CFB Predict Final Results from Recruiting Analysis 1.0

Jacob Tesar

September 2, 2020

Test Question

It is well known that college recruiting will make a large impact on the success of a college football team. The best college football programs will sign (meaning the recuit accepts the scholarship and attends the school) the top high school recruits and most-often continue to have the most success on the field. But each and every year there are teams that do not recuit at the highest level and have success on the field, or vice versa where a team recruits at a very high level and does not achieve as much success as expected on the field. This is not a debate of whether high school recruiting matters to on-field succes, but rather HOW MUCH does high school recruiting correlate to on-field success at the college level.

To conduct this study I am relying on two key data sets that are widely regarded and highly respected for those familiar with college football. The first is 247.com, arguably the leading website in tracking high school recruiting and compiling 'talent levels' of college programs based on their level of high school recruiting. Each season 247.com will post a talent rank for the upcoming season (listed as the variable 'talent_rank' in the data) which will use the roster for the upcoming season and include the evaluation level of each recruit in their high school senior year. The rosters will be ranked using their own formula, which for this study I decided not use the specific number produced from the ranking but rather the ranking itself listed in descending order (1 is the most talented, 2 the second, and so on...). This is not a perfect evaluation..for example since it uses the evaluation from the high school level it does not take into account their production/evaluation at the college level. An incoming freshman may be ranked equally as high as an upcoming senior who is likely to contribute much more to the team in the upcoming season. But I have made the assumption that overall we will proceed that each player has the ability to equally contribute to team success (think of the saying "best players will play").

The second key data set I am using is also highly regarded in the college football world, which is the S&P ratings produced by sports analyst Jeff Sagarin. His rankings will use several unique formulas that go beyond records and standard media rankings, such as including strength of schedule, close wins v large wins, tempo of offense and many more. At the end of the season a final set of S&P ratings are produced that rank all Division 1 football teams using the key formulas. This practice is very well respected in college football and are considered to me far more accurate than the popular coaching or media rankings produced (who are ofter biased to teams with the most coverage, star players, over-empasizing final records, etc...).

For this study I can look in hindsight on past seasons on the 'talent ranking' of a team going into the season based on 247.com and match it with the S&P rating at the end of the respective season. The current set includes the talent level and S&P finish for the 2018 and 2019 seasons combined (I may add 2017 and beyond if I feel the sample size should increase). If you believe high school recruiting matters a lot, then the most talented teams will likely finish in relative order of the talent ratings. On the other hand, if you believe high school recruiting does not matter so much you would expect the talent and final results to not match up as well. The obvious answer is it will be somewhere in the middle, which is where my study comes into play. I will use several techniques, including linear regression, random forest modeling, machine learning among others to evaluate and provide specific results on the correlation of high school recruiting and on-field success in college football.

Load Data

```r
library(readxl)
X2017_19_Team_Talent <- read_excel("C:/Users/Jack Tesar/Desktop/Personal Stat
s Projects/2017-19 Team Talent.xlsx")

# view for reference
# View(X2017_19_Team_Talent)

data_orig <- X2017_19_Team_Talent
head(data_orig)
```

```
## # A tibble: 6 x 9
##    year school conference talent_rank roster_size five_stars four_stars
##   <dbl> <chr>  <chr>            <dbl>       <dbl>      <dbl>      <dbl>
## 1  2019 Alaba~ SEC                  1          85         11         58
## 2  2019 Ohio ~ B10                  2          85         13         47
## 3  2019 Georg~ SEC                  3          85         14         45
## 4  2019 USC    P12                  4          83          6         41
## 5  2019 LSU    SEC                  5          85          7         44
## 6  2019 Flori~ ACC                  6          83          5         36
## # ... with 2 more variables: three_stars <dbl>, sp_finish <dbl>
```

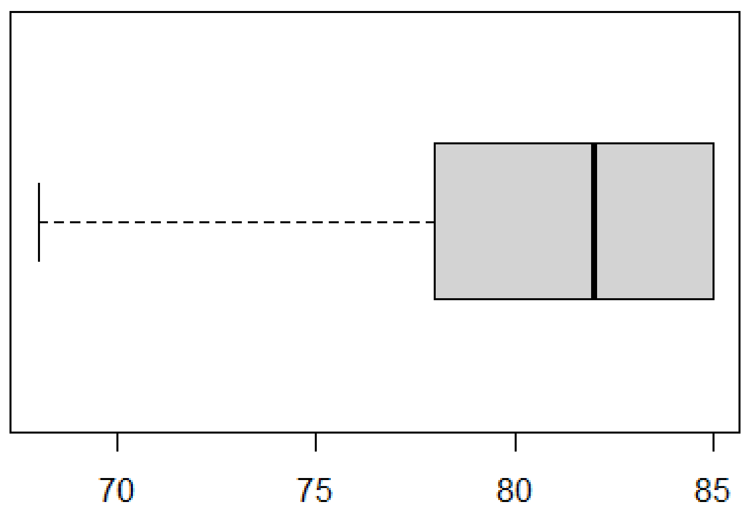
```r
# copy data set to keep original unaltered
data1 <- data_orig
#nrow(data1)
#head(data1)
```

The data is first ranked by year in descending order, which includes the years 2017, 2018 and 2019. And next the variable 'talent_rank' is ordered in ascending order with 75 teams within each year, so we would expect a total amount of 225 rows.

Since I compiled the data myself in Excel prior to loading into R I am familiar with the variables and some potential outliers that may be present. The first potential outlier I will explore is the 'sp_finish', since the final S&P Rankings include over 200 teams and there is the possibility there are teams that finish far below expected and may be considered extreme outliers. Although there could be a long list of reasons as to why a team finshes far below expected (rash of injuries, coaching changes, etc..), these are not reflected in the variables and can be removed if meeting the threshold of an extreme outlier. Another variable I will investigate is 'roster_size' since roster sizes are not the same for all teams and may have an impact when we manipulate the data with prortions, percentages etc..

Explore Outliers

```
## explore outliers of 'roster_size'

# first use boxplot as a visual if there are obvious outliers
boxplot(data1$roster_size,horizontal=TRUE,axes=TRUE,outline=FALSE)
```



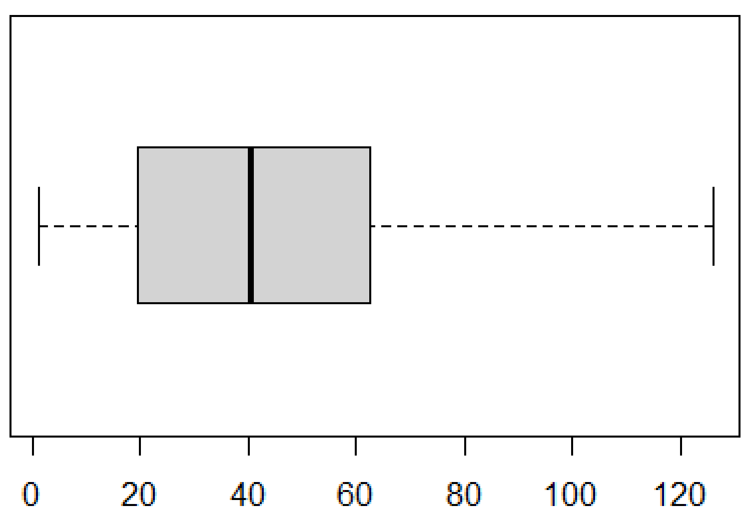

```
# use summary to provide specific values for boxplot
summary(data1$roster_size)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   65.00   78.00   82.00   80.82   85.00   85.00

nrow(data1)

## [1] 225

# calculate IQR_roster (Q3-Q1)
iqr_roster <- (85-78)

# test if appropriate to remove outlier below Q1
78 - (iqr_roster)*1.5

## [1] 67.5
```

Based on the visual of the boxplot there seems to be at least one exteme outlier on the 'lower' end. Based on our IQR calculations we can proceed with removing any values with a roster size less than 67.5.

```
data1 <- data1[which(data1$roster_size > 67.5),]

# confirm outlier has been removed
summary(data1$roster_size)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   68.00   78.00   82.00   80.89   85.00   85.00

nrow(data1)

## [1] 224
```

We successfully removed 1 row of data. We will replicate this process for the variable 'sp_finish'.

```
## explore outliers of 'sp_finish'

# first use boxplot as a visual if there are obvious outliers
boxplot(data1$sp_finish,horizontal=TRUE,axes=TRUE,outline=FALSE)
```

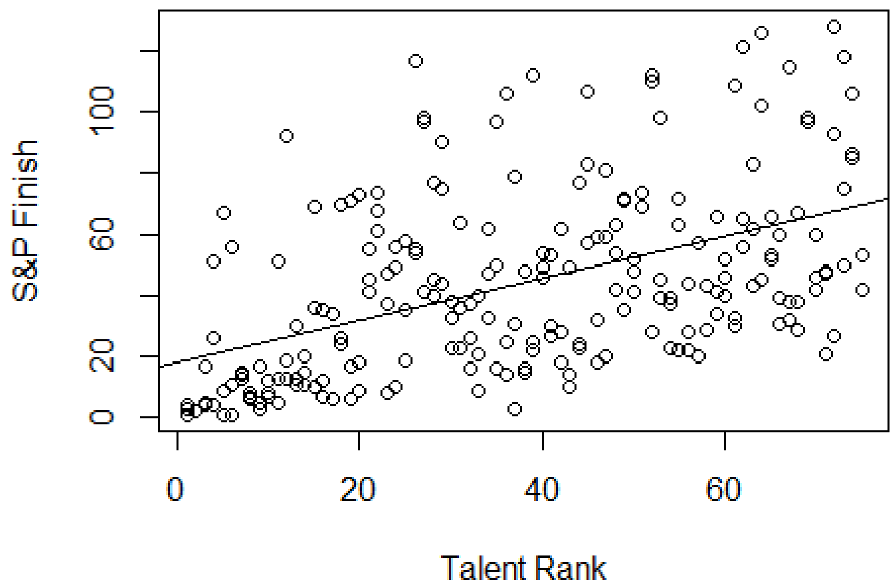



```
# use summary to provide specific values for boxplot
summary(data1$sp_finish)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00   19.75   40.50   45.15   62.25  157.00

# calculate IQR_roster (Q3-Q1)*1.5
iqr_sp <- (63.25-19.75)
63.25 + (iqr_sp)*1.5

## [1] 128.5

data1 <- data1[which(data1$sp_finish < 128.5),]

# confirm outlier has been removed
summary(data1$sp_finish)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00   19.00   40.00   43.85   62.00  128.00

nrow(data1)

## [1] 221
```

We have removed 2 more rows based on our outlier testing. We can conduct alternate outlier testing by creating a dummy linear model and test if there are extreme outliers or leverage points remaining.

```
# create dummy linear model of 'talent_rank' vs 'sp_finish'
lm1 <- lm(sp_finish ~ talent_rank, data = data1)
dim(data1)

## [1] 221   9

# create plot for visual
plot(data1$talent_rank, data1$sp_finish, xlab = "Talent Rank", ylab = "S&P Fi
nish")
abline(lm1)
```



There are no obvious extreme outliers. We can alternatively produce a plot using standardized residuals.

```
plot(data1$talent_rank, rstandard(lm1), xlab = "Talent Rank", ylab = "Standar
dized Residuals of S&P Finish")
abline(h=c(-2,2), lty=2)
```

As a general rule, in "smaller" data sets, points outside of +/- 2 standardized residuals may warrant investigation. There are ~ 10 points outside that qualify as outside +2, but I will make the determination that because the points are relatively spread across the X-Axis (Talent Rank) it is reasonable to include these points. We will proceed with the assumption no further action is needed at this time.

Add Columns on Key Items of Interest

Currently the data includes raw numbers and categorical entries. Manipulating the data to include proportions may prove to yield significant variables in our analysis. I Will include the percentage of the roster that are 5 star prospects, as well as at least 4 star (includes 4 and 5 star prospects) to provide two new variables and provide as a percentage of the roster.

