NBA ML Project: Final Report

Chris Canter, Austin Clinger, Jaival Desai, Bill Hansen, Andrew Pruitt

4/27/2018

Today, the NBA is one of the most popular sports leagues in the world. Millions of people invest their time and energy into following the players and the 30 teams. Many of these people also invest their money, whether it be on the business/organizational side where it goes towards the products, or on the betting side where it goes towards the outcomes. Thus if money is on the line, people are interested. Since the data is converted to per game stats, one can extend the models and utilize them during the active regular season. For example, an organization can run the regression models 20/30/40/50 games into a regular season to predict the number of wins their team will register, or run the classification models to predict whether their team will make or miss the playoffs. Las Vegas Sportsbooks use numerous models to set their betting lines on the over/under for the number of wins a team will register in the regular season, as well as the odds for a team to make the playoffs. Whether you are a part of management for an organization (team), a employee for a sportsbook, a bettor on the outside, or simply a fan of a team, these models can provide insight on what decisions to make in each respective situation.

From the datasets and models, we want to predict a) the number of wins a team would register in the regular season and b) whether a team would make or miss the playoffs in the postseason. The training dataset is a n x p = 300 x 26 data matrix, and the test dataset is a n x p = 90 x 26 data matrix. For a), we decided to look into Ridge Regression and LASSO, and for b), we decided to look into Logistic Regression, SVM, and Tree-based Methods. We also explored the unsupervised learning realm with hierarchical clustering and PCA.

For the regression methods, we first read in the training and test datasets that include regular season wins as a column. For the classification methods, we first read in the training and test datasets and add a column to denote if a team made or missed the playoffs. Then, we converted the datasets into per game stats through dividing certain predictors by 82. From here, we are ready to begin our statistical learning.

We performed both Ridge Regression and LASSO on the datasets to predict the number of wins a team would register in a given regular season. We chose to control model complexity with these regularization methods since they are hot topics in machine learning today. Since Ridge includes all p statistical features, it is appropriate to observe how each coefficient estimate shrinks from the tuning parameter lambda. LASSO does not include all p statistical features, so it is appropriate to observe how it performs feature selection along with how each coefficient estimate shrinks.

```r
# reading in the datasets for Ridge and LASSO
library(stringr)

# training dataset
traindata <- read.csv("/Users/whansen/Desktop/Junior Year/Second Semester/STOR 565/Project/data/nbatrain
traindata$Team <- as.character(traindata$Team)

# testing dataset
testdata <- read.csv("/Users/whansen/Desktop/Junior Year/Second Semester/STOR 565/Project/data/nbatestda
testdata$Team <- as.character(testdata$Team)

# converting datasets into per game stats
for (i in c(6,7,9,10,12,13,15,16,18,19,20,21,22,23,24,25,26))
{
  traindata[ ,i] <- traindata[ ,i]/82
  testdata[ ,i] <- testdata[ ,i]/82
}
```

```r
# performing Ridge regression on the training dataset, then computing the test MSE
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

```r
set.seed(1)

ytrain <- traindata$WINS
xtrain <- model.matrix(WINS ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + PTS, data = traindata)[ ,-1]

ytest <- testdata$WINS
xtest <- model.matrix(WINS ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + PTS, data = testdata)[ ,-1]

NBA.ridge <- glmnet(xtrain, ytrain, alpha = 0)
cv.ridge <- cv.glmnet(xtrain, ytrain, alpha = 0)
bestlam <- cv.ridge$lambda.min

ridge.pred <- predict(NBA.ridge, s = bestlam, newx = xtest)
mean((ridge.pred - ytest)^2)
```

```
## [1] 54.23248
```

```r
# performing LASSO regression on the training dataset, then computing the test MSE
set.seed(1)

NBA.lasso <- glmnet(xtrain, ytrain, alpha = 1)
cv.lasso <- cv.glmnet(xtrain, ytrain, alpha = 1)
bestlam2 <- cv.lasso$lambda.min

lasso.pred <- predict(NBA.lasso, s = bestlam2, newx = xtest)
mean((lasso.pred - ytest)^2)
```

