# Final

2024-06-04

## Contents

# Introduction

## Goal

The purpose of this project is to develop a model that will predict how impactful a defensive tackle (a position in football) is to winning by looking at their counting stats like tackles and assists, and to see which stats are the most important to seeing a defensive tackle's impact.

## Football?

Football is a popular contact sport in America. In this sport, the goal is to score more points than your opponent, which is done through scoring touchdowns (which is when a player reaches the end of a field) or a field goal (where a kicker kicks a ball through two uprights). While this seems simple enough, football is a very complicated sport. In total there are 24 players on the field at a given time (12 on offense and 12 on defense), and the offense has four attempts to cross over an imaginary line that is 10 yards away from the starting position. After crossing this line, the offense is given another four tries to cross another imaginary line that is 10 yards away. It is the defense's job to stop the offense from advancing, so the offense has to kick the ball and give the possession to the opposing team. There are multiple positions on the defense that stop the ball from advancing, such as the cornerback that will follow a reciever and make sure he isn't able to catch the ball, or an edge rusher that will do his best to tackle the quarterback before he is able to trow the ball. The role of a defensive tackle, however, is a little more complicated.

## Why Defensive Tackles?

Defensive tackles are an important defensive position. The scheme of football positions can be very confusing at times, but the main idea is that there are players who play at the line of scrimmage, and players that play outside. While the nuances of the role of a player on the line of scrimmage are often complicated, the two main goals are: trying to tackle the quarterback on a passing play and tackling the running back on a running play. There are normally two types of players on the line of scrimmage: defensive ends and defensive tackles. Defensive ends are at the edge of the players in the line of scrimmage while defensive tackles are in the middle. Due to their positions on the field, defensive ends are given more free reign to try to go to the quarterback and have to worry less about stopping the run, which makes grading them easy. Defensive tackles however, have to worry about stopping the run and "setting the line", as well as presuring the quarterback, all of which makes their jobs a lot more nuanced than a defensive end. Due to this nuance, it makes grading defensive tackles a lot harder than grading defensive ends, which is the goal of this project.

# Setting Up

## Loading the Packages

Here we are loading all the packages that we are going to use.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 1.2.0 --
## v broom        1.0.5      v rsample      1.2.1
## v dials        1.2.1      v tune         1.2.0
## v infer        1.0.7      v workflows    1.1.4
## v modeldata    1.3.0      v workflowsets 1.1.0
## v parsnip      1.2.1      v yardstick    1.3.1
## v recipes      1.0.10
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
library(dplyr)
library(rsample)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(broom)
library(ggplot2)
```

## Loading the Data

The dataset we are using was divided into different sub-datasets, so we are downloading four of the sub-datasets that apply to my project: games, players, plays, and tackles data. This data was downloaded from Kaggle from the NFL Data Bowl 2024 page, which was a competition to see who can develop the most interesting models using NFL data.

```
games_df <- read_csv("games.csv")
```

```
## Rows: 136 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): gameDate, homeTeamAbbr, visitorTeamAbbr
## dbl  (5): gameId, season, week, homeFinalScore, visitorFinalScore
## time (1): gameTimeEastern
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
players_df <- read_csv("players.csv")
```

```
## Rows: 1683 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): height, birthDate, collegeName, position, displayName
## dbl (2): nflId, weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
plays_df <- read_csv("plays.csv")
```

```
## Rows: 12486 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (10): ballCarrierDisplayName, playDescription, possessionTeam, defensiv...
## dbl  (24): gameId, playId, ballCarrierId, quarter, down, yardsToGo, yardline...
## time  (1): gameClock
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tackles_df <- read_csv("tackles.csv")
```

```
## Rows: 17426 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): gameId, playId, nflId, tackle, assist, forcedFumble, pff_missedTackle
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

# Exploring the Data

## Table Heads

Here we are looking at the data. The games dataset has information about the games, like the week the game was on and the teams that were playing. The players dataset has information about all of the players like their weights and heights. The plays dataframe has information about each play that happened in every game. This is the most important dataset, as it has information that is vital like expectedPointsAdded, which keeps track of the points that were expected to gain or lose from the offensive team after the play ended, and homeTeamWinProbabilityAdded/awayTeamWinProbabilityWinAdded, which keeps track of the probability of either team winning after the play ends. These are the two statistics that I will use to see how effective a defensive tackle is, as it is a defensive player's job to lower the opposing side's expected points, as well as contribute the most to their team's chances of winning. Lastly, the tackles dataset keeps track of what players did what during each play, like if the player performed a tackle or assist. In terms of contextual stats, I am interested in offenseFormation and defendersInTheBox. These are two stats that are outside of the defensive tackle's control, but I believe can affect their production. Certain offense formations like shotgun are built for passing, which means that the defensive tackles won't have to worry about containing the run. For defenders in the box, having more people at the line of scrimmage make it a lot easier for the team to stop the run. This may affect aspects like the defensive tackle's job in the play, or the offense's inclincation to pass.

```
head(games_df)
```

```
## # A tibble: 6 x 9
```

```
##       gameId season  week gameDate  gameTimeEastern homeTeamAbbr visitorTeamAbbr
##        <dbl>  <dbl> <dbl> <chr>     <time>          <chr>        <chr>
## 1 2022090800   2022     1 09/08/20~ 20:20           LA           BUF
## 2 2022091100   2022     1 09/11/20~ 13:00           ATL          NO
## 3 2022091101   2022     1 09/11/20~ 13:00           CAR          CLE
## 4 2022091102   2022     1 09/11/20~ 13:00           CHI          SF
## 5 2022091103   2022     1 09/11/20~ 13:00           CIN          PIT
## 6 2022091104   2022     1 09/11/20~ 13:00           DET          PHI
## # i 2 more variables: homeFinalScore <dbl>, visitorFinalScore <dbl>
```

```
head(players_df)
```

```
## # A tibble: 6 x 7
##   nflId height weight birthDate  collegeName    position displayName
##   <dbl> <chr>   <dbl> <chr>      <chr>          <chr>    <chr>
## 1 25511 6-4       225 1977-08-03 Michigan       QB       Tom Brady
## 2 29550 6-4       328 1982-01-22 Arkansas       T        Jason Peters
## 3 29851 6-2       225 1983-12-02 California     QB       Aaron Rodgers
## 4 30842 6-6       267 1984-05-19 UCLA           TE       Marcedes Lewis
## 5 33084 6-4       217 1985-05-17 Boston College QB       Matt Ryan
## 6 33099 6-6       245 1985-01-16 Delaware       QB       Joe Flacco
```

```
head(plays_df)
```

```
## # A tibble: 6 x 35
##       gameId playId ballCarrierId ballCarrierDisplayName playDescription quarter
##        <dbl>  <dbl>         <dbl> <chr>                  <chr>             <dbl>
## 1 2022100908   3537         48723 Parker Hesse           (7:52) (Shotgu~       4
## 2 2022091103   3126         52457 Chase Claypool         (7:38) (Shotgu~       4
## 3 2022091111   1148         42547 Darren Waller          (8:57) D.Carr ~       2
## 4 2022100212   2007         46461 Mike Boone             (13:12) M.Boon~       3
## 5 2022091900   1372         47857 Devin Singletary       (8:33) D.Singl~       2
## 6 2022103001   2165         54616 Tyler Allgeier         (10:14) (Shotg~       3
## # i 29 more variables: down <dbl>, yardsToGo <dbl>, possessionTeam <chr>,
## #   defensiveTeam <chr>, yardlineSide <chr>, yardlineNumber <dbl>,
## #   gameClock <time>, preSnapHomeScore <dbl>, preSnapVisitorScore <dbl>,
## #   passResult <chr>, passLength <dbl>, penaltyYards <dbl>,
## #   prePenaltyPlayResult <dbl>, playResult <dbl>, playNullifiedByPenalty <chr>,
## #   absoluteYardlineNumber <dbl>, offenseFormation <chr>,
## #   defendersInTheBox <dbl>, passProbability <dbl>, ...
```

```
head(tackles_df)
```

```
## # A tibble: 6 x 7
##       gameId playId nflId tackle assist forcedFumble pff_missedTackle
##        <dbl>  <dbl> <dbl>  <dbl>  <dbl>        <dbl>            <dbl>
## 1 2022090800    101 42816      1      0            0                0
## 2 2022090800    393 46232      1      0            0                0
## 3 2022090800    486 40166      1      0            0                0
## 4 2022090800    646 47939      1      0            0                0
## 5 2022090800    818 40107      1      0            0                0
## 6 2022090800   1286 44976      1      0            0                0
```

## Checking for Missing Data

As missing data can make modelling the data harder, we should check to see if there is any missing data. As you can see, there is no missing data in the games dataframe.

```
colSums(is.na(games_df))
```

```
##            gameId            season              week          gameDate
##                 0                 0                 0                 0
##   gameTimeEastern      homeTeamAbbr   visitorTeamAbbr    homeFinalScore
##                 0                 0                 0                 0
## visitorFinalScore
##                 0
```

There is missing data here, but we are not interested in the birthDates of players so it doesn't really matter.

```
colSums(is.na(players_df))
```

```
##       nflId      height      weight   birthDate collegeName    position
##           0           0           0         479           0           0
## displayName
##           0
```

There is also missing data here, but again it isn't that important. We aren't interested in yardlineSide, passResult or passLength. Looking at the penaltyYards and foulNFLID1/foulNFLID2, the documentation said that there would be missing values of there wasn't a penalty, so we can ignore this (this will have to be handled in the future however). The variables that we are interested in: offensiveFormation, defendersInTheBox, and expectedPointsAdded having some missing values is an issue, but the small amount of missing values (4, 5, and 1 respectively) make it so we can just ignore these rows.

```
colSums(is.na(plays_df))
```

```
##                          gameId                          playId
##                               0                               0
##                     ballCarrierId           ballCarrierDisplayName
##                               0                               0
##                   playDescription                         quarter
##                               0                               0
##                              down                       yardsToGo
##                               0                               0
##                    possessionTeam                    defensiveTeam
##                               0                               0
##                      yardlineSide                  yardlineNumber
##                               0                               0
##                         gameClock                 preSnapHomeScore
##                               0                               0
##               preSnapVisitorScore                      passResult
##                               0                            6381
##                        passLength                    penaltyYards
##                               0                           11871
##              prePenaltyPlayResult                      playResult
##                               0                               0
##           playNullifiedByPenalty           absoluteYardlineNumber
##                               0                               0
##                 offenseFormation                defendersInTheBox
##                               4                               5
##                  passProbability    preSnapHomeTeamWinProbability
##                               0                               0
```

```
## preSnapVisitorTeamWinProbability      homeTeamWinProbabilityAdded
##                                 0                                0
##     visitorTeamWinProbilityAdded                   expectedPoints
##                                 0                                0
##              expectedPointsAdded                         foulName1
##                                 1                             11894
##                         foulName2                       foulNFLId1
##                             12461                             11894
##                        foulNFLId2
##                             12461
```

Lastly, there is no missing data here so we should be fine.

```
colSums(is.na(tackles_df))
```

```
##           gameId           playId            nflId           tackle
##                0                0                0                0
##           assist     forcedFumble pff_missedTackle
##                0                0                0
```

## Tidying the Raw Data

First we have to take care of the player heights. Right now they are listed in the format "6-4", so the easy fix for this is to change the height to be measured in inches.

```
players_df$height = as.numeric(substring(players_df$height, 0, 1)) * 12. + as.numeric(substring(players
```

Next we have to get only defensive tackles. For this case, we are considering nose tackles defensive tackles as well, since they perform the same role and only exist on certain teams based off different defensive schemes. Here we are getting all the players that fit our criteria, then merging the tackles dataframe by nflID so we get the defensive stats for each defensive tackle.

```
defensive_tackles_df <- players_df %>% filter(position == 'DT' | position == 'NT')
dt_tackles_df <- tackles_df[tackles_df$nflId %in% defensive_tackles_df$nflId,]
dt_info_df <- inner_join(dt_tackles_df, defensive_tackles_df, by = 'nflId')
```

Now we are merging theplays and games dataframe by gameId, so we get the team of the player. This is important because we have to know if the defensive team is the home team or away team, which is important because there is a stat for home teams or away teams specifically. We then create a new stat called teamProbabilityWinAdded, which will be the defensive team's probability win added because that is the only stat we care about.

```
plays_teams_df <- inner_join(plays_df, games_df, by = c('gameId'))
plays_teams_df <- plays_teams_df %>% mutate(teamProbabilityWinAdded = ifelse(defensiveTeam == homeTeamA
```

Now we merge the two merged dataframes by gameId and playID because the documentation had said that playID was not unique across games. This means that we have every player's information, as well as the information about their plays and games. Since we are considering fouls, we create a new stat called "didFoul" and "penaltyYards", which measure if the player did in fact foul and what the resulting lost yards were from said foul. We then remove all the variables we don't need, which means we are left with gameId, playId, nflId, tackle, assist, forcedFumble, pff_missedTackle, height, weight, offenseFormation, height, weight, offenseFormation (offenses run different formations, which may affect the production of a defensive tackle), defendersInTheBox (the number of people at the line of scrimmage), expectedPointsAdded, teamProbabilityWinAdded, didFoul, and penaltyYardsAtFault. Next we set the offenseFormation stat as a factor and drop all the missing value rows.

```
dt_full_df <- inner_join(dt_info_df, plays_teams_df, by = c('gameId', 'playId'))
dt_full_df <- dt_full_df %>% mutate(didFoul = ifelse(!is.na(foulNFLId1) & foulNFLId1 == nflId | !is.na(
dt_full_df <- dt_full_df %>% mutate(penaltyYardsAtFault = ifelse(didFoul == 1, penaltyYards, 0))

dt_full_df <- dt_full_df %>% select(-c(season, week, gameDate, gameTimeEastern, homeTeamAbbr, visitorTea

dt_full_df$offenseFormation <- as.factor(dt_full_df$offenseFormation)
dt_full_df <- dt_full_df %>% drop_na()
```

