# Final Project

## MGSC 310

Tobi Howe & Zach Rosenstrauch

```
# Always print this out before your assignment
sessionInfo()
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
## [7] base
##
## other attached packages:
## [1] knitr_1.31
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.0.3    magrittr_2.0.1    tools_4.0.3
##  [4] htmltools_0.5.1.1 yaml_2.2.1        stringi_1.5.3
##  [7] rmarkdown_2.6     stringr_1.4.0     xfun_0.21
## [10] digest_0.6.27     rlang_0.4.11      evaluate_0.14
getwd()
## [1] "/Users/tobihowe/Documents/MGSC_310_2021"
```

```
# load all your libraries in this chunk
library('here')
library('tidyverse')
library('fs')
library('rsample')
library('glmnet')
library('glmnetUtils')
library('forcats')
library('dplyr')
library('plotROC')
library('ggplot2')
library('ISLR')
library('yardstick')
library('caret')
library('dbplyr')
library('coefplot')
library('Metrics')
# note, do not run install.packages() inside a code chunk. install them in the console outside o
f a code chunk.
```

# Cleaning the Data

```
#download data for each match
matches <- read.csv(here("datasets", "spi_matches.csv"), stringsAsFactors = TRUE)

#download data for standings
standings <- read.csv(here("datasets", "soccer-standings.csv"), stringsAsFactors = TRUE)

head(matches)
```

| | sea… <int> | date <fct> | league_id <int> | league <fct> | team1 <fct> | team2 <fct> |
|---|---|---|---|---|---|---|
| 1 | 2016 | 2016-07-09 | 7921 | FA Women's Super League | Liverpool Women | Reading |
| 2 | 2016 | 2016-07-10 | 7921 | FA Women's Super League | Arsenal Women | Notts County Lad |
| 3 | 2016 | 2016-07-10 | 7921 | FA Women's Super League | Chelsea FC Women | Birmingham City |
| 4 | 2016 | 2016-07-16 | 7921 | FA Women's Super League | Liverpool Women | Notts County Lad |
| 5 | 2016 | 2016-07-17 | 7921 | FA Women's Super League | Chelsea FC Women | Arsenal Women |
| 6 | 2016 | 2016-07-24 | 7921 | FA Women's Super League | Reading | Birmingham City |

6 rows | 1-8 of 24 columns

```
head(standings)
```

| | X <fct> | Total <fct> | X.1 <fct> | X.2 <fct> | X.3 <fct> | X.4 <fct> | X.5 <fct> | X.6 <fct> | X.7 <fct> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Team | Rank | P | M | W | D | L | G | GA |
| 2 | Manchester City | 1 | 86 | 38 | 27 | 5 | 6 | 83 | 32 |
| 3 | Manchester United | 2 | 74 | 38 | 21 | 11 | 6 | 73 | 44 |
| 4 | Liverpool | 3 | 69 | 38 | 20 | 9 | 9 | 68 | 42 |
| 5 | Chelsea | 4 | 67 | 38 | 19 | 10 | 9 | 58 | 36 |
| 6 | Leicester City | 5 | 66 | 38 | 20 | 6 | 12 | 68 | 50 |

6 rows | 1-10 of 29 columns

When cleaning the data I mutated team1, team2, and league to a factor. I also only selected data from last season and only the games from the Barclays Premier League.

```
#clean data
matches_clean <- matches %>%
  mutate(team1 = as.factor(team1), #converting team1 to factor
         team2 = as.factor(team2), #converting team2 to factor
         league = as.factor(league)) %>%  #converting league to factor
  filter(season == 2020, league == "Barclays Premier League" ) %>% #filtering and selecting only
2020 Barclays Premier League Season
  drop_na() #dropping na

head(matches_clean)
```

| | sea… <int> | date <fct> | league_id <int> | league <fct> | team1 <fct> | team2 <fct> |
|---|---|---|---|---|---|---|
| 1 | 2020 | 2020-09-12 | 2411 | Barclays Premier League | Fulham | Arsenal |
| 2 | 2020 | 2020-09-12 | 2411 | Barclays Premier League | Crystal Palace | Southampto |
| 3 | 2020 | 2020-09-12 | 2411 | Barclays Premier League | Liverpool | Leeds Unite |
| 4 | 2020 | 2020-09-12 | 2411 | Barclays Premier League | West Ham United | Newcastle |
| 5 | 2020 | 2020-09-13 | 2411 | Barclays Premier League | West Bromwich Albion | Leicester Ci |
| 6 | 2020 | 2020-09-13 | 2411 | Barclays Premier League | Tottenham Hotspur | Everton |

6 rows | 1-8 of 24 columns

Next I cleaned the data for the standings dataset. I removed the first row that had all the labels and then I selected just the team and total points from last season as points will be the column that I will try to predict. The team with the most points at the end of the season wins the league.

```
#cleaning standings data
standings <- standings %>% slice(-c(1)) %>% #remove first row with labels
  mutate(team = as.factor(X), #convert team to factor and store
         points = as.numeric(as.character(X.1))) %>% #convert points to numeric and store
  select(team, points) %>% #select only team and sports
  drop_na() #drop nulls

head(standings)
```

| | team<br><fct> | points<br><dbl> |
|---|---|---|
| 1 | Manchester City | 86 |
| 2 | Manchester United | 74 |
| 3 | Liverpool | 69 |
| 4 | Chelsea | 67 |
| 5 | Leicester City | 66 |
| 6 | West Ham United | 65 |

6 rows

Below I created summary statistics for each team who played at home during the 2020-2021 season. It averages the SPI, importance, probability to win, projected score, score, adjusted score, shot-based expected goals, and non-shot based expected goals.

```
#gathering data for home teams
home_summary <- matches_clean %>%
  mutate(team = team1) %>% #making team1 the team variable
  group_by(team) %>% #grouping by team
  summarize(avg_spi = mean(spi1), #average spi for each team
            avg_importance = mean(importance1), #average importance
            avg_prob = mean(prob1), #average probability to win
            avg_proj_score = mean(proj_score1), #average projected score
            avg_score = mean(score1), #average score
            avg_adj_score = mean(adj_score1), #average adjusted score
            avg_shot_exp_goals = mean(xg1), #average shot-based expected goals
            avg_nonshot_exp_goals = mean(nsxg1)) %>% #average non-based expected goals
  arrange(desc(avg_spi)) #arrange by spi

home_summary
```

| team<br><fct> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_proj_score<br><dbl> | avg_score<br><dbl> |
|---|---|---|---|---|---|
| Manchester City | 93.47667 | 52.47222 | 0.7597278 | 2.423889 | 2.1111111 |

| team<br><fct> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_proj_score<br><dbl> | avg_score<br><dbl> |
|---|---|---|---|---|---|
| Liverpool | 88.80263 | 73.53684 | 0.6749526 | 2.206316 | 1.5263158 |
| Chelsea | 86.98737 | 75.87895 | 0.6108474 | 1.878947 | 1.6315789 |
| Manchester United | 85.73278 | 47.98333 | 0.5991444 | 1.944444 | 2.1111111 |
| Arsenal | 81.01474 | 22.22105 | 0.4753579 | 1.563158 | 1.2631579 |
| Tottenham Hotspur | 79.60579 | 47.31053 | 0.5233895 | 1.762632 | 1.8421053 |
| Leicester City | 79.49684 | 71.63158 | 0.5286842 | 1.705263 | 1.7894737 |
| West Ham United | 74.91421 | 41.85263 | 0.4341895 | 1.523684 | 1.6842105 |
| Brighton and Hove Albion | 74.44526 | 18.44737 | 0.4028421 | 1.355789 | 1.1578947 |
| Wolverhampton | 74.02263 | 11.46316 | 0.3747474 | 1.239474 | 1.1052632 |

1-10 of 20 rows | 1-7 of 9 columns Previous **1** 2 Next

As one can see from the data table, Manchester City had the highest average SPI of 93.48 which indicates that there more likely to win every game that they played. Manchester City also had the highest probability of winning, the most projected goals, the highest average goals per game, and the highest average adjusted score. Adjusted score accounts for goals that are more or less significant towards the result of the game. As far as importance, Chelsea had the average importance for each game which means that each game was significant in deciding where they would finish in the league.

