UNIVERSITY OF CALIFORNIA
Los Angeles

On the Move:
An Analysis of Player Tracking Data to Evaluate Offensive Line Play in the National Football League

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics and Data Science

by

Alejandro Demetres Veroulis

2023

ABSTRACT OF THE THESIS

On the Move:
An Analysis of Player Tracking Data
to Evaluate Offensive Line Play
in the National Football League

by

Alejandro Demetres Veroulis
Master of Applied Statistics and Data Science
University of California, Los Angeles, 2023
Professor Frederic R. Paik Schoenberg, Chair

The purpose of this study is to evaluate the performance of offensive linemen in the National Football League by using play-by-play tracking data to predict a player's annual salary. We ran an algorithm on a frame-by-frame data set of plays during the first 8 weeks of the 2021 NFL season that created new variables based on existing data for every offensive lineman on every play. With this information, we tried to use both an unweighted stepwise regression model and a weighted version that emphasized certain play types, but both did not have appropriate residual distributions. As a result, we used box-cox transformations to log transform the response variable, salary, and this change led to evenly distributed residuals, so we tried this model on the test set and got an RMSE of 4.7. We improved upon this number with a series of random forest models, where we iterated over 5 different numbers of trees, ranging from 50 to 150, and discovered that the lowest RMSE of 4.36 was achieved by the model with 75 trees.

The thesis of Alejandro Demetres Veroulis is approved.

Yingnian Wu

Robert L. Gould

Frederic R. Paik Schoenberg, Committee Chair

University of California, Los Angeles

2023

Dedicated to my family . . .

who has supported me

through my academic journey

and inspired me to keep learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I'd like to thank UCLA, the best school in the world, for my four years in the undergraduate Statistics program and the past two years in the Master of Applied Statistics and Data Science program. I've had the privilege of learning from outstanding professors about topics I'm very passionate about, and I'd like to specifically acknowledge everyone in the UCLA Statistics department that guided my academic journey over the past six years.

Outside of the classroom, I want to acknowledge the various organizations that have allowed me to develop my skills in data science and sports analytics, including WagerWire, Bruin Sports Analytics, UCLA Football, USA Hockey, UCLA Golf, and the UCLA Athletic Department at large. I am blessed to have had so much exposure to data in sports, and I am so thankful for the experiences these groups have given me to expand my knowledge in the realms of sports and statistics.

Finally, I'd like to give a huge shoutout to my entire family, who has supported me with the resources and love needed for me to be successful in school, sports, and whatever else I've pursued. I'd like to give a special thanks to my mom, my dad, my brother, and my "Abu" for motivating me to be the best version of myself and encouraging me to pursue my dreams.

CHAPTER 1

Introduction

Since the inception of the sport of football, team executives and coaches have been obsessed with finding the best players for great value, and they have dedicated countless resources and time towards building better football teams. While the quarterback is most often considered the most important player on the team, there have been varying opinions as to which positions come next. However, an emerging school of thought is that offensive linemen are more valuable than ever before, as they are being recognized more and more for their contributions on the field. Changes in player evaluation methods and the use of analytics have led to evolutions in thought about the value of the offensive line in the National Football League.

1.1 Player Evaluation Methods

Player evaluation methods in the National Football League (NFL) have come a long way since the 1956 NFL Draft. Back then, some young scouts from the Los Angeles Rams left a note to their general manager (GM) about Dick Donlin, an obscure prospect from Hamline College in Minnesota, as a joke. The GM, Tex Schramm, still did his due diligence in researching the prospect, and word spread about Donlin to other teams. While the Rams knew this player may not be great, another team, the Baltimore Colts, selected Donlin 21st overall. However, Donlin was cut before his first game, and the Colts looked like absolute fools for relying on hype [Orr 15]. Clearly, there was not much data or information back then to evaluate players, so teams had to rely on word-of-mouth evaluations and gut instincts.

Towards the end of the 20th century and into the 21st century, traditional scouting ruled the player evaluation field in professional football, with events like the NFL combine and

college Pro Days determining which players had the most talent. Scouts were tasked with traveling the country (and even the world) to find the game's best and brightest stars, and many are old-school football lifers that were wary of the analytics movement. One article captured the sentiment of these scouts perfectly: "The last thing football teams need is a computer spitting out numbers or an analyst with a master's degree who has never played the game calculating ways to capture a winning edge," [Bec 21]. Regardless, the emphasis on traditional scouting has been slightly lessened as of late, thanks in part to the rise of data analytics in sports. In the early 2000's, Philadelphia Eagles executive Joe Banner saw a "competitive advantage in analytics" that inspired him to create the first analytics department in the NFL [Bec 21].

Over the past decade, teams across all the professional sports leagues have embraced data to certain degrees. Most teams have developed data analytics groups that dive deep into the numbers and offer insights that scouts cannot. In fact, ESPN has held a yearly survey for the past 3 seasons asking NFL analytics staffers about the teams with the best and worst analytics departments. Coincidentally, this year, staffers were asked about the hardest position to analyze quantitatively, and 5 of 20 respondents said that offensive line was the most difficult to evaluate [Wal 22]. So, even people within the league today have a hard time coming up with sound methods to analyze linemen, which is the goal of this project.

1.2 Historical Perceptions of Linemen

Throughout the 20th century, linemen had been lauded historically for their value in the run game, where they carved easier paths through the defense for teammates to bolt through to score. Linemen were glorified for their toughness and grit in an era where macho culture and violence were much more prevalent than today [Bec 21]. While linemen are still appreciated for their contributions to the run game, they have brought additional value as pass protectors who prevent the quarterback from being tackled. Before the 21st century, linemen were generally smaller in stature than present-day, which presented problems as defensive linemen became more and more powerful. As the game has evolved from run-dominant to pass-dominant

today, linemen have become heavier, taller, and stronger to combat the freakishly athletic pass rushers of the modern game [Sob 17].

Another antiquated notion is that left tackle is the only important position on the offensive line. Left tackle has been considered a premium position for over 30 years because this position protects the blind side for a right-handed quarterback, so that pressure from defenders can be prevented when the quarterback is least expecting it. However, the other positions along the offensive line are just as important in today's game due to new defensive schemes that put pressure on the quarterback from many directions, not just the left side of the offense. In fact, better pass rushers sometimes come from the right side of the offense anyway, so there have to be competent blockers all along the line in order to protect the quarterback [Sob 17].

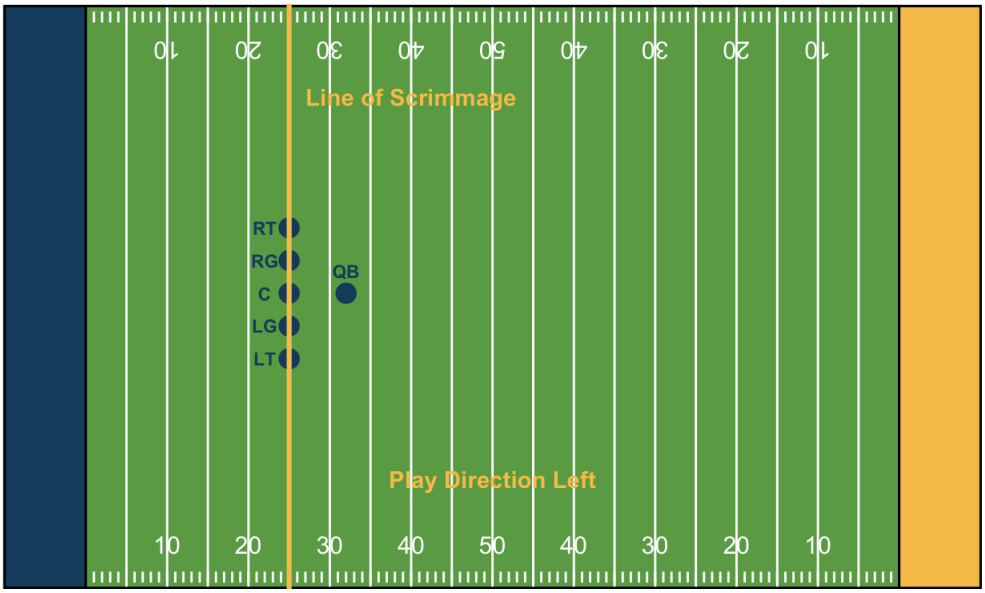

Figure 1.1: Positions along The Offensive Line

If we look at the offensive line unit in Figure 1.1, where the offense is moving to the left (going towards the blue end zone), we see that there are 5 linemen that start on the line of scrimmage (the yard line where the play starts) in front of the quarterback (QB). Namely, the positions from bottom to top are left tackle (LT), left guard (LG), center (C), right guard (RG), and right tackle (RT). The tackles (LT and RT) are tasked with preventing edge rushers on the perimeter of the offensive formation from getting to the quarterback, while the guards (LG and RG) and center (C) all have the role of preventing any defender lined

up across from them on the inside of the formation from getting to the quarterback. Since defensive pressure can come from anywhere in the formation in today's game, it's obvious that all offensive linemen have important roles in protecting the quarterback from defenders.

This idea falls in line with the weak-link framework in team building, where the ideal way to build a team's positional unit (such as an offensive line or defensive secondary) is to ensure that each player within that unit is competent enough to perform at a decent level. This is different from acquiring individual star players for that unit, because the inherent acquisition of stars will constrain the team's ability to get other competent players due to salary concerns, which we will detail further in an upcoming section. The reason the weak-link framework applies to the offensive line specifically is because if one lineman fails to block their defender, the quarterback is going to be disrupted. No matter how good of a job the other linemen did, the result of the play will be undesirable if one individual fails to do their job [Eag 23]. Not to mention, but one study looked at PFF offensive line grades dating back to 2012 and found a 3-win difference between teams with top-5 offensive lines and those with bottom-5 offensive lines. Even more, the study noted that on average, the team that wins the Super Bowl has had a top-12 offensive line over the past decade [Fal 22].

In addition, modern analytics from companies like PFF have given the general population a better understanding of how offensive linemen contribute to the team in today's game. They provide play-by-play grades for every player on every play, as this is their way of seeing how well a player is playing their position relative to their peers across the NFL. More specifically, with regard to their findings on offensive linemen, PFF claims that a good performance by an offensive lineman leads to more clean pockets, where the quarterback has more time to throw the ball and make a productive play downfield [Cha 18]. Not only this, but a quarterback's overall performance is more easily deduced from plays with clean pockets than plays under pressure. This is because pressure plays are often out of the quarterback's control, and these plays produce unstable results. However, clean pocket plays, where the quarterback has ample time to throw and make decisions, have more stable results over time, simply because the quarterback has less to contend with. By providing the quarterback with a high volume of clean pockets, the coaching staff is able to better evaluate the quarterback position in the

long run, so that they can make a decision as to whether they should keep him or move on to another quarterback [Cha 18]. Next, we will discuss the issues around the NFL salary cap, which constrains teams from spending a certain dollar amount on all player acquisitions each year.

1.3 Salary Cap

The NFL salary cap is a strict limit on the money teams in the league can spend on all player acquisitions for a given season. While other sports have salary caps as well, the NFL is the only major professional sports league with a hard cap, where teams absolutely cannot spend over the salary cap amount for any reason [Dec 22]. On the other hand, leagues like the National Basketball Association (NBA) employ soft caps, where there is an agreed-upon upper bound that teams can spend up to, but teams have to pay a luxury tax to the league if they happen to spend over the limit [Pin 23].

For the NFL, the salary cap is calculated by taking a proportion of league revenues from the previous year and dividing that figure by 32, for the 32 teams in the league. Specifically, this proportion is roughly 48 percent; since the season revenue heading into 2022 totaled 433.755 million dollars, the salary cap for the 2022 season was 48 percent of this total revenue, which was approximately 208.2 million dollars [Dec 22]. This was the total amount that each team could spend on player acquisitions for the 2022 season.

There are several motivations behind the NFL's hard cap. For one, having the cap in place prevents teams from overspending on players, which mitigates financial risks for team owners. In addition, since all teams have the same spending limit, the league can achieve more parity throughout the season, so that teams with varying amounts of cash can be on a more level playing field [Dec 22].

1.4 Roster Building with The Salary Cap Constraint

Clearly, the salary cap is a problem that all teams have to plan around in order to achieve an optimal roster construction. So, teams need to decide which players and positions they need to invest in to optimize team success in the NFL. Now, each team will have its own priorities depending on its current roster construction. A team with a league-worst record from the previous season would most likely want to focus on making the roster younger and more talented, while a team that won the Super Bowl the previous year would want to continue their success by keeping their roster fairly similar.

In one research paper, the author examines different roster building methods to see which ones are successful, even with the salary cap as a constraint. First, he identified the salaries of each position under the assumption of an efficient market and wanted to determine which positions were being undervalued. By using some linear regression models, he found that quarterback and guard were among the premium positions that deserved more investment from NFL teams [Mul 16]. Since the quarterback is the leader of a team's offensive attack, it's clear why this would be a valuable position. However, the guard position, which is often valued less than the tackle positions on the offensive line, is a bit of a surprise, given the historical emphasis on the importance of the tackle position. While guards are known more for their role in facilitating run plays, they also have important jobs in pass protection, especially as interior defensive linemen like Aaron Donald have become quicker and stronger. Hence, we will explore the play-by-play data of all offensive linemen, not just offensive tackles, to see which linemen are generating the most on-field value.

CHAPTER 2

Data Snapshot

The data used in this project is provided by the NFL Big Data Bowl's 2023 competition, where the goal is to evaluate the performance of offensive linemen in the National Football League (NFL) based on data from the 2021 NFL season. To do this, the plan is to predict each offensive lineman's salary from the 2021 season using their tracking data from each play they were involved in. There are several data (csv) files available to the public that can be downloaded directly from the competition site, all of which capture information about the first 8 weeks of the 2021 NFL season [NFL 23]. Although this may seem like a limited amount of data, the 17-game NFL season is relatively short compared to other major North American sports leagues, like Major League Baseball (162 games), the National Hockey League (82 games), or the National Basketball Association (82 games). Also, when you compare the length of each sport's season, we find that the NFL's season is roughly 150-160 days (depending on season-to-season variability), while all of the other major sports have season durations over 200 days [Hef 22]. Therefore, front office executives for NFL teams have to use the limited data they may have in the middle of a season to make decisions about which players they should pay going into the following season. By using a half-season's worth of data, we can face a similar resource-constrained challenge that teams routinely brave throughout an NFL season.

