exploratory

Final Project

By Sanaz Ebrahimi

Introduction

Loading Packages

We need to load the necessary packages in here so that we can access functions for our eda and model running later on.

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(tidyverse)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.0.0 --
## v broom        1.0.1     v rsample      1.1.0
## v dials        1.0.0     v tune         1.0.0
## v infer        1.0.3     v workflows    1.1.0
## v modeldata    1.0.1     v workflowsets 1.0.0
## v parsnip      1.0.1     v yardstick    1.1.0
## v recipes      1.0.1
```

```
## -- 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()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(ggplot2)

library(ISLR)
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'
##
## The following objects are masked from 'package:ISLR':
##
##     Auto, Credit
```

```
#install.packages("discrim")
library(discrim)
```

```
##
## Attaching package: 'discrim'
##
## The following object is masked from 'package:dials':
##
##     smoothness
```

```
library(poissonreg)
library(corrr)
library(klaR)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:ISLR2':
##
##     Boston
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggthemes)
tidymodels_prefer()
```

Data Cleaning

```
msocf22 <- read_csv("ucsb_msoc_fall2022.csv") %>%
  clean_names()
```

```
## Rows: 1470 Columns: 41
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (8): Event Date, Event Description, Player Name, Event Type, Event Loca...
## dbl (25): Performance Duration [min], Total Distance [m], Walk Distance [m],...
## lgl  (8): HR Mean [bpm], HR Max [bpm], HR Grey Zone [%], HR Blue Zone [%], H...
##
## 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.
```

```
msocf22 %>%
  filter(event_result != "No Result")
```

```
## # A tibble: 596 x 41
##    event_date event_de~1 playe~2 event~3 event~4 event~5 event~6 segme~7 perfo~8
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>
##  1 8/14/2022  San Jose ~ Athlet~ Game    Away    WIN     preSea~ *           125
##  2 8/20/2022  Westmont   Athlet~ Game    Home    WIN     inSeas~ *           117
##  3 8/25/2022  Missouri ~ Athlet~ Game    Away    DRAW    inSeas~ *           122
##  4 8/28/2022  Californi~ Athlet~ Game    Home    WIN     inSeas~ *           122
##  5 8/28/2022  Californi~ Athlet~ Game    Home    WIN     inSeas~ 1st Ha~      50
##  6 8/28/2022  Californi~ Athlet~ Game    Home    WIN     inSeas~ 2nd Ha~      56
##  7 9/2/2022   Cornell    Athlet~ Game    Home    LOSS    inSeas~ *           126
##  8 9/2/2022   Cornell    Athlet~ Game    Home    LOSS    inSeas~ 1st Ha~      54
##  9 9/2/2022   Cornell    Athlet~ Game    Home    LOSS    inSeas~ 2nd Ha~      56
## 10 9/4/2022   Loyola Ma~ Athlet~ Game    Away    WIN     inSeas~ *           128
## # ... with 586 more rows, 32 more variables: total_distance_m <dbl>,
## #   walk_distance_m <dbl>, jog_distance_m <dbl>, run_distance_m <dbl>,
## #   sprint_distance_m <dbl>, sprint_efforts <dbl>, zone_1_distance_m <dbl>,
## #   zone_2_distance_m <dbl>, zone_3_distance_m <dbl>, zone_4_distance_m <dbl>,
## #   zone_5_distance_m <dbl>, zone_6_distance_m <dbl>, zone_7_distance_m <dbl>,
## #   zone_8_distance_m <dbl>, hard_running_m <dbl>, hard_running_efforts <dbl>,
## #   work_rate_m_min <dbl>, top_speed_m_s <dbl>, intensity <dbl>, ...
```

```
msocf22 %>%
  # filter(event_result != "No Result") %>%
  group_by(event_date)
```

```
## # A tibble: 1,470 x 41
## # Groups:   event_date [46]
##    event_date event_de~1 playe~2 event~3 event~4 event~5 event~6 segme~7 perfo~8
```

```
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>
##  1 8/9/2022   Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            83
##  2 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *            97
##  3 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Warm-Up       9
##  4 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Accel ~      12
##  5 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Practi~      76
##  6 8/10/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *           115
##  7 8/10/2022  Morning P~ Athlet~ Traini~ Other   No Res~ preSea~ *            90
##  8 8/11/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            75
##  9 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *           110
## 10 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Speed ~      27
## # ... with 1,460 more rows, 32 more variables: total_distance_m <dbl>,
## #   walk_distance_m <dbl>, jog_distance_m <dbl>, run_distance_m <dbl>,
## #   sprint_distance_m <dbl>, sprint_efforts <dbl>, zone_1_distance_m <dbl>,
## #   zone_2_distance_m <dbl>, zone_3_distance_m <dbl>, zone_4_distance_m <dbl>,
## #   zone_5_distance_m <dbl>, zone_6_distance_m <dbl>, zone_7_distance_m <dbl>,
## #   zone_8_distance_m <dbl>, hard_running_m <dbl>, hard_running_efforts <dbl>,
## #   work_rate_m_min <dbl>, top_speed_m_s <dbl>, intensity <dbl>, ...
```

```
#take out halftime
```

The clean_names() function here is very useful because it puts every column title in snake case meaning each word is connected with an underscore and everything is lowercase. This will make it much easier later on when we have to call different columns in r. The data had broken every game into three different events, first half, second half, and combined game, to avoid having all three of these we filtered it so that only the full games would be available for data analysis avoiding errors that would be caused to replicates.

```
msocf22 <- msocf22%>%
  mutate(total_impact = impact_light + impact_medium+impact_heavy)
msocf22
```

```
## # A tibble: 1,470 x 42
##    event_date event_de~1 playe~2 event~3 event~4 event~5 event~6 segme~7 perfo~8
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>
##  1 8/9/2022   Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            83
##  2 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *            97
##  3 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Warm-Up       9
##  4 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Accel ~      12
##  5 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Practi~      76
##  6 8/10/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *           115
##  7 8/10/2022  Morning P~ Athlet~ Traini~ Other   No Res~ preSea~ *            90
##  8 8/11/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            75
##  9 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *           110
## 10 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ Speed ~      27
## # ... with 1,460 more rows, 33 more variables: total_distance_m <dbl>,
## #   walk_distance_m <dbl>, jog_distance_m <dbl>, run_distance_m <dbl>,
## #   sprint_distance_m <dbl>, sprint_efforts <dbl>, zone_1_distance_m <dbl>,
## #   zone_2_distance_m <dbl>, zone_3_distance_m <dbl>, zone_4_distance_m <dbl>,
## #   zone_5_distance_m <dbl>, zone_6_distance_m <dbl>, zone_7_distance_m <dbl>,
## #   zone_8_distance_m <dbl>, hard_running_m <dbl>, hard_running_efforts <dbl>,
## #   work_rate_m_min <dbl>, top_speed_m_s <dbl>, intensity <dbl>, ...
```

```r
#msocf22 <- msocf22 %>%
 # mutate(total_impact = factor(total_impact))
#msocf22
```

The outcome variable total_impact did not come in the given data, so we had to mutate a column in for it into the data set. The total_impact is the sum of impact_light, impact_medium and impact_heavy.

```r
msocf22 %>%
  filter(segment_name == "*")
```

```
## # A tibble: 1,040 x 42
##    event_date event_de~1 playe~2 event~3 event~4 event~5 event~6 segme~7 perfo~8
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>
##  1 8/9/2022   Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            83
##  2 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *            97
##  3 8/10/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *           115
##  4 8/10/2022  Morning P~ Athlet~ Traini~ Other   No Res~ preSea~ *            90
##  5 8/11/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            75
##  6 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *           110
##  7 8/12/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ preSea~ *            98
##  8 8/12/2022  Morning P~ Athlet~ Traini~ Home    No Res~ preSea~ *           124
##  9 8/13/2022  Practice   Athlet~ Traini~ Home    No Res~ preSea~ *            95
## 10 8/14/2022  San Jose ~ Athlet~ Game    Away    WIN     preSea~ *           125
## # ... with 1,030 more rows, 33 more variables: total_distance_m <dbl>,
## #   walk_distance_m <dbl>, jog_distance_m <dbl>, run_distance_m <dbl>,
## #   sprint_distance_m <dbl>, sprint_efforts <dbl>, zone_1_distance_m <dbl>,
## #   zone_2_distance_m <dbl>, zone_3_distance_m <dbl>, zone_4_distance_m <dbl>,
## #   zone_5_distance_m <dbl>, zone_6_distance_m <dbl>, zone_7_distance_m <dbl>,
## #   zone_8_distance_m <dbl>, hard_running_m <dbl>, hard_running_efforts <dbl>,
## #   work_rate_m_min <dbl>, top_speed_m_s <dbl>, intensity <dbl>, ...
```