# Exploratory Data Analysis

## Introductory Analysis

Now we are looking at data to see if we can come up with some conjectures. Looking at the head of the table here, it seems that we have a table of only the releveant variables.

```
head(dt_full_df)
```

```
## # A tibble: 6 x 15
##    gameId playId nflId tackle assist forcedFumble pff_missedTackle height weight
##     <dbl>  <dbl> <dbl>  <dbl>  <dbl>        <dbl>            <dbl>  <dbl>  <dbl>
## 1  2.02e9    393 46232      1      0            0                0     75    308
## 2  2.02e9    228 43356      1      0            0                0     74    308
## 3  2.02e9   1874 47796      1      0            0                0     76    315
## 4  2.02e9   2219 52479      1      0            0                0     75    300
## 5  2.02e9   1539 52539      1      0            0                0     74    308
## 6  2.02e9   3821 52522      1      0            0                0     77    335
## # i 6 more variables: offenseFormation <fct>, defendersInTheBox <dbl>,
## #   expectedPointsAdded <dbl>, teamProbabilityWinAdded <dbl>, didFoul <dbl>,
## #   penaltyYardsAtFault <dbl>
```

Here are some normal statistics about all the data. Nothing here really stands out, so we have to look at more.

```
summary(dt_full_df)
```

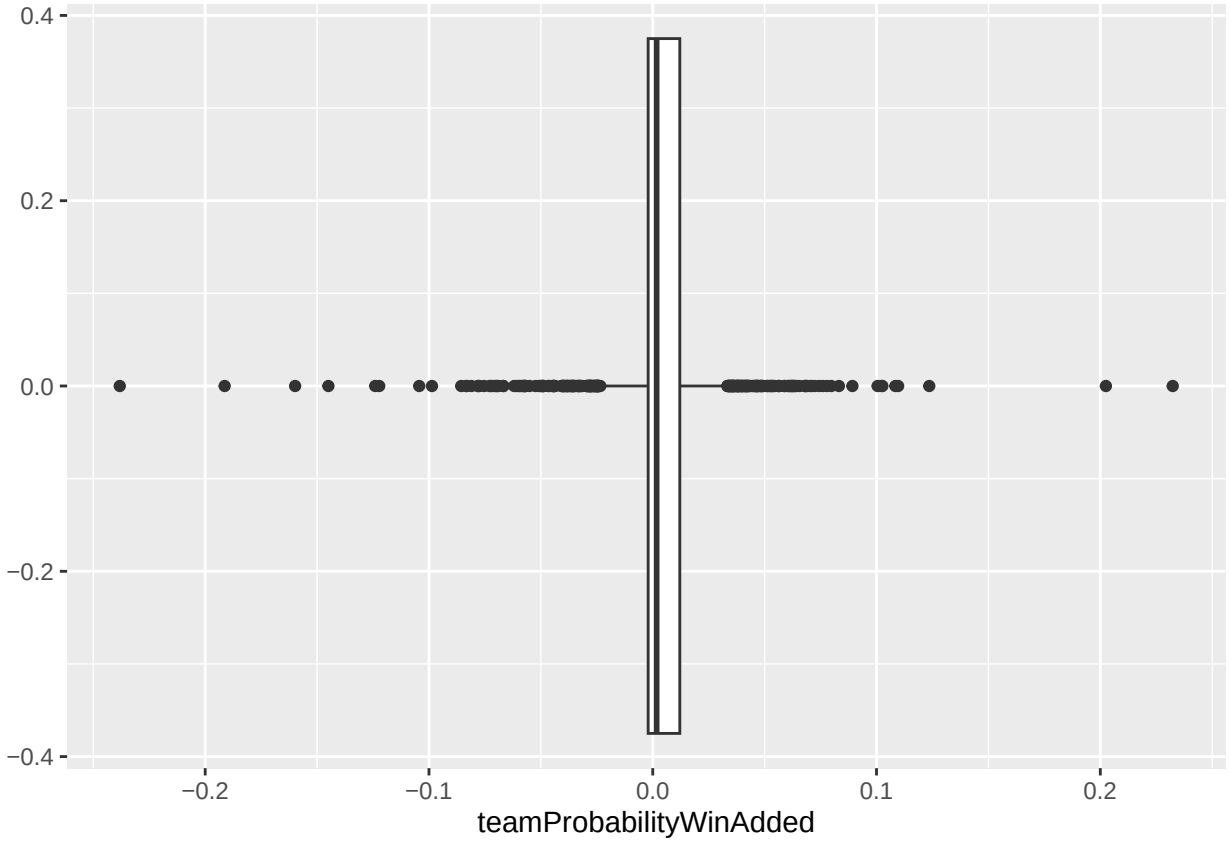
```
##      gameId              playId           nflId           tackle
##  Min.   :2.022e+09   Min.   :  54.0   Min.   :35449   Min.   :0.000
##  1st Qu.:2.022e+09   1st Qu.: 931.2   1st Qu.:43335   1st Qu.:0.000
##  Median :2.022e+09   Median :2067.0   Median :46144   Median :0.000
##  Mean   :2.022e+09   Mean   :2003.3   Mean   :46632   Mean   :0.452
##  3rd Qu.:2.022e+09   3rd Qu.:3039.2   3rd Qu.:48770   3rd Qu.:1.000
##  Max.   :2.022e+09   Max.   :4905.0   Max.   :55241   Max.   :1.000
##
##      assist        forcedFumble      pff_missedTackle     height
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.0000   Min.   :72.00
##  1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:75.00
##  Median :0.0000   Median :0.000000   Median :0.0000   Median :75.00
##  Mean   :0.4464   Mean   :0.003763   Mean   :0.1049   Mean   :75.34
##  3rd Qu.:1.0000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:76.00
##  Max.   :1.0000   Max.   :1.000000   Max.   :1.0000   Max.   :79.00
##
##      weight        offenseFormation defendersInTheBox expectedPointsAdded
##  Min.   :255.0   EMPTY     : 39     Min.   : 3.000    Min.   :-9.9862
##  1st Qu.:300.0   I_FORM    :237     1st Qu.: 6.000    1st Qu.:-0.5634
```

```
##  Median :310.0   JUMBO     : 28     Median : 7.000    Median :-0.3054
##  Mean   :310.8   PISTOL    :121     Mean   : 6.745    Mean   :-0.2308
##  3rd Qu.:318.0   SHOTGUN   :810     3rd Qu.: 7.000    3rd Qu.: 0.1074
##  Max.   :379.0   SINGLEBACK:866     Max.   :11.000    Max.   : 5.2693
##                  WILDCAT   : 25
##  teamProbabilityWinAdded    didFoul         penaltyYardsAtFault
##  Min.   :-0.238169       Min.   :0.000000   Min.   : 0.00000
##  1st Qu.:-0.002056       1st Qu.:0.000000   1st Qu.: 0.00000
##  Median : 0.001739       Median :0.000000   Median : 0.00000
##  Mean   : 0.003933       Mean   :0.001881   Mean   : 0.01881
##  3rd Qu.: 0.012088       3rd Qu.:0.000000   3rd Qu.: 0.00000
##  Max.   : 0.232366       Max.   :1.000000   Max.   :15.00000
##
```

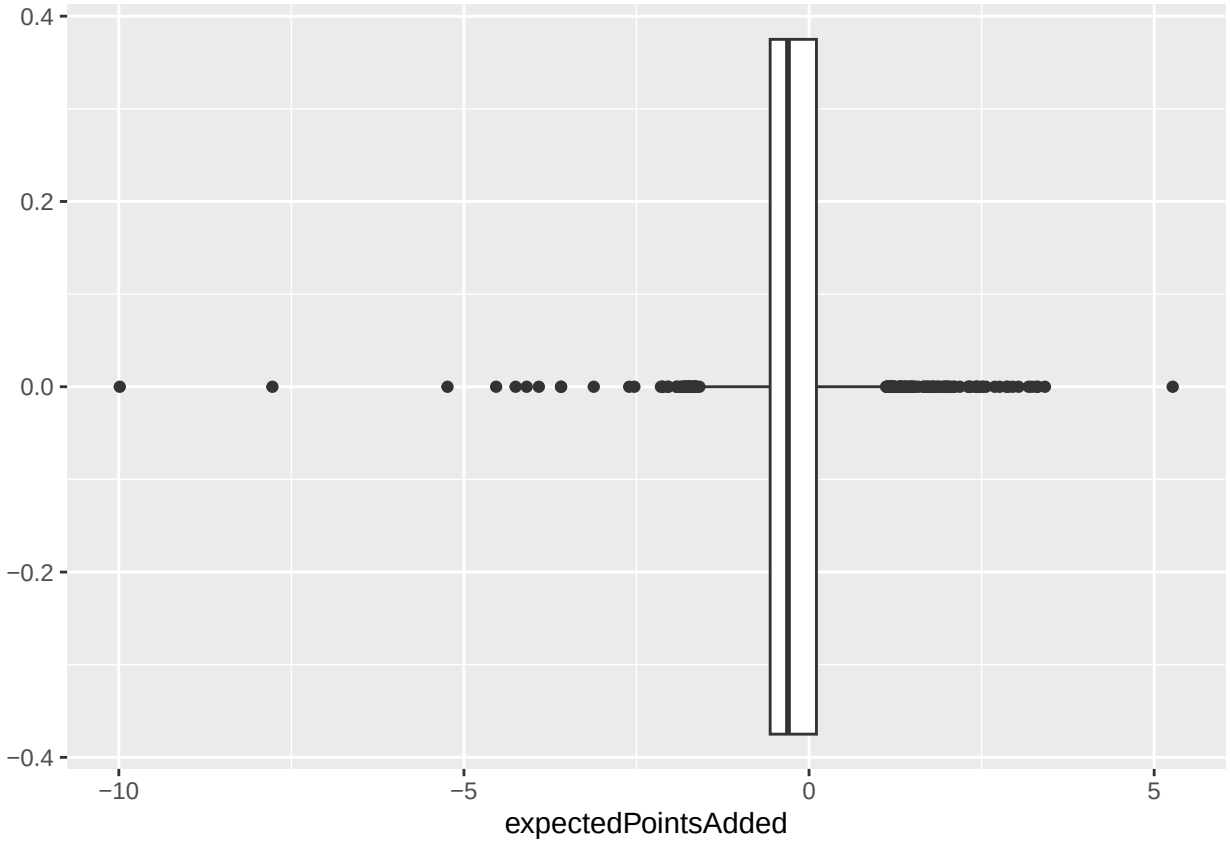
## EDA Visuals

From here, we see that expectedPointsAdded ranges from around -10 to 5, and teamProbaiblityWinAdded ranges from (-.23 to .23). This is important for when we use RMSE to measure how effective the models are.

```
ggplot(dt_full_df, aes(teamProbabilityWinAdded)) + geom_boxplot()
```



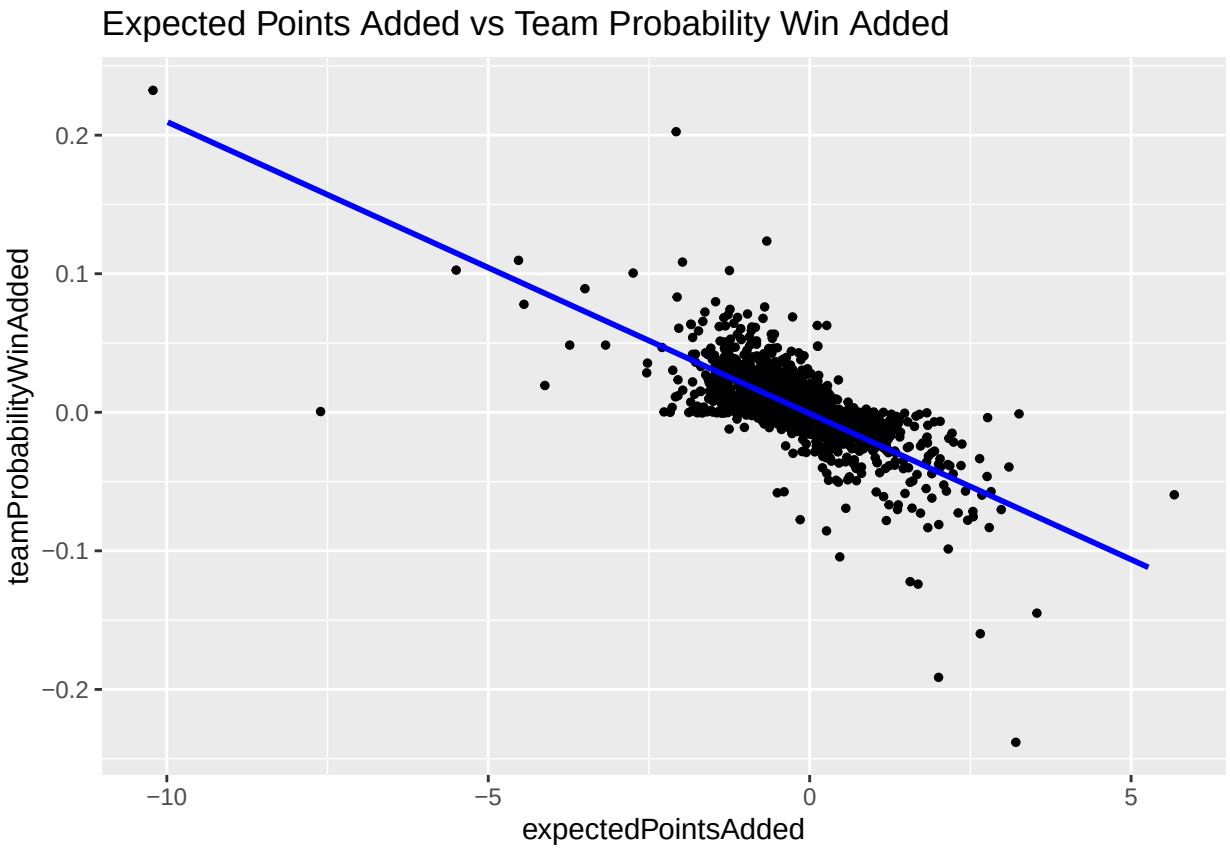

```
ggplot(dt_full_df, aes(expectedPointsAdded)) + geom_boxplot()
```




Since I plan on using both statistics to measure a player's impact, we should make sure that they are correlated in some way. From the graph below, we can see that they are negatively correlated, which makes sense because a defender's team probability to win should increase if the offensive team's expected points decrease. While these statistics are correlated, it is not direct. Due to this, I will be using distinct models to measure the effect of counting stats on both seperately.

