Table of Contents

Abstract / Executive Summary

The National Football League is the premier American Football organization and one of the largest sports leagues in the world. Unfortunately, its players experience injuries at an extremely high rate. This is a growing criticism of the sport, and a detriment to both the potential growth of the league and the safety of its players. Identifying the relationships that various game-related factors such as surface type, temperature, and player position have on player injuries would not only be useful for identifying injurious trends but also help to potentially mitigate the overall injury rate of the league.

Previous research is largely targeted towards either ligament-specific injuries or head trauma as a measurement of injury, but research about general player injury- even minor, is sparse. Additionally, there are very few studies that consider environmental factors as a catalyst for injury. We use several years' worth of nflfastR's play-by-play data to aggregate a dataset that details every play that occurred during the 2019 and 2023 seasons, then created our own injury identification variable using the description of each play to determine if the resulting outcome of each play featured an injury (regardless of its severity). We attempt to discover correlation between injury occurrence and the aforementioned game variables using exploratory data analysis, and then utilized various predictive modeling techniques such as: logistic regression, random forest, and naive bayes classification, to predict potential injury occurrence for a given play.

Ultimately, we found the random forest model to be most effective at predicting injuries. It is largely dependent on player position and an engineered feature called 'injury_frequency'. Aside from these features, other models found variables related to downfield passing to be useful predictors of injury as well. As the NFL becomes more and more passing focused, the implications of our findings as they relate to potential injury mitigation are worth exploring.

An additional analysis, using naïve bayes, on which player positions were most susceptible to injury showed that several groups had a higher incidence of injury as a result of a larger volume of usage. With predictive capabilities, this could make a case for certain groups to rest and rotate more frequently during a game. This could be an especially important research question moving forward as the game continues to get faster and therefore, incorporate a higher volume of total plays.

Project Plan

Predicting Causation of Player Injury in the National Football League Using Play-by-play Data

Jon Griffith
Chris Hicks
Donny Houghton
Aaron Stephenson

Profile and background of business

Business Description:

The National Football League (NFL) is a professional American football league consisting of 32 teams. It is one of the major sport leagues in the United States for one of the most physically demanding sports in the world. The goal of American football is to get from one end of a 100-yd length field to the other in a series of four plays, traveling in ten-yard increments.

The way to prevent scoring is by utilizing defensive players to tackle the players on the offensive side of the ball driving down the field. All players are required to wear protective equipment and adhere to safety guidelines in how they hit and tackle each other. Despite these protocols, an average of 226 injuries per season have been recorded from the past eight seasons (NFL.com, 2023). Research into how to mitigate injuries has been a focus of the league and associated researchers for decades.

Specifications:

Latest Financial Data - 2022
Total Revenue Across NFL and All 32 Teams - $18.6 billion (Statista, 2022)
Number of Players in NFL - 1,696 (Sprtskeeda 2021)
Number of Players in College Football - ~71,000 (Sprtskeeda 2021)
Number of Players in High School Football - ~1.09 million (Sprtskeeda 2021)

Research Questions and Hypotheses:

RQ1: What factors most heavily influence player injuries in the National Football League?

This is an important research question because it can help provide answers for ways to reduce injuries. This can also change variables in the game that can go overlooked, such as weather conditions and whether a stadium should be covered depending on its location. Another variable might be player usage percentage. If there's a strong correlation between a players usage and an injury, that could be a restriction that needs to be implemented.

RQ2: What types of injuries can be predicted in future games?

This research question is important because it can help contribute to making better decisions in player safety. If certain variables in combination with one another contribute more heavily toward player injury, those variables can be focused on to reduce player injuries.

RQ3: Which positions are the most likely to suffer injury given multiple compounded factors as a game goes on?

This research question is important because it dives more deeply into specific player groups to help parse out differentiating factors between the groups. If certain groups can be helped more than others, that is still a success.

RQ4: Can the likelihood of an injury occurring be predicted with any reasonable probability?

This research question is similar to RQ2 but focuses more broadly on if any injury can be predicted. It is important for similar reasons.

RQ variables considered:

1. Indoor vs Outdoor stadiums
2. Weather conditions for outdoor stadiums
3. Surface material of field
4. Player usage percentage in a game

Hypotheses:

H-Variable 1: Indoor stadiums have fewer injuries than outdoor stadiums
H-Variable 2: Severe weather conditions contribute to a higher rate of injury
H-Variable 3: Artificial surfaces have more injuries than natural grass
H-Variable 4: Players with more than 60% usage in a game have a higher rate of injury
H-Variable 5: Passing plays contribute to a higher rate of injury than running plays

Data

The dataset comes from an R package hosted on GitHub called "nflfastR" (2024). This package contains scraped datasets that contain every play ran for all 32 teams in the NFL for every game from 1999 to 2023. The developers of this package are Sebastian Carl and Ben Baldwin. The data consists of 372 variables that together, contain just about every detail you could want. The key variables we are focusing on in this project are:

Weather-related:

- Roof - indicates roof status of the stadiums the game was played in
- Temp - temperature at the stadium only for 'roof' = 'outdoors' or 'open'
- Wind - speed of wind in miles/hour for outdoor stadiums
- Weather - Describes weather of the game
- Stadium - location of the game

Injury-related:

- Desc - Detailed string description for the given play that includes whether a player was injured.
- Player_pos – Position of the player that was injured
- Tot_snap_count – Total number of snaps played by player before injury

Surface

- Surface - what type of ground the game was played on

Other descriptive variables will let us know the exact situation of each injury, such as what quarter it happened in, what the situation of the game was for the injured player's team (down by x number of points, away team, etc.), which drive number it happened during, and things of that nature.

Additional variables will emerge that will be derivatives of aforementioned variables as the data is wrangled, cleaned, and better understood. A lot of these will be indicator variables that will help calculate proportions of injuries with respect to key variables, mentioned in the next section.

Measurements

Player usage percentage: how many plays out of all plays on their side of the ball did a player participate in?
Position group: The proportion of injuries attributed to each position group
Field type: The proportion of injuries attributed to each field type (turf, grass, etc.)
Indoor vs outdoor: The proportion of injuries for indoor vs outdoor stadiums
Weather type: The proportion of injuries for certain weather conditions

One of the key things we will measure is player usage percentage. We want to learn about the usage percentage of each player that was injured to better analyze the impact of things like fatigue and wear-and-tear as the game goes on. If a strong correlation is found, this could have big implications on what a good threshold should be for player usage to promote safety.

Most of the other measurements will be proportions of injuries with respect to things like position group, surface type, indoor vs outdoor, and weather type for outdoor stadiums. These proportions will play a large role in determining the magnitude of each variable's potential contribution toward injuries.

Methodology

For our research question, we will be using a combination of research on existing literature for causations of injury and data visualizations from our described dataset to first understand the data and which variables are heavily involved with player injuries. Knowing those proportions will help us filter our data down further to determine which variables to pursue further and which ones could potentially be thrown out. This can be done using a combination of density graphs, histograms, and pie charts.

Once we decide which variables we'd like to keep, we'll begin building several models to classify the data. Naïve Bayes may be a useful model to incorporate the proportions of each key variable's contribution toward player injury and using maximum likelihood to predict future injuries. Another good model may be Random Forest Classifier, given the number of variables we will be using and potential non-linearities between any one variable and the target variable. If we have large effects from the combination of different variables, Random Forest should help parse out these combinations to predict future injuries. Lastly, logistic regression analysis would be a good model to use as well. Logistic regression will measure log likelihoods for each variable but rather than just have a binary 1 or 0 result, we can also get percentage results that can give us a percentage chance of injury for more precision. We will ensure we use cross validation in some form. Train/test splits and k-fold cross-validation would be good candidates to help us prevent overfitting the data.

Campaign Implementation

The National Football League (NFL) is an entertaining, but dangerous sport. While not all dangers can be mitigated, research into questions that aren't easily identified could help determine ways to mitigate dangers that aren't given enough thought.

Our research questions could benefit current and future players. Implementing new rules and guidelines could help reduce injuries if causation is better understood. Research questions such as "which surface type results in the fewest injuries?" and "Do weather elements have a significant effect on whether an injury occurs?" can provide easy fixes that cost a fixed amount of money. Less tractable problems such as how a player should tackle, how a player should fall, and things of that nature are hard to address and implement solutions. Problems like “which turf should be used?” or “should a stadium have a dome cover?” contain solutions that can be implemented quickly without ambiguity.

Developing successful prediction models that incorporate these variables can help strengthen the argument that these variables should be given more attention and addressed. Even if strict rules are not implemented as a result of this analysis, an education of potential correlations between injury and these variables can at the very least give teams guidelines on how to operate their team and use their players.

Literature Review

Over the past decade, the use of data analytics has gained prevalence in the National Football League. Teams utilize data analytics during games to provide success rates of potential plays by comparing them to the outcomes of historically similar situations. On 4th downs, television broadcasts will show a graphic recommending data-driven action based on field position and the likelihood of success. Data is recorded for nearly every possible variable of a game. It's worth exploring which factors are the most influential in predicting injuries that occur during NFL games.

The stadium alone provides several unique variables. Wind, roof, temperature, and surface type are all variables dependent on where the game is taking place and can dictate the occurrence of such events. Surface type (synthetic turf vs. natural grass) in particular is a contentious topic in football media. There is a commonly held belief that a greater number of injuries occur on synthetic turf than natural grass. A cohort study, 'Higher Rates of Lower Extremity Injury on Synthetic Turf Compared With Natural Turf Among National Football League Athletes' from 2019 hypothesizes that synthetic turf produces a greater number of injuries because synthetics don't allow cleats to release from the surface as easily as they would on natural grass. The authors suggest that lower body injuries that occur without a direct blow to the affected body part are likely the most common injury types associated with surface type. Injury reporting is mandatory for every team in the NFL. The data for this study was collected using injury reporting data from 2012-2016 from all 32 teams to eliminate potential bias. The authors go on to explore the differences between surface types and conclude that even 'hybrid' fields that contain a mix between synthetic and natural grasses were classified as 'natural' rather than 'synthetic' surfaces for the purposes of the research. This is an important consideration, as there are also subclasses of synthetic turf which can either be explored individually or classified simply as 'synthetics'. The authors use a Poisson model to "estimate the influence of surface type on lower body injury groupings" (Mack et al., 2018). The results found that over the span of the five seasons measured there were 16% more lower body injuries on synthetic turf than on natural grass, and that there is a "causal impact on lower extremity injury". Additionally, it is suggested that if the results of their findings were to be applied, it is expected that 319 fewer lower body injuries would have occurred between the 2012-2016 seasons (Mack et al., 2018).

One limitation to this argument is that many different factors can catalyze injury, not just surface type. Just because an injury occurred on synthetic turf, doesn't necessarily mean that the turf was responsible in every case. An additional limitation is that the data is dependent on proper injury reporting, which has been known to be questionable and is an entirely separate issue to explore. We're going to attempt to remedy this by classifying injuries based on their occurrence during a game, rather than relying on team injury reports. This also ensures that the injuries did not occur during practice or any other external surface-independent event. There is a common occurrence of practice-related injuries that qualify players for injury reporting, but these injuries are fundamentally different than the ones that are measured in-game. Although it's the specific focus of the research, this study only provides information about lower body injuries, it doesn't consider concussions or upper body injuries that also occur during games. The authors recommend that synthetic turf should evolve to be more compatible to the game rather than be eliminated altogether.

Additional considerations regarding in-game injuries are environmental and weather conditions. A 2018 study looked at game-day temperature, humidity, dew point, and barometric pressure as potential

influencers of concussions. Regular season games from the 2012-2015 seasons were each paired with respective meteorological data gathered from Weather Underground. Each weather measurement was segmented into intervals rather than individual degrees. For the statistical analysis, the first method used was "…correlation analysis for all the aforementioned environmental factors. A t-test and ANOVA were performed…on all four environmental factors and their role in the incidence of concussion" (Haider et al., 2018). Afterwards, both Poisson and logistic regression were performed on the same factors to predict concussion probability. The results of the ANOVA suggest that there is a "significant decrease in concussion incidence with increasing temperature" (Haider et al., 2018). The Poisson regression model produces a prediction of concussions per game given the temperature in degrees Fahrenheit. Similarly, the authors suggest that given the strong correlation between temperature and dew point, a decrease in dew point also corresponds with increased concussion frequency. The authors also state that their model found no individually significant predictors because of this correlation, so it was difficult to determine the coefficient that resulted in an increase of standard error. A model that incorporates fewer correlated variables may result in improved prediction accuracy. There was nothing to suggest that barometric pressure or humidity had any influence on concussion occurrence. One element that limits the breadth of the study is the sole focus on concussions as a metric for injury. Orthopedic injuries are not considered, and it would be worthwhile to see if these same conditions hold true for other injury classifications. Finally, the study acknowledges that research on weather as a catalyst of NFL injuries is sparse, and further research may be beneficial for improving player safety. An environmental variable that was not considered in this study is the impact that wind has on injuries. Windy games often experience less frequent passing plays since there is less control over the ball's trajectory while in the air. We hypothesize that running backs may experience more frequent injuries in windy conditions due to an overcompensated running strategy as a biproduct of reduced passing opportunities due to the wind.

Another potential category to explore is the effect of fatigue on injuries, using snap count as one proxy. In general, literature pertaining specifically to a correlation between snap count and injuries seems to be sparse. Allahabadi et al. (2022) gave a comprehensive study measuring the differences between seasons from 2013-2019 that all had similar characteristics and the 2020 COVID-19 season. They specifically looked for causation for increased ACL tears in the 2020 season. While their focus was on causation from differences in schedule and in-person training during the season, they provided average measures with upper and lower bounds for multiple variables, including snap count, during the time of injury. They found an average of 2049 plus or minus 2323.2 career snaps at the time of injury. While this provides a large threshold for snap counts per injury and may make the snap count seem trivial, its important to remember that they only measured for ACL tears. Our analysis will include every type of injury and potentially fill the gap between less and more severe injuries.

Furthermore, snap counts may have no significant impact on ACL tears, as they alluded to in their study. They found the average in-game snap count for a player who sustained an ACL rupture was 25.7 plus or minus 19.3 snaps. With around an average of 70 snaps per game, this doesn't seem to support a large snap count being correlated with ACL tears. This could be an important thing to think about with our analysis as we analyze all injuries. Something worth trying may be to remove ACL tears from our analysis, potentially leading to a larger magnitude for all other injuries in relation to snap count.

Another interesting result from their analysis was the most frequent positions of players sustaining tears, giving proportions for each. This would be an interesting new variable to add to the analysis to dig down further into which specific players are more susceptible to injury in general. This broadening of the

analysis could lead to more useful variables and more branches in a decision tree analysis in predicting the likelihood of an injury.

A study conducted at the University of Miami Sports Medicine Institute by Perez et al. (2019) looked at the effect that the amount of rest between games had on injuries. They looked at all injuries between 2013 and 2016 that resulted in play stoppage. They quantified short, regular, and long rest times being four days, six to eight days, and ten plus days, respectively. They found a significant association between rest between games and injuries. But oddly enough, Thursday night games (short rest time) had the fewest injuries per game. They did observe quarterbacks having more injuries on shorter rest weeks but did not reach statistical significance on that finding.

While this result by Perez et al. does not support a fatigue-related positive correlation for injuries, they did have major disparities in their sample size for number of games they recorded. They had 128 short rest weeks, 1530 regular rest weeks, and 262 long rest weeks. Perhaps with a larger sample, such as extending out to seven more years, we can extend our analysis to rest time between games and potentially find a stronger correlation with each category. Another interesting takeaway from their study was that they quantified the rate of injury per game in relation to player position and found that skill positions (wide receiver, running backs, and defensive backs) had the highest rate of injury compared to other position groups. That is another compounding study that supports looking into a positional group breakdown in our analysis.

Injuries are a significant concern in the NFL due to their commonality, impact on players health, teams' performance, and the leagues success. The NFL is extremely physically demanding, involving high-impact and high-speed collisions, which exposes players to a wide variety of injuries. Studies conducted by the NFL and independent sports medicine researchers indicate that there is a noticeable variation in injury rates across different positions. An increase in injury surveillance, game footage, and player health records have been deployed track these injuries. “Shoulder injuries were highest in QBs (76%), LBs (58%) and DBs (60%). Injuries to the remainder of the upper extremity were prevalent in line positions, especially at the elbow (40% of total injuries in DL and OL). Knee injuries were the most common injury in several large epidemiological studies and were most frequent in line positions and RBs. Ankle and hamstring injuries were most common in sprinting players, including: DBs, WRs, RBs, and LBs. Injuries to the head and spine continue to be of high concern, with the incidence of repeat concussions highest in DBs (17.3%), kickoff team (15%), WRs (11.3%), and RBs (10.2%). Head impacts were most frequent in OL, DL, and LB positions” (Nuelle et al, 2016). This type of research provides insight into which positions are the most susceptible to injuries, the nature of the injuries, and the potential causes.

Research indicates that certain positions in the NFL are more prone to injuries than others. Running backs, linebackers and tight ends often experience the highest injury rates. “A variety of injury mechanisms occur in football players due to a variance in positions and the wide range of skills necessary to play each position. The positions have different mechanisms and types of injury based on their different exposures, (Nuelle et al, 2016) These positions are often involved in high impact physical confrontations, abrupt movements, and high-speed impacts. This increases the risk of musculoskeletal injuries like ACL tears, concussions, and hamstring strains. Being hit by blindside tackles and during

throwing motion often leaves quarterbacks with more severe injuries, although less frequent. Wide receivers and defensive backs are more susceptible to hamstrings and other leg muscle injuries. This is due to the explosive high speed plays they are required to make several times a game. Linemen are more susceptible to shoulder, ankle, knee and back injuries due to the repetitive impact on these places. This shows that the players role in each play, position, and frequency of high-impact collisions play a role in the different types of injuries sustained on the field.

It is crucial to develop effective injury prevention and management strategies base on the understanding of NFL positions. The league invests in protective equipment, training programs, and medical protocols to mitigate injury risk, but the human body is still susceptible to all these types of injuries. An increase in position based rehabilitation and training is always being updated and improved. The league has made improvements by rule changes as well. The NFL's concussion protocol is a testament to the leagues efforts to address the head injury. A study by Pellman et al found that helmet to helmet impacts in the NFL cause 61% of head injuries. Continued research and adaption of strategies are essential for the game. Emerging data on injury trends and advancements in sports medicine and technology are bringing a safer environment for players. These studies and more in the future will help to create an evolving solution for the players and the league.

Final Research Questions

RQ1: What factors most heavily influence player injuries in the National Football League?

This is an important research question because it can help provide answers for ways to reduce injuries. This can also change variables in the game that can go overlooked, such as weather conditions and whether a stadium should be covered depending on its location. Another variable might be player usage percentage. If there's a strong correlation between a players usage and an injury, that could be a restriction that needs to be implemented.

RQ2: Can the likelihood of an injury occurring be predicted with any reasonable probability?

This research question is similar to RQ1 but focuses more broadly on if any injury can be predicted. It is important for similar reasons.

RQ3: Which positions are the most likely to suffer injury given multiple compounded factors as a game goes on?

This research question is important because it dives more deeply into specific player groups to help parse out differentiating factors between the groups. If certain groups can be helped more than others, that is still a success.

Exploratory Data Analysis

We used the R library “nflfastR” for the data in our project. All of the data is structured data and was able to be downloaded into csv files directly from https://www.nflfastr.com/ organized by season. We decided to go back 5 years to get a sufficient amount of data and combine all those files into 1 large file using a Python script. This date range from 2019-2023 left us with 246,318 distinct observations. Each observation represents a different play from every single game in the NFL over the course of those seasons. When we filtered the data by whether an injury had occurred on that play, we were left with 4,422 distinct observations. We are leaving all our observations intact so that we have data to compare against in terms of “injury or not”. However, we are going to trim the amount of variables in the dataset. There are 372 total columns in the original dataset. We are only concerned with the ones that impact what we are studying in terms of injuries. This will include 47 different columns, which may seem excessive but we figured we would keep some of them just in case. These are the 47 we are keeping and a brief description of each one:

play_id	Numeric play id that when used with game_id and drive provides the unique identifier for a single play.
game_id	Ten digit identifier for NFL game.
home_team	String abbreviation for the home team.
away_team	String abbreviation for the away team.
season_type	'REG' or 'POST' indicating if the game belongs to regular or post season.
posteam	String abbreviation for the team with possession.
defteam	String abbreviation for the team on defense.
side_of_field	String abbreviation for which team's side of the field the team with possession is currently on.
yardline_100	Numeric distance in the number of yards from the opponent's endzone for the posteam.
game_date	Date of the game.
quarter_seconds_remaining	Numeric seconds remaining in the quarter.
game_half	String indicating which half the play is in, either Half1, Half2, or Overtime.
drive	Numeric drive number in the game.
qtr	Quarter of the game (5 is overtime).
time	Time at start of play provided in string format as minutes:seconds remaining in the quarter.
down	Which down the play was for the series of downs.
goal_to_go	Indicator of whether the play was within ten yards of the opponent end zone

ydstogo	How many yards were needed to gain a first down
yards_gained	How many yards were gained on the play.
ydsnet	How many yards total the drive had accumulated up to that play.
shotgun	Indicator of whether the offense was in the shotgun formation (QB lined up about five yards off the line of scrimmage)
no_huddle	Indicator of whether the offense went into a huddle or not before the play (no huddle means the offense is playing faster)
air_yards	An indicator of how many yards the ball was in the air regardless of whether it was caught or not
yards_after_catch	The amount of yards the player who caught the ball ran before being tackled
score_differential	How many point difference between the offense and defense at the time of the play
solo_tackle	Indicator of whether the player with the ball was tackled by a single player
tackled_for_loss	Indicator of whether the player with the ball was tackled for negative yards from the line of scrimmage
series	Which number series of the game for both teams combined (drives for both teams added up)
play_clock	Amount of seconds on the play clock before the ball is snapped
drive_play_count	Number play of the current offensive drive
desc	Detailed string description for the given play.
play_type	String indicating the type of play: pass (includes sacks), run (includes scrambles), punt, field_goal, kickoff, extra_point, qb_kneel, qb_spike, no_play (timeouts and penalties), and missing for rows indicating end of play.
qb_scramble	Binary indicator for whether or not the QB scrambled.
qb_hit	Binary indicator if the QB was hit on the play.
rush_attempt	Binary indicator for if the play was a run.
pass_attempt	Binary indicator for if the play was a pass attempt (includes sacks).
sack	Binary indicator for if the play ended in a sack.
season	4 digit number indicating to which season the game belongs to.
start_time	Kickoff time in eastern time zone.
time_of_day	Time of day of play in UTC "HH:MM:SS" format. Available 2011 and beyond with source "nfl".
stadium	Game site name.
weather	String describing the weather including temperature, humidity and wind (direction and speed). Doesn't change during the game!

location	Either 'Home' o 'Neutral' indicating if the home team played at home or at a neutral site.
result	Equals home_score - away_score and means the game outcome from the perspective of the home team.
roof	One of 'dome', 'outdoors', 'closed', 'open' indicating indicating the roof status of the stadium the game was played in. (Source: Pro-Football-Reference)
surface	What type of ground the game was played on. (Source: Pro-Football-Reference)
temp	The temperature at the stadium only for 'roof' = 'outdoors' or 'open'.(Source: Pro-Football-Reference)
wind	The speed of the wind in miles/hour only for 'roof' = 'outdoors' or 'open'. (Source: Pro-Football-Reference)
stadium_id	ID of the stadium the game was played in. (Source: Pro-Football-Reference)
game_stadium	Name of the stadium the game was played in. (Source: Pro-Football-Reference)
pass	Binary indicator if the play was a pass play (sacks and scrambles included).
rush	Binary indicator if the play was a rushing play.
first_down	Binary indicator if the play ended in a first down.
aborted_play	Binary indicator if the play description indicates "Aborted".
special	Binary indicator if the play was a special teams play.
play	Binary indicator: 1 if the play was a 'normal' play (including penalties), 0 otherwise.
passer_id	ID of the player in the 'passer' column.
rusher_id	ID of the player in the 'rusher' column.
receiver_id	ID of the player in the 'receiver' column.
name	Name of the 'passer' if it is not 'NA', or name of the 'rusher' otherwise.
id	ID of the player in the 'name' column.

Initially we thought we had all the info we needed in the exact format we needed it in, but that was naïve of us. Some of the biggest problems as far as data cleaning goes were identifying the players who got injured and finding out their positions, as well as extracting weather data from the weather column.

The following are variables that were engineered from the above variables for the final models:

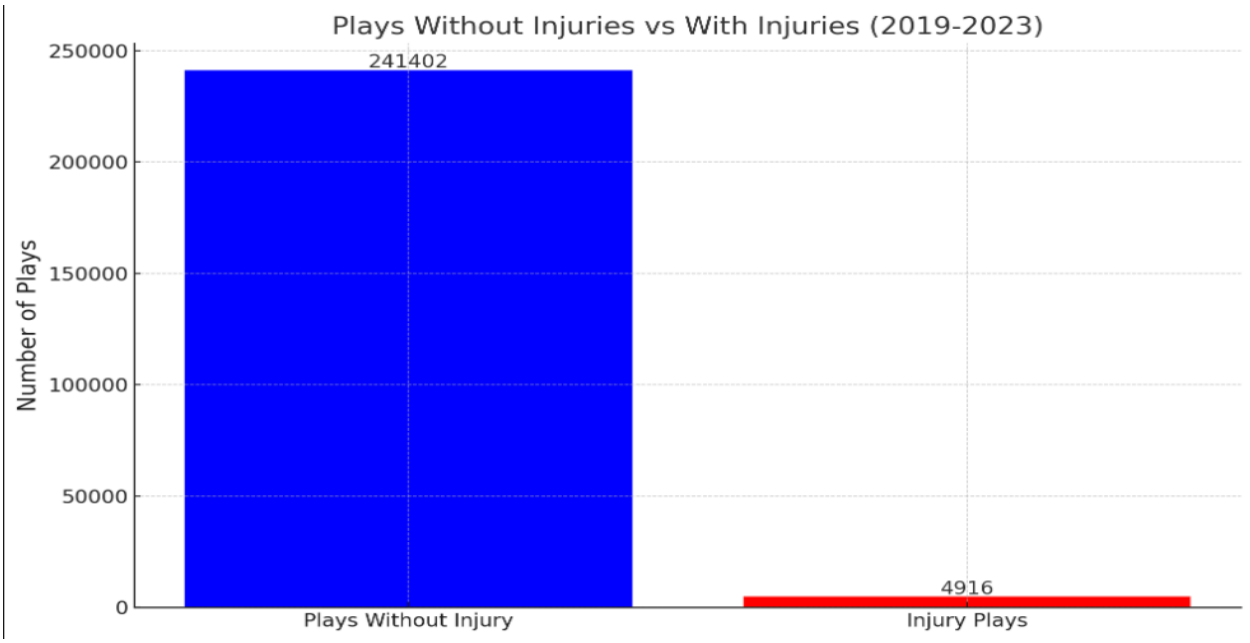
temp	Temperature at the time of kickoff
wind	Indicator of whether there was wind or not
turf	Indicator of whether the field surface was turf or natural grass

outdoor	Indicator of whether the game was outdoor or indoor
total_drive_seconds	How many seconds the current offensive drive has being going
pass_middle	Indicator of whether the play was a pass in the middle of the field or near the sidelines
pass_deep	Indicator of whether the play was a deep pass or short/medium
run_middle	Indicator of whether the play was a run down the middle or run near the sidelines
tot_snap_count	The total amount of snaps the injured player had accumulated up to that play
defense	Indicator variable for whether the player was offense or defense

Identifying the players who got injured was easy to do in Excel, but when we needed to go through 4,000+ rows of injuries we knew we had to create a script of some sort. The players' names were in a "description" column among other information pertaining to that play. Luckily, the NFL stuck to a pattern when identifying the player who got injured that play. It would always be formatted like this: "(3 digit team abbreviation)-(jersey number)-(player name) was injured on the play." However, sometimes there were multiple players injured on one play, so we wrote a Python script that used Regex to identify that specific pattern and then insert the name of the player in a new column. If there was more than one player, the script would create a new row to show a new instance of an injury but keep the same play id so there were still the same number of plays. We ended up with 4,916 distinct injuries from 2019-2023.