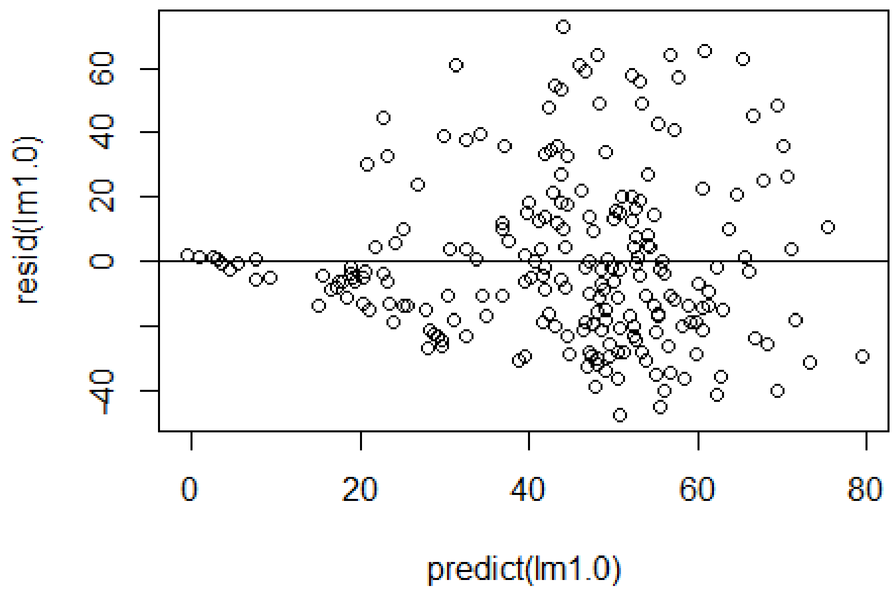
```
# Load tidyverse package
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflict
s() --
```

```
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

# copy data
data2 <- data1

# add new column for percent of five star recruit on roster
data2 <- data2 %>%
  mutate(school, five_star_perc = (100*(five_stars / roster_size)))

# add new column for percent of roster that is at least 4 star (is 4 star or
5 star recruit)
data2 <- data2 %>%
  mutate(school, four_star_min_perc = (100*(four_stars + five_stars) / roster
_size))

# visual confirmation new columns have been included
head(data2)

## # A tibble: 6 x 11
##    year school conference talent_rank roster_size five_stars four_stars
##   <dbl> <chr>  <chr>            <dbl>       <dbl>      <dbl>      <dbl>
## 1  2019 Alaba~ SEC                  1          85         11         58
## 2  2019 Ohio ~ B10                  2          85         13         47
## 3  2019 Georg~ SEC                  3          85         14         45
## 4  2019 USC    P12                  4          83          6         41
## 5  2019 LSU    SEC                  5          85          7         44
## 6  2019 Flori~ ACC                  6          83          5         36
## # ... with 4 more variables: three_stars <dbl>, sp_finish <dbl>,
## #   five_star_perc <dbl>, four_star_min_perc <dbl>
```

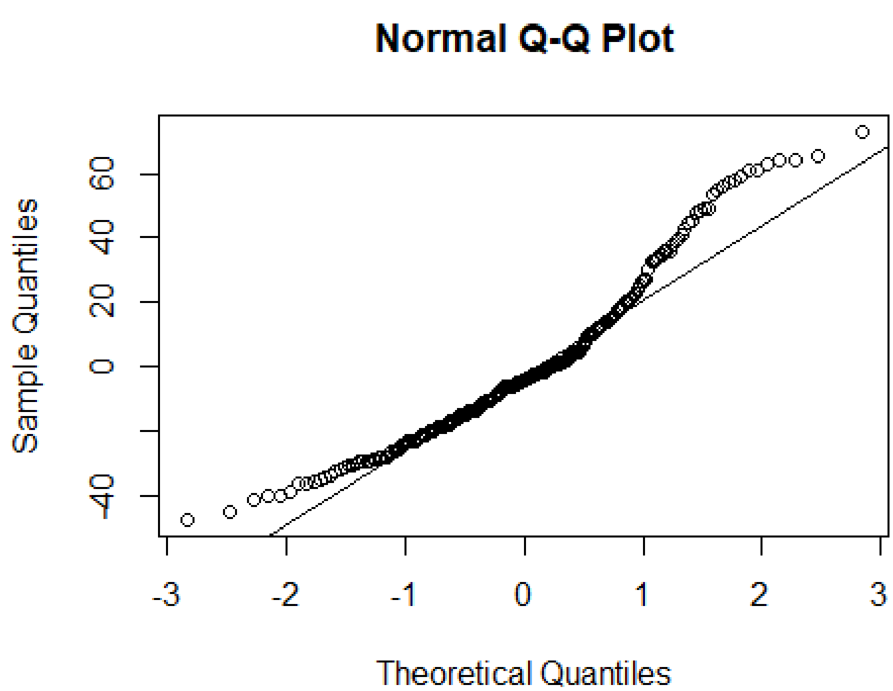
Linear regression analysis

Before we can proceed with our testing we must evaluate that a list of assumptions have been met. Critical assumptions include constant variance and a normal distribution (or normality) of the data set. We can evaluate our assumption of constant variance through a residual plot and our assumption of normality through a QQ-plot. We will begin by creating the linear model with all predictors included and evaluating the assumptions listed.

```
# create linear regression model to test various predictors on the variable '
sp_finish'
lm1.0 <- lm(sp_finish ~ talent_rank + five_stars + four_stars + three_stars +
four_star_min_perc + five_star_perc, data = data2)

# residual plot to assess constant variance
plot(predict(lm1.0), resid(lm1.0))
abline(h=0)
```



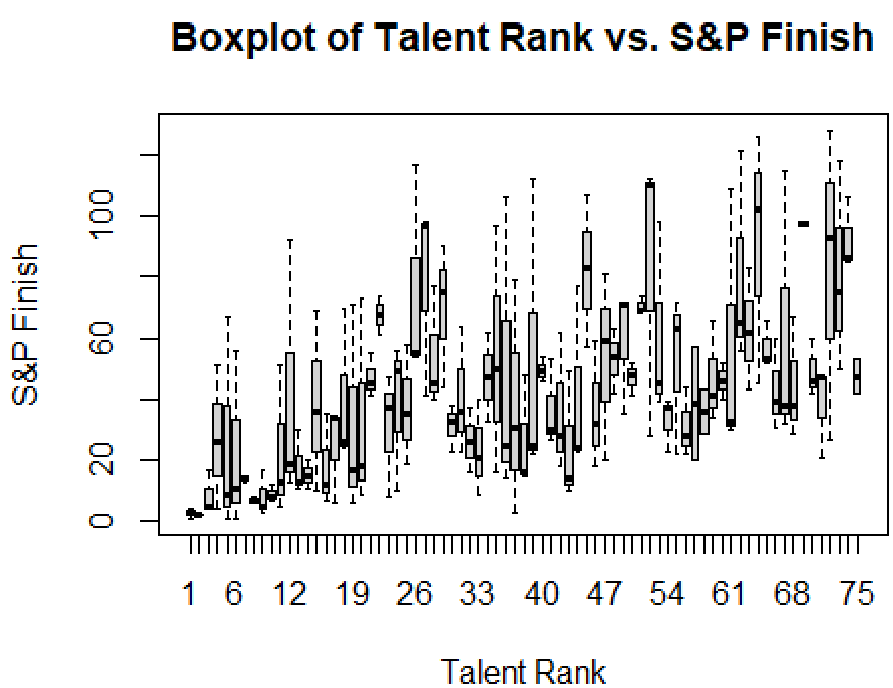

```r
# QQ-plot to assess normality
qqnorm(resid(lm1.0))
qqline(resid(lm1.0))
```



As it currently stands our assumptions for normality have not been met. We can see in the QQ-plot the variance is skewed to the right and there is notabe straying from the QQ-line, indicating the distribution is not normal.

At this stage it would be worthwhile to explore whether to proceed with the study. In other words, is there enough correlation to support further investigating relationships within our explanatory and response variables even with skewed data. We will use a visual reference with a boxplot and analysis with the F-test to test whether there are relevant variables included.

```
# boxplot for reference
boxplot(sp_finish ~ talent_rank, xlab = "Talent Rank", ylab = "S&P Finish", m
ain = "Boxplot of Talent Rank vs. S&P Finish", data = data2)
```

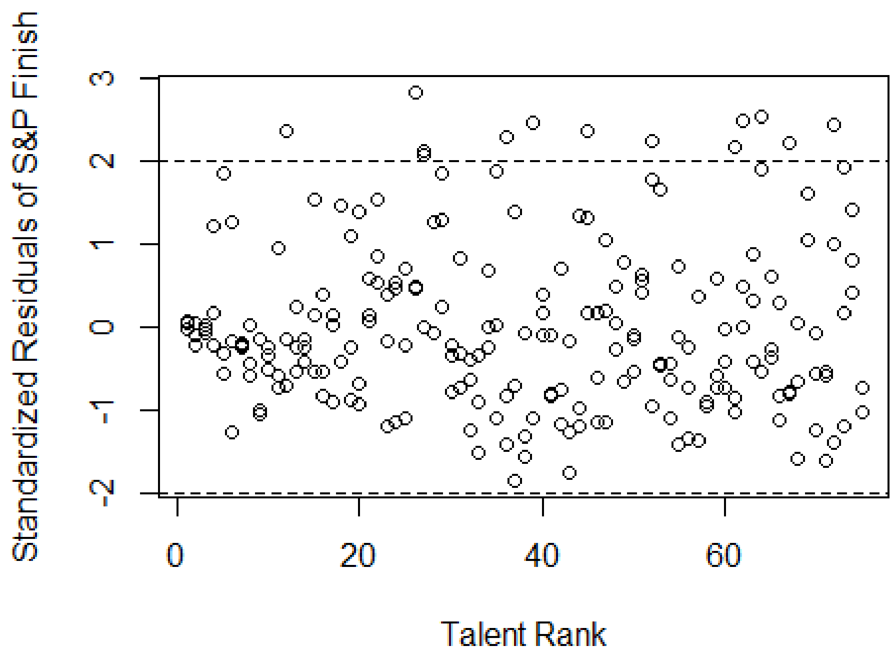



```
# F-test to see if any predictors are "useful"
lm_null <- lm(sp_finish ~ 1, data = data2)
anova(lm_null, lm1.0)

## Analysis of Variance Table
##
## Model 1: sp_finish ~ 1
## Model 2: sp_finish ~ talent_rank + five_stars + four_stars + three_stars +
##     four_star_min_perc + five_star_perc
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    220 202344
## 2    214 145058  6     57286 14.085 1.688e-13 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the boxplot we can see there does appear to be a relationship between the explanatory and response variables, most notably at the "lower" values (i.e. the higher ranking is more associated with a higher finish), while there is more variance and "noise" at the higher end of the data (or lower talent rating). In addition, our F-Test has shown there is at least one significant explanatory variable included in the model. For these reasons we will proceed with the assumptions that the data is "normal enough" and the variance is "constant enough" for our testing.

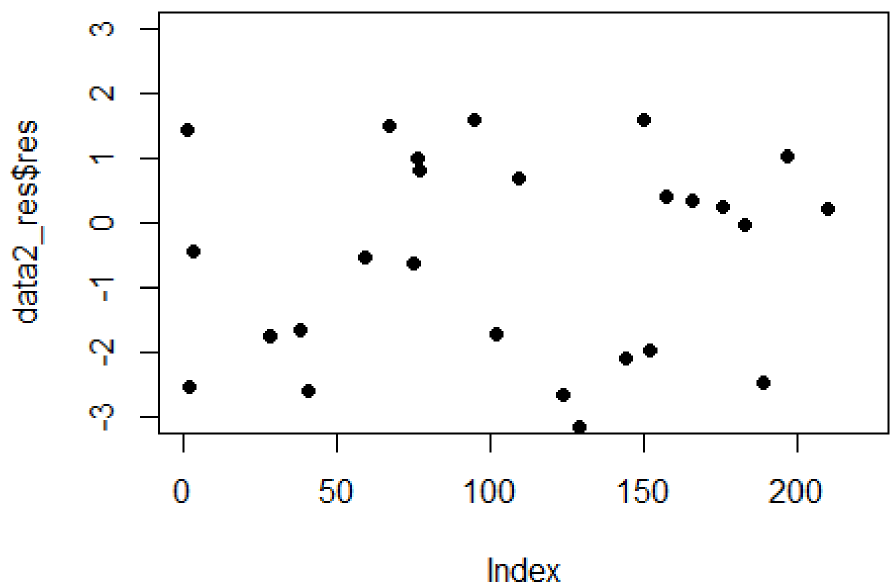
```r
plot(data2$talent_rank, rstandard(lm1.0), xlab = "Talent Rank", ylab = "Stand
ardized Residuals of S&P Finish")
abline(h=c(-2,2), lty=2)
```



I included a +/- of 2 on the residual plot as it is best-practice to remove values that exceed this amount. We can see from the plot there are several points above the +2 residuals. Since our sample size is sufficiently large we may proceed with removing these points.

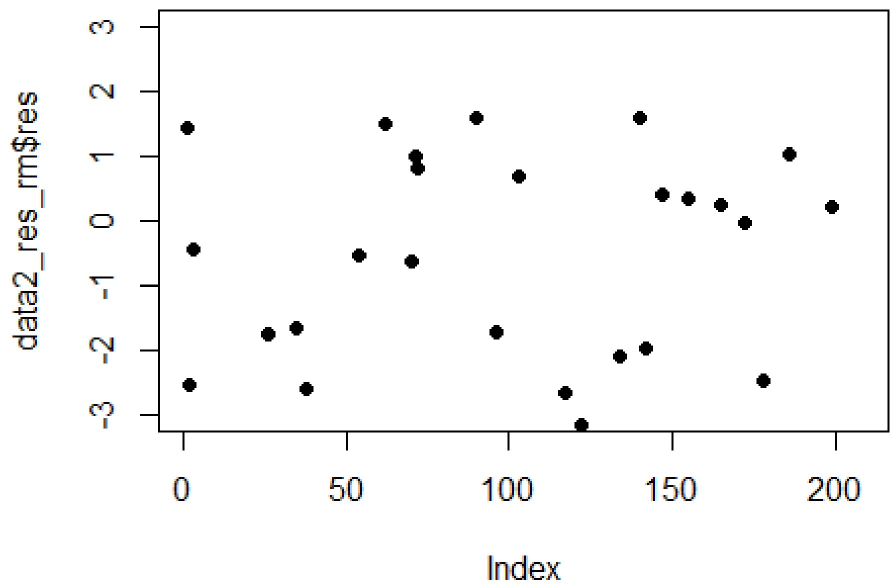
```r
# copy data set for residual work
data2_res <- data2