2.1 Games and Plays Data

The first file is the games data set (games.csv), which has a gameId variable that identifies each unique game played in the first 8 weeks of the 2021 season; in total, there were 122

games in this data set. Each row represents one game played during this time interval. In addition, each game has details about the two teams that were involved and the week, day, and time the given game took place. The second file is the plays data set (plays.csv), which has a playId variable that identifies each unique play and a gameId that identifies which game that play occurred in. Each row is an individual play that occurred during any of the games in the first 8 weeks of the 2021 season. All of these plays are "passing plays," which means that the initial design of the play must be for the quarterback to throw the ball, not for a player to just run with the ball. This way, we are able to evaluate an offensive lineman's ability to protect the quarterback before he throws the ball. With this in mind, there are several different outcomes that could take place from an initial passing play. The passResult variable is a categorical variable that dictates the result of the play. Possibilities include completed pass (C), incomplete pass (I), quarterback sack where the quarterback is tackled (S), intercepted pass where the ball is thrown to the other team (IN), and a scramble where the quarterback runs with the ball (R). Since the last outcome does not involve a throw, we decided to omit all plays with a passResult equal to "R." Furthermore, there is a playDescription variable that gives a short synopsis of what happened on a given play; there are other variables that mention the time the play happened, the yard line where the play took place, the team in possession of the ball, penalties, and much more. After filtering this data set, we were left with a total of 8,108 plays [NFL 23].

2.2 Players and Scouting Data

The third file is the players data set (players.csv), which has a comprehensive list of players that were involved in any of the games during the first 8 weeks of the 2021 season. Players are uniquely identified by their 5-digit nflId, and there are other variables detailing their height, weight, age, school, and position. Each of the 1,679 rows represented an individual player's information. Since there were so many positions to keep track of, we made a new general position variable to group each player into 5 groups: quarterback, offensive lineman, other offensive positions (wide receiver, running back), defensive secondary positions (safety,

cornerback), and other defensive positions (defensive lineman, linebacker). This way, it would be easier to filter out players by position group when analyzing plays later on. As for the fourth file, the Scouting data set (pffScoutingData.csv) is provided by Pro Football Focus (PFF), a football analytics company that provides player evaluation metrics for plays in the NFL and college football. Each of the 188,254 rows represented a single player's activity on a given play. More specifically, this data set captured a player's "role" during each play they were on the field; potential roles include pass, pass route, pass block, pass rush, and coverage. The roles we are interested in are pass (done by the quarterback), pass block (usually done by an offensive lineman), and pass rush (usually done by a defensive lineman). In order to evaluate an offensive lineman's performance, we need consider all instances where the role is pass block. In the aforementioned data set, there are variables describing whether the lineman allowed a hit or sack on the quarterback, whether the opposing defender got past the lineman, and the nflId of the opposing defender that the lineman attempted to block [NFL 23].

2.3 Tracking Data

Next, there are 8 files of tracking data provided by Next Gen Stats from the NFL Big Data Bowl website, one for each week for Weeks 1-8 in the 2021 NFL season. Each of the 8 files has the same structure, as each row corresponds to a single player's coordinates, speed, acceleration, and orientation during a single frame of a play. For each player, there are dozens of frames per play, so each week has roughly 1 million frames of player tracking data. In terms of how coordinates are determined, the range of x-coordinates is 0 to 120 yards, and x increases from left to right on the field, which is illustrated in Figure 2.1. Similarly, the range of y-coordinates is 0 to 53.3 yards, and y increases as you go from the home sideline to the visitor sideline. As for orientation, it increases in a clockwise fashion, starting with the 0 degree mark pointing directly towards the visitor sideline [NFL 23].

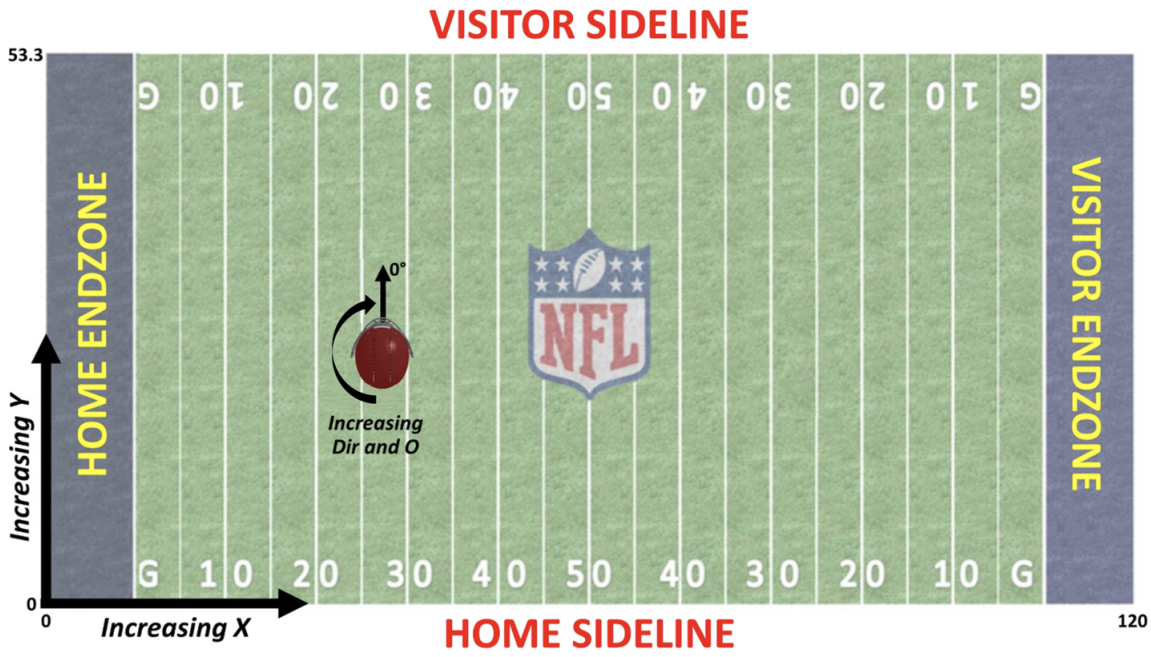

Figure 2.1: A diagram of the coordinates of an NFL field

2.4 Salary Data

Finally, we retrieved salary data from Spotrac, a sports site that keeps track of player salary information across many different sports, including the NFL [Spo 21]. We used the rvest package in R to scrape the top 100 offensive lineman salaries from a table on the site, but we manually copied the rest of the names and salaries because the scraper only worked for the first 100 rows. Then, we removed dollar signs and commas from the salaries using the gsub function in R by replacing these symbols with blank spaces. This way, we could make the salary variable numeric in order to use it as our response variable for our models. After this, there were some clear name mismatches between the display names from the players data set provided by the Big Data Bowl and the player names provided by Spotrac. To correct for these discrepancies, we found the differences between these lists, and changed a name from the Spotrac list to match a similar name from the NFL Big Data Bowl list. For instance, there was a player named "Shaq Mason" in the Spotrac list, but there was a "Shaquille Mason" in the NFL Big Data Bowl list. These instances refer to the same player, an offensive lineman for the Tampa Bay Buccaneers, so these names need to be identical into

order to keep records consistent across data sets. Hence, we manually changed "Shaq Mason" from the Spotrac list to "Shaquille Mason" so that we could perform an inner join between the two lists. After editing two other names, the inner join between the lists produced a data set of player salaries. However, there were two players named "Connor McGovern," which led to a many-to-many relationship from the inner join. This means that R did not know which salary belonged to which Connor McGovern, so it considered all possibilities and assigned each Connor McGovern two salaries. So, we manually filtered out the incorrect salaries, which left one salary for each Connor McGovern. This finalized response data set, called salplayersfin, contains all offensive linemen with both a posted salary and playing time during the first 8 weeks of the 2021 season; each row has the player's name, nflId, and salary.

CHAPTER 3

Data Preparation

The first step in processing the aforementioned data files was to load all of them into R using the read.csv function. As previously mentioned, we filtered out plays classified as scrambles from the plays data set, because these instances would not accurately evaluate the ability of offensive linemen to protect the quarterback before a throw. The next step was to figure out how to properly combine all of this data. In order to do this, we identified the key variables within each data set so that we could join them.

3.1 Joining Games with Plays Data

First of all, the games data set has one key variable, gameId, that uniquely identifies the game played. Likewise, the plays data set has the same gameId variable, in addition to the key playId variable identifying each unique play. Hence, joining the plays and games data sets by the gameId variable was a logical move; we call this new joined data set fullplays. Each row still represents a single play, but now we have game-level details appended to each play. The only amendment made to fullplays after this join is that one play had a missing value for the yard line where the play took place, so we looked at that row and manually input the yard line based on two variables that were present, the yard line side variable (what side of the field the ball is on) and the yard line number variable (a yardage from 0 to 50).

3.2 Joining with Scouting and Tracking Data

Once we had the fullplays data set, we were able to focus on joining with more data sets. Namely, the pff data set has the same key variables of gameId and playId as the fullplays data set, so we joined these data sets based on those shared key variables. This newly joined data set is called pffplays. Each row still represents a single player's role during a given play, but now play and game details are included as well. As a result of the join, the number of player roles in the data set was reduced from 188,254 to 178,376, so a vast majority of the data was kept. Then, we shifted our focus to the week-by-week tracking data sets, which have the key variables of gameId, playId, and nflId (player identification). Since gameId, playId, and nflId are all common key variables of the pffplays data set and each week's tracking data set, we performed an inner join between the pffplays data set and the individual week-by-week tracking data sets. So, we performed 8 inner joins, one for each of the weekly tracking data sets, to get 8 newly joined data sets. Each of these data sets has rows that represent a player's movement during a frame of a given play, with extra info about the game and play. There were roughly 800,000 to 1 million rows of data for each week. Once we have the 8 joined data sets, we can use the rbind function to combine the rows of all of these data sets to form a large data set that has tracking data for all 8 weeks of the 2021 NFL season. We call this data set pffweeks, and it has 7,419,566 player tracking observations.

3.3 Joining with Players Data and Adding Some Variables

With respect to the players data set, after we followed the previously mentioned process of grouping each player into one of five position categories, we joined the pffweeks data set to the players data set since both have the key variable of nflId in common. This way, the resulting data set, which is called pffweeksfull, has more information about the player involved. The next step is to add some variables to this data set by using current data. Using the mutate function from the dplyr package, we created a couple of different variables. The first one was score differential, a binary variable stating whether the game was within 1 score or 2+ scores.

To do this, we looked at the absolute value of the difference in score between the two teams and used the casewhen function to assign one of two possibilities, 1 score or 2+ scores. A one-score game is defined as any game that the winning team is ahead by 7 or fewer points; a touchdown and extra point scored by the losing team would at worst tie the game, since a touchdown and extra point is worth 7 points. Thus, we assigned score differential to be 1 score if the absolute value of the margin between the two teams was less than or equal to 7 points, and 2+ scores in all other situations. The next created variable was "half," which gave the current half the game was in; the two possibilities are either the 1st half (1) or the 2nd half (2). To get this variable, we had to look at the already existing quarter variable, a number 1 through 4 that represented the current quarter of the game. Once again, we used the casewhen function to assign the half variable a value of 1 if the quarter was 2 or less, and a value of 2 if the quarter was greater than 2.

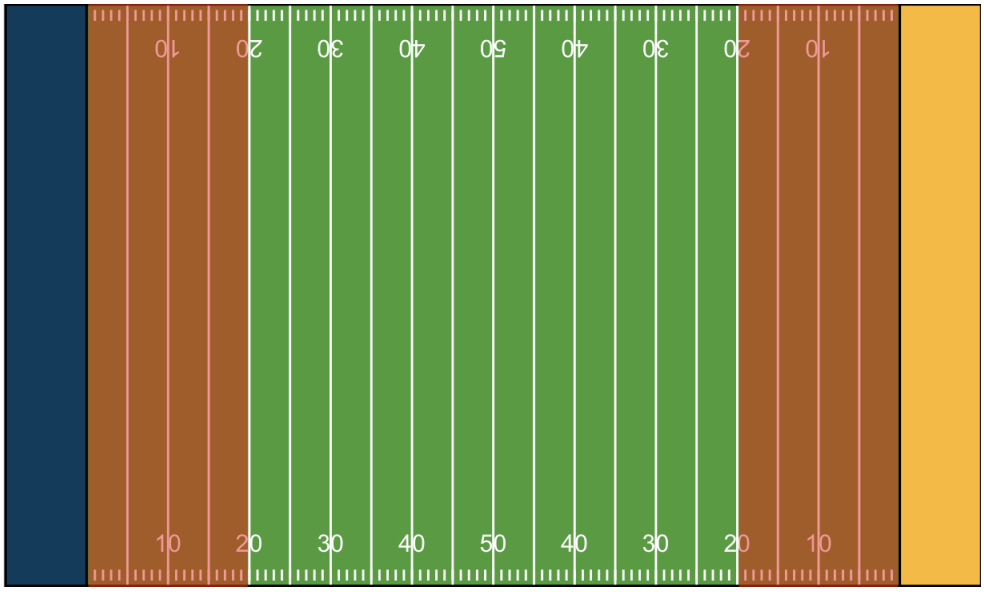

Figure 3.1: Illustration of Red Zone Areas

The third variable created was the categorical "distance to score" variable, which has two options: red zone or non-red zone. The red zone is defined as the final 20 yards a team needs to go in order to score a touchdown, and plays within the red zone are often the most important plays of a game (see Figure 3.1). To determine whether or not a play was in the red zone, we had to use the playDirection variable to see whether the play was going

to the right or to the left, as well as the absoluteYardlineNumber variable to see where the play took place on the field. This yard line number ranges from 0 to 120 yards and follows the previous figure's x-coordinate. If the play direction was to the left, we would subtract 10 from the absoluteYardlineNumber in order to get the remaining yardage the team with the ball needed to score. If the play direction was to the right, we would subtract the absoluteYardlineNumber from 110 in order to get the remaining yardage the team with the ball needed to score. For either playDirection scenario, we would get a yardage from 1 to 99 representing the yardage left for a team to score. Then, if this yardage is 20 yards or less, the "distance to score" variable would be red zone. On the other hand, any yardage greater than 20 would result in the "distance to score" variable being non-red zone. To make these assignments, we used the casewhen function with the previously stated logic.