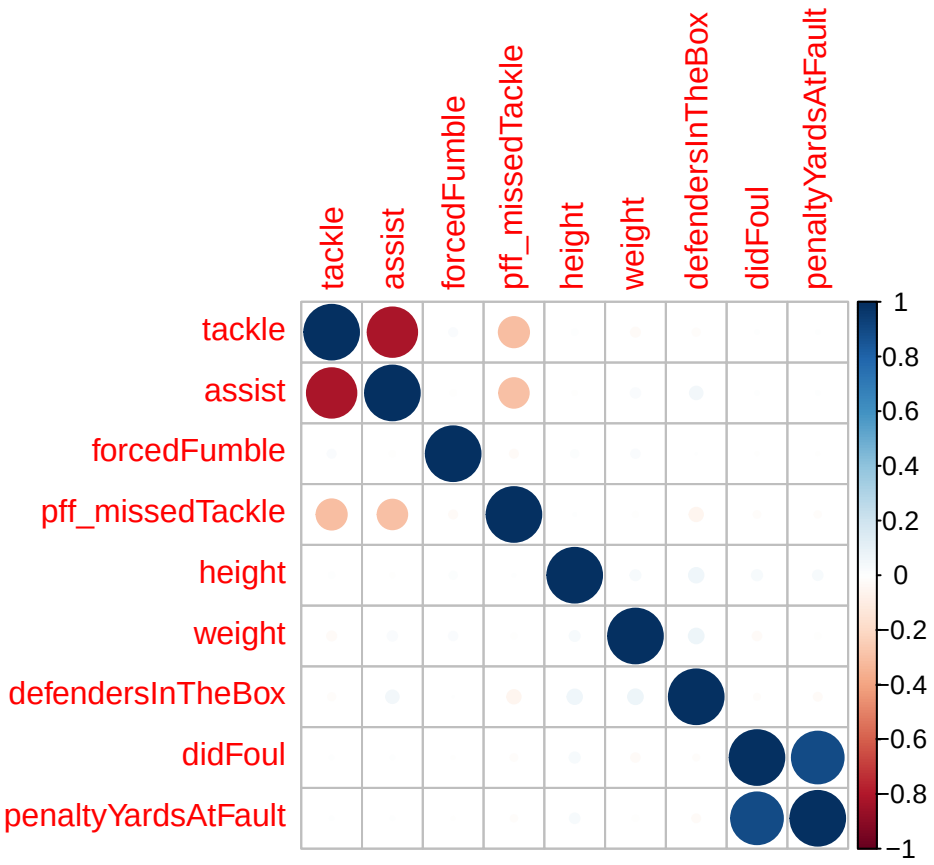
```
dt_full_df %>%
  ggplot(aes(x=expectedPointsAdded, y=teamProbabilityWinAdded)) +
  geom_jitter(width = 0.5, size = 1) +
  geom_smooth(method = "lm", se =F, col="blue") +
  labs(title = "Expected Points Added vs Team Probability Win Added")
```

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




Here is the correlation plot of all the predictors. Suprisingly, very few predictors are correlated to each other at all. However, it makes sense because players normally only record one counting stat for each play. Tackles being negatively correlated with assists and missed tackles makes sense as a player usually can't do both in the same play. I am a little suprised that height and weight aren't positively correlated, but this just shows the variety of body types that appear in defensive tackles. The fouling stats are also postively correlated, which again is intuitive.

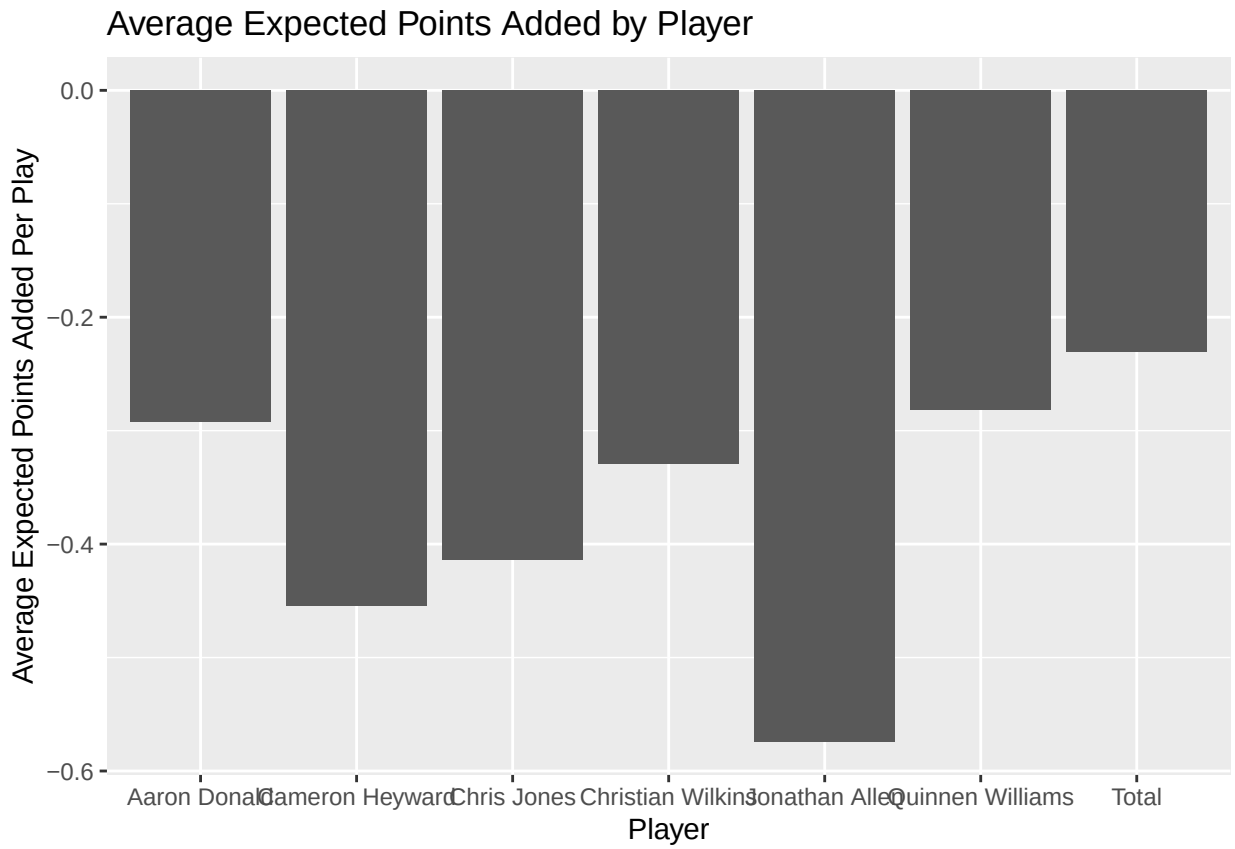
```
vars <- dt_full_df %>% select(tackle, assist, forcedFumble, pff_missedTackle, height, weight, defenders
m <- cor(vars)
corrplot(m)
```




### How "Above Average" are the Best Defensive Tackles

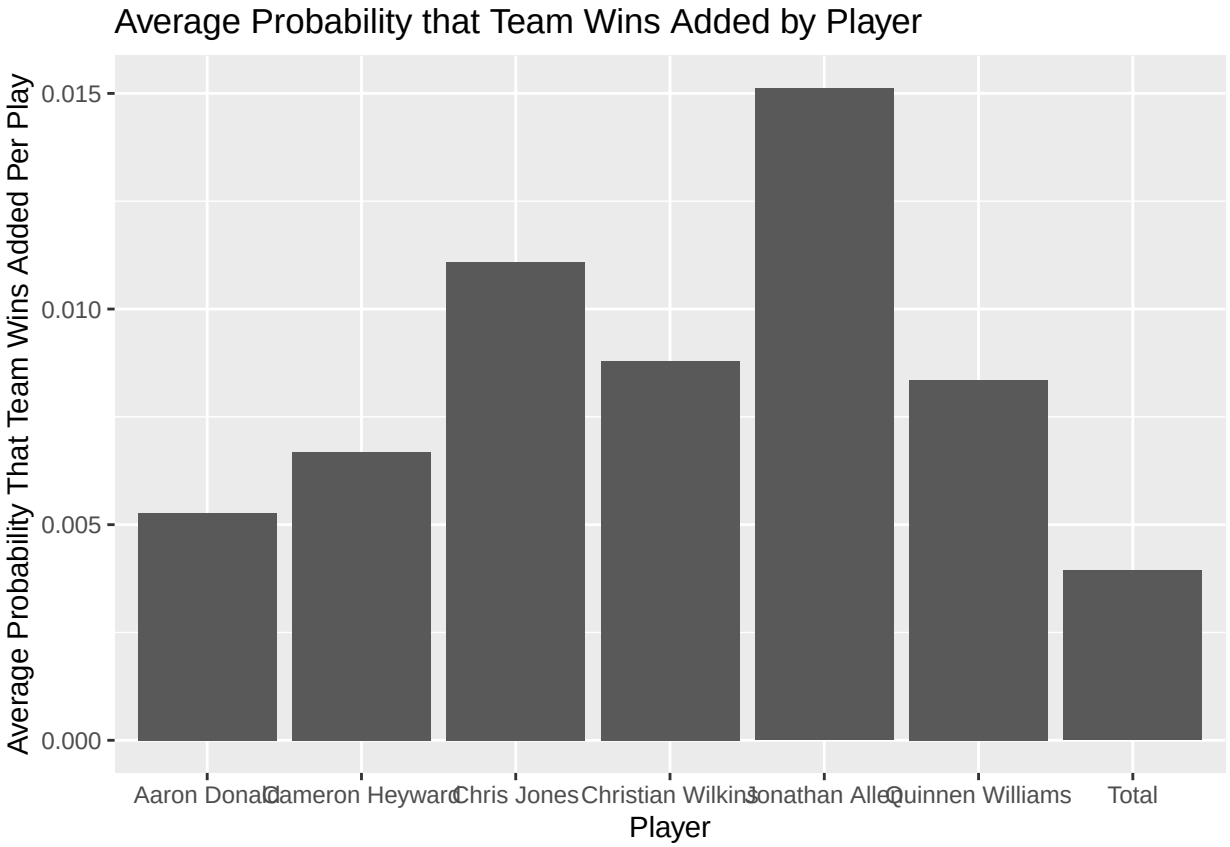
I was interested in seeing how the top defensive tackles would stack up to the average in some of these stats. So, I gathered the top six defensive tackles: Aaron Donald, Chris Jones, Cameron Heyward, Christian Wilkins, Jonathan Allen, and Quinnen Williams and gathered their stats. Here you can see that all of these players outperformed the average EPA, which is good as it shows that the best players do cause a lower EPA. To make this clear, note that EPA refers to the offensive team.

```
players <- c('Aaron Donald', 'Chris Jones', 'Cameron Heyward', 'Christian Wilkins', 'Jonathan Allen', '
player_ids <- players_df %>%
  filter(displayName %in% players) %>%
  select(nflId, displayName)

merged_data <- dt_full_df %>%
  inner_join(player_ids, by = "nflId")

average_epa <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_expectedPointsAdded = mean(expectedPointsAdded))
average_epa <- average_epa %>% add_row(displayName = 'Total', avg_expectedPointsAdded = mean(dt_full_df

ggplot(average_epa, aes(x = displayName, y = avg_expectedPointsAdded)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Expected Points Added by Player",
       x = "Player",
       y = "Average Expected Points Added Per Play")
```



