Big Data Bowl 2024

Sam Burch

2024-1-2

Preliminary

A tackle isn't necessarily a good statistic to have. For example, if a defensive back allows a 20 yard completion and tackles the receiver, this is not good. Meanwhile, if a player is making a tackle for minimal offensive gain, this is usually good. There are some circumstances that make this not the end all be all though. Consider a lot of other defenders near the ball carrier versus no other defenders near them. Obviously, a tackle isn't quite as valuable in the first circumstance, because it's likely someone will get the tackle quickly and the ball carrier won't advance far. Lastly, what's the value of a defender running all the way across the field to make a tackle? Probably somewhere above the average value of a tackle at that spot, but likely depends on the defenders around the ball carrier more.

We will look into three tackle evaluation metrics throughout this study. The first is labeled true tackle rate. This is a combination of tackles, assists, and forced fumbles, with missed tackles removed (tkls + asts + 5(ff) - mts). Tackles and assists are weighted the same because, while during an assist the defender is slightly less responsible more often than not, they are still making the tackle. The defender is penalized for missed tackle as this allows the ball carrier to gain more yardage – most of the time. Fumbles are fluky, but being able to force a fumble can be a legitimate skill, as in the peanut punch. Causing such a fumble can lead to a turnover, which is huge for the defense. The arbitrary weight of 5 times as much as a tackle is due to what it can mean for the defense.

Next, we will look at true stop rate. A true stop is a tackle, assist, or forced fumble where the EPA on the play is negative. Typically defensive backs will have lower rates, as when they let up a pass, this likely goes for positive EPA. Unlike above, this assigns value to the tackle, letting us know if it is actually helping the defense or hurting them. (This metric is inspired by Nathan Jahnke at PFF.)

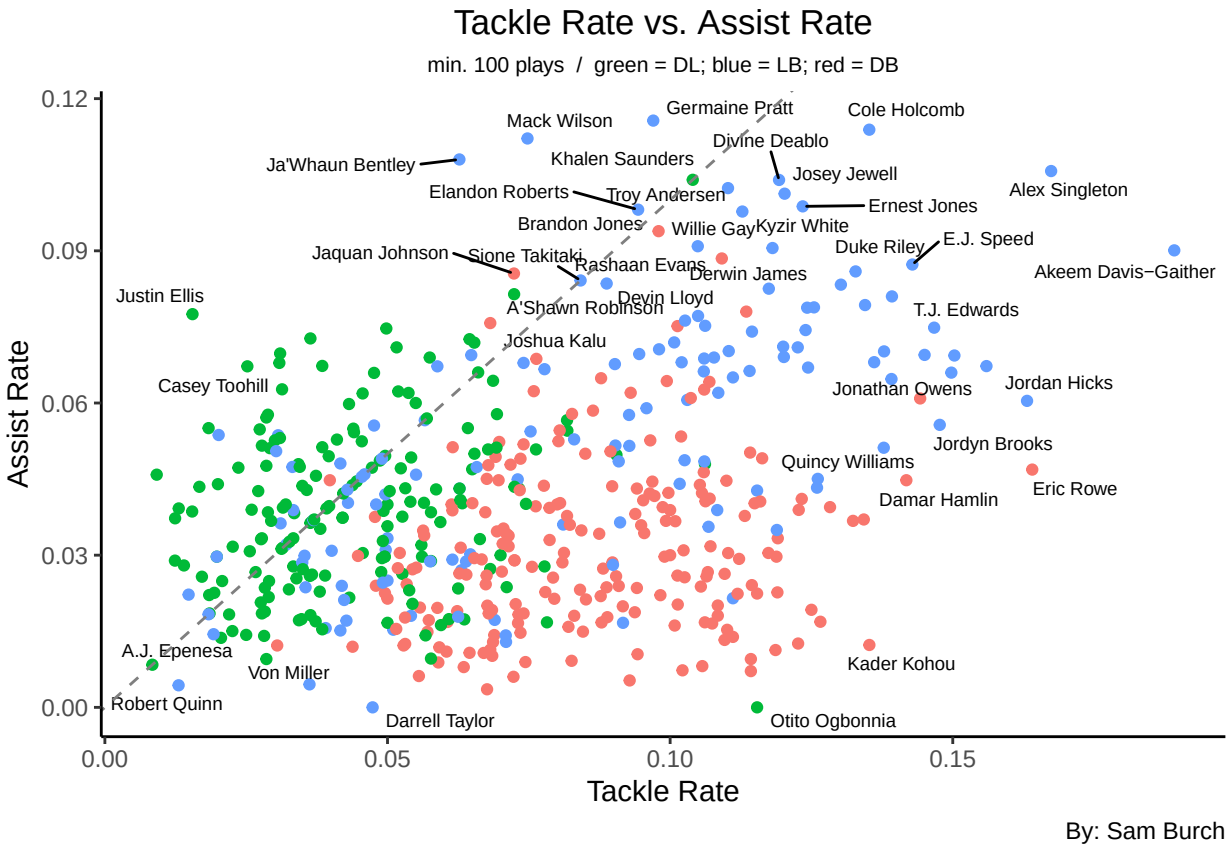
This leads us to expected points added (EPA) per stop. This is the total EPA allowed on a tackle, assist, or forced fumble per stop (aka tackle, assist, or forced fumble). This assigns complete value of the play to the defender with the stop. (Missed tackles are not considered because another player is actually making the stop.) Instead of a success rate style metric, this allows us to see how valuable each stop is on average. Obviously, this value assignment won't be perfect, however it will still give us a good idea – in general – of how valuable of a tackler they are.

Before we start our analysis, we must process the data. After doing so, we have 50 columns of information about each player.

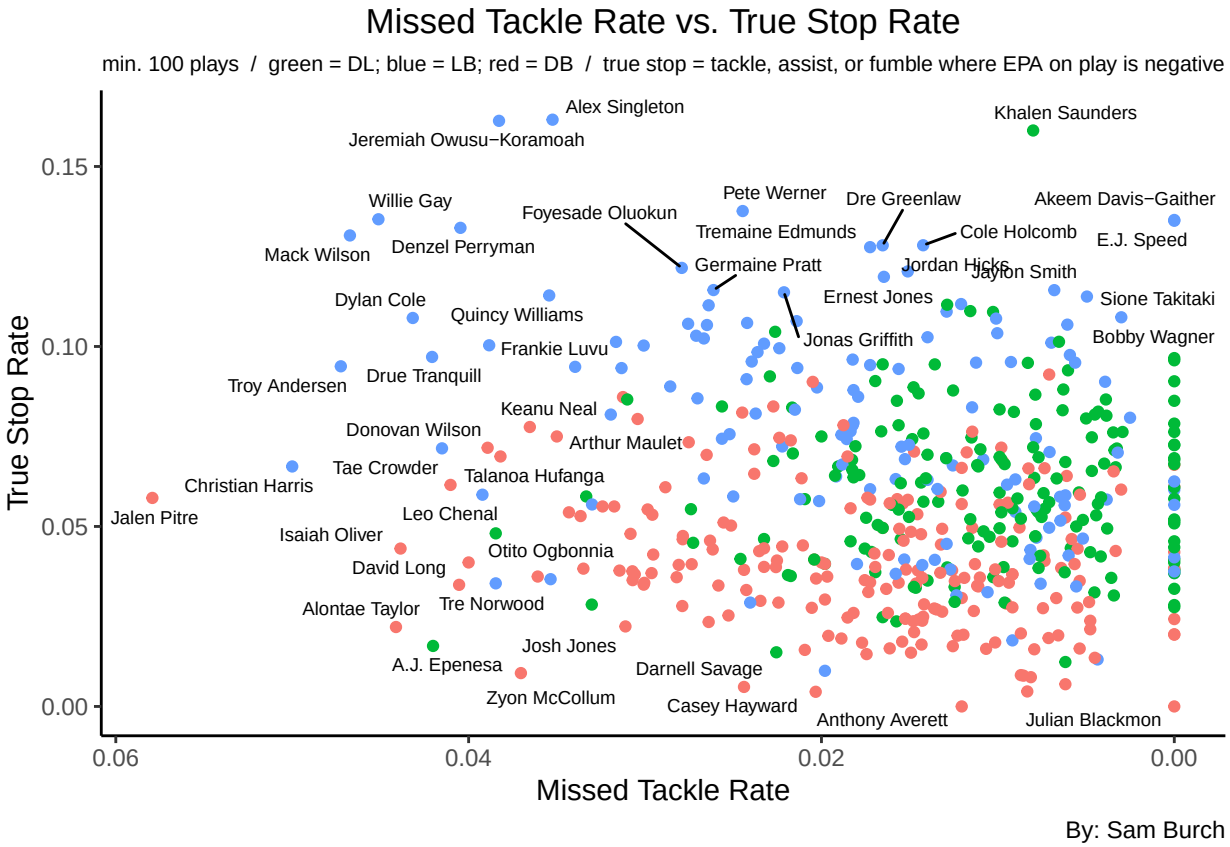
Aside from identification, there are position, height, weight, defenders in the box, speed, acceleration, and distance metrics. We considered age, however this eliminated a large amount of players due to those without birthdates noted. For position, the following three position groups were assigned: defensive back (DB), defensive line (DL), and linebacker (LB).