```
## [1] 47.25993
```

We performed Ridge Regression on the training dataset using all 21 variables (from "FG" to "PTS"), and the response variable was "WINS". The test MSE for Ridge was 54.23. We then performed LASSO on the training dataset using only 12 of the 21 variables (FGA, FG%, 3P%, 2PA, FT%, ORB, TRB, AST, STL, BLK, TOV, PF), and the response variable was WINS. The test MSE for LASSO was 47.26. The feature selection of LASSO eliminated insignificant covariates, and this resulted in a lower test MSE compared to Ridge. Therefore, if we were to recommend a particular model to predict the number of wins for a team in a given regular season, we would recommend LASSO over Ridge.

Next, we performed Logistic Regression on the datasets to predict whether a team would make or miss the playoffs. It is appropriate since it is a natural extension of linear regression to classification, and this scenario fits the two-class problem. We assume a parametric model because the data are from a Gaussian distribution (a few really bad teams, a lot of average teams, and a few really good teams). We also felt it was appropriate to perform LASSO when training the model so as not to overfit the data.

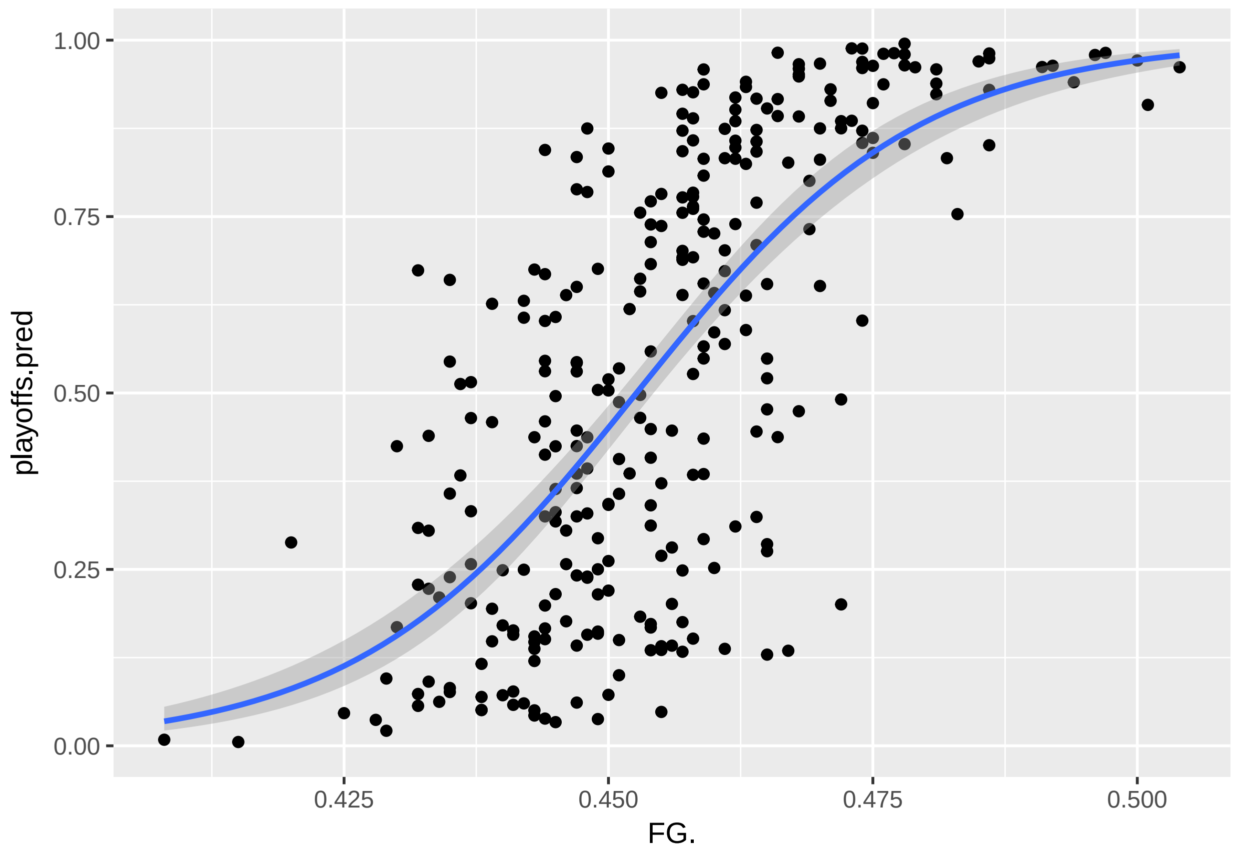
```r
# reading in the datasets and adding a column that denotes if a team made or missed the playoffs
library(stringr)