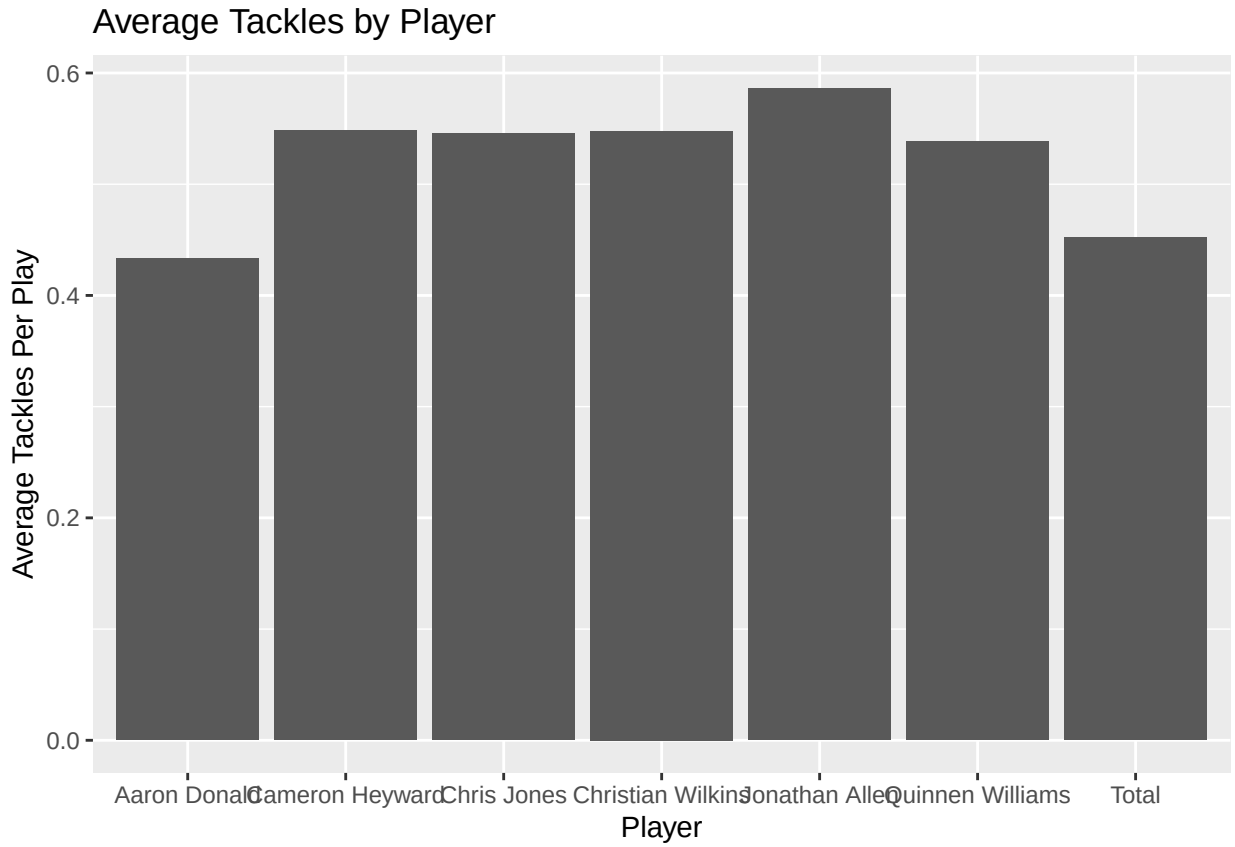

Similarly, the top players also outperformed the average defensive tackle in average probability wins added (TPWA), which shows that these players make plays that lead to their teams having a higher chance of winning at a higher rate than the average player.

```
average_prob <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_teamProbabilityWinAdded = mean(teamProbabilityWinAdded))
average_prob <- average_prob %>% add_row(displayName = 'Total', avg_teamProbabilityWinAdded = mean(dt_fu

ggplot(average_prob, aes(x = displayName, y = avg_teamProbabilityWinAdded)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Probability that Team Wins Added by Player",
       x = "Player",
       y = "Average Probability That Team Wins Added Per Play")
```



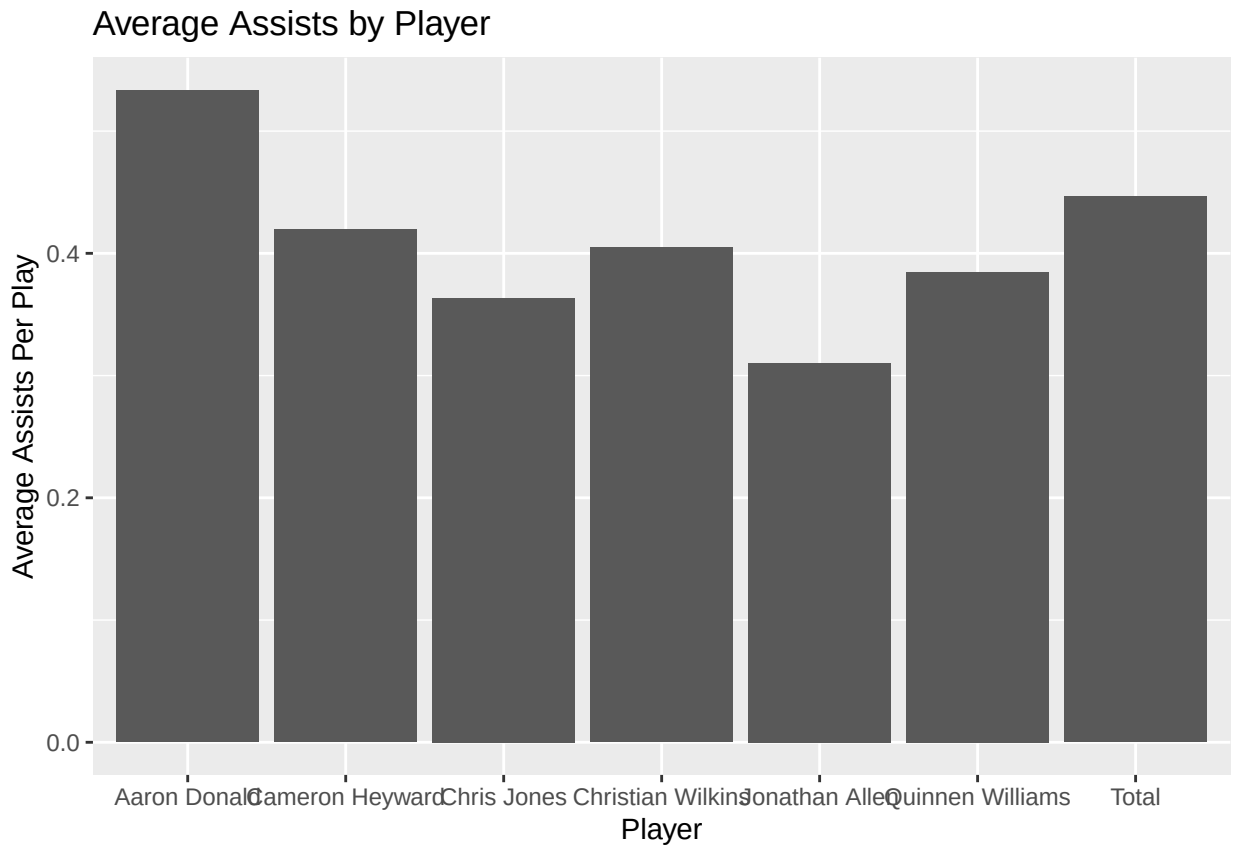

The top players also seem to average more tackles per play than the average player, besides Aaron Donald. This one surprised me, but looking at the stats from earlier it does seem that Aaron Donald is not as good as the other defensive tackles in this position, which leads me to believe that his reputation as a top defensive tackle seems to be carried by his reputation and legacy.

```
average_tackles <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_tackles = mean(tackle))
average_tackles <- average_tackles %>% add_row(displayName = 'Total', avg_tackles = mean(dt_full_df$tack

ggplot(average_tackles, aes(x = displayName, y = avg_tackles)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Tackles by Player",
       x = "Player",
       y = "Average Tackles Per Play")
```



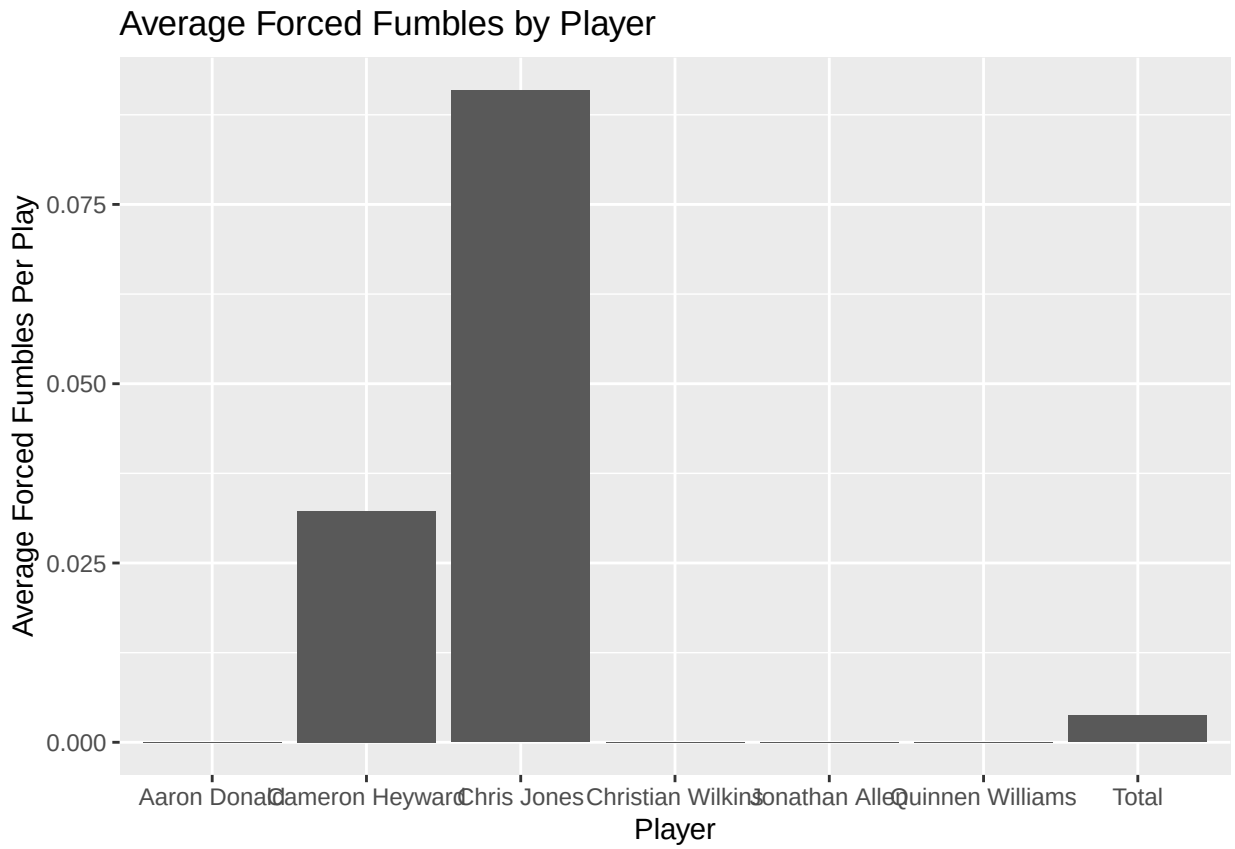

Here we can see that assists seems to vary. more, as while Aaron Donald outperforms the average, the other players do not. There are many interpretations on why this is, one perspective could be the best players usually do the tackling themselves so they never have a chance to assist other players in tackling. Another perspective could be that the top players don't "help" their teammates if they are not the in the forefront of the play. This will be interesting to analyze later. Also, since Aaron Donald seems to be lacking in other stats, it may be that Aaron Donald's main strength is to help his teammates tackle instead of doing the tackling himself.

```
average_assists <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_assists = mean(assist))
average_assists <- average_assists %>% add_row(displayName = 'Total', avg_assists = mean(dt_full_df$ass

ggplot(average_assists, aes(x = displayName, y = avg_assists)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Assists by Player",
       x = "Player",
       y = "Average Assists Per Play")
```



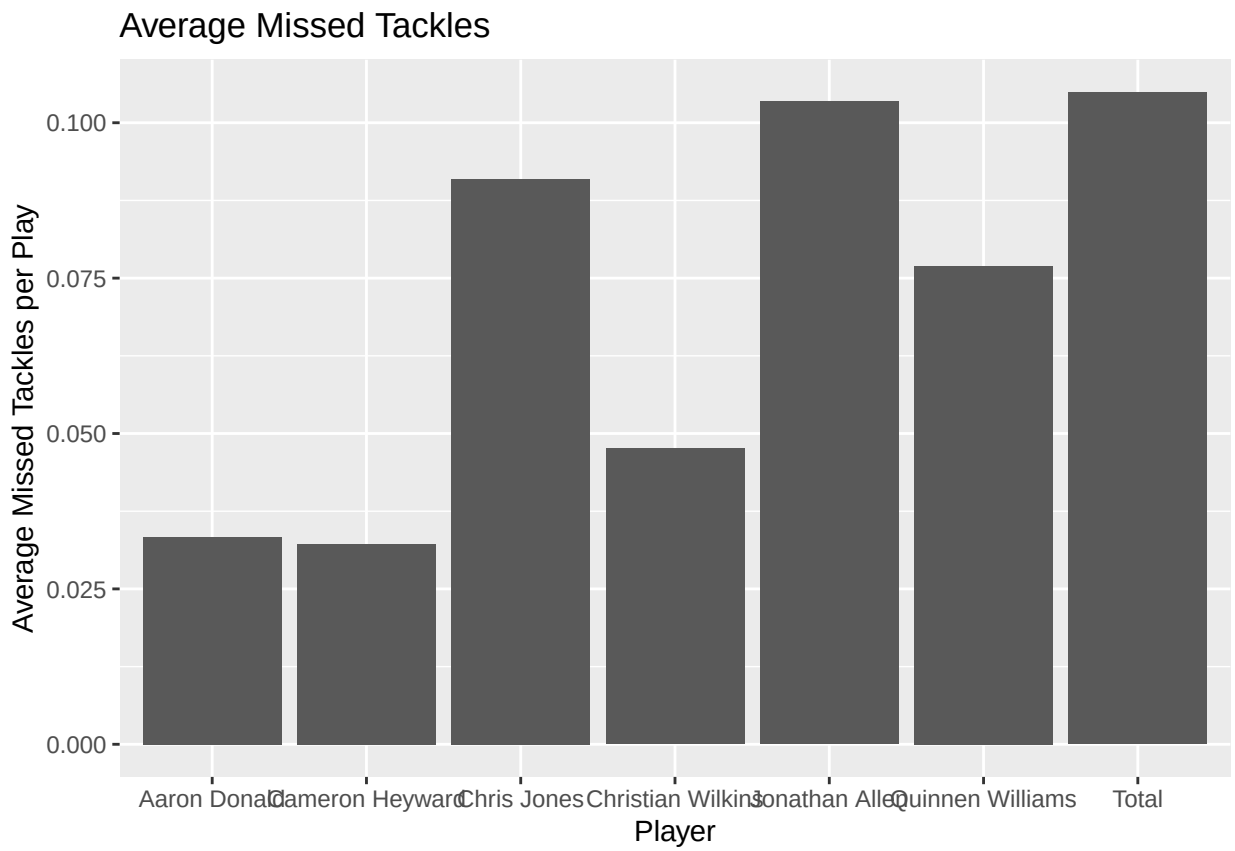

Turnovers are a very big part of football and can change games, but are also very rare. As you can see, 2/3 of the best defensive tackles never recorded any forced fumbles, while Chris Jones and Cameron Heyward out-force fumble the average by a lot. This seems to show that forced fumbles may be luck based, and are not necessarily indicative of skill. It will be interesting to see if forced fumbles happen enough where there is a big correlation between forced fumbles and expected contribution to winning.

```
average_forcedFumbles <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_forcedFumbles = mean(forcedFumble))
average_forcedFumbles <- average_forcedFumbles %>% add_row(displayName = 'Total', avg_forcedFumbles = m

ggplot(average_forcedFumbles, aes(x = displayName, y = avg_forcedFumbles)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Forced Fumbles by Player",
       x = "Player",
       y = "Average Forced Fumbles Per Play")
```