The data was filtered to players at with at least 100, as those with not many snaps won't have as stable results. (For more information on data processing, see the Appendix.)

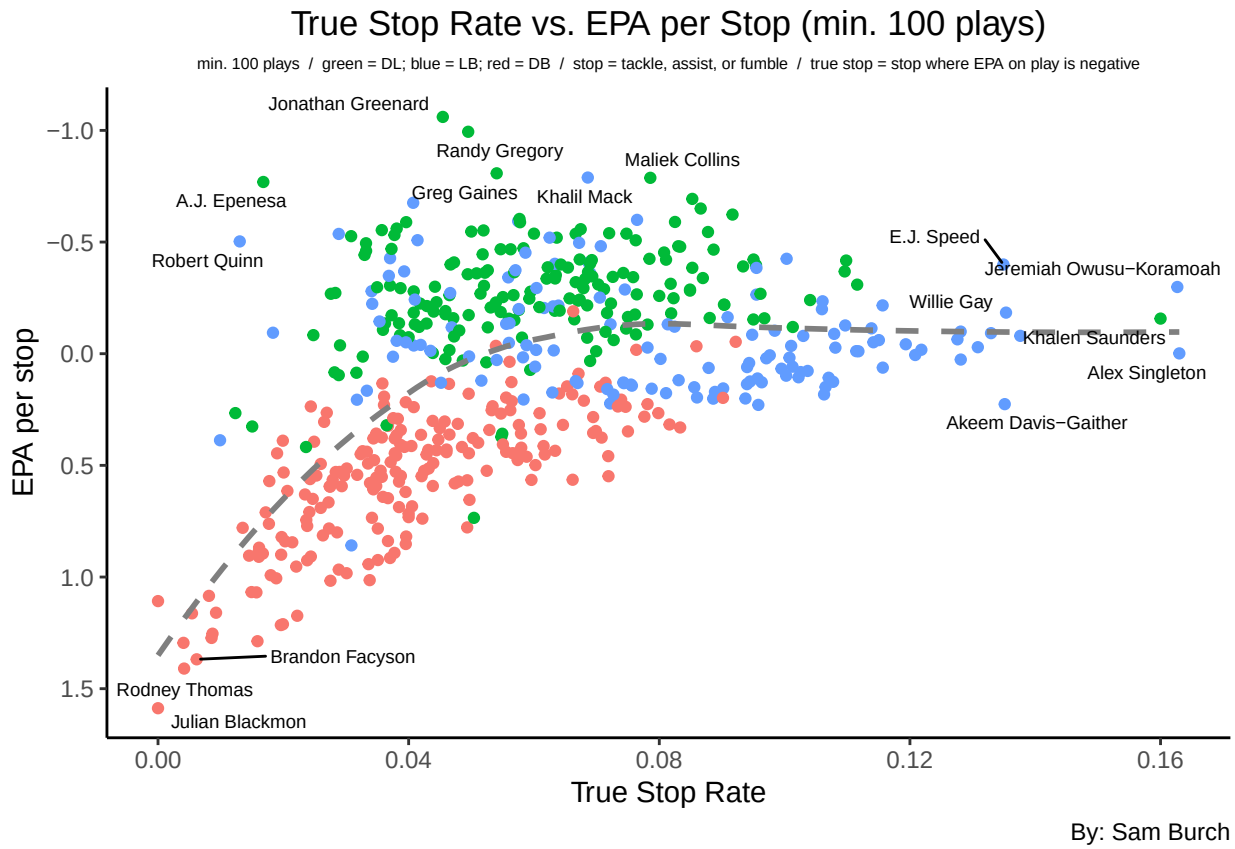
Summary of Variables & Data



It is clear that DBs have higher tackle rates, DL have higher assist rates, and linebackers are the best of both worlds. This makes sense, as coverage players will have more solo tackles than those on the LOS near the ball. Players like Alex Singleton and Cole Holcomb are good at making both tackles and assists, with a player like Robert Quinn struggling.



If one makes a lot of true stops and doesn't miss many tackles, that defender is probably a very good tackler. There isn't much correlation here (.14) and a lot of players have no missed tackles. The best players here are mainly linebackers, but also Khalen Saunders – a DB. Players like JOK and Willie Gay make a lot of true stops, but miss several tackles.



A rate metric (like true stop rate) will be more stable than EPA per stop. So, this shows us whose performances will likely regress. Epenesa has a poor true stop rate, but his EPA per stop is quite good; this means he'll regress to a worse EPA per stop. However, when we consider the different positions, a lot of DL have this trait because they are near the LOS. DBs are the opposite because they are typically far from the LOS. Linebackers once again dominate here with low EPA per stop and a high true stop rate. Also, consider that the more true stops one has, the more EPA per stop stabilizes – around 0.

Let's examine the impact of position on the three metrics...

Position Group Summary
min. 100 plays

Average	DB[1]	DL[2]	LB[3]
Height (in.)	71.99	75.67	74.59
Weight (lbs.)	197.85	294.18	244.27
BMI	256.29	182.32	215.29
Dist B.C.	14.14	9.33	9.29
Dist Ball	13.59	8.51	8.54
Speed	3.19	2.39	2.96
Acceleration	2.08	1.47	2.01
Tackle Rate	0.09	0.04	0.09
Assist Rate	0.03	0.04	0.05
Forced Fumble Rate	0.00	0.00	0.00
Missed Tackle Rate	0.02	0.01	0.02
Stop Rate	0.12	0.08	0.14
True Tackle Rate	0.11	0.07	0.13
True Stop Rate	0.04	0.06	0.08
EPA/stop	0.55	-0.27	-0.07
Players	205.00	171.00	135.00

[NA]By: Sam Burch
[1]defensive back, safety, or corenerback
[2]defensive end, defensive tackle, or nose tackle
[3]inside or outside linebacker

Obviously, each position group has very different heights and weights. Highest BMI, acceleration, and speed are DBs, with DL having the worst.

LBs and DL are closer to the ball carrier (slight lean LB) and ball (slight lean DL) than DBs.

Average forced fumble rate is 0% across all positions, which makes sense since they are rare.

True tackle rate is highest for LBs, then DBs, and DL are last. A big reason for this is the tackle and assist rates. In fact, true tackle rate has a .86 correlation with tackle rate and a .67 correlation with assist rate. This means this metric is not much of an improvement over something like stop rate.

True stop rate on the other hand relies more on distance to the LOS, with LBs being highest, then DL, and DBs last. This is what traditionally people think of as who are the best and worst tacklers, which is a good sign.

EPA per stop has (by far) the biggest difference. DBs allow .55 EPA per stop because of what we talked about in the introduction; they allow a catch to be made before making the stop. LBs are slightly higher than DL due to being in coverage too, but are much better than DBs. DL have the lowest because when they make a stop, it is almost always near the backfield.

