R Project - Classification

4375 Machine Learning with Dr. Mazidi

Anthony Martinez | netid: amm180005

10/17/21

Project: Using data from NBA games spanning from 2004-2015 I will us ML classification algorithms to predict the winner.

The data set was downloaded from Kaggle: https://www.kaggle.com/nathanlauga/nba-games (https://www.kaggle.com/nathanlauga/nba-games)

Load the data

```
df0<-read.csv("data/games.csv", header=TRUE)
```

Data Exploreation

- use at least 5 R functions for data exploration
- create at least 2 informative R graphs for data exploration

I am a little surprised that the mean of points scored for home teams was only 2.8 points higher than the mean of points scored for away teams. I was planning on using home court advantage as a predictor, but maybe it won't make the best predictor.

```
# Use head to get a peak at the first 5 observations in the data
head(df0)
```

	GAME_DATE… <chr>	GAME… <int>	GAME_STATUS_… <chr>	HOME_TEA… <int>	VISITOR_TEAM_ID <int>	SEA… <int>	TEAM_
1	5/26/2021	42000102	Final	1610612755	1610612764	2020	16106
2	5/26/2021	42000132	Final	1610612752	1610612737	2020	16106
3	5/26/2021	42000142	Final	1610612762	1610612763	2020	16106
4	5/25/2021	42000112	Final	1610612751	1610612738	2020	16106
5	5/25/2021	42000152	Final	1610612756	1610612747	2020	16106
6	5/25/2021	42000172	Final	1610612746	1610612742	2020	16106