Below I did the same thing for the away team.

```
#average for all away teams
away_summary <- matches_clean %>%
  mutate(team = team2) %>% #store as team
  group_by(team) %>% #group by team
  summarize(avg_spi = mean(spi2), #average spi
            avg_importance = mean(importance2),#average importance
            avg_prob = mean(prob2), #average probability
            avg_proj_score = mean(proj_score2), #average projected score
            avg_score = mean(score2), #average score
            avg_adj_score = mean(adj_score2), #average adjusted score
            avg_shot_exp_goals = mean(xg2), #average shot-based expected goals
            avg_nonshot_exp_goals = mean(nsxg2)) %>% #average non-shot based expected goals
  arrange(desc(avg_spi)) #arrange by spi

away_summary
```

| team<br><fct> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_proj_score<br><dbl> | avg_score<br><dbl> |
|---|---|---|---|---|---|
| Manchester City | 93.38056 | 46.62778 | 0.6415278 | 2.1177778 | 2.2222222 |

| team<br><fct> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_proj_score<br><dbl> | avg_score<br><dbl> |
|---|---|---|---|---|---|
| Liverpool | 88.59421 | 77.95263 | 0.5616474 | 1.9673684 | 2.0526316 |
| Chelsea | 86.76684 | 74.82632 | 0.5184211 | 1.7042105 | 1.4210526 |
| Manchester United | 85.34579 | 47.72105 | 0.4782789 | 1.6931579 | 1.8421053 |
| Arsenal | 80.52632 | 22.32105 | 0.3663789 | 1.3205263 | 1.6315789 |
| Tottenham Hotspur | 80.09526 | 52.85789 | 0.3943737 | 1.4426316 | 1.7368421 |
| Leicester City | 78.90947 | 67.81579 | 0.3740000 | 1.3336842 | 1.7894737 |
| Brighton and Hove Albion | 74.73579 | 16.92105 | 0.2981947 | 1.1089474 | 0.9473684 |
| Aston Villa | 74.70789 | 28.54737 | 0.3171158 | 1.2431579 | 1.3684211 |
| West Ham United | 74.62684 | 35.55789 | 0.3029421 | 1.2273684 | 1.5789474 |

1-10 of 20 rows | 1-7 of 9 columns Previous **1** 2 Next

For away games, Manchester City has the highest average SPI, average probability, average projected score, average score, and project score. Liverpool has the highest average importance.

Below I joined the two tables I created above that summarized the statistics for the home and away team. I used a full join so that all the data would be included in the final data set.

```
team <- full_join(x = home_summary, #join home team summary
                  y = away_summary, #and away team summary
                  by = "team") #by the team column

team_summary <- team %>% #clean data
  group_by(team) %>%
  summarize(avg_spi = mean(avg_spi.x, avg_spi.y),
            avg_importance = mean(avg_importance.x, avg_importance.y),
            avg_prob = mean(avg_prob.x, avg_prob.y),
            avg_proj_score = mean(avg_proj_score.x, avg_proj_score.y),
            avg_score = mean(avg_score.x, avg_score.y),
            avg_adj_score = mean(avg_adj_score.x, avg_adj_score.y),
            avg_shot_exp_goals = mean(avg_shot_exp_goals.x, avg_shot_exp_goals.y),
            avg_nonshot_exp_goals = mean(avg_nonshot_exp_goals.x, avg_nonshot_exp_goals.y)) %>%
  arrange(desc(avg_spi))

team_summary
```

| team<br><fct> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_proj_score<br><dbl> | avg_score<br><dbl> |
|---|---|---|---|---|---|
| Manchester City | 93.47667 | 52.47222 | 0.7597278 | 2.423889 | 2.1111111 |
| Liverpool | 88.80263 | 73.53684 | 0.6749526 | 2.206316 | 1.5263158 |

| team <fct> | avg_spi <dbl> | avg_importance <dbl> | avg_prob <dbl> | avg_proj_score <dbl> | avg_score <dbl> |
|---|---|---|---|---|---|
| Chelsea | 86.98737 | 75.87895 | 0.6108474 | 1.878947 | 1.6315789 |
| Manchester United | 85.73278 | 47.98333 | 0.5991444 | 1.944444 | 2.1111111 |
| Arsenal | 81.01474 | 22.22105 | 0.4753579 | 1.563158 | 1.2631579 |
| Tottenham Hotspur | 79.60579 | 47.31053 | 0.5233895 | 1.762632 | 1.8421053 |
| Leicester City | 79.49684 | 71.63158 | 0.5286842 | 1.705263 | 1.7894737 |
| West Ham United | 74.91421 | 41.85263 | 0.4341895 | 1.523684 | 1.6842105 |
| Brighton and Hove Albion | 74.44526 | 18.44737 | 0.4028421 | 1.355789 | 1.1578947 |
| Wolverhampton | 74.02263 | 11.46316 | 0.3747474 | 1.239474 | 1.1052632 |

1-10 of 20 rows | 1-7 of 9 columns Previous **1** 2 Next

Below I joined the standings table with the team summary. We will use the other variables to predict total points for the teams in the Barclays Premier League.

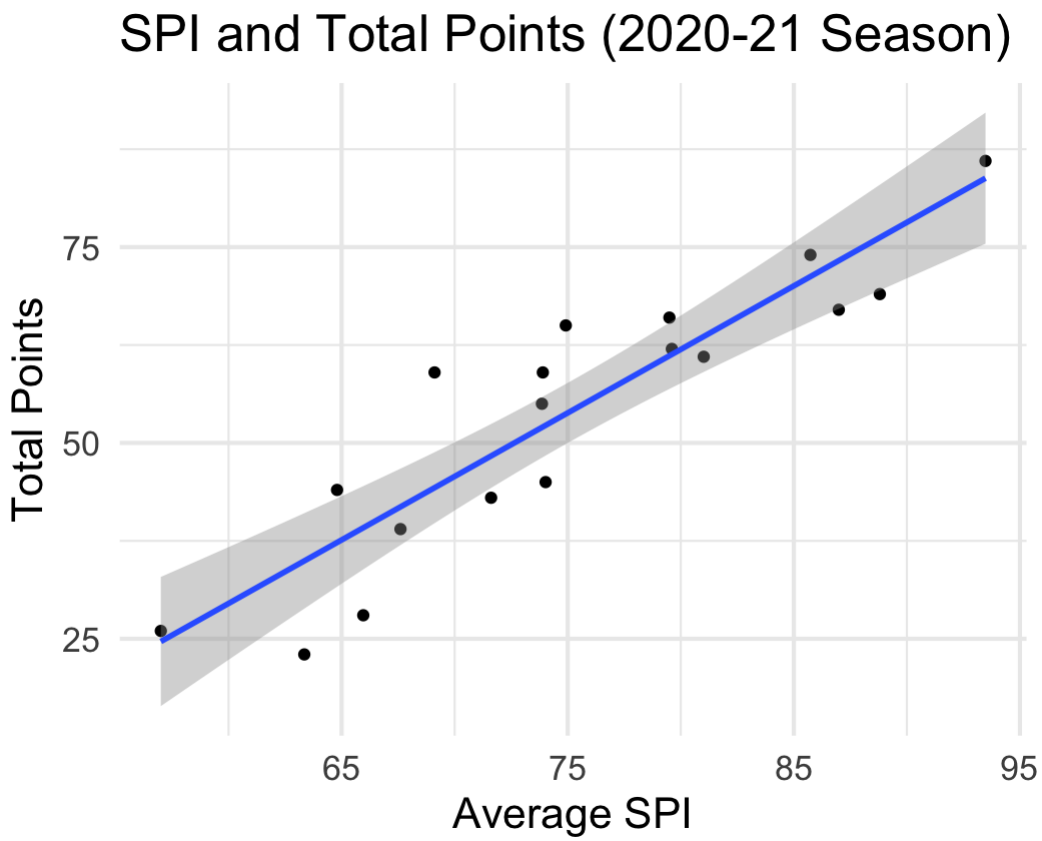
```
league <- full_join(x = standings, #join standings dataframe
                y = team_summary, #with team summary df
                by = "team") #by the team name

league
```

| team <fct> | points <dbl> | avg_spi <dbl> | avg_importance <dbl> | avg_prob <dbl> | avg_proj_score <dbl> | avg |
|---|---|---|---|---|---|---|
| Manchester City | 86 | 93.47667 | 52.47222 | 0.7597278 | 2.423889 | 2 |
| Manchester United | 74 | 85.73278 | 47.98333 | 0.5991444 | 1.944444 | 2 |
| Liverpool | 69 | 88.80263 | 73.53684 | 0.6749526 | 2.206316 | 1.5 |
| Chelsea | 67 | 86.98737 | 75.87895 | 0.6108474 | 1.878947 | 1.6 |
| Leicester City | 66 | 79.49684 | 71.63158 | 0.5286842 | 1.705263 | 1.7 |
| West Ham United | 65 | 74.91421 | 41.85263 | 0.4341895 | 1.523684 | 1.6 |
| Tottenham Hotspur | 62 | 79.60579 | 47.31053 | 0.5233895 | 1.762632 | 1.8 |
| Arsenal | 61 | 81.01474 | 22.22105 | 0.4753579 | 1.563158 | 1.2 |
| Leeds United | 59 | 69.11158 | 11.33684 | 0.3569000 | 1.408421 | 1.4 |
| Everton | 59 | 73.89947 | 24.92632 | 0.4161263 | 1.423684 | 1.2 |