```r
# we want to make sure we don't account for half time just the whole game
msocf22 <- msocf22 %>%
  select(-c(starts_with("hr")))
msocf22 <- msocf22 %>%
  select (-c(impact_light,impact_medium,impact_heavy,event_tags,total_distance_m)) # each impact factor
#msocf22 <- msocf22 %>%
  #select(-c(total_distance_m))
#msocf22
```

Again to deal with the replicate rows that account for each games 1st half, 2nd half and full game we want to filter our data to only include events with segment name "*" because this column includes "1st Half" and "2nd Half". By getting rid of the first and second half we successfully keep the full game data points in our data set. Next we want to take out all the columns starting with "hr" because they are all empty and this way our data will look more concise(same with "event_tags". Lastly we must take out rows "impact_light", "impact_heavy", and "impact_medium" because the sum of these three columns adds up to our outcome. We do not want this in our data set because it would make the data colinear and ruin the predictions we are trying to get by fitting the models later. #IS CONLINEAR CORRECRT The total_distance_m also has to be taken out because it is just the sum of walk_distance_m, jog_distance_m, run_distance_m, and sprint_distance_m.

```r
msocf22 #one last check up on the data
```

```
## # A tibble: 1,470 x 29
##    event_date event_de~1 playe~2 event~3 event~4 event~5 segme~6 perfo~7 walk_~8
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>   <dbl>
##  1 8/9/2022   Afternoon~ Athlet~ Traini~ Home    No Res~ *            83    840.
##  2 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ *            97   1176.
##  3 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Warm-Up       9    125.
##  4 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Accel ~      12    189.
##  5 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Practi~      76    887.
##  6 8/10/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ *           115   1377.
##  7 8/10/2022  Morning P~ Athlet~ Traini~ Other   No Res~ *            90   1033.
##  8 8/11/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ *            75    860.
##  9 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ *           110   1243.
## 10 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ Speed ~      27    343.
## # ... with 1,460 more rows, 20 more variables: jog_distance_m <dbl>,
## #   run_distance_m <dbl>, sprint_distance_m <dbl>, sprint_efforts <dbl>,
## #   zone_1_distance_m <dbl>, zone_2_distance_m <dbl>, zone_3_distance_m <dbl>,
## #   zone_4_distance_m <dbl>, zone_5_distance_m <dbl>, zone_6_distance_m <dbl>,
## #   zone_7_distance_m <dbl>, zone_8_distance_m <dbl>, hard_running_m <dbl>,
## #   hard_running_efforts <dbl>, work_rate_m_min <dbl>, top_speed_m_s <dbl>,
## #   intensity <dbl>, load_2d <dbl>, load_3d <dbl>, total_impact <dbl>, and ...
```

Data Splitting

```r
set.seed(5555) #so that the same random variables are used when we re run the code

msocf22_split <- initial_split(msocf22, prop = 0.70,
                               strata = total_impact) #stratifying on the outcome variable
msocf22_train <- training(msocf22_split)
msocf22_test <- testing(msocf22_split)

dim(msocf22_train)/nrow(msocf22) #checking if the proportions of the split data are correct
```

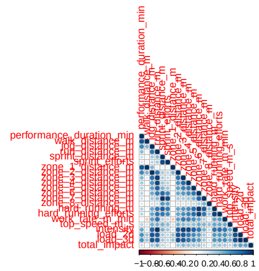
```
## [1] 0.69863946 0.01972789
```

```r
dim(msocf22_test)/nrow(msocf22)
```

```
## [1] 0.30136054 0.01972789
```

We want to split our data into a testing and training data set stratifying on the outcome. We stratify on outcome because we would like to maintain the distribution of the outcome for all the resamples. In the first line of this code block we set.seed() so that the same random variables are used every time we rerun the code.

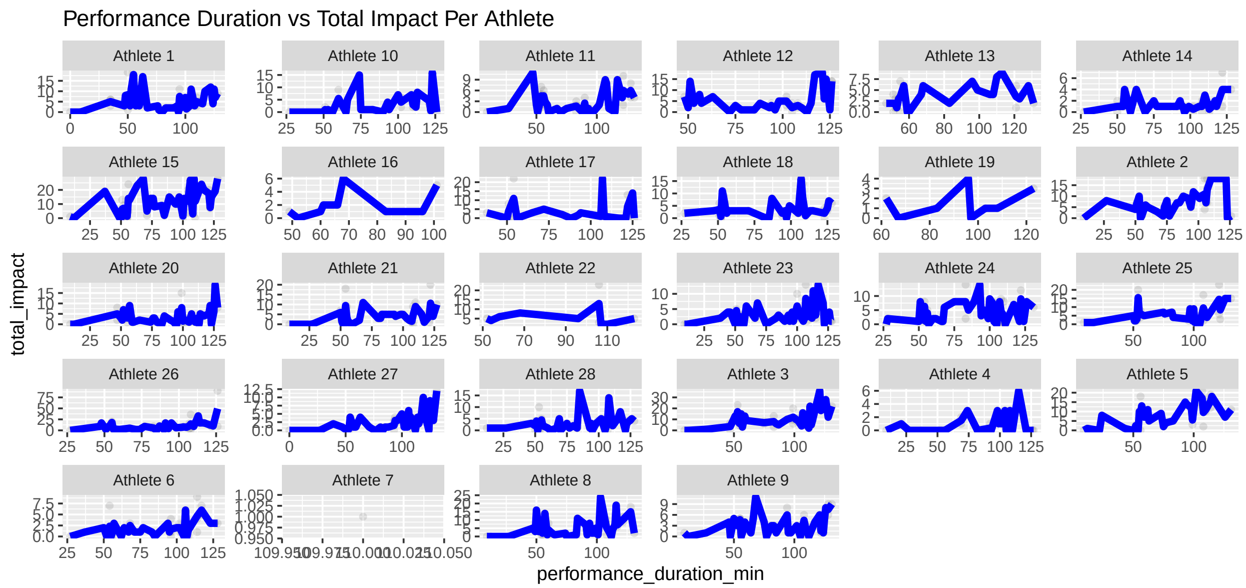
```r
library(corrplot)

msocf22_train %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot(type = "lower")
```



Here we construct a huge correlation plot with all the possible relevant predictors so far to see if anything stands out and if we should be cautious of that. The first thing that should catch your eyes is that "jog_distance_m" and "zone_2_distance_m" have a perfect positive correlation and so does "walk_distance_m" and "zone_distance_1". This can be explained by the fact that these zone speeds are the exact same speeds needed for a player to fall into either the jog, walk, run, or sprint categories. This is helpful to us because we can determine that our recipe does not need each zone distance because that speed is already accounted for within "jog_distance_m", "run_distance_m", "sprint_distance_m", and "walk_distance_m". However, this also opens the possibility to creating interactions between the predictors to again create a more condense model and not waste time looking at a predictor that is already within another one of our predictors. Lastly one should note that we only have positive correlations because every predictor we are looking at has to do with running, speed and efforts put in. Generally the more effort you put in the faster you go explaining most of the positive correlations we are witnessing here. There is no pair of predictors here where doing worse for one will be better for the other. The seed that should be planted into your mind is that if each zone is the speef for each style of running and each style of running summed up is the total distance that complex relationships are we dealing with within our data set? #fix correlation with pca in the recipe

```
msocf22_train %>%
  ggplot(aes(performance_duration_min, total_impact))+ #wish to see the relationship between performanc
  geom_point(alpha=0.1) +
  stat_summary(fun=mean, colour="blue",geom = "line",size = 2)+
  facet_wrap(~player_name,scales="free")+ # i'd like a plot per player so we put "~" player to get all
  labs(
    title = "Performance Duration vs Total Impact Per Athlete"
  )
```