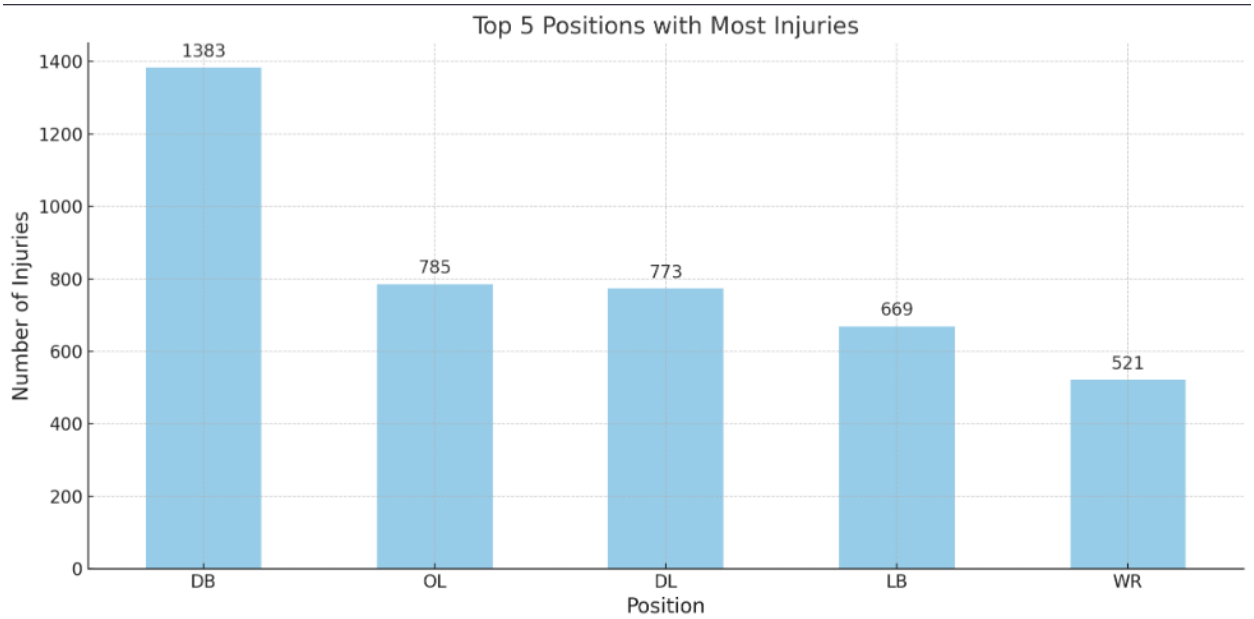
Next, we had to figure out the positions of the players. Not one column in the 372 we had to choose from said "position". So we had to get creative. We looked at the nflreadR package's snap_counts table and it had player information to include position. We created a Python script to create a composite key in both of our tables that looked like this: "(YYYYseason)-(3 digit team abbreviation)-(jersey number)-(player name)". We then joined tables on this key and brought positions into our dataset. One struggle in particular with this process was making sure to include players with suffixes or 2 last names. It was a battle but we ended up winning with the help of some good ol' trial and error in our Regex pattern.

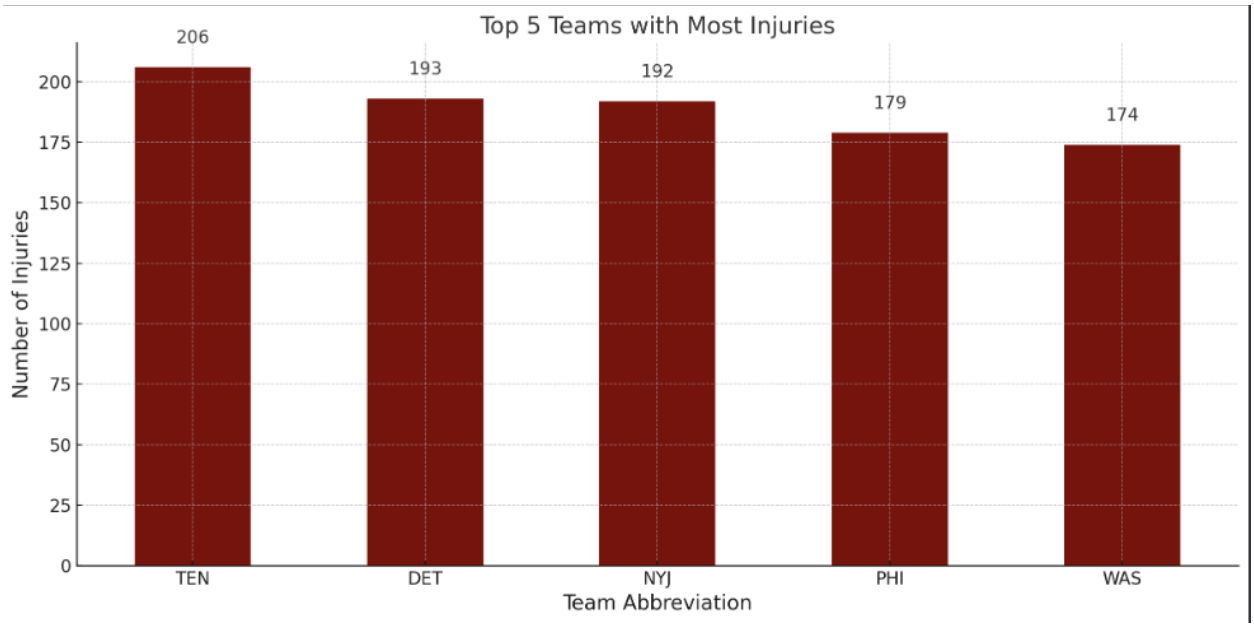
Regex also ended up making the weather column very easy to extract data from because it was all structured the same. One key thing we had to do here was make sure to change the data type of temperature and wind speed to integers.



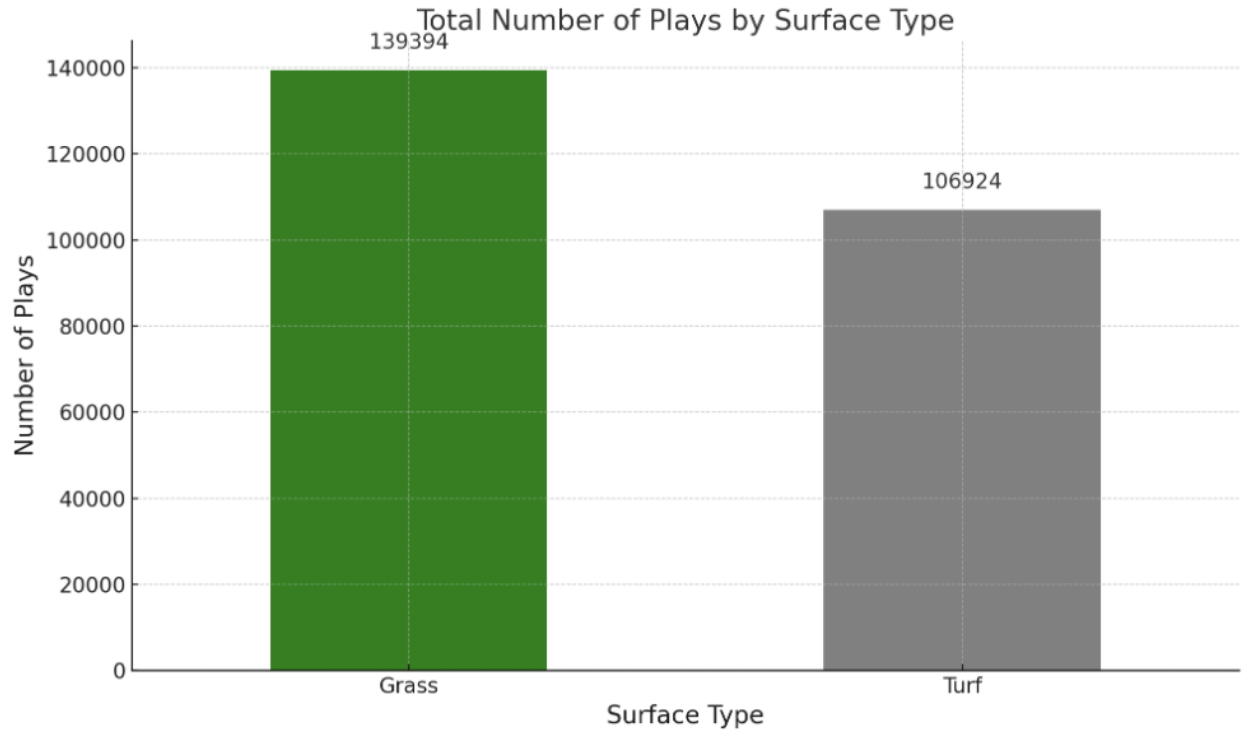
Now lets explore the data! First we'll look at the total number of plays vs the total number of injuries. This gives us a percentage of 1.996%. So about 2% of all plays resulted in injury from 2019-2023.

Now we'll look at the top 5 positions with the most injuries. Defensive Backs seem to get hurt almost 2 times as much as the next most frequent position: the O-Line. From there, it steadily trends down.



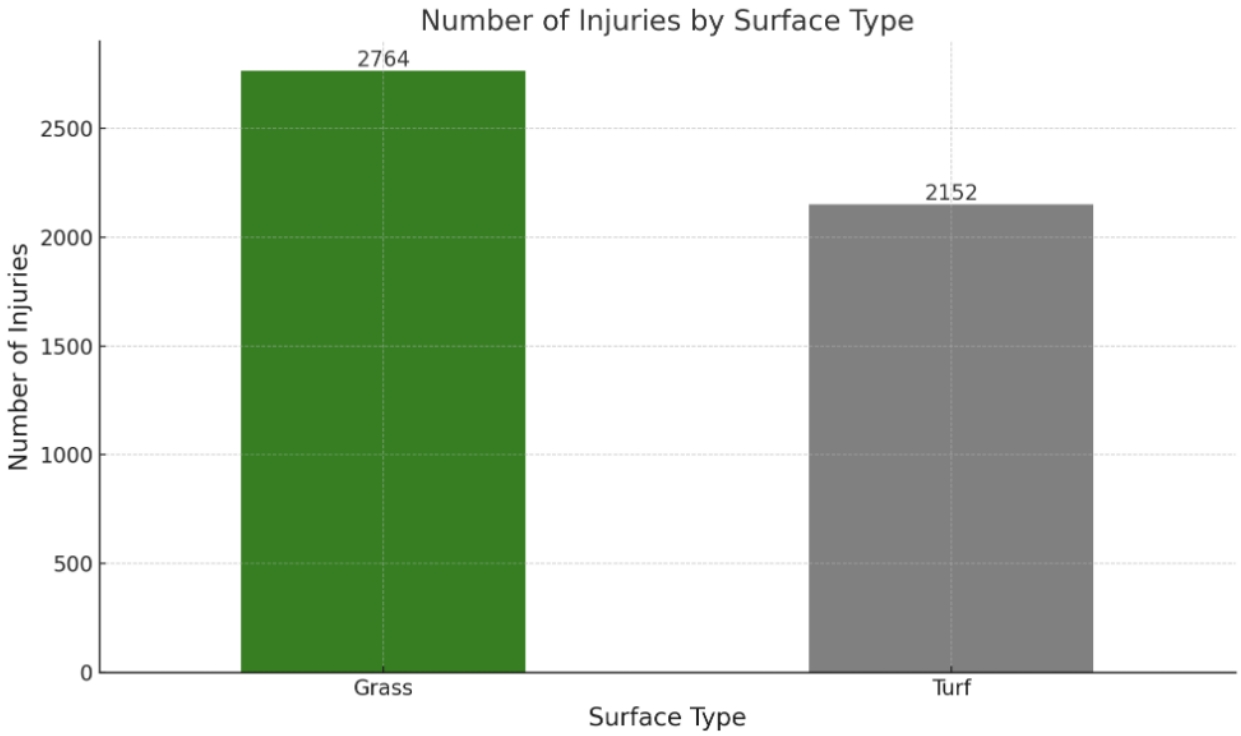


While we are on the topic of positions, it was only a quick tweak to our Python code to get the top 5 teams with the most injuries. We were curious, so here it is. Tennessee Titans leading the way with 206 injuries from 2019-2023.

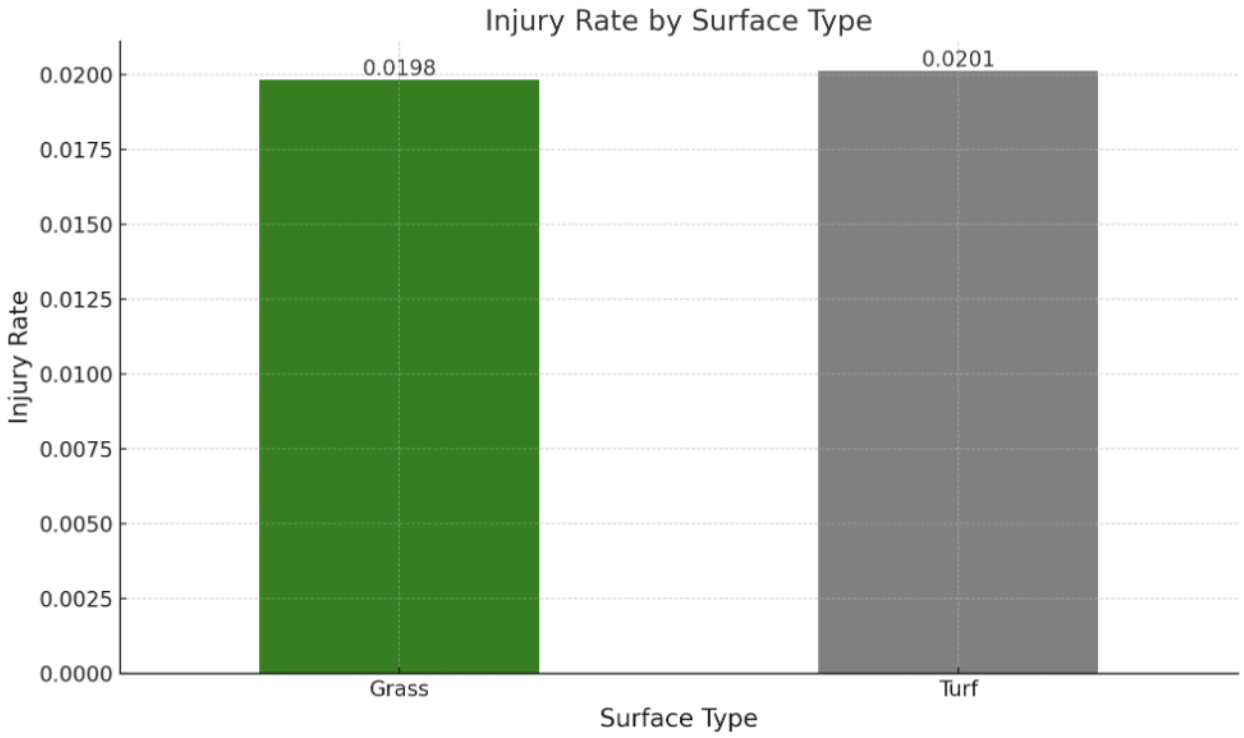


Now we will do a little dive into our surface type analysis. To get an idea of what we're dealing with, we

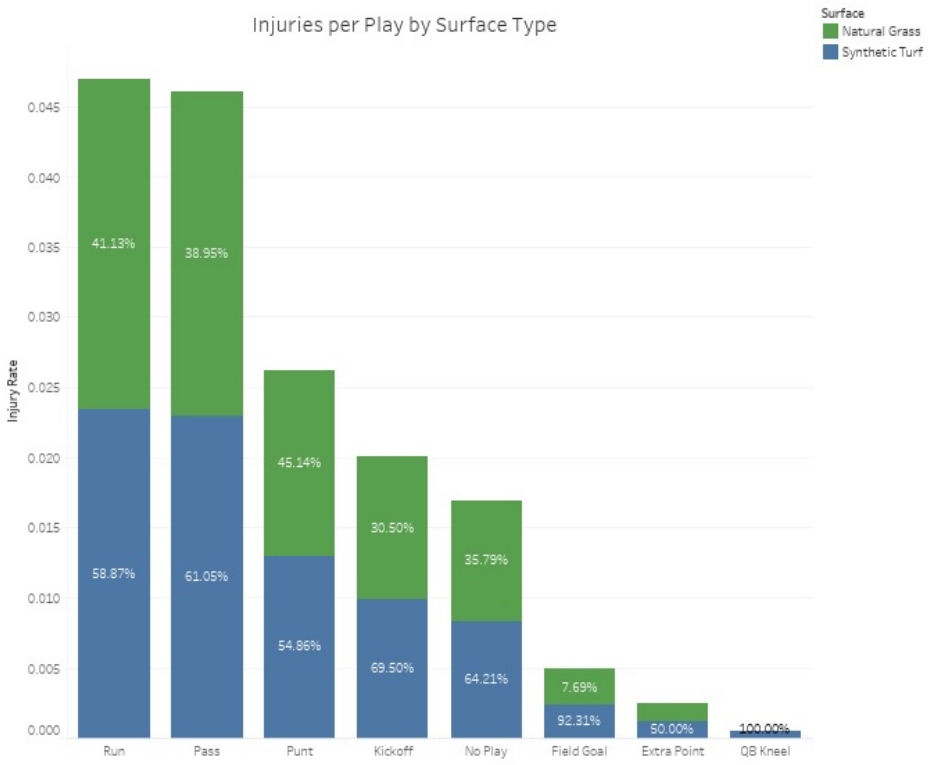
first looked at all plays. The total number of plays played on grass was 139,394. The total plays on turf was 106,924. A little over 3 quarters as many as played on grass.



Similarly, the raw number of injuries on turf is about 3 quarters of the amount of injuries on grass.

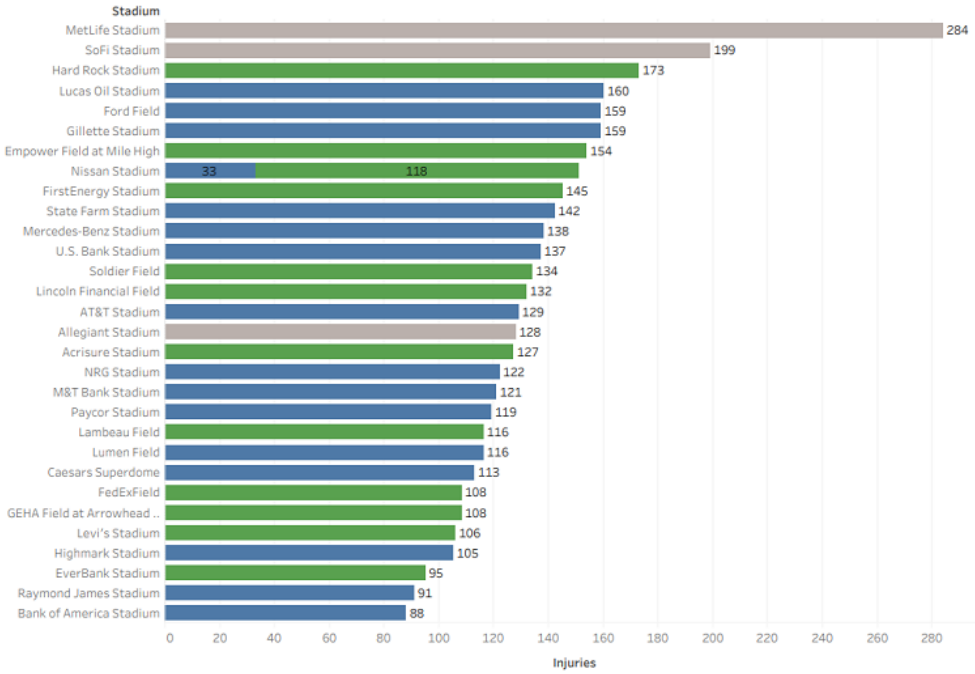


There's a very similar injury rate across the board for both grass and turf surfaces. Taking a more magnified approach while still comparing surface types but also taking play type into consideration yields mostly expected results, but there are a few things to note here.

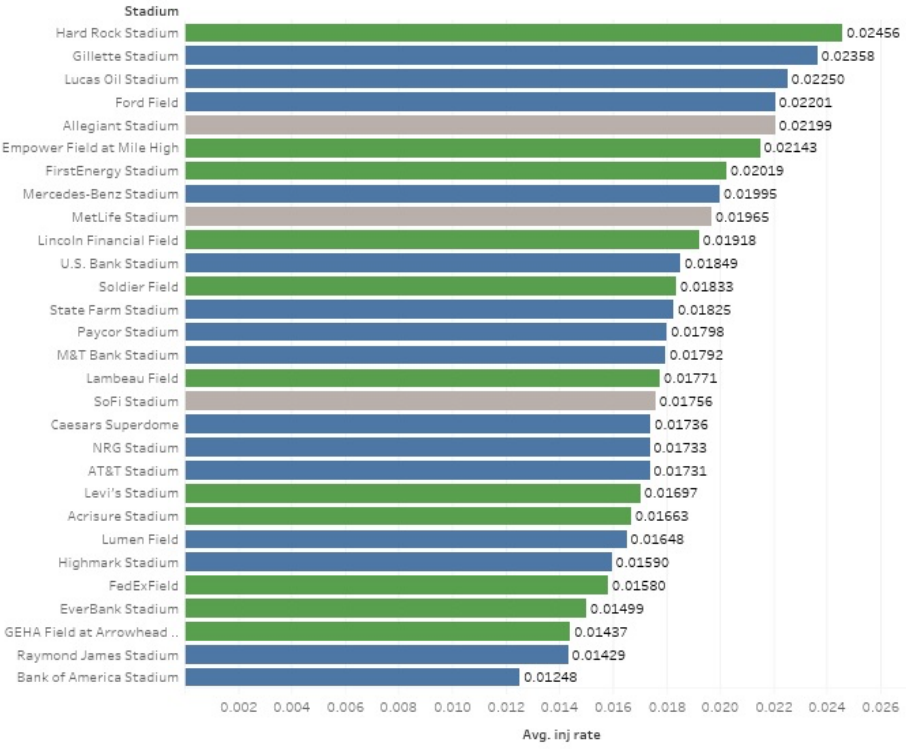


The amount of injuries per play type are roughly proportional to the amount of plays that occur on these surface types respectively. Kicking related plays yield some interesting observations, however. Punts and kickoffs deviate slightly from the ~60:40% ratio between grass and turf, but Field Goals had 12 injuries on turf vs. 1 injury on grass. Another surface-related consideration is whether the stadiums play a role in injuries.

Injury count by surface for each home stadium (2019-2023 seasons)

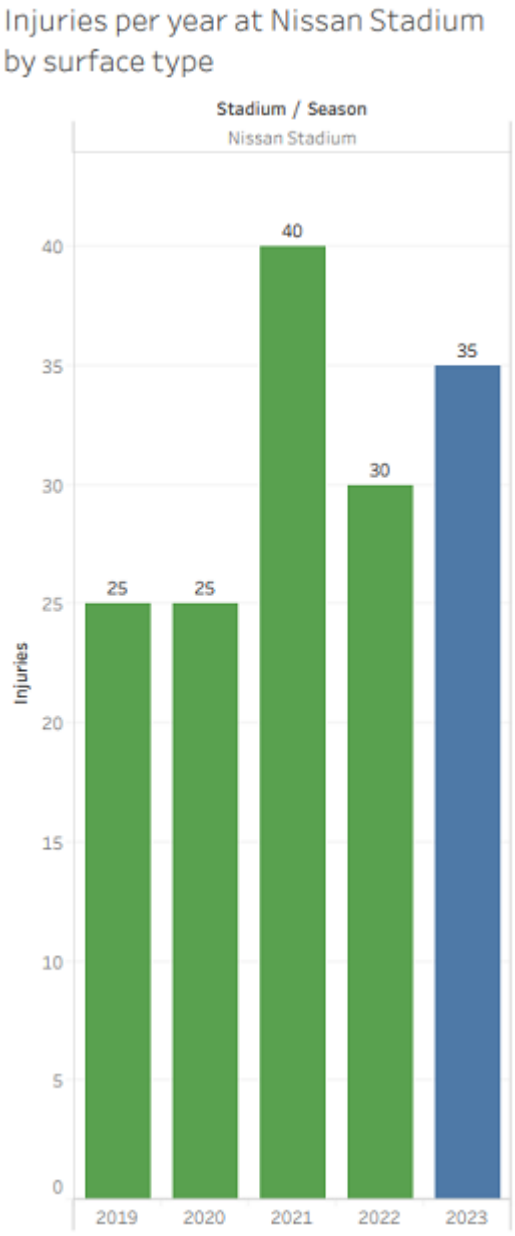


Injury Rate by Surface Type at Home Stadiums (2019-2023 seasons)



At first glance it looked as if MetLife and SoFi (both turf fields) had a noticeably larger amount of injuries compared to the other stadiums. These stadiums should ultimately be omitted, however, since they're the only two stadiums that are home to 2 teams instead of just one (Jets and Giants for MetLife, Rams and Chargers for SoFi). Additionally, Allegiant and SoFi both opened after 2019 which was the start of the collected data, so fewer games took place in these stadiums.

When only considering the remaining stadiums that have more uniform data, Hard Rock stadium - a grass field, is the stadium with the most injuries. Nissan Stadium is interesting to look at, since it was a grass field for the first 4 years of the research period, but switched to turf for the 2023 season.



While it's only one year of data, it's worth noting that the 2023 turf season contained more injuries than the collective average. It doesn't appear as if there's a trend that suggests that stadium injuries are surface dependent.

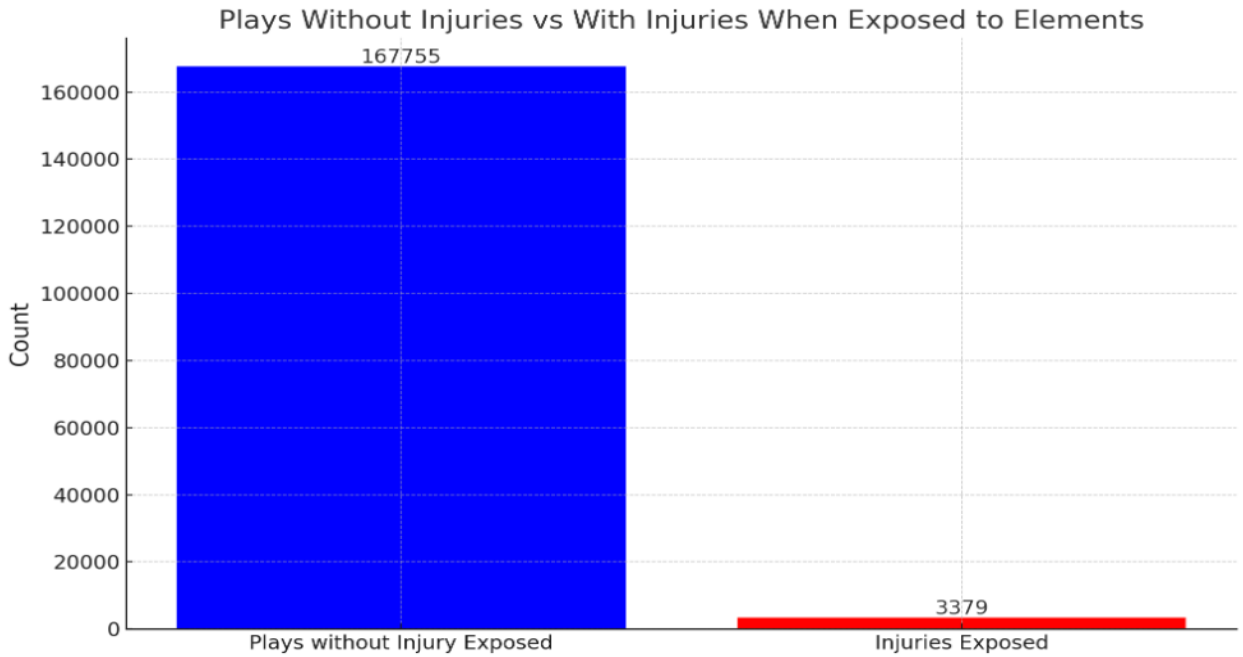
At the end of the day, the difference between injuries on turf vs grass do not pose much of a difference. But what about weather? That's got to impact injuries.