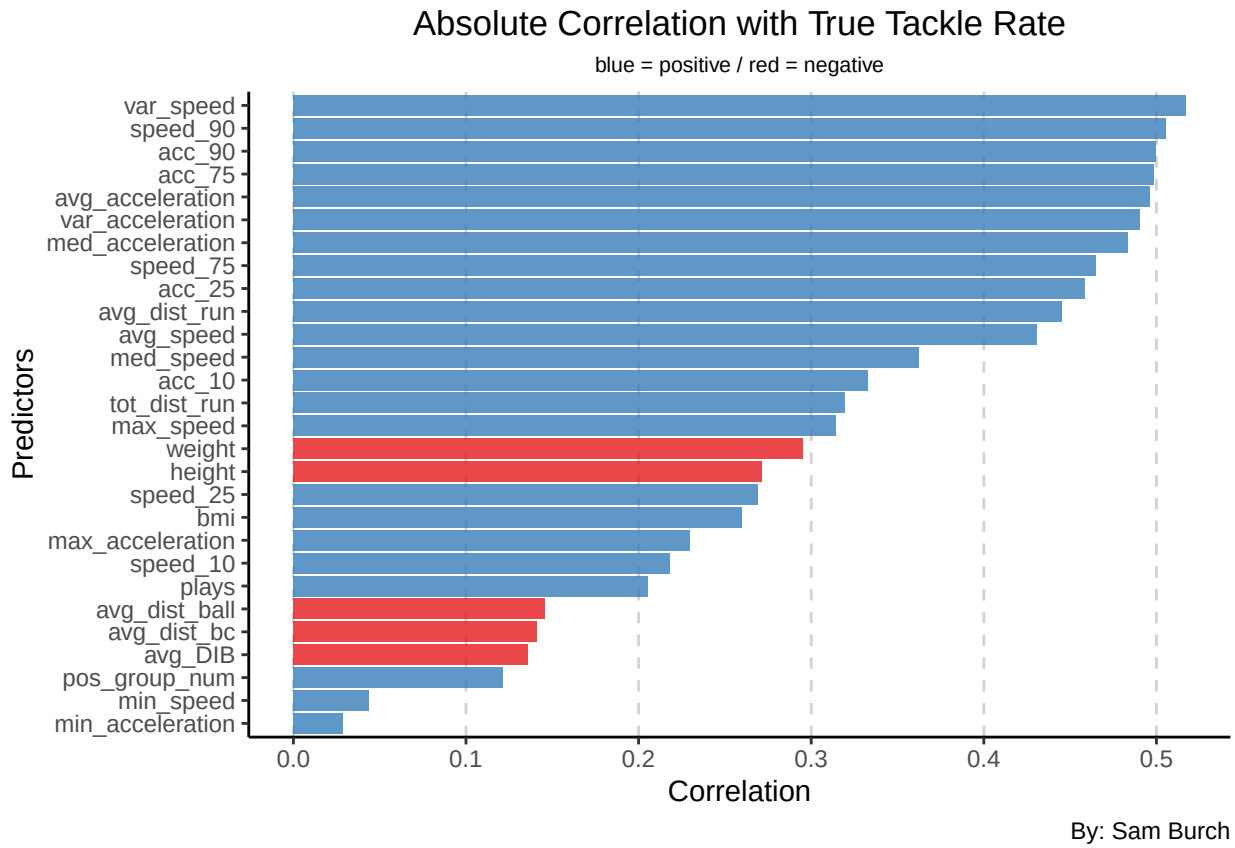
Distributions Table

Statistic	True Tackle Rate[1]	True Stop Rate[2]	EPA/stop
Average	0.100	0.058	0.110
Variance	0.002	0.001	0.209
Minimum	-0.025	0.000	-1.060
10th pct	0.050	0.025	-0.425
25th pct	0.068	0.037	-0.239
Median	0.091	0.055	0.058
75th pct	0.128	0.074	0.430
90th pct	0.164	0.096	0.739
Maximum	0.279	0.163	1.587

[NA]By: Sam Burch
[1]ttr = (tackle + assist - missed tackle + 5*fumble) / plays
[2]tsr = (tackle + assist + fumble | -EPA) / plays

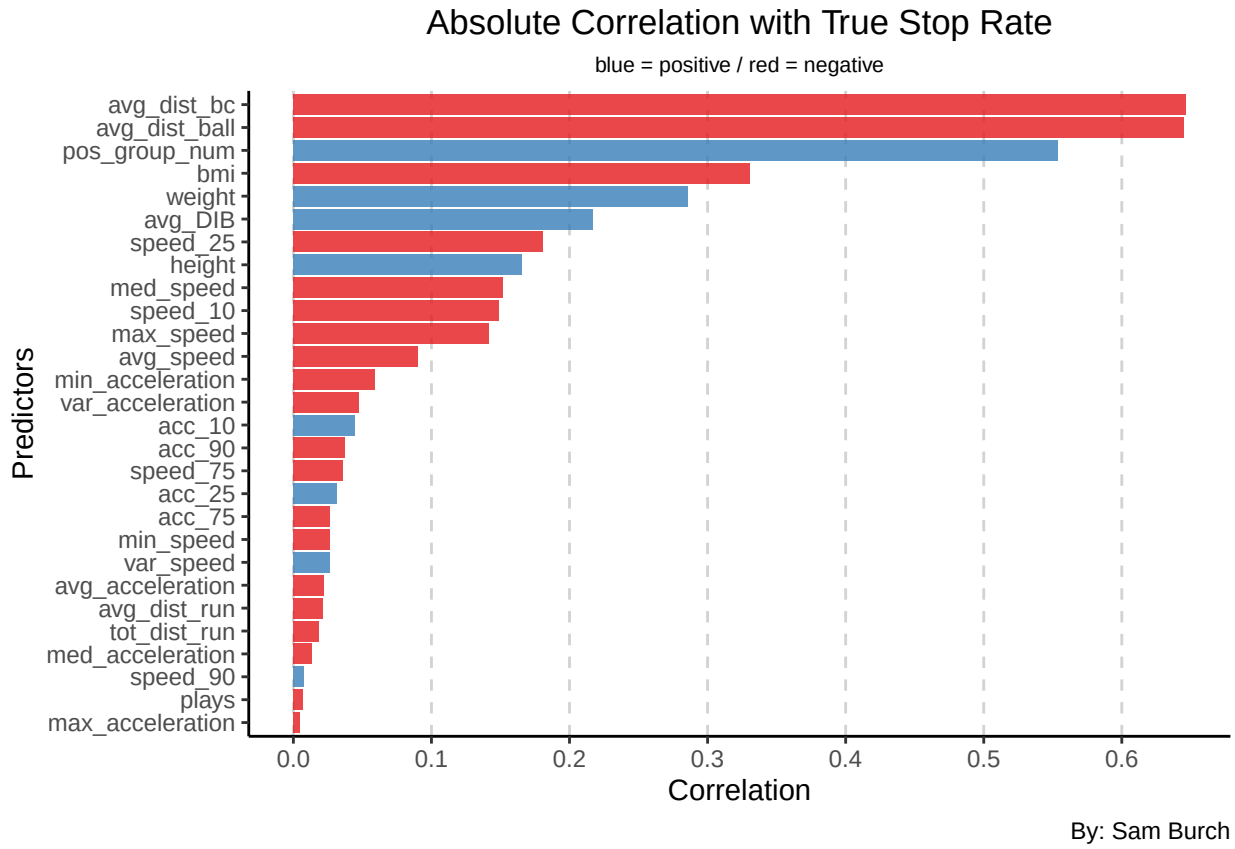
- A lot of variance when it comes to EPA/stop
- The lowest rate is negative for TTR because of missed tackles
- Average is skewed right (worse) for EPA/stop
- Highest (worst) EPA/stop is ~1.5
- Keep these numbers in mind with the analysis below