```
## geom_path: Each group consists of only one observation. Do you need to adjust
## the group aesthetic?
```



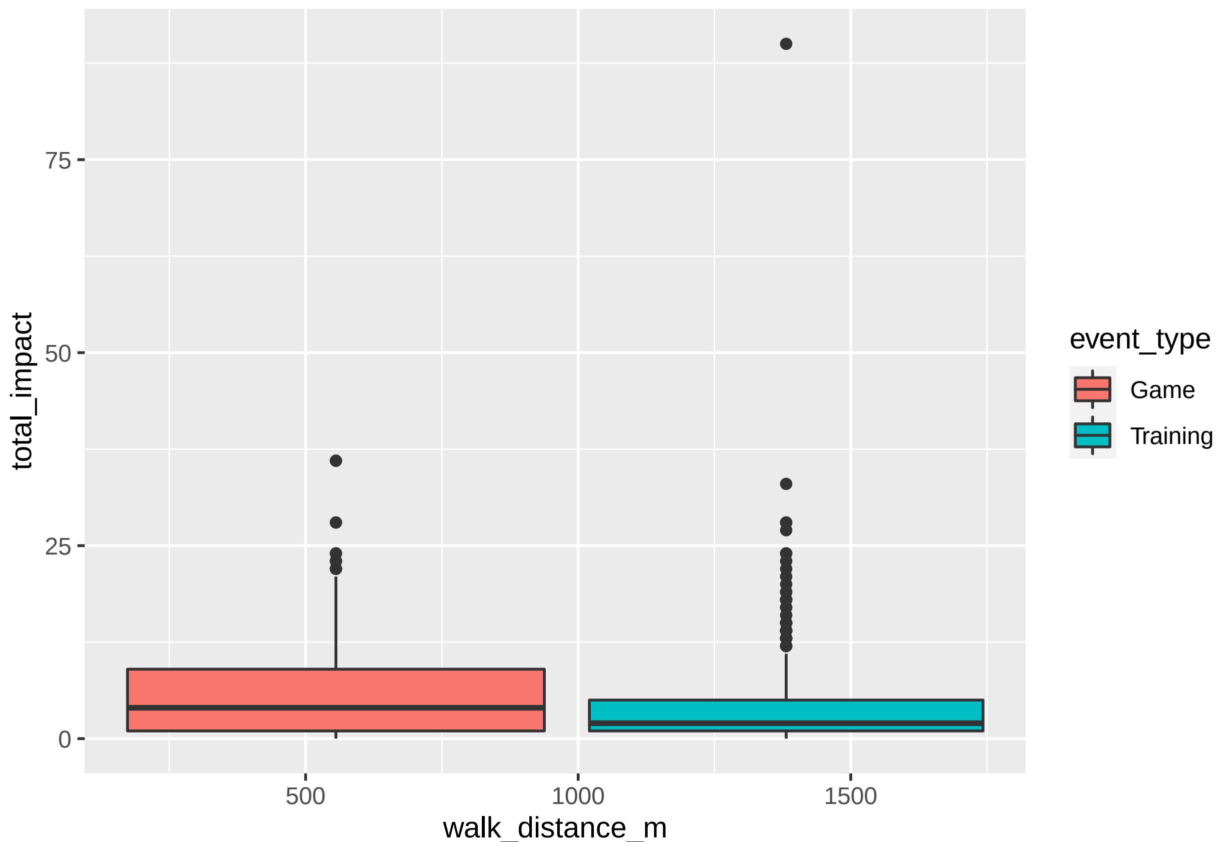
This part of EDA showed us that Athlete 7 has lots of missing data so we should not really take them into account. This is why exploratory data analysis is helpful because we do not have to scrummage through all the data to identity missing data. The reason I wanted to see this plot is mainly because these two predictors had a surprisingly low correlation value of .23 when I expected them not to. This way we can see in detail what typically happens when each player plays for longer. A lot of the plots show their peak impact values roughly around 50 minutes. This to me means that when a player is put into a game they are able to perform best and at their maximum ability if they are not forced to play the entire 90 minute game. The same goes for practice duration.

```
msocf22 <-msocf22[-c(358),]
msocf22
```

```
## # A tibble: 1,469 x 29
##    event_date event_de~1 playe~2 event~3 event~4 event~5 segme~6 perfo~7 walk_~8
##    <chr>      <chr>      <chr>   <chr>   <chr>   <chr>   <chr>     <dbl>   <dbl>
##  1 8/9/2022   Afternoon~ Athlet~ Traini~ Home    No Res~ *            83    840.
##  2 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ *            97   1176.
##  3 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Warm-Up       9    125.
##  4 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Accel ~      12    189.
##  5 8/9/2022   Morning P~ Athlet~ Traini~ Home    No Res~ Practi~      76    887.
##  6 8/10/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ *           115   1377.
##  7 8/10/2022  Morning P~ Athlet~ Traini~ Other   No Res~ *            90   1033.
##  8 8/11/2022  Afternoon~ Athlet~ Traini~ Home    No Res~ *            75    860.
##  9 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ *           110   1243.
## 10 8/11/2022  Morning P~ Athlet~ Traini~ Home    No Res~ Speed ~      27    343.
## # ... with 1,459 more rows, 20 more variables: jog_distance_m <dbl>,
## #   run_distance_m <dbl>, sprint_distance_m <dbl>, sprint_efforts <dbl>,
## #   zone_1_distance_m <dbl>, zone_2_distance_m <dbl>, zone_3_distance_m <dbl>,
## #   zone_4_distance_m <dbl>, zone_5_distance_m <dbl>, zone_6_distance_m <dbl>,
## #   zone_7_distance_m <dbl>, zone_8_distance_m <dbl>, hard_running_m <dbl>,
## #   hard_running_efforts <dbl>, work_rate_m_min <dbl>, top_speed_m_s <dbl>,
## #   intensity <dbl>, load_2d <dbl>, load_3d <dbl>, total_impact <dbl>, and ...
```

After the last plot we saw that Athlete 7 does not have many data points indicating they left the team or were injured, so their data is not necessarily useful to us.

```r
#not my fave
ggplot(msocf22_train, aes(x=walk_distance_m, y=total_impact, fill=event_type)) +
  geom_boxplot()
```




```r
# add another one here
```

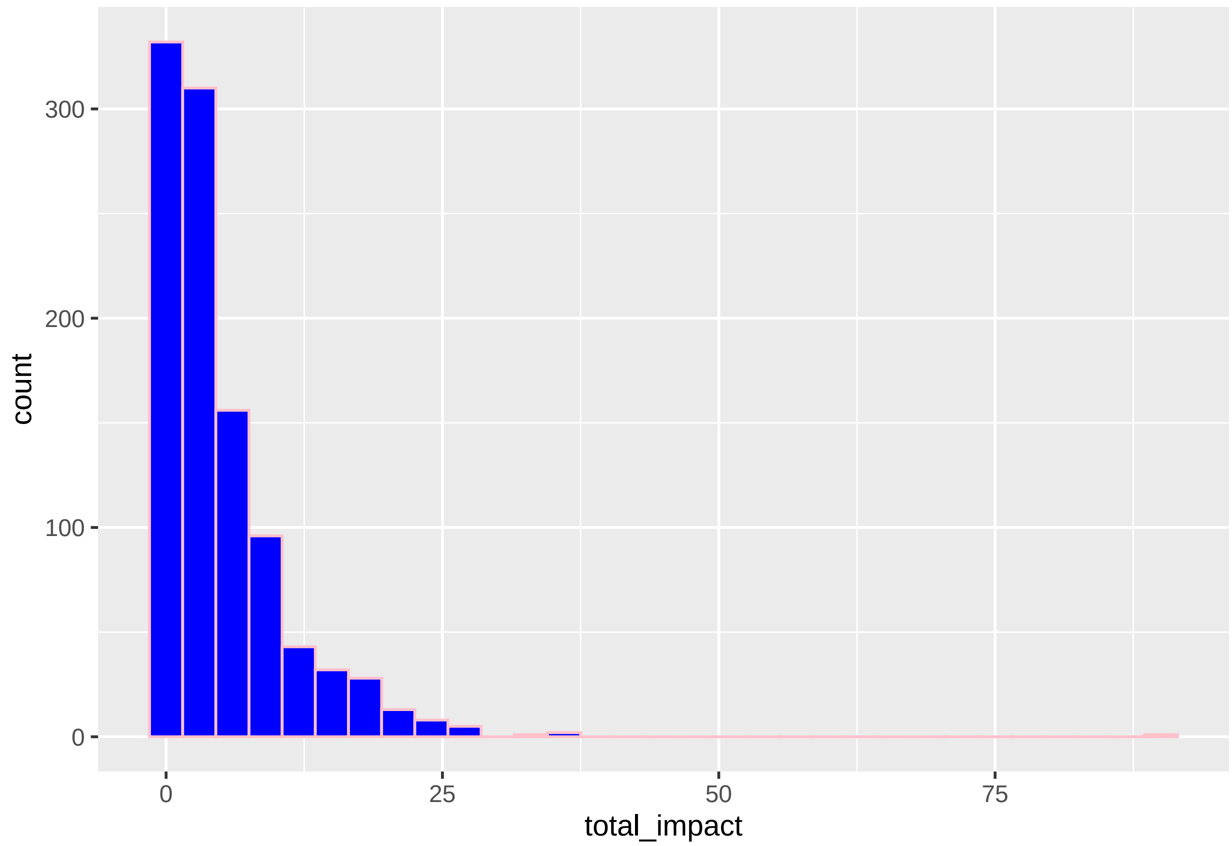
The relationship between This bar plot was mainly to see if players are using training as more of a time to heal and relax their bodies keeping it low impact or not. The means here are pretty similar which can be explained by the extended duration time during a practice versus game making the mean distance similar because of how much longer they are out on the field for training. However the range of the game boxplot makes sense because we would assume that players are covering more distance walking during a game than at practice when they have the chance to stand. I believe the range is also associated with defenders at ucsb because they are put in a position where they can stand/walk around more because our team is usually playing on the attacking end.