1-10 of 22 rows | 1-7 of 10 columns Previous **1** 2 3 Next

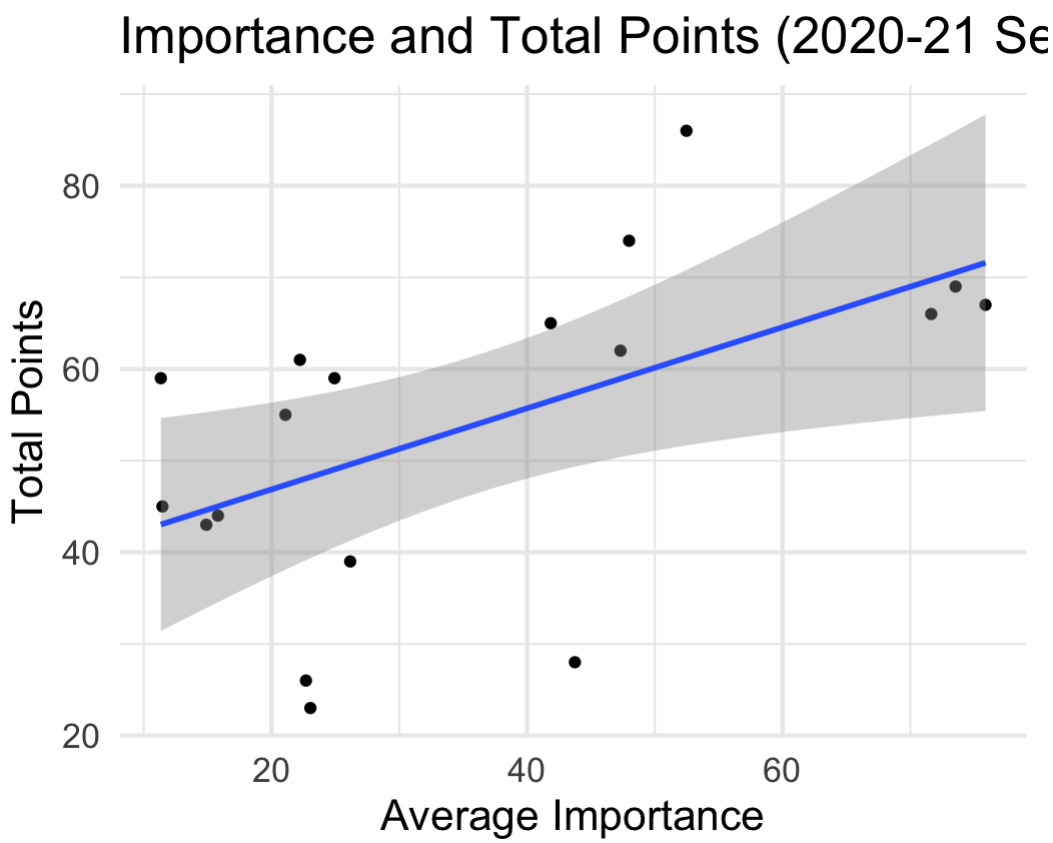
# Elastic Net & Ridge Model

```
#ggplot showing average spi against points
ggplot(league, aes(x = avg_spi, y = points)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average SPI", y = "Total Points") +
  ggtitle("SPI and Total Points (2020-21 Season)")
```



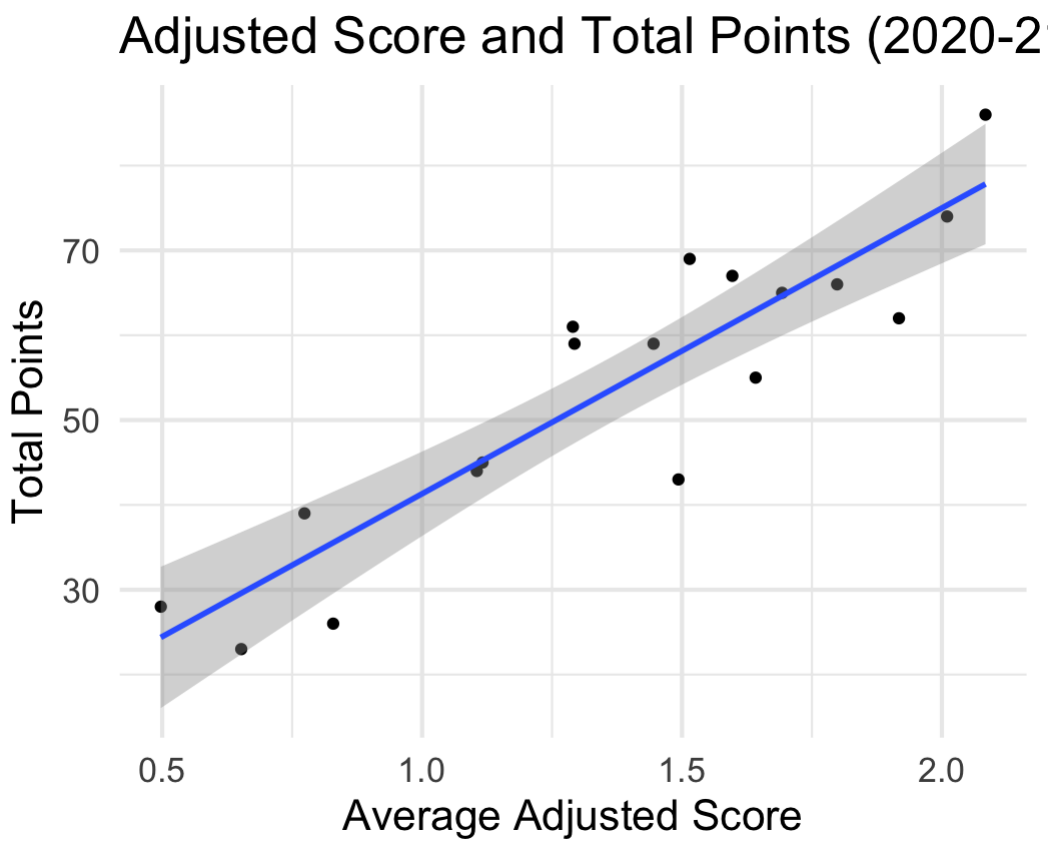

The plot above shows average SPI against the actual total points of the team in the 2020-2021 season. As we can see there is a strong positive linear correlation indicating that the higher the team's SPI is, the more likely they are to finish with more points.

```
#ggplot showing average importance against points
ggplot(league, aes(x = avg_importance, y = points)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average Importance", y = "Total Points") +
  ggtitle("Importance and Total Points (2020-21 Season)")
```




The plot above compares average importance and total points. Importance refers to how important it is for the team to win that game in order to succeed in the league. As one can see there is a very weak positive correlation between the two variables. This suggests that there is some evidence to suggest that the more important the games are for a team, the more likely they are to win. However, for weaker teams in the league, importance does not play a significant role.

```
#ggplot showing average adjusted score against points
ggplot(league, aes(x = avg_adj_score, y = points)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average Adjusted Score", y = "Total Points") +
  ggtitle("Adjusted Score and Total Points (2020-21 Season)")
```