# training dataset
```

```
traindata <- read.csv("/Users/whansen/Desktop/Junior Year/Second Semester/STOR 565/Project/data/nbatrai
traindata$Team <- as.character(traindata$Team)
traindata$playoffs <- NA
for (i in 1:nrow(traindata))
{
  if(endsWith(traindata$Team[i], "*"))
  {
    traindata$playoffs[i] <- 1
  }
  else
  {
    traindata$playoffs[i] <- 0
  }
}

# testing dataset
testdata <- read.csv("/Users/whansen/Desktop/Junior Year/Second Semester/STOR 565/Project/data/nbatestda
testdata$Team <- as.character(testdata$Team)
testdata$playoffs <- NA
for (i in 1:nrow(testdata))
{
  if(endsWith(testdata$Team[i], "*"))
  {
    testdata$playoffs[i] <- 1
  }
  else
  {
    testdata$playoffs[i] <- 0
  }
}

# converting datasets into per game stats
for (i in c(5,6,8,9,11,12,14,15,17,18,19,20,21,22,23,24,25))
{
  traindata[ ,i] <- traindata[ ,i]/82
  testdata[ ,i] <- testdata[ ,i]/82
}
```

```
# performing LASSO (logistic regression) on the training dataset
library(glmnet)
set.seed(1)

y <- traindata$playoffs
x <- model.matrix(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + PTS, data = traindata)[ ,-1]

fit.lasso <- glmnet(x, y, alpha = 1)
cv.lasso <- cv.glmnet(x, y, alpha = 1)
bestlam <- cv.lasso$lambda.min
predict(fit.lasso, s = bestlam, type = "coefficients")
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -1.284651598
## FG            .
## FGA         -0.084680782
## FG.          6.436998682
## X3P          .
## X3PA         .
## X3P.         1.350389826
## X2P          .
## X2PA        -0.020242511
## X2P.         .
## FT           0.001722950
## FTA          .
## FT.          1.120375635
## ORB          .
## DRB          0.004273881
## TRB          0.134898836
## AST          0.046207623
## STL          0.184759992
## BLK          .
## TOV         -0.164350961
## PF          -0.004163559
## PTS          .
```

```r
# performing logistic regression on the training dataset
library(ggplot2)
glm.playoffs <- glm(playoffs ~ FG. + X3P. + FT. + TRB + STL + TOV, data = traindata, family = binomial)
playoffs.pred <- predict(glm.playoffs, data = traindata, type = "response")

ggplot(traindata, aes(x = FG., y = playoffs.pred)) + geom_point() +
geom_smooth(method = "glm", method.args = list(family = "quasibinomial"))
```