3.4 Missing Values and Data Filtering

There were a couple other adjustments we made to the pffweeksfull data set once the variable creations above were completed. First, a plethora of NA's were scattered across several indicator variables, such as those indicating whether an offensive lineman allowed a hit, a hurry, or a sack on a given play. To fix this, we assigned all of these NA's a value of 0, which would indicate that a hit, hurry, or a sack wouldn't have happened on a given play. Since these occurrences are somewhat rare, this was a reasonable assumption to make; after all, stat keepers would most likely have made a note of when these events actually did take place. This left us with 75 variables in the data set, but we only needed a subset of these in order to carry out our analysis. So, we used the filter function in the dplyr package to reduce the number of variables from 75 to 26. This final version of the pffweeksfull data set was what we used for the play-by-play algorithm, which is described in the next chapter.

CHAPTER 4

Feature Engineering Using a Play-by-Play Algorithm

Despite having 26 variables in our pffweeksfull data set, the majority of these pertain to details about the game and play, which wouldn't directly help us in a model about player performance. In fact, other than a few categorical variables about the player's role during the play, the only variables of use would be the tracking data metrics. Specifically, the tracking data metrics include a player's speed, acceleration, orientation, and coordinates on the field during many frames throughout each play. While these are helpful, we can use these metrics to create even more variables for modeling purposes, which is where a play-by-play algorithm comes into play. This algorithm iterates through every single play in our data set to capture players' information on each play. Not only does this involve using the given tracking metrics, but we create new variables for each player within the algorithm based on the tracking data provided.

4.1 Algorithm Preparations

Before formulating the algorithm, we first defined our set of plays that we would iterate over. To do this, we took the pffweeksfull data set and selected distinct combinations of the gameId and playId variables. This list of gameId and playId variables represents all distinct plays from our data set; we use these variables as indices for our algorithm. Then, we extract all players listed as offensive linemen from the players data set; these are all the players that we need to get data for from the algorithm.

4.2 Looking at A Single Play

After this, we chose a single play from the distinct plays list. This is purely for illustrative purposes, as the algorithm performs this process for all plays. To get this play's data, we used dplyr's filter function to get all rows from the pffweeksfull data set that had the corresponding gameId and playId values from that play; we stored this play's data as a data set called pffsample. With this single play to focus on, we use the filter function to extract all frames that fulfilled two criteria: one, the frame corresponds to an offensive lineman involved in the play, and two, that this lineman had an opposing player to block during the play. This means that the blockedPlayer variable needed to identify an opposing player's nflId so that we would know which player our offensive lineman blocked. So, we could track an offensive lineman's performance throughout the play relative to his opponent's movement patterns. Once we had these filtered frames, it was imperative to store the distinct nflId values that corresponded to all offensive linemen involved during the play. This way, we have a list of the names and IDs of the offensive linemen we want in a data frame, where we can later append their metrics from the play. To create the skeleton data frame of results for a play, which we call activeol, we include empty columns for each of the variable names we either kept or created from the initial pffweeksfull data set. Later on in this chapter, we will review more details about these variables.

4.3 Plays without Blocked Players

As previously mentioned, we must get a given play's data, store that data in a data frame called pffsample, and extract all unique offensive linemen from that play. These offensive linemen have their names and IDs stored in a data frame called activeolids. Now, there are rare cases where no blocked players are identified on a given play, in which case there would be no offensive linemen with blocked players that are in the activeolids data frame. Since we wouldn't be able to get any data from this play, we put a check in place on the activeolids data frame. Specifically, if this data frame has no rows present, we simply move on to the

next play. Fortunately, there were only two plays that we omitted for this reason, which left us with 8,106 total plays. With these cases settled, we can use the filter function to get frames from pffsample that directly involve the offensive linemen we want, which allows us to move on to the variables we want to get from our algorithm. These frames are put in a data frame called playerdata, where a single offensive lineman's frames from a play are stored.

4.4 Algorithm Structure for Capturing Existing Variables

Before creating any new variables, we want to identify which pre-existing variables we want from each play. To start, the obvious ones include the lineman's player ID and name, the gameId, and the playId. Then, there are other variables that give us more context behind the play itself, such as the "distance to score" variable that identifies whether or not the play took place in the red zone, the score differential variable that says whether or not the two teams are within 7 points of one another, and the half variable that mentions whether the play took place in the 1st or 2nd half of the game. In terms of variables relating to player roles, there are binary variables stating whether the offensive lineman allowed a hit, hurry, or sack on the quarterback. Not to mention, there is the important blocked player ID that tells us the opposing player our offensive lineman blocked during the play. In order to get these variables into our skeleton data set (activeol), we assigned the first instance of these variables from the playerdata data set to the corresponding variable names in activeol. We are able to do this because variables like half and score differential are constant across all frames of the play.

4.5 Creating New Variables Based Only on Lineman's Movement

Now we can move onto variables that we create using the algorithm. The first new variable is a binary indicator of whether or not the lineman committed a penalty during the play. For context, on each play there are up to two nflId fields defined for players who committed penalties on a given play. So, we used the casewhen function to define the binary penalty

variable as "1" if either penalty field matched the lineman's nflId; this penalty variable is defined as "0" for all other cases. The next set of variables were generated from the orientation data, where a player's orientation is measured from 0 to 360 degrees based on how they are positioned during a frame of a play. In the algorithm, we made an "orientation changes" vector (we will refer to it as ochanges) that found the absolute difference of the n^{th} and $(n+1)^{th}$ orientations across all frames for the lineman during the play, as we can see in Equation 4.1. With this vector, we created three new variables: the maximum, the mean, and the median of the orientation changes.

$$\text{ochanges} = \sum_{n=2}^{\#\text{ of frames}} |\text{orientations}_n - \text{orientations}_{n-1}| \tag{4.1}$$

In addition, we created maximum, mean, and median variables for the already existing speed and acceleration vectors. Next, we used the x-coordinates of each lineman to find their maximum distance traveled backwards during a play, relative to their starting position. In order to protect the quarterback and stay in front of the opposing defender, offensive linemen have to make a move backwards to start the play. To a football outsider, this may be counterintuitive since the rest of the offense is moving in the other direction. As we can see in Figure 4.1, we can see that for a play moving left to right on the field, the lineman moves right to left; so, we want to capture the distance between the two different points in the diagram. The right-most point is the player's starting position, while the left-most point is the place where the lineman moved the farthest back during the play. We took the absolute difference between the initial x-coordinate and every other frame's x-coordinate for a given player, which gave us a vector of x-coordinate differences (called xchanges). This process is summarized in Equation 4.2. We take the maximum of the xchanges vector to get our variable representing the maximum distance traveled backwards during a play for an offensive lineman.

$$\text{xchanges} = \sum_{n=2}^{\#\text{ of frames}} |\text{x}_1 - \text{x}_n| \tag{4.2}$$

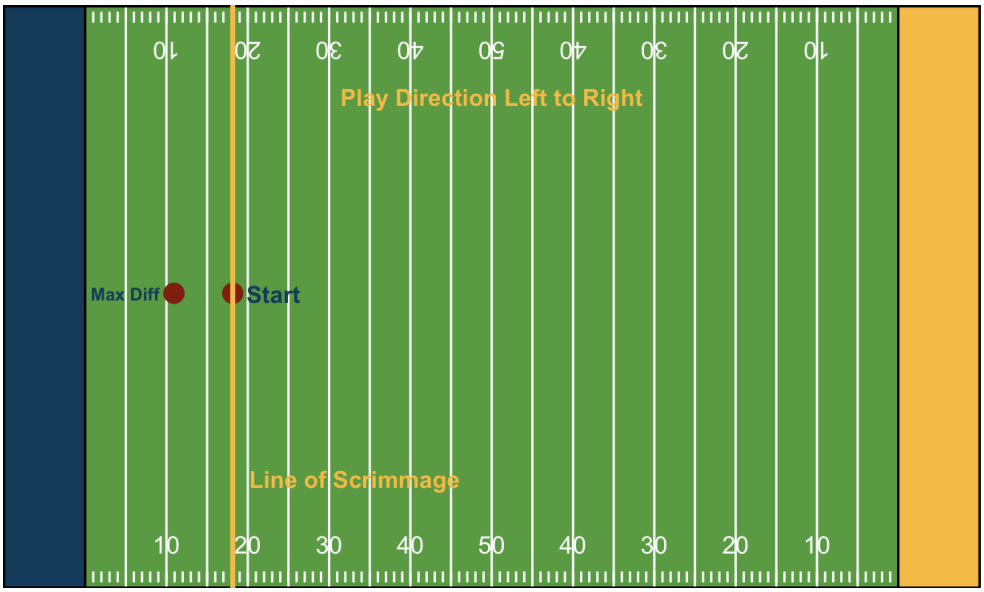

Figure 4.1: Illustration of A Lineman's Movement Backwards for A Play

In Figure 4.1, this distance is roughly 7.23 yards, as the starting x-coordinate is 28.15 yards, while the other point sits at 20.92 yards. We add 10 yards to the standard field yardages due to the end-zone (in dark blue) being 10 yards deep. Then, we shift our focus to the next few variables, which all pertain to the distances traveled by the player from frame to frame during each play. We found a good way of determining the distances traveled between frames with the distance formula, so we created a vector of distances traveled between each frame during the play (we'll call this vector disttrav). In Equation 4.3, we see how the distances are calculated from frame to frame, so that all distances are stored in the disttrav vector. Here, we need the x-coordinates and y-coordinates from all frames so that we can carry out the distance formula. From this disttrav vector, we exclude any elements that are zero. Then, we take the median and mean of this non-zero vector to get two new per-frame distance metrics. In addition, we take the sum of the elements in the disttrav vector to get the total distance traveled during the play.

$$\text{disttrav} = \sum_{n=1}^{\text{\# of frames - 1}} \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \tag{4.3}$$

4.6 Creating New Variables Based on Defender's Movement

Since each offensive lineman has a corresponding blocked player, we can filter each play's data down to frames that match the player ID of the blocked player (the defender). This way, we can get information about the defender's movement, so we can compare the movements of the lineman and the defender to evaluate the lineman's performance. The first variables we created pertain to the distances between the lineman and the defender during all frames of a play. Since we now have access to x-coordinates and y-coordinates of both parties, we can employ the distance formula at each frame to make a vector of distances, which we'll name opoldist. We see in Equation 4.4 how the distance formula for capturing the proximity of the lineman (referred to as ol) and defender (referred to as def) works. From the opoldist vector, we derive its maximum, median, and mean as new variables.

$$\text{opoldist} = \sum_{n=1}^{\#\text{ of frames}} \sqrt{(\text{defx}_n - \text{olx}_n)^2 + (\text{defy}_n - \text{oly}_n)^2} \tag{4.4}$$

We also have vested interest in the x-coordinate differences between the lineman and the defender throughout the play. In order to be a good pass rusher, the defender has to possess a great upfield burst that allows them to pass the offensive lineman and attack the quarterback. This burst is all about speed and timing; not only does the defender have to be quick, but he also has to time his rush according to when the ball is snapped [Con 16]. Conversely, the offensive lineman's job is to prevent a good rush from the defender so that the quarterback has more time to throw the ball and make a good play. Hence, it is extremely important for the offensive lineman to keep the defender in front of him, so that the defender cannot get to the quarterback.

This is where x-coordinate differences come into play; we want to see how well the lineman is keeping the defender in front of him. To do this, we have to consider scenarios where the play direction is to the left and where the play direction is to the right. In Figure 4.2, we observe two different plays, with Play 1 on the left side of the field and Play 2 on the right side of the field. Since the x-coordinate ranges from 0 to 120 yards and increases from left to right, we have to calculate x-coordinate differences according to the play's direction.

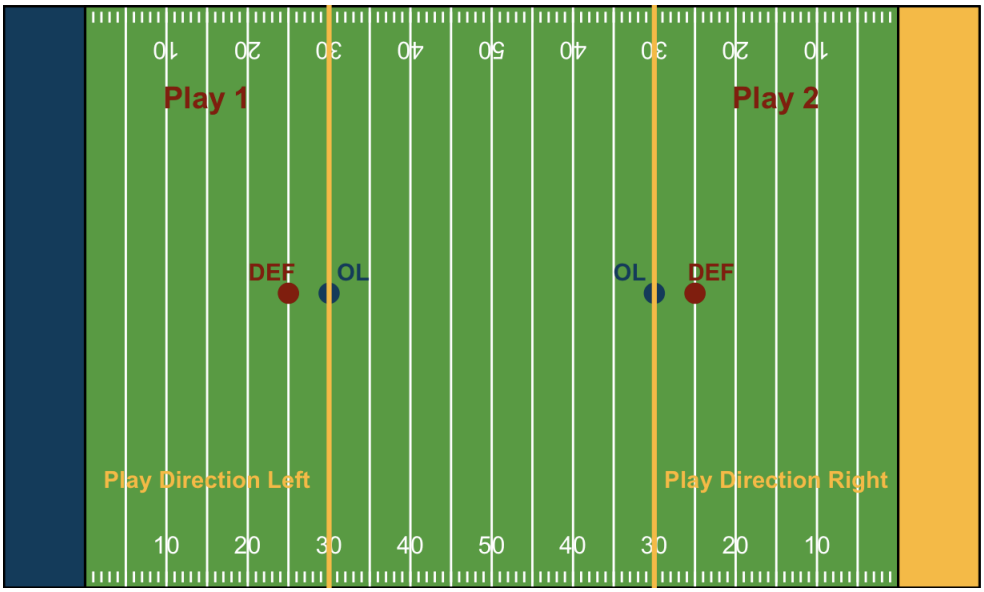

Figure 4.2: Lineman Proximity to Defender Based on Play Direction: Two Cases

For Play 1, the offense in blue is moving to the left towards the blue end zone, so we know the initial x-coordinate of the offensive lineman is greater than that of the defender. In this scenario, we calculate x-coordinate differences by subtracting the defender's x-coordinate from the lineman's x-coordinate, which would result in a difference of 5 yards for Play 1. As a result, if the defender got behind the lineman by moving farther to the right, then this x-coordinate difference would become negative.