```r
msocf22_train$total_impact %>%
  table()
```

```
## .
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
## 180 152 119 106  85  67  42  47  39  36  21  17  11  15  14  12   6  10  12   6
##  20  21  22  23  24  27  28  33  36  90
##   7   2   4   4   4   1   4   1   2   1
```

```
#check out the distribution
```

```
outcome_hist <- ggplot(msocf22_train,aes(x=total_impact)) +
  geom_histogram(color="pink",fill = "blue",binwidth = 3) #making bins more legible
outcome_hist
```



This histogram of our outcome is uni-model and right-sckew. The distribution makes sense because total impact is dependent on how much players are on the field and how hard they choose to go. Our tracker's metrics make it normal to have a lower total_impact because anything too high would mean the player is going out of their comfort zone/ are not accumulated to the workout probably leaving them sore the next day. This is not what we want in order to prevent injury, that is why it is important for us to track things like this to make sure players have a steady and healthy increase of action during games and practices. #what it looks like, average value etc positively sckwed

```
msocf22_recipe <- recipe(total_impact ~ performance_duration_min + walk_distance_m + jog_distance_m+
                            run_distance_m + sprint_distance_m + sprint_efforts + hard_running_m +  work
  step_normalize(all_predictors())%>%
  step_dummy(all_nominal_predictors()) %>%
 step_pca(walk_distance_m,zone_1_distance_m,jog_distance_m,zone_2_distance_m,run_distance_m,zone_3_dista
msocf22_recipe %>%
  prep() %>%
  juice()
```

```
## # A tibble: 1,027 x 14
##    performance~1 sprin~2 hard_~3 work_~4 top_s~5 inten~6 zone_~7 zone_~8 zone_~9
```

```
##            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1        -2.52   -0.558  -0.987 -0.322  -2.50   -1.52   -1.04   -0.964   -0.370
##  2        -0.221   0.271   0.506  0.0995  0.590   0.184   0.214   0.387   -0.154
##  3        -0.120  -0.558  -0.720 -0.658   0.0145 -0.556  -0.714  -0.671   -0.370
##  4         0.590  -0.558  -0.515 -0.608  -0.177  -0.488  -0.629  -0.413   -0.370
##  5        -1.13    1.93    0.160  0.798   0.329  -0.110  -0.0900  0.0497  -0.370
##  6         0.353   1.93    0.556 -0.394   0.843  -0.266  -0.0593  0.747    0.297
##  7        -2.82   -0.558  -0.987 -2.67   -5.65   -1.62   -1.06   -0.964   -0.370
##  8        -2.52   -0.558  -0.980 -0.236  -1.80   -1.52   -1.04   -0.964   -0.370
##  9         1.33    1.10    0.340 -0.545   0.306   0.0599 -0.0938  0.227   -0.370
## 10        -0.390  -0.558  -0.514 -0.498  -0.369  -0.731  -0.479  -0.403   -0.370
## # ... with 1,017 more rows, 5 more variables: total_impact <dbl>, PC1 <dbl>,
## #   PC2 <dbl>, PC3 <dbl>, PC4 <dbl>, and abbreviated variable names
## #   1: performance_duration_min, 2: sprint_efforts, 3: hard_running_m,
## #   4: work_rate_m_min, 5: top_speed_m_s, 6: intensity, 7: zone_5_distance_m,
## #   8: zone_6_distance_m, 9: zone_8_distance_m
```

```
#leaving loads and zone distances out, they are speed of each zone and load 2d and 3d are irrelevant to
#step pca
#explain the relationships why they are correlated and why you did four
```

```
#fold data
msocf22_fold <- vfold_cv(msocf22_train,v=10)
msocf22_fold
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits            id
##    <list>            <chr>
##  1 <split [924/103]> Fold01
##  2 <split [924/103]> Fold02
##  3 <split [924/103]> Fold03
##  4 <split [924/103]> Fold04
##  5 <split [924/103]> Fold05
##  6 <split [924/103]> Fold06
##  7 <split [924/103]> Fold07
##  8 <split [925/102]> Fold08
##  9 <split [925/102]> Fold09
## 10 <split [925/102]> Fold10
```

```
#linear regression model
msocf22_recipe_lr <- recipe(total_impact ~ performance_duration_min + walk_distance_m + jog_distance_m+
                           run_distance_m + sprint_distance_m + sprint_efforts + hard_running_m + work_
  step_dummy(all_nominal_predictors())

lm_model<- linear_reg() %>%
  set_engine("lm")
lm_wkflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(msocf22_recipe)

lm_fit <- fit(lm_wkflow,msocf22_train)
```

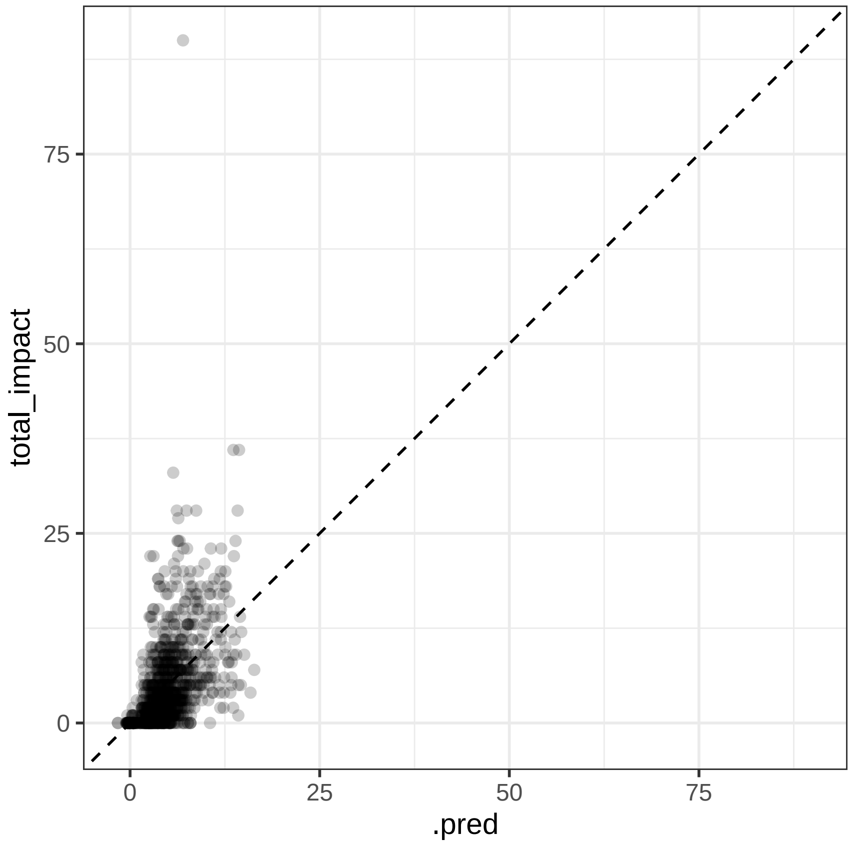
```r
msocf22_train_res <- predict(lm_fit, new_data = msocf22_train %>% select(-total_impact))
msocf22_train_res %>%
  head()
```

```
## # A tibble: 6 x 1
##   .pred
##   <dbl>
## 1 0.279
## 2 5.68
## 3 3.19
## 4 3.70
## 5 4.60
## 6 3.99
```

```r
msocf22_train_res <- bind_cols(msocf22_train_res, new_data = msocf22_train %>% select(total_impact))
msocf22_train_res %>%
  head()
```

```
## # A tibble: 6 x 2
##   .pred total_impact
##   <dbl>        <dbl>
## 1 0.279            0
## 2 5.68             1
## 3 3.19             0
## 4 3.70             1
## 5 4.60             1
## 6 3.99             0
```