6 rows | 1-8 of 22 columns

```
# Using summary to get statistics for each column
summary(df0)
```

```
##  GAME_DATE_EST         GAME_ID         GAME_STATUS_TEXT    HOME_TEAM_ID
##  Length:24677       Min.   :10300001   Length:24677       Min.   :1.611e+09
##  Class :character   1st Qu.:20600878   Class :character   1st Qu.:1.611e+09
##  Mode  :character   Median :21100897   Mode  :character   Median :1.611e+09
##                     Mean   :21644558                      Mean   :1.611e+09
##                     3rd Qu.:21601157                      3rd Qu.:1.611e+09
##                     Max.   :52000211                      Max.   :1.611e+09
##
##  VISITOR_TEAM_ID         SEASON      TEAM_ID_home          PTS_home
##  Min.   :1.611e+09   Min.   :2003   Min.   :1.611e+09   Min.   : 36.0
##  1st Qu.:1.611e+09   1st Qu.:2007   1st Qu.:1.611e+09   1st Qu.: 94.0
##  Median :1.611e+09   Median :2011   Median :1.611e+09   Median :102.0
##  Mean   :1.611e+09   Mean   :2011   Mean   :1.611e+09   Mean   :102.8
##  3rd Qu.:1.611e+09   3rd Qu.:2016   3rd Qu.:1.611e+09   3rd Qu.:111.0
##  Max.   :1.611e+09   Max.   :2020   Max.   :1.611e+09   Max.   :168.0
##                                                         NA's   :99
##   FG_PCT_home      FT_PCT_home      FG3_PCT_home       AST_home
##  Min.   :0.2500   Min.   :0.1430   Min.   :0.0000   Min.   : 6.00
##  1st Qu.:0.4210   1st Qu.:0.6960   1st Qu.:0.2860   1st Qu.:19.00
##  Median :0.4590   Median :0.7650   Median :0.3570   Median :22.00
##  Mean   :0.4603   Mean   :0.7591   Mean   :0.3561   Mean   :22.65
##  3rd Qu.:0.5000   3rd Qu.:0.8280   3rd Qu.:0.4290   3rd Qu.:26.00
##  Max.   :0.6840   Max.   :1.0000   Max.   :1.0000   Max.   :50.00
##  NA's   :99       NA's   :99       NA's   :99       NA's   :99
##     REB_home      TEAM_ID_away          PTS_away       FG_PCT_away
##  Min.   :15.00   Min.   :1.611e+09   Min.   : 33.00   Min.   :0.244
##  1st Qu.:39.00   1st Qu.:1.611e+09   1st Qu.: 91.00   1st Qu.:0.411
##  Median :43.00   Median :1.611e+09   Median : 99.00   Median :0.448
##  Mean   :43.27   Mean   :1.611e+09   Mean   : 99.91   Mean   :0.449
##  3rd Qu.:48.00   3rd Qu.:1.611e+09   3rd Qu.:109.00   3rd Qu.:0.487
##  Max.   :72.00   Max.   :1.611e+09   Max.   :168.00   Max.   :0.674
##  NA's   :99                          NA's   :99       NA's   :99
##   FT_PCT_away      FG3_PCT_away       AST_away       REB_away
##  Min.   :0.1430   Min.   :0.0000   Min.   : 4.0   Min.   :19.00
##  1st Qu.:0.6920   1st Qu.:0.2780   1st Qu.:18.0   1st Qu.:38.00
##  Median :0.7620   Median :0.3500   Median :21.0   Median :42.00
##  Mean   :0.7574   Mean   :0.3494   Mean   :21.3   Mean   :41.97
##  3rd Qu.:0.8280   3rd Qu.:0.4210   3rd Qu.:25.0   3rd Qu.:46.00
##  Max.   :1.0000   Max.   :1.0000   Max.   :46.0   Max.   :81.00
##  NA's   :99       NA's   :99       NA's   :99     NA's   :99
##  HOME_TEAM_WINS
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.5891
##  3rd Qu.:1.0000
##  Max.   :1.0000
##
```

```
# Using names to get the names of the columns in the data set
names(df0)
```

```
##  [1] "GAME_DATE_EST"    "GAME_ID"          "GAME_STATUS_TEXT" "HOME_TEAM_ID"
##  [5] "VISITOR_TEAM_ID"  "SEASON"           "TEAM_ID_home"     "PTS_home"
##  [9] "FG_PCT_home"      "FT_PCT_home"      "FG3_PCT_home"     "AST_home"
## [13] "REB_home"         "TEAM_ID_away"     "PTS_away"         "FG_PCT_away"
## [17] "FT_PCT_away"      "FG3_PCT_away"     "AST_away"         "REB_away"
## [21] "HOME_TEAM_WINS"
```

```
# just out of curiosity I wanted to see if there was any difference between names() and colnames
() functions
colnames(df0)
```

```
##  [1] "GAME_DATE_EST"    "GAME_ID"          "GAME_STATUS_TEXT" "HOME_TEAM_ID"
##  [5] "VISITOR_TEAM_ID"  "SEASON"           "TEAM_ID_home"     "PTS_home"
##  [9] "FG_PCT_home"      "FT_PCT_home"      "FG3_PCT_home"     "AST_home"
## [13] "REB_home"         "TEAM_ID_away"     "PTS_away"         "FG_PCT_away"
## [17] "FT_PCT_away"      "FG3_PCT_away"     "AST_away"         "REB_away"
## [21] "HOME_TEAM_WINS"
```

```
# using str() to get row/column counts and info on each column
str(df0)
```

```
## 'data.frame':    24677 obs. of  21 variables:
##  $ GAME_DATE_EST   : chr  "5/26/2021" "5/26/2021" "5/26/2021" "5/25/2021" ...
##  $ GAME_ID         : int  42000102 42000132 42000142 42000112 42000152 42000172 42000122 4200
0162 42000101 42000151 ...
##  $ GAME_STATUS_TEXT: chr  "Final" "Final" "Final" "Final" ...
##  $ HOME_TEAM_ID    : int  1610612755 1610612752 1610612762 1610612751 1610612756 1610612746 1
610612749 1610612743 1610612755 1610612756 ...
##  $ VISITOR_TEAM_ID : int  1610612764 1610612737 1610612763 1610612738 1610612747 1610612742 1
610612748 1610612757 1610612764 1610612747 ...
##  $ SEASON          : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ TEAM_ID_home    : int  1610612755 1610612752 1610612762 1610612751 1610612756 1610612746 1
610612749 1610612743 1610612755 1610612756 ...
##  $ PTS_home        : int  120 101 141 130 102 121 132 128 125 99 ...
##  $ FG_PCT_home     : num  0.557 0.383 0.544 0.523 0.465 0.536 0.489 0.535 0.495 0.465 ...
##  $ FT_PCT_home     : num  0.684 0.739 0.774 0.955 0.933 0.9 0.9 0.8 0.697 0.833 ...
##  $ FG3_PCT_home    : num  0.429 0.364 0.487 0.447 0.308 0.394 0.415 0.429 0.313 0.321 ...
##  $ AST_home        : int  26 15 28 31 21 23 34 29 27 24 ...
##  $ REB_home        : int  45 54 42 46 31 39 61 35 40 47 ...
##  $ TEAM_ID_away    : int  1610612764 1610612737 1610612763 1610612738 1610612747 1610612742 1
610612748 1610612757 1610612764 1610612747 ...
##  $ PTS_away        : int  95 92 129 108 109 127 98 109 118 90 ...
##  $ FG_PCT_away     : num  0.402 0.369 0.541 0.424 0.45 0.585 0.402 0.479 0.557 0.434 ...
##  $ FT_PCT_away     : num  0.633 0.818 0.763 0.783 0.871 0.542 0.686 0.821 0.8 0.607 ...
##  $ FG3_PCT_away    : num  0.091 0.273 0.348 0.353 0.303 0.529 0.286 0.485 0.4 0.269 ...
##  $ AST_away        : int  22 17 20 23 24 25 20 15 26 19 ...
##  $ REB_away        : int  40 41 33 43 39 34 36 40 41 33 ...
##  $ HOME_TEAM_WINS  : int  1 1 1 1 0 0 1 1 1 1 ...
```

```
# calculating mean on the PTS_home column
# Notice that we get NA for the answer
# This means we must have missing values in this column
mean(df0$PTS_home)
```

```
## [1] NA
```