```
traindata$predplayoffs <- NA
for (i in 1:nrow(traindata))
{
  if (playoffs.pred[i] > 0.5)
  {
    traindata$predplayoffs[i] <- 1
  }
  else
  {
    traindata$predplayoffs[i] <- 0
  }
}

table(traindata$predplayoffs, traindata$playoffs)
```

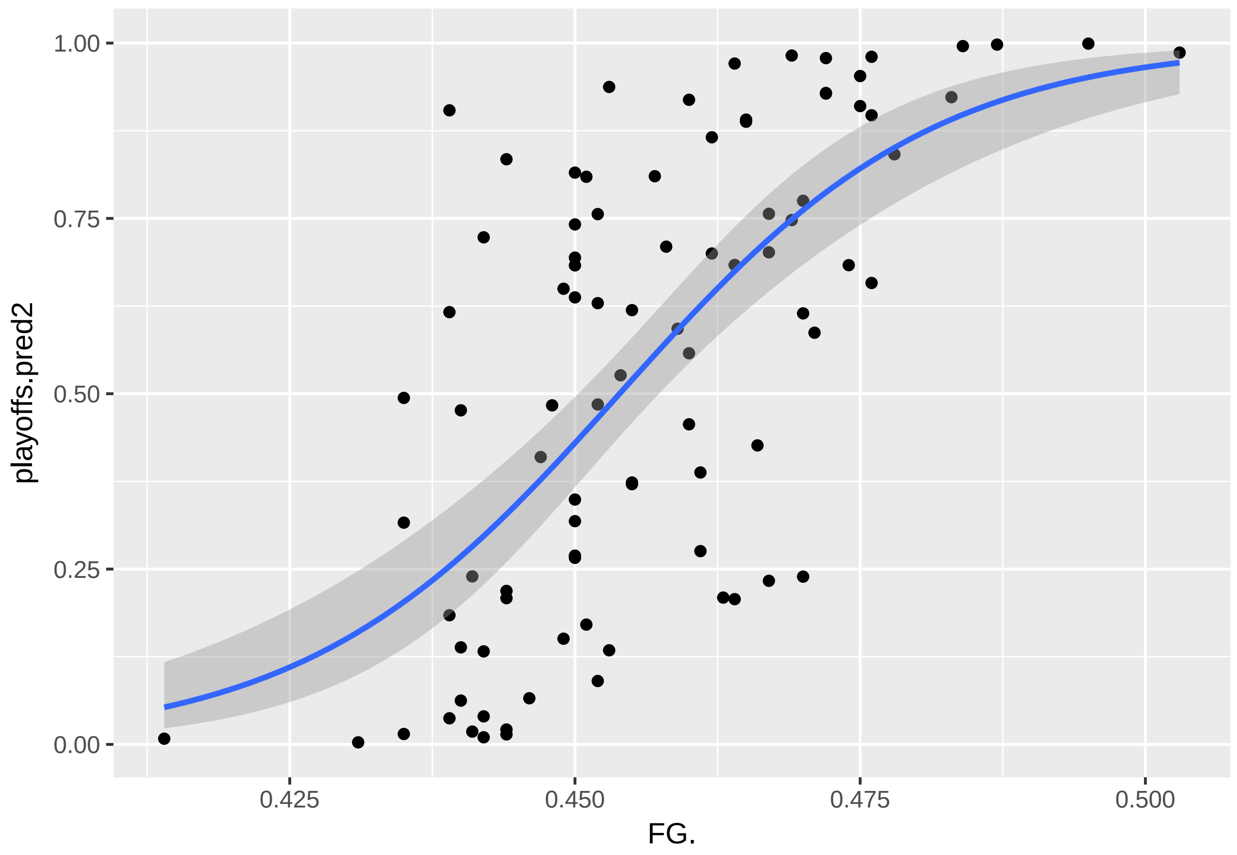
```
##
##       0   1
##   0 109  31
##   1  31 129
```

```
trainerr <- (1 - mean(traindata$predplayoffs == traindata$playoffs))
trainerr
```

```
## [1] 0.2066667
```

```
# performing logistic regression on the testing dataset
glm.playoffs2 <- glm(playoffs ~ FG. + X3P. + FT. + TRB + STL + TOV, data = testdata, family = binomial)
playoffs.pred2 <- predict(glm.playoffs2, data = testdata, type = "response")
```