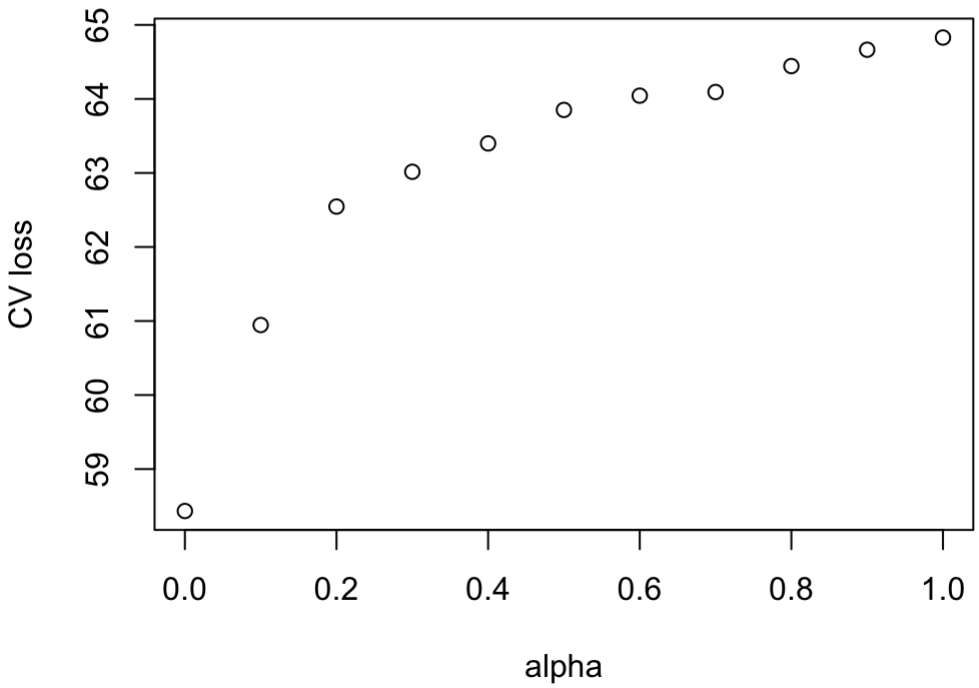

The last plot shows a very strong positive correlation between average adjusted score and total points. Average adjusted score takes the actual scores of the games and adjusts them based on how valuable the goal was to course of the game. Goals scored at the very end or when the opposition is down a player are less important to the game so they are reduced to less than 1.

After looking at the 3 plots, I created the elastic net model below using spi, importance, probability to win, projected score, shot-based expected goals, and non-shot expected score to predict total points. Soccer power index (SPI) is calculated using adjusted goals, shot-based expected goals, non-shot expected goals to predict the strength of the team. The shot-based expected goals are calculated by using the probability of each shot going in using its trajectory. The non-shot expected goals accounts for other moments in the game such as tackles, interceptions, and passes that help contribute to the team winning.

```
#elastic net model
enet_mod <- cva.glmnet(points ~ avg_spi + avg_importance + avg_prob + avg_proj_score + avg_shot_
exp_goals + avg_nonshot_exp_goals,
                       data = league,
                       alpha = seq(0,1, by = 0.1)) #ranging from 0-1 and increasing by 0.1
print(enet_mod)
## Call:
## cva.glmnet.formula(formula = points ~ avg_spi + avg_importance +
##     avg_prob + avg_proj_score + avg_shot_exp_goals + avg_nonshot_exp_goals,
##     data = league, alpha = seq(0, 1, by = 0.1))
##
## Model fitting options:
##     Sparse model matrix: FALSE
##     Use model.frame: FALSE
##     Alpha values: 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
##     Number of crossvalidation folds for lambda: 10
```

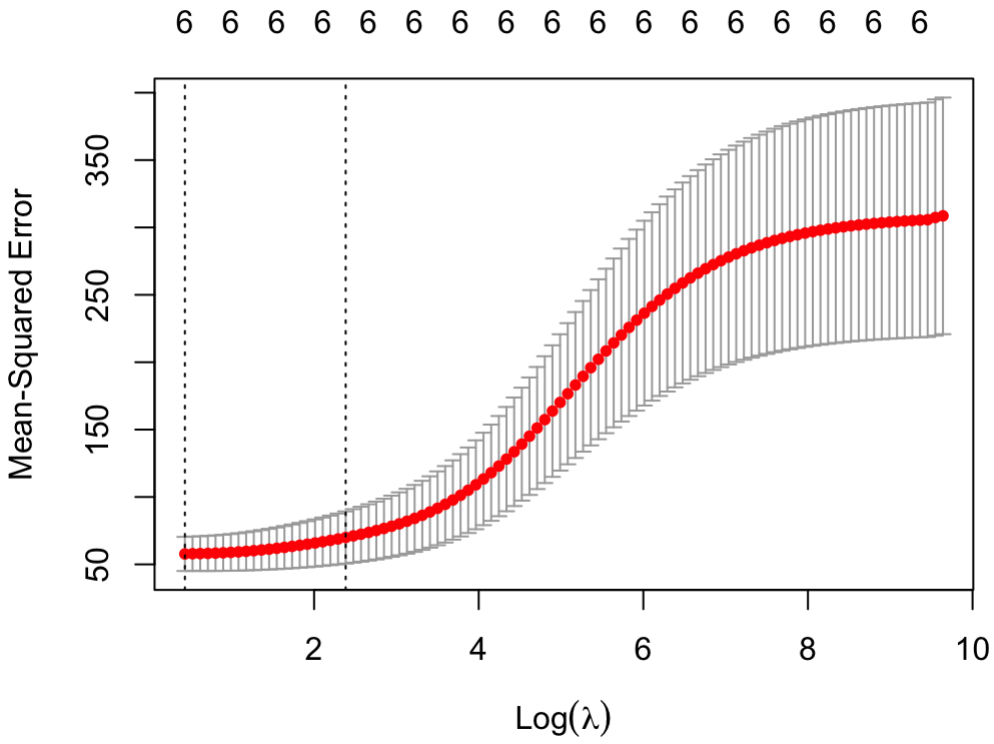
I created a minlossplot to assess whether an elastic net model is the most efficient and which alpha should be used when creating the model.

```
#plotting minlossplot to find best alpha
minlossplot(enet_mod,
            cv.type = "min")
```



This minlossplot shows that when alpha is 0, it has the lowest cross-validated error, and as alpha increases the cross-validated error increases as well. This suggests that we should use an alpha of 0 which is a ridge model. This means that all the predictors used are significant towards predicting total points. Because of this I decided to switch to a ridge model below.

```
#ridge model
ridge_mod <- cv.glmnet(points ~ avg_spi + avg_importance + avg_prob + avg_proj_score + avg_shot_
exp_goals + avg_nonshot_exp_goals,
                       data = league,
                       alpha = 0) #ridge

print(ridge_mod$lambda.min)
## [1] 1.537487
print(ridge_mod$lambda.1se)
## [1] 10.84667

plot(ridge_mod)
```




The plot above is the ridge_mod and the lambda.min is 2.68 while the lambda.1se is 17.27. The lambda.min is the lambda that minimizes error while lambda.1se is lambda.min plus one standard error.

```
#coefficients using lambda.min
coef(ridge_mod, s = "lambda.min")
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                                s1
## (Intercept)          -26.5164014
## avg_spi                0.5941634
## avg_importance        -0.1367512
## avg_prob              24.9410856
## avg_proj_score        13.1668801
## avg_shot_exp_goals    12.1893549
## avg_nonshot_exp_goals -6.1387618
```

Above are the printed coefficients for the ridge model. This can be interpreted as a 1 increase in avg_spi is correlated with a 0.54 increase in total points. It can be seen from the coefficients, the variables that have the largest impact on how a team does is projected score, score, and adjusted score. A team that is projected to score a goal is associated with a 11.40 points in total points. The average shot-based expected goals is also very significant in predicting the total points as a 1 increase in shot-based expected goals increased total points by 11.35. Interestingly a 1 increase in average non-shot based expected goals is correlated with a 3.82 decrease in total points.

```
#predictions using the ridge model for points
ridge_pred <- predict(ridge_mod, s = "lambda.min", newdata = league)

ridge_pred
##       lambda.min
##  [1,]   88.73368
##  [2,]   69.33645
##  [3,]   73.75987
##  [4,]   69.19885
##  [5,]   59.27730
##  [6,]   55.18830
##  [7,]   66.90952
##  [8,]   57.84064
##  [9,]   51.24755
## [10,]   52.46754
## [11,]   54.24280
## [12,]   49.76628
## [13,]   36.66878
## [14,]   48.47149
## [15,]   39.29993
## [16,]   34.49278
## [17,]   29.48945
## [18,]   34.60878
## [19,]   52.99442
## [20,]   41.59826

standings
```

| team<br><fct> | points<br><dbl> |
|---|---|
| Manchester City | 86 |
| Manchester United | 74 |
| Liverpool | 69 |
| Chelsea | 67 |
| Leicester City | 66 |
| West Ham United | 65 |
| Tottenham Hotspur | 62 |
| Arsenal | 61 |
| Leeds United | 59 |
| Everton | 59 |