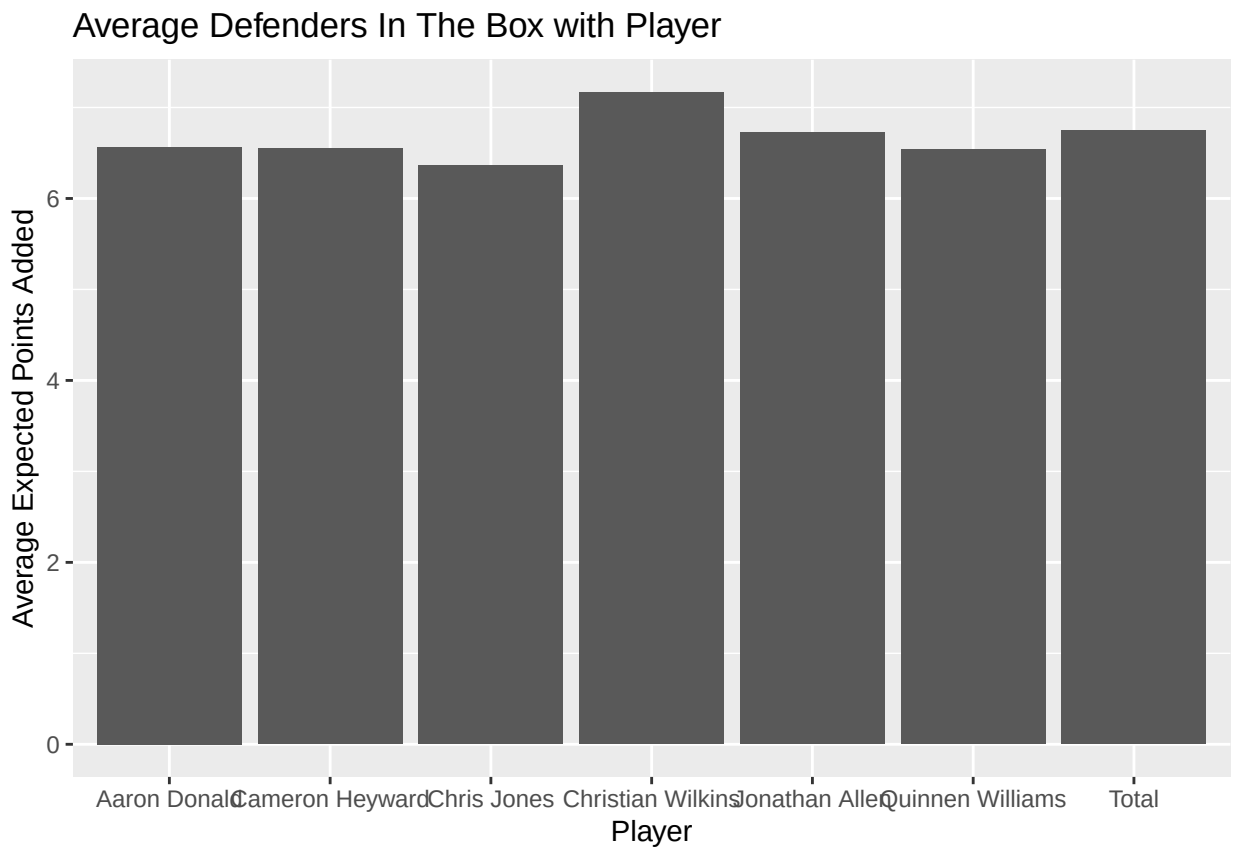

Now we are looking at average missed tackles. The best players in theory should miss less tackles than the average, which was shown to be generally true. However, Chris Jones and Jonathen Allen were awfully close to the average. One idea I had before looking at the data was that players who went for more forced fumbles would miss more tackles, which was shown to be true for Chris Jones. However, Jonathan Allen generated no forced fumbles and missed a lot of tackles, and Cameron Heyward was the best at not missing tackles and generated forced fumbles. Therefore, there doesn't seem to be any correlation.

```
average_missedTackles <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_missedTackles = mean(pff_missedTackle))
average_missedTackles <- average_missedTackles %>% add_row(displayName = 'Total', avg_missedTackles = m

ggplot(average_missedTackles, aes(x = displayName, y = avg_missedTackles)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Missed Tackles",
       x = "Player",
       y = "Average Missed Tackles per Play")
```




The last visual will be defenders in the box. The logic behind this is that the best defenders need less help in doing their job, which may be supported by the following graph as the majority of these players have less defenders in the box than average. However, the difference is so small that I am not sure if this really plays a significant role. This stat may be controlled by the defensive scheme of the team, and not the personnel. This stat will also be interesting to see in the future.

```
average_defendersInBox <- merged_data %>%
  group_by(displayName) %>%
  summarize(avg_defendersInBox = mean(defendersInTheBox))
average_defendersInBox <- average_defendersInBox %>% add_row(displayName = 'Total', avg_defendersInBox =

ggplot(average_defendersInBox, aes(x = displayName, y = avg_defendersInBox)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Defenders In The Box with Player",
       x = "Player",
       y = "Average Expected Points Added")
```




# Preparing for the Models

## Spliting the Data

Since we are using two target variables, we have to create two sets of training/testing sets for both EPA and TPWA. We are using a ratio of 0.8 because are data set is pretty large.

```
set.seed(999)
epa_split <- initial_split(dt_full_df, prop = 0.8, strata = expectedPointsAdded)
epa_train <- training(epa_split)
epa_test <- testing(epa_split)
tpwa_split <- initial_split(dt_full_df, prop = 0.8, strata = expectedPointsAdded)
tpwa_train <- training(tpwa_split)
tpwa_test <- testing(tpwa_split)

epa_folds <- vfold_cv(epa_train, v = 10, strata = expectedPointsAdded)
```

## K-Fold Validation

Here I am also creating the folds for k-fold validation. We are using 10 folds here. We are also stratifying the data by the target variables.

```
epa_folds <- vfold_cv(epa_train, v = 10, strata = expectedPointsAdded)
tpwa_folds <- vfold_cv(tpwa_train, v = 10, strata = expectedPointsAdded)
```

### Creating The Recipe

I am also creating recipes for both EPA and TPWA. The predictors we are using are tackle, assist, forcedFumble, pff_missedTackle, penaltyYardsAtFault, offenseFormation, and defendersInTheBox. We are going to create dummy variables for offenseFormation because it is a factor, and will be normalizing all the predictors.

```
epa_recipe <- recipe(expectedPointsAdded ~ tackle + assist + forcedFumble + pff_missedTackle + penaltyYa
  step_dummy(offenseFormation) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

#prep(epa_recipe) %>%
#  bake(new_data = epa_train) %>%
#  kable() %>%
#  kable_styling(full_width = F) %>%
#  scroll_box(width = "100%", height = "200px")
```

```
tpwa_recipe <- recipe(teamProbabilityWinAdded ~ tackle + assist + forcedFumble + pff_missedTackle + pena
  step_dummy(offenseFormation) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

#prep(tpwa_recipe) %>%
#  bake(new_data = tpwa_train) %>%
#  kable() %>%
#  kable_styling(full_width = F) %>%
#  scroll_box(width = "100%", height = "200px")
```

## Setting Up the Models

### Building the Models

Since our target variable is quantitative, we will be using regression models. For this project, I chose linear regression, ridge, k-nearest neighbors, elastic net, and boosted trees.

```
lm_model <- linear_reg() %>%
  set_engine("lm")

ridge_spec <- linear_reg(mixture = 0,
                         penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

knn_model <- nearest_neighbor(neighbors = tune()) %>%
  set_mode("regression") %>%
  set_engine("kknn")

elastic_spec <- linear_reg(penalty = tune(),
                           mixture = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

boosted_spec <- boost_tree(trees = tune(),
                           learn_rate = tune(),
                           min_n = tune()) %>%
```

```
  set_engine("xgboost") %>%
  set_mode("regression")
```

### Setting up the Workflows

Now I am creating workflows for both statistics : EPA and TPWA, for each model.

```
epa_lm_workflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(epa_recipe)
tpwa_lm_workflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(tpwa_recipe)

epa_ridge_workflow <- workflow() %>%
  add_recipe(epa_recipe) %>%
  add_model(ridge_spec)
tpwa_ridge_workflow <- workflow() %>%
  add_recipe(tpwa_recipe) %>%
  add_model(ridge_spec)

epa_knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(epa_recipe)
tpwa_knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(tpwa_recipe)

epa_elastic_workflow <- workflow() %>%
  add_recipe(epa_recipe) %>%
  add_model(elastic_spec)
tpwa_elastic_workflow <- workflow() %>%
  add_recipe(tpwa_recipe) %>%
  add_model(elastic_spec)

epa_boosted_workflow <- workflow() %>%
  add_recipe(epa_recipe) %>%
  add_model(boosted_spec)
tpwa_boosted_workflow <- workflow() %>%
  add_recipe(tpwa_recipe) %>%
  add_model(boosted_spec)
```

### Setting up the Tuning Grids

Now we are setting up the tuning grids for each model besides the linear regression model, as we are not tuning any variables for that model.

```
penalty_grid <- grid_regular(penalty(range = c(-5,5)), levels = 10)

knn_grid <- grid_regular(neighbors(range = c(1,15)), levels = 10)

elastic_grid <- grid_regular(penalty(range = c(-5, 5)), mixture(range = c(0,1)), levels = 10)

boosted_grid <- grid_regular(trees(range = c(5, 200)), learn_rate(range = c(0.01,0.1), trans = identity
```

### Tuning the Models

Tuning the ridge models. Note that the next chunks of code have "eval=FALSE", this is because this code takes a while to run. So instead, we run the tuning and save the results so we don't have to repeat this every time the file is knit.

```
epa_ridge_tuned <- tune_grid(
  epa_ridge_workflow,
  resamples = epa_folds,
  grid = penalty_grid
)

tpwa_ridge_tuned <- tune_grid(
  tpwa_ridge_workflow,
  resamples = tpwa_folds,
  grid = penalty_grid
)
```

Tuning the knn models.

```
epa_knn_tuned <- tune_grid(
    epa_knn_workflow,
    resamples = epa_folds,
    grid = knn_grid
)

tpwa_knn_tuned <- tune_grid(
    tpwa_knn_workflow,
    resamples = tpwa_folds,
    grid = knn_grid
)
```

Tuning the elastic models.

```
epa_elastic_tuned <- tune_grid(
  epa_elastic_workflow,
  resamples = epa_folds,
  grid = elastic_grid
)

tpwa_elastic_tuned <- tune_grid(
  tpwa_elastic_workflow,
  resamples = tpwa_folds,
  grid = elastic_grid
)
```

Tuning the boosted trees models.

```
epa_boosted_tune_res <- tune_grid(
  epa_boosted_workflow,
  resamples = epa_folds,
  grid = boosted_grid
)

tpwa_boosted_tune_res <- tune_grid(
  tpwa_boosted_workflow,
  resamples = tpwa_folds,
```

```
  grid = boosted_grid
)
```

### Saving the Tuned Models

Saving the results of the tuning so I don't have to tune the models every time the document is knitted.

```
saveRDS(epa_ridge_tuned, file = 'epa_ridge.RDS')
saveRDS(tpwa_ridge_tuned, file = 'tpwa_ridge.RDS')

saveRDS(epa_knn_tuned, file = 'epa_knn.RDS')
saveRDS(tpwa_knn_tuned, file = 'tpwa_knn.RDS')

saveRDS(epa_elastic_tuned, file = 'epa_elastic.RDS')
saveRDS(tpwa_elastic_tuned, file = 'tpwa_elastic.RDS')

saveRDS(epa_boosted_tune_res, file = 'epa_boosted.RDS')
saveRDS(tpwa_boosted_tune_res, file = 'tpwa_boosted.RDS')
```