```r
msocf22_train_res %>%
  ggplot(aes(x=.pred,y=total_impact)) +
  geom_point(alpha = 0.2)+
  geom_abline(lty=2)+
  theme_bw()+
  coord_obs_pred()
```

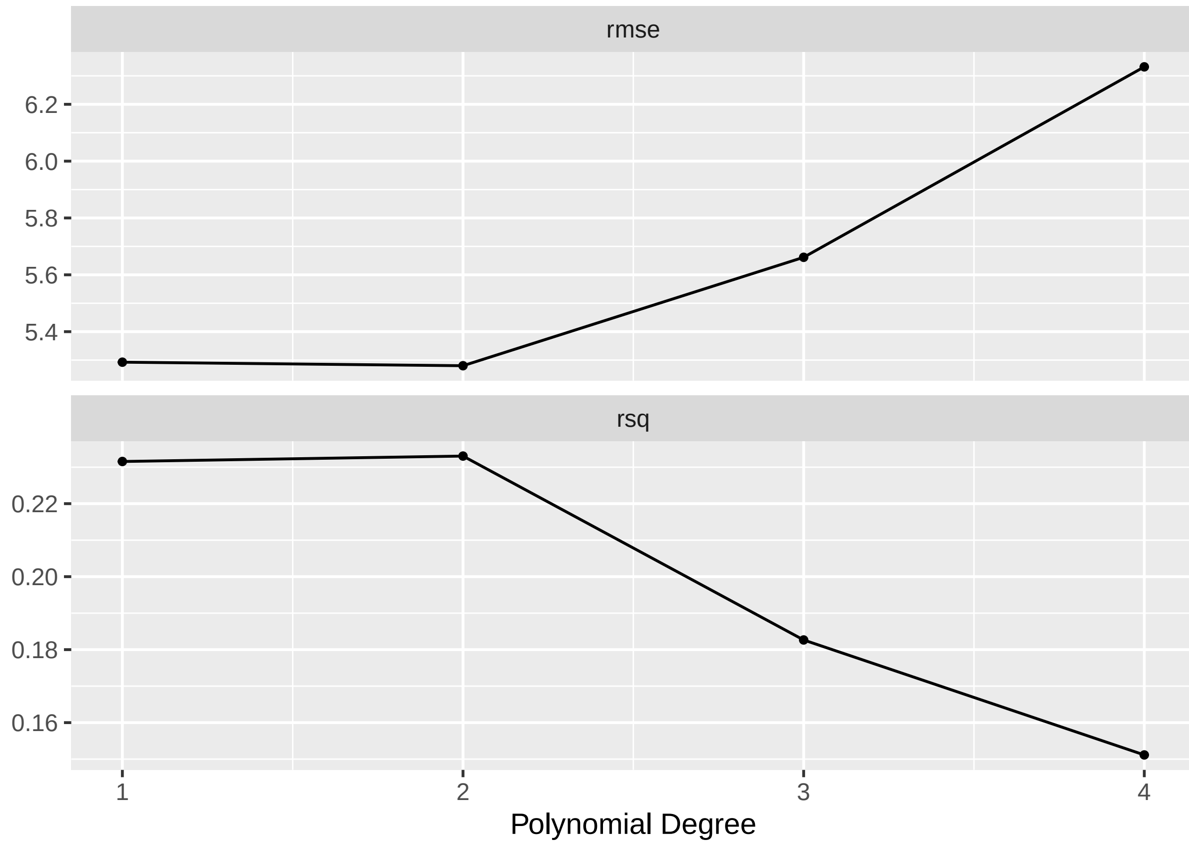



```
lm_train_rmse <- sqrt(mean((msocf22_train_res$total_impact - msocf22_train_res$.pred)^2))
lm_train_rmse
```

```
## [1] 5.371054
```

```
#polynomial regression
poly_rec <-  msocf22_recipe %>%
  step_poly(all_predictors(),degree=tune())


lm_spec <- linear_reg()%>%
  set_mode("regression")%>%
  set_engine("lm")

poly_wkflow <- workflow() %>%
  add_model(lm_spec)  %>%
  add_recipe(poly_rec)

degree_grid_poly <- grid_regular(degree(range=c(1,4)),levels=4)


tune_res_poly <- tune_grid(
  poly_wkflow,
  resamples = msocf22_fold,
  grid = degree_grid_poly
```

```
)
autoplot(tune_res_poly)
```




```
#select best

best_poly <- select_best(tune_res_poly, metric="rsq")

poly_final <- finalize_workflow(poly_wkflow,best_poly)

poly_final_fit <- fit(poly_final, data = msocf22_train)

poly_train_rmse <- augment(poly_final_fit, new_data = msocf22_train) %>%
  rmse(truth = total_impact, estimate = .pred)
poly_train_rmse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        5.31
```

```
augment(poly_final_fit, new_data = msocf22_test) %>%
  rmse(truth = total_impact, estimate = .pred)
```

```
## # A tibble: 1 x 3
```

```
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        10.1
```

```
#ridge regression
ridge_recipe <-
  recipe(formula = total_impact ~., data = msocf22_train) %>%
  step_novel(all_nominal_predictors())%>%
  step_dummy(all_nominal_predictors())%>%
  step_zv(all_predictors())%>%
  step_normalize(all_predictors())

ridge_spec <-
  linear_reg(penalty = tune(), mixture = 0)%>%
  set_mode("regression")%>%
  set_engine("glmnet")
ridge_workflow <- workflow() %>%
  add_recipe(ridge_recipe)%>%
  add_model(ridge_spec)
penalty_grid <- grid_regular(penalty(range=c(-5,5)),levels=50)
penalty_grid
```

```
## # A tibble: 50 x 1
##      penalty
##        <dbl>
##  1 0.00001
##  2 0.0000160
##  3 0.0000256
##  4 0.0000409
##  5 0.0000655
##  6 0.000105
##  7 0.000168
##  8 0.000268
##  9 0.000429
## 10 0.000687
## # ... with 40 more rows
```

```
tune_res <- tune_grid(
  ridge_workflow,
  resamples = msocf22_fold,
  grid = penalty_grid
)
```

```
## ! Fold01: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold02: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold03: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold04: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold05: internal: A correlation computation is required, but `estimate` is constant and ha...
```

```
## ! Fold06: internal: A correlation computation is required, but ‘estimate‘ is constant and ha...
```

```
## ! Fold07: internal: A correlation computation is required, but ‘estimate‘ is constant and ha...
```

```
## ! Fold08: internal: A correlation computation is required, but ‘estimate‘ is constant and ha...
```

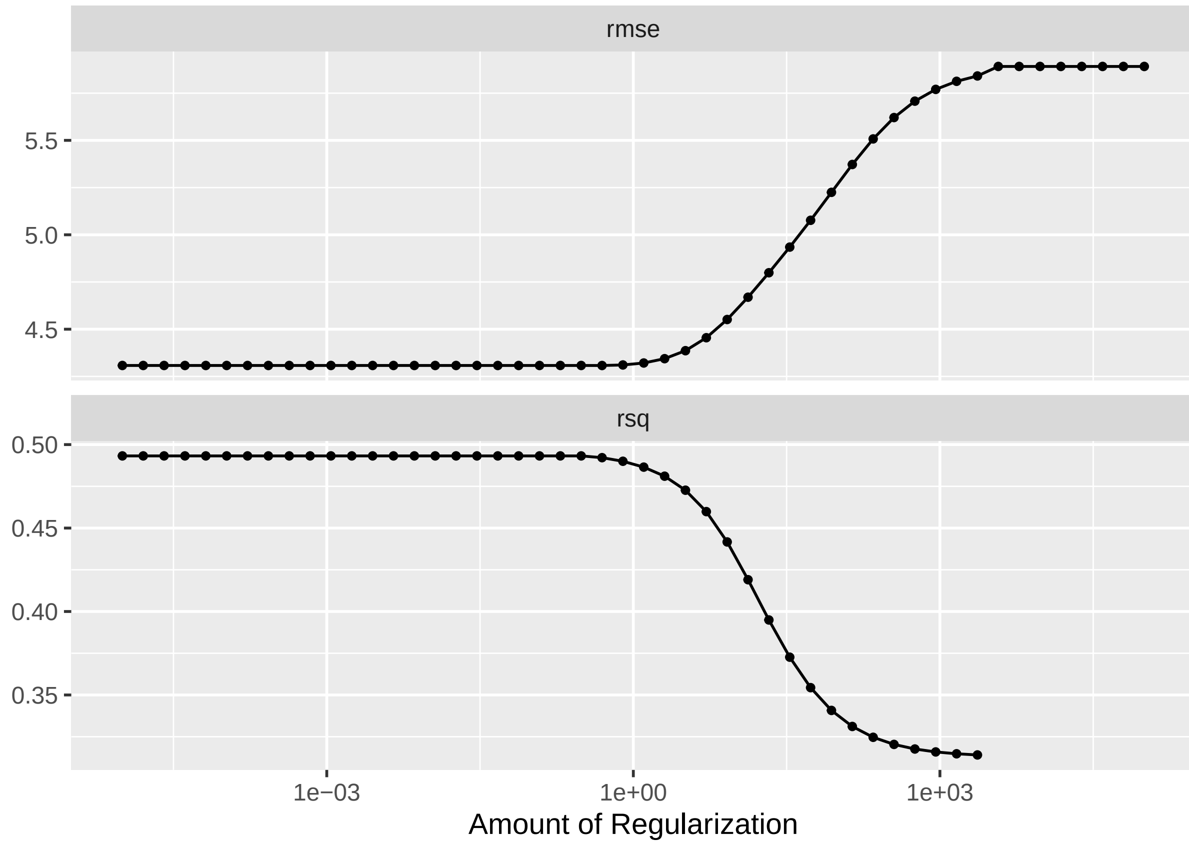
```
## ! Fold09: internal: A correlation computation is required, but ‘estimate‘ is constant and ha...
```