1-10 of 20 rows Previous **1** 2 Next

As we can see from the predicted total points, our model is very accurate at predicting total points but this is not entirely reliable as there is data leakage. We are training the data on the same set as we are testing it on.

```
#find summary statistics for predictions
avg_pred = mean(ridge_pred)
min_pred = min(ridge_pred)
max_pred = max(ridge_pred)

print(avg_pred)
## [1] 53.27963
print(min_pred)
## [1] 29.48945
print(max_pred)
## [1] 88.73368
```

The average predicted total points is 53, the minimum predicted points is 30, and the maximum total points is 85. This fairly accurate to the actual statistics from last years season which can be seen below.

```
#summarize actual summary statistics from season
standings %>%
  summarize(avg_points = mean(points),
            min_points = min(points),
            max_points = max(points))
```

| avg_points<br><dbl> | min_points<br><dbl> | max_points<br><dbl> |
|---|---|---|

| avg_points <dbl> | min_points <dbl> | max_points <dbl> |
|---|---|---|
| 52.85 | 23 | 86 |

1 row

Compared to the actual summary statistics for last season we are very close. The average points is 52.85 compared to the predicted 53.22. The minimum points is 23 compared to the 29 that was predicted and the maximum points is 86 which is very close to the predicted maximum points of 85.61. In order to fully assess our model I will use it on the 2019-2020 season.

```
#creating new data frame based on 2019 Barclays Premier League season
matches2 <- read.csv(here("datasets", "spi_matches.csv"), stringsAsFactors = TRUE)

#cleaning matches data
matches_clean2 <- matches2 %>%
  mutate(team1 = as.factor(team1),
         team2 = as.factor(team2),
         league = as.factor(league)) %>%
  filter(season == 2019, league == "Barclays Premier League" ) %>%
  drop_na()

#cleaning and summarizing home team data
home_summary2 <- matches_clean2 %>%
  mutate(team = team1) %>%
  group_by(team) %>%
  summarize(avg_spi = mean(spi1),
            avg_importance = mean(importance1),
            avg_prob = mean(prob1),
            avg_proj_score = mean(proj_score1),
            avg_score = mean(score1),
            avg_adj_score = mean(adj_score1),
            avg_shot_exp_goals = mean(xg1),
            avg_nonshot_exp_goals = mean(nsxg1)) %>%
  arrange(desc(avg_spi))

#cleaning and summarizing away team data
away_summary2 <- matches_clean2 %>%
  mutate(team = team2) %>%
  group_by(team) %>%
  summarize(avg_spi = mean(spi2),
            avg_importance = mean(importance2),
            avg_prob = mean(prob2),
            avg_proj_score = mean(proj_score2),
            avg_score = mean(score2),
            avg_adj_score = mean(adj_score2),
            avg_shot_exp_goals = mean(xg2),
            avg_nonshot_exp_goals = mean(nsxg2)) %>%
  arrange(desc(avg_spi))

#joining away and home dataframes
team2 <- full_join(x = home_summary2,
                   y = away_summary2,
                   by = "team")

#cleaning team summary data
team_summary2 <- team2 %>%
  group_by(team) %>%
  summarize(avg_spi = mean(avg_spi.x, avg_spi.y),
            avg_importance = mean(avg_importance.x, avg_importance.y),
            avg_prob = mean(avg_prob.x, avg_prob.y),
            avg_proj_score = mean(avg_proj_score.x, avg_proj_score.y),
            avg_score = mean(avg_score.x, avg_score.y),
```

```
            avg_adj_score = mean(avg_adj_score.x, avg_adj_score.y),
            avg_shot_exp_goals = mean(avg_shot_exp_goals.x, avg_shot_exp_goals.y),
            avg_nonshot_exp_goals = mean(avg_nonshot_exp_goals.x, avg_nonshot_exp_goals.y)) %>%
  arrange(desc(avg_spi))

#team_summary2

standings2 <- read.csv(here("datasets", "soccer-standings_2019.csv"), stringsAsFactors = TRUE)

#cleaning standings data
standings2 <- standings2 %>% slice(-c(1)) %>%
  mutate(team = as.factor(X),
         points = as.numeric(as.character(X.1))) %>%
  select(team, points) %>%
  drop_na()

#standings2
#joining team and standings data
league2 <- full_join(x = standings2,
                y = team_summary2,
                by = "team")
```

```
#predictions for 2019 season
ridge_pred2 <- predict(ridge_mod, s = "lambda.min", newdata = league2)

ridge_pred2
##       lambda.min
##  [1,]   91.73381
##  [2,]   96.86866
##  [3,]   64.71896
##  [4,]   76.84321
##  [5,]   60.01632
##  [6,]   65.25261
##  [7,]   56.62342
##  [8,]   59.98630
##  [9,]   47.39107
## [10,]   50.95725
## [11,]   49.26463
## [12,]   57.38436
## [13,]   42.21527
## [14,]   45.04640
## [15,]   37.34972
## [16,]   44.00543
## [17,]   46.63979
## [18,]   33.00596
## [19,]   44.80670
## [20,]   39.88235

league2
```

| team <fct> | points <dbl> | avg_spi <dbl> | avg_importance <dbl> | avg_prob <dbl> | avg_proj_score <dbl> | avg |
|---|---|---|---|---|---|---|
| Liverpool | 99 | 92.75526 | 38.88947 | 0.7419632 | 2.462105 | 2.7 |
| Manchester City | 81 | 94.92895 | 32.05789 | 0.7916684 | 2.801053 | 3.0 |
| Manchester United | 66 | 82.79316 | 56.62105 | 0.5600158 | 1.820000 | 2.1 |
| Chelsea | 66 | 85.30000 | 69.83158 | 0.6379684 | 2.108947 | 1.5 |
| Leicester City | 62 | 80.73000 | 48.52105 | 0.5168526 | 1.693158 | 1.8 |
| Tottenham Hotspur | 59 | 80.79789 | 46.10000 | 0.5245105 | 1.842632 | 1.8 |
| Wolverhampton | 59 | 77.15895 | 33.88421 | 0.4506421 | 1.467368 | 1.4 |
| Arsenal | 56 | 77.78684 | 24.84737 | 0.4843947 | 1.740526 | 1.8 |
| Sheffield United | 54 | 69.33789 | 22.85789 | 0.3524421 | 1.210000 | 1.2 |
| Burnley | 54 | 69.71632 | 17.11053 | 0.3555316 | 1.290526 | 1.2 |

1-10 of 22 rows | 1-7 of 10 columns Previous **1** 2 3 Next

From looking at the data above it is clear to see that our model does okay on the 2019 season. We predicted Manchester City to win the league with around 96.87 points compared to their actual 81 points. Liverpool won the league in 2019 with 99 points but our model predicted they would come second with 91.73 points. From there our model is fairly accurate at predicting the order in which the teams will finish. Teams that are placed incorrectly are only a few spots away from their correct standing.

```
#cleaning predictions
pred_clean <- na.omit(ridge_pred2)

point_clean <- na.omit(league2["points"])

pred_point <- point_clean %>% mutate(pred = pred_clean)

#RMSE for predictions
rmse(pred_point$points, pred_point$pred)
## [1] 7.983314
```

The root mean squared error for the 2019 season is 7.98 which is fairly high considering most teams end with a total of points between 30 and 90. This 7.98 error can drastically alter a team's position in the league.