True Tackle Rate



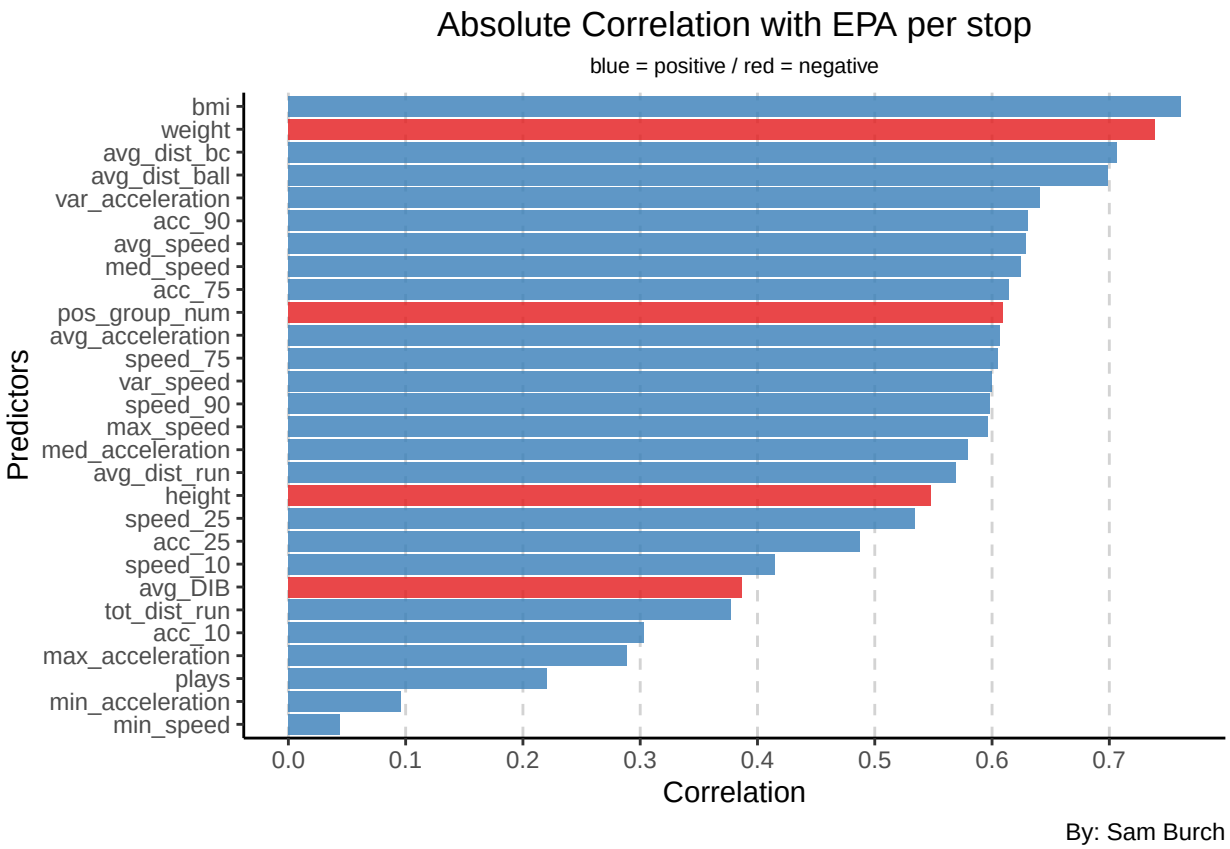
Speed and acceleration seem to matter a lot when looking at TTR, same with distance run. These suggest higher speeds, accelerations, and distances ran lead to a higher TTR. Being a big specimen leads to lower TTR though, considering the impact of weight and height.

True Stop Rate



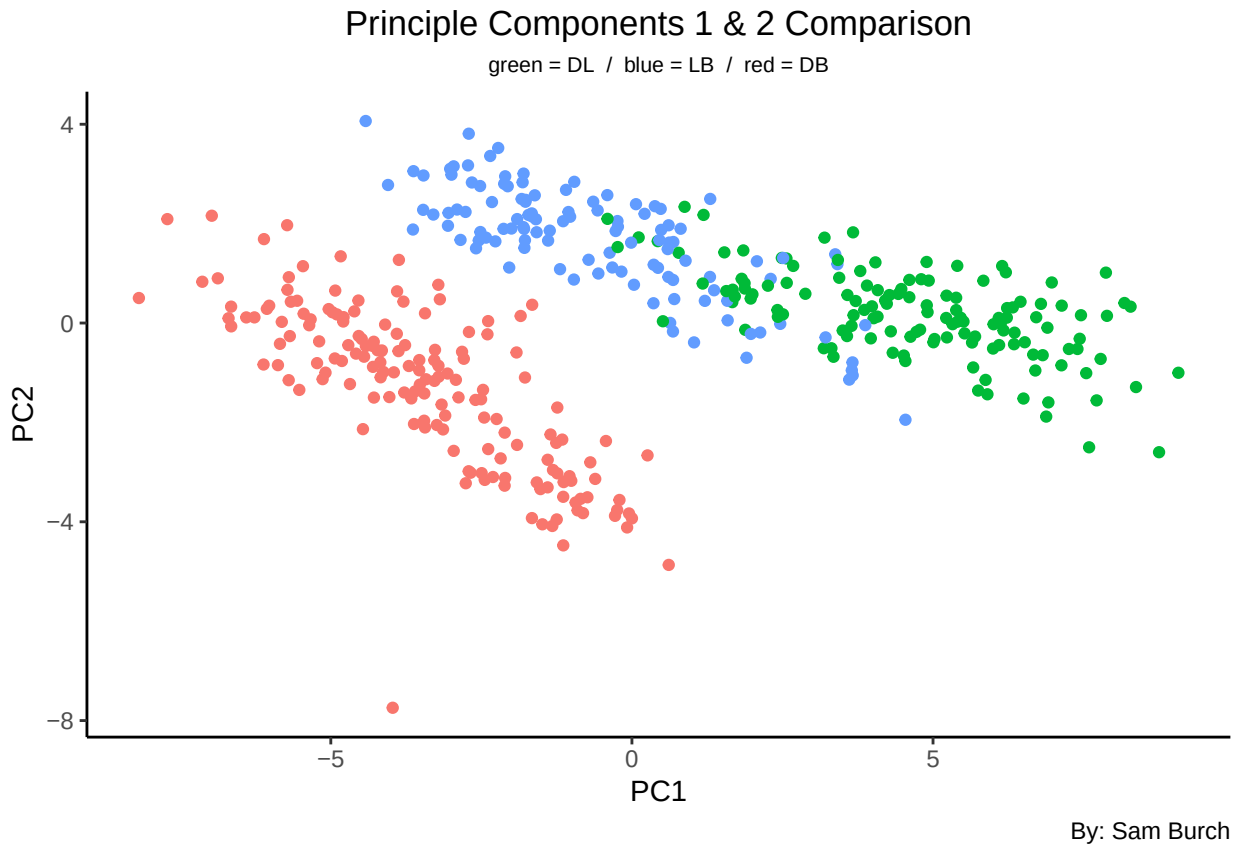
Average distance to the ball and ball carrier are the most important, followed by position group. This suggests that those closer to the ball or the ball carrier (on average) are more likely to make the stop. Obviously, the defender is always going to be very close to the ball & ball carrier when making the stop. Even so, it seems that getting to the ball carrier will make you get the stop more often. Also, consider that DBs are often very close to the ball carrier (given their receiver catches the ball).

EPA / Stop



Keep in mind that a lower EPA/stop is better. Thus, a large weight, height, and amount of defenders in the box – and linebackers & defensive lineman – will all tend to have a better EPA/stop. The large weight and height indicate the position importance here as well. More defenders in the box will help fill gaps and more likely signals a run play – plays with less EPA in general.

PCA



There is a clear split based on position here. The DBs cluster is completely separate from DL and LBs; these two have some overlap. This suggests the PCs are highly based on the position of the players.