```r
ggplot(testdata, aes(x = FG., y = playoffs.pred2)) + geom_point() +
geom_smooth(method = "glm", method.args = list(family = "quasibinomial"))
```



```r
testdata$predplayoffs <- NA
for (i in 1:nrow(testdata))
{
  if (playoffs.pred2[i] > 0.5)
  {
    testdata$predplayoffs[i] <- 1
  }
  else
  {
    testdata$predplayoffs[i] <- 0
  }
}

table(testdata$predplayoffs, testdata$playoffs)
```

```
##
##      0  1
##   0 30 11
##   1 12 37
```

```r
testerr <- (1 - mean(testdata$predplayoffs == testdata$playoffs))
testerr
```

```
## [1] 0.2555556
```

First, we performed LASSO for feature selection. Then, we performed Logistic Regression on the training dataset using only 6 of the 21 variables (FG%, 3P%, FT%, TRB, STL, TOV), and the response variable was "playoffs". The training error for Logistic Regression was 0.2067, and the test error was 0.2556. For this year (2017-18), the model correctly predicted the outcomes for 25 of the 30 teams.

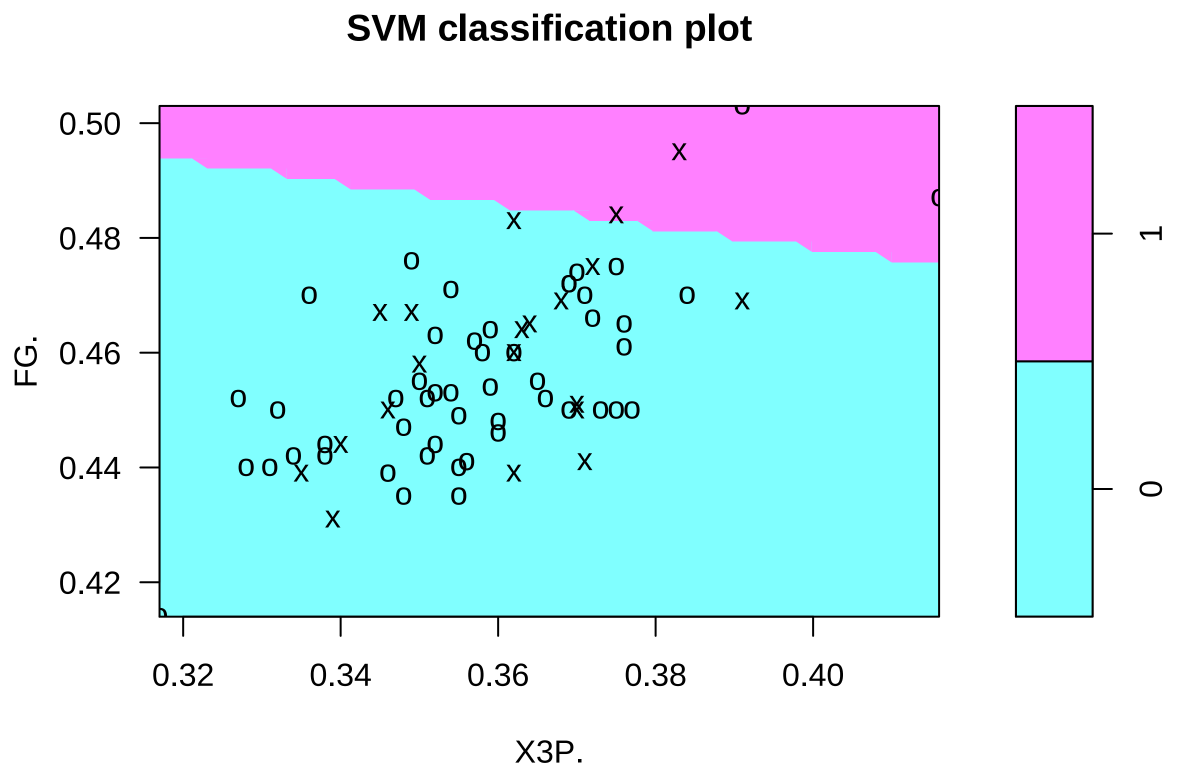
Next, we performed SVM on the datasets to again predict whether a team would make or miss the playoffs. This is another popular topic in machine learning today. It is appropriate since SVM is "hard classification," and this scenario fits due to the two classes. SVM allows us to explore if there is statistical separation (e.g. a hyperplane) between a typical playoff team and a typical non-playoff team. We performed linear, radial, and polynomial SVM after learning these techniques in the course. We also learned how closely related the support vector classifier is to the logistic regression model.