To predict this years season rankings we must create a new dataframe with the spi, importance, and projected scores for this season to predict total points.

```
#cleaning and summarizing data for 2021 Barclays Premier League season
matches3 <- read.csv(here("datasets", "spi_matches_latest.csv"), stringsAsFactors = TRUE)

#cleaning matches data
matches_clean3 <- matches3 %>%
  mutate(team1 = as.factor(team1),
         team2 = as.factor(team2),
         league = as.factor(league)) %>%
  filter(season == 2021, league == "Barclays Premier League" ) %>%
  drop_na()

#cleaning and summarizing home team data
home_summary3 <- matches_clean3 %>%
  mutate(team = team1) %>%
  group_by(team) %>%
  summarize(avg_spi = mean(spi1),
            avg_importance = mean(importance1),
            avg_prob = mean(prob1),
            avg_proj_score = mean(proj_score1),
            avg_score = mean(score1),
            avg_adj_score = mean(adj_score1),
            avg_shot_exp_goals = mean(xg1),
            avg_nonshot_exp_goals = mean(nsxg1)) %>%
  arrange(desc(avg_spi))

#cleaning and summarizing away team data
away_summary3 <- matches_clean3 %>%
  mutate(team = team2) %>%
  group_by(team) %>%
  summarize(avg_spi = mean(spi2),
            avg_importance = mean(importance2),
            avg_prob = mean(prob2),
            avg_proj_score = mean(proj_score2),
            avg_score = mean(score2),
            avg_adj_score = mean(adj_score2),
            avg_shot_exp_goals = mean(xg2),
            avg_nonshot_exp_goals = mean(nsxg2)) %>%
  arrange(desc(avg_spi))

#joining home and away team dataframes
team3 <- full_join(x = home_summary3,
                   y = away_summary3,
                   by = "team")

#cleaning team data
team_summary3 <- team3 %>%
  group_by(team) %>%
  summarize(avg_spi = mean(avg_spi.x, avg_spi.y),
            avg_importance = mean(avg_importance.x, avg_importance.y),
            avg_prob = mean(avg_prob.x, avg_prob.y),
            avg_proj_score = mean(avg_proj_score.x, avg_proj_score.y),
```

```
            avg_score = mean(avg_score.x, avg_score.y),
            avg_adj_score = mean(avg_adj_score.x, avg_adj_score.y),
            avg_shot_exp_goals = mean(avg_shot_exp_goals.x, avg_shot_exp_goals.y),
            avg_nonshot_exp_goals = mean(avg_nonshot_exp_goals.x, avg_nonshot_exp_goals.y)) %>%
  arrange(desc(avg_spi))

#team_summary2

standings3 <- read.csv(here("datasets", "soccer-standings.csv"), stringsAsFactors = TRUE)

#cleaning standings data
standings3 <- standings3 %>% slice(-c(1)) %>%
  mutate(team = as.factor(X),
         prev_year_points = as.numeric(as.character(X.1))) %>%
  select(team, prev_year_points) %>%
  drop_na()

#standings2

#joining team and standings data
league3 <- full_join(x = standings3,
                y = team_summary3,
                by = "team")

league3
```

| team<br><fct> | prev_year_points<br><dbl> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_pr |
|---|---|---|---|---|---|
| Manchester City | 86 | 93.11000 | 76.30000 | 0.7951429 | |
| Manchester United | 74 | 83.81167 | 58.53333 | 0.5075833 | |
| Liverpool | 69 | 90.18143 | 67.98571 | 0.6437857 | |
| Chelsea | 67 | 89.14714 | 58.90000 | 0.6571000 | |
| Leicester City | 66 | 74.22143 | 26.48571 | 0.3700429 | |
| West Ham United | 65 | 76.23571 | 42.94286 | 0.4239143 | |
| Tottenham Hotspur | 62 | 75.83000 | 41.05000 | 0.3897000 | |
| Arsenal | 61 | 76.87714 | 32.18571 | 0.4957143 | |
| Leeds United | 59 | 69.98286 | 26.97143 | 0.3786000 | |
| Everton | 59 | 72.78857 | 24.35714 | 0.4409571 | |

1-10 of 25 rows | 1-6 of 10 columns Previous **1** 2 3 Next

```
#predictions for 2021 season
ridge_pred3 <- predict(ridge_mod, s = "lambda.min", newdata = league3)

ridge_pred3
##       lambda.min
##  [1,]   86.72174
##  [2,]   60.79551
##  [3,]   83.32953
##  [4,]   76.04507
##  [5,]   53.38230
##  [6,]   54.58152
##  [7,]   52.73922
##  [8,]   64.17286
##  [9,]   48.61041
## [10,]   53.25385
## [11,]   45.18200
## [12,]   48.88038
## [13,]   46.21858
## [14,]   44.67084
## [15,]   45.39549
## [16,]   49.38312
## [17,]   45.25612
## [18,]   40.40968
## [19,]   31.39038
## [20,]   29.53984
```

According to our model, we predict that Manchester City will win the league with around 86.72 points, Liverpool to come 2nd with around 83.33, and Chelsea to come 3rd with 76.05. There is a 20 point predicted gap between 3rd place Chelsea and 4th place Manchester United. This year in the Barclays Premier League so far with 14 games played, Chelsea has 33 points in 1st, Manchester City has 32 points in 2nd, and Liverpool has 31 points in 3rd. There is a 9 point gap between 3rd place Liverpool and 4th place West Ham United. The league is very close at the top so there is a possibility our model is accurate. Our model was also accurate at predicting the gap between the top 3 teams and the rest of the league.

```
#summary statistics for predictions
avg_pred3 = mean(ridge_pred3)
min_pred3 = min(ridge_pred3)
max_pred3 = max(ridge_pred3)

print(avg_pred3)
## [1] 52.99792
print(min_pred3)
## [1] 29.53984
print(max_pred3)
## [1] 86.72174

#summary for previous year points
standings3 %>%
  summarize(avg_points = mean(prev_year_points),
            min_points = min(prev_year_points),
            max_points = max(prev_year_points))
```

| avg_points | min_points | max_points |
|---|---|---|
| <dbl> | <dbl> | <dbl> |
| 52.85 | 23 | 86 |

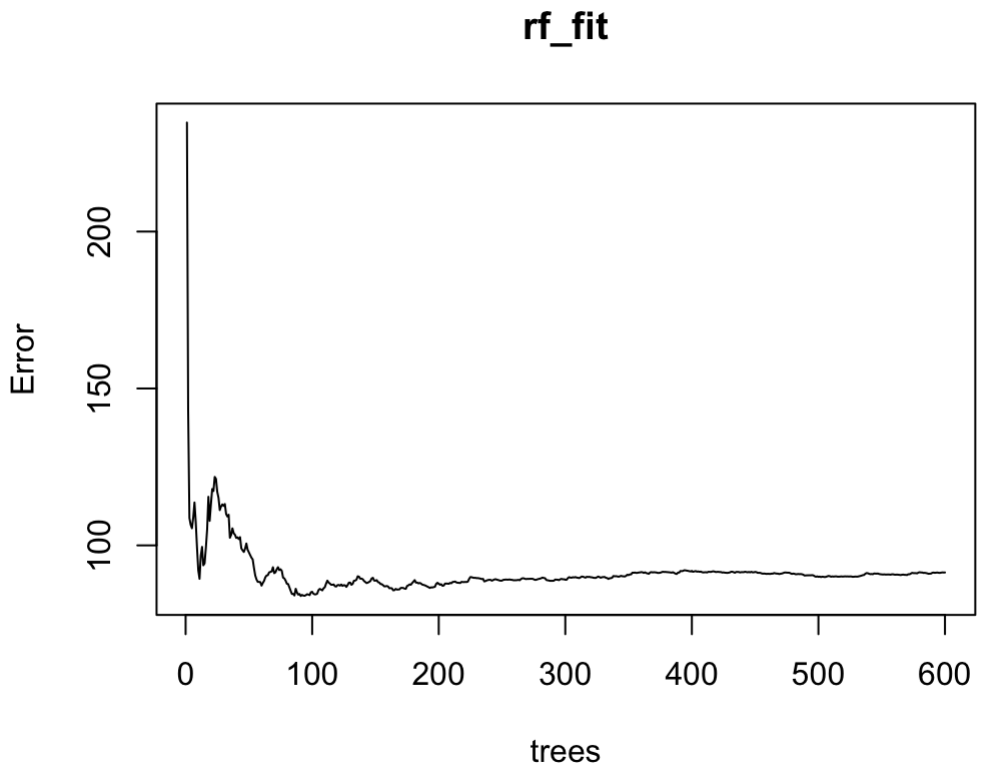
1 row

The average predicted total points for this season is 53.00 points, the minimum is 29.54 and the maximum is 86.72. These numbers are similar to the previous year points but we will see in the future if it is accurate at predicting the current season.