### Loading the Tuned models

Loading up the saved results of the tuning.

```
epa_ridge_tuned <- read_rds(file = "epa_ridge.RDS")
tpwa_ridge_tuned <- read_rds(file = "tpwa_ridge.RDS")

epa_knn_tuned <- read_rds(file = "epa_knn.RDS")
tpwa_knn_tuned <- read_rds(file = "tpwa_knn.RDS")

epa_elastic_tuned <- read_rds(file = "epa_elastic.RDS")
tpwa_elastic_tuned <- read_rds(file = "tpwa_elastic.RDS")

epa_boosted_tuned <- read_rds(file = "epa_boosted.RDS")
tpwa_boosted_tuned <- read_rds(file = "tpwa_boosted.RDS")
```

## Fitting the Models

Now I'm collecting the RMSE for each model, which will then be used to grade the models.

```
epa_lm_fit <- fit_resamples(epa_lm_workflow, resamples = epa_folds)
epa_lm_rmse <- collect_metrics(epa_lm_fit) %>%
  filter(.metric == "rmse")
tpwa_lm_fit <- fit_resamples(tpwa_lm_workflow, resamples = tpwa_folds)
tpwa_lm_rmse <- collect_metrics(tpwa_lm_fit) %>%
  filter(.metric == "rmse")

epa_ridge_rmse <- collect_metrics(epa_ridge_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")
tpwa_ridge_rmse <- collect_metrics(tpwa_ridge_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")

epa_knn_rmse <- collect_metrics(epa_knn_tuned) %>%
```

```
  arrange(mean) %>%
  filter(.metric == "rmse")
tpwa_knn_rmse <- collect_metrics(tpwa_knn_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")

epa_elastic_rmse <- collect_metrics(epa_elastic_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")
tpwa_elastic_rmse <- collect_metrics(tpwa_elastic_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")

epa_boosted_rmse <- collect_metrics(epa_boosted_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")
tpwa_boosted_rmse <- collect_metrics(tpwa_boosted_tuned) %>%
  arrange(mean) %>%
  filter(.metric == "rmse")
```

# Results of the Models

## EPA

Based off the average RMSE of each model, the linear regression model did the best for the EPA statistic, with the Ridge Regression and Elastic Net models coming close.

```
epa_final_compare_tibble <- tibble(Model = c("Linear Regression", "Ridge Regression", "K Nearest Neighb

epa_final_compare_tibble <- epa_final_compare_tibble %>%
  arrange(RMSE)

epa_final_compare_tibble
```

```
## # A tibble: 5 x 2
##   Model                RMSE
##   <chr>               <dbl>
## 1 Linear Regression   0.771
## 2 Ridge Regression    0.790
## 3 Elastic Net         0.793
## 4 Boosted Trees       0.813
## 5 K Nearest Neighbors 1.18
```

## TPWA

Similarly to the models looking at EPA, linear regression did the best in measuring the TPWA statistic, with Ridge Regression and Elastic Net coming next. However, Boosted Trees did not good for TPWA, while doing decently well for EPA.

```
tpwa_final_compare_tibble <- tibble(Model = c("Linear Regression", "Ridge Regression", "K Nearest Neighb

tpwa_final_compare_tibble <- tpwa_final_compare_tibble %>%
  arrange(RMSE)
```

```
tpwa_final_compare_tibble
```

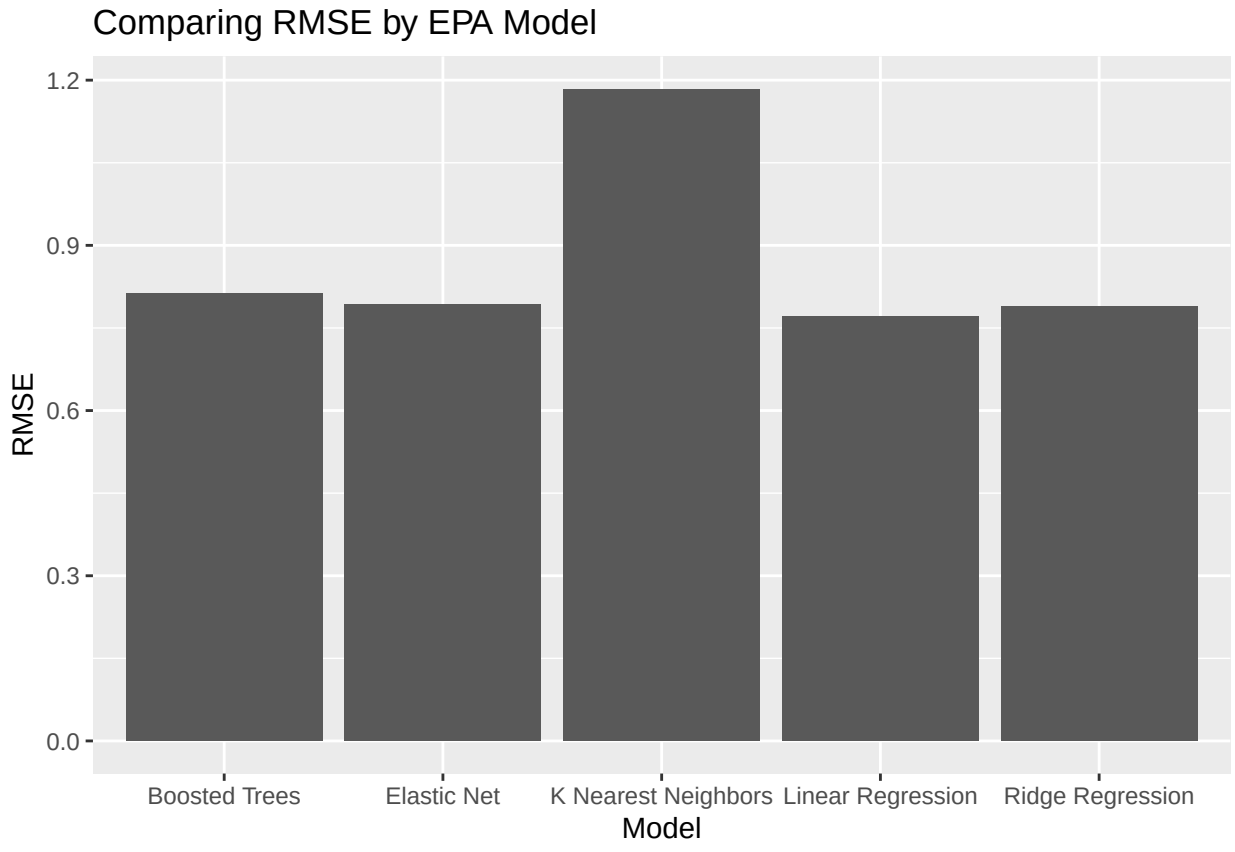
```
## # A tibble: 5 x 2
##   Model                 RMSE
##   <chr>                <dbl>
## 1 Linear Regression   0.0222
## 2 Ridge Regression    0.0224
## 3 Elastic Net         0.0224
## 4 K Nearest Neighbors 0.0313
## 5 Boosted Trees       0.0890
```

# Visualizing the Models

## Visualizing RMSE

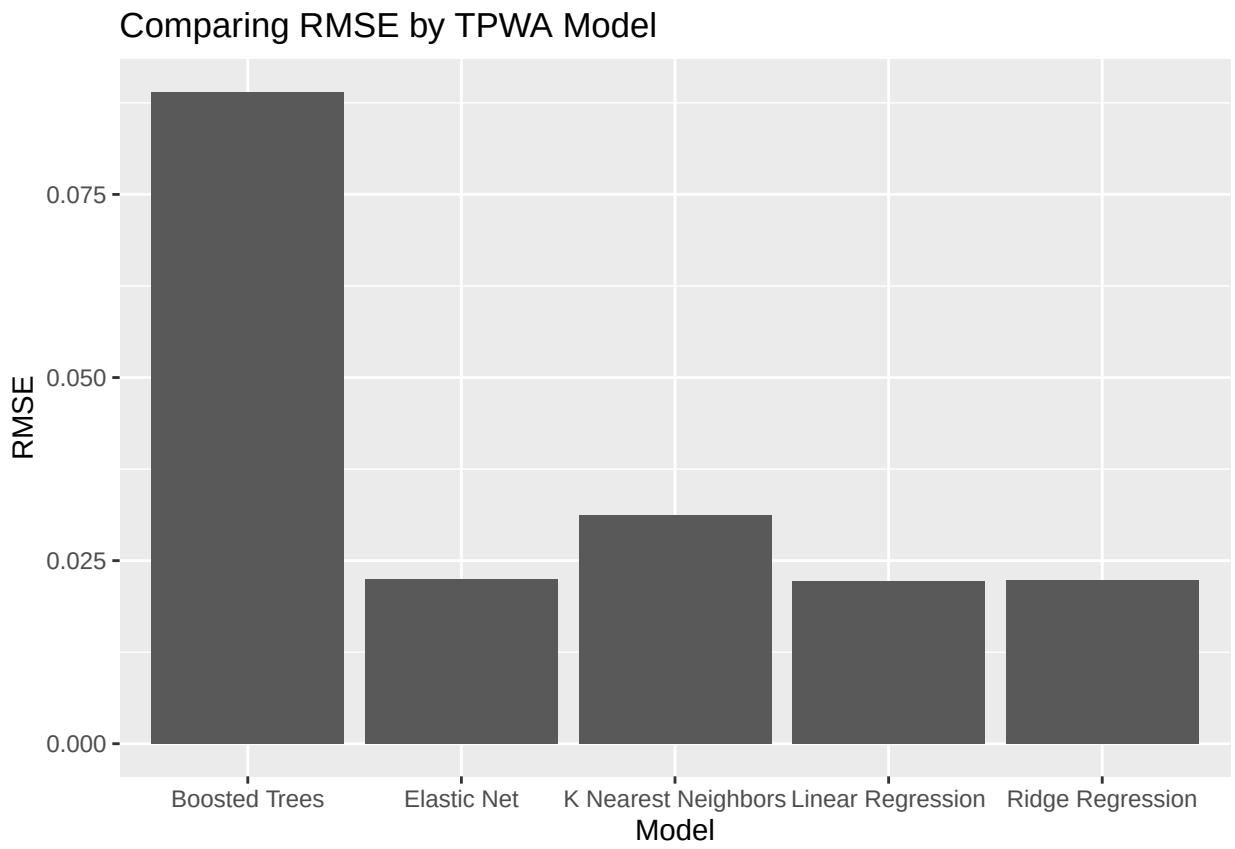
Here are more visualizations for the RMSE for all the models. As you can see, the linear regression model performed the best, suggesting the data fits a rather linear trend. While k-nearest neighbors performed the worse for EPA, the difference doesn't seem to be very drastic.

```
epa_models <- data.frame(Model = c("Linear Regression", "Ridge Regression",  "K Nearest Neighbors", "Ela
                         RMSE = c(epa_lm_rmse$mean, mean(epa_ridge_rmse$mean), mean(epa_knn_rmse$mean),

ggplot(epa_models, aes(x=Model, y=RMSE)) +
  geom_bar(stat = "identity") +
  theme(legend.position = "none") +
  labs(title = "Comparing RMSE by EPA Model")
```



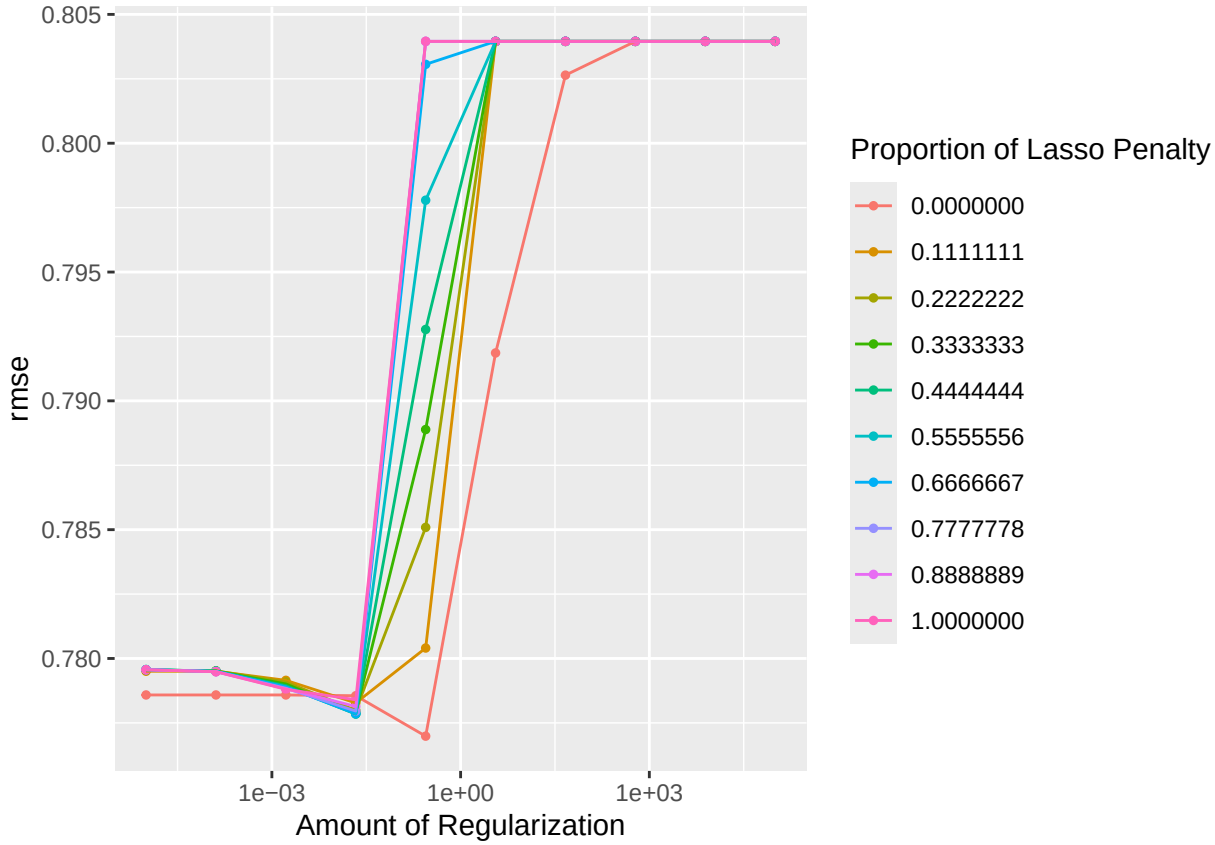
Visualizing the RMSE for the TPWA models, the main thing that sticks out is the stark difference in RMSE between boostd trees and the rest of the models. One explaination for this may be that TPWA had more outliers than EPA, which led to the Boosted Trees being less accurate as they are very sensitive to outliers.

```
tpwa_models <- data.frame(Model = c("Linear Regression", "Ridge Regression",  "K Nearest Neighbors", "E
                          RMSE = c(tpwa_lm_rmse$mean, mean(tpwa_ridge_rmse$mean), mean(tpwa_knn_rmse$mea

ggplot(tpwa_models, aes(x=Model, y=RMSE)) +
  geom_bar(stat = "identity") +
  theme(legend.position = "none") +
  labs(title = "Comparing RMSE by TPWA Model")
```