# store residuals as a new column
data2_res$res <- resid(lm1.0)

# calculate 2 standard deviations (i.e. 2 residuals)
```

```r
sd2 <- 2*sd(resid(lm1.0))
sd2

## [1] 51.35584

# 2 standard deviations is ~50.9

# identify values that are above 2 residuals by adding new column and identif
ying as 1 if above, 0 otherwise
data2_res$outside <- ifelse(abs(data2_res$res)>sd2, 1, 0)

# note number of rows for reference
nrow(data2_res)

## [1] 221

# plot
plot(data2_res$res, col = data2_res$outside + 1, pch=16,  ylim = c(-3, 3))
```

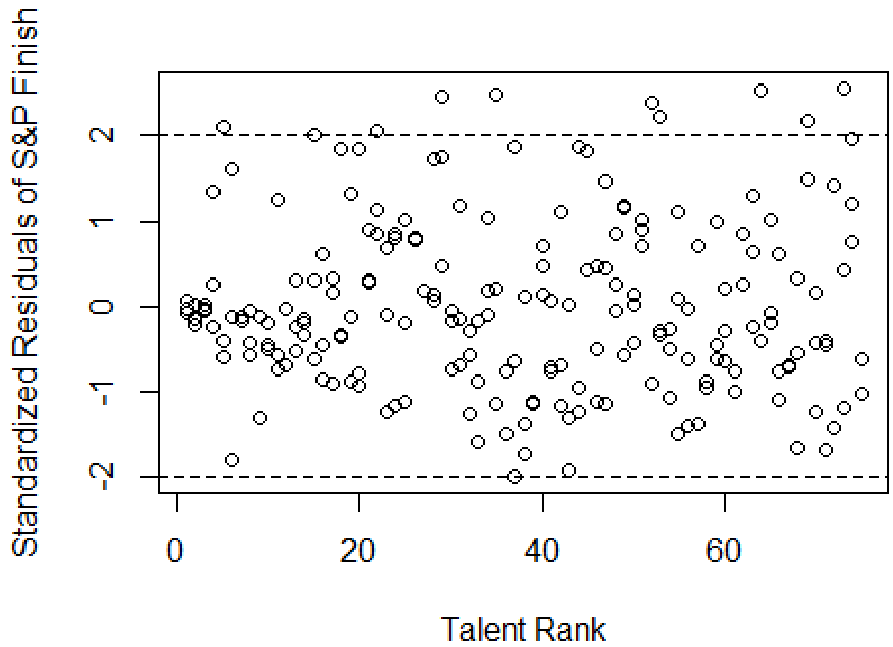



```r
# copy dataset and include only values with standard residual < 2
data2_res_rm <- data2_res[!data2_res$outside, ]

# confirm rows have been removed
nrow(data2_res_rm)

## [1] 208
```

```r
# RESULT: 9 rows removed

# plot
plot(data2_res_rm$res, col = data2_res_rm$outside + 1, pch=16,  ylim = c(-3,
3))
```




```r
# copy lm1.0 using data set with residuals > 2 removed
lm1.0 <- lm(sp_finish ~ talent_rank + five_stars + four_stars + three_stars +
four_star_min_perc + five_star_perc, data = data2_res_rm)

# plot for reference
plot(data2_res_rm$talent_rank, rstandard(lm1.0), xlab = "Talent Rank", ylab =
"Standardized Residuals of S&P Finish")
abline(h=c(-2,2), lty=2)
```




```
# create linear regression model to test various predictors on the variable '
sp_finish'
summary(lm1.0)

##
## Call:
## lm(formula = sp_finish ~ talent_rank + five_stars + four_stars +
##     three_stars + four_star_min_perc + five_star_perc, data = data2_res_rm
)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.547 -14.980  -2.806  13.669  53.980
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        125.8948    31.2268   4.032 7.86e-05 ***
## talent_rank         -0.1486     0.2252  -0.660  0.51001
## five_stars         -15.6789    19.3113  -0.812  0.41781
## four_stars           0.8286     2.6586   0.312  0.75562
## three_stars         -0.9351     0.3067  -3.049  0.00261 **
## four_star_min_perc  -2.0351     2.2647  -0.899  0.36992
## five_star_perc      13.8067    17.6116   0.784  0.43399
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.51 on 201 degrees of freedom
```

```
## Multiple R-squared:  0.3328, Adjusted R-squared:  0.3129
## F-statistic: 16.71 on 6 and 201 DF,  p-value: 1.307e-15
```

The P-values are very high and statistically insignificant, but there is a good chance many of the variables are highly correlated. We can correct this using a Variance Inflation Factor which will provide figures for high correlation between variables. As a rule of thumb a VIF over 5 is concerning and it would be best practice to remove variables from highest VIF to lowest in order (taking into account the respective P-values).

```
# load faraway package
library(faraway)

# VIF on model "lm1" (round to 2 decimals)
round(vif(lm1.0), 2)
```

```
##        talent_rank          five_stars          four_stars         three_star
s
##              10.64             1508.79              768.91                9.0
9
## four_star_min_perc     five_star_perc
##            1058.46            1799.47
```

We will proceed with removing variables that are statistically significant and remove one by one. Since the variable 'five_stars' is highlight insignificant and carries the highest VIF I Will remove this variable first. I will continue this process until only significant variables remain and the VIF values are acceptable.

```
# duplicate linear model "lm1" but remove variable 'five_stars'
lm1.1 <- lm(sp_finish ~ talent_rank + four_stars + three_stars + four_star_mi
n_perc + five_star_perc, data = data2)
summary(lm1.1)
```

```
## 
## Call:
## lm(formula = sp_finish ~ talent_rank + four_stars + three_stars + 
##     four_star_min_perc + five_star_perc, data = data2)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -47.681 -18.412  -4.828  12.935  73.221 
## 
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)    
## (Intercept)        131.4018    36.9673   3.555 0.000465 ***
## talent_rank         -0.1571     0.2624  -0.599 0.550048    
## four_stars           0.2368     2.2322   0.106 0.915599    
## three_stars         -0.9283     0.3655  -2.540 0.011802 *  
## four_star_min_perc  -1.6073     1.9898  -0.808 0.420122    
## five_star_perc       0.0959     1.9231   0.050 0.960273    
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26 on 215 degrees of freedom
## Multiple R-squared:  0.2819, Adjusted R-squared:  0.2652
## F-statistic: 16.88 on 5 and 215 DF,  p-value: 4.568e-14
```

The remaining variables are still insignificant but trending towards potential significance, an encouraging sign there may be significant variables included after removing the correlated and/or insignificant variables.

```
round(vif(lm1.1), 2)
```

```
##        talent_rank         four_stars        three_stars four_star_min_per
c
##              10.42             381.03               9.01             573.3
5
##     five_star_perc
##              14.88
```

As seen above the most insignificant variable is talent_rank, but since there are other VIF values much higher, I will not remove this variable (yet). Instead I will remove 'four_star_min_prop' since it is insignificant and carries a very high VIF.

```
# remove 'four_stars'
lm1.2 <- lm(sp_finish ~ talent_rank + four_stars + three_stars + five_star_pe
rc, data = data2)
summary(lm1.2)
```

```
##
## Call:
## lm(formula = sp_finish ~ talent_rank + four_stars + three_stars +
##     five_star_perc, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.872 -18.740  -4.967  13.762  74.882
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    117.61830   32.76696   3.590  0.00041 ***
## talent_rank     -0.08275    0.24555  -0.337  0.73645
## four_stars      -1.51904    0.50686  -2.997  0.00305 **
## three_stars     -0.79386    0.32516  -2.441  0.01543 *
## five_star_perc  -1.31757    0.79697  -1.653  0.09974 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.98 on 216 degrees of freedom
## Multiple R-squared:  0.2797, Adjusted R-squared:  0.2664
## F-statistic: 20.97 on 4 and 216 DF,  p-value: 1.267e-14
```

```r
round(vif(lm1.2), 2)

##    talent_rank     four_stars    three_stars five_star_perc 
##           9.14          19.68           7.14           2.56
```

Here we can see the VIF values have become much closer to acceptable, which is an encouraging sign. Now we can follow the process of removing the most insignificant variables based on the p-value knowing our risk of removing correlated variables is much lower. Now we will remove 'talent_rank' since it is the most statistically insignificant variable and there are no obvious violations in VIF.

```r
# remove 'three_stars'
lm1.3 <- lm(sp_finish ~ four_stars + three_stars + five_star_perc, data = dat
a2)
summary(lm1.3)

## 
## Call:
## lm(formula = sp_finish ~ four_stars + three_stars + five_star_perc, 
##     data = data2)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -46.202 -18.944  -4.727  13.511  75.521 
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    108.2873    17.4846   6.193 2.92e-09 ***
## four_stars      -1.3670     0.2303  -5.936 1.15e-08 ***
## three_stars     -0.7269     0.2569  -2.830   0.0051 ** 
## five_star_perc  -1.2716     0.7836  -1.623   0.1061    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 25.92 on 217 degrees of freedom
## Multiple R-squared:  0.2794, Adjusted R-squared:  0.2694 
## F-statistic: 28.04 on 3 and 217 DF,  p-value: 2.3e-15

round(vif(lm1.3), 2)

##     four_stars    three_stars five_star_perc 
##           4.08           4.48           2.48
```

Alas our VIF values are all under 5, where we can confidently say there is very little correlation between the variables. We will continue to remove based on level of significance where we will remove 'five_star_perc.'

```r
# remove 'five_star_prop'
lm1.4 <- lm(sp_finish ~ four_stars + three_stars, data = data2)
summary(lm1.4)
```

```
##
## Call:
## lm(formula = sp_finish ~ four_stars + three_stars, data = data2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -47.07 -18.57  -4.50  14.67  76.38
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  98.8205    16.5443   5.973 9.37e-09 ***
## four_stars   -1.4544     0.2247  -6.472 6.30e-10 ***
## three_stars  -0.5718     0.2393  -2.389   0.0177 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.02 on 218 degrees of freedom
## Multiple R-squared:  0.2706, Adjusted R-squared:  0.2639
## F-statistic: 40.44 on 2 and 218 DF,  p-value: 1.153e-15
```

Our level of significance for 'three_star' would be acceptable with a 10% Type 1 error level. But since 'four_star' is highly significant (less than 1%) I will remove the 'three_star' variable.

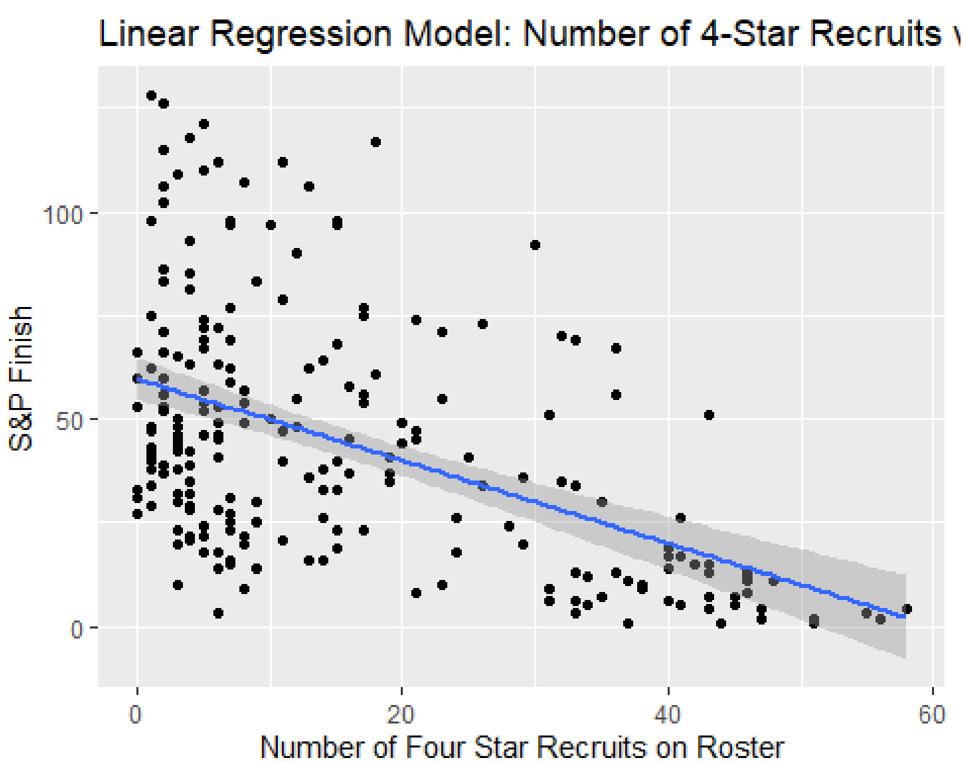
```r
# remove 'three_stars'
lm1.5 <- lm(sp_finish ~ four_stars, data = data2)
summary(lm1.5)
```

```
##
## Call:
## lm(formula = sp_finish ~ four_stars, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.806 -18.775  -4.806  12.202  75.103
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  59.7599     2.5629  23.317  < 2e-16 ***
## four_stars   -0.9924     0.1157  -8.579 1.77e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.3 on 219 degrees of freedom
## Multiple R-squared:  0.2515, Adjusted R-squared:  0.2481
## F-statistic: 73.59 on 1 and 219 DF,  p-value: 1.769e-15
```

After removing the insignificant variables in the model we are left with one highly significant variable of 'four_stars', which is the number of high school recruits rated as a four star on the roster. To interpret the model, an increase in 1 four star recruit on the roster is associated with an DECREASE in final ranking of -1.12. Please note final ranking is listed in descending order with 1 as the "highest" ranked team so an increase in four star players is associated with a higher final ranking.

I am a little surprised this was the remaining variable, and my suspician is because there were highly correlated variables there is a strong possibility other variables would be highly significant if left in the model as the lone variable. I will test this by creating an alternate linear regression model with a different remaining variable that is likely correlated with 'four star.'