# Random Forest Model

I will use the cleaned standings and matches and put them into a random forest to predict how many trees to use.

```
library('randomForest')

#random forest model
rf_fit <- randomForest(points ~
                         avg_spi +  avg_adj_score + avg_importance,
                       data = league,
                       mtry = 3,
                       na.action = na.roughfix,
                       ntree = 600,
                       importance = TRUE)

print(rf_fit)
##
## Call:
##  randomForest(formula = points ~ avg_spi + avg_adj_score + avg_importance,      data = leagu
e, mtry = 3, ntree = 600, importance = TRUE,      na.action = na.roughfix)
##                Type of random forest: regression
##                      Number of trees: 600
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 91.36867
##                     % Var explained: 63.12
plot(rf_fit)
```




As you can see, the data suggests to use 300 trees in the random forest. Next, I am going to use the rf_fit and the data set for the current (2021/22) Premier League season to predict how many points each of the teams will end up with. The three variables I chose to run with this Random Forest (average SPI, average adjusted score, and

average importance of the games) account for 63.13% of the variance in points according to model.

```
#predictions for 2021 season
tree_pred <- predict(rf_fit, newdata = league3)

tree_pred
##        1        2        3        4        5        6        7
## 77.80894 66.65694 75.22897 75.01614 49.92236 61.54664 54.32714
##        8        9       10       11       12       13       14
## 59.83017 45.26569 52.50503 51.86608       NA 37.89094 51.16422
##       15       16       17       18       19       20       21
## 46.14664       NA 50.74350       NA       NA       NA 45.97633
##       22       23       24       25
## 48.24219 50.94458 47.48567 31.52547
```

Next I am comparing the tree_pred with the current Premier League season. The tree_pred is put in order of last seasons standings but has new predicted points underneath.

```
league3
```

| team<br><fct> | prev_year_points<br><dbl> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_pr |
|---|---|---|---|---|---|
| Manchester City | 86 | 93.11000 | 76.30000 | 0.7951429 | |
| Manchester United | 74 | 83.81167 | 58.53333 | 0.5075833 | |
| Liverpool | 69 | 90.18143 | 67.98571 | 0.6437857 | |
| Chelsea | 67 | 89.14714 | 58.90000 | 0.6571000 | |
| Leicester City | 66 | 74.22143 | 26.48571 | 0.3700429 | |
| West Ham United | 65 | 76.23571 | 42.94286 | 0.4239143 | |
| Tottenham Hotspur | 62 | 75.83000 | 41.05000 | 0.3897000 | |
| Arsenal | 61 | 76.87714 | 32.18571 | 0.4957143 | |
| Leeds United | 59 | 69.98286 | 26.97143 | 0.3786000 | |
| Everton | 59 | 72.78857 | 24.35714 | 0.4409571 | |

1-10 of 25 rows | 1-6 of 10 columns Previous **1** 2 3 Next

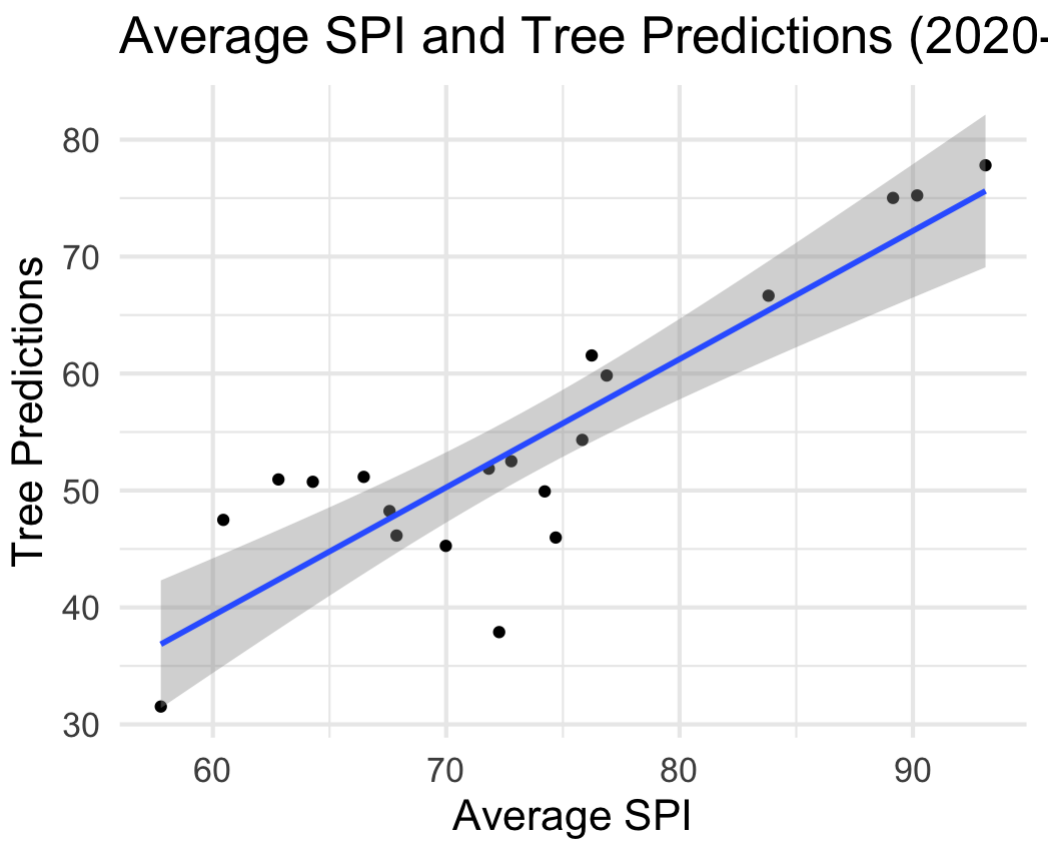
The data shows that Manchester City are expected to win the premier league season with 78.14639 expected points. Liverpool is favorite to get second with 75.77314, and closely followed by Chelsea who are expected to finish with 75.64878. This is represented in the original SPI as well because Manchester City has an SPI of 93.11000, followed by Liverpool's 90.18143, and then Chelsea's 89.14714.

```
#adding predictions
league3 <- league3 %>%
        mutate(tree_pred = tree_pred)

league3
```

| team<br><fct> | prev_year_points<br><dbl> | avg_spi<br><dbl> | avg_importance<br><dbl> | avg_prob<br><dbl> | avg_pr |
|---|---|---|---|---|---|
| Manchester City | 86 | 93.11000 | 76.30000 | 0.7951429 | |
| Manchester United | 74 | 83.81167 | 58.53333 | 0.5075833 | |
| Liverpool | 69 | 90.18143 | 67.98571 | 0.6437857 | |
| Chelsea | 67 | 89.14714 | 58.90000 | 0.6571000 | |
| Leicester City | 66 | 74.22143 | 26.48571 | 0.3700429 | |
| West Ham United | 65 | 76.23571 | 42.94286 | 0.4239143 | |
| Tottenham Hotspur | 62 | 75.83000 | 41.05000 | 0.3897000 | |
| Arsenal | 61 | 76.87714 | 32.18571 | 0.4957143 | |
| Leeds United | 59 | 69.98286 | 26.97143 | 0.3786000 | |
| Everton | 59 | 72.78857 | 24.35714 | 0.4409571 | |

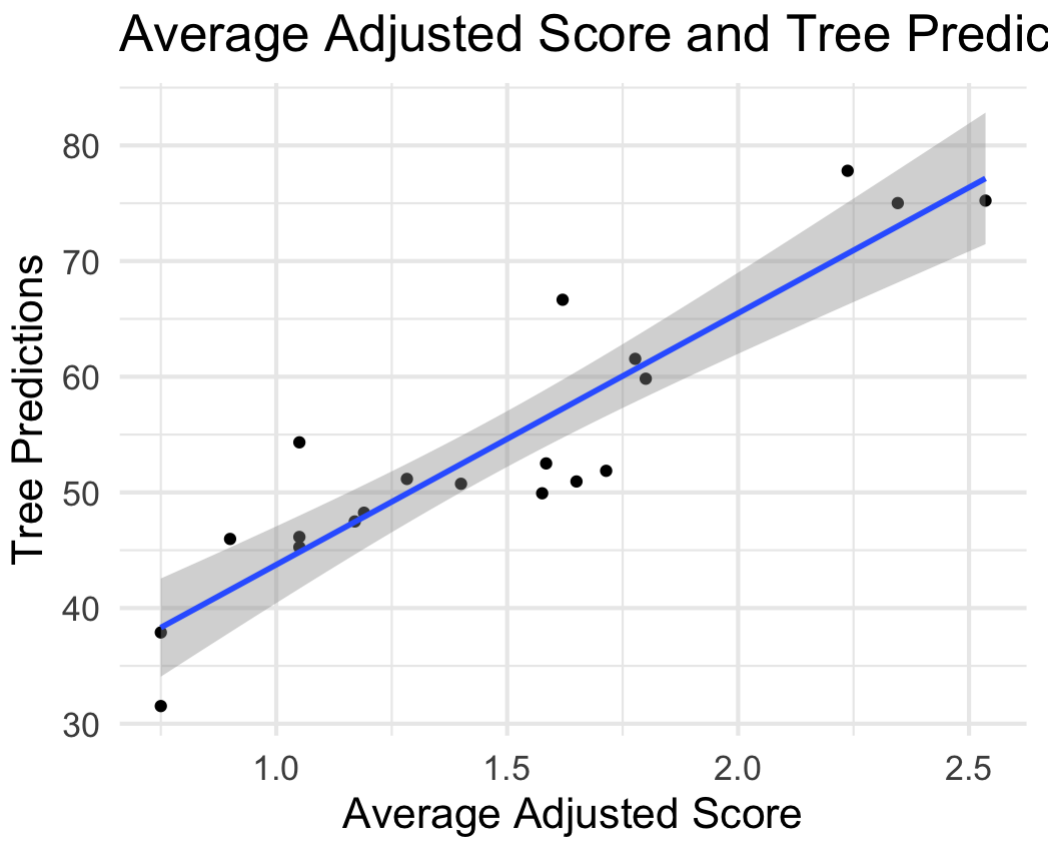
1-10 of 25 rows | 1-6 of 11 columns Previous **1** 2 3 Next

```
#plotting average spi against predictions
ggplot(league3, aes(x = avg_spi, y = tree_pred)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average SPI", y = "Tree Predictions") +
  ggtitle("Average SPI and Tree Predictions (2020-21 Season)")
```



