MarchMadness

Sam Porter

5/11/2020

```
# Introduction
## This project will be looking at College Basketball data from the 2014-2019 seasons.
## The data was collected from the following location: https://www.kaggle.com/andrewsundberg/college-ba
## The goal of this project will be to predict if a team will make it to the March Madness tournament,
## In order to determine if a user will make it to the March Madness Tournament, we will need to perfor
## We will predict based off a binary variable (1 = you made the March Madness Tournament and 0 = you d
## Given this is a binary variable, we will clean up the data and perform a logistic regression, decisi
## To assess the model accuracy, we will review a ROC curve and confusion matrix to determine the overa

### Install Packages
install.packages('dplyr', repos = "http://cran.us.r-project.org")
```

```
## package 'dplyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```
install.packages('caret', repos = "http://cran.us.r-project.org")
```

```
## package 'caret' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```
install.packages('purrr', repos = "http://cran.us.r-project.org")
```

```
## package 'purrr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```
install.packages('tidyr', repos = "http://cran.us.r-project.org")
```

```
## package 'tidyr' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```r
install.packages('ggplot2', repos = "http://cran.us.r-project.org")
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```r
install.packages('InformationValue', repos = "http://cran.us.r-project.org")
```

```
## package 'InformationValue' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\portesa\AppData\Local\Temp\RtmpAx3mkO\downloaded_packages
```

```r
### Load necessary packages
library(dplyr)
```

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

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

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

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

```r
library(tidyr)
library(ggplot2)
library(InformationValue)
```

```
##
## Attaching package: 'InformationValue'
## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, sensitivity, specificity
```

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Data Import
### Import 5 years worth of datatsets
data1 <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb14.csv")
data1 <- subset(data1, select=-c(REC))
data2 <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb15.csv")
data2 <- subset(data2, select=-c(POSTSEASON))
data3 <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb16.csv")
data3 <- subset(data3, select=-c(POSTSEASON))
data4 <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb17.csv")
data4 <- subset(data4, select=-c(POSTSEASON))
data5 <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb18.csv")
data5 <- subset(data5, select=-c(POSTSEASON))
test_data <- read.csv("C:\\Users\\portesa\\Desktop\\Dataset\\cbb19.csv")
test_data <- subset(test_data, select=-c(POSTSEASON))

### Combine Data from 2014-2018 to serve as our training set
data <- union(data1,data2)
data <- union(data,data3)
data <- union(data, data4)
data <- union(data, data5)

## We have several columns of data. Let me first explain what is being stored in each column.
### TEAM = The Division 1 college basketball school
### CONF = The Athletic Conference in which the school participates in
### G = Number of Games played
### W = Number of games won
### ADJOE = Adjusted Offensive Efficiency (An estimate of the offensive officiency (points scored per 1
### ADJDE = Adjusted Defensive Efficiency (An estimate of the defensive efficiency (points allowed per
### BARTHAG = Power Rating (Chance of beating an average D1 team)
### EFG_O = Effective Field Goal Percentage Shot
### EFG_D = Effective Field Goal Percentage Allowed
### TOR = Turnover Percentage Allowed (Turnover Rate)
### TORD = Turnover Percentage Committed (Steal Rate)
### ORB = Offensive Rebound Percentage
### DRB = Defensive Rebound Percentage
### FTR = Free Throw Rate (How often the given team shoots Free Throws)
### FTRD = Free Throw Rate Allowed
### 2P_O = Two-Point Shooting Percentage
### 2P_D = Two-Point Shooting Percentage Allowed
### 3P_O = Three-Point Shooting Percentage
### 3P_D = Three-Point Shooting Percentage Allowed
### ADJ_T = Adjusted Tempo (An estimate of the temp (possessions per 40 minutes) a team would have agai
### WAB = Wins above Bubble (The bubble refers to the cut off between making the NCC March Madness Toru
### POSTSEASON = Round where the given team was eliminated or where their season ended
### Seed = Seed in the NCAA March Madness Tournament

head(data)
```

```
##         TEAM CONF  G  W ADJOE ADJDE BARTHAG EFG_O EFG_D  TOR TORD  ORB  DRB
## 1 Louisville Amer 37 31 118.8  87.6  0.9710  53.5  43.9 15.3 25.0 37.1 32.7
## 2    Arizona  P12 38 33 116.2  87.4  0.9636  51.7  42.3 15.7 19.1 36.4 27.3
## 3    Florida  SEC 39 36 115.9  88.4  0.9575  52.2  45.4 17.5 21.3 35.3 28.0
## 4   Virginia  ACC 37 30 114.6  89.5  0.9449  50.8  44.2 16.5 18.4 33.9 25.8
## 5  Wisconsin  B10 38 30 122.7  95.9  0.9441  53.3  47.2 12.7 15.3 28.1 27.4
## 6       Duke  ACC 35 26 125.9  98.6  0.9432  53.8  49.3 14.6 18.5 35.2 31.3
##    FTR FTRD X2P_O X2P_D X3P_O X3P_D ADJ_T  WAB SEED
## 1 41.2 38.4  52.7  44.3  36.8  28.6  68.8  5.3    4
## 2 41.0 34.2  50.7  40.2  36.4  32.0  64.3  9.4    1
## 3 42.4 31.2  51.3  43.5  35.9  33.0  63.1 11.7    1
## 4 42.0 32.5  49.0  42.1  36.9  32.3  61.2  8.2    1
## 5 42.7 27.1  51.3  45.9  37.6  34.1  63.9  7.9    2
## 6 38.8 40.8  50.3  50.3  39.5  30.7  66.7  6.5    3
```