```
## ggplot

# load ggplot library
library(ggplot2)

# ggplot of lm1.5
ggplot(data = data2, aes(four_stars, sp_finish)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(y = "S&P Finish", x = "Number of Four Star Recruits on Roster") +
  ggtitle("Linear Regression Model: Number of 4-Star Recruits vs. S&P Finish"
)

## `geom_smooth()` using formula 'y ~ x'
```




```
# alternative single linear regression with 'four_star_min_perc' as lone vari
able
```

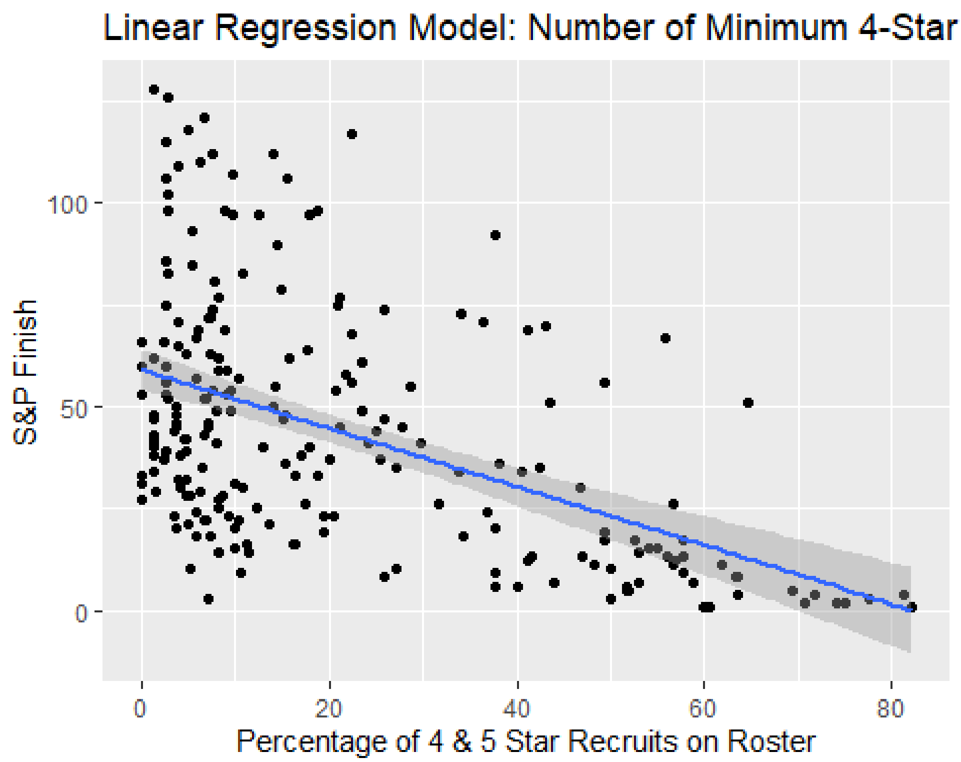
```r
lm1.5_alt_1 <- lm(sp_finish ~ four_star_min_perc, data = data2)
summary(lm1.5_alt_1)
```

```
##
## Call:
## lm(formula = sp_finish ~ four_star_min_perc, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.111 -18.236  -4.665  12.041  73.907
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        59.19681    2.51783  23.511  < 2e-16 ***
## four_star_min_perc -0.72045    0.08409  -8.568  1.9e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.31 on 219 degrees of freedom
## Multiple R-squared:  0.251,  Adjusted R-squared:  0.2476
## F-statistic: 73.41 on 1 and 219 DF,  p-value: 1.9e-15
```

Interpretation: an increase in 1 percentage of the roster that is a 4 or 5 star recruit is associated with a DECREASE in final ranking of -0.77. In other words, increasing the percentage of 4 & 5 star players is assocaited with improved S&P final results.

```r
# ggplot of lm1.5_alt_1 using 'four_star_min_perc' as alternate lone variable
ggplot(data = data2, aes(four_star_min_perc, sp_finish)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(y = "S&P Finish", x = "Percentage of 4 & 5 Star Recruits on Roster") +
  ggtitle("Linear Regression Model: Number of Minimum 4-Star Recruits vs. S&P
Finish")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

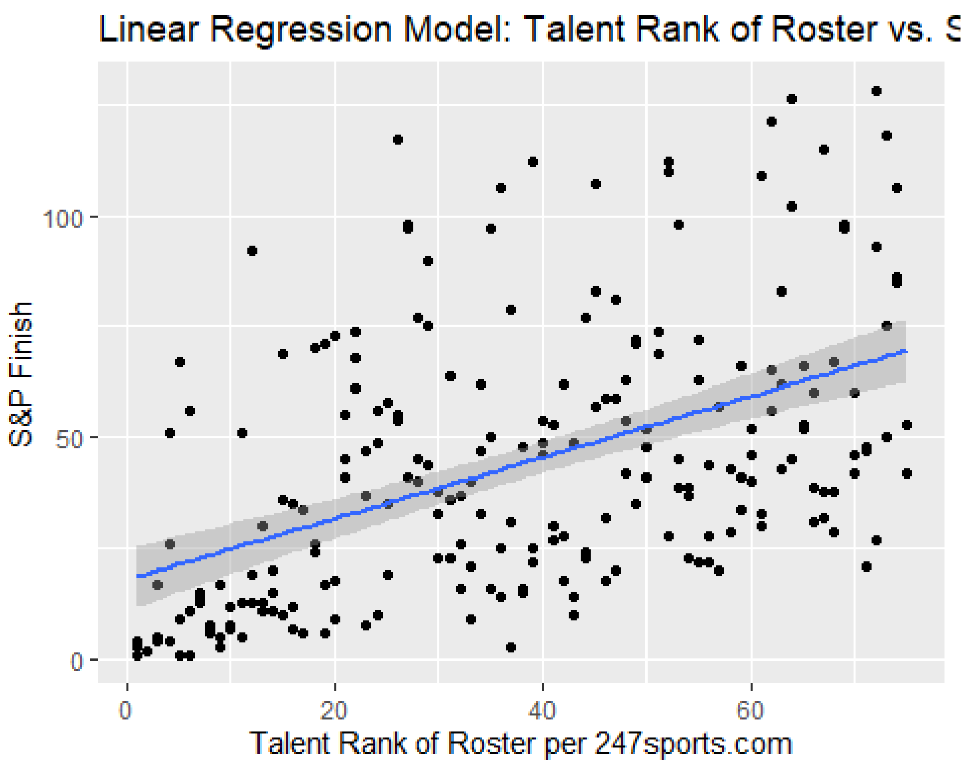



```
# alternative single linear regression with 'talent_rank' as lone variable
lm1.5_alt_2 <- lm(sp_finish ~ talent_rank, data = data2)
summary(lm1.5_alt_2)
```

```
##
## Call:
## lm(formula = sp_finish ~ talent_rank, data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.88 -18.37  -7.68  15.88  81.07
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.0527     3.5851   5.035 9.95e-07 ***
## talent_rank   0.6876     0.0829   8.295 1.10e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.51 on 219 degrees of freedom
## Multiple R-squared:  0.2391, Adjusted R-squared:  0.2356
## F-statistic: 68.81 on 1 and 219 DF,  p-value: 1.102e-14
```

Interpretation: an increase in 1 talent rank is associated with an INCREASE in S&P final ranking of 0.75. Alternatively we could say a decrease in 1 talent rank (the ranking is improved by 1) is associated with a DECREASE in S&P final ranking of 0.75, meaning the final ranking was improved.

```r
# ggplot of lm1.5_alt_2 using 'talent_rank' as alternate lone variable
ggplot(data = data2, aes(talent_rank, sp_finish)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(y = "S&P Finish", x = "Talent Rank of Roster per 247sports.com") +
  ggtitle("Linear Regression Model: Talent Rank of Roster vs. S&P Finish")

## `geom_smooth()` using formula 'y ~ x'
```

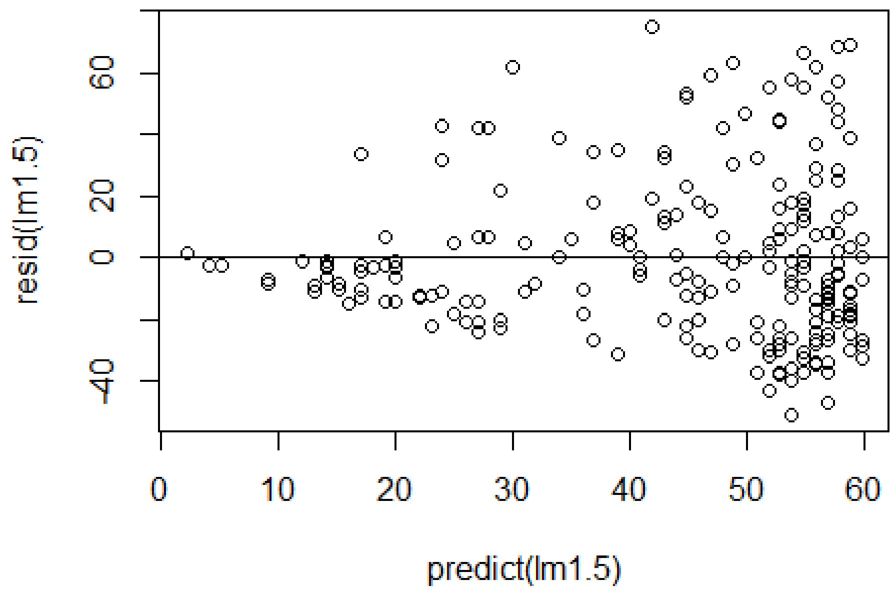


As shown in alternate linear models (lm1.5_alt_1 & lm1.5_alt_2) we could have used alternate lone variables to use for our linear regression model with alternate interpretations. The bottom line remains that there are strong correlations present to the level of recruiting and final S&P rankings.

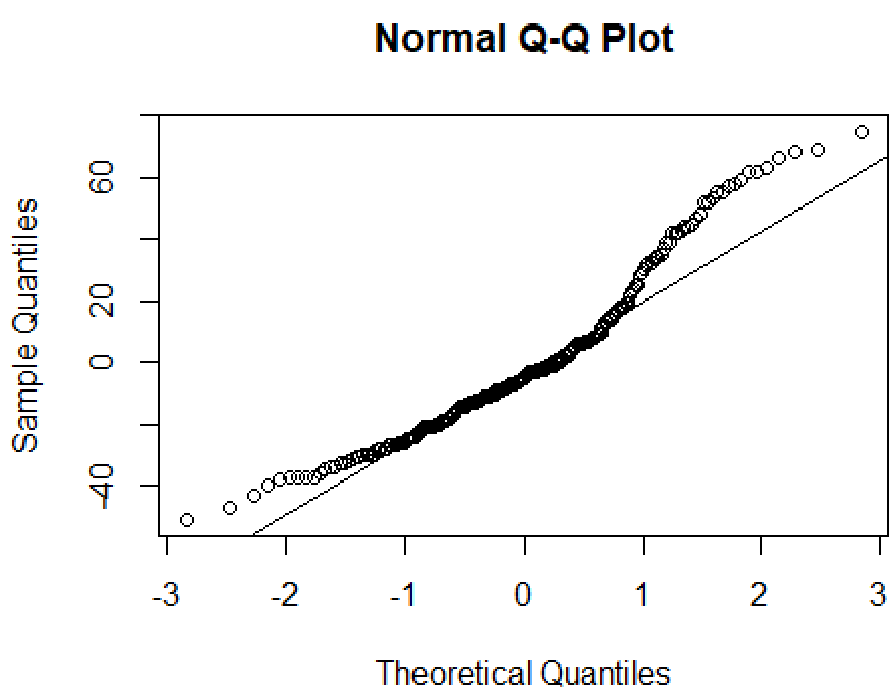
Further validation of constant variance & normality of the data

After we have produced a linear regression model we can reassess the validity of the model. We will do so by re-checking our assumptions of constant variance and normality.

```r
# residual plot to assess constant variance
plot(predict(lm1.5), resid(lm1.5))
abline(h=0)
```




```r
# QQ-plot to assess normality
qqnorm(resid(lm1.5))
qqline(resid(lm1.5))
```