```r
# tuning a support vector classifier (linear svm) on the training dataset
library(e1071)
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.ltune <- tune(svm, playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS,
data = traindata, kernel = "linear", ranges = list(cost = c(0.4, 0.5, 0.6, 0.7, 0.8)))
# summary(NBA.ltune)
```

```r
# tuning a radial svm on the training dataset
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.rtune <- tune(svm, playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS,
data = traindata, kernel = "radial", ranges = list(cost = c(0.1, 1, 5, 10, 50),
gamma = c(0.001, 0.01, 0.05, 0.1, 0.5)))
# summary(NBA.rtune)
```

```r
# tuning a polynomial svm on the training dataset
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.ptune <- tune(svm, playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS,
data = traindata, kernel = "polynomial", ranges = list(cost = c(1, 5, 10, 50, 100),
degree = c(2, 3, 4, 5, 6)))
# summary(NBA.ptune)
```

```r
# performing linear svm with best cost
NBA.svml <- svm(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS, data = traindata, kernel = "linear", cost = 0.5)
# summary(NBA.svml)
plot(NBA.svml, testdata, FG. ~ X3P.)
```



```r
svml.pred <- predict(NBA.svml, testdata)
table(svml.pred, testdata$playoffs)
```

```
##
## svml.pred  0  1
##         0 23  2
##         1 19 46
```

```r
ltesterr <- (1 - mean(svml.pred == testdata$playoffs))
ltesterr
```

```
## [1] 0.2333333
```

```r
# performing radial svm with best cost and gamma
NBA.svmr <- svm(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS, data = traindata, kernel = "radial", cost = 5, gamma = 0.01)
# summary(NBA.svmr)

svmr.pred <- predict(NBA.svmr, testdata)
table(svmr.pred, testdata$playoffs)
```

```
##
## svmr.pred  0  1
##         0 25  4
##         1 17 44
```

```r
rtesterr <- (1 - mean(svmr.pred == testdata$playoffs))
rtesterr
```

```
## [1] 0.2333333
```

```r
# performing polynomial svm with best cost and degree
NBA.svmp <- svm(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB +
AST + STL + BLK + TOV + PF + PTS, data = traindata, kernel = "polynomial", cost = 50, degree = 3)
# summary(NBA.svmp)