As for Play 2, the offense in blue is moving to the right towards the gold end zone, so we know the initial x-coordinate of the offensive lineman is less than that of the defender. In this scenario, we calculate x-coordinate differences by subtracting the lineman's x-coordinate from the defenders's x-coordinate, which would also result in a difference of 5 yards. As a result, if the defender got behind the lineman by moving farther to the right, then this x-coordinate difference would also become negative. This methodology ensures that no matter the play's direction, the x-coordinate differences between the lineman and the defender remain the same. From here, we can find x-coordinate differences for all frames during a play, and once we have a vector of differences, we take the minimum, maximum, median, and mean of these distances as new variables.

4.7 Creating New Variables Based on Defender-QB Proximity

Another important step is accounting for the defender's proximity to the quarterback (QB) during all frames of a play. This gives us a good indication of how well the defender is rushing the quarterback, and in the process, getting by the offensive lineman. So, if a defender has a small distance from the quarterback, this tells us that the offensive lineman hasn't done a good job of protecting the quarterback. On the other hand, a great offensive lineman will consistently prevent the defender from getting to the quarterback during the entirety of the play, acting as a moving fortress. We see in Figure 4.3 that the lineman is in the middle of the defender and the quarterback, and the quarterback remains behind the offensive lineman. The goal is for the lineman to remain in between the defender and the quarterback throughout the play.

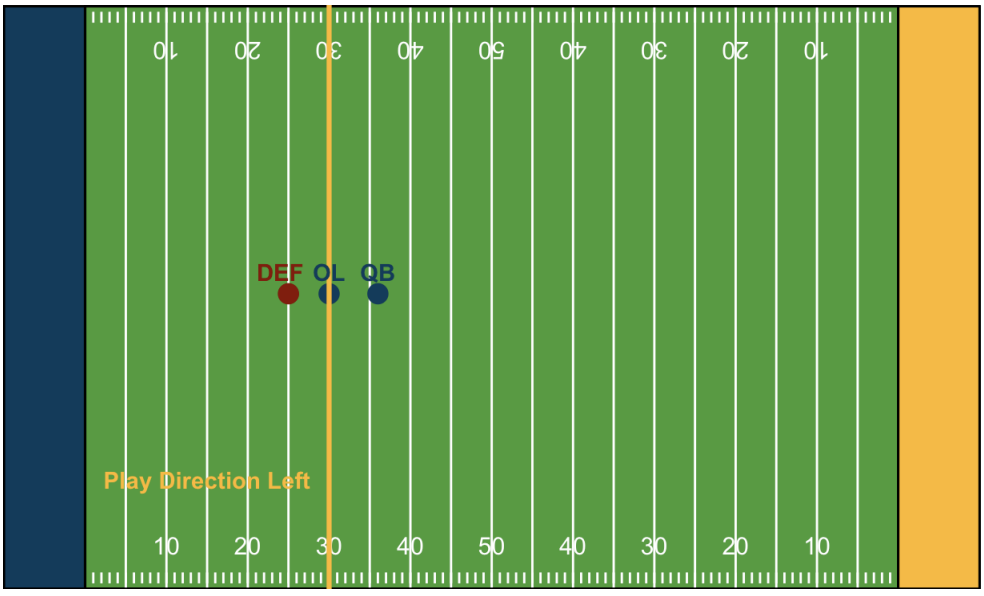

Figure 4.3: Defender Proximity to Quarterback Impeded by Lineman

To find the distances between the defender and quarterback during frames of a play, we have to first acquire all frames of the play that pertain to the quarterback. This is done by searching for all frames in the pffsample data set where the player's designed role is to pass. This subset of the play is the quarterback's data, which we will use for finding his proximity to the other players. Since we want to find the distance between the quarterback and the

defender for all frames, we use the distance formula to compare each player's x-coordinates and y-coordinates. In Equation 4.5, we see how the distance formula works to find proximities between the defender and the quarterback; we refer to the quarterback as qb and the defender as def in the equation. Once we find the defqbdist vector, we take the minimum, maximum, median, and mean values of the vector as new variables.

$$\text{defqbdist} = \sum_{n=1}^{\text{\# of frames}} \sqrt{(\text{defx}_n - \text{qbx}_n)^2 + (\text{defy}_n - \text{qby}_n)^2} \tag{4.5}$$

We can go one step further with examining the frame-by-frame distances between the defender and the quarterback. Namely, we can derive a "distance closed" variable that captures the percentage of initial distance covered by the defender between these two players. We do this by subtracting each frame's distance from the initial frame distance, where the distance between the two players is usually its highest, and divide this difference by the initial frame distance. In Equation 4.6, we demonstrate how the "distance closed" vector (called defqbclose) is calculated as a proportion of the initial distance between the two players. We take this vector's maximum as a new variable, as this tells us the maximum proportion of distance closed by the defender; the closer this number is to 1, the better the defender performed.

$$\text{defqbclose} = \sum_{n=1}^{\text{\# of frames - 1}} \frac{\text{defqbdist}_1 - \text{defqbdist}_n}{\text{defqbdist}_1} \tag{4.6}$$

We can look at Figure 4.4 for a visual example of how this metric works. The first play frame starts with the defender at the left-most red dot, and the quarterback at the blue dot. As the play progresses, we stop at frame 2 during the middle of the play, where the defender ends up at the middle red dot, while the quarterback remains in his initial position at the blue dot. Clearly, the defender got closer to the quarterback, but we want to determine how much closer this distance is compared to the initial frame. Since the quarterback is at the 36-yard line and the initial defender position is at the 25-yard line, this initial distance is 11 yards. However, the defender crept up to the 30-yard line during the play, which reduced the defender-quarterback distance to 6 yards. Now, the difference in initial and mid-play

distances is 5 yards, so the defender covered 5 of the 11 yards needed to get to the quarterback. As a decimal, this proportion is roughly 0.455, which means that the defender covered 45.5 percent of the initial distance to get to the quarterback.

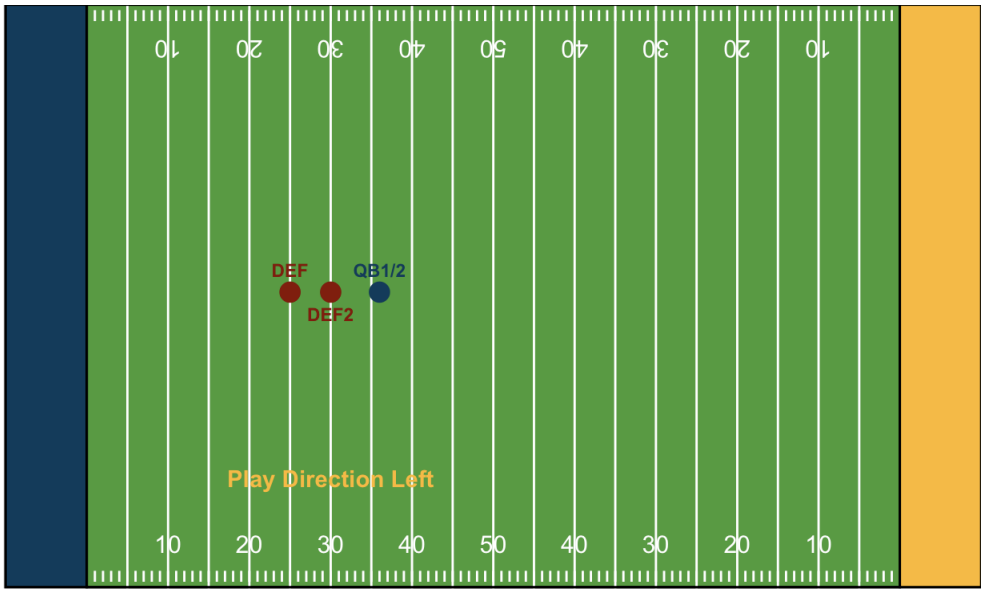

Figure 4.4: Defender Distance Closed to Quarterback

4.8 Creating New Variables Based on Lineman-QB Proximity

Now that we have data sets for play frames pertaining to both the lineman and the quarterback, we can compare their proximities as well. Like the defender-quarterback relationship, a larger distance between players would be a good thing for the lineman's performance. After all, the quarterback tends to perform better without others close to him, so he can have more room to throw and move around to make a play [Cha 18]. In order to get lineman-quarterback distances for all frames during a play, we use the distance formula like in the previous sections. Equation 4.7 shows the notation for this scenario specifically, where the quarterback is defined as qb and the lineman is defined as ol. The olqbdist vector is where all frame-by-frame distances between the two players are stored, and we take the minimum, maximum, median, and mean distances from this vector to create new variables.

$$\text{olqbdist} = \sum_{n=1}^{\#\text{ of frames}} \sqrt{(\text{olx}_n - \text{qbx}_n)^2 + (\text{oly}_n - \text{qby}_n)^2} \tag{4.7}$$

In Figure 4.5, we can observe a visual example of how this metric works. The first play frame starts with the lineman at the line of scrimmage (where the play starts), and the quarterback is at the right-most blue dot. As the play continues, we stop at a frame during the middle of the play, where the lineman backs up to the middle blue dot, while the quarterback remains in his initial position at the right-most blue dot. The lineman approached the quarterback in an effort to protect him by backpedaling, a common lineman technique. Since the quarterback is at the 36-yard line and the initial lineman position is at the 25-yard line, this initial distance is 11 yards. However, the lineman backed up to the 29-yard line during the play, which reduced the defender-quarterback distance to 7 yards. Our goal is to capture these distances for all frames during a play and find relevant benchmarks, such as minimum distance between the players.

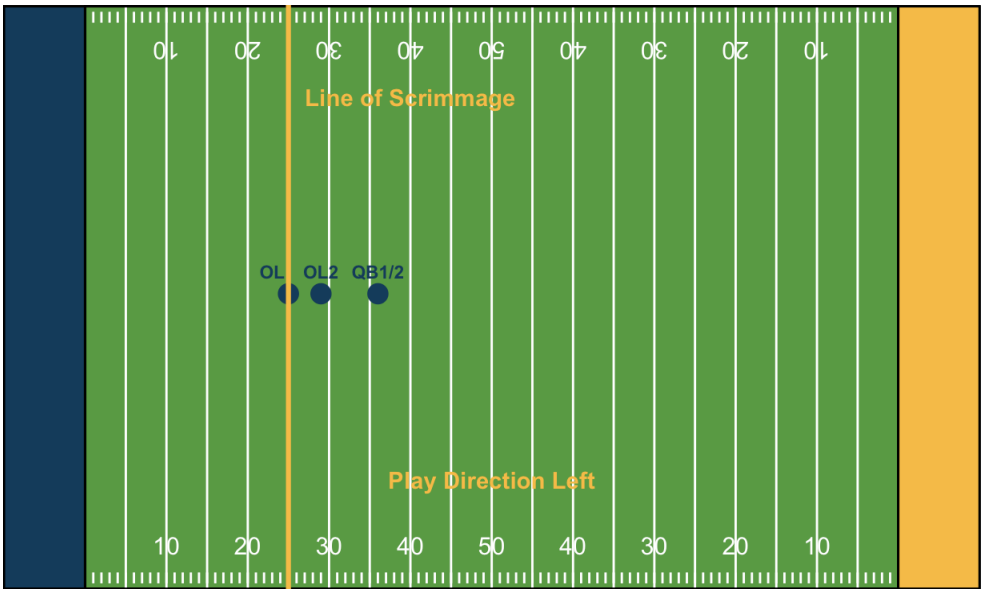

Figure 4.5: Distance between Lineman and Quarterback throughout A Play

4.9 Final Algorithm Steps

Once we iterated through a play with our algorithm, we used the rbind function to add the rows of that play's data to the data from other plays; this was done until all data from our 8,106 distinct plays was acquired. This combined data set, which we call oldata, had 39,579 rows, where each row was a single offensive lineman's data on a particular play. Most of the time, there was data for 5 offensive linemen per play, although there was the occasional play where 4 or 6 offensive linemen were captured, depending on the structure of the play and the defenders involved. We then verified that no variables in oldata were NA values; once we saw there were none present, we created a final variable that kept track of the number of plays each lineman in our data set participated in. Finally, we excluded all players with less than 20 total snaps played, which reduced our number of linemen to 234. From here, we jumped into some exploratory analysis by looking at correlations between variables and distributions of predictors.

CHAPTER 5

Exploratory Analysis

5.1 Correlations between Variables

Now that we finally have the data we need for our offensive linemen, we can take a look at how the data is structured. First, we looked at the correlations between numeric variables. To do this, we grouped the oldata data set by player ID and took the means of the 35 numeric variables for each player across their set of plays. This gives us a set of averages that are not weighted by any non-numeric variables yet. We joined this data set of player averages with the player salaries; however, we changed the player salary variable by dividing it by 1 million, so that the numbers would be more reasonable to work with down the line. With this new data set of numeric predictors and the salary variable, we then ran the cor function to obtain a correlation matrix involving these variables. Out of this matrix, we were most intrigued by the last column, which was a set of variable correlations with salary, so that we could see which predictors were most correlated with the response.

After sorting the magnitudes of these correlations, we took the top 15 most-correlated variables with the response and created a correlation matrix with just those variables. In Figure 5.1, this correlation matrix is displayed visually for us, where we notice that dark blue colors refer to strong positive correlations, while dark gold colors refer to strong negative correlations. The size of the dots also correspond to the strength of the correlations, which is why the diagonal has the largest dots, where variables are obviously correlated to themselves with correlation 1.