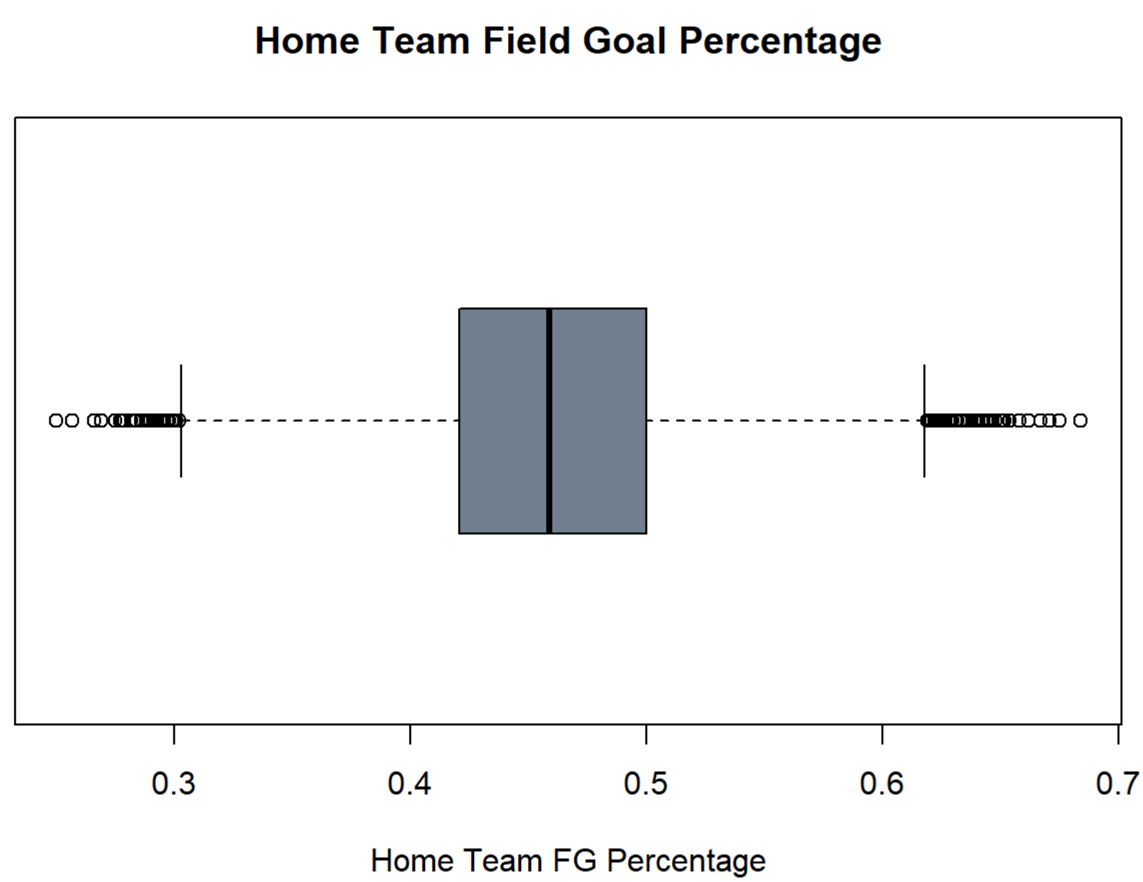
```
# I will remove the na's from the columns that will be used in the model during the data cleanin
g portion
mean(df0$PTS_home, na.rm=TRUE)
```

```
## [1] 102.7666
```