Regression Analysis

As we continue the modeling process, this will consist of developing six models for each of the three metrics. The first will be the average for that metric as a baseline. On top of this, we will fit a simple linear regression (SLR) with the position group. The next model will be a SLR with the most correlated variable as the predictor. Because of multicollinearity issues that would arise, using multiple predictors with linear regression is not good. Thus, we chose applying a ridge regression to reduce this issue.

Two other models were fit were k-nearest neighbors (KNN) and random forest. These modeling methods typically are more efficient and produce lower errors than linear regression; although some interpretability is lost.

We will compare each model based on their root mean square error (RMSE). This tells us how off our model is on average. To do so, we randomly split our data into 80% training and 20% testing so that we can evaluate based on the testing error.

True Tackle Rate

- Average (.0452)
- Position Group (.0446)
- Variance Speed (.0376)
- Ridge (.0304)
- KNN (.0296)
- Random Forest (.0280)

Random forest produces the lowest error at being off by only 2.8% on average. This is 40% lower than the RMSE for the average, which is good. In fact, with each model fit, the RMSE continued to drop. Ridge and KNN both performed well too, but random forest edges it out.

True Stop Rate

- Average (.0257)
- Position Group (.0210)
- Mean Distance to Ball Carrier (.0183)
- Ridge (.0169)
- KNN (.0169)
- Random Forest (.0163)

Again, we got a lower (or equal) error with each new model. Thus, random forest performs the best by being off by 1.63% on average. (Keep in mind that each metric has its own scale, so the errors cannot be compared across metrics.) We also see the ridge and KNN models performing very similarly, as their errors are approximately equal. Furthermore, position group performs quite well here since there is a significantly lower rate for that model compared to the average.

EPA / Stop

- Average (.4372)
- Position Group (.3666)
- BMI (.2622)
- Ridge (.2551)
- KNN (.2563)
- Random Forest (.2602)

The selected model here is ridge, with being off by .2551 on average. These models are not as accurate when considering the range from the 10th to 90th percentile is ~1.1. Regardless, this error is still ~42% less than the average and ~30% less than the position group model. Also, ridge seems to have won because a lot of variables are important, which is one advantage of ridge regression.

Conclusion

RMSE Summary Table
lowest error = selected model

Model Type	True Tackle Rate[1]	True Stop Rate[2]	EPA/stop
Average	0.0452	0.0257	0.4372
Postion Group	0.0446	0.0210	0.3666
1 Predictor	0.0376	0.0183	0.2622
Ridge	0.0304	0.0169	0.2551
KNN	0.0296	0.0169	0.2563
Random Forest	0.0280	0.0163	0.2602

[NA]By: Sam Burch
[1]ttr = (tackle + assist - missed tackle + 5*fumble) / plays
[2]tsr = (tackle + assist + fumble | -EPA) / plays

After going through model selection, we derived good models for each of the three metrics. By doing so, we can develop an expected TTR, TSR, and EPA/Stop. This allows us to see who are the good tacklers, who's going above (or below) & beyond, and how stable their performances are. For example, if one's xTSR is high and their actual TSR is low, then their true play is probably somewhere in between. They might be getting to the ball carrier well, but not finishing. Hence, if they can clean up some of their tackling technique, positioning, or just get better luck, they can see their TSR regress positively and become a better tackler.

Future analysis should build upon using EPA per play as a valuation metric. This is a good valuation for the defense as a whole, but assigning all the value to the tackler may not be ideal. If an expected points saved on a rate or per stop basis can be established, this would be ideal. This valuation may consider surrounding defenders, account for coverage ability, and even surrounding offensive players who can set up blocks. That being said, these metrics still tell us much of the story and can at least be building blocks for future analysis.

Thank you for reading!

Appenicies

References

- Nathan Jahnke's Article [https://www.pff.com/news/pro-defensive-stops-a-more-comprehensive-metric-than-tackles]