One of the largest positive correlations with salary was the number of plays a lineman participated in, with a correlation of 0.213. This relationship makes sense because good

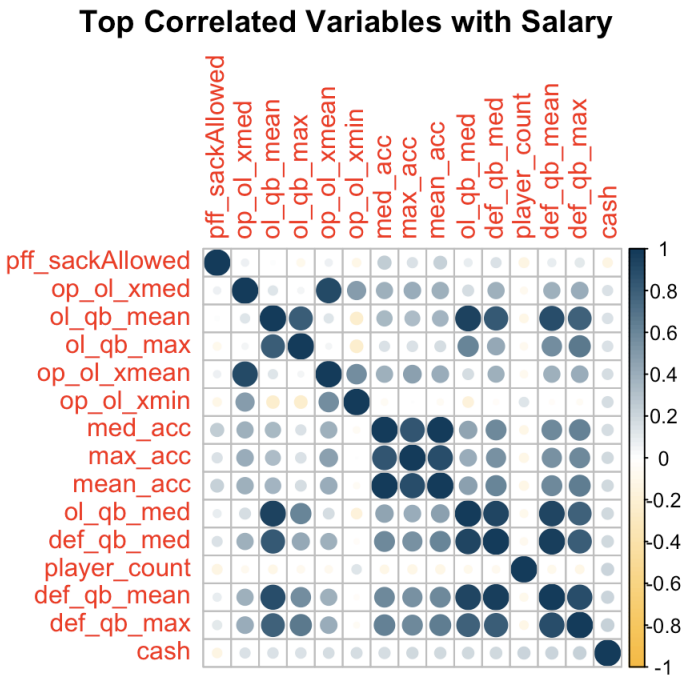

Figure 5.1: Highest Magnitude Correlations between Salary and Predictors

players tend to be on the field often. Also, variables related to the defender-quarterback distance were all positively correlated with the response, with correlations ranging from 0.19 to 0.23 for the 3 variables associated with the defender-quarterback distance. These correlations also track, since a lineman is doing a good job if this proximity is maximized, as the quarterback is less likely to be disrupted.

5.2 Salaries and Snaps

If we look at key metrics for the response variable, salary, we observe that the median salary is 2.08 million dollars, while the mean salary is 4.45 million dollars. This discrepancy can be explained by the right-skewed distribution of lineman salaries in the NFL, where a select few linemen have salaries that dwarf those of their counterparts. Take the salary of Trent Williams, who is considered one of the best linemen in the game for his technique, experience, and athleticism. Williams made a "total cash" salary of 32.3 million dollars in 2021, just under 9 million dollars more than the next highest lineman. In fact, out of all the linemen in

the league, only 3 made at least 20 million dollars in total cash in 2021, which shows that elite play as a lineman can elevate one's salary to enormous heights, even compared to other good linemen!

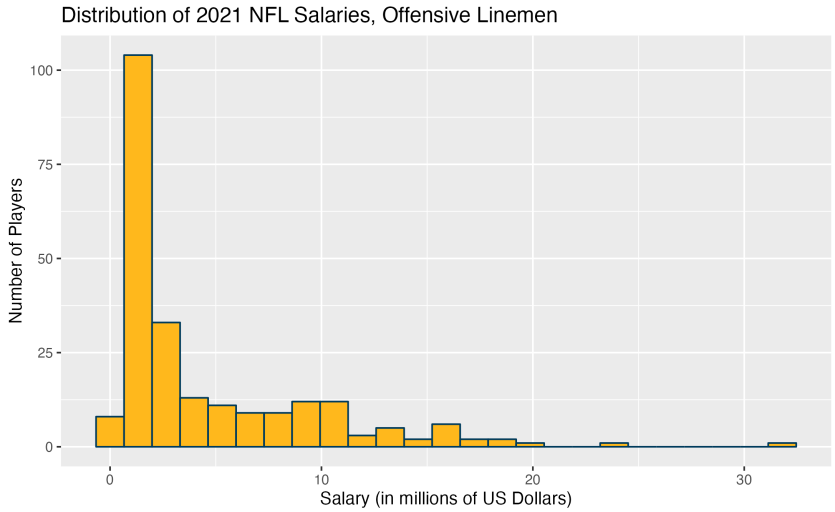

Figure 5.2: Distribution of the response variable, salary

To better understand the distribution of salaries across the league, we created a histogram (Figure 5.2) of linemen salaries in millions of US dollars. The first thing that jumps out about this graph is the spike on the left; roughly half of the linemen have salaries ranging from 500,000 dollars to 2.08 million dollars. These salaries are still massive when compared to the general population, but they fall far short of the highest earners at the position. When looking at the right side of the graph, the top 25 percent of earners made over 6.85 million dollars each in 2021.

On a related note, the variable with one of the strongest positive correlations with salary is the number of snaps a player participated in during the first 8 weeks of the 2021 NFL season. Even though this is the case, we see in Figure 5.3 that the distribution of snaps played is quite different from the right-skewed distribution of salary. The snaps played variable is more evenly distributed than salary, but most players either play a lot of snaps or play very few snaps. The reason for this is there are only 5 linemen that can play on a team at a time, but teams often keep a lot of backup linemen on the bench in case if a starter gets injured. So, unless if an injury happens, the 5 starting linemen will keep playing, and linemen on the

bench won't get on the field very much. Looking at quantiles, roughly 25 percent of linemen took less than 100 snaps, while another 44 percent had 200 or more snaps.

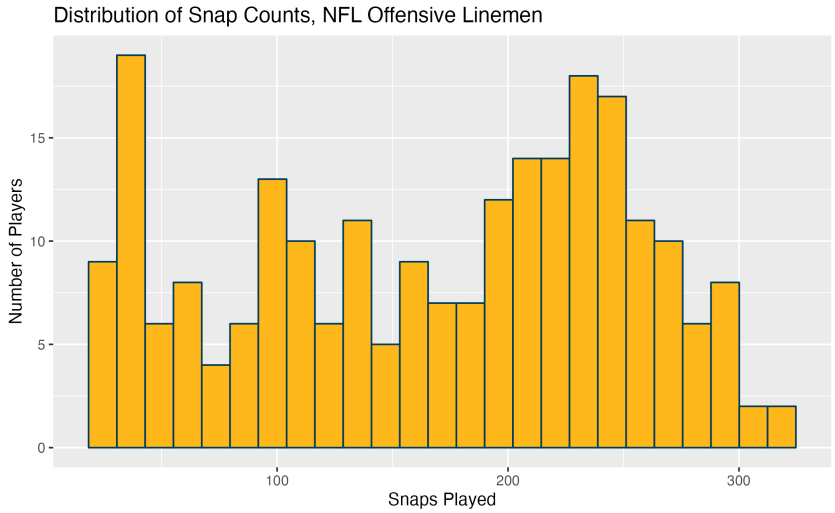

Figure 5.3: Number of snaps played by offensive linemen, Weeks 1-8 of the 2021 NFL Season

5.3 Distance Variables

Another variable with a strong correlation to salary is the median distance between the defender and the quarterback throughout a play. For each lineman, these median distances are compiled across all plays and then averaged, so that each lineman contributes one observation to the histogram in Figure 5.4. Here, there is a curve centered around 5.6 yards; the median is 5.61 yards and the mean is 5.65 yards, so this distribution is, once again, different from the right-skewed distribution of salary. It is worth noting that, while the middle 50 percent of linemen all range from 5.32 yards to 5.99 yards, there are a few high performers to the right with large distances maintained between the defender and quarterback. In particular, there are two players with at least 200 snaps that are above 6.5 yards: Rob Havenstein of the Los Angeles Rams and Orlando Brown of the Kansas City Chiefs. Both were solid starters for their teams that helped their teams make playoff runs. In fact, Brown and the Chiefs went to Super Bowl LV that year.

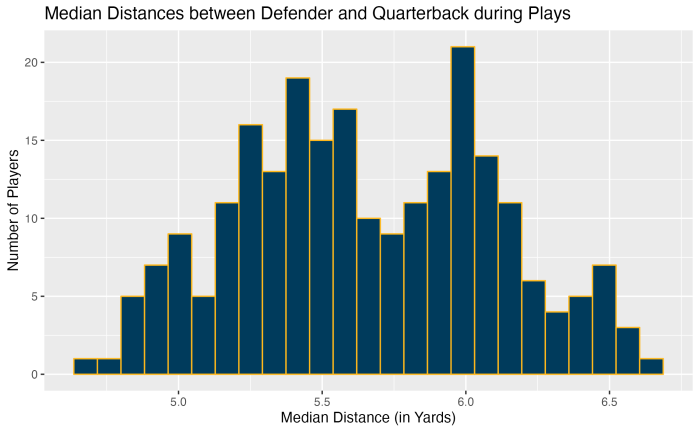

Figure 5.4: Median distances between defender and quarterback, averaged over each lineman's set of plays

Furthermore, acceleration was another top correlated variable, with the maximum acceleration having a decent correlation to salary. Acceleration is an important part of being an offensive lineman, as linemen have to quickly respond to the movements of defenders to protect the quarterback. Good acceleration helps linemen take up space so that defenders don't have a direct path to rush and tackle the quarterback. Looking at the distribution of maximum accelerations averaged across each lineman's play set (Figure 5.5), we see this variable is similar to the median distance metric from Figure 5.4.

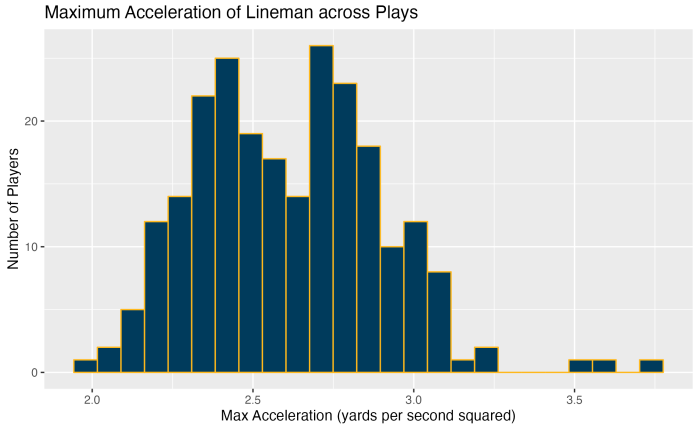

Figure 5.5: Maximum Acceleration of Lineman during Play, averaged over each lineman's set of plays

The approximate bell curve is centered around 2.6 yards per second squared ($\frac{\text{yds}}{\text{s}^2}$), with a median of 2.60 and a mean of 2.61. Once again, the middle 50 percent of players are concentrated in a tight band, ranging from 2.39 to 2.81, while the few outliers have their own stories to tell. For instance, the top 3 linemen in this metric surprisingly had decent snap volumes. Christian Darrisaw (83 snaps) of the Minnesota Vikings, Brandon Shell (164 snaps) of the Seattle Seahawks, and Andrew Whitworth (209 snaps) of the Los Angeles Rams were all starters for their respective teams.

5.4 Potential Weighting Variables

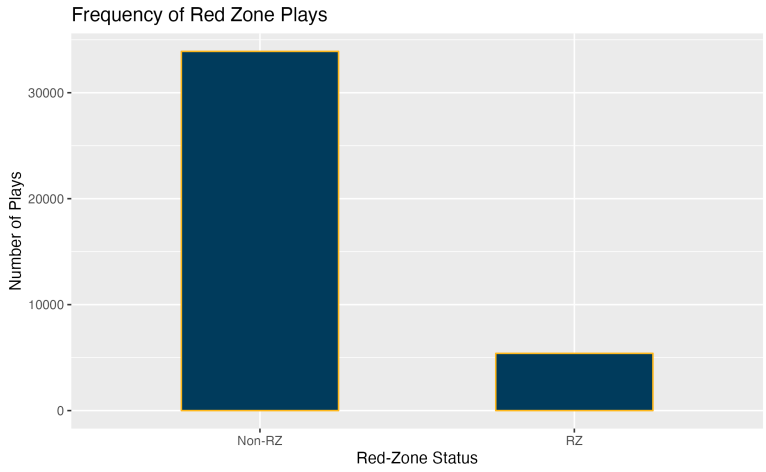

Figure 5.6: Frequency of plays in the red zone in the first 8 weeks of the 2021 NFL Season

Even though we didn't look at non-numeric variables in the correlation analysis, it's important to get a sense of what these variables look like in case we use them to weigh our predictors for modeling purposes. The first categorical variable of interest is the red zone indicator. As we mentioned previously, a play is characterized as a red zone play if the offense is within 20 yards of scoring a touchdown. Most plays are outside of the red zone because for any offensive possession, only 20 percent of the field is considered the red zone. However, these red zone plays are the most important plays of the game due to the heightened potential to score. As we turn our attention to Figure 5.6, we see a bar chart comparing the frequencies of red zone plays compared to plays outside of the red zone. The

discrepancy between these play types is expected, as only 13.8 percent of plays in our data set took place in the red zone (labeled as RZ in the graph).

Another important categorical variable is the one-score game indicator. From before, a play is characterized as happening during a one-score game if the two teams playing the game are within 7 points of one another. In most circles, this is considered a close and competitive game, and plays under these circumstances are more important because greater swings in win probability occur, which make the stakes higher. We would expect a slightly majority of plays to be during one-score games, simply because games start out with a 0-0 score, so all games inherently start out as one-score games. As we can see in Figure 5.7, we see a bar chart comparing the frequencies of one-score plays compared to plays that happen when a team is up by 2 possessions or more. Once again, our hypothesis about the difference between play types is correct, as 62.2 percent of plays in our data set took place in one-score games (labeled as 1 score in the graph).