```
# Data Exploration/Data Visualization and Data Cleaning
## In this section we will be taking a look at the distributions of the different features.
## This will help give us a better understanding of the data and what type of feature engineering we ma
## Before moving forward, we should check and address how we will deal with null values, convert any ca

## First let's look at a summary of the data and check to see how many NA values are in the dataset by
summary(data)
```

```
##      TEAM               CONF                 G               W        
##  Length:1755        Length:1755        Min.   :15.00   Min.   : 0.00  
##  Class :character   Class :character   1st Qu.:30.00   1st Qu.:11.00  
##  Mode  :character   Mode  :character   Median :31.00   Median :16.00  
##                                        Mean   :31.45   Mean   :16.23  
##                                        3rd Qu.:33.00   3rd Qu.:21.00  
##                                        Max.   :40.00   Max.   :38.00  
##                                                                       
##      ADJOE           ADJDE           BARTHAG           EFG_O     
##  Min.   : 76.7   Min.   : 84.00   Min.   :0.0077   Min.   :39.4  
##  1st Qu.: 98.8   1st Qu.: 99.15   1st Qu.:0.2842   1st Qu.:47.8  
##  Median :103.4   Median :103.80   Median :0.4740   Median :49.8  
##  Mean   :103.8   Mean   :103.79   Mean   :0.4938   Mean   :49.9  
##  3rd Qu.:108.5   3rd Qu.:108.30   3rd Qu.:0.7135   3rd Qu.:51.9  
##  Max.   :129.1   Max.   :124.00   Max.   :0.9842   Max.   :59.8  
##                                                                  
##      EFG_D            TOR             TORD            ORB       
##  Min.   :39.60   Min.   :11.90   Min.   :10.20   Min.   :15.00  
##  1st Qu.:48.10   1st Qu.:17.20   1st Qu.:17.00   1st Qu.:27.15  
##  Median :50.10   Median :18.50   Median :18.40   Median :29.90  
##  Mean   :50.09   Mean   :18.54   Mean   :18.47   Mean   :29.86  
##  3rd Qu.:52.00   3rd Qu.:19.80   3rd Qu.:19.80   3rd Qu.:32.55  
##  Max.   :59.50   Max.   :26.10   Max.   :28.00   Max.   :42.10  
##                                                                  
##       DRB             FTR            FTRD          X2P_O           X2P_D      
##  Min.   :18.40   Min.   :21.6   Min.   :22.1   Min.   :38.30   Min.   :37.70  
##  1st Qu.:28.00   1st Qu.:32.9   1st Qu.:32.3   1st Qu.:46.60   1st Qu.:46.70  
##  Median :30.00   Median :36.4   Median :36.5   Median :48.70   Median :49.00  
##  Mean   :30.06   Mean   :36.6   Mean   :36.9   Mean   :48.81   Mean   :48.98  
##  3rd Qu.:32.00   3rd Qu.:40.2   3rd Qu.:41.0   3rd Qu.:51.00   3rd Qu.:51.30  
##  Max.   :40.40   Max.   :58.6   Max.   :60.7   Max.   :62.60   Max.   :59.80  
##                                                                                
##      X3P_O           X3P_D           ADJ_T            WAB         
##  Min.   :25.20   Min.   :27.10   Min.   :57.20   Min.   :-25.200  
##  1st Qu.:32.60   1st Qu.:33.10   1st Qu.:65.70   1st Qu.:-12.900  
##  Median :34.60   Median :34.70   Median :67.90   Median : -8.300  
##  Mean   :34.57   Mean   :34.76   Mean   :67.91   Mean   : -7.768  
##  3rd Qu.:36.40   3rd Qu.:36.40   3rd Qu.:70.00   3rd Qu.: -3.050  
##  Max.   :44.10   Max.   :43.10   Max.   :83.40   Max.   : 13.100  
##                                                                    
##       SEED       
##  Min.   : 1.000  
##  1st Qu.: 5.000  
##  Median : 9.000  
##  Mean   : 8.794  
##  3rd Qu.:13.000  
##  Max.   :16.000  
##  NA's   :1415
```

```
sapply(data, function(x) sum(is.na(x)))
```

```
##    TEAM    CONF       G       W   ADJOE   ADJDE BARTHAG   EFG_O   EFG_D     TOR 
##       0       0       0       0       0       0       0       0       0       0 
##    TORD     ORB     DRB     FTR    FTRD   X2P_O   X2P_D   X3P_O   X3P_D   ADJ_T 
##       0       0       0       0       0       0       0       0       0       0 
##     WAB    SEED 
##       0    1415
```

```
## View variable types to see which variables need to be converted to categorical
sapply(data,class)
```

```
##        TEAM        CONF           G           W       ADJOE       ADJDE 
## "character" "character"   "integer"   "integer"   "numeric"   "numeric" 
##     BARTHAG       EFG_O       EFG_D         TOR        TORD         ORB 
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##         DRB         FTR        FTRD       X2P_O       X2P_D       X3P_O
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##       X3P_D       ADJ_T         WAB        SEED
##   "numeric"   "numeric"   "numeric"   "integer"
```

```
## View how many conferences we have and then convert them to a numeric categorical variable
data <- data %>% mutate(CONF = toupper(CONF))
conference <- data %>% group_by(CONF) %>% summarize(ct = n())
conference <- conference %>% mutate(CONF_rating = as.numeric(factor(conference$CONF, levels = conferenc
data <- data %>% inner_join(conference)
```