```
## ! Fold10: internal: A correlation computation is required, but ‘estimate‘ is constant and ha...
```

```
tune_res
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits            id     .metrics           .notes
##    <list>            <chr>  <list>             <list>
##  1 <split [924/103]> Fold01 <tibble [100 x 5]> <tibble [1 x 3]>
##  2 <split [924/103]> Fold02 <tibble [100 x 5]> <tibble [1 x 3]>
##  3 <split [924/103]> Fold03 <tibble [100 x 5]> <tibble [1 x 3]>
##  4 <split [924/103]> Fold04 <tibble [100 x 5]> <tibble [1 x 3]>
##  5 <split [924/103]> Fold05 <tibble [100 x 5]> <tibble [1 x 3]>
##  6 <split [924/103]> Fold06 <tibble [100 x 5]> <tibble [1 x 3]>
##  7 <split [924/103]> Fold07 <tibble [100 x 5]> <tibble [1 x 3]>
##  8 <split [925/102]> Fold08 <tibble [100 x 5]> <tibble [1 x 3]>
##  9 <split [925/102]> Fold09 <tibble [100 x 5]> <tibble [1 x 3]>
## 10 <split [925/102]> Fold10 <tibble [100 x 5]> <tibble [1 x 3]>
##
## There were issues with some computations:
##
##   - Warning(s) x10: A correlation computation is required, but ‘estimate‘ is constant...
##
## Run ‘show_notes(.Last.tune.result)‘ for more information.
```

```
autoplot(tune_res)
```




```r
#plot of the ridge
#working on finding the best penalty for our data
collect_metrics(tune_res)
```

```
## # A tibble: 100 x 7
##      penalty .metric .estimator  mean     n std_err .config
##        <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1 0.00001   rmse    standard   4.31     10  0.534  Preprocessor1_Model01
##  2 0.00001   rsq     standard   0.493    10  0.0363 Preprocessor1_Model01
##  3 0.0000160 rmse    standard   4.31     10  0.534  Preprocessor1_Model02
##  4 0.0000160 rsq     standard   0.493    10  0.0363 Preprocessor1_Model02
##  5 0.0000256 rmse    standard   4.31     10  0.534  Preprocessor1_Model03
##  6 0.0000256 rsq     standard   0.493    10  0.0363 Preprocessor1_Model03
##  7 0.0000409 rmse    standard   4.31     10  0.534  Preprocessor1_Model04
##  8 0.0000409 rsq     standard   0.493    10  0.0363 Preprocessor1_Model04
##  9 0.0000655 rmse    standard   4.31     10  0.534  Preprocessor1_Model05
## 10 0.0000655 rsq     standard   0.493    10  0.0363 Preprocessor1_Model05
## # ... with 90 more rows
```

```r
best_penalty_rsq <- select_best(tune_res, metric = "rsq")
best_penalty_rsq
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
```

```
## 1 0.00001 Preprocessor1_Model01
```

```
best_penalty_rmse <- select_best(tune_res,metric="rmse")
best_penalty_rmse
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1   0.494 Preprocessor1_Model24
```

```
ridge_final <- finalize_workflow(ridge_workflow,best_penalty_rsq)

ridge_final_fit <- fit(ridge_final, data = msocf22_train)
ridge_train_rmse <- augment(ridge_final_fit, new_data = msocf22_train) %>%
  rmse(truth = total_impact, estimate = .pred)
ridge_train_rmse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        4.20
```

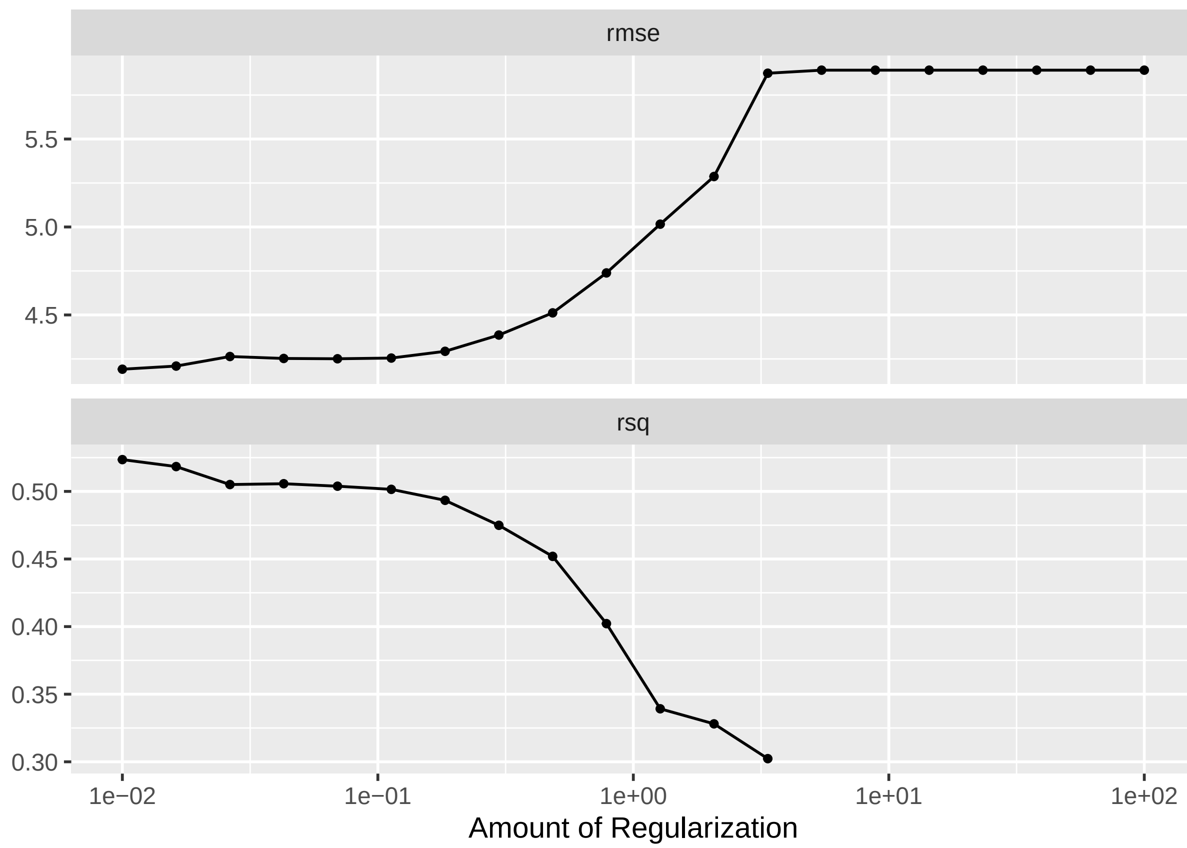
```
augment(ridge_final_fit, new_data = msocf22_test) %>%
  rmse(truth = total_impact, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        9.27
```

```
lasso_recipe <-
  recipe(formula = total_impact~., data = msocf22_train) %>%
  step_novel(all_nominal_predictors())%>%
  step_dummy(all_nominal_predictors())%>%
  step_zv(all_predictors())%>%
  step_normalize(all_predictors())

lasso_spec <-
  linear_reg(penalty = tune(),mixture = 1)%>%
  set_mode("regression")%>%
  set_engine("glmnet")

lasso_workflow <- workflow() %>%
  add_recipe(lasso_recipe)%>%
  add_model(lasso_spec)

penalty_grid <- grid_regular(penalty(range = c(-2,2)),levels=20)

tune_res <- tune_grid(
  lasso_workflow,
  resamples = msocf22_fold,
  grid = penalty_grid
)
```

```
## ! Fold01: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold02: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold03: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold04: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold05: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold06: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold07: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold08: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold09: internal: A correlation computation is required, but `estimate` is constant and ha...