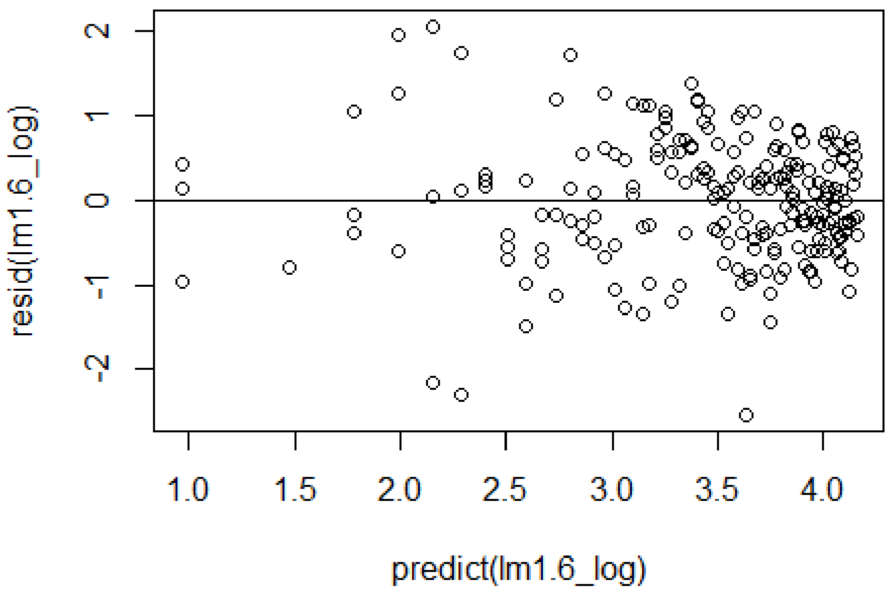
Based on the predict v residual plot there are certainly concerns about constant variance for our linear model produced. We may consider transformations to improve our model to meet assumptions of constant variance.

```
# log transformation
lm1.6_log <- lm(log(sp_finish) ~ log(talent_rank), data = data2)
summary(lm1.6_log)
```

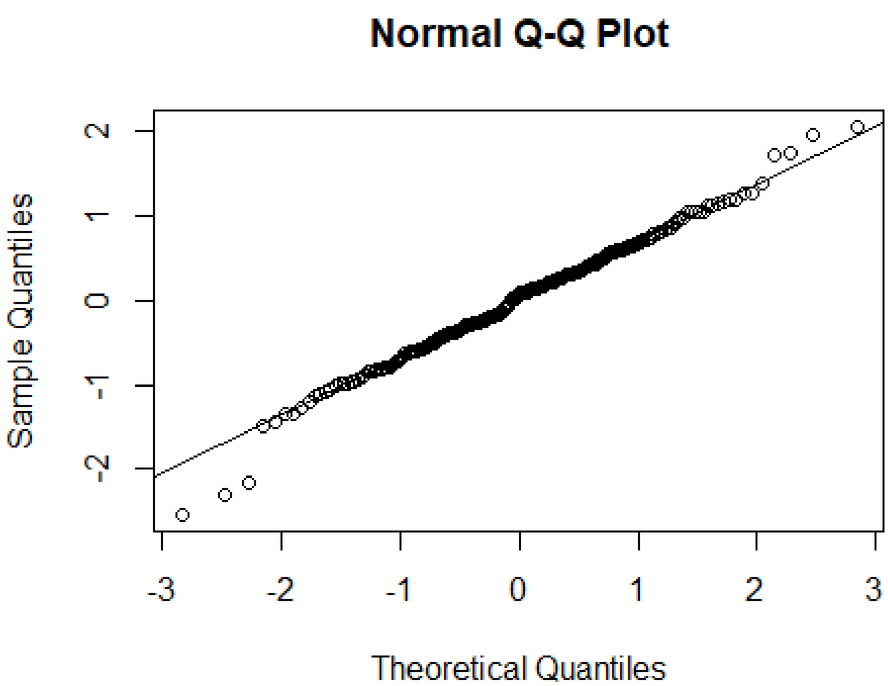
```
##
## Call:
## lm(formula = log(sp_finish) ~ log(talent_rank), data = data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.53519 -0.45225  0.07671  0.46725  2.04991
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.96547    0.18515   5.215 4.26e-07 ***
## log(talent_rank)  0.73896    0.05343  13.831  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.722 on 219 degrees of freedom
## Multiple R-squared:  0.4662, Adjusted R-squared:  0.4638
## F-statistic: 191.3 on 1 and 219 DF,  p-value: < 2.2e-16
```

Our p-value for the variable 'talent_rank' is acceptable.

```
# residual plot to assess constant variance
plot(predict(lm1.6_log), resid(lm1.6_log))
abline(h=0)
```




```r
# QQ-plot to assess normality
qqnorm(resid(lm1.6_log))
qqline(resid(lm1.6_log))
```

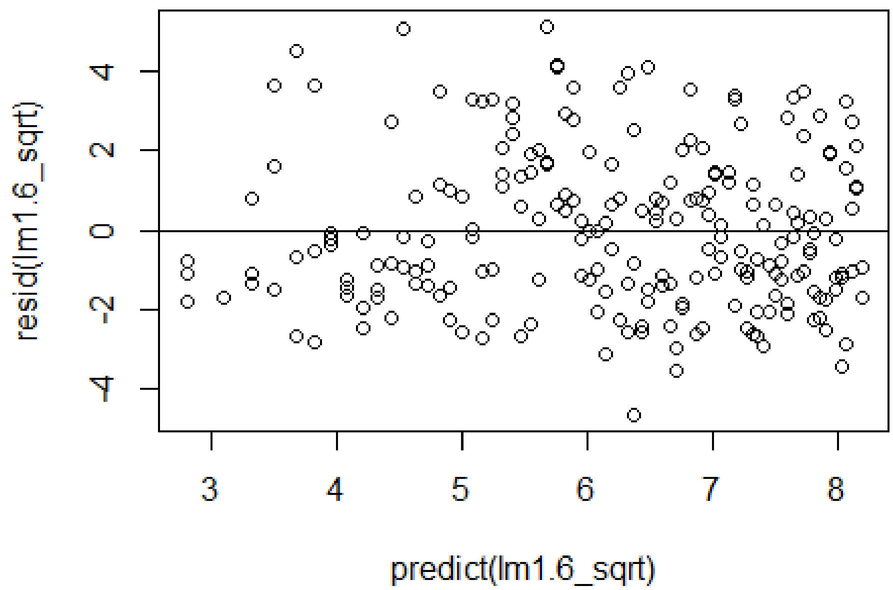


Using a log transformation did not improve our level of variance. Next we will try a square root transformation.

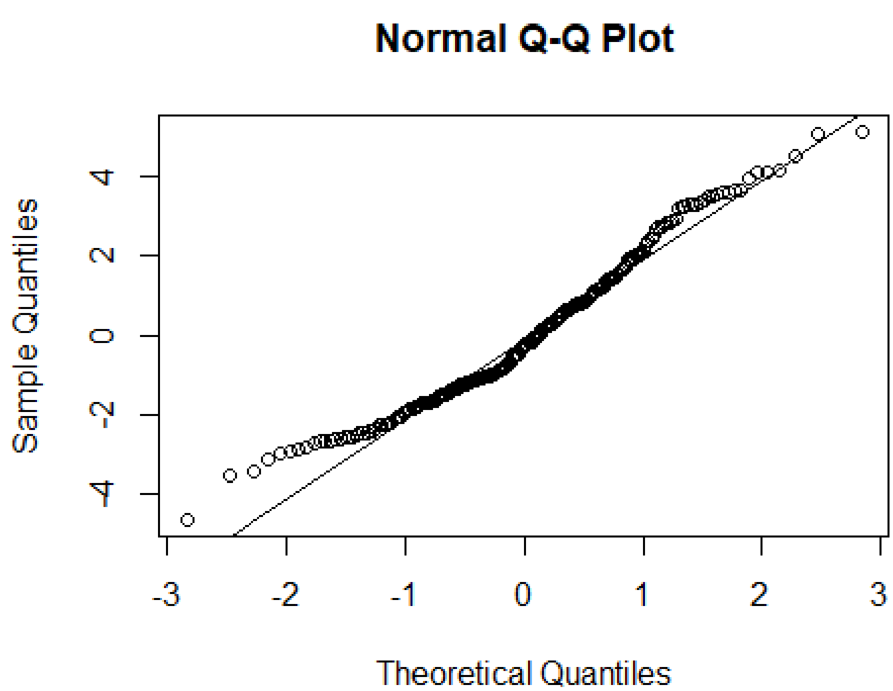
```
# square root transformation
lm1.6_sqrt <- lm(sqrt(sp_finish) ~ sqrt(talent_rank), data = data2)
summary(lm1.6_sqrt)
```

```
##
## Call:
## lm(formula = sqrt(sp_finish) ~ sqrt(talent_rank), data = data2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6400 -1.4893 -0.2326  1.2220  5.1366
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.0933     0.4061   5.155 5.67e-07 ***
## sqrt(talent_rank)   0.7034     0.0663  10.609  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.973 on 219 degrees of freedom
## Multiple R-squared:  0.3395, Adjusted R-squared:  0.3365
## F-statistic: 112.6 on 1 and 219 DF,  p-value: < 2.2e-16
```

```
# residual plot to assess constant variance
plot(predict(lm1.6_sqrt), resid(lm1.6_sqrt))
abline(h=0)
```




```r
# QQ-plot to assess normality
qqnorm(resid(lm1.6_sqrt))
qqline(resid(lm1.6_sqrt))
```



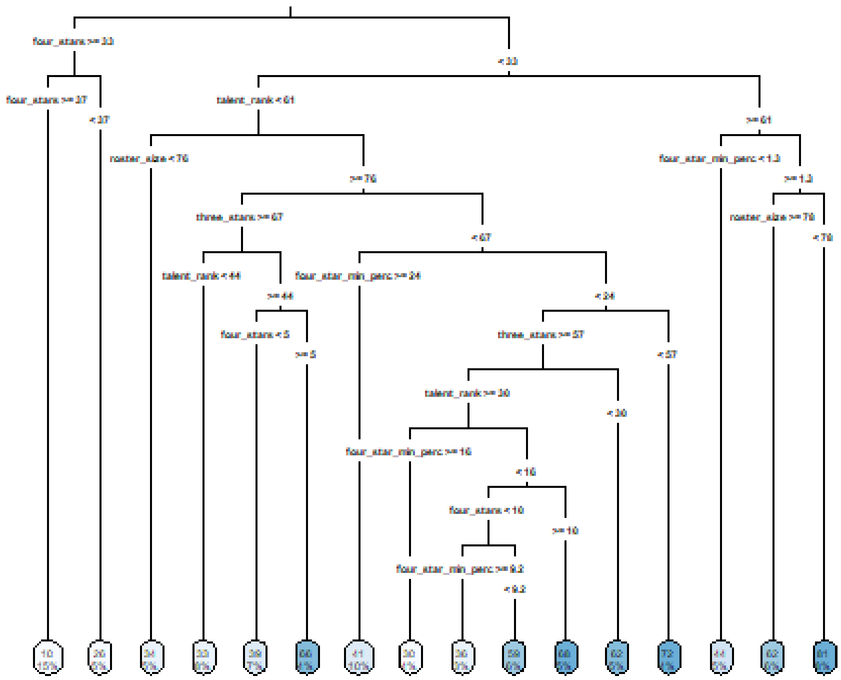
As shown above the amount of variance has improved greatly. This will slightly alter our interpretation of the model in the way that an increase of 1 talent ranking (square root of 1 = 1) is associated with an increase in sqrt(0.75) = 0.86 in S&P finish.