```
## Joining, by = "CONF"
```

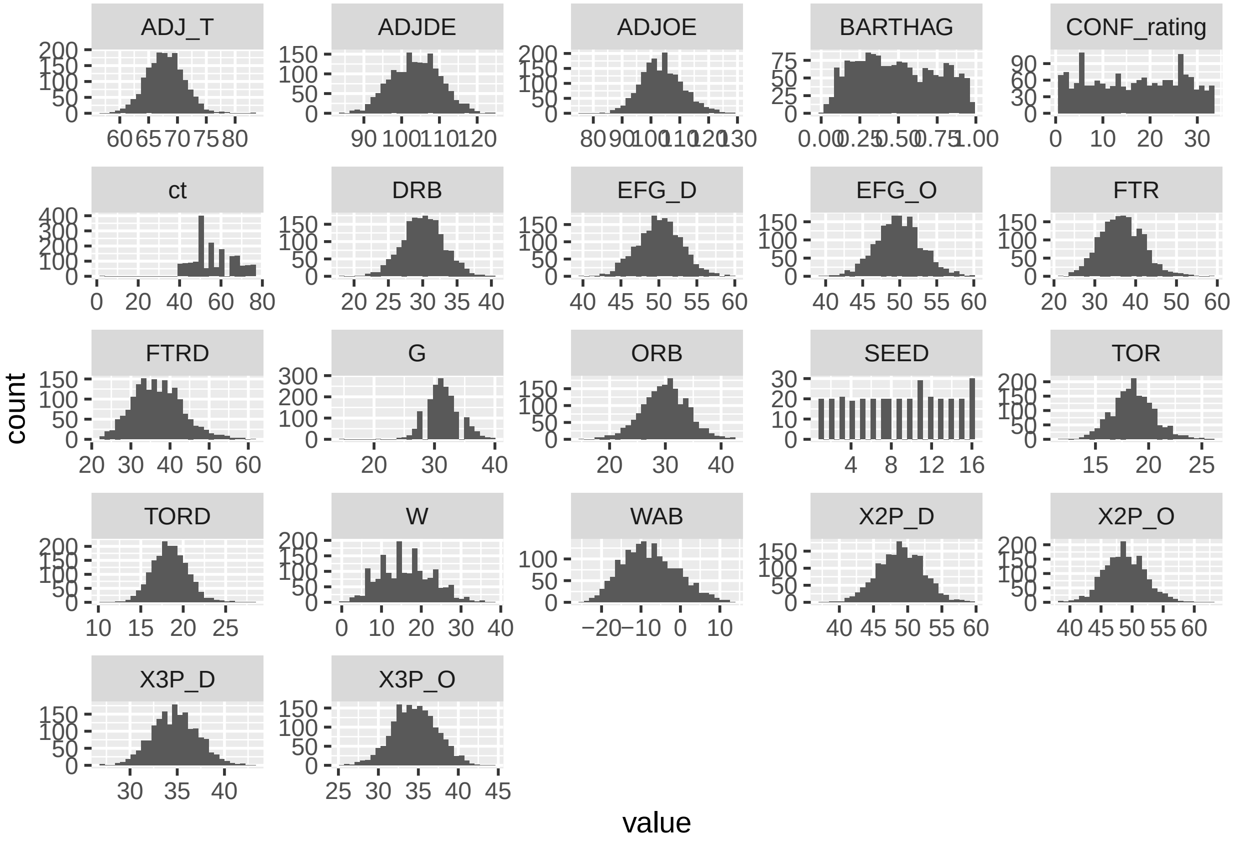
```
## Same as above, but with the test_data
test_data <- test_data %>% mutate(CONF = toupper(CONF))
conference <- test_data %>% group_by(CONF) %>% summarize(ct = n())
conference <- conference %>% mutate(CONF_rating = as.numeric(factor(conference$CONF, levels = conferenc
test_data <- test_data %>% inner_join(conference)
```

```
## Joining, by = "CONF"
```

```
## Plot a histogram of all the variables to see what the distribution
data %>% keep(is.numeric) %>% gather() %>% ggplot(aes(value)) + facet_wrap(~ key, scales = "free") + ge
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1415 rows containing non-finite values (stat_bin).
```



```
## Convert seed value to binary 1 or 0 response, so that we can have this as our categorical dependent
data <- data %>% mutate(SEED = ifelse(data$SEED > 0,1, 0))

## Same as above, but with the test_data
test_data <- test_data %>% mutate(SEED = ifelse(test_data$SEED > 0,1, 0))

## Look at the distribution of the y_variable (if a team makes it to the NCAA March Madness Tournament)
data %>% group_by(SEED) %>% summarize(ct = n())
```

```
## # A tibble: 2 x 2
##    SEED    ct
##   <dbl> <int>
## 1     1   340
## 2    NA  1415
```

```
## Convert NA values in SEED column to 0
data[is.na(data)] <- 0
data %>% group_by(SEED) %>% summarize(ct = n())
```

```
## # A tibble: 2 x 2
##    SEED    ct
##   <dbl> <int>
## 1     0  1415
## 2     1   340
```

```
test_data[is.na(test_data)] <- 0
test_data %>% group_by(SEED) %>% summarize(ct = n())
```

```
## # A tibble: 2 x 2
##    SEED    ct
##   <dbl> <int>
## 1     0   285
## 2     1    68
```

```
## Remove TEAM and CONF variable(s) from data set
data <- subset(data, select=-c(TEAM,CONF))
test_data <- subset(test_data, select=-c(TEAM,CONF))

## Look at correlation matrix of variables. Check for multicollinearity between features and little to
cor_matrix <- cor(data)
cor_matrix
```