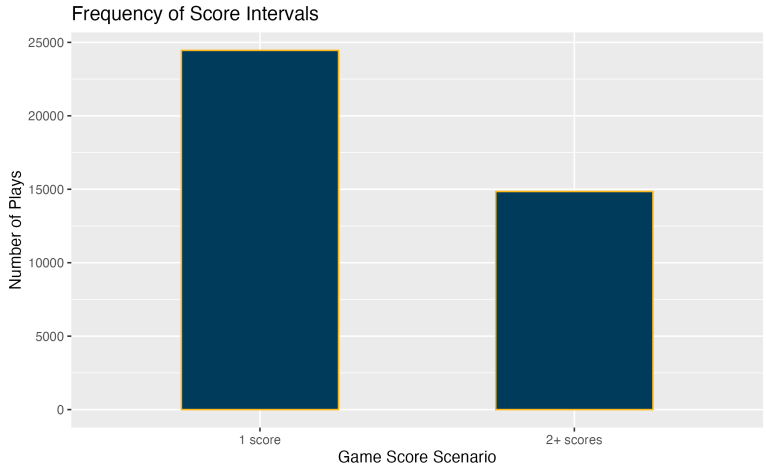

Figure 5.7: Frequency of plays that happen in One-Score Games

CHAPTER 6

Models

6.1 Model Preparations

Before we dove headfirst into the modeling process, there were some logistical aspects that needed to be completed. The first action we took was setting a seed (904138) for reproducibility; this way, other people can run this code with the same results. Then, we used the createDataPartition function from the caret package in R to split observations into training and test sets in order to train the model; we determined that the training set would get 75 percent of observations, while the test set would get 25 percent of observations. Then, once we performed this split, we needed to standardize the numeric predictors because the magnitudes of some predictors are on different scales than others, so this would mitigate any imbalances in scale that certain predictors would have over others; also, we are able to reduce multicollinearity in our model [Fro 18]. To standardize the predictors, we used the scale function on our numeric variables in both the training and test sets. With these steps complete, we were able to proceed to modeling.

6.2 Unweighted Stepwise Regression Model

The first model was an unweighted stepwise regression model. By unweighted, we mean that we do not account for any categorical variables, like red zone status, the half of the game when the play happened, or the score differential of the game during the play. We are only considering variables that took place within the play, which mostly involves player tracking metrics. This stepwise regression model was derived by taking the full set of predictors and

using the step function in order to find the best subset of variables to model our response variable, salary. The full model was not a consideration, as there were too many variables with high correlations to one another, so using a stepwise regression model helped reduce multicollinearity.

With regard to the stepwise model, the initial iteration had 8 variables, but 3 of them were highly correlated since they all pertained to the distance between the lineman and the quarterback. To reduce the variance inflation factors of the variables in the model, I only kept the median distance between the lineman and the quarterback, as it had the highest positive correlation with the response out of the 3 highly-correlated variables.

Variable	Estimate	Std. Error	P-value
Intercept	4.53	0.34	2e-16
beatenByDef	-1.17	0.38	0.002
meanacc	3.98	0.89	1.37e-05
olmeandist	-3.33	0.85	0.0001
defqbmean	1.29	1.10	0.24
olqbmed	-0.1616	1.00	0.87
playercount	1.24	0.35	0.0005

Table 6.1: Model summary for unweighted stepwise model

Table 6.1 displays the model output for the stepwise regression, where the variables kept were the rate that the lineman was beaten by the defender (beatenByDef), the average acceleration of the lineman (meanacc), the mean distance the lineman traveled (olmeandist), the mean distance between the defender and the quarterback (defqbmean), the median distance between the lineman and the quarterback (olqbmed), and the number of snaps the lineman played (playercount). While these variables have strong correlations with the response, it seems as if there are some transformations necessary due to the patterns in the model's residual plot (Figure 6.1).

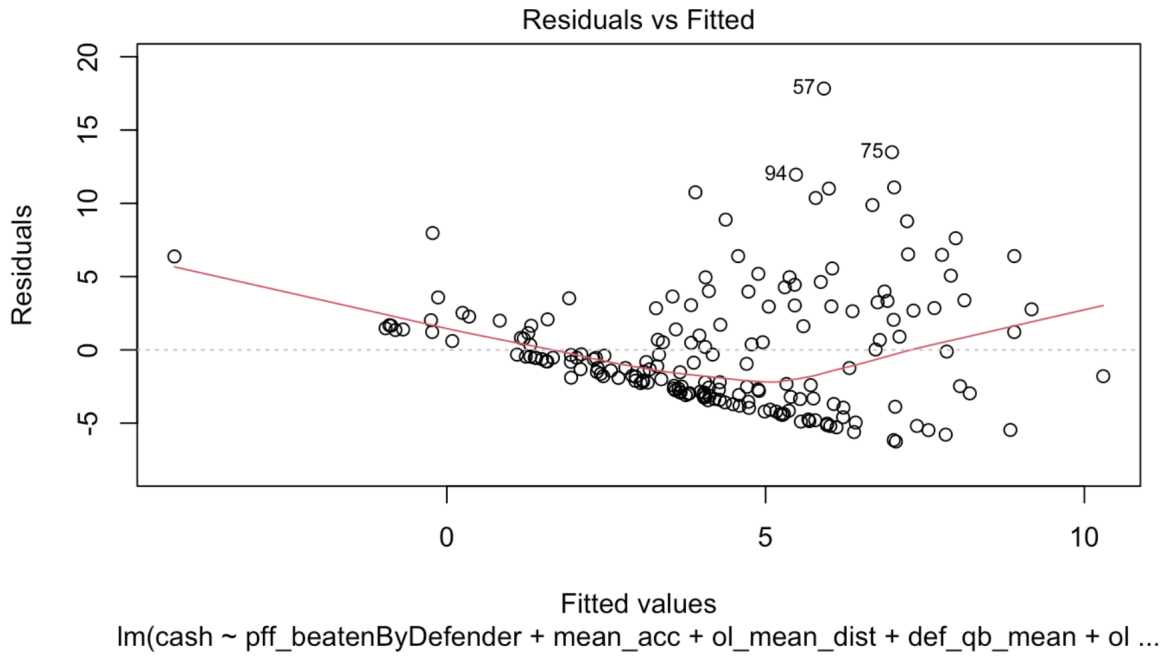

Figure 6.1: Diagnostic residual plot for unweighted stepwise regression model

Specifically, the residuals tend to increase in magnitude as the fitted values increase, which is a classic sign of heteroskedasticity. So, this model's results cannot be trusted, which necessitates alterations to the model, even with an adjusted R-squared of 0.2443. Next, we will explore weighing our numeric predictors with some categorical variables to see if the residual plot improves.

6.3 Weighted Stepwise Regression

The next model also involves running a stepwise regression model, but the initial variables that we subset upon are different; this is because we added weights to each predictor. Specifically, we grouped plays into several buckets based on 2 categorical variables: red zone status and the score differential of the game during the play. We considered a third variable, the half of the game the play occurred in, but there was a good balance of plays from both halves, so we didn't find a need to weigh upon the half variable. In Table 6.2, we have the frequencies of various play types. The labels NonRZ and RZ refer to whether a play is in the red zone or not, while the labels 1score and 2score refer to whether a play happened during a one-score

or a two-score game.

Red Zone	Score	Frequency
NonRZ	1score	21107
NonRZ	2score	12796
RZ	1score	3349
RZ	2score	2053

Table 6.2: Frequency Table of Play Types

In order to come up with weights for these categories, we looked at the corresponding frequencies for each. Since only 14 percent of plays took place in the red zone, we assigned 75 percent of the weight to red zone plays to make up for their scarcity and to highlight their importance in games, while all other plays received 25 percent of the weight. Similarly, plays that took place in one-score games received 60 percent of the weight, while others received 40 percent. The weights are not as strong for this variable because there was only a slight difference between the amount of one-score and two-score games, and plays that occur in the red zone are generally more important than plays that happen in one-score games.

Variable	Estimate	Std. Error	P-value
Intercept	4.53	0.34	2e-16
beatenByDef	-1.15	0.36	0.002
meanacc	4.07	0.84	3.14e-06
olmeandist	-3.38	0.82	5.33e-05
defqbmean	0.80	0.98	0.415
olqbmed	0.30	0.91	0.74
playercount	1.30	0.35	0.0003

Table 6.3: Model summary for weighted stepwise model

After re-calculating the predictors with these weights, we followed the same stepwise regression process as the first model to get the results found in Table 6.3, which contained

the same variables as before. There are some slight differences with variable slopes and significances compared to the unweighted model, yet the weighted model had a similar adjusted R-squared at 0.2431. The diagnostic plot of the residuals (Figure 6.2) is also very similar to the unweighted model, so this indicates that the weights we added did not improve upon the model structure. The next step is to try transforming variables from this weighted model.

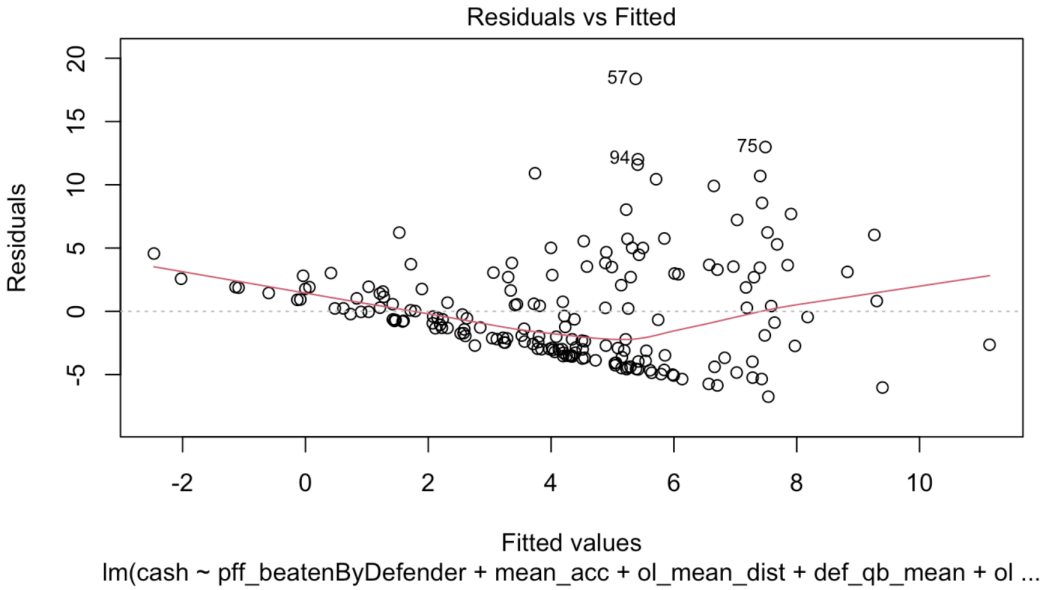

Figure 6.2: Diagnostic residual plot for weighted stepwise regression model

6.4 Transformed Model

In order to improve model diagnostics, we have to consider variable transformations to account for non-normal distributions of these variables. A great way to identify possible changes is to use box-cox transformations, which determine the best possible transformations for each variable in the context of the model. Since regression often has the assumption that the variables involved are normally distributed, when you input variables that don't follow these rules, then the results can become unreliable. The great thing about box-cox transformations is that they transform variables so that they can better approximate normal distributions, which will help stabilize results in regression models. Below we have the equations of box-cox

transformations, where $y(\lambda)$ represents the transformed variable, y is the original variable, and λ represents the transformation applied to y [Plu 18]. The value of λ is calculated by maximizing the log-likelihood function, and based on this value, several transformations can be applied.

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log(y), & \text{if } \lambda = 0 \end{cases}$$

Essentially, if λ is equal to 0, then the log transform is applied to the variable, but if λ is non-zero, then a power transformation is applied, with the exponent being λ. An interesting case is when λ is equal to 1, the variable is shifted down, but no other transformation is applied, which indicates that the variable does not need to be transformed in the first place.

Variable	Lambda
beatenByDef	-0.30
meanacc	-0.26
olmeandist	-0.30
defqbmean	-0.26
olqbmed	-0.26
playercount	-0.26
cash	-0.30

Table 6.4: Lambda Values from Box-Cox Transformations

In the case of the variables in our model, most of them have skewed distributions, so it would make sense that most of them would get the log transform. To check the optimal lambda value for each variable, we ran the boxcox function from the MASS package in R. Not surprisingly, the results of the function were what we expected, with most variables having values close to 0. In Table 6.4, we observe that lambda values obtained were either -0.26 or -0.30, with mean lineman acceleration (meanacc), mean distance between the defender and quarterback (defqbmean), player snap count (playercount), and median distance between

lineman and quarterback (olqbmed) being closest to 0.

Even though all lambda values are close to 0, we don't want to simply log transform all variables, as this would cause the model to become quite volatile in nature. So, we want to pick a subset of the variables to do log transforms on. The first idea was to transform the variables with the smallest magnitude lambda values: meanacc, defqbmean, playercount, and olqbmed. However, most of these predictors have values that are too small to be log transformed, as this would result in NA values for the model. Hence, we took a simplified approach by just transforming the response variable, salary (called cash in Table 6.4). In doing so, we were able to improve the residual plot to have a more appropriate distribution (Figure 6.3).

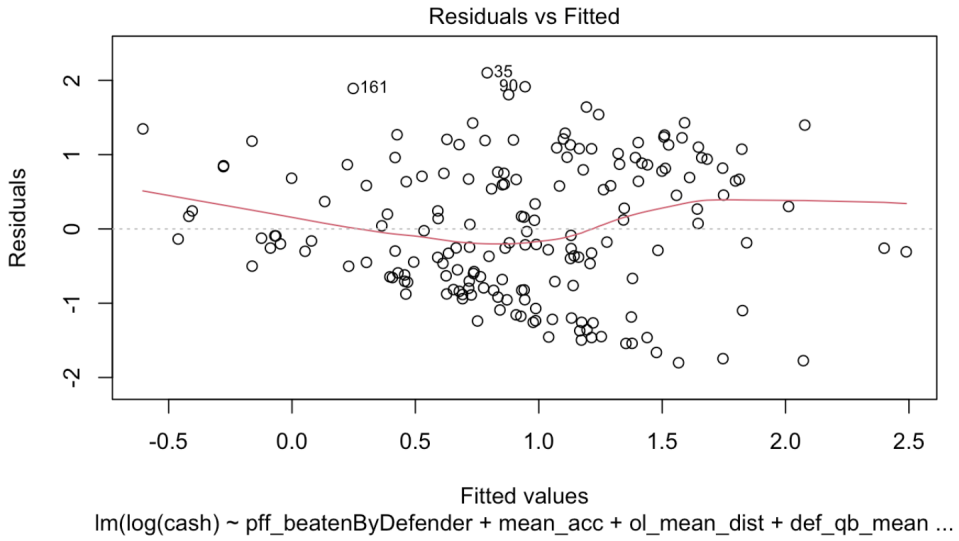

Figure 6.3: Diagnostic residual plot for transformed model

As seen below (Table 6.5), we have the model summary for this transformed model, which has an adjusted R-squared of 0.2483 after training, which beats out the results we obtained from the previous two models.