Code

Raw Code for Data Processing, Summary, and Modeling.

```
# Data Inport

games = read_csv("coding-projects/big-data-bowl-24/games.csv")
players = read_csv("coding-projects/big-data-bowl-24/players.csv")
pbp = read_csv("coding-projects/big-data-bowl-24/plays.csv")
tackles = read_csv("coding-projects/big-data-bowl-24/tackles.csv")
all_wks = read_csv("coding-projects/big-data-bowl-24/tracking_weeks_all.csv")[, -1]

## How got all_wks DF

wk1 = read_csv("coding-projects/big-data-bowl-24/tracking_week_1.csv")
wk2 = read_csv("coding-projects/big-data-bowl-24/tracking_week_2.csv")
wk3 = read_csv("coding-projects/big-data-bowl-24/tracking_week_3.csv")
wk4 = read_csv("coding-projects/big-data-bowl-24/tracking_week_4.csv")
wk5 = read_csv("coding-projects/big-data-bowl-24/tracking_week_5.csv")
wk6 = read_csv("coding-projects/big-data-bowl-24/tracking_week_6.csv")
wk7 = read_csv("coding-projects/big-data-bowl-24/tracking_week_7.csv")
wk8 = read_csv("coding-projects/big-data-bowl-24/tracking_week_8.csv")
wk9 = read_csv("coding-projects/big-data-bowl-24/tracking_week_9.csv")
all_wks = wk1 |>
  full_join(wk2) |>
  full_join(wk3) |>
  full_join(wk4) |>
  full_join(wk5) |>
  full_join(wk6) |>
  full_join(wk7) |>
  full_join(wk8) |>
  full_join(wk9)
write.csv(all_wks, file = "tracking_weeks_all.csv")
```

```
# Data Processing 1

# Starting point for player info
players_red = players |>
  dplyr::select(nflId, position)
head(players_red)


# Getting Defensive Positions
all_positions = players_red |>
  dplyr::select(position) |>
  unique() |>
  pull()
def_positions = all_positions[c(5, 6, 7, 8, 10, 11, 12, 15, 16, 19)]
def_positions

players2 = players |>
  filter(position %in% def_positions) |>
  mutate(height = 12*as.double(str_sub(height, 1, 1)) + as.double(str_sub(height, 3, length(height))))
  mutate(bmi = 703 * (height / weight)) |>
```

```r
  # Age is not that important, so decided to drop to add in 115 plays
  # mutate(birthDate = ymd(as.Date(birthDate))) |>
  # mutate(age = interval(birthDate, Sys.Date()) %/% months(1)) |>
  # mutate(age = age / 12) |>
  dplyr::select(nflId:weight, bmi, everything()) |>
  dplyr::select(nflId, height:bmi)
head(players2)
```

```r
# Data Processing 2

# PBP useful metrics
red_pbp = pbp |>
  # Remove this restriction when not focusing on pass / run split
  filter(
    # !is.na(passProbability),
         !is.na(defendersInTheBox)) |>
  dplyr::select(gameId, playId, defendersInTheBox,
                # passProbability,
                expectedPointsAdded) |>
  # mutate(xpass = if_else(passProbability >= .7, 1, 0),
  #        xrun = if_else(passProbability <= .3, 1, 0)) |>
  # Successful for offense
  mutate(success = if_else(expectedPointsAdded >= 0, 1, 0))
head(red_pbp)


# PBP & Tackle Data
tkls = tackles |>
  left_join(red_pbp, by = c("gameId", "playId")) |>
  group_by(nflId) |>
  summarise(tackles = sum(tackle, na.rm = TRUE),
            assists = sum(assist, na.rm = TRUE),
            forcedFumbles = sum(forcedFumble, na.rm = TRUE),
            missed_tackles = sum(pff_missedTackle, na.rm = TRUE),
            stops = tackles + assists + forcedFumbles,

            avg_DIB = sum(defendersInTheBox[pff_missedTackle == 0], na.rm = TRUE) / stops,

            # pass_tackles = sum(tackle, xpass == 1, na.rm = TRUE),
            # pass_assists = sum(assist, xpass == 1, na.rm = TRUE),
            # pass_forcedFumbles = sum(forcedFumble, xpass == 1, na.rm = TRUE),
            # pass_missed_tackles = sum(pff_missedTackle, xpass == 1, na.rm = TRUE),
            # pass_stops = pass_tackles + pass_assists + pass_forcedFumbles,

            # run_tackles = sum(tackle, xrun == 1, na.rm = TRUE),
            # run_assists = sum(assist, xrun == 1, na.rm = TRUE),
            # run_forcedFumbles = sum(forcedFumble, xrun == 1, na.rm = TRUE),
            # run_missed_tackles = sum(pff_missedTackle, xrun == 1, na.rm = TRUE),
            # run_stops = run_tackles + run_assists + run_forcedFumbles,

            tot_epa = sum(expectedPointsAdded[pff_missedTackle == 0], na.rm = TRUE),
            avg_stop_epa = tot_epa / stops,
```

```
            successes_allowed = sum((success == 1 & tackle == 1) | (success == 1 & assist == 1) | (succ
            success_rate_allowed = successes_allowed / stops,

            .groups = "drop")
tkls
```

```
# Data Processing 3

# Distance Helpers
runner_df = pbp |> dplyr::select(gameId, playId, ballCarrierId) |> mutate(ball_carrier = 1)

bc = all_wks |>
  left_join(runner_df, by = c("gameId", "playId", "nflId" = "ballCarrierId")) |>
  # mutate(ball_carrier = if_else(is.na(ball_carrier), 0, 1)) |>
  filter(ball_carrier == 1) |>
  dplyr::select(gameId, playId, frameId, x, y) |>
  rename(bc_x = x,
         bc_y = y)
bc

ball = all_wks |>
  filter(club == "football") |>
  dplyr::select(gameId, playId, frameId, x, y) |>
  rename(ball_x = x,
         ball_y = y)
ball


e_distance = function(x1, y1, x2, y2) {
  return(sqrt((x2 - x1)^2 + (y2 - y1)^2))
}
```

```
# Data Processing 4

# FULL DF
df_full = all_wks |>
  left_join(players_red, by = "nflId") |>
  filter(position %in% def_positions) |>
  left_join(bc, by = c("gameId", "playId", "frameId")) |>
  left_join(ball, by = c("gameId", "playId", "frameId")) |>
  mutate(bc_dist = e_distance(x, y, bc_x, bc_y)) |>
  mutate(ball_dist = e_distance(x, y, ball_x, ball_y)) |>
  # filter(!is.na(nflId)) |>
  group_by(nflId, displayName, position) |>
  summarise(tot_dist_run = sum(dis),
            # Filter to only 1st frame?
            avg_dist_bc = mean(bc_dist),
            avg_dist_ball = mean(ball_dist),

            min_speed = min(s),
            speed_10 = quantile(s, .1),
            speed_25 = quantile(s, .25),
            med_speed = median(s),
```

```
            speed_75 = quantile(s, .75),
            speed_90 = quantile(s, .9),
            max_speed = max(s),
            avg_speed = mean(s),
            var_speed = var(s),
            min_acceleration = min(a),
            acc_10 = quantile(a, .1),
            acc_25 = quantile(a, .25),
            med_acceleration = median(a),
            acc_75 = quantile(a, .75),
            acc_90 = quantile(a, .9),
            max_acceleration = max(a),
            avg_acceleration = mean(a),
            var_acceleration = var(a),

            plays = n_distinct(playId),
            .groups = "drop") |>
left_join(players2, by = "nflId") |>
left_join(tkls, by = "nflId") |>

# Removes people without tackles
## They were on a small sample size anyway
filter(!is.na(tackles)) |>

mutate(pos_group = case_when(
  (position == "CB" | position == "DB" | position == "FS" | position == "SS") ~ "DB",
  (position == "DE" | position == "DT" | position == "NT") ~ "DL",
  (position == "ILB" | position == "MLB" | position == "OLB") ~ "LB")) |>
mutate(avg_dist_run = tot_dist_run / plays,

       tackle_rate = tackles / plays,
       assit_rate = assists / plays,
       ff_rate = forcedFumbles / plays,
       missed_tackle_rate = missed_tackles / plays,
       stop_rate = stops / plays,
       # Metric 1
       true_stops = stops - successes_allowed,
       true_stop_rate = true_stops / plays,
       # Metric 2
       ## Edit forcedFumbles?
       true_tackles = tackles + assists + 5*(forcedFumbles) - missed_tackles,
       true_tackle_rate = true_tackles / plays,

       # XPass
       # pass_tackle_rate = pass_tackles / plays,
       # pass_assist_rate = pass_assists / plays,
       # pass_ff_rate = pass_forcedFumbles / plays,
       # pass_missed_tackles_rate = pass_missed_tackles / plays,
       # pass_stop_rate = pass_stops / plays,

       # XRun
       # run_tackle_rate = run_tackles / plays,
       # run_assist_rate = run_assists / plays,
```

```
        # run_ff_rate = run_forcedFumbles / plays,
        # run_missed_tackles_rate = run_missed_tackles / plays,
        # run_stop_rate = run_stops / plays

        pos_group_num = case_when(pos_group == "DB" ~ 1,
                                  pos_group == "DL" ~ 2,