```
##                       G           W       ADJOE      ADJDE     BARTHAG
## G            1.00000000  0.72741530  0.60821747 -0.6083603  0.69111583
## W            0.72741530  1.00000000  0.75481409 -0.7015319  0.82715747
## ADJOE        0.60821747  0.75481409  1.00000000 -0.5084621  0.86753300
## ADJDE       -0.60836034 -0.70153195 -0.50846210  1.0000000 -0.84572219
## BARTHAG      0.69111583  0.82715747  0.86753300 -0.8457222  1.00000000
## EFG_O        0.34162805  0.61287960  0.73018854 -0.2133903  0.54296282
## EFG_D       -0.48457007 -0.60496068 -0.31786887  0.7895983 -0.62287269
## TOR         -0.33984937 -0.46324822 -0.61325708  0.2141322 -0.47963312
## TORD         0.04872899  0.13059830 -0.13307952 -0.2272126  0.03767243
## ORB          0.26286748  0.29884952  0.26458412 -0.2900252  0.31211379
## DRB         -0.19828609 -0.38393679 -0.27348907  0.3762657 -0.35856088
## FTR          0.08875274  0.14307770  0.10068912 -0.1118690  0.12975523
## FTRD        -0.27299135 -0.33633504 -0.35256428  0.2043428 -0.32364957
## X2P_O        0.33278525  0.58075380  0.64739786 -0.2389901  0.51038591
## X2P_D       -0.44042776 -0.53817756 -0.31741363  0.7377599 -0.59242809
## X3P_O        0.23376291  0.44041517  0.58230979 -0.1100500  0.39830254
## X3P_D       -0.36049984 -0.47116430 -0.18708797  0.5542103 -0.41980766
## ADJ_T       -0.04730856 -0.01447517  0.05865379  0.2157609 -0.08131731
## WAB          0.66042635  0.91047203  0.84743224 -0.7992955  0.94064751
## SEED         0.51273772  0.61427628  0.54927685 -0.4947986  0.57569763
## ct           0.25249579  0.14859110  0.21777907 -0.2223562  0.25938951
## CONF_rating -0.22646604 -0.14860795 -0.24703048  0.2257733 -0.26860431
##                   EFG_O       EFG_D         TOR        TORD          ORB
## G            0.34162805 -0.48457007 -0.33984937  0.04872899  0.262867476
## W            0.61287960 -0.60496068 -0.46324822  0.13059830  0.298849522
## ADJOE        0.73018854 -0.31786887 -0.61325708 -0.13307952  0.264584124
## ADJDE       -0.21339030  0.78959830  0.21413216 -0.22721259 -0.290025215
## BARTHAG      0.54296282 -0.62287269 -0.47963312  0.03767243  0.312113792
## EFG_O        1.00000000 -0.10441756 -0.36119178 -0.13481262 -0.150245393
## EFG_D       -0.10441756  1.00000000  0.09312867 -0.00253898 -0.351246798
## TOR         -0.36119178  0.09312867  1.00000000  0.10275940  0.105882548
## TORD        -0.13481262 -0.00253898  0.10275940  1.00000000  0.090427807
## ORB         -0.15024539 -0.35124680  0.10588255  0.09042781  1.000000000
## DRB         -0.31717413  0.18169747  0.17302351  0.25245828  0.006822504
## FTR         -0.06522234 -0.21155100  0.12741818  0.07369014  0.305153216
## FTRD        -0.37582375  0.11041193  0.28269571  0.35094602  0.129249506
## X2P_O        0.89583705 -0.13607217 -0.29297426 -0.07386592 -0.089119420
## X2P_D       -0.09653918  0.91423521  0.08652803  0.04364090 -0.343974802
## X3P_O        0.77075159 -0.03473390 -0.31614358 -0.16778084 -0.144829053
## X3P_D       -0.07959945  0.72124288  0.05951326 -0.09735751 -0.216605655
## ADJ_T        0.11925478  0.28252025 -0.09095158 -0.03991964 -0.104923300
## WAB          0.56499047 -0.61885255 -0.47827622  0.07869446  0.330585400
## SEED         0.36668569 -0.38009418 -0.30757065  0.07886072  0.223222217
## ct           0.01828070 -0.13930976 -0.08559411  0.03046872  0.167313583
## CONF_rating -0.11829201  0.10919006  0.16726977  0.07176024 -0.032608267
##                      DRB         FTR        FTRD       X2P_O       X2P_D
## G           -0.198286088  0.08875274 -0.27299135  0.33278525 -0.44042776
## W           -0.383936786  0.14307770 -0.33633504  0.58075380 -0.53817756
## ADJOE       -0.273489067  0.10068912 -0.35256428  0.64739786 -0.31741363
## ADJDE        0.376265691 -0.11186896  0.20434279 -0.23899007  0.73775990
## BARTHAG     -0.358560878  0.12975523 -0.32364957  0.51038591 -0.59242809
## EFG_O       -0.317174126 -0.06522234 -0.37582375  0.89583705 -0.09653918
## EFG_D        0.181697467 -0.21155100  0.11041193 -0.13607217  0.91423521
## TOR          0.173023509  0.12741818  0.28269571 -0.29297426  0.08652803
## TORD         0.252458284  0.07369014  0.35094602 -0.07386592  0.04364090
## ORB          0.006822504  0.30515322  0.12924951 -0.08911942 -0.34397480
## DRB          1.000000000  0.09358617  0.25978076 -0.28458627  0.21142781
## FTR          0.093586172  1.00000000  0.24562548 -0.01277812 -0.18990468
## FTRD         0.259780765  0.24562548  1.00000000 -0.35378281  0.10541539
## X2P_O       -0.284586271 -0.01277812 -0.35378281  1.00000000 -0.12008941
## X2P_D        0.211427806 -0.18990468  0.10541539 -0.12008941  1.00000000
## X3P_O       -0.249603179 -0.09109361 -0.26343959  0.41895111 -0.04056623
## X3P_D        0.065672214 -0.14087387  0.08795645 -0.11217027  0.38619936
## ADJ_T        0.005635813 -0.03110244 -0.02897766  0.15341478  0.27760537
## WAB         -0.323179142  0.17169680 -0.32691902  0.53093879 -0.57335587
## SEED        -0.163452376  0.10553267 -0.22081746  0.34025810 -0.33914280
## ct           0.063139199  0.08030849 -0.07650024  0.05873517 -0.13014637
## CONF_rating  0.106462800  0.07739226  0.21496195 -0.11470394  0.11325549
##                   X3P_O       X3P_D        ADJ_T         WAB        SEED
## G            0.23376291 -0.36049984 -0.047308561  0.66042635  0.51273772
## W            0.44041517 -0.47116430 -0.014475165  0.91047203  0.61427628
## ADJOE        0.58230979 -0.18708797  0.058653792  0.84743224  0.54927685
## ADJDE       -0.11005002  0.55421026  0.215760876 -0.79929553 -0.49479862
## BARTHAG      0.39830254 -0.41980766 -0.081317307  0.94064751  0.57569763
## EFG_O        0.77075159 -0.07959945  0.119254784  0.56499047  0.36668569
## EFG_D       -0.03473390  0.72124288  0.282520253 -0.61885255 -0.38009418
## TOR         -0.31614358  0.05951326 -0.090951575 -0.47827622 -0.30757065
## TORD        -0.16778084 -0.09735751 -0.039919639  0.07869446  0.07886072
## ORB         -0.14482905 -0.21660566 -0.104923300  0.33058540  0.22322222
## DRB         -0.24960318  0.06567221  0.005635813 -0.32317914 -0.16345238
## FTR         -0.09109361 -0.14087387 -0.031102436  0.17169680  0.10553267
## FTRD        -0.26343959  0.08795645 -0.028977657 -0.32691902 -0.22081746
## X2P_O        0.41895111 -0.11217027  0.153414782  0.53093879  0.34025810
## X2P_D       -0.04056623  0.38619936  0.277605369 -0.57335587 -0.33914280
## X3P_O        1.00000000 -0.01307475  0.029215806  0.41611471  0.27614625
## X3P_D       -0.01307475  1.00000000  0.169875574 -0.44071107 -0.28936929
## ADJ_T        0.02921581  0.16987557  1.000000000 -0.06114658 -0.02533463
## WAB          0.41611471 -0.44071107 -0.061146583  1.00000000  0.64495850
## SEED         0.27614625 -0.28936929 -0.025334629  0.64495850  1.00000000
## ct          -0.04334642 -0.10433557 -0.003012734  0.23199150  0.11404568
## CONF_rating -0.07647019  0.06361994  0.047068267 -0.23008867 -0.16459425
##                       ct CONF_rating
## G            0.252495789 -0.22646604
## W            0.148591097 -0.14860795
## ADJOE        0.217779071 -0.24703048
## ADJDE       -0.222356217  0.22577328
## BARTHAG      0.259389512 -0.26860431
## EFG_O        0.018280701 -0.11829201
## EFG_D       -0.139309764  0.10919006
## TOR         -0.085594108  0.16726977
## TORD         0.030468715  0.07176024
## ORB          0.167313583 -0.03260827
## DRB          0.063139199  0.10646280
## FTR          0.080308489  0.07739226
## FTRD        -0.076500245  0.21496195
## X2P_O        0.058735172 -0.11470394
## X2P_D       -0.130146367  0.11325549
## X3P_O       -0.043346420 -0.07647019
## X3P_D       -0.104335568  0.06361994
## ADJ_T       -0.003012734  0.04706827
## WAB          0.231991502 -0.23008867
## SEED         0.114045679 -0.16459425
## ct           1.000000000 -0.19638948
## CONF_rating -0.196389482  1.00000000
```