In addition, we tried the model on the test set, and after exponentiating our predictions to reverse the log transform, we achieved a root mean square error (RMSE) of 4.7, which is respectable considering the small amount of data we had to work with. In looking at the observations with the largest errors, we saw that Ronnie Stanley, a high-paid tackle for the

Variable	Estimate	Std. Error	P-value
Intercept	0.93	0.07	2e-16
beatenByDef	-0.25	0.08	0.001
meanacc	0.67	0.17	0.0002
olmeandist	-0.59	0.17	0.0005
defqbmean	0.35	0.20	0.09
olqbmed	-0.08	0.19	0.65
playercount	0.32	0.07	1.36e-05

Table 6.5: Model summary for transformed model

Baltimore Ravens, yielded the largest error of all: his predicted salary of just 1.41 million dollars was dwarfed by his actual 23.75 million dollar salary, a whopping difference of 22.34 million dollars. Thankfully, there is a practical explanation for this discrepancy, as Stanley was dealing with a severe leg injury that drastically limited his playing time in 2021. Even when he was on the field, his performance was not up to his standard. He was considered one of the best linemen in the league just a couple years prior, which explains the large salary he got. Unfortunately for Stanley and the Ravens, the injury compromised both parties, and the Ravens barely got any value out of Stanley's contract for 2021. On the other hand, the largest positive residual was Orlando Brown, a young lineman for the Chiefs. Since Brown was so young, he was on a relatively cheap rookie salary of 3.384 million dollars, but his play helped lead the Chiefs to the Super Bowl. As a result, his predicted salary was 8.09 million dollars, a healthy difference of 4.7 million bucks.

6.5 Random Forest

Finally, we tried the random forest model to see if a different method than stepwise regression may be more effective in predicting NFL salaries. The idea behind random forests is to use a group of predictors, or an ensemble, to make predictions. Specifically, a random forest

chooses a subset of predictors and makes many decision trees from these predictor sets [Joh 23]. Then, by looking at all of these decision trees and averaging them out, we can obtain a solid framework to predict future results based on benchmark values of certain predictors. Since random forests use decision trees, there is no need to standardize the predictors, so we used the weighted but unstandardized versions of all predictors as inputs for the random forest. The only variables excluded from the random forest were nflId and displayName, as these don't have anything to do with a lineman's performance from play to play.

Number of Trees	Mean of Squared Residuals
50	26.25
75	25.03
100	24.99
125	25.14
150	25.42

Table 6.6: Random Forest Training Results

In order to compare different random forest types, we ran 5 different variations based on the number of trees generated. Namely, we chose 50, 75, 100, 125, and 150 trees, as these values are all centered around the default tree value of 100. We didn't choose any numbers that were particularly high because the data which we are working with is not especially large, so the suspicion is that a large number of trees may not perform much better than a smaller number of trees. Also, we left the number of variables tried at each split alone, as the computer chose 11 variables for all iterations of the random forest. After we ran these random forests on the training data, we compared the mean of squared residuals across the 5 iterations. In Table 6.6, we can see how the models did on the training data. Even though we would think a higher number of trees would lead to better results due to the increased rigor of the model, this was not the case in training. In fact, there was a sweet spot at 100 trees, where the minimum of the mean of the squared residuals resided. However, the models with 75 trees, 100 trees, and 125 trees had extremely close metrics, so there wasn't a clear

winner. Next, we will mention the test results.

Moving on to the test results, we ran a loop that iterated over each model and extracted its RMSE with respect to the test set. As we can see in Table 6.7, we were somewhat surprised to see the 75-tree model with the best RMSE.

Number of Trees	RMSE
50	4.42
75	4.36
100	4.39
125	4.43
150	4.45

Table 6.7: Random Forest Test RMSE Values

While this model performed well in the training stage, it was not first place, but it overtook the 100-tree model for the test data. In addition, the trees with 50, 125, and 150 trees didn't rival either of the other two trees, as all three were above 4.4. This shows that you don't necessarily need as many trees for smaller data sets, as the complexity of larger random forests might even hurt test accuracy. However, all of these models still outperformed the transformed regression model.

Since the 75-tree model performed the best out of the various random forests, we wanted to take a closer look at what variables had the biggest influence in this model. In Figure 6.4, we have the variable importance plot for this model, where the most important variables have dots farther to the right. We notice that several of the variables that were kept in the stepwise regression models also have high importances in our best random forest model. Specifically, there were several variables that had significant impacts on both model types, including the rate that the lineman was beaten by the defender (pffbeatenByDefender), the number of snaps the lineman took (playercount), and mean acceleration by the lineman throughout his plays (meanacc). Interestingly, the max acceleration (maxacc) by the lineman averaged across each of his plays was the most important variable in the random forest model, but it was

not kept in stepwise regression since meanacc was kept instead. Moreover, there were three variables from the stepwise regression models that were not considered as important in the random forest. Namely, they were the average distance traveled by the lineman (olmeandist), the average distance between the defender and quarterback (defqbmean), and the median distance between lineman and quarterback (olqbmed). All three of these variables are distance metrics, so the random forest did not consider these kinds of variables as important as the stepwise regression models did.

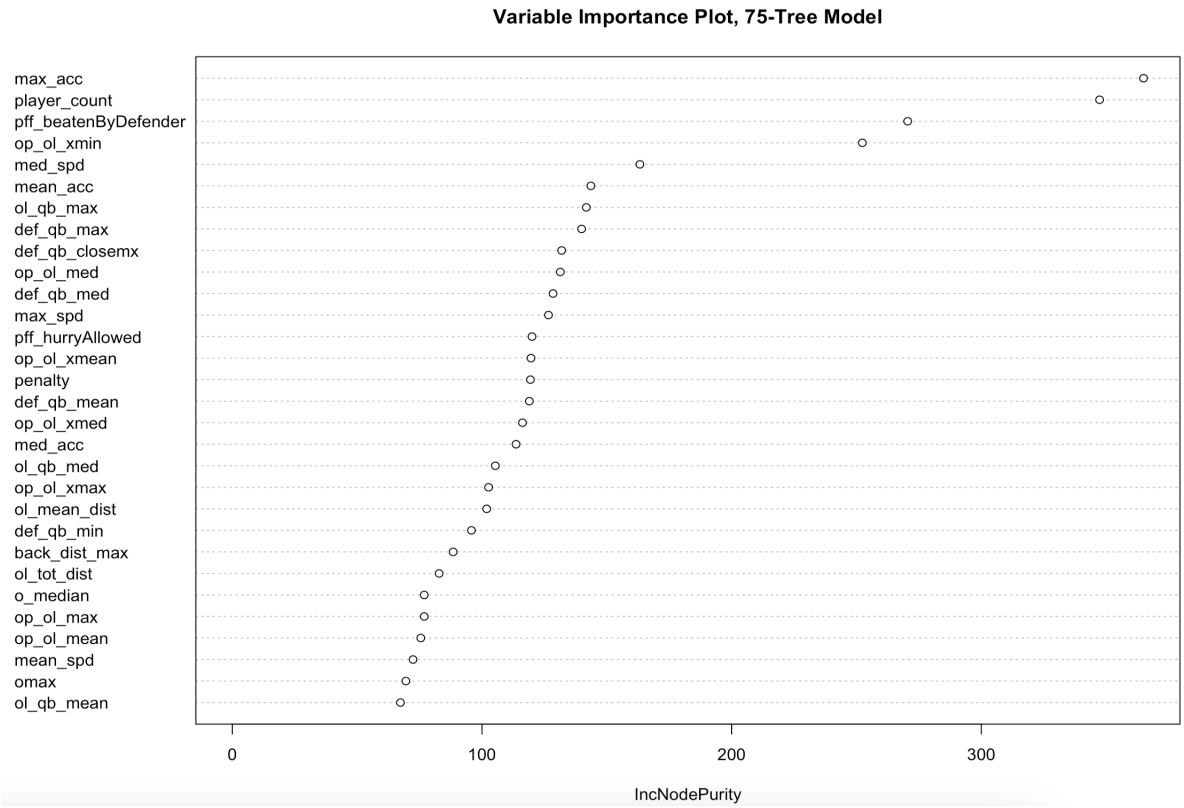

Figure 6.4: Variable Important Plot for 75-Tree Model

CHAPTER 7

Conclusion

7.1 Main Takeaways

At the end of this analysis, we realized why there hasn't been much analysis done on offensive linemen. Unlike the quintessential quarterback or the wild wide receiver that we can use well-known scoring and yardage metrics to objectively measure their output, offensive linemen don't have many well-known metrics to evaluate their game-by-game performances. If you were to look at a box score of any NFL game, it's likely that unless a lineman recovered a fumble or somehow caught a reception, there will be no sign of the big guys on the score sheet. In fact, across the four major professional sports in the United States, the only position where this is the case is an offensive lineman. All they get credit for is their presence at games, despite their efforts in the trenches. However, with the recent rise of analytics and tracking technology, there are some new ways that we can assign credit to linemen other than games played or salary.

The goal of this paper was to bring linemen out of obscurity with a hard look at play-by-play tracking data, with an emphasis on important situations like red zone plays. Even though we attempted various feature engineering techniques and model types, the results were middling. Granted, coming into this project, we recognized that this would be a difficult endeavor, with limited tracking data from the first 8 weeks of the 2021 NFL season. However, there were some encouraging results.

The weighting and transforms we carried out resulted in model validity and a slightly increased adjusted R-squared (0.2483) on the training data in comparison to previous regression models; this model's RMSE was 4.7. Then, the random forest models improved

upon the RMSE of the transformed regression model. The 5 random forest models we tested each had a different number of trees, ranging from 50 to 150 with intervals of 25. Their RMSE values, in increasing order of trees, were 4.42, 4.36, 4.39, 4.43, and 4.45. So, the random forest with 75 trees had the best RMSE, and we discovered that one doesn't need a random forest with an excessive amount of trees to perform well on a small data set.

While the transformed stepwise regression model gave us an RMSE of 4.7, the random forest models all had values below 4.5. Even though this was the case, there were similarities between the stepwise regression model and the random forest with 75 trees. The variable importance plot for the 75-tree model showed that 3 of the 6 variables from the stepwise regression models were also in the top 6 of variable importance for the random forest. These variables were the rate that the lineman was beaten by the defender (pffbeatenByDefender), the number of snaps the lineman took (playercount), and mean acceleration by the lineman throughout his plays (meanacc).

Finally, when we looked at some large errors in context from our models, we were able to understand why Ronnie Stanley had such a low predicted salary (due to injury) and why Orlando Brown outperformed his contract (a solid year on a cheap rookie contract). So, the modeling may not be entirely to blame, especially once we understand why some predictions may be off for certain players.

7.2 Potential Improvements

Due to the limited nature of offensive line data, it has proven difficult to devise an effective and consistent method of measuring success at the position. But with more types of data on the way, it may not be so hard in the near future. With respect to this project, I was fortunate to get such detailed data, but it was over a fairly short interval. Ideally, we would like to have several seasons' worth of this type of tracking data, but this type of information is relatively new, so we'll have to wait a few years to have a great repository of tracking data. With enough data, I would have liked to carry out a position-by-position analysis along the offensive line that would compare linemen at the same position to one another. So, instead

of looking at the position in general, we could compare left tackles, left guards, centers, right guards, and right tackles. Plus, this additional data could allow us to better understand how much the success of certain positions are correlated with others. Although football has been known to lag behind other sports in the analytics race, things are speeding up in the NFL with regard to the quality of data collection and evaluation across the league. This is only the beginning for what insights could be captured from future data collection.