To set the stage for weather related injuries, we first had to create a subset of the data that included either an open roof or an outdoor game. These games are referred to as "exposed to the elements".

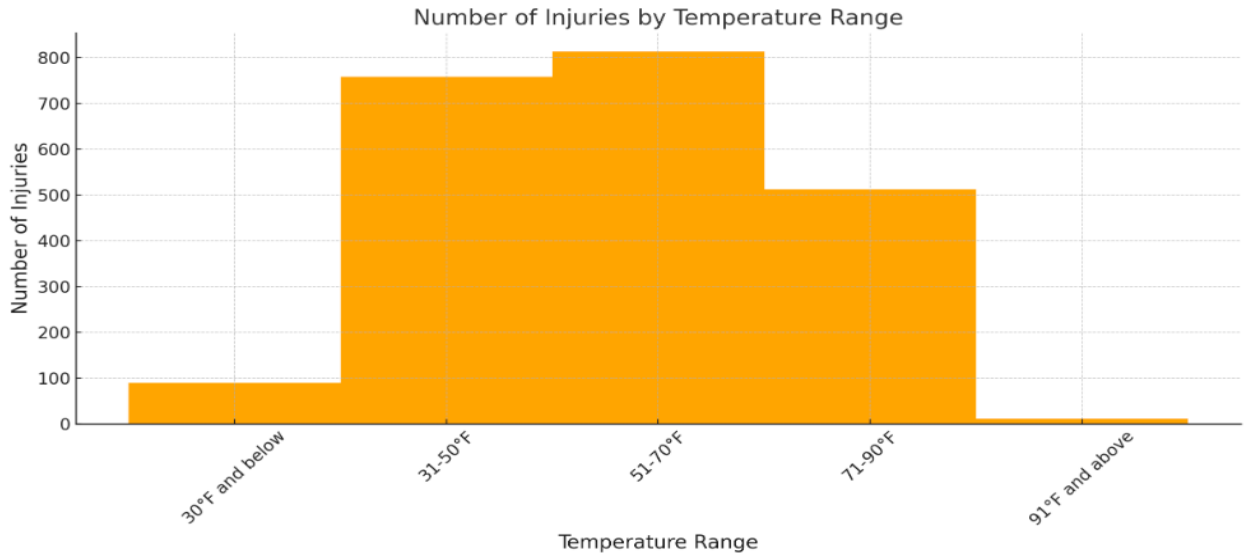
The total number of games played exposed to the elements is 965.

The total number of plays exposed to the elements is 171,134.

The total number of injuries occurring during these plays is 3,379.



We had to break down the weather variable to extract temperatures and wind speeds from it. The temperatures are in degrees Fahrenheit and are bucketized into different ranges as seen below.



The graph is actually pretty normally distributed (most likely due to the number of games played in normal temps vs extreme temps). But, we were curious if temperature had an impact on injuries occurring or not, so we computed the ratios of injuries in each bucketized temperature range. Here's what we discovered:

30°F and below: 89 injuries out of 4,361 plays (Injury Ratio: 0.0204)

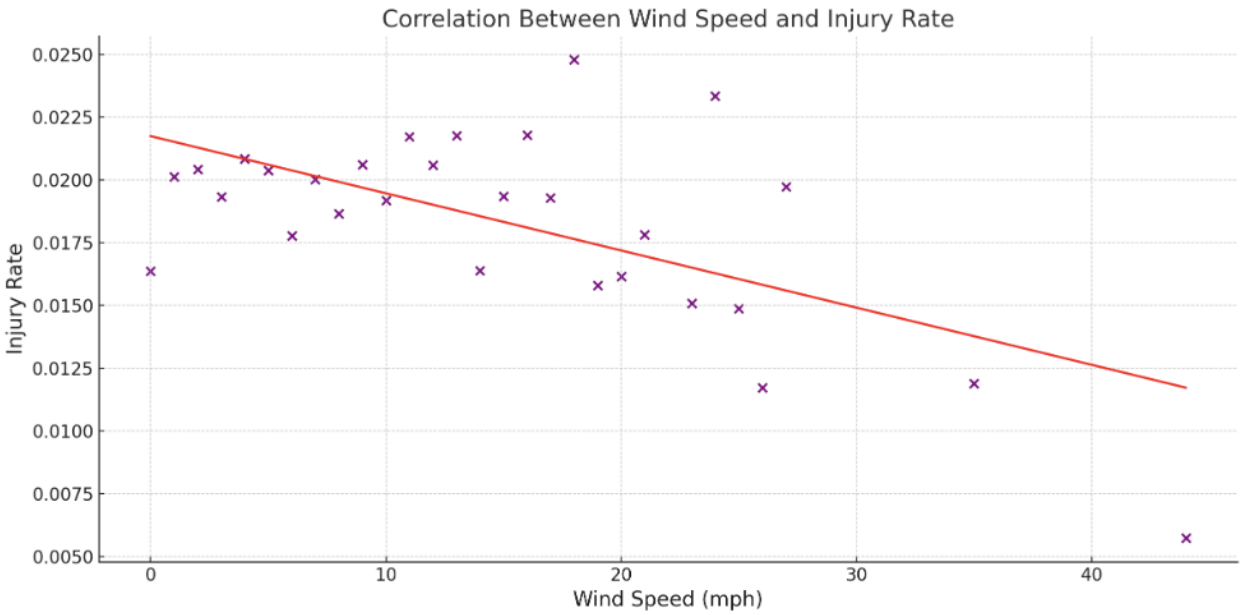
31-50°F: 758 injuries out of 37,244 plays (Injury Ratio: 0.0204)

51-70°F: 814 injuries out of 42,325 plays (Injury Ratio: 0.0192)

71-90°F: 512 injuries out of 26,868 plays (Injury Ratio: 0.0191)

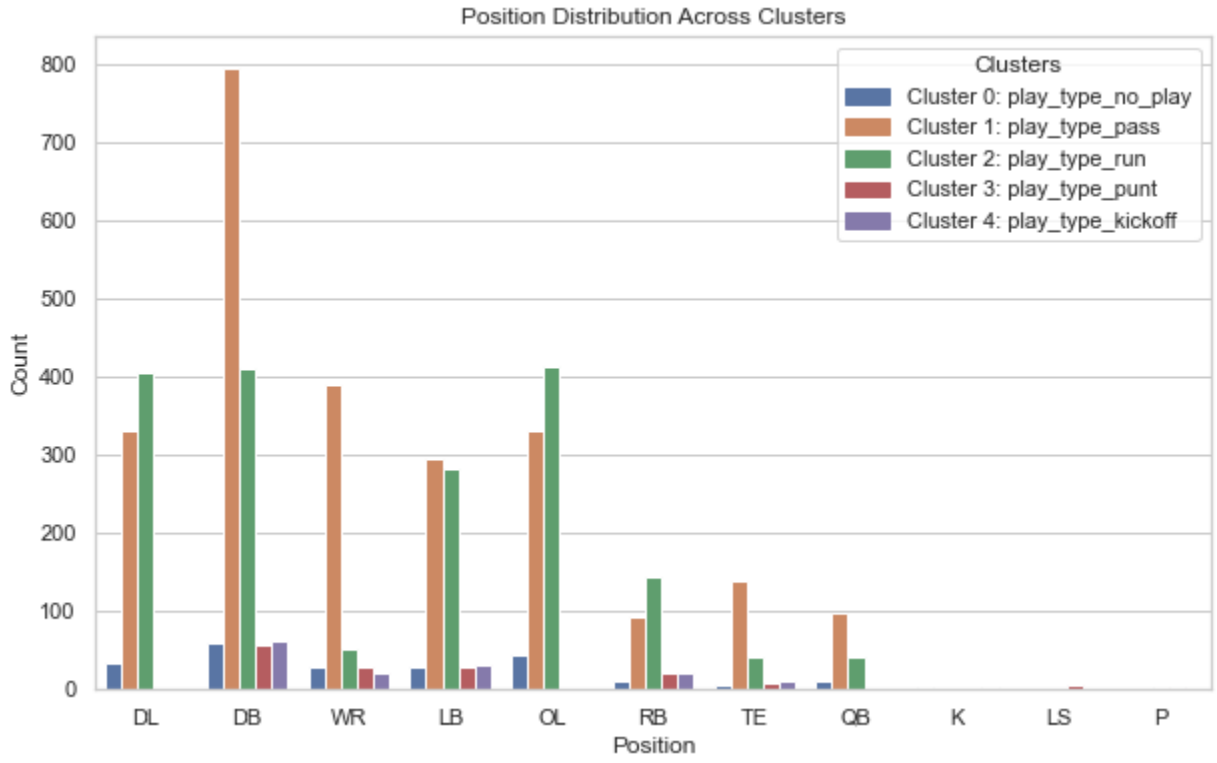
91°F and above: 11 injuries out of 1,030 plays (Injury Ratio: 0.0107)

The injury ratios are pretty consistent all the way through.

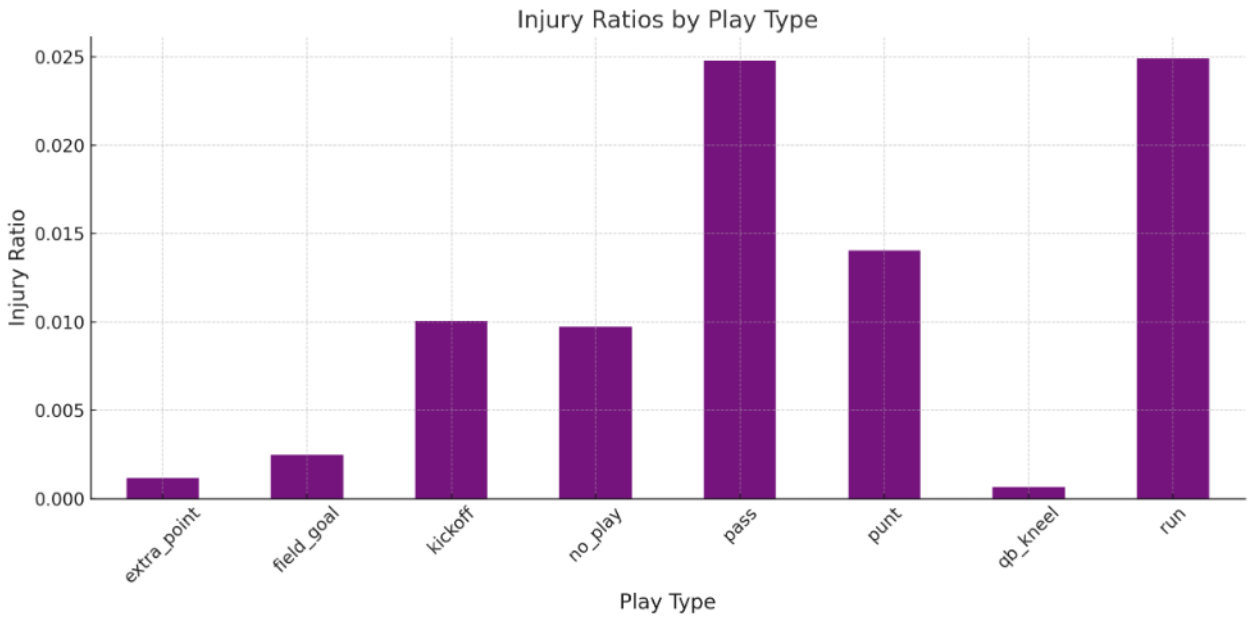


Maybe wind speed has an effect? Here is a graph with injury rate plotted against wind speed.

Surprisingly, there is actually a downward trend here. However, when we think about it, it is again probably just due to less games being played in high wind. And actually, it looks like there is a slight upward trend from 0-20mph winds before losing consistency in the data, which is what we hypothesized in the beginning.



One last variable to check out is Play Type. We were curious on what type of plays end up in the most injuries. Passes and Runs lead the way, but these also lead the way in overall play types performed by a landslide.

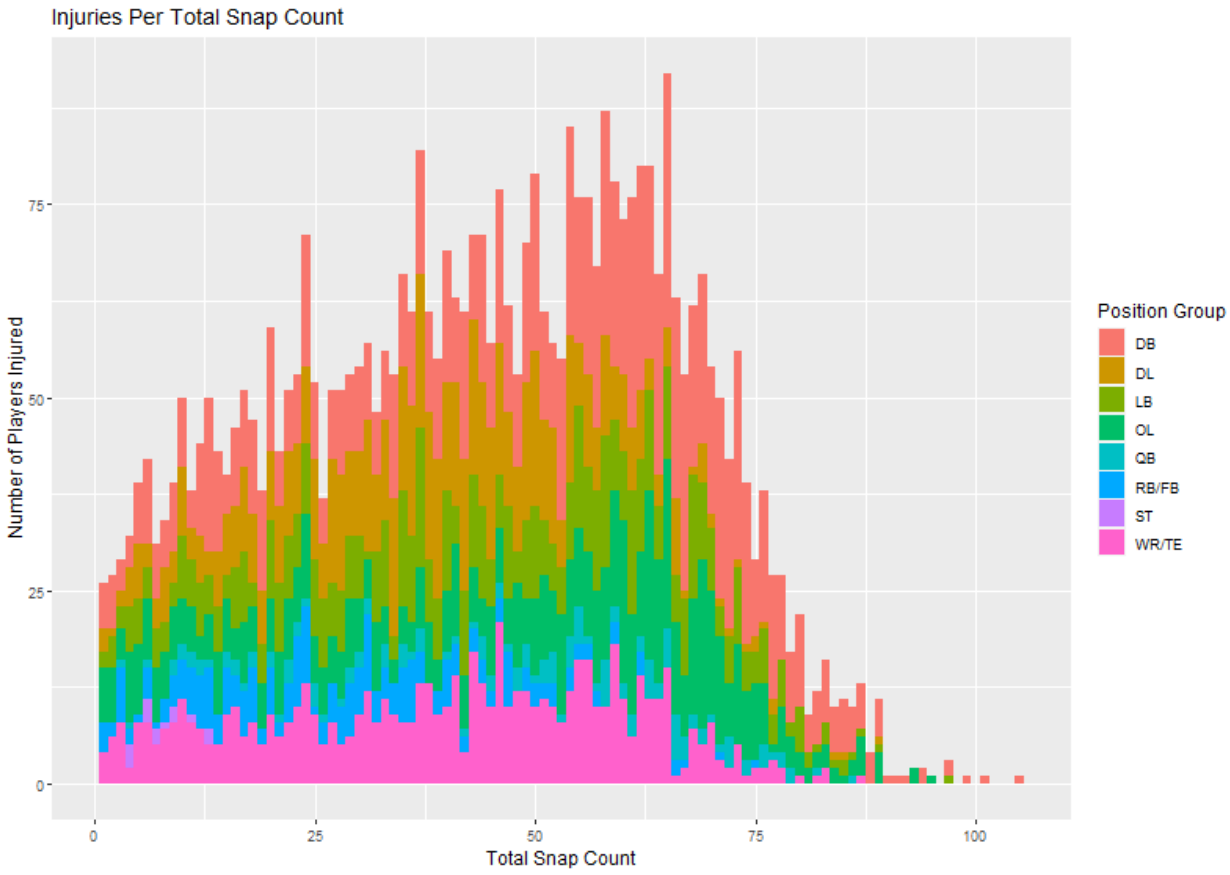


So what are the odds a player will get injured on a specific play type? Is passing a more dangerous play than running?

Nope. Running took the lead.

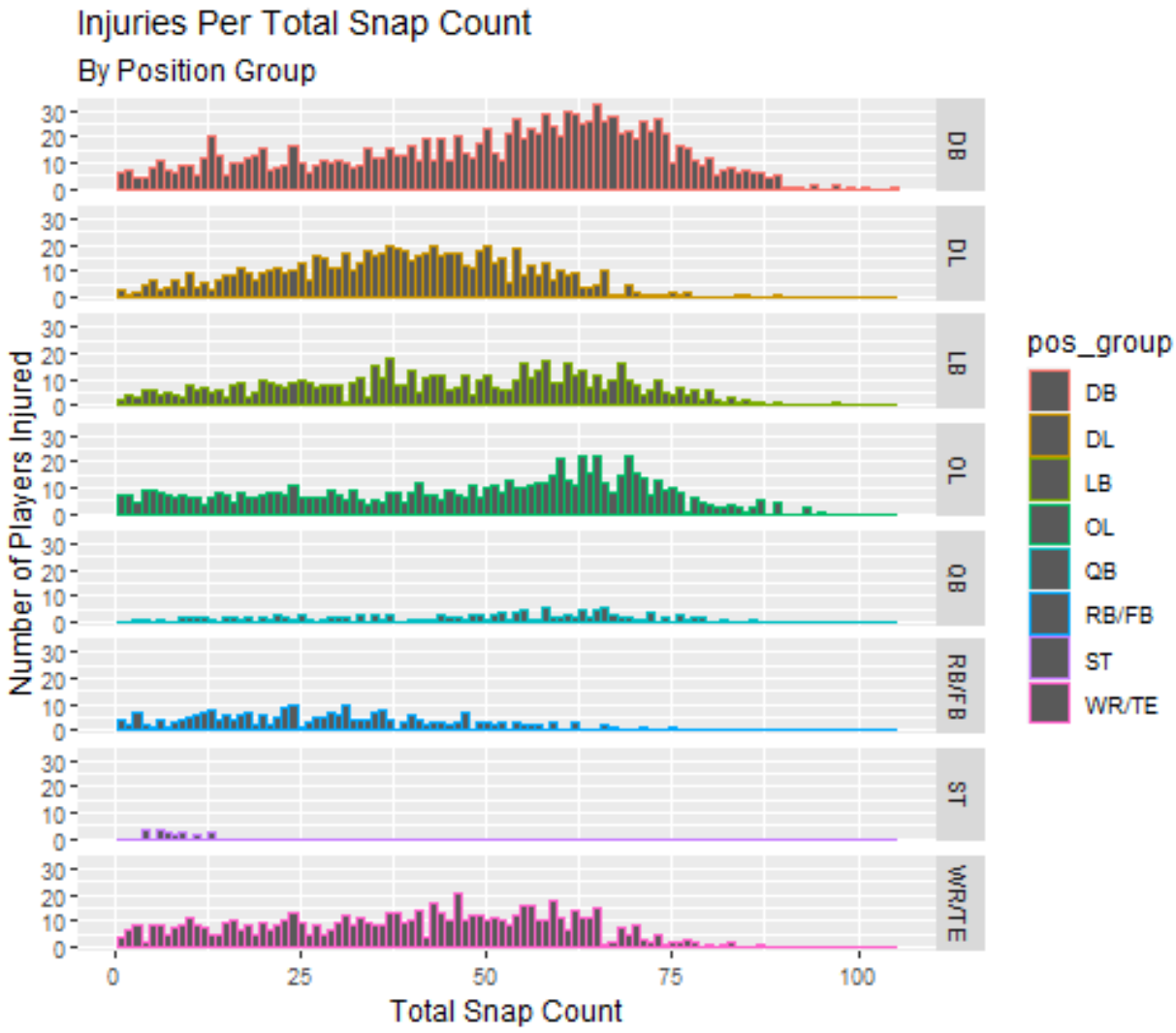
Next we'll explore whether or not the volume of player usage could have an impact on injuries, using player snap counts as proxies.

Now we'll start an EDA based on a different type of data. We looked at the total number of injuries from a wide range of snap counts ranging from a player getting injuries after participating in just a few plays to a player participating in the entire game only to be injured at the very end.

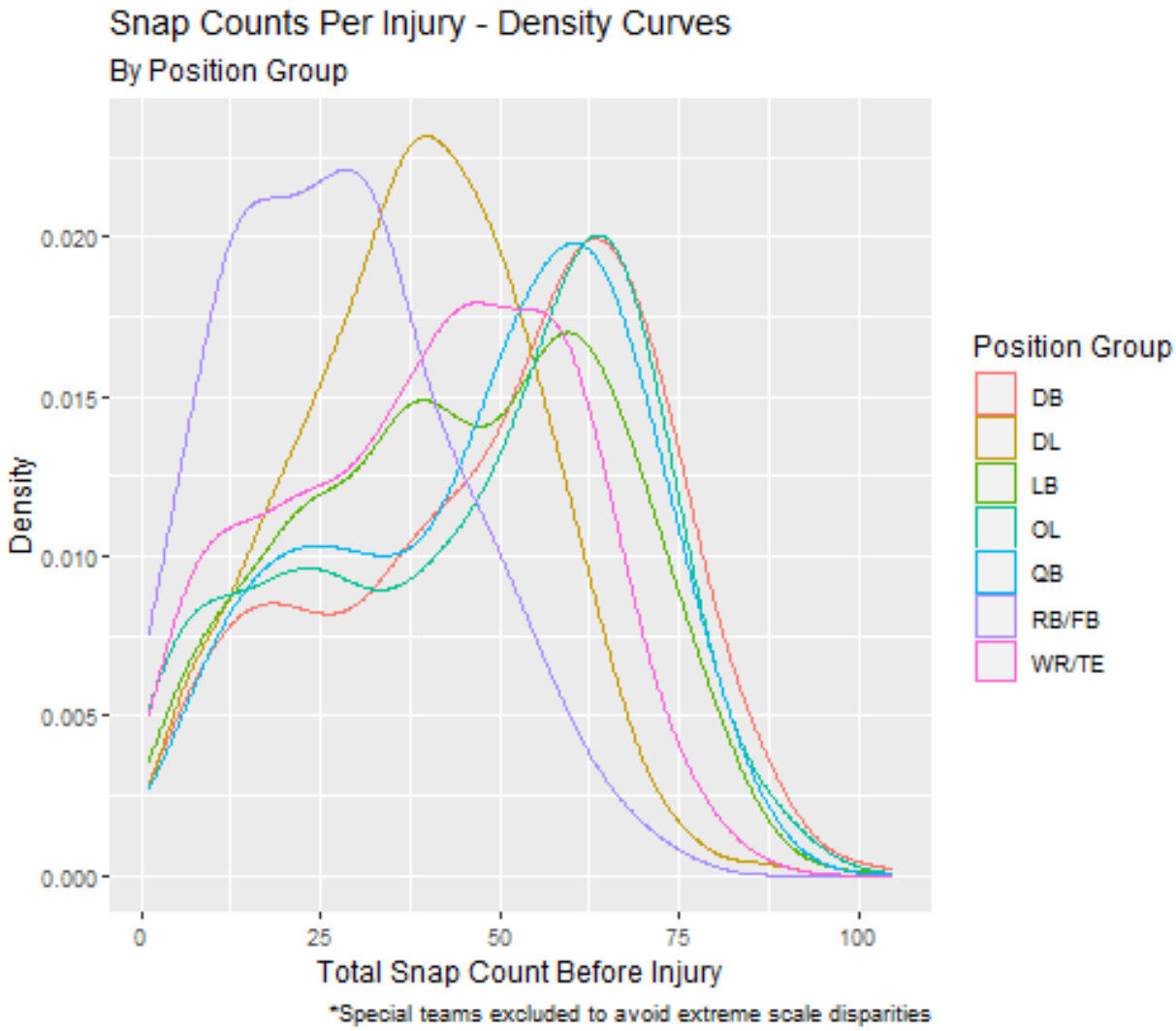


We have the injuries per snap count further broken out by position group. You can kind of see which groups suffer the most at certain snap count ranges, but its not entirely clear yet. What is clear is that there is a trend of more injuries occurring the more plays the player is involved in, which makes sense.

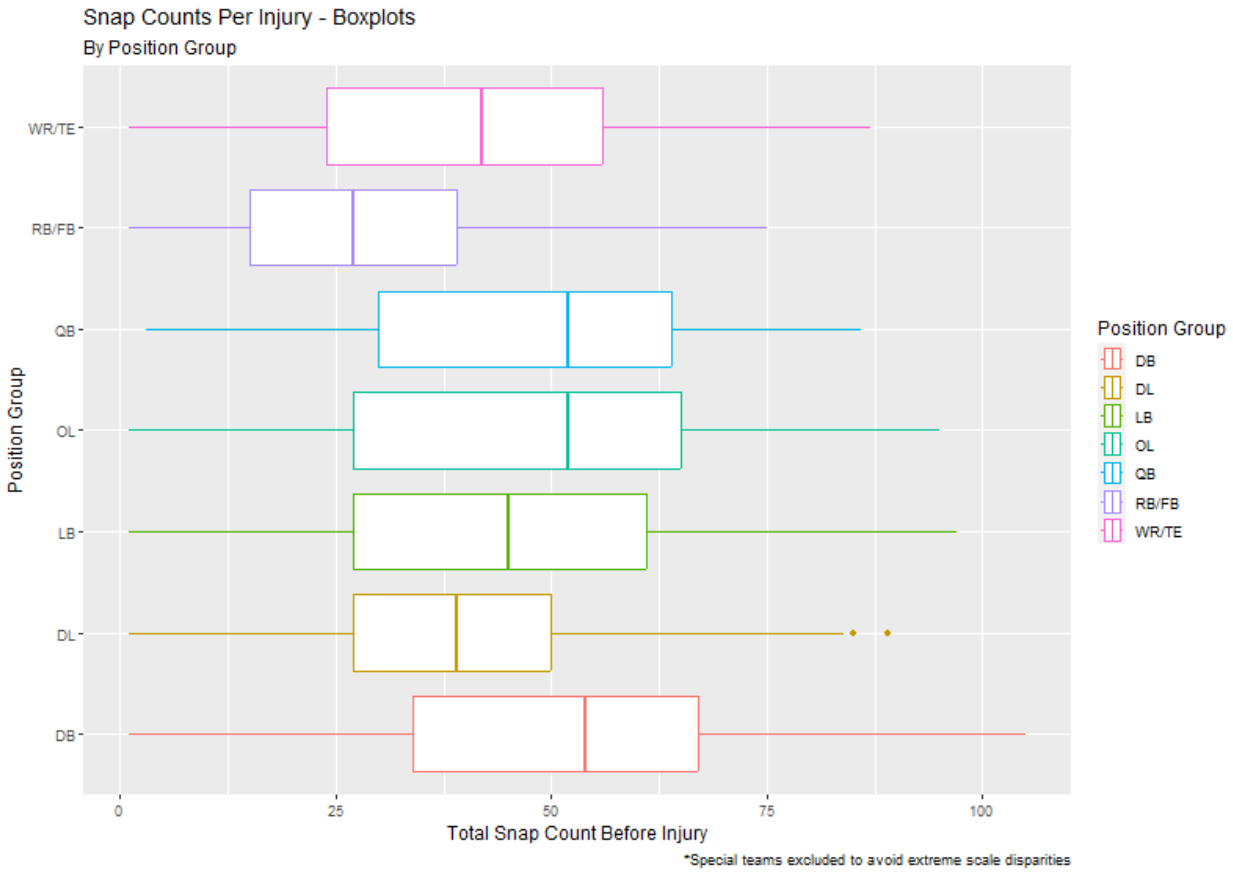
Let’s take a closer look at the same graphic, but now broken out individually by position group so that we can see a little more clearly.