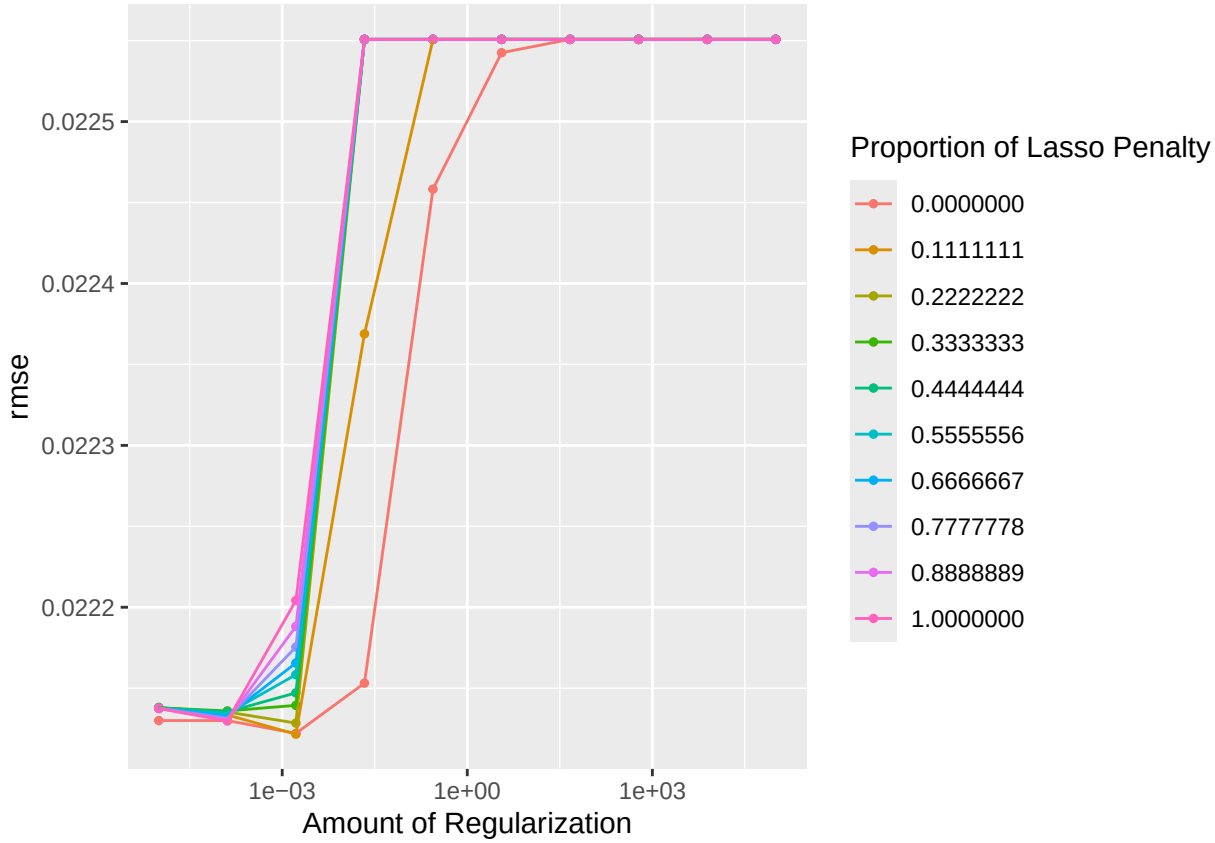

## Autopots for all the Tuned Models Here are the autoplots for the EPA elastic models. With less regularization it seems that all the models performed very similarly. However, as the regularization went up, the differences in lasso penalty proportions seemed to affect the RMSE, namely the lower the proportion the better the model did. For very high amounts of regularization, all the models seemed to end up with the same highest RMSE.

```
autoplot(epa_elastic_tuned, metric = 'rmse')
```



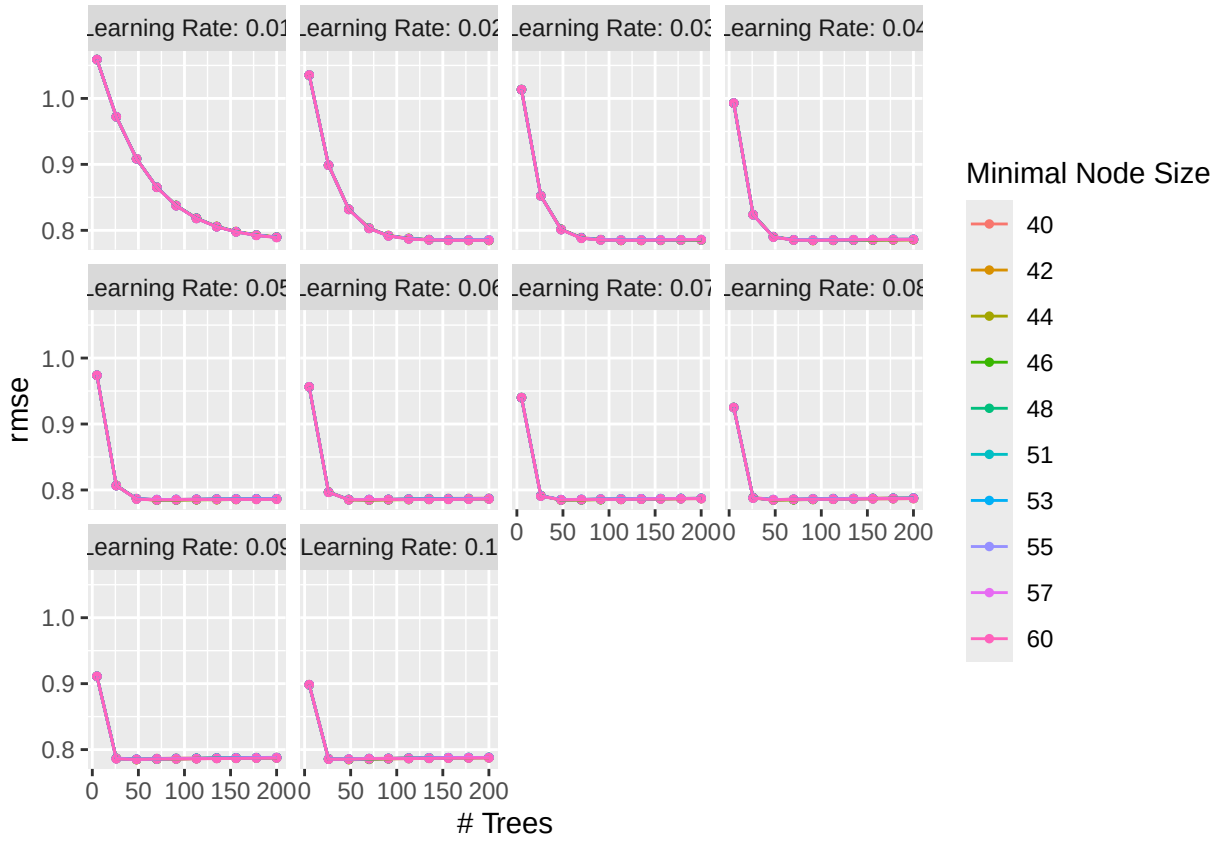

The autoplots for the TWWA elastic models seemed to tell the same story as the EPA models. The main differences I see is that the RMSE between the proportions of lasso penalties from 0.2222 to 1 are a lot more similar compared to the Elastic EPA models.

```
autoplot(tpwa_elastic_tuned, metric = 'rmse')
```



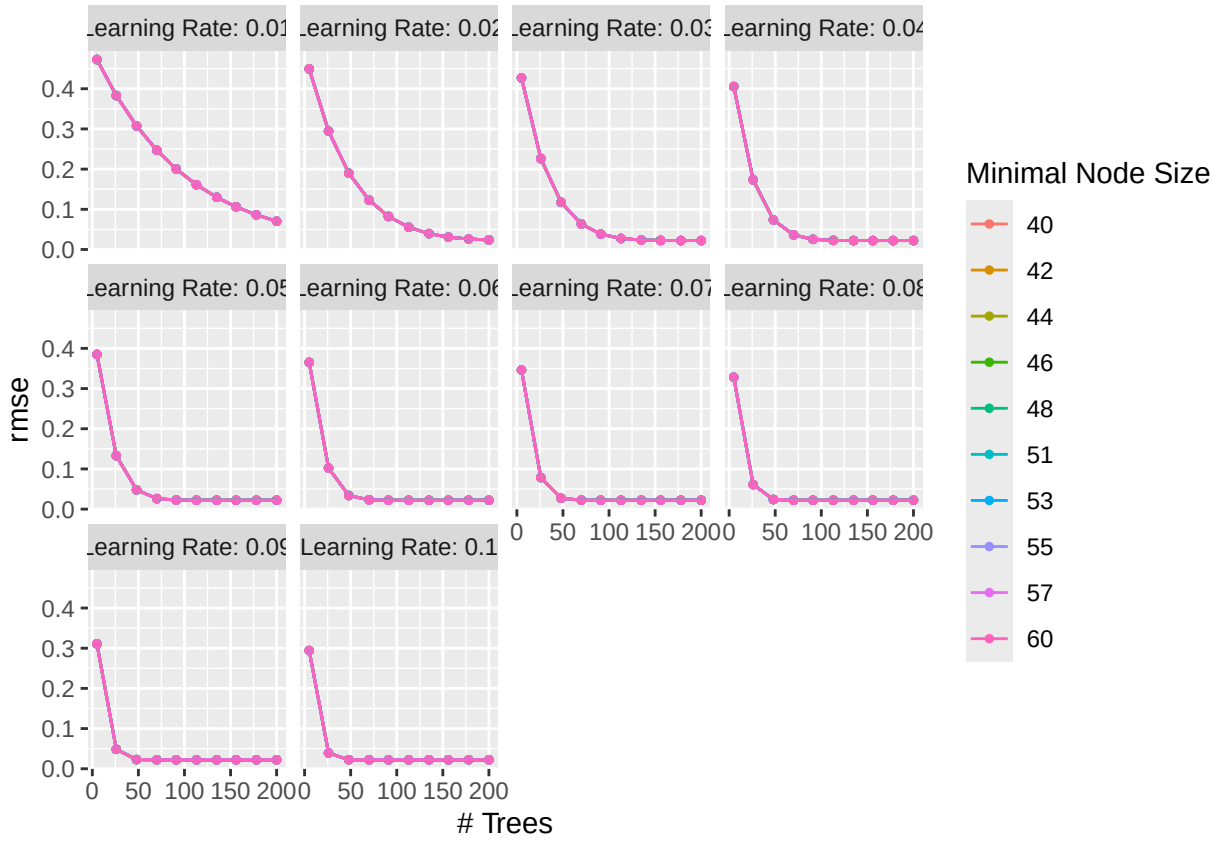

Here are the autoplots for the boosted tree models. For the EPA models, it seemed that the minimal node size didn't have much effect on the RMSE. The learning rate seemed to change how fast the RMSE dropped off. For learning rates less and equal to 0.5, the data took less time to drop off with a learning rate of 0.01 having the slowest drop off. Learning rates above 0.5 had very quick drop offs in RMSE. Lastly, for all the models, as the number of trees went up the RMSE went down.

```
autoplot(epa_boosted_tuned, metric = 'rmse')
```




For TPWA, all the plots seem to share the same trends as the EPA boosted trees with one major difference: the drop off in RMSE is a lot slower compared to the EPA boosted trees. However, besides that, these boosted trees shared the same trends as the EPA ones.

```
autoplot(tpwa_boosted_tuned, metric = 'rmse')
```




# Fitting the Best Model and Finding the Most Impactful Stats

## EPA

Looking at all the EPA ridge regression models, it seems that the best model had a penalty of 0.278 with an RMSE of 0.7769851, which is lower than the RMSE for the linear regression model, which means that this model fitted the data the best. Looking at the range of expected points: -10 to 5, it seems that the model did decently well at explaining the variation of the data.

```
epa_ridge_tuned %>%
  collect_metrics() %>%
  arrange(mean) %>%
    filter(.metric == "rmse")
```

```
## # A tibble: 10 x 7
##          penalty .metric .estimator  mean     n std_err .config
##            <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1      0.278    rmse    standard   0.777    10  0.0356 Preprocessor1_Model05
##  2      0.0215   rmse    standard   0.779    10  0.0356 Preprocessor1_Model04
##  3      0.00001  rmse    standard   0.779    10  0.0356 Preprocessor1_Model01
##  4      0.000129 rmse    standard   0.779    10  0.0356 Preprocessor1_Model02
##  5      0.00167  rmse    standard   0.779    10  0.0356 Preprocessor1_Model03
##  6      3.59     rmse    standard   0.792    10  0.0360 Preprocessor1_Model06
##  7     46.4      rmse    standard   0.803    10  0.0361 Preprocessor1_Model07
##  8    599.       rmse    standard   0.804    10  0.0362 Preprocessor1_Model08
##  9   7743.       rmse    standard   0.804    10  0.0362 Preprocessor1_Model09
```

```
## 10 100000        rmse    standard   0.804    10  0.0362 Preprocessor1_Model10
```

```
epa_best_ridge <- select_best(epa_ridge_tuned, metric = 'rmse')
```

Now we are fitting the best ridge model to the testing data, in which we get a RMSE of 0.69, which was lower than the RMSE on the training data.

```
epa_finalworkflow <- finalize_workflow(epa_ridge_workflow, epa_best_ridge)
epa_finalfit <- fit(epa_finalworkflow, data = epa_train)

epa_predictions <- augment(epa_finalfit, epa_test)
epa_predictions %>% metrics(truth = expectedPointsAdded, estimate = .pred) %>%
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.690
```

Now we get to look at the variable importance by looking at the constants associated with each variable. At least for EPA, it seems that pff_missedTackles and penaltyYardsAtFault had the worst effect on EPA, while forced fumbles and tackles had the best effect on EPA.