```
# Analysis/Interpretation
## First I decided to look at all of the variables to see how many NA values were present.
## After further exploration, I was able to tell the the SEED and POSTSEASON columns were the only vari
## This is because if a team does not make the March Madness tournament, they are not given a postseaso
## We view the class type of the data and convert the conference variable to a categorical variable. We
## After plotting a histogram of all of the variables, I can conclude that all variables (except SEED,
## This could cause a potential issue later on because if we do not have a balanced data set in terms o
## I noticed that we had 1,132 rows of NA values in the SEED column. In order to make this a binary pre
## Finally, we remove any unnecessary columns and run a correlation matrix to ensure all variables have

## Split the data into 10% test set and 90% train set
### Seeing that the data set is not too large (as far as big data goes), we will have the largest possi
set.seed(42)
test_index <- createDataPartition(y = data$SEED, times=1, p=0.1, list=FALSE)
test_set <- data[test_index,]
train_set <- data[-test_index,]

## K-fold Cross validation
cv_param <- trainControl(method="cv", number = 11)

# Model Building
## Logistic Regression
log_reg <- train(SEED~G+ADJOE+ADJDE+ct, data = train_set, method = 'glm')
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

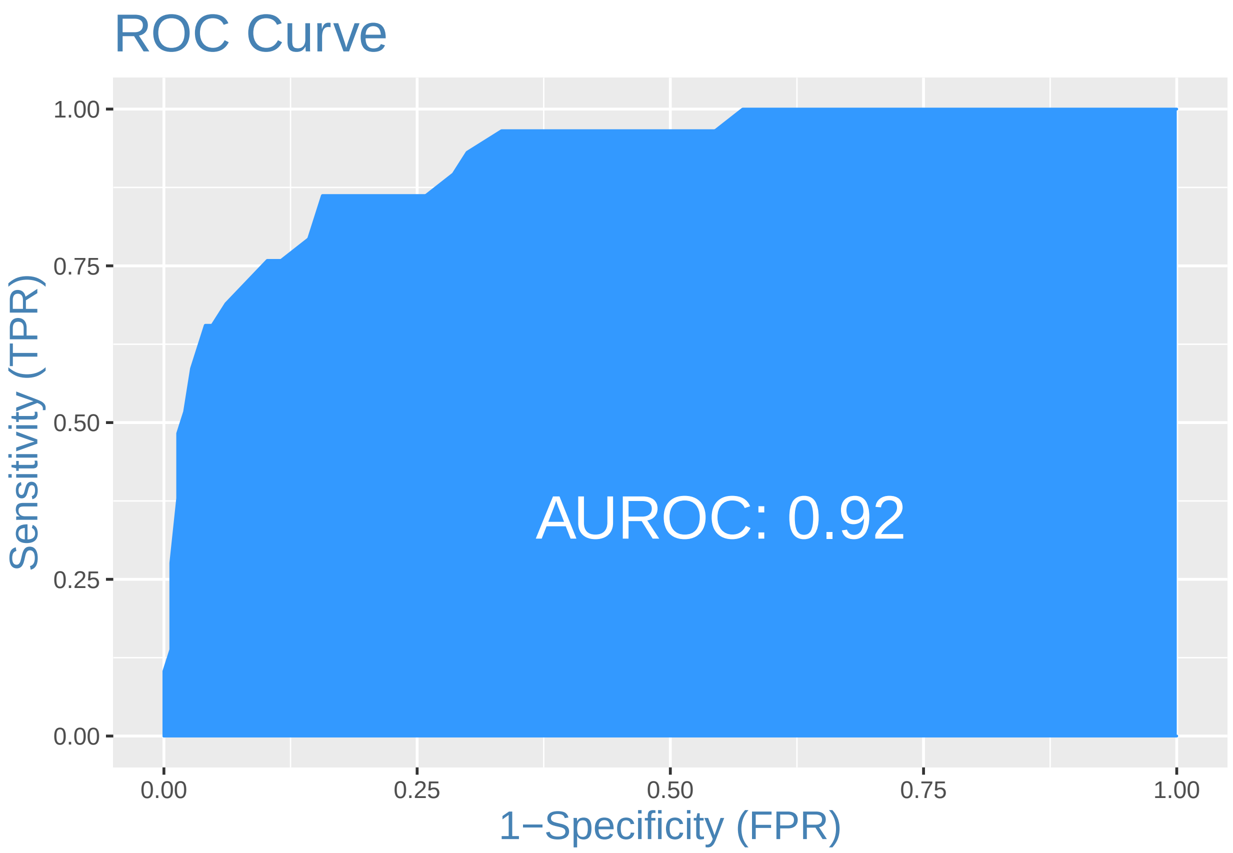
```
summary(log_reg)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.72564  -0.21978  -0.05676   0.15542   1.05386
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.0604518  0.2743861  -3.865 0.000116 ***
## G            0.0289815  0.0041222   7.031 3.06e-12 ***
## ADJOE        0.0180153  0.0013780  13.074  < 2e-16 ***
## ADJDE       -0.0133845  0.0015459  -8.658  < 2e-16 ***
## ct          -0.0023993  0.0008356  -2.871 0.004141 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.09697998)
##
##     Null deviance: 249.75  on 1578  degrees of freedom
## Residual deviance: 152.65  on 1574  degrees of freedom
## AIC: 803.8
##
## Number of Fisher Scoring iterations: 2
```

```
log_predictions <- predict(log_reg,test_set)
confusionMatrix(round(log_predictions,digits=0), test_set$SEED)
```

```
##     0  1
## 0 144  3
## 1  14 15
```

```
plotROC(test_set$SEED,log_predictions)
```



```
### Our logistic regression shows an Area under the curve score of .9253.

## Decision Tree
tree_ml <- train(SEED~., data = train_set, method = 'rpart', trControl = cv_param)
```