## ! Fold10: internal: A correlation computation is required, but `estimate` is constant and ha...
```

```
autoplot(tune_res)
```



```
best_penalty <- select_best(tune_res, metric="rmse")

lasso_final <- finalize_workflow(lasso_workflow,best_penalty)

lasso_final_fit <- fit(lasso_final, data = msocf22_train)
lasso_train_rmse <- augment(lasso_final_fit, new_data = msocf22_train) %>%
  rmse(truth = total_impact, estimate = .pred)
lasso_train_rmse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        4.09
```

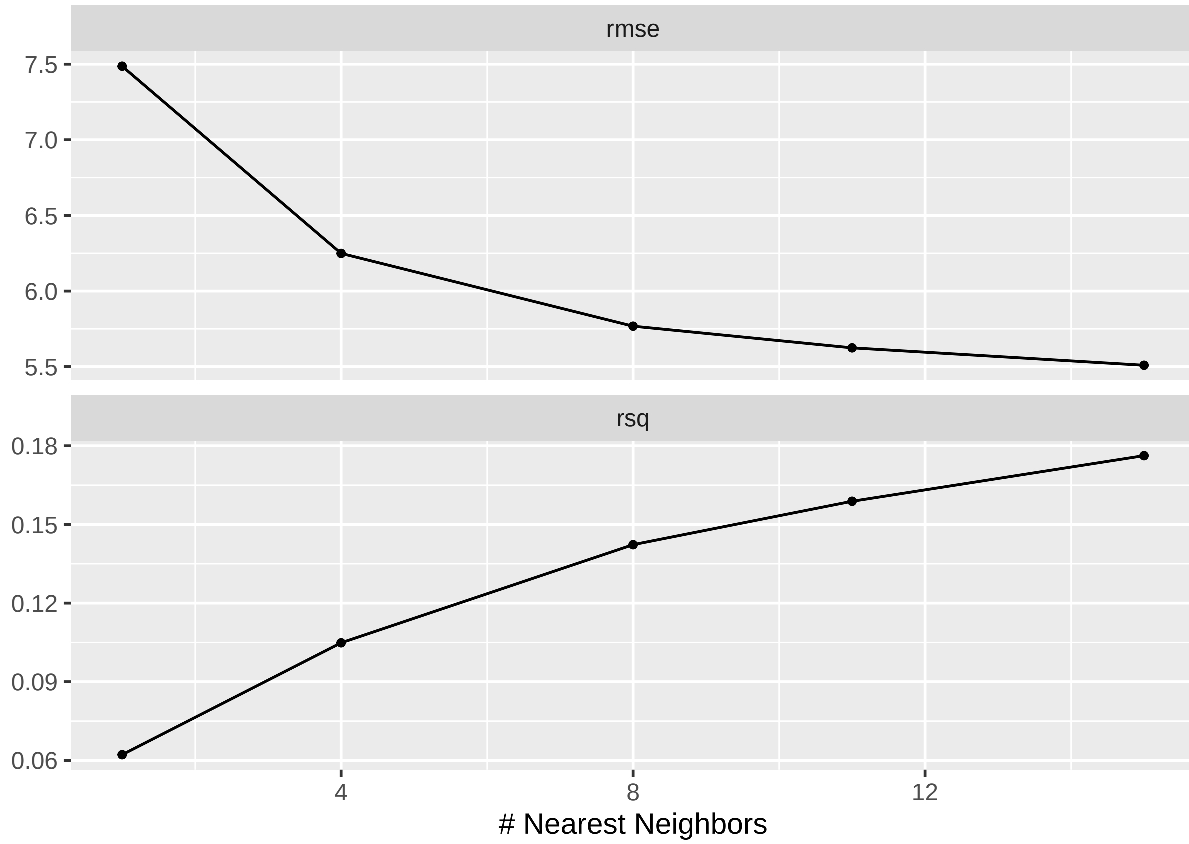
```
augment(lasso_final_fit, new_data = msocf22_test) %>%
  rmse(truth = total_impact, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        8.78
```

```
#k-nearest neighbor
#install.packages("kknn")
knn_model <-
  nearest_neighbor(
    neighbors = tune(),
    mode = "regression") %>%
  set_engine("kknn")

knn_workflow <- workflow() %>%
  add_model(knn_model)%>%
  add_recipe(msocf22_recipe)

knn_params <- parameters(knn_model)
```

```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## Please use `hardhat::extract_parameter_set_dials()` instead.
```

```
knn_grid <- grid_regular(knn_params, levels = 5)

knn_tune <- knn_workflow %>%
  tune_grid(
    resamples = msocf22_fold,
    grid = knn_grid)


autoplot(knn_tune)
```




```
best_knn <- select_best(knn_tune, metric = "rmse")


knn_final <- finalize_workflow(knn_workflow,best_knn)

knn_final_fit <- fit(knn_final, data = msocf22_train)

knn_train_rmse <- augment(knn_final_fit, new_data = msocf22_train) %>%
  rmse(truth = total_impact, estimate = .pred)
knn_train_rmse
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        4.49
```

```
augment(knn_final_fit, new_data = msocf22_test) %>%
  rmse(truth = total_impact, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        10.3
```

```r
#random forest
#install.packages("rpart.plot")
#install.packages("vip")
#install.packages("janitor")
#install.packages("randomForest")
#install.packages("xgboost")
library(rpart.plot)
```

```
## Loading required package: rpart

##
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':
##
##     prune
```

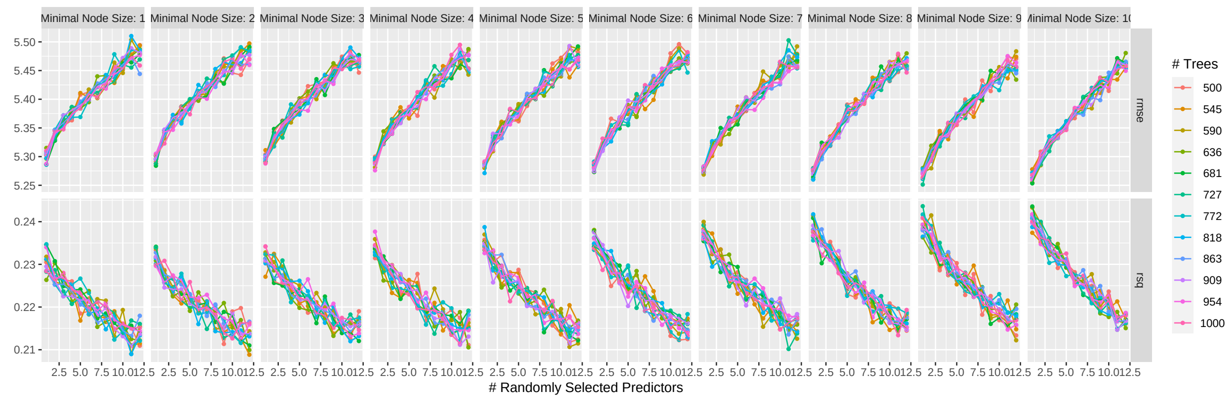
```r
library(vip)
library(janitor)
library(randomForest)
```

```
## randomForest 4.7-1.1

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

```r
library(xgboost)

rf_spec <- rand_forest(mtry=tune(),trees=tune(),min_n=tune())%>%
  set_engine("ranger",importance="impurity")%>%
  set_mode("regression")
rf_wk <- workflow()%>%
  add_recipe(msocf22_recipe)%>%
  add_model(rf_spec)

param_grid_rf <- grid_regular(mtry(range=c(1,12)),trees(range=c(500,1000)),min_n(range = c(1,10)),level
```

```r
#fitting random forest
#install.packages("ranger")

tune_res_rf <- tune_grid(
  rf_wk,
  resamples = msocf22_fold,
  grid = param_grid_rf,
)

autoplot(tune_res_rf)
write_rds(tune_res_rf, file = "tune_rf.rds")
```