svmp.pred <- predict(NBA.svmp, testdata)
table(svmp.pred, testdata$playoffs)
```

```
##
## svmp.pred  0  1
##         0 22  2
##         1 20 46
```

```r
ptesterr <- (1 - mean(svmp.pred == testdata$playoffs))
ptesterr
```

```
## [1] 0.2444444
```

For each SVM, we first tuned it on the training dataset using all 21 variables to determine the best cost/gamma/degree, and the response variable was “playoffs”. The CV error for linear SVM was 0.1433. Then, using the best cost of 0.5, the number of support vectors for linear SVM was 108 with a test error of 0.2333. The CV error for radial SVM was 0.1467. Then, using the best cost of 5 and gamma of 0.01, the number of support vectors for radial SVM was 144 with a test error of 0.2333. The CV error for polynomial SVM was 0.1967. Then, using the best cost of 50 and degree of 3, the number of support vectors for polynomial SVM was 141 with a test error of 0.2444. If we were to recommend a particular SVM to predict whether a team will make or miss the playoffs, we would recommend the linear SVM over radial and polynomial.

Next, we performed different tree-based methods on the datasets to again predict whether a team would make or miss the playoffs. We first created a classification decision tree with the leaves containing the qualitative response (0 for missing playoffs or 1 for making playoffs). Then we utilized tree pruning to obtain a subtree of the original decision tree, which resulted in a lower test error as expected. However, these methods produce trees with high variance, so we then performed bagging to reduce the variance through averaging. Finally, we performed random forests after noticing that “field goal percentage” was a very strong predictor in our other models. This method decorrelates the trees in some sense so that the variance reduction is larger compared to bagging. The decision tree method is appropriate due to its interpretability for a general audience such as organization management. Bagging and random forests methods are appropriate since they help decrease the inevitable high variance of individual trees.

```r
# creating a classification decision tree on the training dataset with all variables (NBA.tree)
library(tree)
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.tree <- tree(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + PTS, data = traindata)
# summary(NBA.tree)
# plot(NBA.tree)
# text(NBA.tree)
```

```r
# predicting the response on the test dataset and computing the test error rate with all variables
set.seed(1)
testdata$playoffs <- as.factor(testdata$playoffs)
tree.pred <- predict(NBA.tree, testdata, type = "class")
table(tree.pred, testdata$playoffs)
```

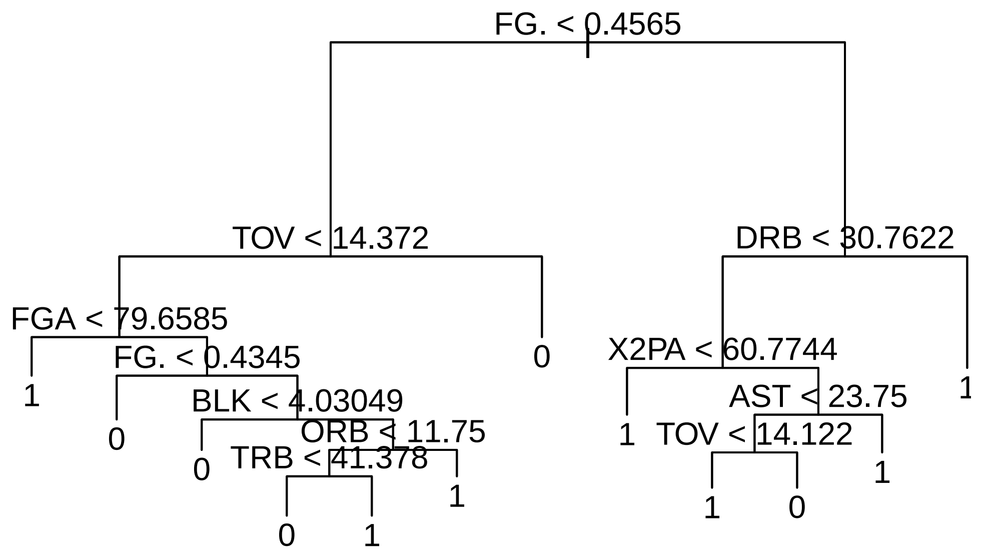
```
##
```

```
## tree.pred  0  1
##         0 20  9
##         1 22 39
```

```r
decisiontesterr <- (1 - mean(tree.pred == testdata$playoffs))
decisiontesterr
```

```
## [1] 0.3444444
```

```r
# determining optimal tree size, then performing tree pruning, then computing the test error rate
set.seed(1)
cv.NBAtree <- cv.tree(NBA.tree, FUN = prune.misclass)
prune.NBAtree <- prune.misclass(NBA.tree, best = 12)
# summary(prune.NBAtree)
plot(prune.NBAtree)
text(prune.NBAtree)
```