Here we can see a much clearer picture of who is more susceptible to injury at different levels of play participation. Defensive backs and offensive lineman appear to participate in the majority of the game before sustaining an injury, while defensive lineman and runningbacks/fullbacks suffer injury toward the middle range of total snap counts. No clear trend can be immediately determined for the other positions, they seem to be more spread out. Let's make a better comparison by overlapping their densities.



That is better, but it's still difficult to discern the average amount of snaps per each group. Let's add a boxplot visual to see the summary statistics better and where the average lies for each group.



Now we can clearly see which position groups are affected the most by volume of play participation. Special teams players such as punters and kickers have been excluded due to being outliers. They all tend to have very few snaps before an injury which should not be surprising since they participate in very few snaps overall.

We see that running backs/fullbacks tend to get injured with many fewer plays under their belt than other position players. A strong conjecture we could make as to why this is would be that running backs/fullbacks are head on into the play more often than other positions. A defensive back may go half the game or longer without getting near the ball (positive correlation to their skill level). Therefore, we wouldn't expect defensive backs to get hurt early on with few snaps like we would a running back/fullback.

Similarly, we see that defensive lineman appear to get injured with many fewer snaps relative to the other position groups. A conjecture for this could be that they are the biggest players on the defensive side of the ball dealing with the biggest players on the offensive side of the ball. They run into someone hard on just about every single play and if all goes well, they end up on the ground in a pile with the ball at the bottom of that pile. You can imagine how multiple heavy players hitting and falling on one another

can lead to more injuries sooner than other position groups, save for running backs/fullbacks. This may beg the question as to why offensive lineman, who are involved in many of the same impacts as the defensive lineman, don't seem to suffer injury quite as early as their defensive counterparts. While they are still close, they're far enough apart to raise wonder. A conjecture for this would be that they have a lot less randomness in where they start and where they finish in a play. They know where the ball is going and more or less stick to their same area every single play. This gives them familiarity and knowledge of what to expect. Defensive lineman on the other hand have to switch up where they go, what gaps they hit, guess where the ball is going, and deal with maneuvers by the offense they don't know are coming. This element of randomness could explain the gap in snap counts per injury with their offensive counterparts.

Hopefully you enjoyed our data exploration! Im sure you are dying to know who that 1 injury was on a QB Kneel. It was October 22, 2023. Washington's #94 Daron Payne; Defensive Tackle. Ironic his last name is Payne…

Methodology

RQ1 - What type of factors most heavily influence player injuries in the National Football League?

Data Organization

There are many factors that can lead to an injury in the National Football League. We are attempting to decide the leading factors thatcontribute to these injuries in games. We are using a combination of two datasets to get a play-by-play description of each play, who was injured, and what position they play. To prepare our data we first had to determine all the columns that would most likely be useful for our research. The data set had 372 different columns in which we found 47 to be potentially useful for our research. These 47 columns were separated as potentially useful but are not necessarily used throughout the analysis. Using the 'play_by_play' dataset, in the 'desc' column, we are able to extract any time a player was injured from 2019-2023. Then merging that dataset with the 'player_posistion19to23' we can get all the information on the player. After removing the unnecessary columns and adding the player information, our dataset is ready for use.

There are several factors that can be looked at for the probability of a player being injured on a given play. However, the key factors being looked at are weather, stadium conditions, turf, player position, and player usage. These categories were most represented in the various literature reviews that were conducted. The set of nearly 247,000 plays was dissected find only the plays that resulted in a player injury. K- means clustering was used to cluster injuries based on position vs play type. The plays were represented by 5 clusters and included; No-plays, passes, runs, punts, and kickoffs. This allowed for the breakdown of injuries for each position based on the play that was taking place. The no play cluster is determined to be a play that happened in real time but was called as a 'no-play' due to stoppage, penalties or other situations.

Modeling Techniques:

Logistic Regression

Coefficients are calculated for each predictor variable to determine how each variable

changes the log-odds of an injury during a play.

Decision Tree

Figure out the tree 'branch' that can determine a predicted class. The decision tree attempts to find variable values d for the data that best split the predicted classes. This determines the most accurate class predictors for each class.

RQ3 – Which positions are the most likely to suffer injury given multiple compounded factors as a game goes on?

Maximum Likelihood Estimation

Our predictions for which positions are most likely to suffer an injury given compounding factors during a game relies on multiple variables that together, contextualize a specific play where the injury happened. These variables give information on when the injury happened during a drive. Key variables give information on things like which down and how many net yards were gained at that time; what the play type was, run or pass; and what the player usage was by using their total snap count at that point in the game as a proxy.

We'll divide up just the injury observations into an 80:20 train/test split to ensure proper cross-validation. We'll run the model on the train data and test it on the test data. Once we have our results, we'll plug them into a confusion matrix and find the accuracy of the model. The confusion matrix will provide us with the sensitivity of the model, which tells how exact the model is at correctly predicting a player position. We'll have eight different sensitivities, one for each position group.

Modeling Techniques:

Naïve Bayes

Takes conditional probabilities for each variable in the target category, conditioned on all the variables we provide the model to train on. Using these conditional probabilities, it gives maximum likelihood estimates to make predictions.

RQ3 – Can the likelihood of an injury occurring be predicted with any reasonable probability?

Based on all the factors that were looked at it is difficult to determine what factors overall feed into injuries. It was hypothesized that turf and weather were going to be major factors in whether an injury occurred or not. However, after our EDA it shows that there is slight difference in these categories. We realized that there may be other factors that maybe are overlooked that contribute to higher levels of injuries. Using a random tree to figure out if we can accurately predict injuries based on factors using random noise. We were able correctly predict non-injury plays with a 99% accuracy while only getting an accuracy score of 67% on injury predictions. The accuracy score was 99% indicating a high rate of correctly predicted outcomes across both classes.

Gradient boosting was used to predict if weather was a major factor in influencing injuries. We evaluated the model's performance using classification metrics such as precision, recall, and F1-score. Although the outcome was a high accuracy for predicting plays that an injury did not occur, it was terrible at predicting plays where an injury would occur. This is likely due to the dataset being so imbalanced. We tried combating this by adjusting class weights, but still no luck. This highlighted one of the many challenges of trying to predict such a relatively rare event in such a largely imbalanced dataset.

Random Forest

Creates an ensemble of decision trees in which each tree is trained with specific random noise. Combines the output of multiple decision trees to form a single result for predicting injuries.

Gradient Boosting:

We chose Gradient Boosting to predict injuries based on weather conditions due to its effectiveness in handling complex datasets.

Naive Bayes

We chose naive bayes because of its ability to take conditional probabilities to determine maximum likelihood of an outcome. This method of analysis is well-suited for finding targeted datapoints that are grouped together based on their similar probabilities.

Logistic Regression

Determines likelihood of a target variable occurring based on log likelihoods. Similar to Naive Bayes but this takes a linear approach of determining a formula that weights each variable's effect on the target variable. This method will help us determine how much impact each variable has on the target variable and also will give us probability percentages of the target variable occurring rather than just a binary 1 or 0.

Analysis

Data from 2019 to 2023 on every NFL play was combined into one dataframe. The description column of each play was used to discern whether or not an injury occurred on the play, forming a new indicator variable column that depicted this result. This dataframe with the new response variable would make up a large portion of our analysis in predicting whether or not a player would be injured on a play, and also the probability of an injury.

An additional question that required further data wrangling was which position groups were most susceptible to injury, and in order to predict the probability of a certain position group being injured, we had to find a way to parse out the position type of each injury. To do this we combined a dataset from 2019 to 2023 that gave the name, position, and snap count of every player involved in every play. Taking the name from the description column in the first dataset and using that in conjunction with the game ID that both datasets had in common, we were able to pair up the position of the injured player along with their snap count total at the time of injury. The snap count was also used in the position prediction analysis.

For RQ1 and RQ2, we used logistic regression, random forest, and naïve bayes modeling techniques. Random forest and logistic regression have a slight edge in how informative they are in these analyses due to their transparency of which variables were most important to the model. Random forest can rank the variables in a very straightforward way while logistic regression can show statistical significance of each variable along with the log likelihood contributed by each. The naïve bayes model added a new flavor of differentiation to the questions with its conditional probability technique. While it is a black box in terms of which variables are more important, it provides a new angle of attack at the prediction that can handle irrelevant variables quite well. By feeding all variables we deem significant, it will produce the best possible model without being affected by variables that don't matter.

For RQ3 we relied on naïve bayes for its ability to find outcomes for many different factors in a response variable. Simple logistic regression didn't work well here due to having more than two levels in our response variable we were trying to predict. A more robust method such as multinomial logistic regression could have been used to predict this, but we decided to focus most of our time on the first two research questions, which we deemed to be more important.

We'll begin our analysis with the results for RQ1 and RQ2:

RQ1 – What factors most heavily influence player injuries in the NFL?

RQ2 – Can the likelihood of an injury occurring be predicted with any reasonable probability?

The independent variables used for RQ1 and RQ2 using **naïve bayes** model are in the following table:

Variable	**Definition**
week	Which week of the season the game was played
yardline_100	What spot on the 100-yd long field the play started on
drive	Number drive of the game for the driving team on offense
down	Which down the play was for the series of downs
goal_to_go	Indicator of whether the play was within ten yards of the opponent end zone
ydstogo	How many yards were needed to gain a first down

yards_gained	How many yards were gained on the play
ydsnet	How many yards total the drive had accumulated up to that play
shotgun	Indicator of whether the offense was in the shotgun formation (QB lined up about five yards off the line of scrimmage)
no_huddle	Indicator of whether the offense went into a huddle or not before the play (no huddle means the offense is playing faster)
air_yards	An indicator of how many yards the ball was in the air regardless of whether it was caught or not
yards_after_catch	The amount of yards the player who caught the ball ran before being tackled
score_differential	How many point difference between the offense and defense at the time of the play
solo_tackle	Indicator of whether the player with the ball was tackled by a single player
tackled_for_loss	Indicator of whether the player with the ball was tackled for negative yards from the line of scrimmage
series	Which number series of the game for both teams combined (drives for both teams added up)
play_clock	Amount of seconds on the play clock before the ball is snapped
drive_play_count	Number play of the current offensive drive
temp	Temperature at the time of kickoff
wind	Indicator of whether there was wind or not
turf	Indicator of whether the field surface was turf or natural grass
outdoor	Indicator of whether the game was outdoor or indoor
total_drive_seconds	How many seconds the current offensive drive has being going
pass_middle	Indicator of whether the play was a pass in the middle of the field or near the sidelines
pass_deep	Indicator of whether the play was a deep pass or short/medium
run_middle	Indicator of whether the play was a run down the middle or run near the sidelines

Using the independent variables in the above table, we generated a naïve bayes model using training data, then produced a ROC curve using the model to make predictions on test data. This was done with a 70/30 train/test split.

The independent variables used for RQ1 and RQ2 using the **logistic regression** model are in the following table:

week	Which week of the season the game was played
yardline_100	What spot on the 100-yd long field the play started on
down	Which down the play was for the series of downs
ydstogo	How many yards were needed to gain a first down
yards_gained	How many yards were gained on the play
ydsnet	How many yards total the drive had accumulated up to that play
play_type	Pass/run/special teams play
tot_snap_count	The total amount of snaps the injured player had accumulated up to that play

defense	Indicator variable for whether the player was offense or defense

The above table of independent variables were chosen for the logistic regression model through a hybrid reverse subset selection to determine a statistical significance threshold in allowing variables into the model.

RQ1 – What factors most heavily influence player injuries in the NFL?