Machine Learning Part 1

Regression Tree Modeling - Continious

The next portion of our analysis will include Regression Tree modeling. We will first gauge the level of variance that can be explained by the model. If the amount is high, it is clear there are notable trends in the data set, while if the variance is lower the data is more random.

```
# for reference
head(data2)

## # A tibble: 6 x 11
##    year school conference talent_rank roster_size five_stars four_stars
##   <dbl> <chr>  <chr>            <dbl>       <dbl>      <dbl>      <dbl>
## 1  2019 Alaba~ SEC                  1          85         11         58
## 2  2019 Ohio ~ B10                  2          85         13         47
## 3  2019 Georg~ SEC                  3          85         14         45
## 4  2019 USC    P12                  4          83          6         41
## 5  2019 LSU    SEC                  5          85          7         44
## 6  2019 Flori~ ACC                  6          83          5         36
## # ... with 4 more variables: three_stars <dbl>, sp_finish <dbl>,
## #   five_star_perc <dbl>, four_star_min_perc <dbl>

# load packages
library(rpart)

##
## Attaching package: 'rpart'

## The following object is masked from 'package:faraway':
##
##     solder

library(rpart.plot)

# copy data set
data2alt <- data2

# create model for decision tree to predict 'sp_finish' from the given variab
les
decision_tree1 <- rpart(sp_finish ~ talent_rank + roster_size + five_stars +
four_stars + three_stars + five_star_perc + four_star_min_perc, data = data2a
lt, method = "anova")
rpart.plot(decision_tree1, type = 3)
```



```r
# summary of decision tree
summary(decision_tree1)

## Call:
## rpart(formula = sp_finish ~ talent_rank + roster_size + five_stars +
##     four_stars + three_stars + five_star_perc + four_star_min_perc,
##     data = data2alt, method = "anova")
##   n= 221
##
##           CP nsplit rel error    xerror       xstd
## 1 0.24768572      0 1.0000000 1.0115904 0.09382574
## 2 0.05474949      1 0.7523143 0.8126425 0.08250490
## 3 0.03380393      2 0.6975648 0.8322237 0.08113653
## 4 0.01406145      3 0.6637609 0.8775363 0.08636253
## 5 0.01389365      4 0.6496994 0.8926949 0.09363033
## 6 0.01308073      9 0.5778010 0.8950587 0.09362843
## 7 0.01111862     12 0.5385588 0.9027732 0.09505156
## 8 0.01083320     13 0.5274402 0.9047287 0.09498594
## 9 0.01000000     15 0.5057738 0.9047287 0.09498594
##
## Variable importance
## four_star_min_perc         four_stars        talent_rank        three_star
s
##                 24                 23                 20                 1
5
##     five_star_perc         five_stars        roster_size
##                  8                  8                  2
```

```
##
## Node number 1: 221 observations,    complexity param=0.2476857
##   mean=43.85068, MSE=915.584
##   left son=2 (46 obs) right son=3 (175 obs)
##   Primary splits:
##       four_stars         < 32.5      to the right, improve=0.2476857, (0 m
issing)
##       four_star_min_perc < 43.78852  to the right, improve=0.2371532, (0 m
issing)
##       talent_rank        < 20.5      to the left,  improve=0.2216185, (0 m
issing)
##       three_stars        < 43.5      to the left,  improve=0.1850096, (0 m
issing)
##       five_star_perc     < 3.098039  to the right, improve=0.1499531, (0 m
issing)
##   Surrogate splits:
##       four_star_min_perc < 39.04762  to the right, agree=0.986, adj=0.935,
(0 split)
##       talent_rank        < 16.5      to the left,  agree=0.964, adj=0.826,
(0 split)
##       three_stars        < 42.5      to the left,  agree=0.959, adj=0.804,
(0 split)
##       five_star_perc     < 3.614983  to the right, agree=0.905, adj=0.543,
(0 split)
##       five_stars         < 2.5       to the right, agree=0.896, adj=0.500,
(0 split)
##
## Node number 2: 46 observations,    complexity param=0.01111862
##   mean=14.47826, MSE=260.0321
##   left son=4 (34 obs) right son=5 (12 obs)
##   Primary splits:
##       four_stars         < 36.5      to the right, improve=0.1880860, (0 m
issing)
##       three_stars        < 33.5      to the left,  improve=0.1336836, (0 m
issing)
##       five_stars         < 8.5       to the right, improve=0.1287471, (0 m
issing)
##       five_star_perc     < 10.98434  to the right, improve=0.1287471, (0 m
issing)
##       four_star_min_perc < 49.70588  to the right, improve=0.1209129, (0 m
issing)
##   Surrogate splits:
##       four_star_min_perc < 47.64706  to the right, agree=0.913, adj=0.667,
(0 split)
##       three_stars        < 41.5      to the left,  agree=0.870, adj=0.500,
(0 split)
##       talent_rank        < 14.5      to the left,  agree=0.804, adj=0.250,
(0 split)
##       roster_size        < 77.5      to the right, agree=0.761, adj=0.083,
(0 split)
```

```
##       five_star_perc     < 1.197648  to the right, agree=0.761, adj=0.083,
(0 split)
##
## Node number 3: 175 observations,    complexity param=0.05474949
##   mean=51.57143, MSE=801.5135
##   left son=6 (132 obs) right son=7 (43 obs)
##   Primary splits:
##       talent_rank        < 60.5      to the left,  improve=0.07898083, (0
missing)
##       four_stars         < 2.5       to the right, improve=0.02878465, (0
missing)
##       roster_size        < 84.5      to the right, improve=0.02760750, (0
missing)
##       four_star_min_perc < 23.79002  to the right, improve=0.02745908, (0
missing)
##       three_stars        < 66.5      to the right, improve=0.01932029, (0
missing)
##   Surrogate splits:
##       four_star_min_perc < 4.558824  to the right, agree=0.886, adj=0.535,
(0 split)
##       four_stars         < 3.5       to the right, agree=0.874, adj=0.488,
(0 split)
##       roster_size        < 74.5      to the right, agree=0.789, adj=0.140,
(0 split)
##
## Node number 4: 34 observations
##   mean=10.32353, MSE=86.39533
##
## Node number 5: 12 observations
##   mean=26.25, MSE=564.5208
##
## Node number 6: 132 observations,    complexity param=0.01389365
##   mean=47.0303, MSE=673.2415
##   left son=12 (12 obs) right son=13 (120 obs)
##   Primary splits:
##       roster_size        < 75.5      to the left,  improve=0.02489805, (0
missing)
##       talent_rank        < 44.5      to the left,  improve=0.02395854, (0
missing)
##       four_star_min_perc < 5.989957  to the left,  improve=0.02115834, (0
missing)
##       three_stars        < 66.5      to the right, improve=0.02055809, (0
missing)
##       four_stars         < 4.5       to the left,  improve=0.01631416, (0
missing)
##   Surrogate splits:
##       talent_rank < 59.5      to the right, agree=0.917, adj=0.083, (0 spl
it)
##
## Node number 7: 43 observations,    complexity param=0.03380393
```

```
##   mean=65.51163, MSE=937.6452
##   left son=14 (11 obs) right son=15 (32 obs)
##   Primary splits:
##       four_star_min_perc < 1.316701  to the left,  improve=0.16964880, (0
missing)
##       roster_size        < 77.5      to the right, improve=0.13217640, (0
missing)
##       four_stars         < 1.5       to the left,  improve=0.07224547, (0
missing)
##       talent_rank        < 71.5      to the left,  improve=0.06141416, (0
missing)
##       three_stars        < 52        to the right, improve=0.04686958, (0
missing)
##   Surrogate splits:
##       four_stars  < 1.5       to the left,  agree=0.907, adj=0.636, (0 spl
it)
##       roster_size < 80.5      to the right, agree=0.791, adj=0.182, (0 spl
it)
##       three_stars < 70.5      to the right, agree=0.791, adj=0.182, (0 spl
it)
##
## Node number 12: 12 observations
##   mean=34.08333, MSE=156.7431
##
## Node number 13: 120 observations,    complexity param=0.01389365
##   mean=48.325, MSE=706.4527
##   left son=26 (41 obs) right son=27 (79 obs)
##   Primary splits:
##       three_stars        < 66.5      to the right, improve=0.03434232, (0
missing)
##       four_star_min_perc < 5.989957  to the left,  improve=0.03010955, (0
missing)
##       talent_rank        < 44.5      to the left,  improve=0.02875566, (0
missing)
##       roster_size        < 84.5      to the right, improve=0.02464707, (0
missing)
##       four_stars         < 4.5       to the left,  improve=0.01856359, (0
missing)
##   Surrogate splits:
##       four_star_min_perc < 7.453704  to the left,  agree=0.808, adj=0.439,
(0 split)
##       four_stars         < 8.5       to the left,  agree=0.792, adj=0.390,
(0 split)
##       talent_rank        < 39.5      to the right, agree=0.742, adj=0.244,
(0 split)
##
## Node number 14: 11 observations
##   mean=44, MSE=118.7273
##
## Node number 15: 32 observations,    complexity param=0.01406145
```

```
##   mean=72.90625, MSE=1005.397
##   left son=30 (14 obs) right son=31 (18 obs)
##   Primary splits:
##       roster_size        < 77.5      to the right, improve=0.08843674, (0
missing)
##       four_star_min_perc < 2.758752  to the right, improve=0.05524881, (0
missing)
##       talent_rank        < 71.5      to the left,  improve=0.05170108, (0
missing)
##       four_stars         < 2.5       to the right, improve=0.04489926, (0
missing)
##       three_stars        < 52        to the right, improve=0.02111709, (0
missing)
##   Surrogate splits:
##       four_star_min_perc < 2.597841  to the left,  agree=0.688, adj=0.286,
(0 split)
##       three_stars        < 66.5      to the right, agree=0.656, adj=0.214,
(0 split)
##       talent_rank        < 69.5      to the right, agree=0.625, adj=0.143,
(0 split)
##
## Node number 26: 41 observations,    complexity param=0.01389365
##   mean=41.4878, MSE=621.1279
##   left son=52 (18 obs) right son=53 (23 obs)
##   Primary splits:
##       talent_rank    < 43.5      to the left,  improve=0.09558784, (0 miss
ing)
##       five_stars     < 0.5       to the right, improve=0.08621773, (0 miss
ing)
##       five_star_perc < 0.5882353 to the right, improve=0.08621773, (0 miss
ing)
##       roster_size    < 84.5      to the right, improve=0.07363561, (0 miss
ing)
##       four_stars     < 7.5       to the left,  improve=0.04520607, (0 miss
ing)
##   Surrogate splits:
##       four_stars         < 5.5       to the right, agree=0.902, adj=0.778,
(0 split)
##       four_star_min_perc < 6.73454   to the right, agree=0.902, adj=0.778,
(0 split)
##       five_stars         < 0.5       to the right, agree=0.659, adj=0.222,
(0 split)
##       three_stars        < 71.5      to the left,  agree=0.659, adj=0.222,
(0 split)
##       five_star_perc     < 0.5882353 to the right, agree=0.659, adj=0.222,
(0 split)
##
## Node number 27: 79 observations,    complexity param=0.01389365
##   mean=51.87342, MSE=713.8827
##   left son=54 (22 obs) right son=55 (57 obs)
```

```
##   Primary splits:
##       four_star_min_perc < 23.79002  to the right, improve=0.05868977, (0
missing)
##       four_stars         < 18.5      to the right, improve=0.05858210, (0
missing)
##       talent_rank        < 44.5      to the left,  improve=0.04815265, (0
missing)
##       roster_size        < 79.5      to the right, improve=0.01632708, (0
missing)
##       five_stars         < 2.5       to the right, improve=0.01538650, (0
missing)
##   Surrogate splits:
##       four_stars     < 18.5      to the right, agree=0.987, adj=0.955, (0
split)
##       talent_rank    < 23.5      to the left,  agree=0.937, adj=0.773, (0
split)
##       three_stars    < 51.5      to the left,  agree=0.886, adj=0.591, (0
split)
##       five_stars     < 2.5       to the right, agree=0.823, adj=0.364, (0
split)
##       five_star_perc < 3.080495  to the right, agree=0.823, adj=0.364, (0
split)
##
## Node number 30: 14 observations
##   mean=62.21429, MSE=649.1684
##
## Node number 31: 18 observations
##   mean=81.22222, MSE=1124.395
##
## Node number 52: 18 observations
##   mean=32.77778, MSE=479.284
##
## Node number 53: 23 observations,    complexity param=0.01389365
##   mean=48.30435, MSE=626.2987
##   left son=106 (15 obs) right son=107 (8 obs)
##   Primary splits:
##       four_stars         < 4.5       to the left,  improve=0.25547340, (0
missing)
##       four_star_min_perc < 5.472822  to the left,  improve=0.19882860, (0
missing)
##       roster_size        < 84        to the right, improve=0.12539830, (0
missing)
##       talent_rank        < 52.5      to the right, improve=0.11329500, (0
missing)
##       three_stars        < 70.5      to the right, improve=0.09515263, (0
missing)
##   Surrogate splits:
##       four_star_min_perc < 5.472822  to the left,  agree=0.957, adj=0.875,
(0 split)
##       talent_rank        < 45.5      to the right, agree=0.783, adj=0.375,
```

```
(0 split)
##       three_stars        < 67.5      to the right, agree=0.696, adj=0.125,
(0 split)
##
## Node number 54: 22 observations
##   mean=41.45455, MSE=553.2479
##
## Node number 55: 57 observations,    complexity param=0.01308073
##   mean=55.89474, MSE=717.8135
##   left son=110 (49 obs) right son=111 (8 obs)
##   Primary splits:
##       three_stars        < 56.5      to the right, improve=0.05991470, (0
missing)
##       five_stars         < 0.5       to the left,  improve=0.04177071, (0
missing)
##       five_star_perc     < 0.5882353 to the left,  improve=0.04177071, (0
missing)
##       talent_rank        < 27.5      to the right, improve=0.04130873, (0
missing)
##       four_star_min_perc < 21.11455  to the left,  improve=0.03572384, (0
missing)
##   Surrogate splits:
##       talent_rank        < 23        to the right, agree=0.895, adj=0.250,
(0 split)
##       four_stars         < 17.5      to the left,  agree=0.877, adj=0.125,
(0 split)
##       four_star_min_perc < 22.01984  to the left,  agree=0.877, adj=0.125,
(0 split)
##
## Node number 106: 15 observations
##   mean=39.06667, MSE=188.0622
##
## Node number 107: 8 observations
##   mean=65.625, MSE=987.9844
##
## Node number 110: 49 observations,    complexity param=0.01308073
##   mean=53.2449, MSE=666.5931
##   left son=220 (39 obs) right son=221 (10 obs)
##   Primary splits:
##       talent_rank        < 29.5      to the right, improve=0.03153997, (0
missing)
##       roster_size        < 83.5      to the left,  improve=0.02960742, (0
missing)
##       four_star_min_perc < 15.95633  to the right, improve=0.02498806, (0
missing)
##       five_stars         < 0.5       to the left,  improve=0.02093590, (0
missing)
##       five_star_perc     < 0.5882353 to the left,  improve=0.02093590, (0
missing)
##   Surrogate splits:
```

```
##       four_stars         < 14.5      to the left,  agree=0.939, adj=0.7, (
0 split)
##       four_star_min_perc < 17.78933  to the left,  agree=0.939, adj=0.7, (
0 split)
##       three_stars        < 58.5      to the right, agree=0.837, adj=0.2, (
0 split)
##       five_stars         < 1.5       to the left,  agree=0.816, adj=0.1, (
0 split)
##       five_star_perc     < 1.825821  to the left,  agree=0.816, adj=0.1, (
0 split)
##
## Node number 111: 8 observations
##   mean=72.125, MSE=725.1094
##
## Node number 220: 39 observations,    complexity param=0.01308073
##   mean=50.92308, MSE=669.2505
##   left son=440 (8 obs) right son=441 (31 obs)
##   Primary splits:
##       four_star_min_perc < 15.95633  to the right, improve=0.1708301, (0 m
issing)
##       talent_rank        < 33.5      to the left,  improve=0.1204381, (0 m
issing)
##       four_stars         < 13.5      to the right, improve=0.1188162, (0 m
issing)
##       five_stars         < 0.5       to the left,  improve=0.0200003, (0 m
issing)
##       five_star_perc     < 0.5882353 to the left,  improve=0.0200003, (0 m
issing)
##   Surrogate splits:
##       four_stars  < 13.5      to the right, agree=0.974, adj=0.875, (0 spl
it)
##       talent_rank < 32.5      to the left,  agree=0.923, adj=0.625, (0 spl
it)
##
## Node number 221: 10 observations
##   mean=62.3, MSE=553.21
##
## Node number 440: 8 observations
##   mean=29.875, MSE=216.8594
##
## Node number 441: 31 observations,    complexity param=0.0108332
##   mean=56.35484, MSE=642.1644
##   left son=882 (21 obs) right son=883 (10 obs)
##   Primary splits:
##       four_stars         < 9.5       to the left,  improve=0.095445580, (0
missing)
##       four_star_min_perc < 11.9462   to the left,  improve=0.095445580, (0
missing)
##       three_stars        < 62.5      to the right, improve=0.073782800, (0
missing)
```

```
##       roster_size        < 78.5      to the left,  improve=0.043293320, (0
missing)
##       talent_rank        < 40        to the right, improve=0.009220256, (0
missing)
##   Surrogate splits:
##       four_star_min_perc < 11.9462   to the left,  agree=1.000, adj=1.0, (
0 split)
##       talent_rank        < 38.5      to the right, agree=0.935, adj=0.8, (
0 split)
##
## Node number 882: 21 observations,    complexity param=0.0108332
##   mean=50.95238, MSE=507.4739
##   left son=1764 (7 obs) right son=1765 (14 obs)
##   Primary splits:
##       four_star_min_perc < 9.150718  to the right, improve=0.233089500, (0
missing)
##       talent_rank        < 45.5      to the left,  improve=0.111231200, (0
missing)
##       four_stars         < 6.5       to the right, improve=0.012557540, (0
missing)
##       three_stars        < 63.5      to the right, improve=0.011263000, (0
missing)
##       roster_size        < 78.5      to the left,  improve=0.009693968, (0
missing)
##   Surrogate splits:
##       four_stars  < 7.5       to the right, agree=0.952, adj=0.857, (0 spl
it)
##       talent_rank < 45.5      to the left,  agree=0.857, adj=0.571, (0 spl
it)
##
## Node number 883: 10 observations
##   mean=67.7, MSE=735.01
##
## Node number 1764: 7 observations
##   mean=35.57143, MSE=541.3878
##
## Node number 1765: 14 observations
##   mean=58.64286, MSE=313.0867
```