```r
prune.pred <- predict(prune.NBAtree, testdata, type = "class")
table(prune.pred, testdata$playoffs)
```

```
##
## prune.pred  0  1
##          0 21  8
##          1 21 40
```

```r
prunetesterr <- (1 - mean(prune.pred == testdata$playoffs))
prunetesterr
```

```
## [1] 0.3222222
```

```r
# performing bagging on the training dataset, then computing the test error rate/OOB error rate
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
```

```
##
##     margin
```

```
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.bag <- randomForest(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. +
FT + FTA + FT. + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + PTS, data = traindata, mtry = 21, impor

yhat.bag <- predict(NBA.bag, testdata)
table(yhat.bag, testdata$playoffs)
```

```
##
## yhat.bag  0  1
##        0 14  6
##        1 28 42
```

```
bagtesterr <- (1 - mean(yhat.bag == testdata$playoffs))
bagtesterr
```

```
## [1] 0.3777778
```

```
# NBA.bag
```

```
# performing random forests on the training dataset, then computing the test error rate/OOB error rate
set.seed(1)
traindata$playoffs <- as.factor(traindata$playoffs)

NBA.forest <- randomForest(playoffs ~ FG + FGA + FG. + X3P + X3PA + X3P. + X2P + X2PA + X2P. + FT + FTA

yhat.forest <- predict(NBA.forest, testdata)
table(yhat.forest, testdata$playoffs)
```

```
##
## yhat.forest  0  1
##           0 12  5
##           1 30 43
```

```
foresttesterr <- (1 - mean(yhat.forest == testdata$playoffs))
foresttesterr
```

```
## [1] 0.3888889
```

```
# NBA.forest
```

First, we created a classification decision tree on the training dataset using all 21 variables, and the response variable was "playoffs". The test error for the decision tree was 0.3444. Then, we performed tree pruning to obtain a subtree. The test error for the pruned tree was 0.3222. After working with the decision tree, we performed bagging as a way to reduce the variance. The test error for bagging was 0.3778, which is higher than both errors for the original decision tree and pruned tree. However, the OOB bagging error was only 0.2167. We also performed random forests as a way to decorrelate the trees for a larger variance reduction. The test error for random forests was 0.3889, which is the highest. However, the OOB random forests error was only 0.2233. If we were to recommend a particular tree-based method to predict whether a team will make or miss the playoffs, we would recommend either a) bagging (based on its OOB error) or b) a pruned tree (based on its test error).

Now suppose we were hired to present our findings to the Charlotte Hornets organization, where they asked us to report on the best way to predict a) the number of games they will win in a given regular season and b) whether they will make or miss the playoffs. We would tell them that, based on our findings, the best way to predict wins would be to use the LASSO model, and the best way to determine whether they make the playoffs or not would be to use the Logistic Regression model or the linear SVM model. For simplicity, we would likely recommend the Logistic Regression model since many in the audience do not have statistical backgrounds. It is also easier explaining the $P(Y = 1|X)$ probability produced from the predict() function (i.e. "the model predicts your team has a 60% chance of making the playoffs") as supposed to explaining how SVM works.

One potential drawback is the small pool of observations we were able to draw from. We needed to make sure our data was consistent from season to season. Today the NBA has 30 teams, and each team plays 82 games in the regular season. Therefore, we needed to find the earliest year in which this was the case, which happened to be 2004. Thus, the training data is from 04-05 to 14-15, which amounts to $n = 300$ observations (note: the data from 11-12 was excluded since the teams only played 66 games due to a lockout). The test data is from 15-16 to 17-18, which amounts to $n = 90$ observations. While this is nice for the 80%/20% training/test split, we wish we had more than $n = 390$ total observations.

One pitfall is the concern of overfitting. For some of the models such as tree-based methods, we used 21 predictors on the 300 training observations and then computed the respective test errors with the 90 test observations. An improvement of the analysis could be to use more cross-validation and other validation set approaches to address overfitting, as well as collect more observations over time and rerun the training models.