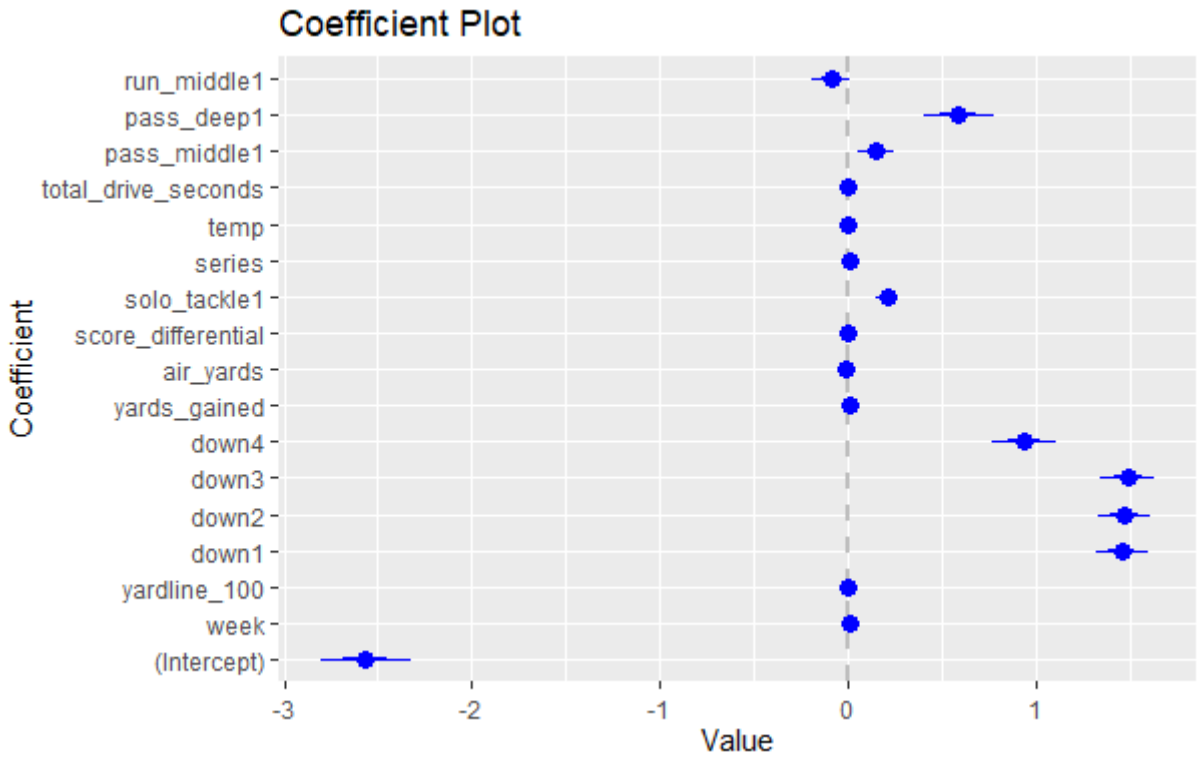
This question was less of a prediction question and more of a complementarity to RQ2. Using the coefficient results from logistic regression, we can see which variables out of our starting variable population were most significant. All of these variables were statistically significant at the 0.1 level or higher, with all but three being at the 0.001 level.

```
Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)         -2.5668155  0.1189380 -21.581  < 2e-16 ***
week                 0.0107450  0.0030104   3.569 0.000358 ***
yardline_100         0.0030619  0.0006640   4.612 4.00e-06 ***
down1                1.4565630  0.0693607  21.000  < 2e-16 ***
down2                1.4683356  0.0701629  20.928  < 2e-16 ***
down3                1.4863067  0.0730891  20.336  < 2e-16 ***
down4                0.9371964  0.0834116  11.236  < 2e-16 ***
yards_gained         0.0120935  0.0020111   6.013 1.82e-09 ***
air_yards           -0.0140751  0.0035290  -3.988 6.65e-05 ***
score_differential  -0.0057014  0.0014354  -3.972 7.13e-05 ***
solo_tackle1         0.2075768  0.0310362   6.688 2.26e-11 ***
series               0.0095527  0.0008955  10.667  < 2e-16 ***
temp                 0.0023324  0.0012189   1.914 0.055674 .
total_drive_seconds  0.0005326  0.0001236   4.308 1.64e-05 ***
pass_middle1         0.1497101  0.0494911   3.025 0.002486 **
pass_deep1           0.5856420  0.0936168   6.256 3.96e-10 ***
run_middle1         -0.0899539  0.0521306  -1.726 0.084429 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

This is valuable information and exciting to see this many variables being this statistically significant, but to get a better idea of which variables out of these are the best, we look at the absolute value of the estimate. The equation we can plug these coefficients into to determine the overall log likelihood for a given observation is:

Log (likely) = -2.5668 + 0.0107(week) + 0.0031(yardline_100) + 1.4566(down1) + 1.4683(down2) + 1.4863(down3) + 0.9372(down4) + 0.0121(yards_gained) – 0.0141(air_yards) – 0.0057(score_differential) + 0.2076(solo_tackle) + 0.0096(series) + 0.0023(temp) + 0.0005(total_drive_seconds) + 0.1497(pass_middle) + 0.5856(pass_deep1) – 0.09(run_middle)

It’s not clear from just looking at these numbers, so we look at the following coefficient chart:



Here we see that deep passing plays, passes over the middle of the field, plays involving one solo tackler, and the down the injury occurred had the largest influence on injuries for the logistic regression model.

Immediate takeaways from this model point to passing plays having the largest effect overall with player injury, especially when taking into account the negative weight of running plays suggesting that these are safer overall. As we've moved into a more pass-heavy era in football, these results could sound an alarm to work on finding more effective safety protocols for passing plays. An interesting expansion of this analysis would be to look further back than just 2019. By looking at years when running plays were more prevalent than passing plays, we'd be able to add additional context to the magnitude of these results.

RQ2 – Can the likelihood of an injury occurring be predicted with any reasonable probability?

For this question, we look at the following table to see how well we can predict injuries.

Method	Sensitivity	Specificity	ROC
Logistic Regression	0.372	0.759	0.635
Naïve Bayes	0.552	0.642	0.65
Random Forest	0.66	1.00	0.94

From these results, we can see that all models performed pretty comparably, and relatively admirably given the complexity of the research questions we examined. The logistic regression model would probably have the longest lasting relevance in its ability to be updated. This is due to its clear weighting and statistical significance for each variable. We know which variables it would behoove researchers to

look further into, as well as any new variables that could be added. New variables could be screened out more easily than in the case with naïve bayes, where we can't see any statistical relationship between individual independent variables, and random forest, where the interconnectedness of each variable is tough to parse out.

The Random Forest Classifier was used to initially train with the default parameters to serve as a baseline. This was crucial for establishing a performance benchmark before trying to tune it up. The Random Forest model was chosen because of its robustness and capability of handling large complex data with little preprocessing steps. Multiple decision trees are created to reduce overfitting and improve the overall prediction accuracy. The results of the model were quite good from the beginning.

	Precision	Recall	F1-Score	Support
0	0.99	1.00	0.99	47759
1	0.99	0.67	0.80	1505

Despite some of the downsides with naïve bayes and random forest, it would still be a good idea to incorporate these and other additional models to continue to attack the questions at hand from every angle.

The 'GridSearchCV' was used for hyperparameter tuning with a focus on the 'n_estimators' and 'max_depth', to find the optimal model configuration. The process systematically varied key parameters of the Random Forest model and evaluated their impact on the model performance. Varying the number of the trees was helpful for obtaining better model performance but was detrimental due to the computational power required to run the operation. It also ran the risk of overfitting with higher values. Max depth was another variable that was used to try and capture more complex patterns but also ran into a risk of overfitting. In the end the 'GridSearchCV' resulted in very similar but worse results. The computation time was also another factor in removing it from being processed again. Accuracy dropped to .66 from .67 and was overall not worth the time invested.

Best Score: 0.9411435671846787				
	Precision	Recall	F1-Score	Support
0	.99	1.00	.99	47759
1	.99	0.66	0.79	1505

The tuning was based on the ROC AUC score which was chosen as it provided a comprehensive measure of the model's ability to decipher whether an injury occurred or not across all possible classifications. Though the 'GridSearchCV' was not useful after first use, it did give some interesting looks into the trees that were being produced by the Random Forest. Thus, allowing us to understand how complex the question and predictions were. One of the main findings from this was that it required a robust model that was capable of learning from a very imbalanced and diverse set of data. It is quite obvious that the player position is the most referenced for the model though it took a engineered feature to give us the end results that were worth noting.

Random Forest Importance Factors		
	Feature	Importance
0	injury_frequency	0.213277
1	position_DB	0.068319
2	pass	0.062914
3	play_type_pass	0.062619
4	position_DL	0.062147
5	first_down	0.059360
6	pass_attempt	0.055217
7	season	0.044257
8	position_OL	0.041927
9	position_LB	0.034099

The selected hyperparameters fully demonstrated the trade-off between computational cost and performance. The Random Forest was chosen for this reason, and it was apparent in the final product. The Random Forest scored better than the Logistic Regression as well as the Naïve Bias models. It is important to note that the Precision was exceptionally high, showing the model's effectiveness in correctly identifying true injury events. Unfortunately, the trade-off is a lower Recall, which indicated the proportion of actual positives that were identified correctly. This was to be expected due to the highly imbalanced nature of the dataset. The model tended to be conservative in predicting injuries, though very accurate when it did. However, the model in turn missed a larger number of true injuries.

Another major indicator to look at is the ROC AUC Score. The model performed exceptionally well with a 0.94 overall. The accuracy may be a bit misleading because of the imbalance in the dataset because of dominance in the majority class. The score remained high which indicates that it is capable of correctly discriminating between injury and non-injury across various thresholds. Its high AUC score also indicates that the model that it is more likely to correctly rank a positive instance than a randomly chosen negative one.

It is clear the Recall value was a focal point for the project. The model had a very good Recall rate compared to all other models that we tried. With a 0.67 representing that an injury was correctly predicted 2 out of 3 times. Though specificity was extremely high, it does not show us much because it is only representing the plays where injuries did not occur.

Finally, we looked at feature importance to reveal which factors were the most critical in determining these scores. The 'injury_frequency' category was engineered within the initial clustering of our data points. It was created by using the player position and play type. These two were mapped together in order to turn the categorical columns into one numerical factor. This seemed to be the deciding factor in enabling the model to work. Creating the variable 'injury_frequency' was a powerful tool in enriching and enhancing the dataset. This was crucial in improving the data performance as well as giving us insights into the model.

Now we'll dig into our final research question, RQ3.

RQ3 – Which positions are the most likely to suffer injury given multiple compounded factors as a game goes on?

This question uses the player position for the injured player as the response variable we are trying to predict. The response variable is broken up into eight categories for each position group. The position groups are defensive line, defensive backs, wide receivers/tight ends, linebackers, offensive line, running back/full back, quarterback, and special teams.

The independent variables used for RQ3 using **naïve bayes** model are in the following table:

week	Which week of the season the game was played
yardline_100	What spot on the 100-yd long field the play started on
down	Which down the play was for the series of downs
ydstogo	How many yards were needed to gain a first down
yards_gained	How many yards were gained on the play
ydsnet	How many yards total the drive had accumulated up to that play
play_type	Pass/run/special teams play
tot_snap_count	The total amount of snaps the injured player had accumulated up to that play
defense	Indicator variable for whether the player was offense or defense

The above variables were chosen using a hybrid reverse selection of subtracting variables from the model that did not seem to add much. This was not, however, a statistically rigorous procedure, it was pretty crude since there's no sure way to tell statistical significance for a given variable with naïve bayes.

The model was trained on training data and then implemented to make predictions on testing data. This was done using an 80/20 train/test split on the original dataset.

Below is a table of how well each variable did in terms of sensitivity and specificity for predicting the player position that was injured given the above independent variables.

	Defensive Back	**Defensive Line**	**Linebackers**	**Offensive Line**	**Quarterback**	**Running back/ Full back**	**Special Teams**	**Wide Receiver/ Tight End**
Sensitivity	0.6654	0.5411	0.0606	0.5793	0	0.102	0.6667	0.5
Specificity	0.7806	0.8767	0.9421	0.8782	1	0.9656	0.9943	0.8984
ROC	0.725							

We can see pretty clearly that this model does a good job of predicting a group not being injured, which isn't quite what we want. The sensitivity, or ability to predict true positives for an injured group, is a little more complicated. We can see the model is pretty good at predicting defensive backs, defensive lineman, offensive lineman, and special teams, but is not so good at the rest. But most importantly, we see a ROC AUC value of 0.725 which is pretty good.

From these results, we can infer that certain positions make a strong case for providing additional rest and mixing up the play types they are involved with. While it's not a clear path to what specifically those prescriptions would be for each group, the results lend themselves to a strong case for additional

research into this question. With more time, we would use more models and dig further into causation using additional variables, specifically variables engineered from our original set.

Data Visualizations

Naïve Bayes

We used the Naïve Bayes method for two different models. The first model we built was used to predict that a player would be injured during a game given various conditions. The second model we built was used to predict which player positions are the most likely to suffer injury given multiple compounded factors as a game goes on.

Since the Naïve Bayes method doesn't have any structured method to evaluate and narrow down variables, an iterative process was performed. We looked for variables that we thought would have a differentiating probability across all the position groups and slowly added variables to get an optimal model.

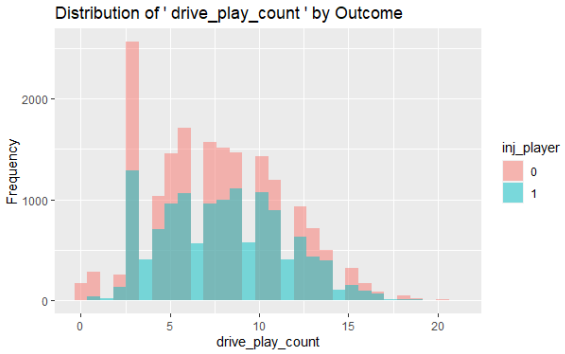
Model 1: Predicting player injury

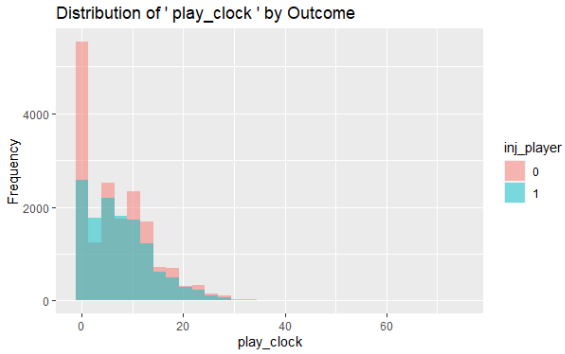
Final variables selected:

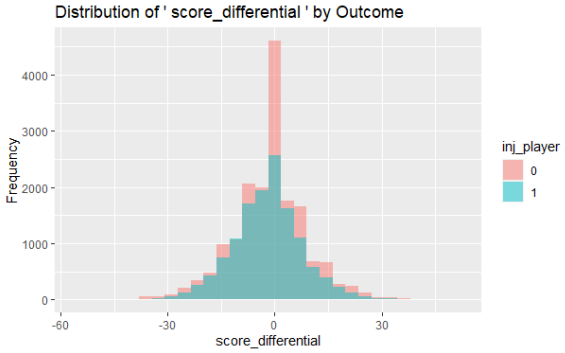
week	Which week of the season the game was played
yardline_100	What spot on the 100-yd long field the play started on
drive	Number drive of the game for the driving team on offense
down	Which down the play was for the series of downs
goal_to_go	Indicator of whether the play was within ten yards of the opponent end zone
ydstogo	How many yards were needed to gain a first down
yards_gained	How many yards were gained on the play
ydsnet	How many yards total the drive had accumulated up to that play
shotgun	Indicator of whether the offense was in the shotgun formation (QB lined up about five yards off the line of scrimmage)
no_huddle	Indicator of whether the offense went into a huddle or not before the play (no huddle means the offense is playing faster)
air_yards	An indicator of how many yards the ball was in the air regardless of whether it was caught or not
yards_after_catch	The amount of yards the player who caught the ball ran before being tackled
score_differential	How many point difference between the offense and defense at the time of the play
solo_tackle	Indicator of whether the player with the ball was tackled by a single player
tackled_for_loss	Indicator of whether the player with the ball was tackled for negative yards from the line of scrimmage
series	Which number series of the game for both teams combined (drives for both teams added up)
play_clock	Amount of seconds on the play clock before the ball is snapped
drive_play_count	Number play of the current offensive drive
temp	Temperature at the time of kickoff
wind	Indicator of whether there was wind or not
turf	Indicator of whether the field surface was turf or natural grass
outdoor	Indicator of whether the game was outdoor or indoor

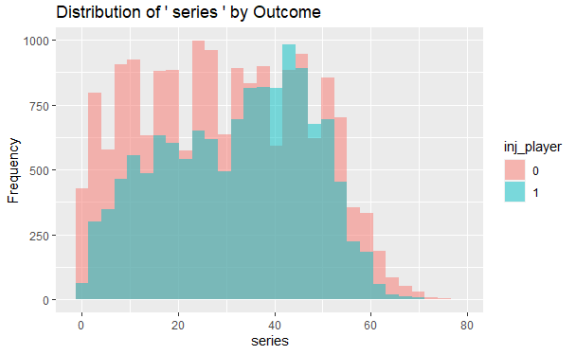
total_drive_seconds	How many seconds the current offensive drive has being going
pass_middle	Indicator of whether the play was a pass in the middle of the field or near the sidelines
pass_deep	Indicator of whether the play was a deep pass or short/medium
run_middle	Indicator of whether the play was a run down the middle or run near the sidelines

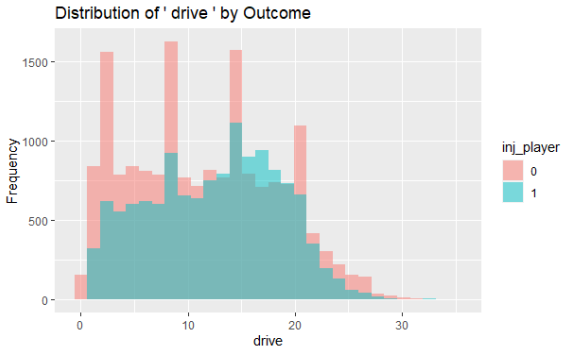
Now lets visualize the distribution of our variables. **Quantitative variables:**

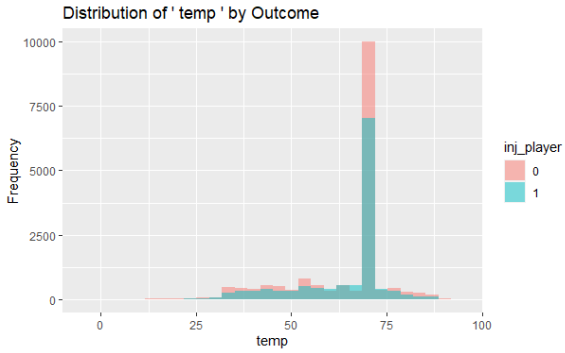


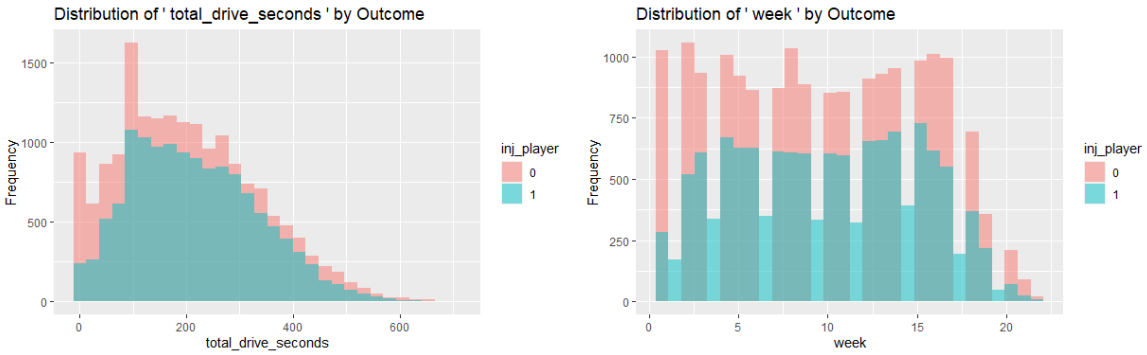


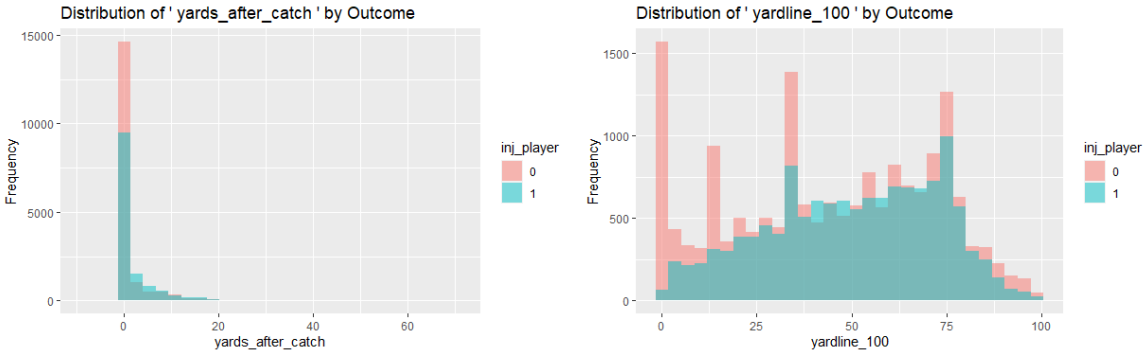


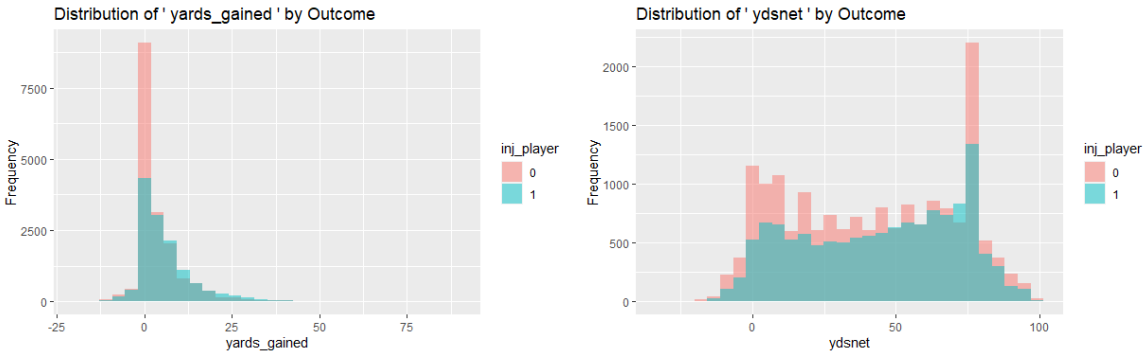


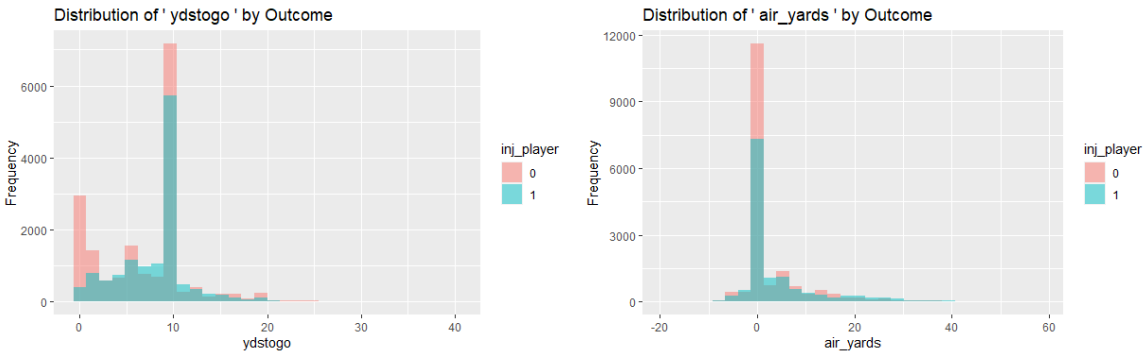




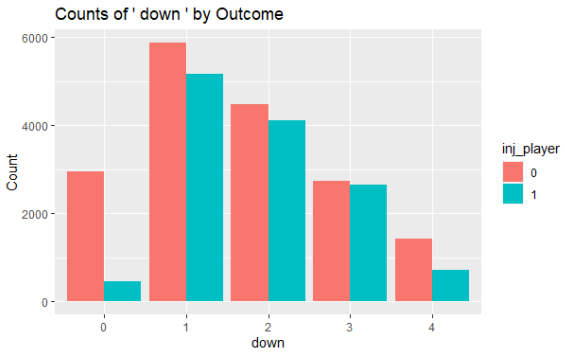


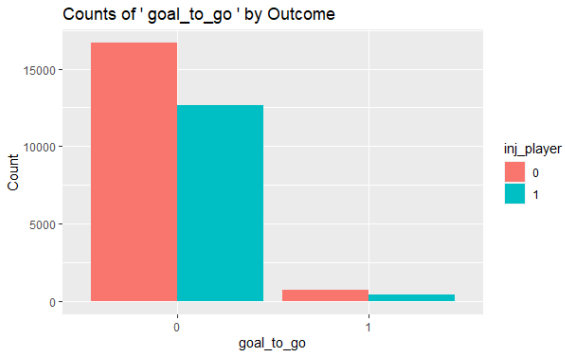


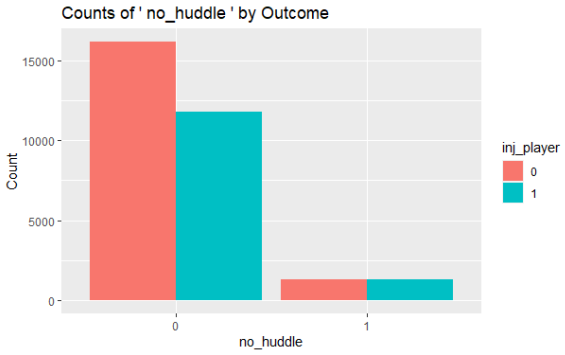


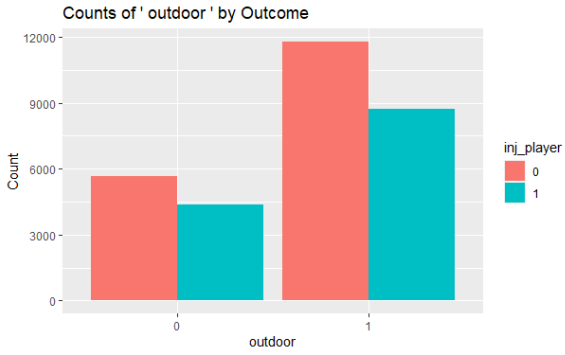


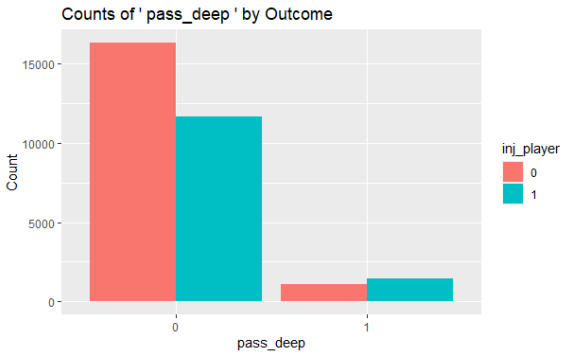
Qualitative variables:

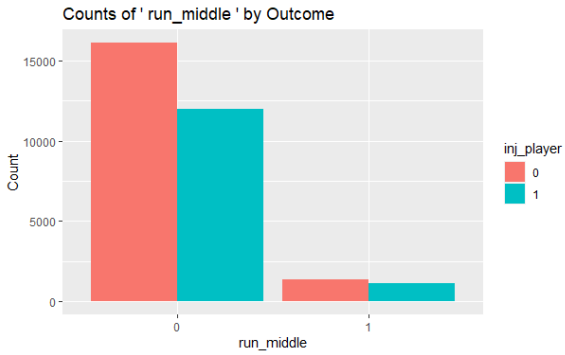

Counts of ' down ' by Outcome
Count
down
inj_player
0
1

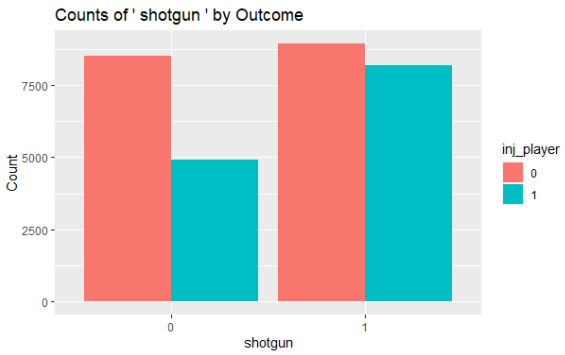

Counts of ' goal_to_go ' by Outcome
Count
goal_to_go
inj_player
0
1

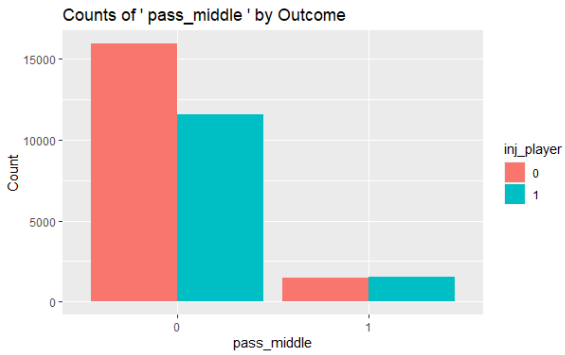

Counts of ' no_huddle ' by Outcome
Count
no_huddle
inj_player
0
1

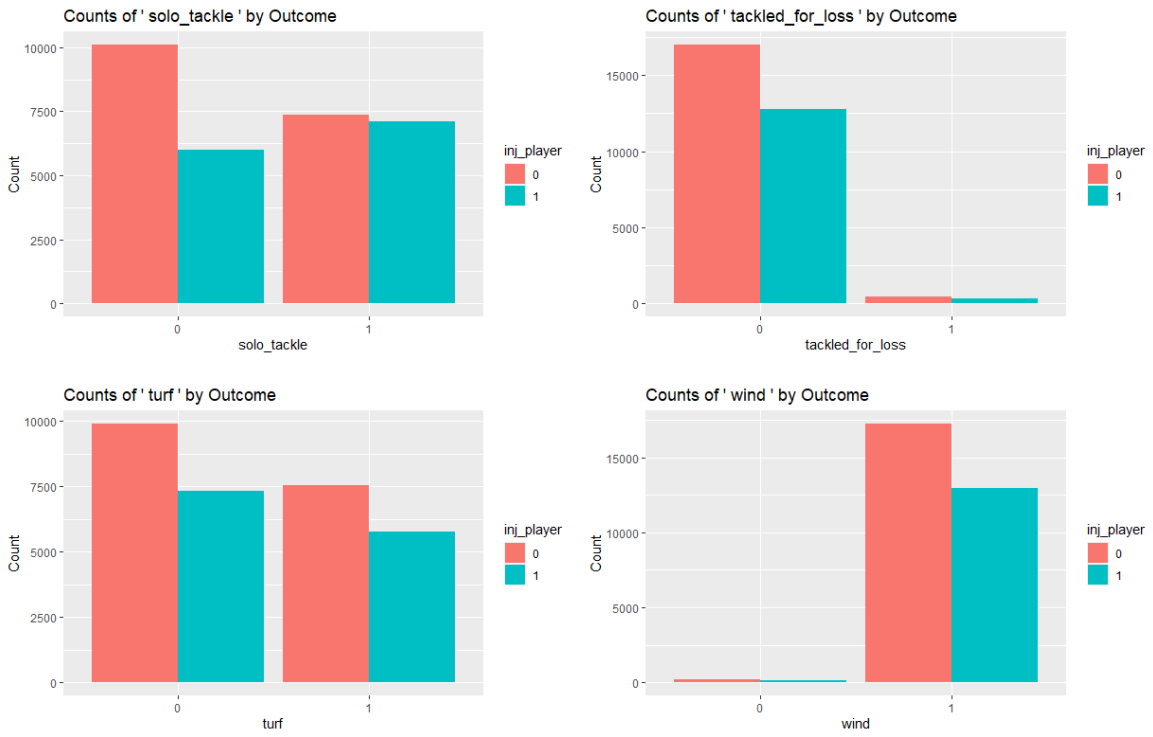

Counts of ' outdoor ' by Outcome
Count
outdoor
inj_player
0
1


Counts of ' pass_deep ' by Outcome
Count
pass_deep
inj_player
0
1


Counts of ' run_middle ' by Outcome
Count
run_middle
inj_player
0
1


Counts of ' shotgun ' by Outcome
Count
shotgun
inj_player
0
1


Counts of ' pass_middle ' by Outcome
Count
pass_middle
inj_player
0
1



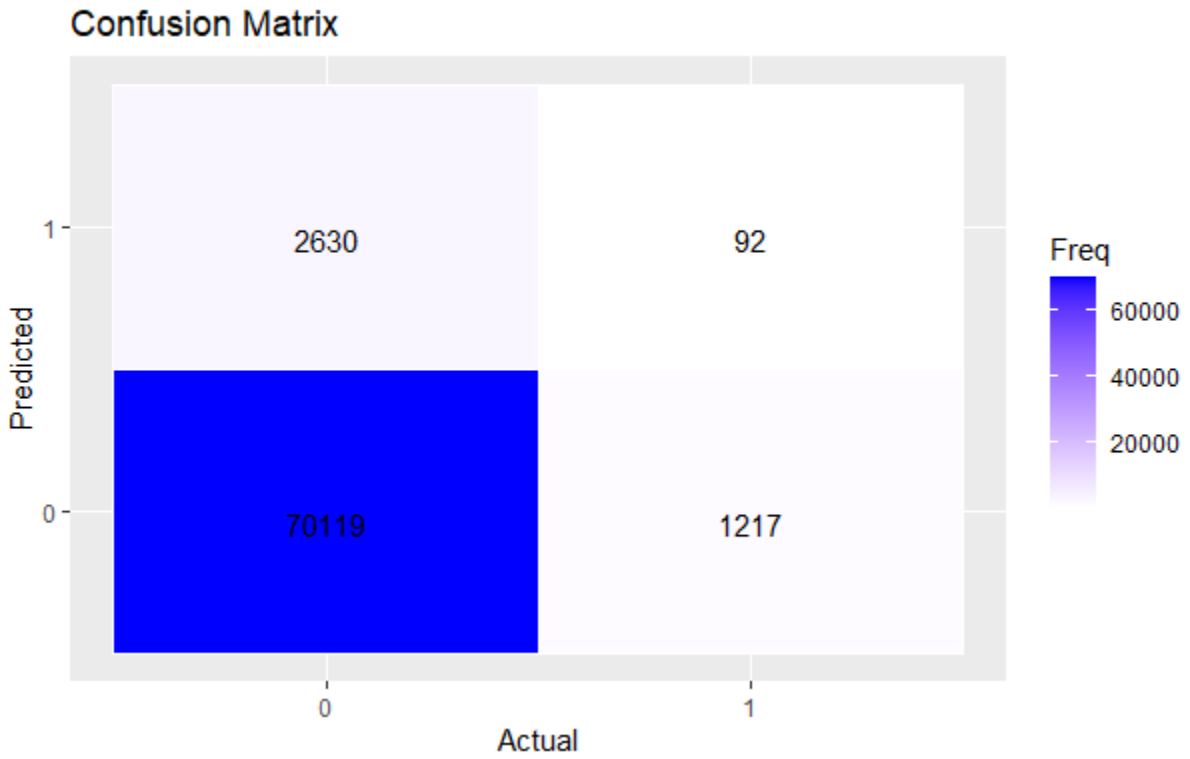
Final results for variables used using **all original observations**:

For our train/test split cross-validation, we created a 70/30 ratio. But because the positive response variable represented a small proportion of the dataset, in order to get a more representative split between the train/test split, we revolved the split around the response variable being split 70/30 among the rest of the data that still represented a 70/30 split overall.

Using the train data to train the data and test data to test the new model, here are our results with all original observations in the dataset:

Statistics Summary: Naïve Bayes	
Accuracy	0.9481
95% CI	0.9464 0.9496
No Information Rate	0.9823
P-Value [Acc > NIR]	1
Kappa	0.0223
Mcnemar's Test P-Value	<2e-16
Sensitivity	0.0703
Specificity	0.9639
Pos Pred Value	0.9829
Neg Pred Value	0.0338
Prevalence	0.9823
Detection Rate	0.9468
Detection Prevalence	0.9633
Balanced Accuracy	0.5171
Positive Class	0

Here we see that our results are pretty abysmal when trying to predict true positives (7%). Our ability to predict true negatives is great (96.4%) but this is to be expected with a dataset this underrepresented of the response variable we are trying to predict. We can attribute the abysmal positive prediction power to the same reason; we just don't have a good representation of our response variable. Let's visualize this aberration in a confusion matrix to get a better feel for the disproportionate representation.



We can see that indeed both the positive prediction value and the representation of the response variable are both abysmal. We'll need to fix this.

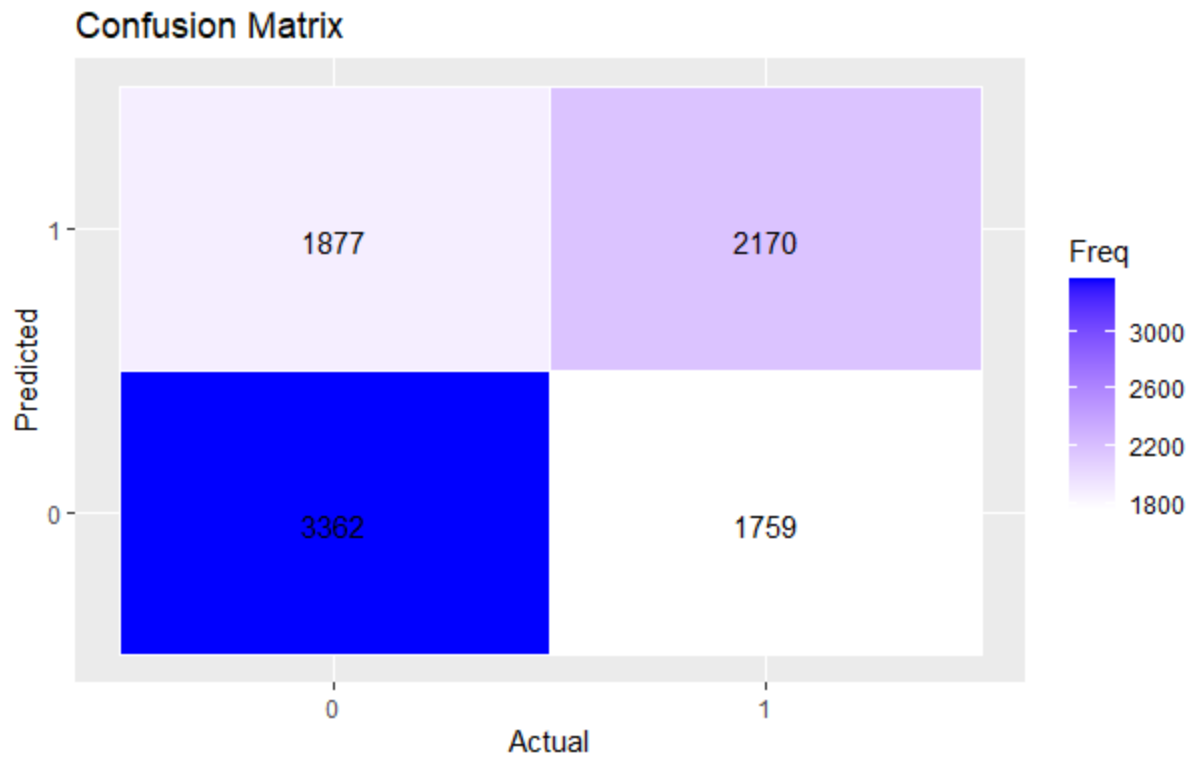
To account for the bad results due to having a low positive response variable observation population relative to the entire dataset, we resampled the data into a new dataset using a technique called the Synthetic Minority Oversampling Technique (SMOTE). What SMOTE does is takes the positive response variable observations and oversamples them into a new dataset, then takes the negative response variable observations and undersamples all of those to create a more equal proportioned dataset.

For the new SMOTE dataset, we first did the same train/test split as described above, and we set the sampling of the response variable to be twice the original amount, and we used the average of the five nearest neighbors to create the duplicate values. The combination of undersampling the negative response variable value observations and oversampling the positive response variable value observations resulted in a new dataset with a little over 30,000 observations.

Final results for variables used using **newly generated SMOTE dataset**:

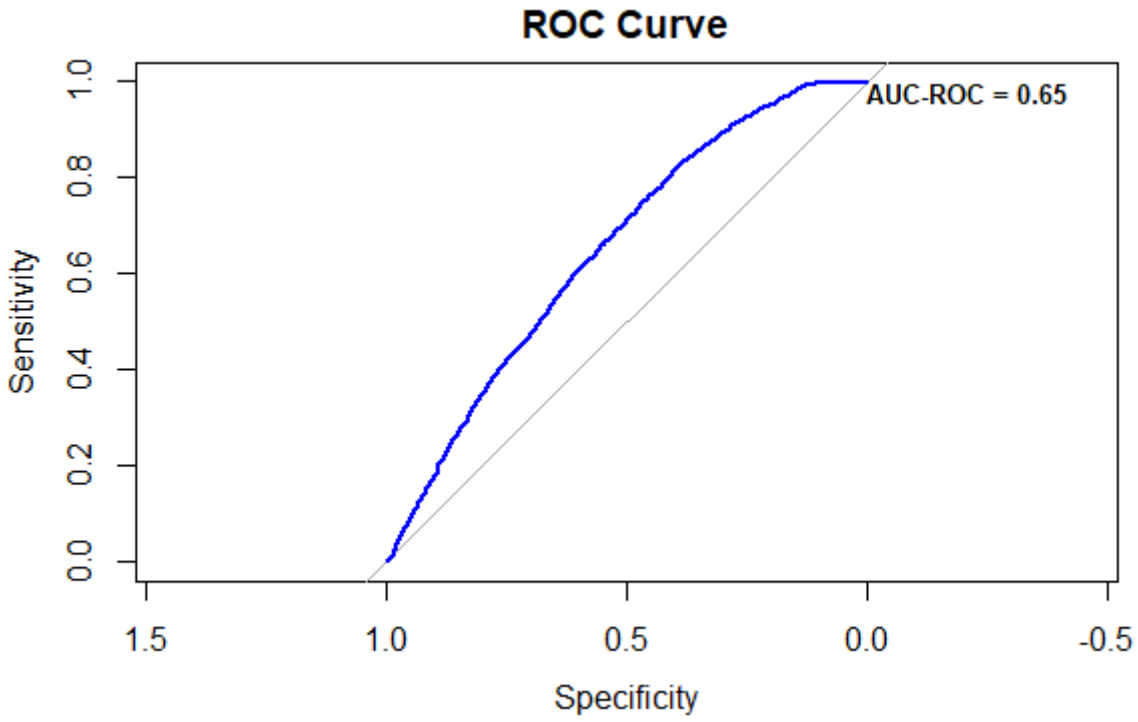
Statistics Summary: SMOTE	
Accuracy	0.6034
95% CI	0.5933 0.6134
No Information Rate	0.5714
P-Value [Acc > NIR]	2.93E-10
Kappa	0.1933
Mcnemar's Test P-Value	0.05234
Sensitivity	0.5523
Specificity	0.6417
Pos Pred Value	0.6565
Neg Pred Value	0.5362
Prevalence	0.5714
Detection Rate	0.3667
Detection Prevalence	0.5586
Balanced Accuracy	0.5970
Positive Class	0

We can see a very good p-value and an overall good prediction score for both a true positive (55.2%) and true negative (64.2%). This result is much better than the previous result using the original dataset. Let's visualize the results with the confusion matrix to get a better feel for the prediction power.



Again, it is clear that we have a more realistic model that has proper proportions resampled using accepted statistical techniques.

Now lets take a look at the ROC curve for this model. The ROC Curve gives a read of discriminatory power of the model between sensitivity and specificity. A model with no prediction power is the gray vertical line that represents a value of 0.5. The closer the curve is to the top left, the closer the value is to 1 and the better the discriminatory power of the model.



Here we see that the prediction power overall taking the actual probabilities into account shows an even better 65% overall result. This seems to be a relatively accurate model.

Model 2: Predicting likelihood of position being injured given player usage and multiple compounding in-game factors.

Final variables selected:

week	Which week of the season the game was played
yardline_100	What spot on the 100-yd long field the play started on
down	Which down the play was for the series of downs
ydstogo	How many yards were needed to gain a first down
yards_gained	How many yards were gained on the play
ydsnet	How many yards total the drive had accumulated up to that play
play_type	Pass/run/special teams play
tot_snap_count	The total amount of snaps the injured player had accumulated up to that play
defense	Indicator variable for whether the player was offense or defense

Final results for the variables we used:

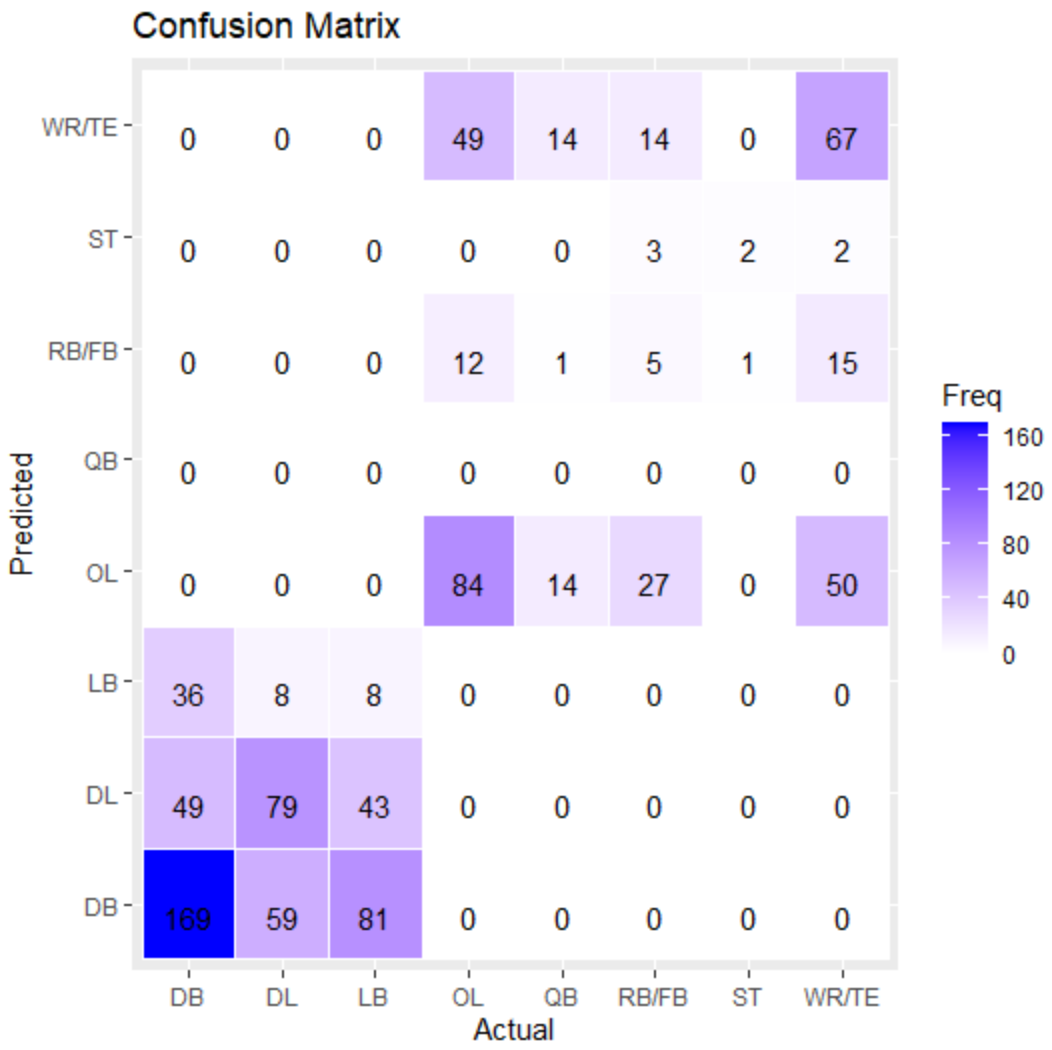
`Statistics`

Statistics Summary: Model 2		
Accuracy:	0.4641	
95% CI:	0.431	0.4975
No Information Rate:	0.2848	
P-Value [Acc > NIR]:	<2.2e-16	
Kappa:	0.3327	
Mcnemar's Test P-Value:	NA	

STATISTICS BY POSITION TYPE								
Statistic	DB	DL	LB	OL	QB	RB/FB	ST	WR/TE
Sensitivity	0.665	0.541	0.061	0.579	0	0.102	0.667	0.5
Specificity	0.781	0.877	0.942	0.878	1	0.966	0.994	0.898
Pos Pred Value	0.547	0.462	0.154	0.480	NaN	0.147	0.286	0.465
Neg Pred Value	0.854	0.907	0.852	0.915	0.967	0.949	0.999	0.910
Prevalence	0.285	0.164	0.148	0.163	0.033	0.055	0.003	0.150
Detection Rate	0.190	0.089	0.009	0.094	0	0.006	0.002	0.075
Detection Prevalence	0.346	0.192	0.058	0.196	0	0.038	0.008	0.161
Balanced Accuracy	0.723	0.709	0.501	0.729	0.5	0.534	0.831	0.699

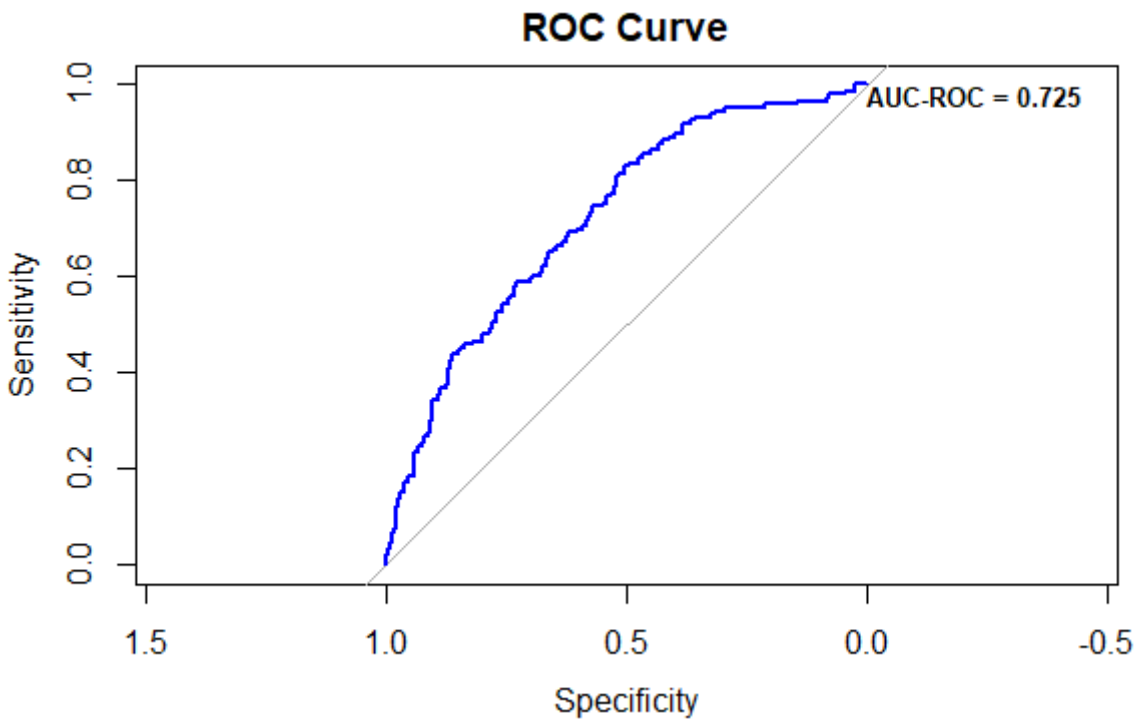
We can see a really good P-Value and the overall accuracy of 0.4641. The accuracy doesn't seem promising on the surface but when looking at the sensitivity for each variables, we can see that some of the classes are predicted fairly well. Classes like defensive backs, defensive line, offensive line, special teams, and wide receiver/tight ends are all above fifty percent. Defensive backs and special teams are particularly good at above sixty percent, but we should be cautious about the special teams as there are few data points and the scenario for a special teams player to get injured are very limited as is.

Let's once again visualize the results with the confusion matrix to get a better feel for the prediction power.



Looking at the confusion matrix, we can see more clearly not only how many players were accurately and inaccurately predicted, but also the volume of each is given to better assess the prediction power. The results for defensive backs look even stronger seeing the sample size and the correct vs incorrect predictions. The offensive and defensive line look particularly good as well with the volume available for those.

Now for a final evaluation which is arguably the most informative, we'll take a look at the ROC curve.



Here we see a prediction power of 0.725, which is not bad at all and certainly better than the initial 0.4641 result we observed initially.

Logistic Regression:

The next method we use to determine the same question of player injury prediction as the first Naïve Bayes model is logistic regression. Logistic regression calculates log-likelihood values of the response variable occurring with different weighted magnitudes for each variable it uses to make predictions. These log likelihoods are aggregated and used in conjunction with one another to output percentage probabilities of the response variable occurring. To get an absolute prediction of yes or no, we round up these values from .5 – 1 as a yes and everything else as a no.

For this method, we used the same train/test split as described in the Naïve Bayes models that focused on a specifically proportioned split on the response variable since it is significantly underrepresented in the dataset. We also ended up using the same SMOTE dataset generated for the Naïve Bayes model. The original data was ran first and is not even worth posting since it could not converge on a result for the same reason as Naïve Bayes.

Here are the results using all variables that were used for the first Naïve Bayes model.

Original results:

Model: 'glm(formula = inj_player ~ ., family = binomial, data = traindata)'				
Coefficients:	**Estimate**	**Std. Error**	**z value**	**Pr(> |z|)**
(Intercept)	-2.7165	0.1911275	-14.213	<2e-16 ***
week	0.0111	0.0030706	3.618	0.000297 ***
yardline_100	0.0029	0.0007712	3.804	0.000142 ***
drive	0.0014	0.0067346	0.213	0.831046
down1	1.5250	0.0836509	18.23	<2e-16 ***
down2	1.5383	0.0813862	18.901	<2e-16 ***
down3	1.5643	0.0833933	18.758	<2e-16 ***
down4	0.9842	0.0898462	10.955	<2e-16 ***
goal_to_go	-0.0454	0.0820059	-0.554	0.579727
ydstogo	-0.0043	0.004292	-1.006	0.31442
yards_gained	0.0142	0.0026604	5.329	9.86e-08 ***
ydsnet	0.0006	0.0008599	0.683	0.494596
shotgun1	-0.0429	0.0353378	-1.214	0.224668
no_huddle1	0.0406	0.0556015	0.731	0.464992
air_yards	-0.0146	0.0036529	-3.991	6.57e-05 ***
yards_after_catch	-0.0066	0.0042527	-1.553	0.120411
score_differential	-0.0062	0.0014904	-4.138	3.51e-05 ***
solo_tackle1	0.2078	0.0313181	6.634	3.26e-11 ***
tackled_for_loss1	0.0375	0.0918728	0.409	0.682877
series	0.0092	0.0026837	3.443	0.000575 ***
play_clock	0.0007	0.002436	0.274	0.783931
drive_play_count	-0.0129	0.008681	-1.489	0.13657
temp	0.0024	0.0013237	1.808	0.070619
wind1	0.1402	0.1312373	1.068	0.285363
turf1	0.0379	0.0329945	1.149	0.250691
outdoor1	0.0056	0.0376027	0.15	0.880644
total_drive_seconds	0.0008	0.0002114	3.563	0.000367
pass_middle1	0.1692	0.0500933	3.378	0.000730 ***
pass_deep1	0.5876	0.5875932	0.0942414	4.52e-10 ***
run_middle1	-0.1113	0.0531345	-2.094	0.03623 *
				Signif Codes: 0: '***' | 0.01: '*'
Null deviance:	29229 on 21393 degrees of freedom			
Residual deviance:	27622 on 21364 degrees of freedom			
AIC:	27682			
# of Fisher Scoring Iterations:	4			

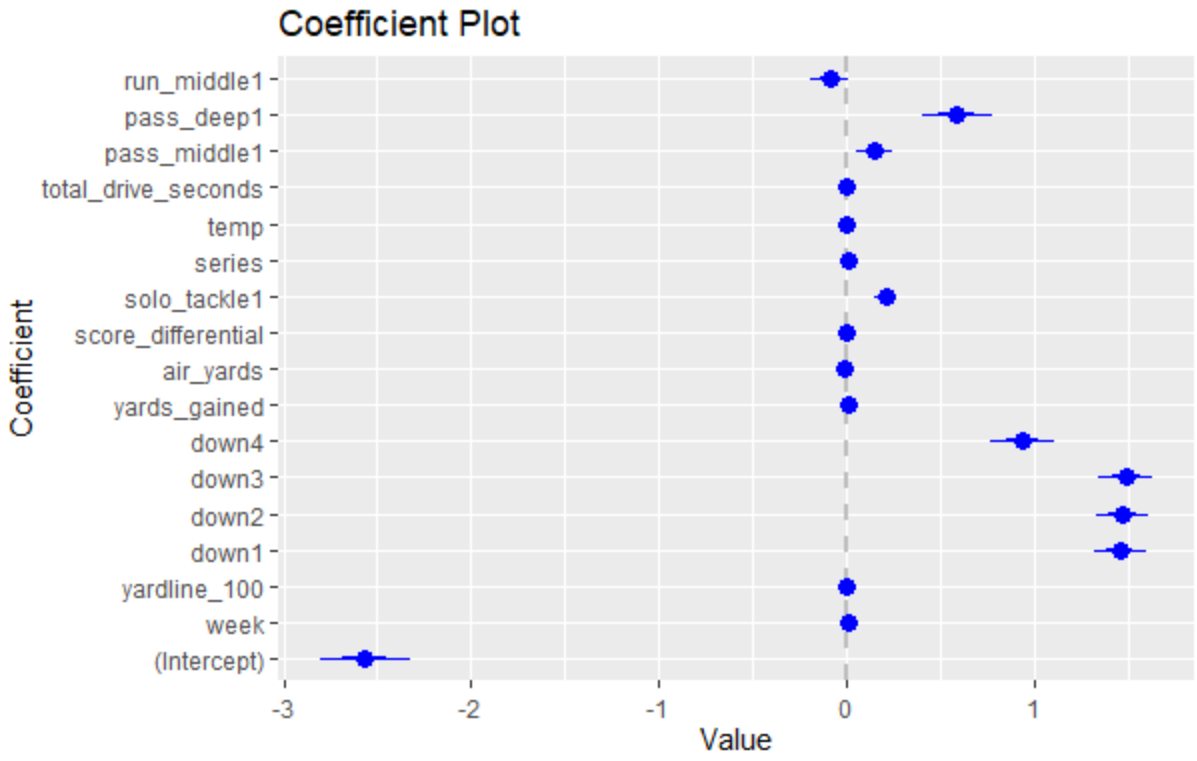
These results show a lot of variables that are insignificant at statistically accepted significance levels. Next we run the model again, but this time only with variables that meet the 0.1 or higher threshold for statistical significance.

Filtered results:

Model: 'glm(formula = inj_player ~ ., family = binomial, data = traindata)'				
Coefficients:	**Estimate**	**Std. Error**	**z value**	**Pr(> |z|)**
(Intercept)	-2.5668	0.118938	-21.581	<2e-16 ***
week	0.0107	0.0030104	3.569	0.000358 ***
yardline_100	0.0031	0.000664	4.612	4.00e-06 ***
down1	1.4566	0.0693607	21	<2e-16 ***
down2	1.4683	0.0701629	20.928	<2e-16 ***
down3	1.4863	0.0730891	20.336	<2e-16 ***
down4	0.9372	0.0834116	11.236	<2e-16 ***