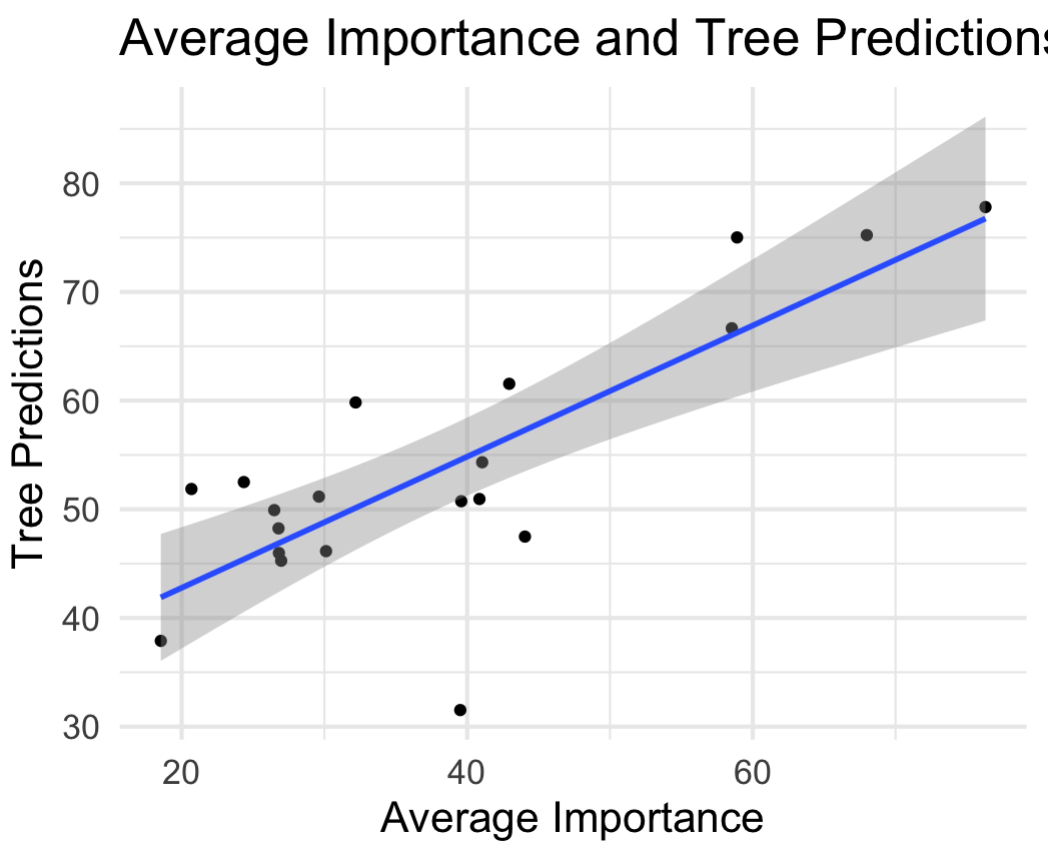

According to our model, the higher the SPI, the more points they are expected to finish with in the season. Manchester City are predicted to have around 78 points, while Liverpool are expected to have 77 points, and Chelsea are set to finish with 76 points.

```
#plotting average adjusted score against tree predictions
ggplot(league3, aes(x = avg_adj_score, y = tree_pred)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average Adjusted Score", y = "Tree Predictions") +
  ggtitle("Average Adjusted Score and Tree Predictions (2020-21 Season)")
```




This ggplot predicts that the three teams at the top of the table with the most points will also have the highest adjusted average goals. Manchester City are again predicted to win the Premier League with a lower adjusted score than Liverpool and Chelsea. The lower adjusted score is due to goals that are insignificant to the final score of the games. Manchester City are predicted to have around 78 points, while Liverpool and Chelsea are set to have about 76 points each. Manchester City are predicted to have around 78 points, while Liverpool are expected to have 77 points, and Chelsea are set to finish with 76 points.

```
#plotting average importance against tree predictions
ggplot(league3, aes(x = avg_importance, y = tree_pred)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_minimal(base_size = 16) +
  labs(x = "Average Importance", y = "Tree Predictions") +
  ggtitle("Average Importance and Tree Predictions (2020-21 Season)")
```




This model predicts that Manchester City will finish top of the premier league season having won the most important games. This means that they are expected to win more of the games against close rivals such as Liverpool and Chelsea. Liverpool and Chelsea, are however, right behind them in points and important games won. Manchester City are predicted to have around 78 points, while Liverpool and Chelsea are set to have about 76 points each.

```
#tree_pred
#summary statistics
avg_pred4 = mean(tree_pred, na.rm = TRUE) #removing nulls
min_pred4 = min(tree_pred, na.rm = TRUE) #removing nulls
max_pred4 = max(tree_pred, na.rm = TRUE) #removing nulls

print(avg_pred4)
## [1] 54.00468
print(min_pred4)
## [1] 31.52547
print(max_pred4)
## [1] 77.80894
```

The average point prediction for this season is 54, the minimum is 31.53 and the maximum is 77.81 points. The minimum points is predicted to be by Norwich and the maximum is predicted to be achieved by Manchester City.

```
#predictions for 2019 season
tree_pred2 <- predict(rf_fit, newdata = league2)

pred_clean2 <- na.omit(tree_pred2)

point_clean2 <- na.omit(league2["points"])

pred_point2 <- point_clean %>% mutate(pred = pred_clean2)

#RMSE for model on 2019 season
rmse(pred_point2$points, pred_point2$pred)
## [1] 10.98338
```

The RMSE for the 2019 season for the random forest model is 10.98 which is significantly high. As mentioned before, a team usually finishes between 30 and 90 points and 10.98 error can greatly affect where a team finishes.

# Conclusion

Below are the full predictions for this current season using the ridge model. We took away the teams that were in the bottom 3 last season and were relegated as they are no longer in the league.

1. Man City
2. Liverpool
3. Chelsea
4. Arsenal
5. Man United
6. West Ham United
7. Leicester City
8. Everton
9. Tottenham
10. Burnley
11. Newcastle United
12. Leeds United
13. Wolverhampton
14. Southampton
15. Aston Villa
16. Crystal Palace
17. Brentford
18. Watford
19. Brighton And Hove Albion
20. Norwich City

Below is the prediction for the random forest model for the current season.

1. Manchester City
2. Liverpool
3. Chelsea
4. Manchester United
5. West Ham
6. Arsenal

7. Spurs
8. Everton
9. Aston Villa
10. Newcastle
11. Burnley
12. Crystal Palace
13. Leicester
14. Brentford
15. Watford
16. Brighton
17. Southampton
18. Leeds
19. Wolves
20. Norwich

Our models have similar predictions such as the top 3 predictions and who we predicted to be last. We will see in the future whether one of us was more accurate than the other. The RMSE for the ridge model was slightly lower than the random forest model on the 2019 season. This suggests the ridge model is slightly more accurate.