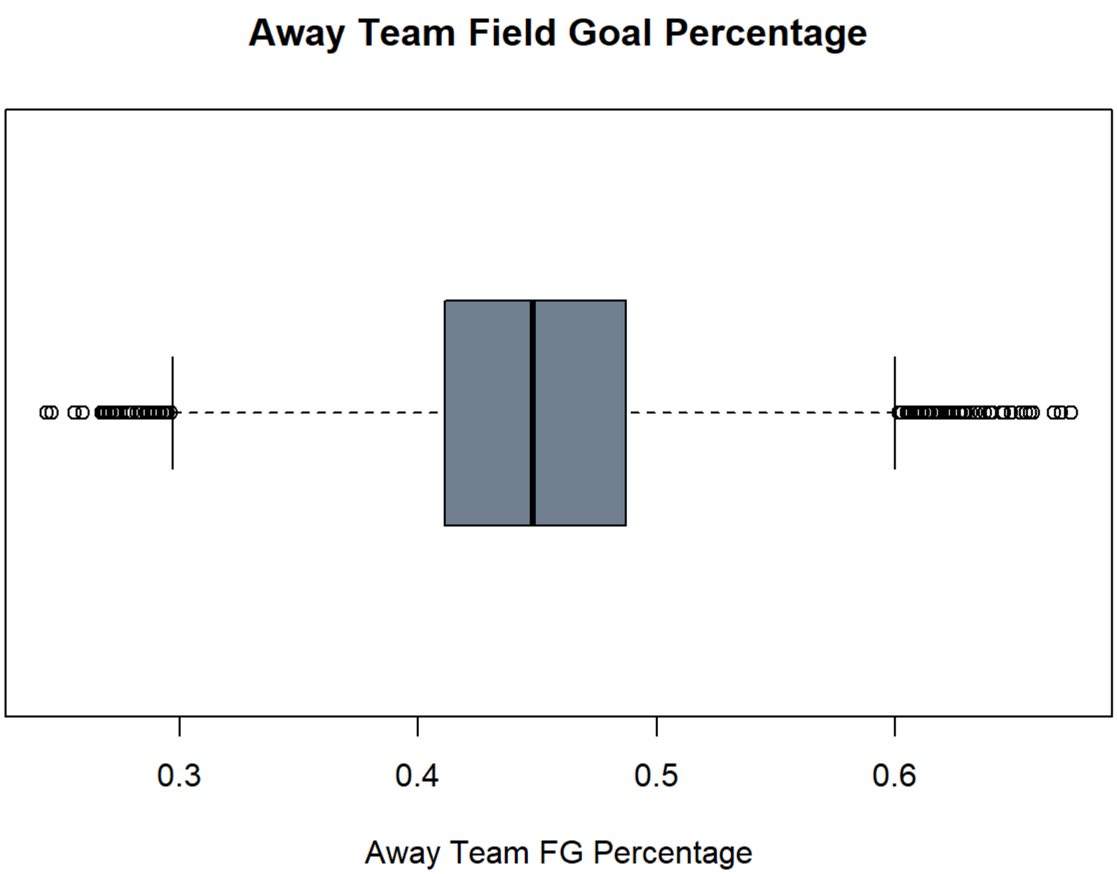
```
# Get mean of PTS scored for away teams
mean(df0$PTS_away, na.rm=TRUE)
```

```
## [1] 99.90764
```