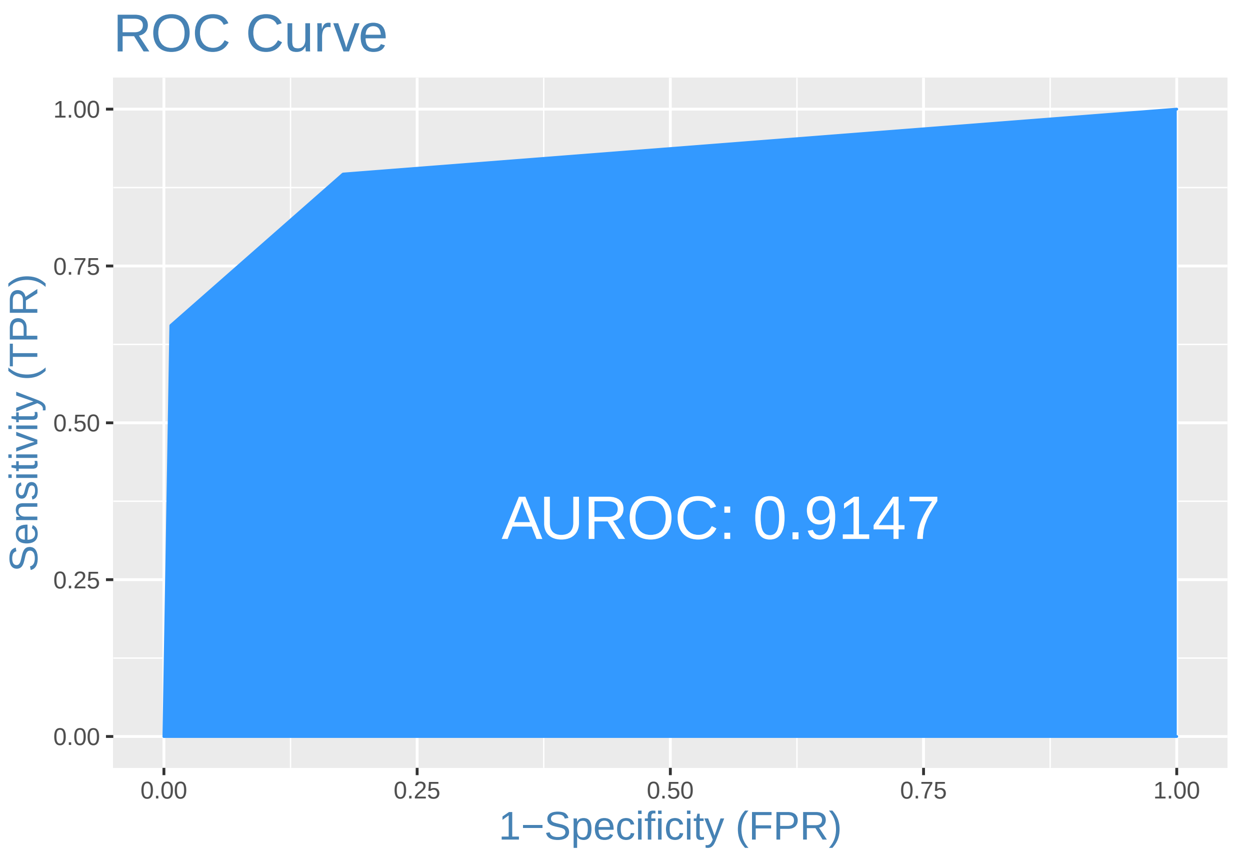
```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
tree_predictions <- predict(tree_ml, test_set)
confusionMatrix(round(tree_predictions,digits=0), test_set$SEED)
```

```
##      0   1
## 0 146   1
## 1  10  19
```

```
plotROC(test_set$SEED,tree_predictions)
```



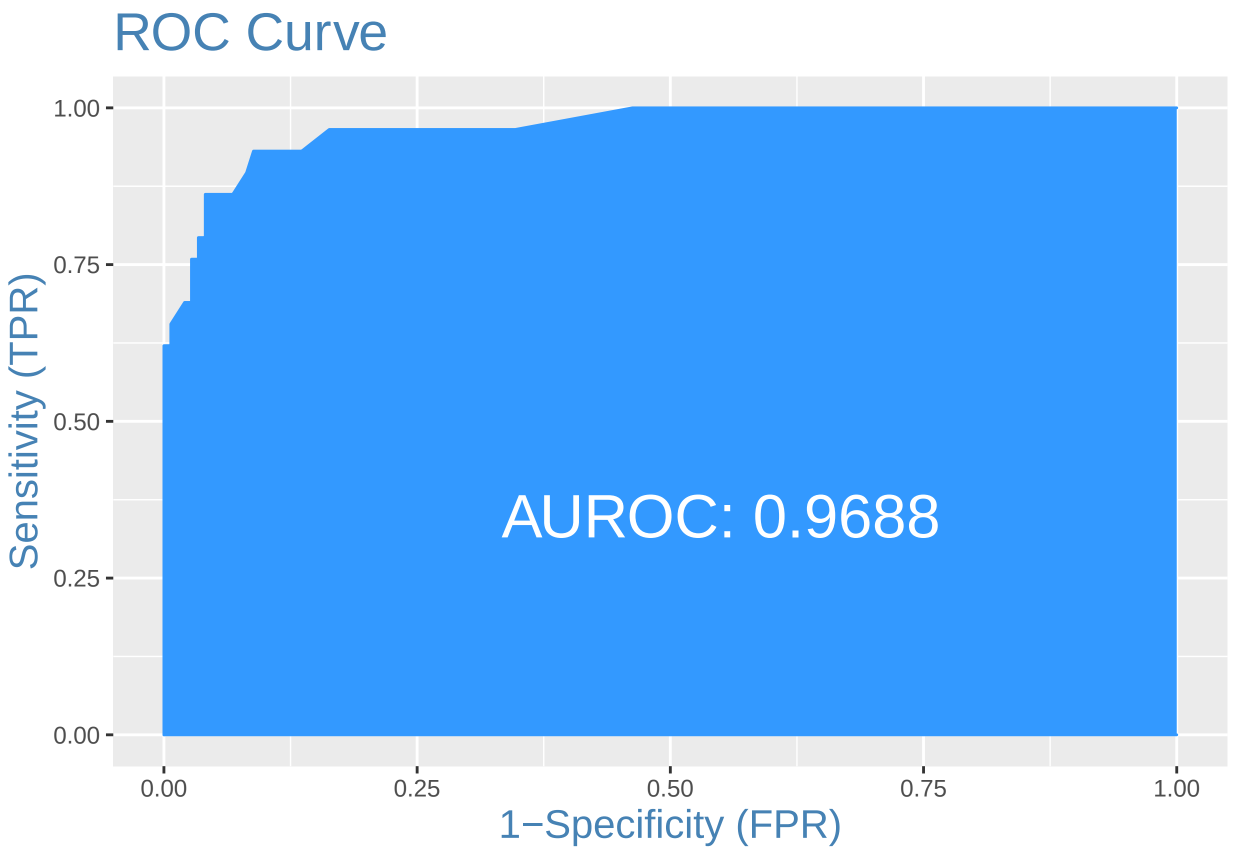
```
### Our decision tree shows an Area under the curve score of .7583.

## Random Forest
set.seed(42)
rf_model <- randomForest(SEED~., data = train_set, boosting=TRUE, trControl = cv_param)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
rf_predictions <- predict(rf_model, test_set)
confusionMatrix(round(rf_predictions,digits=0),test_set$SEED)
```

```
##     0  1
## 0 146  1
## 1  10 19
```

