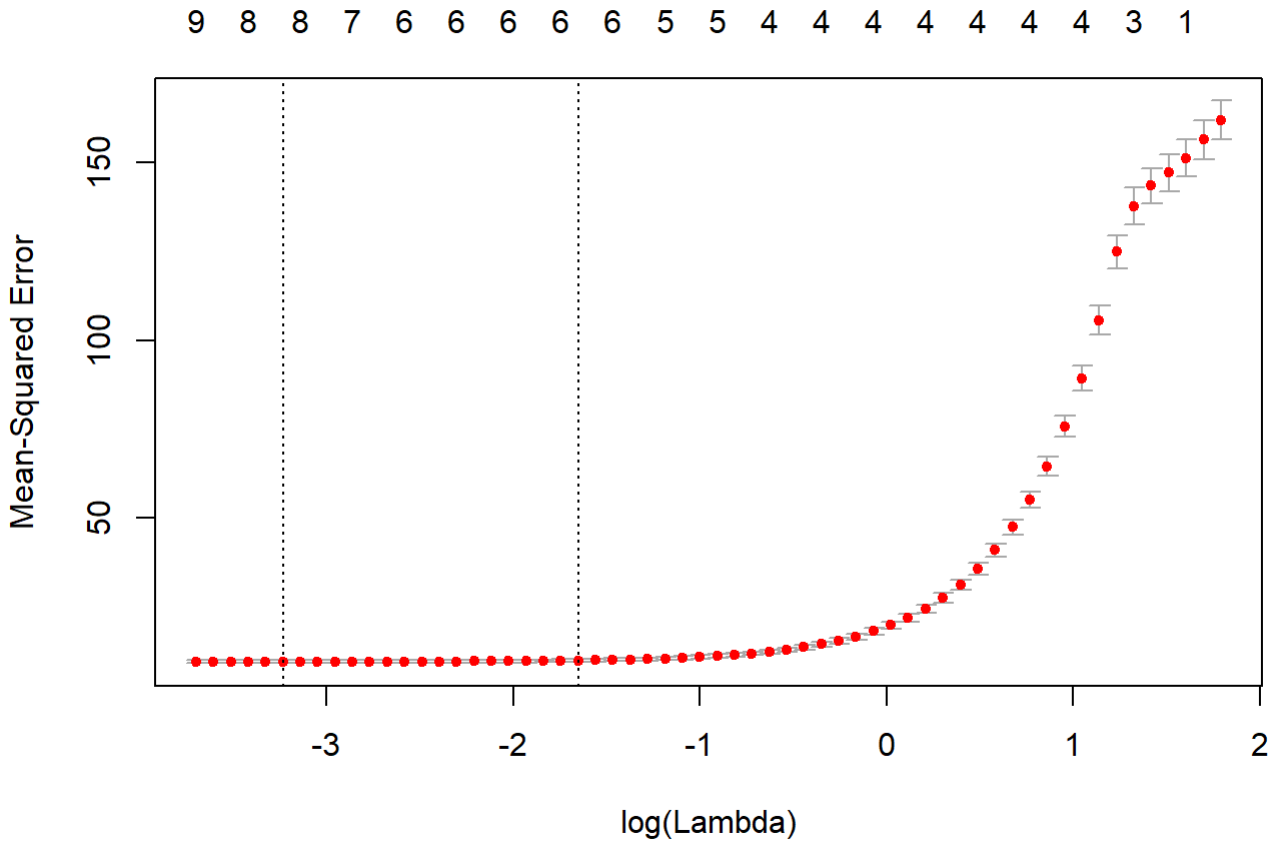
```
#Ridge testing
xpred=model.matrix(W~.,NBA_test)[,-1]
y_test=NBA_test$W
ridge_ypred = predict(whole_data_ridge,xpred)
sqrt(mean((ridge_ypred-y_test)^2))
```

```
## [1] 4.060678
```

LASSO

```
lasso.mod=glmnet(x,y,alpha=1,lambda=grid)

#Cross validation
cv.out=cv.glmnet(x,y,alpha=1)
#Plot MSE as lambda increases
plot(cv.out)
```




```
#Use the best lambda up to one standard deviation
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 0.03944986
```

```
#Create linear model
whole_data_lasso = glmnet(x,y,alpha=1,lambda=bestlam)
coef(whole_data_lasso)
```

```
## 17 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  3.629776e+01
## PTS          3.141050e-02
## oppPTS      -3.150254e-02
## FG           .
## FGA         -2.306877e-04
## X2P          .
## X2PA        -2.133882e-05
## X3P          .
## X3PA         .
## FT           .
## FTA          .
## ORB          .
## DRB          1.929619e-03
## AST          1.259915e-03
## STL          .
## BLK          3.513199e-03
## TOV         -1.061161e-03
```

```
#Testing
lasso_ypred = predict(whole_data_lasso,xpred)
sqrt(mean((lasso_ypred-y_test)^2))
```

```
## [1] 3.131909
```

Forward and backward stepwise selection seem to perform the best on the test set.