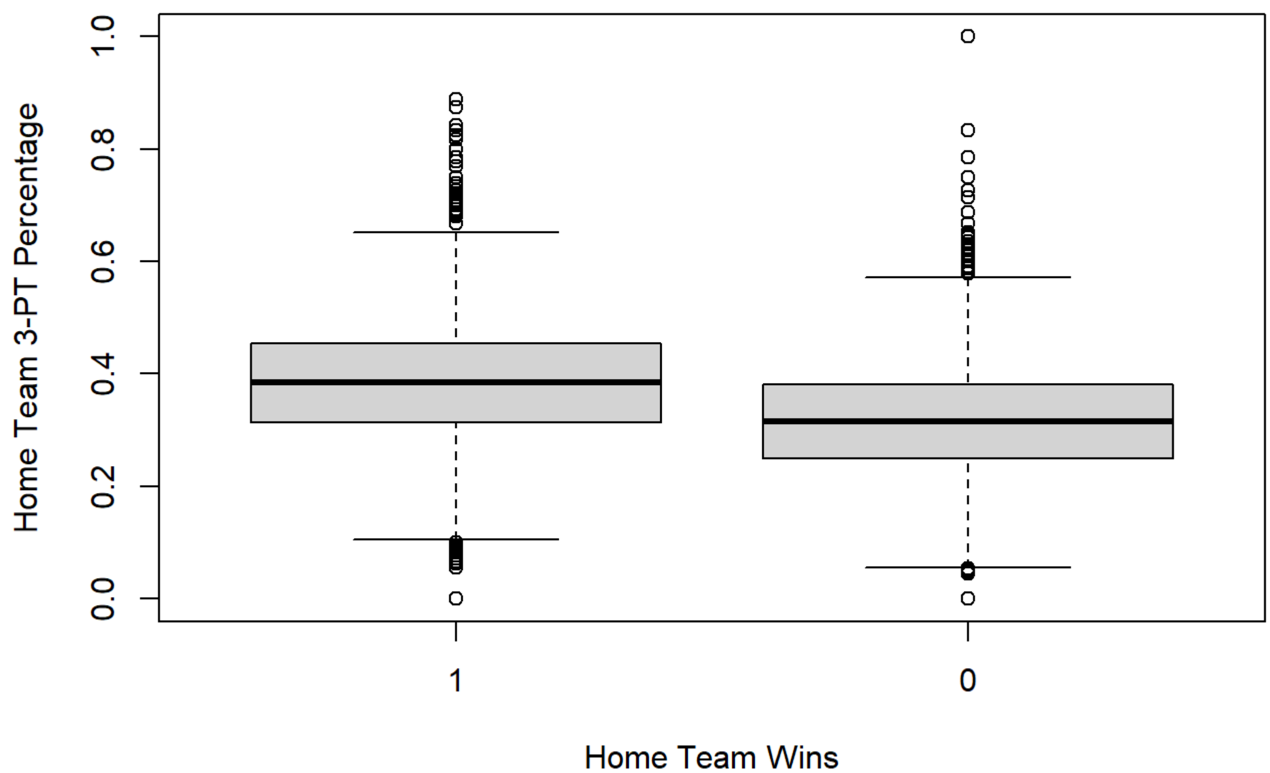
```
boxplot(df0$FG_PCT_home, col="slategray", horizontal=TRUE, xlab="Home Team FG Percentage",
main="Home Team Field Goal Percentage")
```



```
boxplot(df0$FG_PCT_away, col="slategray", horizontal=TRUE, xlab="Away Team FG Percentage",
main="Away Team Field Goal Percentage")
```




```
# making column factor so we can plot
df0$HOME_TEAM_WINS <- factor(df0$HOME_TEAM_WINS, levels=c("1", "0"))
plot(df0$HOME_TEAM_WINS,df0$FG3_PCT_home, xlab="Home Team Wins", ylab="Home Team 3-PT Percentag
e")
```




```
#There seens to be a relationship between the home team winning and the home team's
# 3 point percentage
```