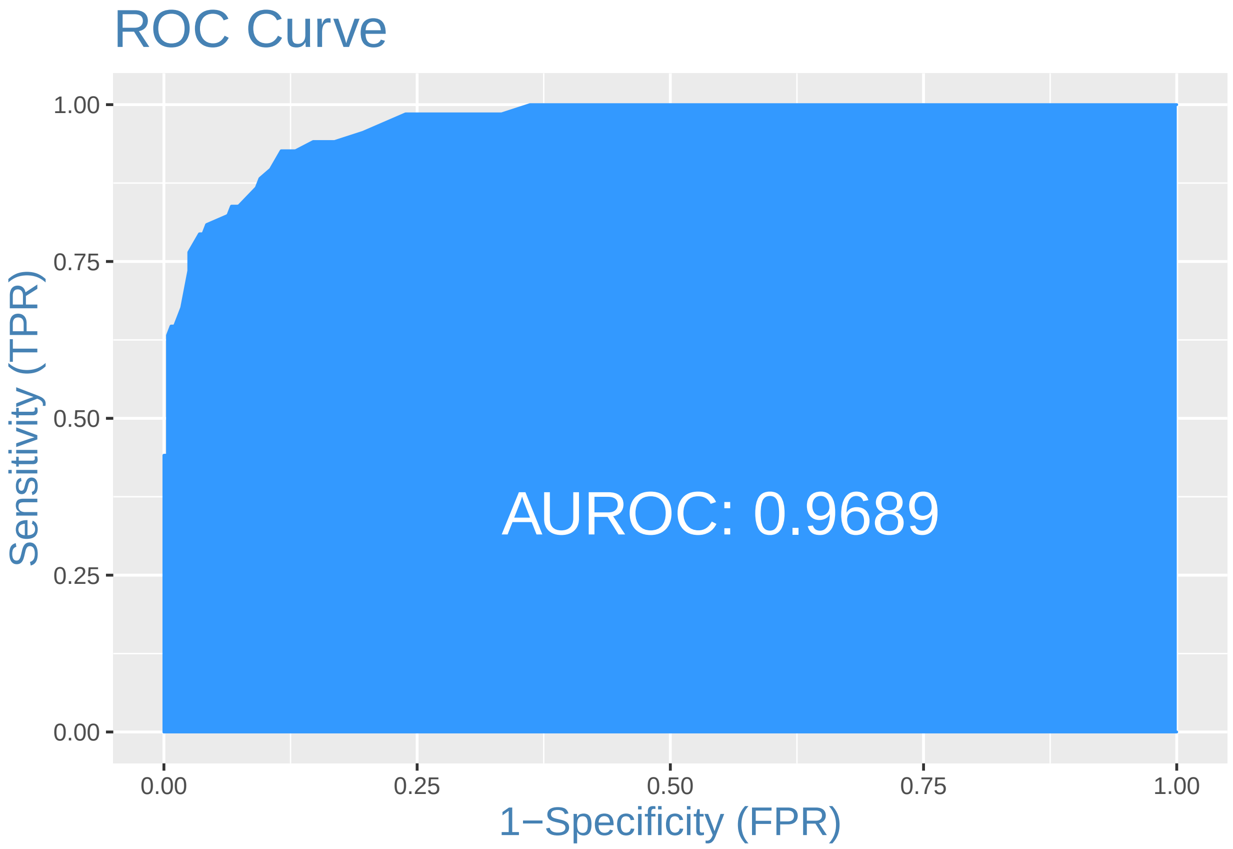
```
plotROC(test_set$SEED,rf_predictions)
```



```
### Our random forest shows an Area under the curve score of .935.

# Results Using our Best Model
final_predictions <- predict(rf_model,test_data)
confusionMatrix(round(final_predictions,digits=0),test_data$SEED)
```

```
##     0  1
## 0 278  7
## 1  18 50
```

```
plotROC(test_data$SEED,final_predictions)
```



```
## We plot an ROC curve to see how much better our model is to someone randomly guessing.
## An ROC Curve plots the True Positive rate (predicted 1 that were true 1's) over our False Positive R
## Using our Random Forest model, we are able to predict which teams will make the NCAA March Madness T
## It appears that the most significant variables in predicting whether a team will make the NCAA March
## Other variables that showed statistical significance include BARTHAG and WAB, but had to be removed
## We are also able to see that we had a 89.29% True Positive Rate and a 93.94% True Negative Rate.
## Given the disproportinate number of teams that did not make the tournament, it makes sense that we a

# Conclusion
## Given that we had a training dataset of 1,750 teams over 5 years, we could have a slightly overfit m
## Even though this dataset accounts for roughly 55,000+ NCAA basketball games, more data would definit
## In conclusion, it appears that we did a great job cleaning up the dataset to predict whether or not
## It is currently limited because there are many other factors that could come into play as to why a t
## Some of those factors include: winning conference championships, location of conference championship
## These limitations will mostly impact the lower seeds in the bracket as they do not have a guaranteed
## In the future, I could have looked into adding more data and incorporating additional factors that w
```