yards_gained	0.0121	0.0020111	6.013	1.82e-09 ***
air_yards	-0.0141	0.003529	-3.988	6.65e-05 ***
score_differential	-0.0057	0.0014354	-3.972	7.13e-05 ***
solo_tackle1	0.2076	0.0310362	6.688	2.26e-11 ***
series	0.0096	0.0008955	10.667	<2e-16 ***
temp	0.0023	0.0012189	1.914	0.055674
total_drive_seconds	0.0005	0.0001236	4.308	1.64e-05 ***
pass_middle1	0.1497	0.0494911	3.025	0.002486 **
pass_deep1	0.5856	0.0936168	6.256	3.96e-10 ***
run_middle1	-0.0900	0.0521306	-1.726	0.084429
				Signif Codes: 0:'***' | 0.001:'**'
Null deviance:	29220 on 21393 degrees of freedom			
Residual deviance:	27634 on 21377 degrees of freedom			
AIC:	27682			
# of Fisher Scoring Iterations:	4			

We can see that the leftover variables we use here are all statistically significant to at least the 0.1 level, and most are above 0.05. They are all very impressively significant. To get a better feel visually for which variables have the most weight in calculating log-likelihood, lets plot the coefficients:

We can see that solo tackling, deep passing plays, passing plays down the middle of the field, and the down the injury occurs on have the largest effect on whether or not a player is injured. The rest of the variables have a marginal effect, though all have high levels of statistical significance. We can also see the error bound on each point to help better understand the range of log-likelihoods for each variable.



Accuracy:

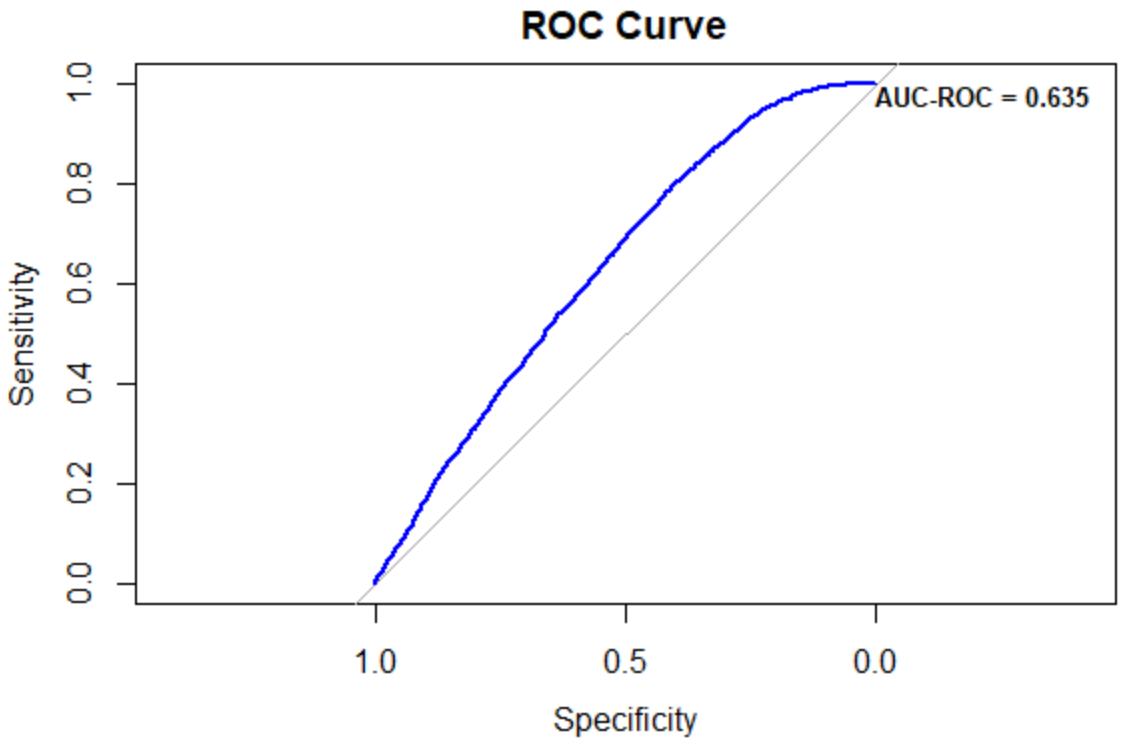
Using this model to make predictions on the test dataset, we observed:

Overall accuracy: 59.3%

Sensitivity: 37.2%

Specificity: 75.9%

This doesn't seem great but also turns out to be not entirely accurate for how we calculated these. Since logistic regression models calculate likelihoods for a response variable occurring, its not a binary yes or no as we talked about above. These are the results for a binary yes or no, but we can get a better feel for the model's accuracy based on its log likelihoods with a ROC Curve.

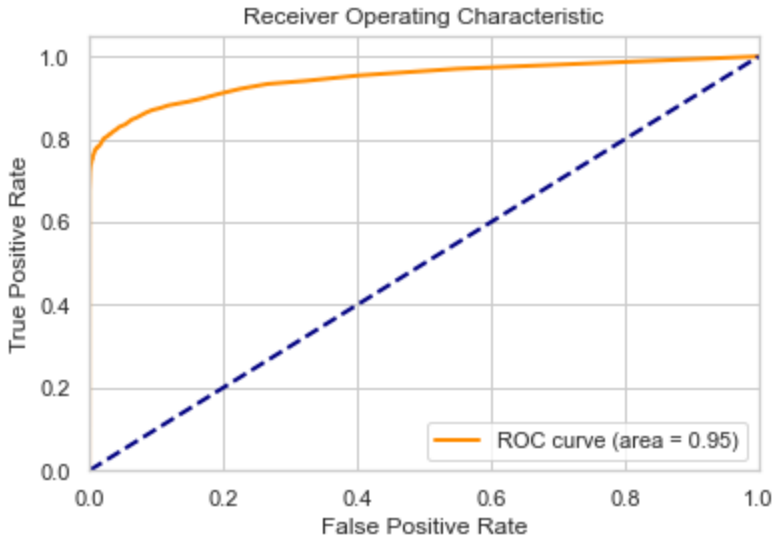


Our model does a fair job with a value of 0.635. Despite the specificity being much higher than the sensitivity, when both are taken into account and given a joint value, we have a pretty fair model.

When looking at the first accuracy percentage breakout, it was easy to jump to the conclusion that this model can't hold a candle to the Naïve Bayes model. But when we look at the results from the ROC Curve, we can see that its actually much better than those initial binary results. An important thing to note is that we can use this model to predict log likelihoods of a player injury which can give more information than just a binary yes or no for whether a player will be injured. You can set tiered percentages for chances of injury and decide appropriate measures to take at each. This is impossible to do with a binary predictor such as Naïve Bayes.

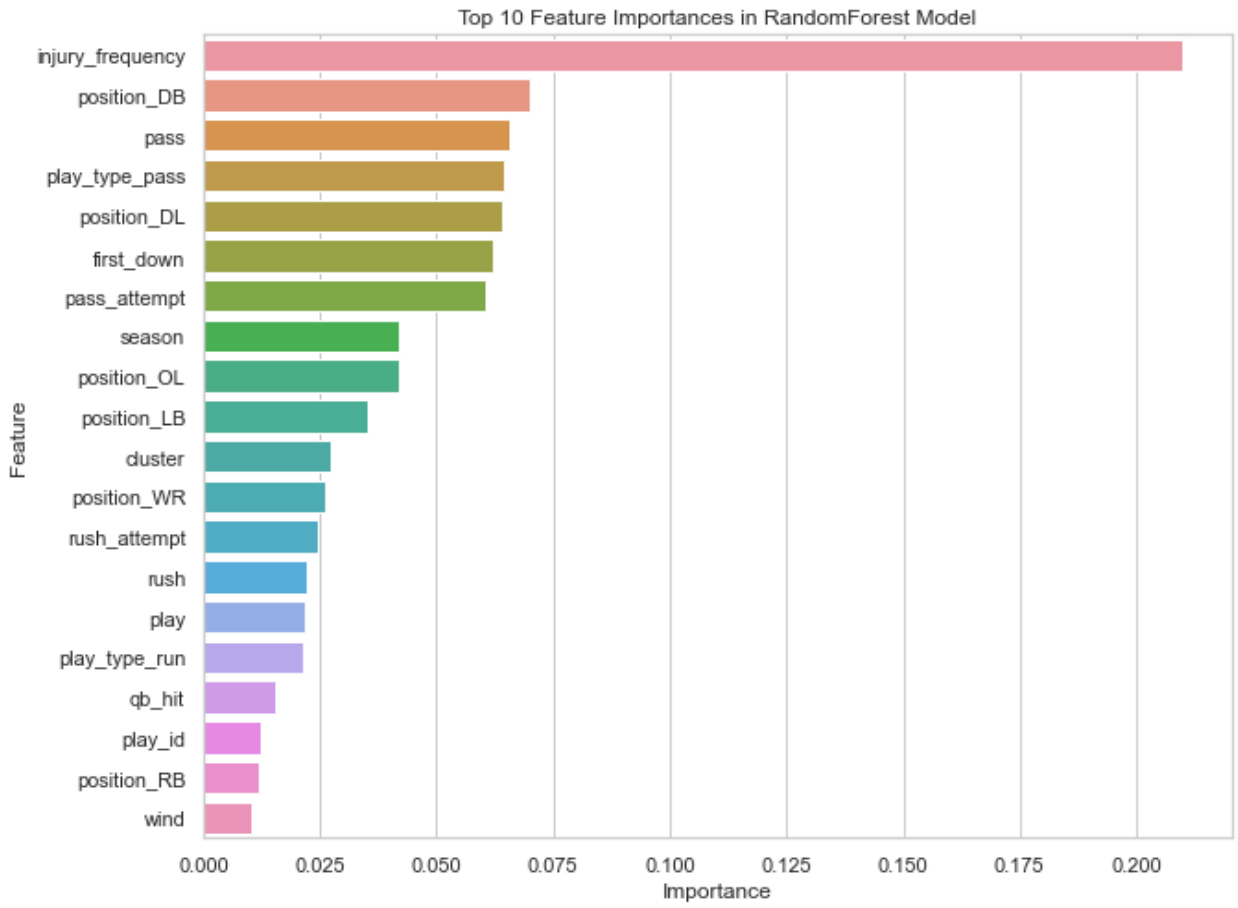
Random Forest:

The model performed exceptionally well in identifying no-injury plays but was less effective in correctly identifying all injury plays. With a 67% recall for injuries overall. A high precision for injury predictions (99%) means that when an injury is predicted, it is very likely to be correct. The model does tend to miss some of the actual injury cases though being at 67%. The F1-score represents the balance between precision and recall. For the non-injury it is 99% but is a lower 80% for the injury class. Though our accuracy is at 99% it can be incredibly misleading due to how imbalanced the dataset is. Therefore, it is more informative to look at the recall, precision, and F1-score for the minority class. The model appears to be quite effective at predictions for both non-injuries and injuries.



	precision	recall	F1-score	support
0	0.99	1.00	0.99	47759
1	0.99	0.67	0.80	1505
accuracy	-	-	0.99	49264
macro avg	0.99	0.83	0.90	49264
weighted avg	0.99	0.99	0.99	49264

Also, due to the massive imbalance in our data set the weighted average is more relevant over the macro average, which treats both classes equally. Overall, the Random Forest method has proven to be an overall good predictor across the board.



To get a further understanding of the importance of each feature being used for predictions we have created the 'importances_df' data frame. This allows us to take the column names used for predictions and see what the model is looking at. The graph above displays the top 20 features used for our Random Forest prediction model and their weights. The 'injury_frequency' feature has the highest importance score, indicating that the historical injury data and trends are crucial in predicting future injuries. Position injuries are spread throughout the list highlighting the role of gameplay dynamics and player roles. Certain positions have a much higher probability of injury due to high-impact plays.

Further attempts to hyper tune, use SMOTE, and gridsearch have been used in order to show that this is the best results we could obtain. Though using both hyper tuning and SMOTE resulted in a higher accuracy of 100%. It is negligible because the predictions for injuries went down with both other methods. Though the drop was only to 66% and 64% it still is below the results of the RandomForest model. Therefore, the RandomForest model is the best model for the predictions of injuries against our dataset.

Ethical Recommendations

The process of exploring what factors cause injuries in the NFL and their potential usage for predicting injuries has several ethical implications. Injury prevention is one of the most important concerns that the NFL works to address. Within our research, there were on average 882 plays per season that resulted in injury. Being able to identify the impact that environmental and other measurable features have on game-time injuries and using that information to predict the likelihood of injury occurrence can help to mitigate potentially avoidable injuries and improve player safety.

The results of our research suggest that the degree to which environmental features such as weather and surface have injury occurrence is not immediately obvious. There are certainly some differences in injury counts when exploring these features, but not to a degree that you could definitively state that these factors are causal. It's a common complaint that players don't like playing on artificial surfaces. While some positions do experience greater injuries on turf, others experience fewer. It appears as if the costs associated with resurfacing turf stadiums would likely outweigh the very small injury mitigation that may or may not even be surface related to begin with. An additional finding of our research suggests that running backs experience injuries earlier in games compared to other positions. This finding largely makes sense in the context of the structure of the game, and any ethical consideration to try and avoid this issue would require sweeping coaching changes and how the position (and the rest of the gameplan) is executed in general.

On the other hand, the treatment and usage of running backs has become quite contentious as of late. Just as they're more likely to be injured earlier in a game, they're also the position that seemingly 'burns out' the fastest and has the shortest career lifespan of any position. "27-year-olds who play other positions are in the primes of their careers. Meanwhile, 27-year-old running backs are being portrayed as fossils hanging on to any hope of a meaningful NFL career for dear life." (Barnwell, 2023). The consensus is that they're largely underpaid for the value that they provide, they're overused while they're young, and are an inexpensive position to replace once the player becomes 'old' and injured (often due to overuse). The results of our research corroborate these claims, and perhaps a mandatory position-specific snap count should be taken into consideration for running back safety.

Usage of positions such as offensive lineman and defensive backs show that injuries tend to occur later in a game at higher snap counts. This could potentially indicate that players may need to be rotated at a certain point in the game to prevent an injury. This may seem like a related argument to that of running backs but with these specific positions, this solution is not as obvious. Running backs are essentially thrown into maximum contact in every play whether that be blocking a defensive lineman much bigger than them or running the ball into the bulk of players on defense that want to put them on the ground. With offensive lineman and defensive backs, they rarely see the volume and nature of contact experienced by running backs. More times than not, it is one on one contact for each group that usually leaves them upright at the end of the play, as opposed to hitting the ground in unpredictable manners.

With the trend of late game usage injuries for offensive lineman and defensive backs, we would make the conjecture backed by related research (Brenner & Watson, 2024) that there are natural factors resulting in these injuries that can be mitigated. In their research on overuse injuries, overtraining, and burnout in young athletes, Brenner & Watson point to over usage of players without proper recovery being a key factor for multiple different injuries. This is largely due to accumulated musculoskeletal damage that isn't being given proper time to repair. Vegso et. al. (2007) conducted a study looking at the

correlation between hours worked and injuries on the job over several different industries and found a strong positive correlation between the two, suggesting that even occupations that aren't nearly as physically demanding as the NFL have other factors that contribute aside from just being hit and intense physical exertion. The factors that seem to cause injuries in these industries are mental attributes such as awareness and judgment. The more tired you are, the less likely you are able to make the same sound decisions as you would fresh and rested.

All of these factors applied to the NFL, negatively correlated with fatigue, are things such as lack of judgment in making safer maneuvers during tackles and evasion of other players, and over usage of their muscles in general that result in taking less contact with each play to cause injury.

This seems to us to be a promising finding that could be mitigated immediately by identifying certain times and conditions in the game when certain player groups should be rotated. With that said, we recognize that there are incentives to not think about these issues deeply due to large tradeoffs faced mutually by players and organizations. If defensive backs and offensive lineman are regularly given rest through rotation, this requires having a deeper talent pool for all members of the positions and thus, requiring more money to be spent. The salary cap limit in the NFL is a large restriction that would cripple teams implementing these player safety measures. They'd have to bench their best players and have gaps with mediocre talent executing the plays. This could not only result in the implementing team having less success with the execution of their plays on offense but could also provide an incentive for other teams to gameplan for these moments of weakness and attack their defense. However, if all teams were required to implement this rotation strategy, it could potentially balance out and just result in higher scoring games that demand another level of strategy, where every team game plans for this new variable. Another solution could be to have a bulk of average talent at these positions, with no elite players, resulting in a smoothed over but average performance. This doesn't seem like a game-winning strategy either. Elite players have the incentive not to support this in order to ensure they retain their value to the team. If they are counted on for more time in a game with consistent play to back it up, they demand a larger salary. A player rotation would force teams to undercut this to some degree. On the other hand, the less talented players would probably support this in order to acquire a portion of that lost salary for themselves due to having more opportunities and a stronger demand for their less talented services. With some of the challenges on the table, we recommend, at the very least, that this issue be researched and studied more than it currently is.

Regarding stadium conditions, it's difficult to come to a conclusion that satisfies everyone. There are an overwhelming number of examples of players who dislike turf, some have even claimed that they won't join a specific team due to the condition of their practice fields because of the potential for career-altering injuries. The data doesn't seem to suggest that surface plays as big of a role as it's made out, however. For example, Detroit Lions quarterback Jared Goff publicly criticized Carolina Panthers' Bank of America Stadium stating, "I thought the field conditions were below NFL level standard, specifically pregame … I don't know what the deal is here, but they need to figure out a way for the turf to not feel like cement."(Salvador, 2022). Following the game, the NFLPA even filed a grievance against the Panthers for the field conditions (WCNC, 2022). What's interesting about this is that our findings suggest that Bank of America Stadium is actually the safest field to play on and hosted the fewest injuries of any venue during the research period by a significant margin. The actual culprit was more than likely the fact that the game took place in 20-degree weather. The ethical dilemma here becomes how to handle this situation. Players are largely dissatisfied with artificial surfaces due to injury potential, but to restate an

earlier claim, there's not much evidence to suggest that the costs associated with resurfacing would actually mitigate a worthwhile number of injuries.

Ultimately, the prevention of injuries is something that the both the NFL and future research should prioritize as an ethical consideration. Trying to define specific characteristics that cause injuries is an arduous task but it's one that absolutely should be further inspected. Solving this problem holds the potential to implement safer playing conditions and help keep players healthy, potentially extend their careers, and preserve their long term health for their life post-retirement.

Challenges

Throughout the project, we encountered several challenges that tested our analytical and problem-solving skills. In the beginning, finding a comprehensive football dataset with a sufficient volume of data to support a robust analysis was proving to be pretty tough. But, Donny came in clutch when he found the package nflfastR.

After we found our dataset, cleaning it proved to be a challenge, specifically the task of extracting specific pieces of information from strings using regular expressions (regex). This process was intricate due to the complexity and variability of the data, necessitating meticulous attention to detail in cleaning and preprocessing to make it suitable for our analyses.

Once we got our data cleaned, we realized we had a highly unbalanced dataset, which significantly impacted the reliability of our predictive models. The models favored the majority class in every situation. In an effort to mitigate this issue, Professor Samara recommended employing the Synthetic Minority Over-sampling Technique (SMOTE). Despite this strategic approach, the unbalanced nature of our dataset continued to haunt us, until Aaron and Jon came out on top with their Random Forest, Naive Bayes, and Logistic Regression models.

Additionally, we navigated through variables we hypothesized would be significant predictors of injuries, such as weather conditions and surface types, only to discover they were dead ends. These factors, which we initially believed would yield crucial insights into injury causation, ultimately did not demonstrate the expected correlations, leading us to reassess our analytical frameworks and focus areas.

Our journey through this project really brought to light the intricate nature of working with data, especially when it's vast, messy, or skewed one way. Our biggest takeaway is that that being adaptable in how we approach our research and staying open to changing our methods is key. It also taught us that there are limitations of the tools and techniques we use for analyzing data. There seems to be a balance between pushing for creative ways to solve problems and understanding that sometimes the data just doesn't play along the way we hope.

Recommendations and Next Steps

If we were to do this analysis again on the same research questions, there are several things we would change. Now knowing which variables were the most relevant, we'd focus on engineering additional variables based on these. For example, knowing that passing plays had a large effect on the probability of player injury, we'd do a separate sub analysis of all position groups during both passing and running plays isolated by themselves. From there, we'd try to create additional indicator variables that pertain to each play type and see if that reduces some of the residual error in the model. The play down the injury occurred on also had a large effect on injury relative to the other variables so another example of something we may do would be to isolate each play and do further exploratory data analysis on all downs. From there, we'd hope to do the same thing with passing plays in trying to engineer more precise indicator variables to reduce residual error in the models.

We would also incorporate more models in general. Our analysis of research question three on predicting the position group of an injured player could have been more robust with additional models, specifically a multinomial logistic regression model. Overall the project was successful in applying various means to predict NFL injuries with significant model accuracy and feature importance analysis. However, due to the high level of feature imbalance there could be some improvements. Continuous model refinement and the use of advanced data preprocessing could potentially enhance our recall. Another area that was lacking was the model interpretability and expanding the dataset with more detailed features.

The complexity and nuances of predicting something as high risk as sports injuries was quite prevalent during this project. Understanding the trade-offs of performance metrics was very time consuming and drove our project in a direction we did not initially choose. Future time would be better used in the preprocessing area to improve injury sensitivity while maintaining the precision and specificity we achieved. More detailed engineered features would allow for more precision and insight calculations. There is a lot more information out there that was not included in our dataset, but was not available due to time constraints. Rule changes, equipment updates and styles of play are some of the things that may give us an insight of what is helping to reduce injuries. Lastly potentially being able to normalize injury data could allow us to better account for the difference in sample sizes.

Expansion of our dataset would be another clear and obvious addition to our analysis. We only covered four years, but we could expand that to two decades for an incredibly representative sample of all the changes that have happened over the years combined into one dataset. We could focus analysis on different years and create temporal indicators that tease out changes that have happened in the NFL for the better and worse in regard to play injury incidence. This methodology could also be useful for identifying how injury trends have changed over the span of two decades, and could potentially be useful in identifying future or emerging trends.

Source Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file_path = '/mnt/data/merged_play_by_play_positions.csv'
data = pd.read_csv(file_path)

first_few_rows = data.head()
data_types = data.dtypes
summary = data.describe(include='all')

data = pd.read_csv(file_path)
data['injury_flag'] = data['my_key_injury'].notna()

total_plays = len(data)
non_injury_plays = len(data)-data['injury_flag'].sum()
injury_plays = data['injury_flag'].sum()

play_counts_actual = pd.DataFrame({
    'Type': ['Plays Without Injury', 'Injury Plays'],
    'Count': [non_injury_plays, injury_plays]
})

plt.figure(figsize=(10, 6))
plt.bar(play_counts_actual['Type'], play_counts_actual['Count'], color=['blue', 'red'])
plt.title('Plays Without Injuries vs With Injuries (2019-2023)', fontsize=16)
plt.ylabel('Number of Plays', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
for i, count in enumerate(play_counts_actual['Count']):
    plt.text(i, count + 1000, count, ha='center', fontsize=12)
plt.tight_layout()
plt.show()
```

```
injury_percentage = (injury_plays / total_plays) * 100

injury_data = data[data['injury_flag'] == 1]
injury_by_position = injury_data['position'].value_counts().head(5)

plt.figure(figsize=(12, 6))
injury_by_position.plot(kind='bar', color='skyblue')
plt.title('Top 5 Positions with Most Injuries', fontsize=16)
plt.ylabel('Number of Injuries', fontsize=14)
plt.xlabel('Position', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=12)
for index, value in enumerate(injury_by_position):
    plt.text(index, value + 30, str(value), ha='center', fontsize=12)
plt.tight_layout()
plt.show()

injury_data['team_abbreviation_corrected'] = injury_data['key12'].str.extract(r'^\d{4}-(\w{2,3})-')
injury_by_team = injury_data['team_abbreviation_corrected'].value_counts().head(5)

plt.figure(figsize=(12, 6))
injury_by_team.plot(kind='bar', color='maroon')
plt.title('Top 5 Teams with Most Injuries', fontsize=16)
plt.ylabel('Number of Injuries', fontsize=14)
plt.xlabel('Team Abbreviation', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=12)
for index, value in enumerate(injury_by_team):
    plt.text(index, value + 10, str(value), ha='center', fontsize=12)
plt.tight_layout()
plt.show()

data['surface_type'] = data['surface'].apply(lambda x: 'Turf' if x != 'Grass' else 'Grass')
```

```
injury_surface_counts_actual = data[data['injury_flag']]['surface_type'].value_counts()

plt.figure(figsize=(10, 6))
injury_surface_counts_actual.plot(kind='bar', color=['green', 'gray'])
plt.title('Number of Injuries by Surface Type', fontsize=16)
plt.ylabel('Number of Injuries', fontsize=14)
plt.xlabel('Surface Type', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=12)
for index, value in enumerate(injury_surface_counts_actual):
    plt.text(index, value + 20, str(value), ha='center', fontsize=12)
plt.tight_layout()
plt.show()

surface_counts = data['surface'].value_counts()
total_plays_surface_counts = data['surface_type'].value_counts()

plt.figure(figsize=(10, 6))
total_plays_surface_counts.plot(kind='bar', color=['green', 'grey'])
plt.title('Total Number of Plays by Surface Type', fontsize=16)
plt.ylabel('Number of Plays', fontsize=14)
plt.xlabel('Surface Type', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=12)
for index, value in enumerate(total_plays_surface_counts):
    plt.text(index, value + 5000, str(value), ha='center', fontsize=12)
plt.tight_layout()
plt.show()

injury_rates = (injury_surface_counts_actual / total_plays_surface_counts).sort_index()

plt.figure(figsize=(12, 6))
injury_rates.plot(kind='line', marker='o', color='darkblue')
plt.title('Injury Rate by Surface Type', fontsize=16)
```

```
plt.ylabel('Injury Rate', fontsize=14)
plt.xlabel('Surface Type', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(True)
for index, value in enumerate(injury_rates):
    plt.text(injury_rates.index[index], value, f"{value:.4f}", ha='center', va='bottom', fontsize=12)
plt.tight_layout()
plt.show()

exposed_games = data[data['roof'].isin(['open', 'outdoors'])]
num_games_exposed = exposed_games['game_id'].nunique()
total_plays_exposed = len(exposed_games)
plays_without_injury_exposed = total_plays_exposed - exposed_games['injury_flag'].sum()
total_injuries_exposed = exposed_games['injury_flag'].sum()

exposed_counts = pd.DataFrame({
    'Type': ['Plays without Injury Exposed', 'Injuries Exposed'],
    'Count': [plays_without_injury_exposed, total_injuries_exposed]
})

plt.figure(figsize=(10, 6))
plt.bar(exposed_counts['Type'], exposed_counts['Count'], color=['blue', 'red'])
plt.title('Plays Without Injuries vs With Injuries When Exposed to Elements', fontsize=16)
plt.ylabel('Count', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
for i, count in enumerate(exposed_counts['Count']):
    plt.text(i, count + 1000, count, ha='center', fontsize=12)
plt.tight_layout()
plt.show()

exposed_games['temp_numeric'] = pd.to_numeric(exposed_games['temp'], errors='coerce')
```

```
exposed_games['temp_range'] = pd.cut(exposed_games['temp_numeric'], bins=[0, 30, 50, 70, 90, float('inf')], labels=['30°F and below', '31-50°F', '51-70°F', '71-90°F', '91°F and above'], right=False)

injuries_by_temp_range = exposed_games[exposed_games['injury_flag']]['temp_range'].value_counts().reindex(['30°F and below', '31-50°F', '51-70°F', '71-90°F', '91°F and above'], fill_value=0)

total_plays_by_temp_range = exposed_games['temp_range'].value_counts().reindex(['30°F and below', '31-50°F', '51-70°F', '71-90°F', '91°F and above'], fill_value=0)

injury_ratio_by_temp_range = injuries_by_temp_range / total_plays_by_temp_range


plt.figure(figsize=(12, 6))

injuries_by_temp_range.plot(kind='bar', color='orange', width=1)

plt.title('Number of Injuries by Temperature Range', fontsize=16)

plt.ylabel('Number of Injuries', fontsize=14)

plt.xlabel('Temperature Range', fontsize=14)

plt.xticks(rotation=45, fontsize=12)

plt.yticks(fontsize=12)

plt.tight_layout()

plt.show()


exposed_games['wind_speed_mph'] = exposed_games['weather'].str.extract(r'(\d+)\s*mph', expand=False).astype(float)

injuries_by_wind_speed = exposed_games.groupby('wind_speed_mph')['injury_flag'].sum()

total_plays_by_wind_speed = exposed_games.groupby('wind_speed_mph')['injury_flag'].count()

injury_rate_by_wind_speed = injuries_by_wind_speed / total_plays_by_wind_speed

injury_rate_by_wind_speed.dropna(inplace=True)


plt.figure(figsize=(12, 6))

plt.scatter(injury_rate_by_wind_speed.index, injury_rate_by_wind_speed, color='purple')

z = np.polyfit(injury_rate_by_wind_speed.index, injury_rate_by_wind_speed, 1)

p = np.poly1d(z)

plt.plot(injury_rate_by_wind_speed.index, p(injury_rate_by_wind_speed.index), color='red')

plt.title('Correlation Between Wind Speed and Injury Rate', fontsize=16)

plt.xlabel('Wind Speed (mph)', fontsize=14)

plt.ylabel('Injury Rate', fontsize=14)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.grid(True)
```

```
plt.tight_layout()
plt.show()

total_plays_by_play_type = exposed_games['play_type'].value_counts()
injuries_by_play_type = exposed_games[exposed_games['injury_flag']]['play_type'].value_counts()
injury_ratio_by_play_type = injuries_by_play_type / total_plays_by_play_type
injury_ratio_by_play_type.dropna(inplace=True)

plt.figure(figsize=(12, 6))
injury_ratio_by_play_type.plot(kind='bar', color='purple')
plt.title('Injury Ratios by Play Type', fontsize=16)
plt.ylabel('Injury Ratio', fontsize=14)
plt.xlabel('Play Type', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```

R Source Code