                                  pos_group == "LB" ~ 3)


  )


# After processing the data, we have 50 columns of information about each player.
#
# Aside from identification, there are position, height, weight, defenders in the box, speed, accelerat
#
# For speed and acceleration, the 10th, 25th, 50th, 75th, and 90th percentile, along with minimum, maxi
#
# Distance consisted of total distance ran, average distance ran, average distance to the ball, and ave
#
# Average defenders in the box was considered.
#
# There were many tackle metrics calculated as well: tackles, assists, forced fumbles, missed tackles,
#
# Total EPA on every stop led us to EPA per stop.
#
# Successes allowed is used to calculate true stops, but we also got a success rate allowed for each st
#
# The data was then filtered to players at least 100.
```

```
# TTR 1

# TT-Rate
ggplot(df_full |> filter(plays >= 100), aes(x = true_tackle_rate)) +
  geom_histogram(color = "black", fill = "blue", binwidth = .02) +
  theme_minimal()

ggplot(df_full |> filter(plays >= 100), aes(x = true_tackle_rate)) +
  geom_boxplot() +
  scale_x_continuous(breaks = seq(-.1, .3, .05)) +
  theme_minimal()

## Outliers
df_full |>
  filter(plays >= 100, true_tackle_rate < 0 | true_tackle_rate > .22) |>
  dplyr::select(displayName, position, tackles:stops, true_tackle_rate, plays)


# We can see that most players have a ttr around 8%. The graph is skewed right, meaning the average is
```

```
# TTR 2

ggplot(df_full |> filter(plays >= 100), aes(x = pos_group, y = true_tackle_rate)) +
```

```
  geom_boxplot() +
  theme_minimal()

# By position, linebackers have the highest ttr on average, with defensive lineman having the lowest. T
```

```
# TSR 1

# TS-Rate
ggplot(df_full |> filter(plays >= 100), aes(x = true_stop_rate)) +
  geom_boxplot() +
  scale_x_continuous(breaks = seq(0, .2, .02)) +
  theme_minimal()

## Outliers
df_full |>
  filter(plays >= 100, true_stop_rate >= .13) |>
  dplyr::select(displayName, position, tackles:stops, true_stop_rate, plays)

# Players median tsr is at ~5% and is skewed right. All but one of the outliers are linebackers (DT) an
```

```
# TSR 2

ggplot(df_full |> filter(plays >= 100), aes(x = pos_group, y = true_stop_rate)) +
  geom_boxplot() +
  theme_minimal()

# Postion group is reflected this way, as DB have lower tsrs than LBs on average. DL is in the middle,
```

```
# EPA/stop 1

# EPA
ggplot(df_full |> filter(plays >= 100), aes(x = avg_stop_epa)) +
  geom_boxplot() +
  scale_x_reverse() +
  # scale_x_continuous(breaks = seq(-1.2, 2, .3)) +
  theme_minimal()

## Outliers
df_full |>
  filter(plays >= 100, avg_stop_epa > 1.5) |>
  dplyr::select(displayName, position, tackles:stops, avg_stop_epa, plays)

# The average for stop epa is around 0 and fairly normally distributed. The highest players have around
```

```
# EPA/stop 2

ggplot(df_full |> filter(plays >= 100), aes(x = pos_group, y = avg_stop_epa)) +
  geom_boxplot() +
  theme_minimal()

# DL have the best with LBs not too far behind. This is because these positions are more likely to make
```

```r
### TTR

# Linear Regression

m1 = lm(true_tackle_rate ~ ., data = train_data)
summary(m1)

set.seed(123)

m2 = step(m1, trace = 0)
summary(m2)

sqrt(vif(m2))
# A LOT of mulitcollinearity

# cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, med_ac
#
# abs(cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, me

# Avg Dist Ball, Avg Dist BC, BMI
# Speeds & Accelerations, BMI, Height & Weight
# Plays

m3 = lm(true_tackle_rate ~ var_speed, data = train_data)
summary(m3)


par(mfrow = c(2, 2))
plot(m3)
# Looks ok
```

```r
### TTR

set.seed(123)

x_train = as.matrix(scale(train_data[, -(ncol(train_data) - 1)]))  # Standardize predictor variables fo
y_train = as.matrix(train_data$true_tackle_rate)

x_test = as.matrix(scale(test_data[, -(ncol(test_data) - 1)]))  # Standardize predictor variables for t
y_test = as.matrix(test_data$true_tackle_rate)


ridge_model = cv.glmnet(x_train, y_train, alpha = 0)

plot(ridge_model)

best_lambda = ridge_model$lambda.min

final_model = glmnet(x_train, y_train, alpha = 0, lambda = best_lambda)

final_model$beta

pred = predict(final_model, newx = x_test)
```

```r
sqrt(mean((pred - y_test)^2))*100
```

```r
### TTR


# By Average
m_avg = rep(df_full2 |> summarise(mean(true_tackle_rate)) |> pull(), 103)

sqrt(mean((m_avg - y_test)^2))*100

# By Position
m_pos = lm(true_tackle_rate ~ pos_group_num, data = train_data)
summary(m_pos)

pred = predict(m_pos, newdata = test_data)

sqrt(mean((pred - y_test)^2))*100


pred = predict(m3, newdata = test_data)

sqrt(mean((pred - y_test)^2))*100
```

```r
### TTR

### KNN

error = numeric(50)

set.seed(123)

for (i in 1:50) {

  knn.fit = kknn(true_tackle_rate ~ ., train = train_data,
                 test = test_data |> dplyr::select(-true_tackle_rate),
                 k = i, kernel = "rectangular")

  test.pred = knn.fit$fitted.values

  error[i] = sqrt(mean((test.pred - (test_data |> dplyr::select(true_tackle_rate) |> pull()))^2))


}

min(error)*100
```

```r
### TTR

K = which.min(error[error != 0 & !is.na(error)])
K

knn.fit = kknn(true_tackle_rate ~ ., train = train_data,
               test = test_data |> dplyr::select(-true_tackle_rate),
```

```
               k = 18, kernel = "rectangular")

test.pred = knn.fit$fitted.values


sqrt(mean((test.pred - (test_data |> dplyr::select(true_tackle_rate) |> pull()))^2))


knn_df = data.frame(knn = test.pred, actual = test_data |> dplyr::select(true_tackle_rate) |> pull())


ggplot(knn_df, aes(x = knn, y = actual)) +
  geom_point() +
  labs(title = "KNN Predictions",
       # subtitle = "RMSE = .66; k = ?",
       x = "KNN Prediction",
       y = "TT Rate") +
  # scale_y_continuous(breaks = seq(.5, 6, .5)) +
  # scale_x_continuous(breaks = seq(.5, 6, .5)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.grid = element_blank(),
        panel.background = element_blank()) +
  geom_smooth(method = "lm", se = FALSE, color = "grey", linetype = 2)
```

```
### TTR

### RF

set.seed(123)
# Random Forests

train_data$true_tackle_rate
m1 = rfsrc(true_tackle_rate ~ ., data = as.data.frame(train_data))
# OOB Error Rate
tail(m1$err.rate, 1)
```

```
### TTR

tuning_grid = expand.grid(mtry = c(1, 5, 10, 15, 20), nodesize = c(1, 5, 10, 15, 20))
tuned_models = vector(mode = "list", length = 20)
oob_error_rates = numeric(20)
set.seed(123)
for (i in 1:nrow(tuning_grid)) {
  rf_model = rfsrc(true_tackle_rate ~ ., data = as.data.frame(train_data),
  mtry = tuning_grid[i, 1], nodesize = tuning_grid[i, 2])
  tuned_models[[i]] = rf_model
  oob_error_rates[i] = tail(rf_model$err.rate, 1)
}
# OOB ER for each model
oob_error_rates
```

```
### TTR

# Find the index of the minimum OOB error rate
best_index = which.min(oob_error_rates)
# Best tuning parameters
best_tuning = tuning_grid[best_index, ]
best_tuning

# Extract the random forest model with the best tuning parameters
best_rf_model = tuned_models[[best_index]]
best_rf_model
```

```
### TTR


# Calculate the variable importance for the best model
variable_importance = vimp(best_rf_model)
sort(variable_importance$importance, decreasing = TRUE)
```

```
### TTR

pred_rf = predict(best_rf_model, newdata = test_data)

sqrt(mean((pred_rf$predicted - test_y)^2))*100
```

```
### TSR

### Linear Regression
n = nrow(df_full3)

set.seed(123)

train_ind = sample(1:n, .8*n)

train_data = df_full3[train_ind, ]
test_data = df_full3[-train_ind, ]

test_y = test_data |> dplyr::select(true_stop_rate) |> pull()

m1 = lm(true_stop_rate ~ ., data = train_data)
summary(m1)

set.seed(123)

m2 = step(m1, trace = 0)
summary(m2)

sqrt(vif(m2))
# A LOT of mulitcollinearity

# cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, med_ac
#
# abs(cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, me
```

```r
# Avg Dist Ball, Avg Dist BC, BMI
# Speeds & Accelerations, BMI, Height & Weight
# Plays

m3 = lm(true_stop_rate ~ avg_dist_bc, data = train_data)
summary(m3)