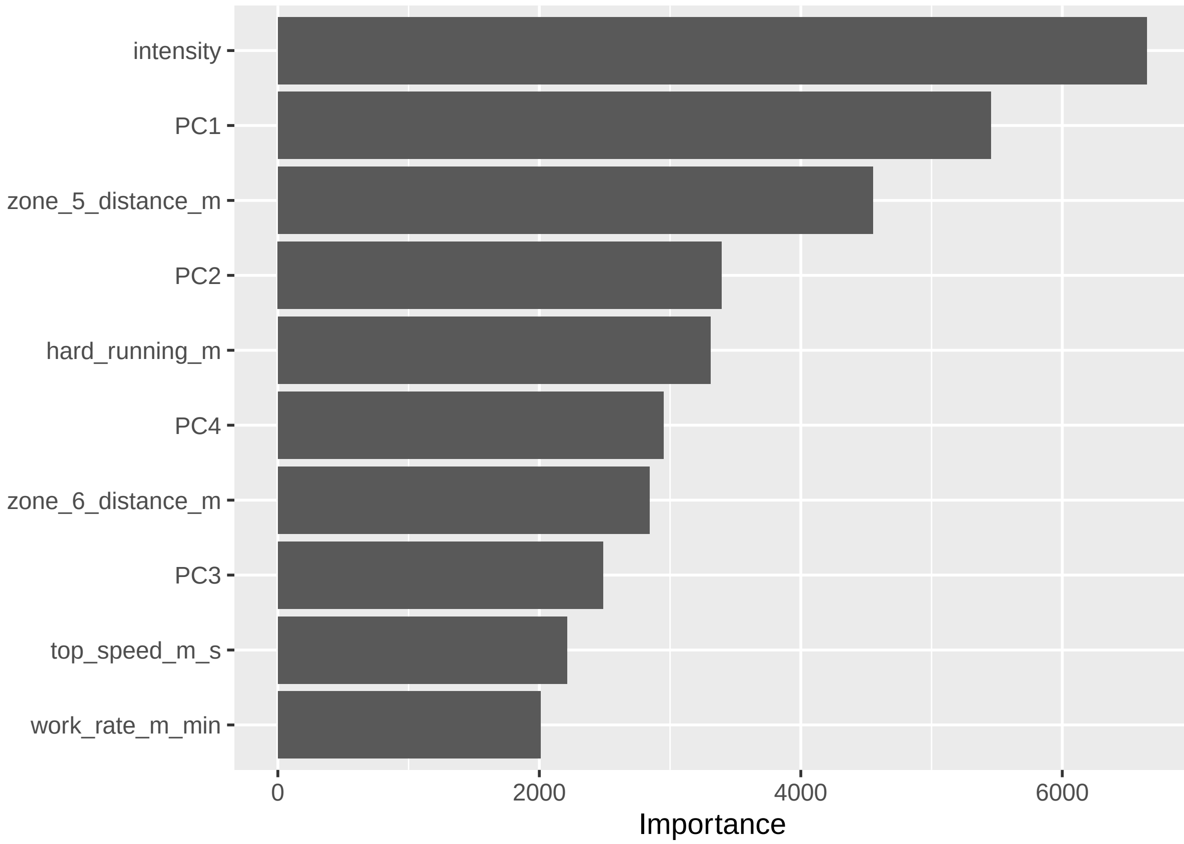
```r
rf_tree <- read_rds("tune_rf.rds")
autoplot(rf_tree)
```




```r
best_rf <- rf_tree %>%
  collect_metrics()%>%
  arrange(desc(mean))%>%
  head(1)
best_rf
```

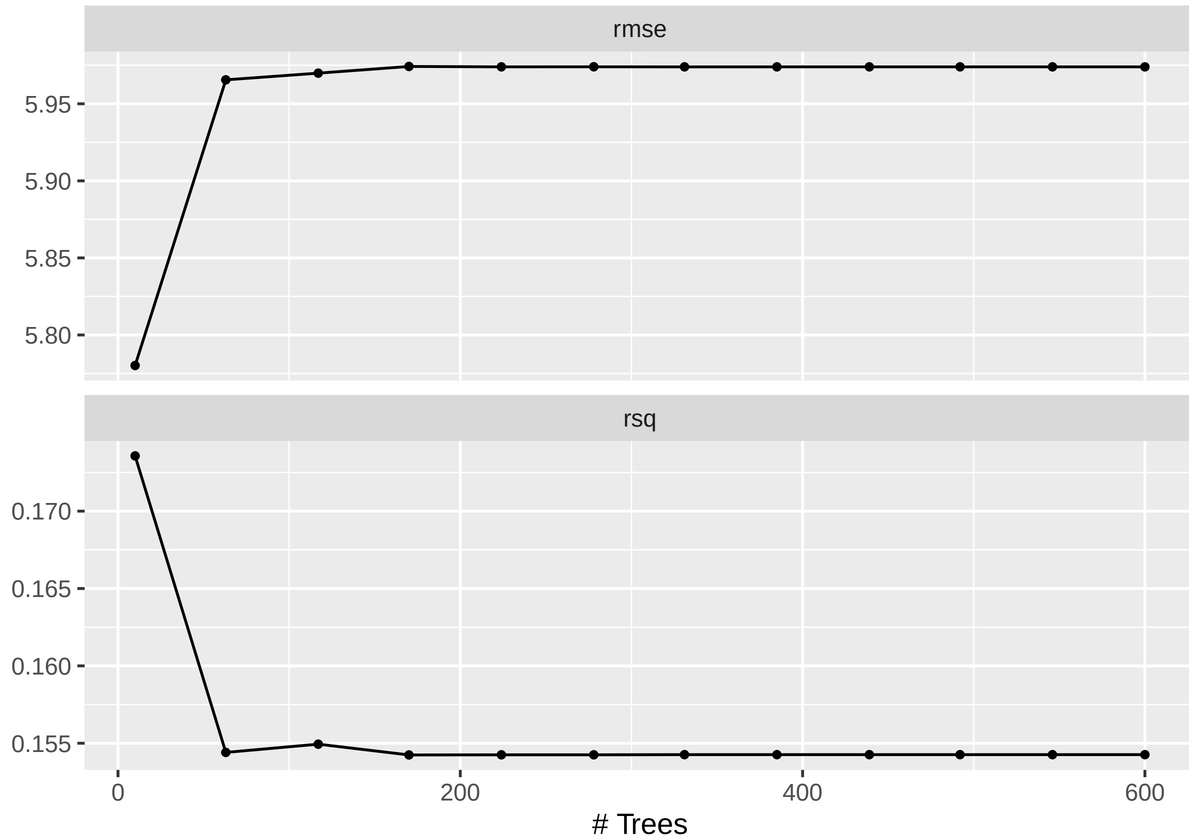
```
## # A tibble: 1 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    11   818     1 rmse    standard    5.51    10   0.472 Preprocessor1_Model0~
```

```r
rf_tree_final <- finalize_workflow(rf_wk,best_rf)
rf_tree_final_fit <- fit(rf_tree_final, data = msocf22_train)


#install.packages("vip")
library(vip)
rf_tree_final_fit %>%
  extract_fit_engine()%>%
  vip()
```




```
#boosting
boost_spec <- boost_tree(trees=tune())%>%
  set_engine("xgboost")%>%
  set_mode("regression")
boost_wf <- workflow()%>%
  add_recipe(msocf22_recipe)%>%
  add_model(boost_spec)

param_grid_boost <- grid_regular(trees(range=c(10,600)),levels=12)
param_grid_boost

tune_res_boost <- tune_grid(
  boost_wf,
  resamples = msocf22_fold,
  grid=param_grid_boost
)

autoplot(tune_res_boost)
write_rds(tune_res_boost,file="tune_boost.rds")
```

```
boost_tree <- read_rds("tune_boost.rds")
autoplot(boost_tree)
```




```
best_boost <- boost_tree %>%
  collect_metrics()%>%
  arrange(desc(mean))%>%
  head(1)
best_boost
```

```
## # A tibble: 1 x 7
##   trees .metric .estimator  mean     n std_err .config
##   <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1   170 rmse    standard    5.97    10   0.542 Preprocessor1_Model04
```

```
#after fitting all these models we want to see which ones did the best based on their root square
library(tibble)
best_tibble <- tibble(Models=c("Linear Regression", "Polynomial Rregression","Ridge","Lasso","KNN","Ranc
best_tibble
```

```
## # A tibble: 7 x 2
##   Models                  RMSE_bestvalues
##   <chr>                             <dbl>
## 1 Linear Regression                  5.37
## 2 Polynomial Rregression             5.31
## 3 Ridge                              4.20
## 4 Lasso                              4.09
## 5 KNN                                4.49
## 6 Random Forest                      5.51
```

```
## 7 Boosted Trees                      5.97
```

```r
#Fitting the best model to the testing data
augment(lasso_final_fit, new_data = msocf22_test) %>%
  rmse(truth = total_impact, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        8.78
```