```
library(dplyr)
library(tidyr)
library(nflfastR)
library(stringr)
library(ggplot2)
library(e1071)
library(caTools)
library(caret)
library(performanceEstimation)
library(pROC)
library(forestplot)
library(coefplot)

#RQ3 Analysis
df = nflreadr::load_pbp(2019:2023)
```

```
df = df[,1:30]


dfInj <- df %>% filter(grepl("was injured", desc))



# Empty container for each desc entry
splitVec = c()

# Empty container for names of injured players
nameVec = c()

# Change period in suffixes to r and omit space between last name and suffix (to be fixed later)
for (i in 1:length(dfInj$desc)) {
  if (grepl("Jr.", dfInj$desc[i])) {
    splitVec = append(splitVec, gsub(" Jr.", "Jrr", dfInj$desc[i]))
  }
  else if (grepl("Sr.", dfInj$desc[i])) {
    splitVec = append(splitVec, gsub(" Sr.", "Srr", dfInj$desc[i]))
  }
  else {
    splitVec = append(splitVec, dfInj$desc[i])
  }
}

# Split every vector into list of words that make up each string entry
for (i in 1:length(splitVec)) {
  splitVec[i] = strsplit(splitVec[[i]], "[.]")
}



# Pair injury vector with last character in previous vector to get first name initial
for (i in splitVec) {
  for (j in 1:length(i)) {
    if (grepl("injured", i[j]) && j != 1){
```

```
    if (str_detect(substr(i[j-1], nchar(i[j-1]), nchar(i[j-1])), "[[:upper:]]")) {
      nameVec = append(nameVec, paste(substr(i[j-1], nchar(i[j-1]), nchar(i[j-1])), i[j]))
    } else {
      nameVec = append(nameVec, paste(substr(i[j-1], nchar(i[j-1])-2, nchar(i[j-1])), i[j]))
    }
   }
  }
}

# Pair first name initial with last name
for (i in 1:length(nameVec)) {
 temp = strsplit(nameVec[i], " ")
 tempcombo = temp[[1]]
 nameVec[i] = paste(tempcombo[1], tempcombo[2])
}

# Detect how many injuries in each play
countVec = c()
for (i in 1:length(splitVec)) {
 injCount = 0
 for (j in splitVec[i]) {
   for (k in j) {
    if (grepl("injured", k)) {
      injCount = injCount + 1
    }
   }
 }
 countVec = append(countVec, injCount)
}

# Create new data frame to store all observations with duplicate entries for multiple injury plays
countVec = unlist(countVec)          # Convert countVec back to vector
idx <- rep(1:nrow(dfInj), countVec)    # Create vector of which rows need to be duplicated for new df
dfDuplicates = dfInj[idx,]           # Create new vector with all observations and duplicates
```

```
dfDuplicates$inj_player = nameVec      # Create new column with injured players

# Find duplicate names that are back to back and remove
for (i in 2:nrow(dfDuplicates)) {
  if (dfDuplicates$inj_player[i] == dfDuplicates$inj_player[i-1]) {
    dfDuplicates = dfDuplicates[-i,]
  }
}

for (i in 1:length(dfDuplicates$inj_player)) {
  if (grepl("Jrr",dfDuplicates$inj_player[i])) {
    dfDuplicates$inj_player[i] = gsub("Jrr", " Jr.", dfDuplicates$inj_player[i])
  }
  if (grepl("Srr", dfDuplicates$inj_player[i])) {
    dfDuplicates$inj_player[i] = gsub("Srr", " Sr.", dfDuplicates$inj_player[i])
  }
  if (substr(dfDuplicates$inj_player[i], 1, 1) == "-") {
    dfDuplicates$inj_player[i] = gsub("-", "", dfDuplicates$inj_player[i])
  }
}

# Add snap count and position data
sc = nflreadr::load_snap_counts(2019:2023)

dfDuplicates[,"player"] = NA
dfDuplicates[,"position"] = NA
dfDuplicates[,"offense_snaps"] = NA
dfDuplicates[,"defense_snaps"] = NA
dfDuplicates[,"st_snaps"] = NA

for (i in 1:nrow(dfDuplicates)) {
  if (nchar(dfDuplicates$inj_player[i]) > 4) {
    tempSplit = strsplit(dfDuplicates$inj_player[i], " ")
    for (j in 1:nrow(sc)) {
```

```
    if (dfDuplicates$game_id[i] == sc$game_id[j] &&
        substr(tempSplit[[1]][1], 1, 1) == substr(strsplit(sc$player[j], " ")[[1]][1], 1, 1) &&
        tempSplit[[1]][2] == strsplit(sc$player[j], " ")[[1]][2]) {
      dfDuplicates$player[i] = sc$player[j]
      dfDuplicates$position[i] = sc$position[j]
      dfDuplicates$offense_snaps[i] = sc$offense_snaps[j]
      dfDuplicates$defense_snaps[i] = sc$defense_snaps[j]
      dfDuplicates$st_snaps[i] = sc$st_snaps[j]
    }
  }
 }
}

for (i in 1:nrow(dfFinal)) {
 if (is.na(dfFinal$player[i])) {
   dfFinal = dfFinal[-i,]
 }
}
dfInjuries = dfInjuries[-3577,]

# Create columns for offense and defense
defense = c("DE", "CB", "FS", "LB", "NT", "SS", "DT", "DB", "S", "DL", "ILB", "OLB", "CB/R")
offense = c("WR", "C", "T", "TE", "RB", "QB", "G", "FB", "P", "OL", "C/G", "G/T", "OT", "LS", "K")
linebackers = c("LB", "ILB", "OLB")
def_line = c("NT", "DT", "DL", "DE")
def_secondary = c("CB", "FS", "SS", "DB", "S", "CB/R")
skill_pos = c("QB", "WR", "TE", "RB", "FB")
off_line = c("C", "T", "G", "C/G", "G/T", "OT", "OL")

dfInjuries = dfFinal %>% mutate(tot_snap_count = offense_snaps + defense_snaps + st_snaps,
                        defense = ifelse(position %in% defense, 1, 0),
                        offense = ifelse(position %in% offense, 1, 0),
                        linebackers = ifelse(position %in% linebackers, 1, 0),
                        def_line = ifelse(position %in% def_line, 1, 0),
```

```
                    def_secondary = ifelse(position %in% def_secondary, 1, 0),
                    skill_pos = ifelse(position %in% skill_pos, 1, 0),
                    off_line = ifelse(position %in% off_line, 1, 0),
                    pos_group = ifelse(linebackers == 1, "LB",
                                ifelse(def_line == 1, "DL",
                                    ifelse(def_secondary == 1, "DB",
                                        ifelse(position == "QB", "QB",
                                            ifelse(position == "WR" | position == "TE", "WR/TE",
                                                ifelse(position == "RB" | position == "FB", "RB/FB",
                                                    ifelse(off_line == 1, "OL", "ST")))))))

write.csv(dfInjuries, "C:/Users/Jon/datasets/nflInjuries.csv")



dfInjuries = read.csv('nflInjuries.csv')

# Visualizations

# Breakout of number of players injured per total snap count
# Number of players injured per each position group
ggplot(data = dfInjuries, aes(x=pos_group, fill = pos_group)) +
 geom_bar() +
 labs(title="Total Injuries",
     subtitle="By Position Group",
     x = "Position Group",
     y = "Count") +
 guides(fill=guide_legend(title="Position Group"))

# Breakout of number of players injured per total snap count
ggplot(data = dfInjuries, aes(x=tot_snap_count, fill = pos_group), col(x = pos_group)) +
 geom_histogram(binwidth = 1) +
 #facet_grid(rows = vars(HOF)) +
 labs(title = 'Injuries Per Total Snap Count',
     x='Total Snap Count',
```

```
      y='Number of Players Injured') +
  guides(fill=guide_legend(title="Position Group"))

# Breakout of injuries per total snap count
ggplot(data = dfInjuries, aes(x=tot_snap_count, col = pos_group)) +
  geom_histogram(binwidth = 1) +
  facet_grid(rows = vars(pos_group)) +
  labs(title = 'Injuries Per Total Snap Count',
       subtitle="By Position Group",
       x='Total Snap Count',
       y='Number of Players Injured')