Data Cleaning

- Link to data: https://www.kaggle.com/nathanlauga/nba-games (https://www.kaggle.com/nathanlauga/nba-games)
- describe what steps you had to do for data cleaning (more points for messier data that needed cleaning)

I Performed the following steps: 1) Deleted columns that were note needed or unable to help us predict winning teams 2) count the NAs in each of the columns 3) Replace NA with mean

```
# The follwing columns are not very useful for what we are trying to accomplish
# which is, predicting who will win

#game_id: col2
#Game_Status_Text : col 3
#home_team_id id : col 4
#visitor_team_id: col 5
#team_id_home : 7
#team_id_away: 14
#game status

#count NAs
sapply(df0, function(x) sum(is.na(x)))
```

```
##    GAME_DATE_EST          GAME_ID GAME_STATUS_TEXT     HOME_TEAM_ID
##                0                0                0                0
##  VISITOR_TEAM_ID           SEASON     TEAM_ID_home         PTS_home
##                0                0                0               99
##      FG_PCT_home      FT_PCT_home     FG3_PCT_home         AST_home
##               99               99               99               99
##         REB_home     TEAM_ID_away         PTS_away      FG_PCT_away
##               99                0               99               99
##      FT_PCT_away     FG3_PCT_away         AST_away         REB_away
##               99               99               99               99
##   HOME_TEAM_WINS
##                0
```

```
# remove columns that are unnessary
df <- df0[-c(2,3,4,5,7,14)]


# removing those columns coinicidently made it easier to replace NAs in the columns that contain
ed NAs
df$PTS_home[is.na(df$PTS_home)] <- mean(df$PTS_home, na.rm=TRUE)
df$FG_PCT_home[is.na(df$FG_PCT_home)] <- mean(df$FG_PCT_home, na.rm=TRUE)
df$FT_PCT_home[is.na(df$FT_PCT_home)] <- mean(df$FT_PCT_home, na.rm=TRUE)
df$FG3_PCT_home[is.na(df$FG3_PCT_home)] <- mean(df$FG3_PCT_home, na.rm=TRUE)
df$AST_home[is.na(df$AST_home)] <- mean(df$AST_home, na.rm=TRUE)
df$REB_home[is.na(df$REB_home)] <- mean(df$REB_home, na.rm=TRUE)
df$PTS_away[is.na(df$PTS_away)] <- mean(df$PTS_away, na.rm=TRUE)
df$FG_PCT_away[is.na(df$FG_PCT_away)] <- mean(df$FG_PCT_away, na.rm=TRUE)
df$FT_PCT_away[is.na(df$FT_PCT_away)] <- mean(df$FT_PCT_away, na.rm=TRUE)
df$FG3_PCT_away[is.na(df$FG3_PCT_away)] <- mean(df$FG3_PCT_away, na.rm=TRUE)
df$AST_away[is.na(df$AST_away)] <- mean(df$AST_away, na.rm=TRUE)
df$REB_away[is.na(df$REB_away)] <- mean(df$REB_away, na.rm=TRUE)

#show na's are deleted
sapply(df, function(x) sum(is.na(x)))
```

```
##  GAME_DATE_EST        SEASON      PTS_home    FG_PCT_home    FT_PCT_home
##              0             0             0              0              0
##   FG3_PCT_home      AST_home      REB_home       PTS_away    FG_PCT_away
##              0             0             0              0              0
##    FT_PCT_away  FG3_PCT_away      AST_away       REB_away HOME_TEAM_WINS
##              0             0             0              0              0
```