par(mfrow = c(2,2))
plot(m3)
```

```r
### TSR

set.seed(123)

x_train = as.matrix(scale(train_data[, -(ncol(train_data) - 1)]))  # Standardize predictor variables fo
y_train = as.matrix(train_data$true_stop_rate)

x_test = as.matrix(scale(test_data[, -(ncol(test_data) - 1)]))  # Standardize predictor variables for t
y_test = as.matrix(test_data$true_stop_rate)


ridge_model = cv.glmnet(x_train, y_train, alpha = 0)

plot(ridge_model)

best_lambda = ridge_model$lambda.min

final_model = glmnet(x_train, y_train, alpha = 0, lambda = best_lambda)

final_model$beta

pred = predict(final_model, newx = x_test)

sqrt(mean((pred - y_test)^2))*100
```

```r
### TSR

# By Average
m_avg = rep(df_full3 |> summarise(mean(true_stop_rate)) |> pull(), 103)

sqrt(mean((m_avg - y_test)^2))*100

# By Position
m_pos = lm(true_stop_rate ~ pos_group_num, data = train_data)
summary(m_pos)

pred = predict(m_pos, newdata = test_data)

sqrt(mean((pred - y_test)^2))*100

pred = predict(m3, newdata = test_data)

sqrt(mean((pred - y_test)^2))*100
```

```
### TSR

### KNN

error = numeric(50)

set.seed(123)

for (i in 1:50) {

  knn.fit = kknn(true_stop_rate ~ ., train = train_data,
                 test = test_data |> dplyr::select(-true_stop_rate),
                 k = i, kernel = "rectangular")

  test.pred = knn.fit$fitted.values

  error[i] = sqrt(mean((test.pred - (test_data |> dplyr::select(true_stop_rate) |> pull()))^2))


}

min(error)*100
```

```
### TSR

K = which.min(error[error != 0 & !is.na(error)])
K

knn.fit = kknn(true_stop_rate ~ ., train = train_data,
               test = test_data |> dplyr::select(-true_stop_rate),
               k = K, kernel = "rectangular")

test.pred = knn.fit$fitted.values


sqrt(mean((test.pred - (test_data |> dplyr::select(true_stop_rate) |> pull()))^2))


knn_df = data.frame(knn = test.pred, actual = test_data |> dplyr::select(true_stop_rate) |> pull())


ggplot(knn_df, aes(x = knn, y = actual)) +
  geom_point() +
  labs(title = "KNN Predictions",
       # subtitle = "RMSE = .66; k = ?",
       x = "KNN Prediction",
       y = "TT Rate") +
  # scale_y_continuous(breaks = seq(.5, 6, .5)) +
  # scale_x_continuous(breaks = seq(.5, 6, .5)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.grid = element_blank(),
```

```
        panel.background = element_blank()) +
  geom_smooth(method = "lm", se = FALSE, color = "grey", linetype = 2)
```

```
### TSR

### RF

set.seed(123)
# Random Forests

m1 = rfsrc(true_stop_rate ~ ., data = as.data.frame(train_data))
# OOB Error Rate
tail(m1$err.rate, 1)
```

```
### TSR

tuning_grid = expand.grid(mtry = c(1, 5, 10, 15, 20), nodesize = c(1, 5, 10, 15, 20))
tuned_models = vector(mode = "list", length = 20)
oob_error_rates = numeric(20)
set.seed(123)
for (i in 1:nrow(tuning_grid)) {
  rf_model = rfsrc(true_stop_rate ~ ., data = as.data.frame(train_data),
  mtry = tuning_grid[i, 1], nodesize = tuning_grid[i, 2])
  tuned_models[[i]] = rf_model
  oob_error_rates[i] = tail(rf_model$err.rate, 1)
}
# OOB ER for each model
oob_error_rates
```

```
### TSR

# Find the index of the minimum OOB error rate
best_index = which.min(oob_error_rates)
# Best tuning parameters
best_tuning = tuning_grid[best_index, ]
best_tuning

# Extract the random forest model with the best tuning parameters
best_rf_model = tuned_models[[best_index]]
best_rf_model
```

```
### TSR


# Calculate the variable importance for the best model
variable_importance = vimp(best_rf_model)
sort(variable_importance$importance, decreasing = TRUE)
```

```
### TSR

pred_rf = predict(best_rf_model, newdata = test_data)
```

```r
sqrt(mean((pred_rf$predicted - test_y)^2))*100
```

```r
### EPA_Stop

### Linear Regression

n = nrow(df_full4)

set.seed(123)

train_ind = sample(1:n, .8*n)

train_data = df_full4[train_ind, ]
test_data = df_full4[-train_ind, ]

test_y = test_data |> dplyr::select(avg_stop_epa) |> pull()

m1 = lm(avg_stop_epa ~ ., data = train_data)
summary(m1)

set.seed(123)

m2 = step(m1, trace = 0)
summary(m2)

sqrt(vif(m2))
# A LOT of mulitcollinearity

# cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, med_ac
#
# abs(cor(df_full2 |> dplyr::select(avg_dist_bc, avg_dist_ball, speed_10, var_speed, acc_10, acc_25, me

# Avg Dist Ball, Avg Dist BC, BMI
# Speeds & Accelerations, BMI, Height & Weight
# Plays

m3 = lm(avg_stop_epa ~ bmi, data = train_data)
summary(m3)
```

```r
### EPA_Stop

set.seed(123)

x_train = as.matrix(scale(train_data[, -(ncol(train_data) - 1)]))  # Standardize predictor variables fo
y_train = as.matrix(train_data$avg_stop_epa)

x_test = as.matrix(scale(test_data[, -(ncol(test_data) - 1)]))  # Standardize predictor variables for t
y_test = as.matrix(test_data$avg_stop_epa)


ridge_model = cv.glmnet(x_train, y_train, alpha = 0)

plot(ridge_model)
```

```r
best_lambda = ridge_model$lambda.min

final_model = glmnet(x_train, y_train, alpha = 0, lambda = best_lambda)

final_model$beta

pred = predict(final_model, newx = x_test)

sqrt(mean((pred - y_test)^2))
```

```r
### EPA_Stop

# By Average
m_avg = rep(df_full4 |> summarise(mean(avg_stop_epa)) |> pull(), 103)

sqrt(mean((m_avg - y_test)^2))

# By Position
m_pos = lm(avg_stop_epa ~ pos_group_num, data = train_data)
summary(m_pos)

pred = predict(m_pos, newdata = test_data)

sqrt(mean((pred - y_test)^2))

pred = predict(m3, newdata = test_data)

sqrt(mean((pred - y_test)^2))
```

```r
### EPA_Stop

### KNN

error = numeric(75)

set.seed(123)

for (i in 1:75) {

  knn.fit = kknn(avg_stop_epa ~ ., train = train_data,
                 test = test_data |> dplyr::select(-avg_stop_epa),
                 k = i, kernel = "rectangular")

  test.pred = knn.fit$fitted.values

  error[i] = sqrt(mean((test.pred - (test_data |> dplyr::select(avg_stop_epa) |> pull()))^2))


}

min(error)
```

```
### EPA_Stop

K = which.min(error[error != 0 & !is.na(error)])
K

knn.fit = kknn(avg_stop_epa ~ ., train = train_data,
               test = test_data |> dplyr::select(-avg_stop_epa),
               k = K, kernel = "rectangular")

test.pred = knn.fit$fitted.values


sqrt(mean((test.pred - (test_data |> dplyr::select(avg_stop_epa) |> pull()))^2))


knn_df = data.frame(knn = test.pred, actual = test_data |> dplyr::select(avg_stop_epa) |> pull())


ggplot(knn_df, aes(x = knn, y = actual)) +
  geom_point() +
  labs(title = "KNN Predictions",
       # subtitle = "RMSE = .66; k = ?",
       x = "KNN Prediction",
       y = "TT Rate") +
  # scale_y_continuous(breaks = seq(.5, 6, .5)) +
  # scale_x_continuous(breaks = seq(.5, 6, .5)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        panel.grid = element_blank(),
        panel.background = element_blank()) +
  geom_smooth(method = "lm", se = FALSE, color = "grey", linetype = 2)
```

```
### EPA_Stop

### RF

set.seed(123)
# Random Forests

m1 = rfsrc(avg_stop_epa ~ ., data = as.data.frame(train_data))
# OOB Error Rate
tail(m1$err.rate, 1)
```

```
### EPA_Stop

tuning_grid = expand.grid(mtry = c(1, 5, 10, 15, 20), nodesize = c(1, 5, 10, 15, 20))
tuned_models = vector(mode = "list", length = 20)
oob_error_rates = numeric(20)
set.seed(123)
for (i in 1:nrow(tuning_grid)) {
  rf_model = rfsrc(avg_stop_epa ~ ., data = as.data.frame(train_data),
  mtry = tuning_grid[i, 1], nodesize = tuning_grid[i, 2])
```

```
  tuned_models[[i]] = rf_model
  oob_error_rates[i] = tail(rf_model$err.rate, 1)
}
# OOB ER for each model
oob_error_rates
```

```
### EPA_Stop

# Find the index of the minimum OOB error rate
best_index = which.min(oob_error_rates)
# Best tuning parameters
best_tuning = tuning_grid[best_index, ]
best_tuning

# Extract the random forest model with the best tuning parameters
best_rf_model = tuned_models[[best_index]]
best_rf_model
```

```
### EPA_Stop


# Calculate the variable importance for the best model
variable_importance = vimp(best_rf_model)
sort(variable_importance$importance, decreasing = TRUE)
```

```
### EPA_Stop

pred_rf = predict(best_rf_model, newdata = test_data)


sqrt(mean((pred_rf$predicted - test_y)^2))
```