Based on the rpart plot we can see there are several trees or "nodes" of interest among the variables included. At the bottom we can see the number of data points that followed each branch sequence. This is a great starting point to visualize the model but we will need to further explore the fit and accuracy of the model and see if we can make predictions.

```
## if we want to make predictions we can split data into training & test sets
# use 60% of sample size for training set
smp_size <- floor(0.6*nrow(data2alt))

# set seed to keep training & test sets the same rows (use random number)
set.seed(10)
```

```
# create training set using ~60% of data
train_rows <- sample(seq_len(nrow(data2alt)), size = smp_size)
rpart_train_1 <- data2alt[train_rows, ]
rpart_test_1 <- data2alt[-train_rows, ]

library(tree)

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli

# remove character variables (columns 2&3)
data3alt <- data2alt[-c(2,3)]

# create training set
train.set <- sample(1:nrow(data3alt), nrow(data3alt)*0.6)

# create regression tree
tree.data <- tree(sp_finish ~ ., data3alt, subset = train.set)

# create yhat for predicted values
yhat <- predict(tree.data, newdata = data3alt[-train.set,])

# test set on y variable "sp_finish"
data.test <- data3alt[-train.set, "sp_finish"]

data.test.vector <- as.vector(data.test['sp_finish'])
class(data.test.vector)

## [1] "tbl_df"     "tbl"        "data.frame"

#head(data.test.vector)

#str(data.test)

# mean square error (MSE)
mean((yhat - data.test.vector)^2)

## Warning in mean.default((yhat - data.test.vector)^2): argument is not nume
ric or
## logical: returning NA

## [1] NA

str(data3alt)

## tibble [221 x 9] (S3: tbl_df/tbl/data.frame)
##  $ year              : num [1:221] 2019 2019 2019 2019 2019 ...
##  $ talent_rank       : num [1:221] 1 2 3 4 5 6 7 8 9 10 ...
##  $ roster_size       : num [1:221] 85 85 85 83 85 83 85 85 80 79 ...
```

```
##  $ five_stars        : num [1:221] 11 13 14 6 7 5 3 5 7 4 ...
##  $ four_stars        : num [1:221] 58 47 45 41 44 36 43 45 33 46 ...
##  $ three_stars       : num [1:221] 13 25 25 31 32 40 37 31 33 25 ...
##  $ sp_finish         : num [1:221] 4 2 5 26 1 56 15 7 3 8 ...
##  $ five_star_perc    : num [1:221] 12.94 15.29 16.47 7.23 8.24 ...
##  $ four_star_min_perc: num [1:221] 81.2 70.6 69.4 56.6 60 ...
```

```
# replicate model using only training set
decision_tree1_train <- rpart(sp_finish ~ talent_rank + five_stars + four_sta
rs + three_stars + four_star_min_perc + five_star_perc, data = rpart_train_1)
#decision_tree1_train

# use analysis from training model to make predictions on test set and store
as vector
rpart_predict_1 <- c(predict(decision_tree1_train, rpart_test_1))

# list all predictions on test set
rpart_predict_1
```

```
##        1        2        3        4        5        6        7        8
## 12.17647 12.17647 12.17647 12.17647 12.17647 12.17647 42.50000 42.50000
##        9       10       11       12       13       14       15       16
## 12.17647 42.50000 42.50000 71.63636 71.63636 40.91667 40.91667 35.50000
##       17       18       19       20       21       22       23       24
## 40.91667 40.91667 35.50000 35.50000 35.50000 58.14286 35.50000 35.50000
##       25       26       27       28       29       30       31       32
## 58.14286 35.50000 88.00000 88.00000 88.00000 88.00000 88.00000 88.00000
##       33       34       35       36       37       38       39       40
## 88.00000 88.00000 45.11111 12.17647 12.17647 12.17647 12.17647 12.17647
##       41       42       43       44       45       46       47       48
## 12.17647 42.50000 71.63636 71.63636 35.50000 35.50000 40.91667 40.91667
##       49       50       51       52       53       54       55       56
## 40.91667 35.50000 35.50000 58.14286 35.50000 35.50000 35.50000 58.14286
##       57       58       59       60       61       62       63       64
## 35.50000 35.50000 58.14286 88.00000 88.00000 88.00000 45.11111 88.00000
##       65       66       67       68       69       70       71       72
## 88.00000 88.00000 88.00000 12.17647 12.17647 12.17647 12.17647 42.50000
##       73       74       75       76       77       78       79       80
## 42.50000 42.50000 42.50000 42.50000 71.63636 71.63636 40.91667 35.50000
##       81       82       83       84       85       86       87       88
## 40.91667 58.14286 58.14286 35.50000 58.14286 88.00000 88.00000 45.11111
##       89
## 45.11111
```

```
# convert predicted results as numeric in order to create vector
rpart_predict_1 <- as.numeric(rpart_predict_1)

# verify vector is TRUE if length is equal to sample size
length(rpart_predict_1)
```

```
## [1] 89
```

```
# view specific entries included in the test set
rpart_test_1
```

```
## # A tibble: 89 x 11
##     year school conference talent_rank roster_size five_stars four_stars
##    <dbl> <chr>  <chr>            <dbl>       <dbl>      <dbl>      <dbl>
##  1  2019 Alaba~ SEC                  1          85         11         58
##  2  2019 Ohio ~ B10                  2          85         13         47
##  3  2019 Georg~ SEC                  3          85         14         45
##  4  2019 LSU    SEC                  5          85          7         44
##  5  2019 Oklah~ B12                  8          85          5         45
##  6  2019 Texas~ SEC                 12          85          2         40
##  7  2019 Tenne~ SEC                 16          85          4         32
##  8  2019 Oregon P12                 17          85          1         31
##  9  2019 Washi~ P12                 19          85          1         41
## 10  2019 South~ SEC                 21          84          1         23
## # ... with 79 more rows, and 4 more variables: three_stars <dbl>,
## #   sp_finish <dbl>, five_star_perc <dbl>, four_star_min_perc <dbl>
```

Now that we have a list of predictions I will work to merge the predicted values into the data frame of the training set to illustrate the predicted finish with the actual finish in a single data frame.

```
# remove columns so only 'year', 'school' and 'sp_finish' remain
predicted_results_sub <- rpart_test_1[c(1,2,9)]

# for reference
head(predicted_results_sub, 3)
```

```
## # A tibble: 3 x 3
##    year school     sp_finish
##   <dbl> <chr>          <dbl>
## 1  2019 Alabama            4
## 2  2019 Ohio State         2
## 3  2019 Georgia            5
```

```
# merge predicted results into testing set dataframe
predicted_results_merge <- predicted_results_sub %>%
  mutate(year, pred_finish = rpart_predict_1)

# now we can see side-by-side the predicted finish and sp_finish for each tea
m (by year) included in the test set
predicted_results_merge
```

```
## # A tibble: 89 x 4
##     year school         sp_finish pred_finish
##    <dbl> <chr>              <dbl>       <dbl>
##  1  2019 Alabama                4        12.2
##  2  2019 Ohio State             2        12.2
##  3  2019 Georgia                5        12.2
##  4  2019 LSU                    1        12.2
```

```
##  5  2019 Oklahoma              7        12.2
##  6  2019 Texas A&M            19        12.2
##  7  2019 Tennessee            35        42.5
##  8  2019 Oregon                6        42.5
##  9  2019 Washington           17        12.2
## 10  2019 South Carolina       55        42.5
## # ... with 79 more rows
```

Based on the dataframe above we can see the predicted finish based on our decision tree model in the far right column, and the actual S&P Finish in the column to the left. We can see some estimates are accurate while others not so much. Could these be outliers? How accurate are these predictions? One metric we can use is the Mean Absolute Error which we can be interpreted as the mean different between the actual and predicted results.

```
# calculate mean square error (MSE)
mean((predicted_results_merge$pred_finish - predicted_results_merge$sp_finish
)^2)

## [1] 841.2366
```

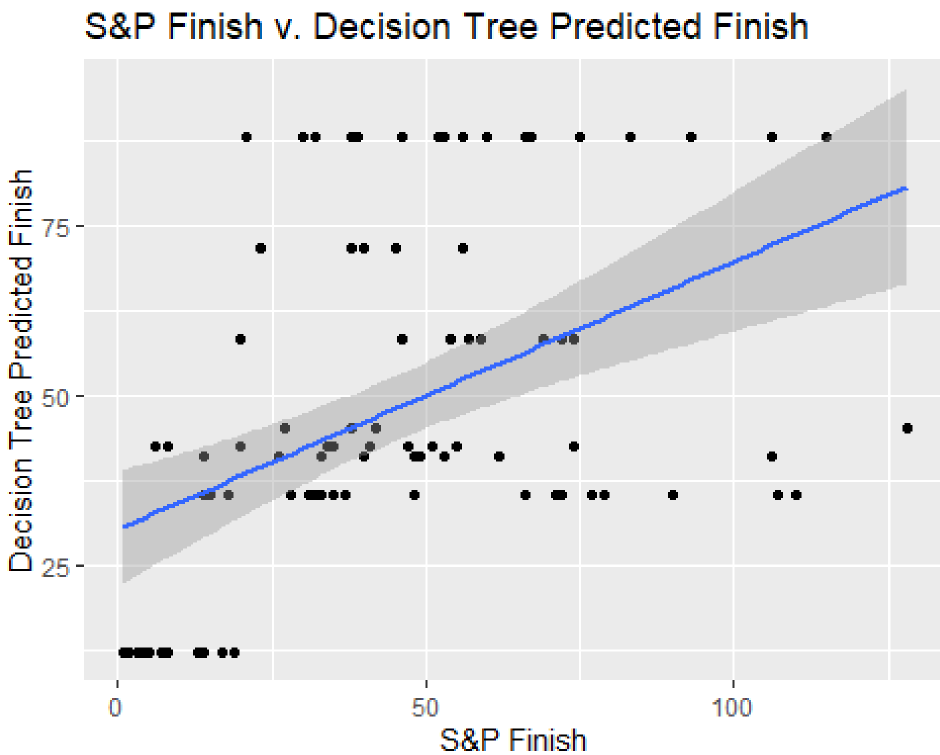
The MSE is ~946, which we can interpret as the average of the square of the residuals. Meaning, if we sqaured each residual (difference between the actual and predicted value) it would give us the average. This number is great for telling us the accuracy of our model but not so great to interpret in context. For that reason we will use the Mean Absolute Error.

```
# load package to calculate MAE
library(Metrics)

# Mean Absolute Error (MAE)
mae(predicted_results_merge$sp_finish, predicted_results_merge$pred_finish)

## [1] 21.34838
```