Divide train/test

- Divide into 75/25 train/test, using seed 1234

```
# your code here

set.seed(1234)
i <- sample(1:nrow(df), .75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Algorithm 1: Logistic Regression

- code to run the algorithms
- commentary on feature selection you selected and why
- code to compute your metrics for evaluation as well as commentary discussing the

Commentary on Features Chosen: I decided to predict based on the field goal percentage of both 2-pointers and 3 pointers for each team. Winning comes down to who scores more points. However, predicting simply on the number of points is too simple. The number of points scored can vary due to the quality of defence from team to team. The percentage of made shots will vary less.

Commentary on Results: The algorithm was able to predict the home team winning with only 50% accuracy.

```
attach(df)
glm1 <- glm(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_away, data=train, family
="binomial")

summary(glm1)
```

```
##
## Call:
## glm(formula = HOME_TEAM_WINS ~ FG_PCT_home + FG_PCT_away + FG3_PCT_home +
##     FG3_PCT_away, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1512  -0.6237  -0.2079   0.6176   3.2858
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.5941     0.2288  -2.596  0.00942 **
## FG_PCT_home  -25.8019     0.5110 -50.495  < 2e-16 ***
## FG_PCT_away   26.4695     0.5164  51.259  < 2e-16 ***
## FG3_PCT_home  -4.2062     0.2102 -20.010  < 2e-16 ***
## FG3_PCT_away   4.0325     0.2086  19.332  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25031  on 18506  degrees of freedom
## Residual deviance: 15087  on 18502  degrees of freedom
## AIC: 15097
##
## Number of Fisher Scoring iterations: 5
```

```
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>.5,2,1)
log_reg_acc <- mean(pred==test$HOME_TEAM_WINS)
print(paste("accuracy = ", log_reg_acc))
```

```
## [1] "accuracy =  0.50194489465154"
```

```
table(pred,test$HOME_TEAM_WINS)
```

```
##
## pred    1    0
##    1 3097  661
##    2  490 1922
```

```
detach()
```

Alogorithm 2: Naive Bayes

- code to run the algorithms
- commentary on feature selection you selected and why
- code to compute your metrics for evaluation as well as commentary discussing the results

Commentary on Features Chosen: I decided to predict based on the field goal percentage of both 2-pointers and 3 pointers for each team. Winning comes down to who scores more points. However, predicting simply on the number of points is too simple. The number of points scored can vary due to the quality of defence from team to team. The percentage of made shots will vary less.

Commentary of Results: The Naive Bayes model had an accuracy of 80% which is very good. State of the art NBA prediting projects have an accuracy around 85%.

```
attach(df)
set.seed(1234)
i <- sample(1:nrow(df), .75*nrow(df), replace=FALSE)
train2 <- df[i,]
test2 <- df[-i,]

library(e1071)

nb1 <- naiveBayes(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_away, data=train2)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        1        0
## 0.591614 0.408386
##
## Conditional probabilities:
##    FG_PCT_home
## Y        [,1]       [,2]
##   1 0.4802080 0.05260234
##   0 0.4315018 0.04880490
##
##    FG_PCT_away
## Y        [,1]       [,2]
##   1 0.4288209 0.05001222
##   0 0.4779595 0.05057603
##
##    FG3_PCT_home
## Y        [,1]      [,2]
##   1 0.3833086 0.1106196
##   0 0.3166135 0.1033954
##
##    FG3_PCT_away
## Y        [,1]      [,2]
##   1 0.3224652 0.1056632
##   0 0.3876930 0.1075137
```

```
p1 <- predict(nb1, newdata=test2, type="class")
table(p1, test2$HOME_TEAM_WINS)
```