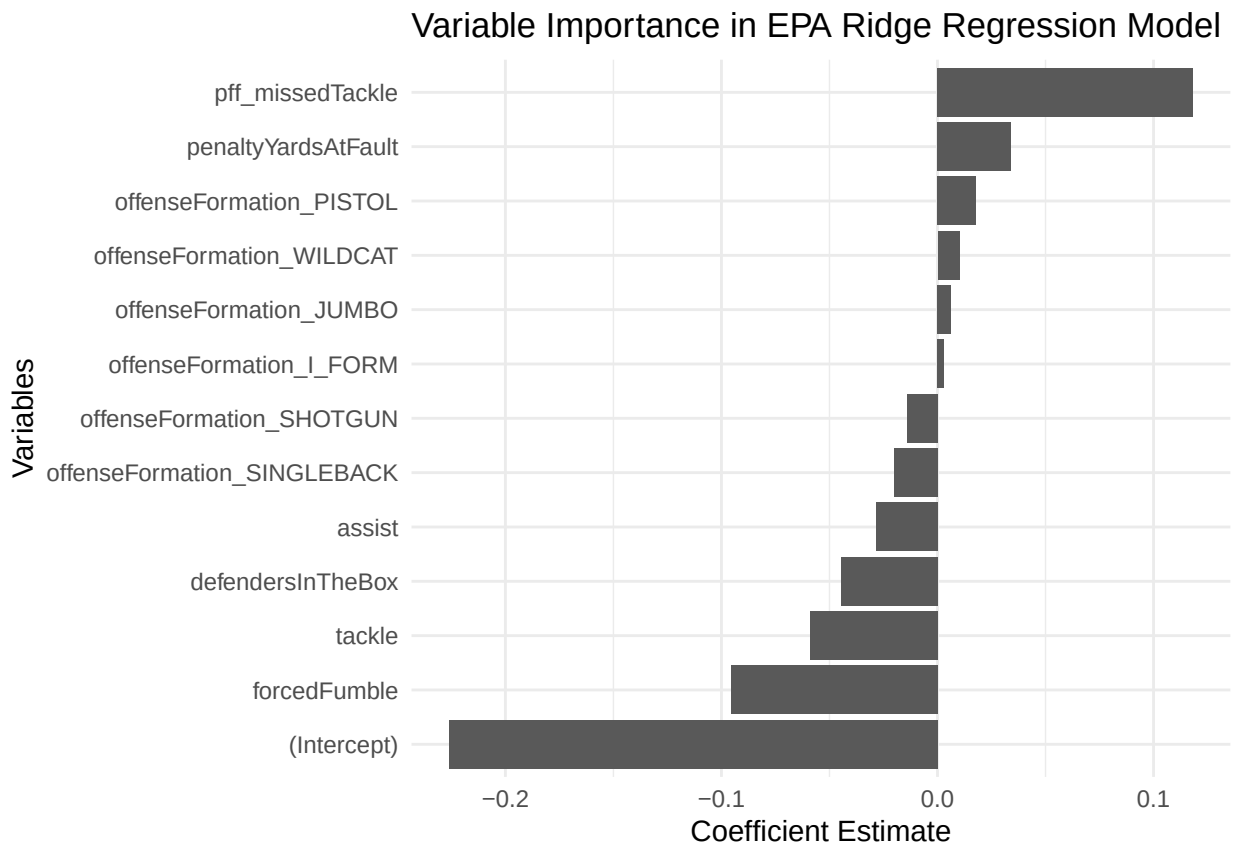
```
final_model <- extract_fit_parsnip(epa_finalfit)
ridge_coefs <- tidy(final_model)
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

```
ggplot(ridge_coefs, aes(x = reorder(term, estimate), y = estimate)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Variable Importance in EPA Ridge Regression Model",
       x = "Variables",
       y = "Coefficient Estimate") +
  theme_minimal()
```




## TPWA Now we are looking at the best ridge regression model for TPWA, which had a penalty of 0.001668 and had an RMSE of 0.022122, which again outperformed the linear regression model. Since TPWA ranged from -.23 to .23, it also seems that model did decently well in explaining the variance in TPWA.

```
tpwa_ridge_tuned %>%
  collect_metrics() %>%
  arrange(mean) %>%
    filter(.metric == "rmse")
```

```
## # A tibble: 10 x 7
##          penalty .metric .estimator   mean     n std_err .config
##            <dbl> <chr>   <chr>       <dbl> <int>   <dbl> <chr>
##  1      0.00167  rmse    standard   0.0221    10 0.00141 Preprocessor1_Model03
##  2      0.00001  rmse    standard   0.0221    10 0.00140 Preprocessor1_Model01
##  3      0.000129 rmse    standard   0.0221    10 0.00140 Preprocessor1_Model02
##  4      0.0215   rmse    standard   0.0222    10 0.00146 Preprocessor1_Model04
##  5      0.278    rmse    standard   0.0225    10 0.00150 Preprocessor1_Model05
##  6      3.59     rmse    standard   0.0225    10 0.00150 Preprocessor1_Model06
##  7     46.4      rmse    standard   0.0226    10 0.00150 Preprocessor1_Model07
##  8    599.       rmse    standard   0.0226    10 0.00150 Preprocessor1_Model08
##  9   7743.       rmse    standard   0.0226    10 0.00150 Preprocessor1_Model09
## 10 100000        rmse    standard   0.0226    10 0.00150 Preprocessor1_Model10
```

```
tpwa_best_ridge <- select_best(tpwa_ridge_tuned, metric = 'rmse')
```

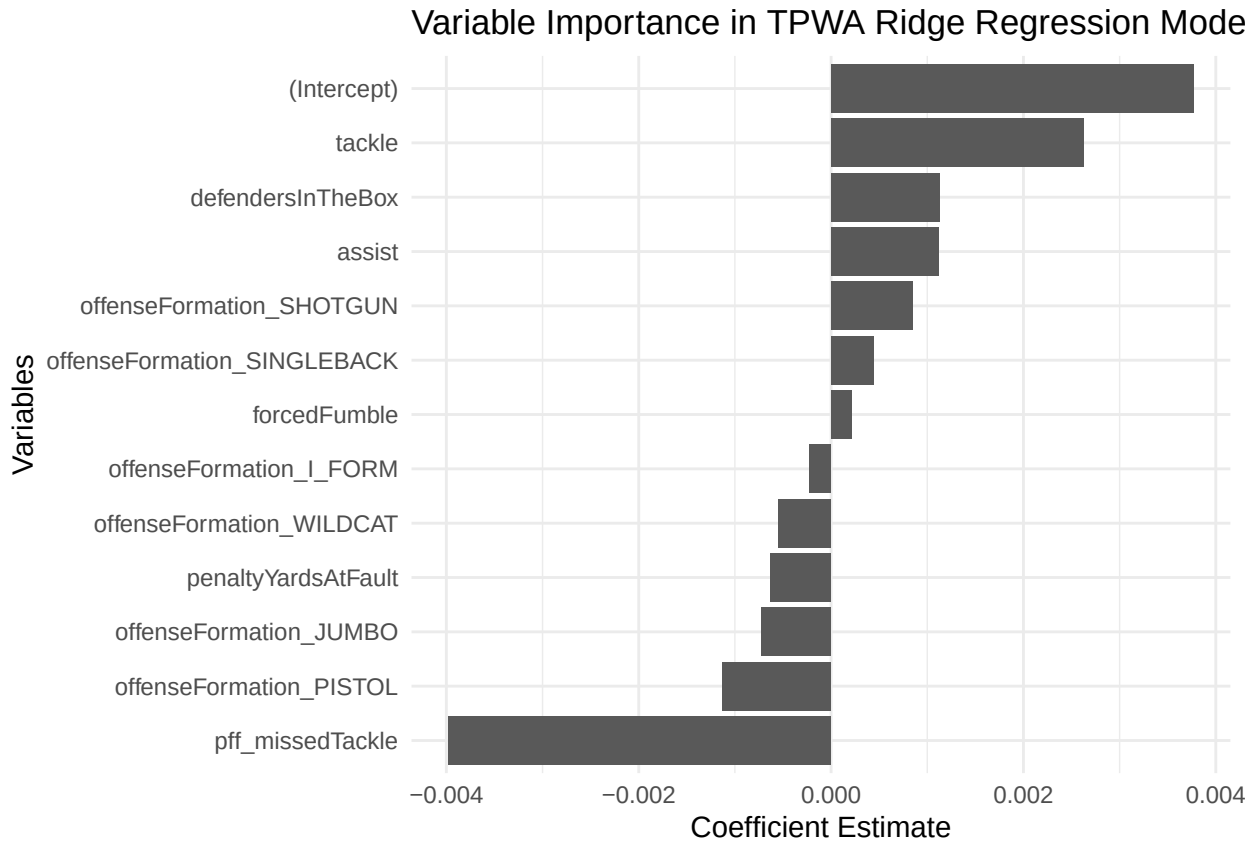
Now we are fitting the best ridge regression model on to the TPWA testing set. Again the RMSE of model on the testing data was lower than the RMSE for the training data.

```
tpwa_finalworkflow <- finalize_workflow(tpwa_ridge_workflow, tpwa_best_ridge)
tpwa_finalfit <- fit(tpwa_finalworkflow, data = tpwa_train)

tpwa_predictions <- augment(tpwa_finalfit, tpwa_test)
tpwa_predictions %>% metrics(truth = teamProbabilityWinAdded, estimate = .pred) %>%
  filter(.metric == "rmse")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0209
```

Looking at the constants for each variable, we see that pff_missedTackles and the pistol offensive formation had the worst effect, while tackles and defenders in the box had the greatest impact on TPWA.

```
final_model <- extract_fit_parsnip(tpwa_finalfit)
ridge_coefs <- tidy(final_model)
ggplot(ridge_coefs, aes(x = reorder(term, estimate), y = estimate)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Variable Importance in TPWA Ridge Regression Model",
       x = "Variables",
       y = "Coefficient Estimate") +
  theme_minimal()
```




## Analysis Interestingly, EPA and TPWA had different constants for the variables that were used to predict. While the tackle stat was consistently important as a "winning" stat for both statistics and pff_missedTackle as the most important "losing" stat, everything else was different. The TPWA model

valued the offensive formations and defenders in the box more, suggesting that the plays dictated the success of the team and not the players themselves. The TPWA model also didn't value forced fumbles as high as the EPA model did. I believe this is because that while it does stop the opposing team from scoring, what the offense does with the turnover is different for every team and as such will cause a difference in team's chances of winning. However, something consistent for both models was assisting having a slightly positively contribution for the defensive team. Something to note is that the pistol formation was consistently good for the offense, while the singleback and shotgun formations were consistently good for the defense.

# Conclusion

In this project I used two sets of models to analyze a defensive player's stats for each play, as well as the contextual stats, effects on two stats: expected points added and team probability win added. Using linear regression, ridge regression, lasso, k-nearest neighbors, and boosted trees, I found that the ridge regression model with a specifically tuned penalty performed the best while the boosted trees model did the worst for TPWA while k-nearest neighbors did the worst for EPA. Using the specific ridge regression model, I found that a player's individual stats affected the opposing team's expected points, while the contextual stats like defenders in the box and the offensive formation affected the team's probability to win more. Consistently however, tackles were always the most beneficial plays for a defensive tackle and missed tackles were the worst plays.

One question I have at the moment were the importance of the plays. From the data, it seemed that pistol and wildcat were the best offensive formations while shotgun aned singleback were the worst. I don't know if this applies only for defensive tackles and the plays they make, as shotgun and singleback offense formations are more geared towards passing and pistol/wildcat are more geared towards running the ball. However, it has been apparent in the NFL that passing is a more efficient form of offense compared to running the ball, which has led to teams favoring passing offenses. Does this mean that defensive tackles are more likely to make positively impactful plays during passing plays, but are less likely to for running plays, or is there another reason that I am not seeing.

For future research, I think using more defensive player statistics will make the model better, as the RMSE was rather high for the range of both expected points added and team probability win added. There are a lot of statistics that were missing, such as sacks and tackles behind the line of scrimmage, both of which are very integral stats for people in the line of scrimmage. Also, there may be a way to combine both EPA and TPWA that I'm not aware of, which would make modelling this a lot more efficient. Lastly, there is nearly 50 years of data for defensive tackles, so gathering even more data can finally lead to a model in which we are able to see what stats lead to the most impactful defensive tackles.