Our MAE is ~ 23, which we can interpret as our decision tree model can predict the final result within 23 (or +/- 11.5) spaces away form the observed value. In context, based on the recruiting criteria included if a team were to finish the season at #40, our model would have predicted their finish between ~28 and ~52.

```
## ggplot

# create data frame of actual v predicted results
mae_df <- data.frame(predicted_results_merge$sp_finish, predicted_results_mer
ge$pred_finish)

# confirm data frame created
# head(mae_df)

mae_plot <- ggplot(data=mae_df, aes(predicted_results_merge$sp_finish, predic
ted_results_merge$pred_finish)) +
  geom_point() +
```

```
  geom_smooth(method = "lm") +
  labs(x = "S&P Finish", y = "Decision Tree Predicted Finish") +
  ggtitle("S&P Finish v. Decision Tree Predicted Finish")

mae_plot

## `geom_smooth()` using formula 'y ~ x'
```



Machine Learning Part 2

Random Forest Modeling - Classification

In our next portion we will explore the specific 'decisions' made by the tree and the importance of each, as well as key metrics. Random forest modeling is most widely used for classifying data based on the variables included. To classify the data we will need a categorical dependent variable, which we will utilize the independent variables to make predictions on how to classify the dependent variable. Currently our dependent variable of interest 'sp_finish' is continious, but we can convert this to categorical whether a team finised inside the the Top 25, a widely used measuing stick for college football fans.

```
## we will convert our data set so the dependent variable is categorical
data3 <- data2

# add new column where "yes" = finish top 25 & "no" = did not finish top 25
data3 <- data3 %>%
  mutate(school, fin_top_25 = cut(sp_finish, breaks = c(0, 25, 117), labels =
```

```
c("yes", "no")))

# for our decision trees the variables 'year', 'school', & 'conference' are g
ood for reference but will likley add unnecessary noise to a decision tree mo
del. We will remove
data3 <- within(data3, rm(year, school, conference))

# in addition we will remove 'sp_finish' variable since this is directly rela
ted to the 'fin_top_25' variable we are attempting to predict with the remain
ing variables
data3 <- within(data3, rm(sp_finish))

# notice 12th column 'fin_top_25' has been added as categorical variable
head(data3,3)

## # A tibble: 3 x 8
##   talent_rank roster_size five_stars four_stars three_stars five_star_perc
##         <dbl>       <dbl>      <dbl>      <dbl>       <dbl>          <dbl>
## 1           1          85         11         58          13           12.9
## 2           2          85         13         47          25           15.3
## 3           3          85         14         45          25           16.5
## # ... with 2 more variables: four_star_min_perc <dbl>, fin_top_25 <fct>

## divide data into training & test sets

# use 75% of sample size for training set
smp_size <- floor(0.75*nrow(data3))

# set seed to keep training & test sets the same rows (use random number)
set.seed(55)

# create training set using ~75% of data (112 rows)
train_rows <- sample(seq_len(nrow(data3)), size = smp_size)
top25_train <- data3[train_rows, ]
top25_test <- data3[-train_rows, ]

# prop table for reference
prop.table(table(top25_train$fin_top_25))

##
##       yes        no
## 0.3580247 0.6419753

prop.table(table(top25_test$fin_top_25))

##
##       yes        no
## 0.2545455 0.7454545

# if proportions are relatively equal for both sets (within 5% margin) we can
proceed
```

The proportion tables are realtively equal. We will proceed with the assumption the testing and training sets are equally representative of the data.

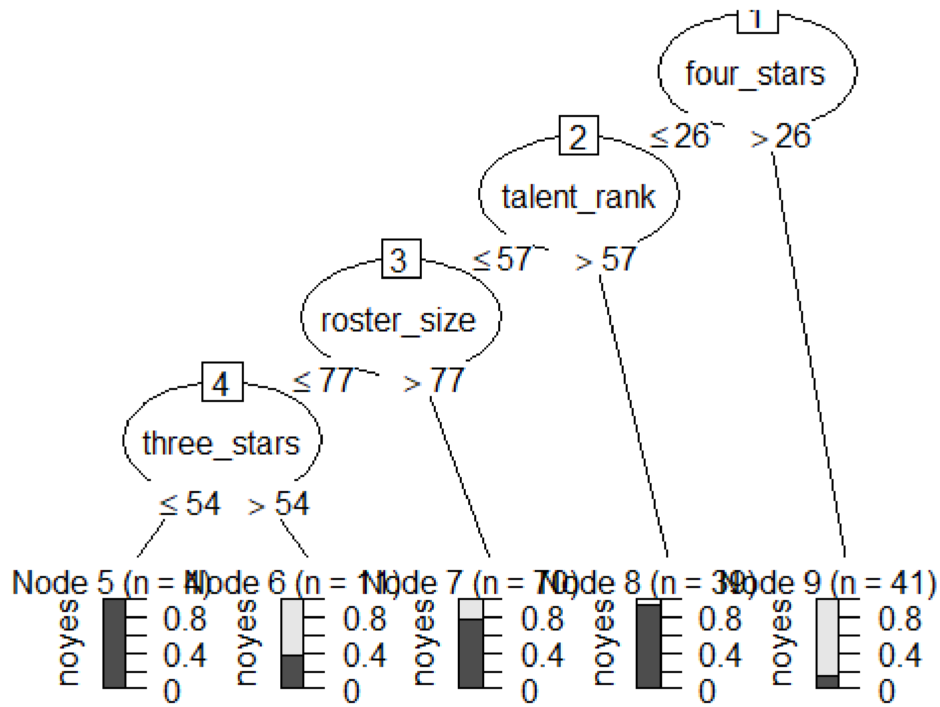
```
# show classification tree using c5.0 package
library(C50)

# produce classification tree (note the 11th column is removed since 1-10 are
predictors)
tree1 <- C5.0(top25_train[-8], top25_train$fin_top_25)
tree1

##
## Call:
## C5.0.default(x = top25_train[-8], y = top25_train$fin_top_25)
##
## Classification Tree
## Number of samples: 165
## Number of predictors: 7
##
## Tree size: 5
##
## Non-standard options: attempt to group attributes
```

We can see from our classification tree there were 165 samples included (the training set), 7 predictors utilized, and a tree size of 8. In other words, there were 8 "tree nodes" or decisions. This is an indication several of the variables included are 'useful' in categorizing the dependent variable.

```
# summary of classification tree for further details
summary(tree1)

##
## Call:
## C5.0.default(x = top25_train[-8], y = top25_train$fin_top_25)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Jan 07 10:51:39 2021
## -------------------------------
##
## Class specified by attribute `outcome'
## *** ignoring cases with bad or unknown class
##
## Read 162 cases (8 attributes) from undefined.data
##
## Decision tree:
##
## four_stars > 26: yes (41/6)
## four_stars <= 26:
## :...talent_rank > 57: no (36)
##     talent_rank <= 57:
```

```
##     :...roster_size > 77: no (70/16)
##         roster_size <= 77:
##         :...three_stars <= 54: no (4)
##             three_stars > 54: yes (11/4)
##
##
## Evaluation on training data (162 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       5   26(16.0%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      42    16    (a): class yes
##      10    94    (b): class no
##
##
##  Attribute usage:
##
##  100.00% four_stars
##   74.69% talent_rank
##   52.47% roster_size
##    9.26% three_stars
##
##
## Time: 0.0 secs
```

```
plot(tree1)
```



Based on the table produced from our random forest model, we can see the model used 4 key variables/nodes (equal to 8 brances since there are 2 branches per node). The first node is regarded as the most important, which is 'four_stars', which complements our findings from the linear regression analysis earlier.

It is worth noting the model was much more effective in classifying the teams that did NOT finish inside the Top 25. This could suggest recruiting will certainly hamper teams from achieving success on the field if their recruiting does not meet a certain threshold, but there are other factors not included in our model that are key to finishing inside the Top 25.

Next we will explore this same concept but change the classification to finishing inside the Top 10 and see if our model performs better or worse, another widely regarded measuring stick from college football fans.

```
## we will convert our data set so the dependent variable is categorical
data3alt <- data2

# add new column where "yes" = finish top 10 & "no" = did not finish top 10
data3alt <- data3alt %>%
  mutate(school, fin_top_10 = cut(sp_finish, breaks = c(0, 10, 117), labels =
c("yes", "no")))

# for our decision trees the variables 'year', 'school', & 'conference' are g
ood for reference but will likley add unnecessary noise to a decision tree mo
del. We will remove
data3alt <- within(data3alt, rm(year, school, conference))
```

```r
# in addition we will remove 'sp_finish' variable since this is directly rela
ted to the 'fin_top_10' variable we are attempting to predict with the remain
ing variables
data3alt <- within(data3alt, rm(sp_finish))

# notice 12th column 'fin_top_10' has been added as categorical variable
head(data3alt,3)

## # A tibble: 3 x 8
##   talent_rank roster_size five_stars four_stars three_stars five_star_perc
##         <dbl>       <dbl>      <dbl>      <dbl>       <dbl>          <dbl>
## 1           1          85         11         58          13           12.9
## 2           2          85         13         47          25           15.3
## 3           3          85         14         45          25           16.5
## # ... with 2 more variables: four_star_min_perc <dbl>, fin_top_10 <fct>

## divide data into training & test sets

# use 75% of sample size for training set
smp_size <- floor(0.75*nrow(data3alt))

# set seed to keep training & test sets the same rows (use random number)
set.seed(55)

# create training set using ~75% of data (112 rows)
train_rows <- sample(seq_len(nrow(data3alt)), size = smp_size)
top10_train <- data3alt[train_rows, ]
top10_test <- data3alt[-train_rows, ]

# prop table for reference
prop.table(table(top10_train$fin_top_10))

##
##       yes        no
## 0.1419753 0.8580247

prop.table(table(top10_test$fin_top_10))

##
##       yes        no
## 0.1272727 0.8727273

# if proportions are relatively equal for both sets (within 5% margin) we can
proceed
```

The proportion tables of the training and testing data sets are relatively equal.

```r
# produce classification tree (note the 11th column is removed since 1-10 are
predictors)
```

```r
tree2 <- C5.0(top10_train[-8], top10_train$fin_top_10)
tree2

##
## Call:
## C5.0.default(x = top10_train[-8], y = top10_train$fin_top_10)
##
## Classification Tree
## Number of samples: 165
## Number of predictors: 7
##
## Tree size: 8
##
## Non-standard options: attempt to group attributes
```

The tree will utilize 7 predictors, but interestingly the tree size is only 2 nodes now. This is likely an indicator there is a key variable that stands out above the others as opposed to our tree1 model.

```r
summary(tree2)

##
## Call:
## C5.0.default(x = top10_train[-8], y = top10_train$fin_top_10)
##
##
## C5.0 [Release 2.07 GPL Edition]      Thu Jan 07 10:51:39 2021
## -------------------------------
##
## Class specified by attribute `outcome'
## *** ignoring cases with bad or unknown class
##
## Read 162 cases (8 attributes) from undefined.data
##
## Decision tree:
##
## four_star_min_perc > 57.64706: yes (12/1)
## four_star_min_perc <= 57.64706:
## :...four_stars <= 19: no (113/3)
##     four_stars > 19:
##     :...four_star_min_perc > 51.94805: no (11)
##         four_star_min_perc <= 51.94805:
##         :...four_star_min_perc > 49.41177: yes (4)
##             four_star_min_perc <= 49.41177:
##             :...five_star_perc > 2.380952: no (9)
##                 five_star_perc <= 2.380952:
##                 :...roster_size <= 84: no (5/1)
##                     roster_size > 84:
##                     :...four_stars <= 33: yes (6/2)
##                         four_stars > 33: no (2)
##
```

```
##
## Evaluation on training data (162 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       8    7( 4.3%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##      19     4    (a): class yes
##       3   136    (b): class no
##
##
##  Attribute usage:
##
##  100.00% four_star_min_perc
##   92.59% four_stars
##   13.58% five_star_perc
##    8.02% roster_size
##
##
## Time: 0.0 secs
```

The summary of our tree2 analysis yields a lower error rate, which is encouraging that we can better trust our results. In addition, the variable 'five_stars' is now the key branch in finishing inside or outside the top 10. Intuitively this makes sense as five star players are rated as the very best high school recruits, so it is reasonable to assume the college teams that accumulate the most five star players would finish at the very top. It is worth noting we have a similar problem as with tree1, where the model is much more effective at predicting teams finishing OUTSIDE the top 10, but all of the variance is from predicting teams INSIDE the top 10. This is further evidence that if a team does not accumulate a certain threshold of five star players (greater than or equal to 8) it will hamper the teams success, but there are likely other varaibles not included in our model that would explain the teams that do finish inside the top 10.