```
##
## p1     1    0
##   1 3089  697
##   0  498 1886
```

```
naive_acc <- mean(p1==test$HOME_TEAM_WINS)
print(naive_acc)
```

```
## [1] 0.8063209
```

```
detach(df)
```

Algorithm 3: SVM

- code to run the algorithms
- commentary on feature selection you selected and why
- code to compute your metrics for evaluation as well as commentary discussing the results

Commentary on Features Chosen: I decided to predict based on the field goal percentage of both 2-pointers and 3 pointers for each team. Winning comes down to who scores more points. However, predicting simply on the number of points is too simple. The number of points scored can vary due to the quality of defence from team to team. The percentage of made shots will vary less.

Commentary of Results: The SVM model had an accuracy of 81% which is the highest out of all 3 models.

```
attach(df)
set.seed(1234)
i <- sample(1:nrow(df), .75*nrow(df), replace=FALSE)
train3 <- df[i,]
test3 <- df[-i,]

library(e1071)
svm1 <- svm(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_away, data=train3, kerne
l="linear", cost=10, scale=TRUE)
detach(df)
```

SVM Results

```
summary(svm1)
```

```
##
## Call:
## svm(formula = HOME_TEAM_WINS ~ FG_PCT_home + FG_PCT_away + FG3_PCT_home +
##     FG3_PCT_away, data = train3, kernel = "linear", cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  8367
##
##  ( 4183 4184 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 0
```

```
pred3 <- predict(svm1, newdata=test3)
table(pred3,test3$HOME_TEAM_WINS)
```

```
##
## pred3    1    0
##     1 3082  643
##     0  505 1940
```

```
svm_acc <- mean(pred3==test3$HOME_TEAM_WINS)
svm_acc
```

```
## [1] 0.8139384
```

Results analysis

- rank the algorithms from best to worst performing on your data
- add commentary on the performance of the algorithms
- your analysis concerning why the best performing algorithm worked best on that data
- commentary on what your script was able to learn from the data (big picture) and if this is likely to be useful

Ranking Commentary: As we can see from the results SVM had the highest accuracy at 81.4%. Naive Bayes came in at a close second with 80.6% and logistic regression scored the lowest with only 50.2% accuracy.

Commentary on Performance: Logistic Regression: The algorithm was the worst preforming out of the three with only 50% accuracy. It correctly classifed 3097 postive values and 1922 negative values. While incorrectly classifying 661 positive values and 490 negative values.

Naive Bayes: Naive Bayes had the 2nd highest accuracy at 80.6%. The model was able to correctly classify 3089 postive values and 1886 negaitve values. While incorrectly classifying 498 negative values and 687 positive values.

SVM: SVM had the highest accuracy score with 81%. The model correctly classified 3082 postive values and 1940 negative values. While misclassifying 505 negative values and 643 positive values.

Analysis on best preforming algorithm: The fact that my SVM model was the top preforming alrogithm was not surprising to me. SVMs are used by the top preforming NBA game predicting projects. The reason why SVMs work well with data sets such as NBA statistics is because they contain many interdependent and related features. For example, points scored is somewhat related to the amount of shots taken, or the number of assisted is somewhat intertwined with the number of points scored, etc. SVMs are able to create multi-dimensional feature vectors which means they are capable of capturing the interactions between these realted feautres in the statistics.

Big Picture: What the script was able to from the data is that sports statistics, such as NBA statistics for this data set, have many intertwined statsitcs that have some relationship with one another. Because of this, some ML algorithms can have a difficult time correctly classifying results. When it comes to dealing with data sets with many related features, such as statistics for sports, Support Vector Machines are a good option to deal with these related features due to thier abillity to create multi-dimensional feature vectors.

```
print(paste("Logistic Regression accuracy", log_reg_acc))
```

```
## [1] "Logistic Regression accuracy 0.50194489465154"
```

```
print(paste("Naive Bayes accuracy", naive_acc))
```

```
## [1] "Naive Bayes accuracy 0.806320907617504"
```

```
print(paste("Support Vector Machine (SVM) accuracy", svm_acc))
```

```
## [1] "Support Vector Machine (SVM) accuracy 0.813938411669368"
```