CHAPTER 8

Appendix

```
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)             4.533759   0.338536  13.392   <2e-16 ***
pff_beatenByDefender   -1.054354   0.471737  -2.235   0.0270 *
pff_hitAllowed          0.003026   0.440368   0.007   0.9945
pff_hurryAllowed       -0.378417   0.492613  -0.768   0.4437
pff_sackAllowed        -0.058116   0.477793  -0.122   0.9034
penalty                 0.091739   0.400249   0.229   0.8190
o_median                0.649811   0.862752   0.753   0.4526
o_mean                 -0.477212   1.317429  -0.362   0.7177
omax                    0.580920   1.048344   0.554   0.5804
max_acc                -2.881011   1.849495  -1.558   0.1215
med_acc                -4.119834   4.814766  -0.856   0.3936
mean_acc               11.269827   6.593853   1.709   0.0896 .
max_spd                 1.984197   2.566577   0.773   0.4408
med_spd                -4.796997   8.701151  -0.551   0.5823
mean_spd               -1.613907   7.986959  -0.202   0.8402
back_dist_max           1.849338   1.323816   1.397   0.1646
ol_tot_dist             0.673411   1.811035   0.372   0.7106
ol_med_dist             9.324686   9.537657   0.978   0.3299
ol_mean_dist          -11.525925   8.151273  -1.414   0.1595
op_ol_max              -0.627330   2.288609  -0.274   0.7844
op_ol_med              -1.484139   2.019480  -0.735   0.4636
op_ol_mean              2.238299   3.518005   0.636   0.5256
op_ol_xmax             -2.629466   2.937546  -0.895   0.3722
op_ol_xmin             -0.283595   1.358648  -0.209   0.8350
op_ol_xmed             -0.144436   2.192654  -0.066   0.9476
op_ol_xmean             2.614747   4.909764   0.533   0.5952
def_qb_min             -1.248492   2.630070  -0.475   0.6357
def_qb_max             -2.662175   4.200638  -0.634   0.5273
def_qb_med             -4.951566   5.359070  -0.924   0.3571
def_qb_mean             9.520566  10.329422   0.922   0.3583
def_qb_closemx         -0.231448   1.305177  -0.177   0.8595
ol_qb_min               2.855114   2.337027   1.222   0.2239
ol_qb_max               7.336853   3.690960   1.988   0.0488 *
ol_qb_med              11.015497   4.346647   2.534   0.0124 *
ol_qb_mean            -19.254494   8.409500  -2.290   0.0235 *
player_count            0.708454   0.410108   1.727   0.0863 .
```

Figure 8.1: Full Regression Model Output

Game data

- **`gameId`** : Game identifier, unique (numeric)
- `season` : Season of game
- `week` : Week of game
- `gameDate` : Game Date (time, mm/dd/yyyy)
- `gameTimeEastern` : Start time of game (time, HH:MM:SS, EST)
- `homeTeamAbbr` : Home team three-letter code (text)
- `visitorTeamAbbr` : Visiting team three-letter code (text)

Play data

- **`gameId`** : Game identifier, unique (numeric)
- **`playId`** : Play identifier, not unique across games (numeric)
- `playDescription` : Description of play (text)
- `quarter` : Game quarter (numeric)
- `down` : Down (numeric)
- `yardsToGo` : Distance needed for a first down (numeric)
- `possessionTeam` : Team abbr of team on offense with possession of ball (text)
- `defensiveTeam` : Team abbr of team on defense (text)
- `yardlineSide` : 3-letter team code corresponding to line-of-scrimmage (text)
- `yardlineNumber` : Yard line at line-of-scrimmage (numeric)
- `gameClock` : Time on clock of play (MM:SS)
- `preSnapHomeScore` : Home score prior to the play (numeric)
- `preSnapVisitorScore` : Visiting team score prior to the play (numeric)
- `passResult` : Dropback outcome of the play (`C` : Complete pass, `I` : Incomplete pass, `S` : Quarterback sack, `IN` : Intercepted pass, `R` : Scramble, text)
- `penaltyYards` : yards gained by offense by penalty (numeric)

Figure 8.2: Variable Definitions 1

- `prePenaltyPlayResult`: Net yards gained by the offense, before penalty yardage (numeric)
- `playResult`: Net yards gained by the offense, including penalty yardage (numeric)
- `foulName[i]`: Name of the i-th penalty committed during the play. i ranges between 1 and 3 (text)
- `foulNFLId[i]`: `nflId` of the player who comitted the i-th penalty during the play. i ranges between 1 and 3 (numeric)
- `absoluteYardlineNumber`: Distance from end zone for possession team (numeric)
- `offenseFormation`: Formation used by possession team (text)
- `personnelO`: Personnel used by offensive team (text)
- `defendersInTheBox`: Number of defenders in close proximity to line-of-scrimmage (numeric)
- `personnelD`: Personnel used by defensive team (text)
- `dropbackType`: Dropback categorization of quarterback (text)
- `pff_playAction`: indicator for whether offense executes play action fake on the play. Variable provided by PFF (binary)
- `pff_passCoverage`: Coverage scheme of defense. Variable provided by PFF (text)
  - Possible values:
    - `Cover-0`: A Man to Man coverage across the board with no deep defenders. This coverage is typically accompanied with a blitz
    - `Cover-1`: When a defense plays any form of Man defense across the board with a Defensive Player as a Single High Man concept
    - `Cover-2`: A two deep safety concept where any zone principle is applied
    - `2-Man`: A two deep safety concept where a man principle is applied
    - `Cover-3`: Any 3 Deep, 4 Under concept
    - `Quarters`: A Quarters concept on both halves of the field. In general it will be a 4 Deep, 3 Under concept where the corners are on #1, safeties on #2, and backside safety rotation dependent on formation
    - `Cover-6`: A Quarters Concept on half the field and a 2 Deep concept on the other half
    - `Bracket`: Recorded in the field and up to the 12 yard line in the red zone – when two offensive players have an in and out bracket by two defenders
    - `Goal Line`: Calls where a Goal Line defense is used.
    - `Red Zone`: Calls that are typically specific to the Red Zone and do not occur in the field often
    - `Prevent`: Special end of half or end of game situations where a Prevent defense is utilized
    - `Miscellaneous`: Coverage concepts that we feel do not comfortably fit into any of our coverage categories

Figure 8.3: Variable Definitions 2

- `pff_passCoverageType` : Whether defense's coverage type was man, zone or other. Variable provided by PFF (text)

Player data

- **`nflId`** : Player identification number, unique across players (numeric)
- `height` : Player height (text)
- `weight` : Player weight (numeric)
- `birthDate` : Date of birth (YYYY-MM-DD)
- `collegeName` : Player college (text)
- `officialPosition` : Official player position (text)
- `displayName` : Player name (text)

PFF Scouting data

- **`gameId`** : Game identifier, unique (numeric)
- **`playId`** : Play identifier, not unique across games (numeric)
- **`nflId`** : Player identification number, unique across players (numeric)
- `pff_role` : The player's role on this play (text)
  - Possible values:
    - `Coverage` : Defensive player. Player whose initial goal is to play man or zone coverage
    - `Pass` : Offensive player. Player identified as the passer
    - `Pass block` : Offensive player. Anyone fully blocking a defender from the QB, or anyone in a clear pass block stance
    - `Pass route` : Offensive player. Any player not identified as a Pass Blocker or Passer
    - `Pass rush` : Defensive player. Any player whose initial intent is to rush the passer
- `pff_positionLinedUp` : Position that the player was aligned at the snap of the ball on this play (text)
- `pff_hit` : If player is a defensive player, indicator for whether they are credited with recording a hit on this play (binary)
- `pff_hurry` : If player is a defensive player, indicator for whether they are credited with recording a hurry on this play (binary)
- `pff_sack` : If player is a defensive player, indicator for whether they are credited with recording a sack on this play (binary)
- `pff_beatenByDefender` : If player is a blocking offensive player, indicator for whether they are by a defender but was not charged for yielding a hit, hurry or sack (binary)

Figure 8.4: Variable Definitions 3

- `pff_hitAllowed`: If player is a blocking offensive player, indicator for whether they are responsible for a hit on the QB (binary)
- `pff_hurryAllowed`: If player is a blocking offensive player, indicator for whether they are responsible for a hurry on the QB (binary)
- `pff_sackAllowed`: If player is a blocking offensive player, indicator for whether they are responsible for a sack on the QB (binary)
- `pff_nflIdBlockedPlayer`: If player is a blocking offensive player, the `nflId` of the first defender the offensive player blocked (numeric)
- `pff_blockType`: If player is a blocking offensive player, the type of block that the offensive player is executing on the defender (text)
  - Possible values:
    - `BH`: Backfield Help - A block from a player aligned in the backfield on which the blocker merely helps on a block rather than fully engaging his assignment. Usually seen when a blocker is clearing up a block or picking up a defender when he has broken through or been missed by another blocker
    - `CH`: Chip Block - This is only to be used for players who chip a pass rusher when they release for their route
    - `CL`: Second Level – A block made at the second level, this must be at least two yards across the line of scrimmage
    - `NB`: No Block - If a blocker executes no block on a play but simply runs his path or takes his pass set then we will note him with one all blocking line with this block type
    - `PA`: Play Action Pass Protection - A blocker pass protecting inline on a play action pass selling the play action by stepping in to show a run block before converting to pass protect
    - `PP`: Pass Protection - A standard pass protection block from an inline blocker
    - `PR`: Pocket Roll Block - This block type will be used any time the offense is executing a “rolling pocket” by which the entire offensive line moves with the QB’s rollout to stay in front of him but without ever taking a “conventional” pass set. There will be flexibility here to record the PR – Pocket Roll Block type in the same way as PA & RP block types in that individual matchups & responsibilities won’t always be obvious or necessary, so PR block types can be recorded by multiple blockers on an individual defender on the same play
    - `PT`: Post Block - A post block by an offensive player in pass protection to control a defender for another blocker while clearly demonstrating that he is not, at least initially, trying to fully engage with the block
    - `PU`: Backfield Pickup - A pass protection pick-up by a player aligned in the backfield
    - `SR`: Set & Release - A blocker who sets to pass protect a defender before releasing. This block will cover both players releasing from a set to block for a screen as well as “hold ups” by tight ends before they leak into the flat
    - `SW`: Switch Block - A blocker who passes off (or attempts to pass off) a defender. Most often used on stunts but can also be used for pass offs when pass rushers are slanting across the pocket or interior defenders are working to the edge to replace a dropping edge rusher, with an interior offensive lineman passing them out rather than staying with them
    - `UP`: Pull Pass Protection - A blocker pulling in pass protection from an inline alignment to block a defender in pass protection

Figure 8.5: Variable Definitions 4

- `pff_backFieldBlock` :If player is a blocking offensive player, indicator for whether block occured in offensive backfield.

Tracking data

Files `week[week].csv` contains player tracking data from week `[week]`.

- **`gameId`** : Game identifier, unique (numeric)
- **`playId`** : Play identifier, not unique across games (numeric)
- **`nflId`** : Player identification number, unique across players. When value is NA, row corresponds to ball. (numeric)
- `frameId` : Frame identifier for each play, starting at 1 (numeric)
- `time` : Time stamp of play (time, yyyy-mm-dd, hh:mm:ss)
- `jerseyNumber` : Jersey number of player (numeric)
- `club` : Team abbrevation of corresponding player (text)
- `playDirection` : Direction that the offense is moving (left or right)
- `x` : Player position along the long axis of the field, 0 - 120 yards. See Figure 1 below. (numeric)
- `y` : Player position along the short axis of the field, 0 - 53.3 yards. See Figure 1 below. (numeric)
- `s` : Speed in yards/second (numeric)
- `a` : Acceleration in yards/second^2 (numeric)
- `dis` : Distance traveled from prior time point, in yards (numeric)
- `o` : Player orientation (deg), 0 - 360 degrees (numeric)
- `dir` : Angle of player motion (deg), 0 - 360 degrees (numeric)
- `event` : Tagged play details, including moment of ball snap, pass release, pass catch, tackle, etc (text)

Figure 8.6: Variable Definitions 5

REFERENCES

[Bec 22] Bechtold, Taylor. "How the Analytics Movement Has Changed the NFL and Where It Has Fallen Short." *The Analyst*, 1 Dec. 2022, theanalyst.com/na/2021/04/evolution-of-the-analytics-movement-in-the-nfl/.

[Cha 18] Chahrouri, George, and Eric Eager. "The Stability of Play from a Clean Pocket by NFL QBS: NFL News, Rankings and Statistics." *PFF*, 7 Aug. 2018, www.pff.com/news/pro-pff-forecast-the-stability-of-play-from-a-clean-pocket-by-nfl-qbs.

[Con 16] Conti, Sal. "Evaluating Upfield Burst for Defensive Linemen." *Inside The Pylon*, 22 Aug. 2016, insidethepylon.com/football-101/scouting/2016/07/19/evaluating-upfield-burst-defensive-linemen/.

[Dec 22] DeCort, Callie. "NFL Salary Cap Explained." *The 33rd Team*, 26 Sept. 2022, www.the33rdteam.com/category/analysis/explaining-nfl-salary-cap/.

[Eag 23] Eager, Eric. "The Weak-Link Nature of Football." *SumerSports*, 16 May 2023, sumersports.com/the-zone/the-weak-link-nature-of-football/.

[Fal 22] Falce, Andrew. "NFL Study: How Important Is an Offensive Line to Success?" *NFL Spin Zone*, 7 Sept. 2022, nflspinzone.com/2022/09/07/nfl-study-important-offensive-line-success/.

[Fro 18] Frost, Jim. "When Do You Need to Standardize the Variables in a Regression Model?" *Statistics By Jim*, 12 Nov. 2018, statisticsbyjim.com/regression/standardize-variables-regression/.

[Hef 22] Heffernan, Brendan. "Which Sport Has the Longest Season? (Surprising Answer!)." *Dunk or Three*, 12 Jan. 2022, dunkorthree.com/which-sport-has-longest-season/.

[Joh 23] Johnson, Daniel. "R Random Forest Tutorial with Example." *Guru99*, 27 May 2023, www.guru99.com/r-random-forest-tutorial.html.

[Mul 16] Mulholland, J. (2016). "Optimizing the Allocation of Funds of an NFL Team under the Salary Cap, while Considering Player Talent," *Joseph Wharton Scholars*.

[NFL 23] "NFL Big Data Bowl 2023." *Kaggle*, 2023, www.kaggle.com/competitions/nfl-big-data-bowl-2023.

[Orr 15] Orr, Conor. "Deep Dive into the Wild History of the High-Stakes World of Modern Scouting." *NFL.com*, 18 Mar. 2015, www.nfl.com/news/sidelines/the-history-of-scouting.

[Pin 23] Pincus, Eric. "NBA 2022-23 Luxury Tax Tracker." *Sports Business Classroom*, 27 May 2023, www.sportsbusinessclassroom.com/nba-2022-23-luxury-tax-tracker/.

[Plu 22] Plummer, Andrew. "Box-Cox Transformation and Target Variable: Explained." *Box-Cox Transformation and Target Variable: Explained*, 16 Sept. 2022, builtin.com/data-science/box-cox-transformation-target-variable.

[Sob 17] Sobleski, Brent. "Offensive Line Value Continues to Evolve in Pass-Happy NFL." *Bleacher Report*, 20 Sept. 2017, bleacherreport.com/articles/2650157-offensive-line-value-continues-to-evolve-in-pass-happy-nfl.

[Spo 21] "NFL OL Salary Rankings 2021." *Spotrac.com*, `www.spotrac.com/nfl/rankings/2021/cash/offensive-line/`.

[Wal 22] Walder, Seth. "NFL Analytics Survey 2022: Teams That Use Advanced Metrics Most, Least." *ESPN*, 6 Dec. 2022.