# Density number of players injured per each position group
# Special teams excluded because it messed up scale
dfInjuries %>% filter(pos_group != "ST") %>%
  ggplot(aes(x=tot_snap_count)) +
  geom_density(mapping = aes(x=tot_snap_count, col = pos_group)) +
  labs(title = 'Snap Counts Per Injury - Density Curves',
       subtitle="By Position Group",
       caption = "*Special teams excluded to avoid extreme scale disparities",
       x = 'Total Snap Count Before Injury',
       y='Density',
       color = "Position Group")

# Boxplots of average number of plays before injury for each position group
dfInjuries %>% filter(pos_group != "ST") %>%
  ggplot(aes(x=tot_snap_count, y=pos_group)) +
  geom_boxplot(mapping = aes(x=tot_snap_count, col = pos_group)) +
  labs(title = 'Snap Counts Per Injury - Boxplots',
       subtitle="By Position Group",
       caption = "*Special teams excluded to avoid extreme scale disparities",
       x = 'Total Snap Count Before Injury',
       y='Position Group',
       color = "Position Group")
```

```
# Number of players injured per each position group
ggplot(data = dfInjuries, aes(x=week, fill = pos_group)) +
 geom_bar() +
 labs(title="Total Injuries",
     subtitle="By Position Group",
     x = "Position Group",
     y = "Count") +
 guides(fill=guide_legend(title="Position Group"))



## Naive Bayes
# Narrow down dataset
dfML = dfInjuries[, c("week", "yardline_100", "down", "ydstogo", "yards_gained", "ydsnet", "play_type", "tot_snap_count",
"defense",
                "pos_group")]


## Create training and testing datasets
set.seed(12)
mask = sample.split(dfML$pos_group, SplitRatio = 0.80)
traindata = subset(dfML, mask == TRUE)
testdata = subset(dfML, mask == FALSE)
mod = naiveBayes(x = traindata[,-which(names(traindata) == "pos_group"), drop = FALSE],
            y = traindata$pos_group)


summary(mod)


y_pred = predict(mod, testdata[,-which(names(testdata) == "pos_group"), drop = FALSE])


testdata$pos_group = factor(testdata$pos_group, levels = levels(y_pred))
conf_matrix <- confusionMatrix(y_pred, testdata$pos_group)
conf_matrix
```

```
# Convert confusion matrix to data frame
conf_matrix_df <- as.data.frame(as.table(conf_matrix$table))

# Confusion Matrix
ggplot(conf_matrix_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
  scale_fill_gradient(low = "white", high = "blue") +
  labs(title = "Confusion Matrix",
       x = "Actual",
       y = "Predicted")

# ROC Curve
y_pred_probs <- predict(mod, newdata = testdata, type = "raw")
y_pred_probs_pos <- y_pred_probs[, 1]

roc_curve <- roc(testdata$pos_group, y_pred_probs_pos)
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
auc_value <- auc(roc_curve)
text(0, 1, paste("AUC-ROC =", round(auc_value, 3)), adj = c(0, 1), cex = 0.8, col = "black", font = 2)

#RQ1 and RQ2 analysis
nflreadr::.clear_cache()
df = nflreadr::load_pbp(2019:2023)
df_inj = read.csv('nflinjuries.csv')

vars = c("play_id", "game_id", "week", "yardline_100", "drive", "down", "goal_to_go", "ydstogo", "yards_gained", "ydsnet",
"shotgun",
         "no_huddle", "pass_length", "pass_location", "air_yards", "yards_after_catch", "run_location", "score_differential",
"solo_tackle",
         "tackled_for_loss", "series", "start_time", "weather", "play_clock", "drive_play_count", "drive_time_of_possession",
"roof", "surface", "temp",
         "wind")
df = df[,vars]
```

```
# Create indicator variables

# Columns to remove ass we create indicator variables

colrmv = c("surface", "roof", "drive_time_of_possession", "weather", "start_time", "pass_location", "pass_length",
"run_location")

# Turf

df$turf = ifelse(df$surface == "grass", 0, 1)


# Outdoor

df$outdoor = ifelse(df$roof=="outdoors", 1, 0)


# Total seconds of drive

df$total_drive_seconds = NA


for (i in 1:nrow(df)) {

  if (!is.na(df$drive_time_of_possession[i])) {

    df$total_drive_seconds[i] = strtoi(strsplit(df$drive_time_of_possession[i], ":")[[1]][1]) * 60 +
strtoi(strsplit(df$drive_time_of_possession[i], ":")[[1]][2])

  }

}

#checkpoint

df_cp = df


# Wind

df$wind = ifelse(df$weather == "N/A (Indoors) Temp: ° F, Wind:   mph", 0, 1)


# pass location

df$pass_middle = ifelse(df$pass_location == "middle", 1, 0)


# pass deep

df$pass_deep = ifelse(df$pass_length == "deep", 1, 0)


# runn middle

df$run_middle = ifelse(df$run_location == "middle", 1, 0)


# Delete columns used to make indicator variables
```

```
df = df[, !names(df) %in% colrmv]
#checkpoint
df_cp2 = df



# Add injured players and total snap counts
df$inj_player = 0
#df$snap_count_inj = 0
for (i in 1:nrow(df_inj)) {
 for (j in 1:nrow(df)) {
   if (df$play_id[j] == df_inj$play_id[i] && df$game_id[j] == df_inj$game_id[i]) {
     df$inj_player[j] = 1
     #df$snap_count_inj[j] = df_inj$tot_snap_count[i]
   }
 }
}
#checkpoint
df_cp3 = df

# Deal with NA values
df$temp[is.na(df$temp)] = 70
df$pass_middle[is.na(df$pass_middle)] = 0
df$pass_deep[is.na(df$pass_deep)] = 0
df$run_middle[is.na(df$run_middle)] = 0
df$drive[is.na(df$drive)] = 0
df$down[is.na(df$down)] = 0
df$air_yards[is.na(df$air_yards)] = 0
df$yards_after_catch[is.na(df$yards_after_catch)] = 0
df$yardline_100[is.na(df$yardline_100)] = 0
df$ydsnet[is.na(df$ydsnet)] = 0
df$total_drive_seconds[is.na(df$total_drive_seconds)] = 0
df$tackled_for_loss[is.na(df$tackled_for_loss)] = 0
df$solo_tackle[is.na(df$solo_tackle)] = 0
df$score_differential[is.na(df$score_differential)] = 0
```

```
df$outdoor[is.na(df$outdoor)] = 0
df$wind[is.na(df$wind)] = 0
df$drive_play_count[is.na(df$drive_play_count)] = 0

# Fix goal_to_go indicator
for (i in 1:nrow(df)) {
  if (df$goal_to_go[i] >= 0.5) {
    df$goal_to_go[i] = 1
  } else {
    df$goal_to_go[i] = 0
  }
}

#Control for all players getting through the game without injury. Average number of snaps is 50
df$snap_count_inj[which(df$snap_count_inj == 0)] = 50

colSums(is.na(df))
#sum(df$snap_count_inj == 0)



write.csv(df, "C:/Users/Jon/datasets/nfl_cleaned.csv")
df = read.csv("nfl_cleaned.csv")

# Remove first three columns
df = df[,4:30]
df_noSMOTE = df



# Code variables as factors
df$inj_player = as.factor(df$inj_player)
df$down = as.factor(df$down)
df$shotgun = as.factor(df$shotgun)
df$no_huddle = as.factor(df$no_huddle)
df$solo_tackle = as.factor(df$solo_tackle)
```

```
df$tackled_for_loss = as.factor(df$tackled_for_loss)

df$wind = as.factor(df$wind)

df$turf = as.factor(df$turf)

df$outdoor = as.factor(df$outdoor)

df$pass_middle = as.factor(df$pass_middle)

df$pass_deep = as.factor(df$pass_deep)

df$run_middle = as.factor(df$run_middle)

df$goal_to_go = as.factor(df$goal_to_go)


# Create SMOTE dataset

set.seed(123)

df_SMOTE = smote(inj_player ~ ., df_noSMOTE, perc.over = 2, k = 5, perc.under = 2)


sum(df_SMOTE$inj_player == 1)



# Visualizations for SMOTE variables

quant_vars = c("week", "yardline_100", "drive", "ydstogo", "yards_gained", "ydsnet", "air_yards", "yards_after_catch",
"score_differential",

            "series", "play_clock", "drive_play_count", "temp", "total_drive_seconds")


# Plotting histograms for each quantitative variable

for (var in quant_vars) {

  p <- ggplot(df_SMOTE, aes(x = !!sym(var), fill = !!sym("inj_player"))) +

    geom_histogram(alpha = 0.5, position = "identity", bins = 30) +

    ggtitle(paste("Distribution of", "'", var, "'", "by Outcome")) +

    xlab(var) +

    ylab("Frequency") +

    scale_fill_discrete(name = "inj_player")

  print(p)

}


# Plotting factor variables

factor_vars <- c('down', 'goal_to_go', 'shotgun', 'no_huddle', 'solo_tackle', 'tackled_for_loss', 'wind', 'turf', 'outdoor',
'pass_middle',
```

```
            'pass_deep', 'run_middle')

# Plotting bar plots for each factor variable
for (var in factor_vars) {
  p <- ggplot(df_SMOTE, aes(x = !!sym(var), fill = !!sym("inj_player"))) +
    geom_bar(position = "dodge") +
    ggtitle(paste("Counts of", "'", var, "'", "by Outcome")) +
    xlab(var) +
    ylab("Count") +
    scale_fill_discrete(name = "inj_player")

  print(p)
}

# Naive Bayes
## train test split for SMOTE data
set.seed(123)
splitIndex <- createDataPartition(df_SMOTE$inj_player, p = 0.7, list = FALSE)

traindata <- df_SMOTE[splitIndex, ]
testdata <- df_SMOTE[-splitIndex, ]

mod = naiveBayes(x = traindata[,-which(names(traindata) == "inj_player"), drop = FALSE],
            y = traindata$inj_player)

summary(mod)

y_pred = predict(mod, testdata[,-which(names(testdata) == "inj_player"), drop = FALSE])

testdata$inj_player = factor(testdata$inj_player, levels = levels(y_pred))
conf_matrix <- confusionMatrix(y_pred, testdata$inj_player)
conf_matrix

# ROC Curve
```

```
y_pred_probs <- predict(mod, newdata = testdata, type = "raw")
y_pred_probs_pos <- y_pred_probs[, 1]

roc_curve <- roc(testdata$inj_player, y_pred_probs_pos)
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
auc_value <- auc(roc_curve)
text(0, 1, paste("AUC-ROC =", round(auc_value, 3)), adj = c(0, 1), cex = 0.8, col = "black", font = 2)



# Naive Bayes
# No SMOTE
set.seed(123)
splitIndex <- createDataPartition(df_noSMOTE$inj_player, p = 0.7, list = FALSE)

traindata <- df_noSMOTE[splitIndex, ]
testdata <- df_noSMOTE[-splitIndex, ]
# mask = sample.split(dfML$pos_group, SplitRatio = 0.80)
# traindata = subset(dfML, mask == TRUE)
# testdata = subset(dfML, mask == FALSE)



mod = naiveBayes(x = traindata[,-which(names(traindata) == "inj_player"), drop = FALSE],
                 y = traindata$inj_player)

summary(mod)

y_pred = predict(mod, testdata[,-which(names(testdata) == "inj_player"), drop = FALSE])

testdata$inj_player = factor(testdata$inj_player, levels = levels(y_pred))
conf_matrix <- confusionMatrix(y_pred, testdata$inj_player)
conf_matrix

# Convert confusion matrix to data frame
conf_matrix_df <- as.data.frame(as.table(conf_matrix$table))
```

```
# Confusion Matrix
ggplot(conf_matrix_df, aes(x = Reference, y = Prediction, fill = Freq)) +
 geom_tile(color = "white") +
 geom_text(aes(label = sprintf("%d", Freq)), vjust = 1) +
 scale_fill_gradient(low = "white", high = "blue") +
 labs(title = "Confusion Matrix",
      x = "Actual",
      y = "Predicted")




# Logistic regression
set.seed(123)
splitIndex <- createDataPartition(df_SMOTE$inj_player, p = 0.7, list = FALSE)

traindata <- df_SMOTE[splitIndex, ]
testdata <- df_SMOTE[-splitIndex, ]

# New dataset
df_SLGM = df_SMOTE[, !names(df_SMOTE) %in% c("drive", "goal_to_go", "ydstogo", "ydsnet", "shotgun", "no_huddle",
"yards_after_catch", "tackled_for_loss",
                                             "play_clock", "drive_play_count", "wind", "turf", "outdoor")]

set.seed(123)
splitIndex <- createDataPartition(df_SLGM$inj_player, p = 0.7, list = FALSE)

traindata <- df_SLGM[splitIndex, ]
testdata <- df_SLGM[-splitIndex, ]

lmod <- glm(inj_player ~ ., traindata, family=binomial)
summary(lmod)
```

```
# See Accuracy
# Test Data
testdata$ypred = predict(lmod, newdata = testdata[,which(names(testdata) != "inj_player")], type = "response")
testdata$ypred = ifelse(testdata$ypred >= 0.5, 1, 0)

confM = table(testdata$inj_player, testdata$ypred)
confM

(confM[1,1]+confM[2,2])/sum(confM)

train_sens = (confM[2,2])/sum(confM[2,1], confM[2,2])
train_spec = (confM[1,1])/sum(confM[1,1], confM[1,2])



#Accuracy measures
# ROC Curve
predicted_probs <- predict(lmod, newdata = testdata, type = "response")
roc_curve <- roc(testdata$inj_player, predicted_probs)
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
auc_value <- auc(roc_curve)
text(0, 1, paste("AUC-ROC =", round(auc_value, 3)), adj = c(0, 1), cex = 0.8, col = "black", font = 2)

# Coef plot
coefplot(lmod)
"""
Created on Thu Jan 18 18:22:48 2024

@author: Aaron Stephenson
"""

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.patches as mpatches

#File path

file_path = 'D:/ASU/DAT-490/Capstone/merged_play_by_play_positions.csv'

data = pd.read_csv(file_path)




print('/nBasic Summary: ', data.describe())

print('/nData Types: ', data.dtypes)

print('/nMissing Values: ', data.isnull().sum())

#Check for missing values in 'desc' column.

missing_desc = data['desc'].isna().sum()

#Extract rows where 'desc' mentions injury

injury_keywords = ['injury', 'injured', 'hurt']

injury_data = data[data['desc'].str.contains(' | '.join(injury_keywords), case = False, na = False)]

#Overview of the injury data

injury_data_overview = {

   "total_rows": len(data),

   "missing_desc": missing_desc,

   "injury_reports": len(injury_data)

   }

print(injury_data_overview, injury_data.head())

#Analyzing the position injury frequency
```

```
injury_by_position = injury_data['position'].value_counts(normalize = True)

#Creating a dataframe for better visualization
injury_by_position_df = injury_by_position.reset_index()
injury_by_position_df.columns = ['Position', 'Injury Count']

print(injury_by_position_df.head(10))

#Set style for the plot
sns.set(style = "whitegrid")

#Plot
plt.figure(figsize=(10,6))
sns.barplot(x= 'Injury Count', y = 'Position', data = injury_by_position_df, palette="Blues_d")

plt.title('NFL Injuries by Position')
plt.xlabel('Number of Injuries')
plt.ylabel('Position')
plt.show()

#%%
#Select relevant features for clustering
features = data[['position', 'play_type']]

#Adding injury frequency
data['injury_frequency'] = data['position'].map(injury_by_position)

#Add dummies of play types
data_dummies = pd.get_dummies(data, columns=['position', 'play_type'])

#Selecting cluster features
clustering_features = data_dummies[['injury_frequency'] + [col for col in data_dummies if col.startswith('play_type_')]]
```

```
#Fill missing values and scale
clustering_features = clustering_features.fillna(0)
scaler = StandardScaler()
features_scaled = scaler.fit_transform(clustering_features)

#Number of K clusters
number_of_clusters = 5

#K-means clustering
kmeans = KMeans(n_clusters=number_of_clusters, random_state=0)
clusters = kmeans.fit_predict(features_scaled)

#Add cluster labels based on original DF
data['cluster'] = clusters

#Analyze cluster centers
cluster_centers = kmeans.cluster_centers_

#Create DF for the cluster_centers
centers_df = pd.DataFrame(cluster_centers, columns=clustering_features.columns)

#Analyze dominant feature for each cluster
cluster_labels = []
for i in range(number_of_clusters):
    # Extracting only play type features for the current cluster center
  play_type_features = centers_df.iloc[i][[col for col in centers_df.columns if col.startswith('play_type_')]]
  # Identifying the dominant play type (the one with the highest value)
  dominant_play_type = play_type_features.idxmax()
  # Creating the label using the dominant play type
  label = f"Cluster {i}: {dominant_play_type}"
  cluster_labels.append(label)

#Display cluster labels
```

```
for label in cluster_labels:
    print(label)


#Plot positions distributed across
plt.figure(figsize=(10,6))
ax = sns.countplot(x='position', hue='cluster',data=data)
plt.title('Position Distribution Across Clusters')
plt.xlabel('Position')
plt.ylabel('Count')
legend_patches = [mpatches.Patch(color = ax.get_legend().legendHandles[i].get_facecolor(), label=cluster_labels[i]) for i in range(number_of_clusters)]
plt.legend(handles=legend_patches, title="Clusters", loc = "upper right")
plt.show()


#%%
#Identify non-numeric columns
non_numeric_columns = data.select_dtypes(include=['object']).columns
print("Non-numeric columns:", non_numeric_columns)


columns_to_drop = ['game_id', 'home_team', 'away_team', 'season_type', 'posteam', 'defteam',
                   'side_of_field', 'game_date', 'game_half', 'time', 'start_time',
                   'time_of_day', 'stadium', 'weather', 'location', 'roof', 'surface',
                   'stadium_id', 'game_stadium', 'passer_id', 'rusher_id', 'receiver_id',
                   'name', 'id', 'my_key_injury', 'key12', 'my_key']
data = data.drop(columns_to_drop, axis=1)


#encoding categorical variables
categorical_cols = ['play_type', 'position']
data = pd.get_dummies(data, columns=categorical_cols)


#%%
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import classification_report, roc_auc_score



# Define your injury keywords

injury_keywords = ['injury', 'injured', 'hurt']


# Create a binary target variable

data['injury_occurred'] = data['desc'].str.contains('|'.join(injury_keywords), case=False, na=False).astype(int)


X = data.drop(['injury_occurred', 'desc'], axis = 1)

y= data['injury_occurred']


#Training and testing sets

X_train, X_test, y_train, y_test, = train_test_split(X, y, test_size = 0.2, random_state = 42)


# Replace infinite values and handle missing values

X_train.replace([np.inf, -np.inf], np.nan, inplace=True)

X_test.replace([np.inf, -np.inf], np.nan, inplace=True)


X_train = X_train.fillna(X_train.median())

X_test = X_test.fillna(X_test.median())


#Create and fit Random Forest

rf_model = RandomForestClassifier(random_state = 42)

rf_model.fit(X_train, y_train)


#Predictions and model eval

rf_predictions = rf_model.predict(X_test)

print(classification_report(y_test, rf_predictions))

print("ROC AUC Score:", roc_auc_score(y_test, rf_model.predict_proba(X_test)[:, 1]))


#%%

from sklearn.impute import SimpleImputer
```

```
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from imblearn.pipeline import Pipeline as ImbPipeline
from sklearn.model_selection import GridSearchCV

# Assuming 'X' and 'y' are defined and ready for preprocessing and model fitting

# Create a pipeline that includes SMOTE, scaling, and the RandomForestClassifier
pipeline = ImbPipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('smote', SMOTE(random_state=42)),
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(random_state=42, class_weight='balanced'))
])

# Define the parameter grid for GridSearchCV
param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [10, 20, None],
    # Add more parameters based on your dataset and model
}

# Initialize GridSearchCV with the pipeline
grid_search = GridSearchCV(estimator=pipeline,
                           param_grid=param_grid,
                           cv=3, n_jobs=1, scoring='roc_auc', verbose=2)

# Fit GridSearchCV
grid_search.fit(X_train, y_train)

# Print best parameters and score
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

```
# Evaluate the best model found by GridSearchCV
best_model = grid_search.best_estimator_
predictions = best_model.predict(X_test)
print(classification_report(y_test, predictions))



#%%
"""from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE"""

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Impute missing values
imputer = SimpleImputer(strategy = 'mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

#Ensure no infinite values
X_train_imputed = np.nan_to_num(X_train_imputed, nan=np.nan, posinf=np.max(X_train_imputed),
neginf=np.min(X_train_imputed))
X_test_imputed = np.nan_to_num(X_test_imputed, nan=np.nan, posinf=np.max(X_test_imputed),
neginf=np.min(X_test_imputed))



#Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_imputed, y_train)

#It's important to apply scaling after SMOTE to ensure the synthetic samples are also scaled
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_smote)
X_test_scaled = scaler.transform(X_test_imputed)
```

```
#Fit the Random Forest model to the SMOTE-augmented training set
rf_model = RandomForestClassifier(n_estimators=300, random_state=42)
rf_model.fit(X_train_scaled, y_train_smote)

"""X_train_imputed = np.nan_to_num(X_train_imputed, nan=np.nan)
X_test_imputed = np.nan_to_num(X_test_imputed, nan=np.nan)

# Then, reapply the imputer if necessary
X_train_imputed = imputer.fit_transform(X_train_imputed)
X_test_imputed = imputer.transform(X_test_imputed)"""

# Predictions and evaluation
predictions = rf_model.predict(X_test_scaled)
print(classification_report(y_test, predictions))
print("ROC AUC Score:", roc_auc_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1]))

#%%

# Assuming 'best_model' is your trained RandomForest model from GridSearchCV
feature_importances = best_model.named_steps['classifier'].feature_importances_

# Assuming 'X_train' is your training dataset before applying SMOTE and scaling
# If you've applied transformations that alter the feature names, adjust accordingly
feature_names = X_train.columns

# Create a DataFrame to hold the feature names and their importance scores
importances_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importances
}).sort_values(by='Importance', ascending=False).reset_index(drop=True)

# Display the top 10 most important features
```

```
print(importances_df.head(10))

# Plotting feature importances
plt.figure(figsize=(10, 8))
sns.barplot(x='Importance', y='Feature', data=importances_df.head(20))
plt.title('Top 10 Feature Importances in RandomForest Model')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()

#%%

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

#%%
TN, FP, FN, TP = cm.ravel()

sensitivity = TP / (TP + FN)
specificity = TN / (TN + FP)

print(f"Sensitivity (Recall): {sensitivity:.2f}")
print(f"Specificity: {specificity:.2f}")

from sklearn.metrics import classification_report

report = classification_report(y_test, predictions)
print(report)
```

```
#%%
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

#%%
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score

precision, recall, _ = precision_recall_curve(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])
average_precision = average_precision_score(y_test, rf_model.predict_proba(X_test_scaled)[:, 1])

plt.figure()
plt.step(recall, precision, where='post', label='Precision-Recall curve (area = %0.2f)' % average_precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve')
plt.legend(loc="upper right")
plt.show()
```

```
#%%
from sklearn.inspection import permutation_importance

perm_importance = permutation_importance(rf_model, X_test_scaled, y_test)
sorted_idx = perm_importance.importances_mean.argsort()

plt.figure(figsize=(12, 8))
plt.barh(range(len(sorted_idx)), perm_importance.importances_mean[sorted_idx], align='center')
plt.yticks(range(len(sorted_idx)), [feature_names[i] for i in sorted_idx])
plt.xlabel("Permutation Importance")
plt.show()

#%%
from sklearn.model_selection import learning_curve

train_sizes, train_scores, test_scores = learning_curve(rf_model, X_train_scaled, y_train_smote, cv=3)
train_scores_mean = np.mean(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)

plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")
plt.xlabel("Training examples")
plt.ylabel("Score")
plt.legend(loc="best")
plt.show()
```

References

Carl, S., & Baldwin, B. (2024). *nflfastR: Functions to Efficiently Access NFL Play by Play Data* (Version 4.6.1.9001) [Computer software]. https://github.com/nflverse/nflfastR, https://www.nflfastr.com/.

Nfl. (2023, October 17). Injury data since 2015. *NFL.com*. https://www.nfl.com/playerhealthandsafety/health-and-wellness/injury-data/injury-data

Cesconetto, G. I. (2021, September 27). How many NCAA football players make it to the NFL? *Sportskeeda*. https://www.sportskeeda.com/college-football/how-many-ncaa-football-players-make-nfl

Gough, C. (2023, September 5). NFL revenue 2022. *Statista*. https://www.statista.com/statistics/193457/total-league-revenue-of-the-nfl-since-2005/#statisticContainer

Allahabadi, S., Gatto, A. P., & Pandya, N. K. (2022). ACL tears in the National Football League from 2013 to 2020: Analysis of the 2020 season after delays and schedule changes from the early COVID-19 pandemic relative to prior seasons. *Orthopaedic Journal of Sports Medicine, 10*(2), 23259671221076044. https://doi.org/10.1177/23259671221076045

Perez, J. R., Burke, J., Zalikha, A., Schiller, N., Buskard, A. N., Damodar, D., Kaplan, L. D., & Baraga, M. G. (2019). Does short rest with Thursday night games influence injury rates in the NFL? *Orthopaedic Journal of Sports Medicine, 7*(7_suppl5), 2325967119S00341. https://doi.org/10.1177/2325967119S00341

Brenner, J. S., & Watson, A. (2024, January 22). Overuse injuries, overtraining, and burnout in young athletes. *American Academy of Pediatrics*. https://publications.aap.org/pediatrics/article/153/2/e2023065129/196435

Vegso, S., Cantley, L., Slade, M., Taiwo, O., Sircar, K., Rabinowitz, P., Fiellin, M., Russi, M.B., & Cullen, M.R. (2007). Extended work hours and risk of acute occupational injury: A case-crossover study of workers in manufacturing. *American Journal of Industrial Medicine, 50*, 597-603. https://doi.org/10.1002/ajim.20486

Haider, S., et al. (2018). Does the environment influence the frequency of concussion incidence in professional football? *Cureus, 10*(11), e3627. https://doi.org/10.7759/cureus.3627

Mack, C. D., et al. (2019). Higher rates of lower extremity injury on synthetic turf compared with natural turf among National Football League athletes: Epidemiologic confirmation of a biomechanical hypothesis. *The American Journal of Sports Medicine, 47*(1), 189-196. https://doi.org/10.1177/0363546518808499

Nuelle, C., Sherman, S., krumme, j., & Gray, A. (2016). Incidence and variance of injuries by position in American football players: An epidemiologic review. https://www.researchgate.net/publication/301203948

Pellman, E. J., Viano, D. C., Tucker, A. M., Casson, I. R., & Waeckerle, J. F. (2003). Concussion in professional football: Reconstruction of game impacts and injuries. *Neurosurgery, 53*(3), 799-814. https://doi.org/10.1093/neurosurgery/53.3.799

Barnwell, B. (n.d.). Why star NFL running backs have been devalued: What's next. *ESPN*. Retrieved January 29, 2024, from https://www.espn.com/nfl/story/_/id/37985469/why-star-nfl-running-backs-devalued-next-2023-free-agency-future

NFLPA expected to file grievance against Panthers for "rough" field. (n.d.). *WCNC.com*. Retrieved January 29, 2024, from https://www.wcnc.com/article/sports/nfl/panthers/nflpa-grievance-carolina-panthers-detroit-lions-playing-surface-hard-field-bank-of-america-stadium-jared-goff-frank-ragnow/275-83234a1b-3fe4-406c-9ded-d609cbd864a7

Salvador, J. (n.d.). Lions' Jared Goff rips field conditions at Panthers' stadium. *SI.com*. Retrieved January 29, 2024, from https://www.si.com/nfl/2022/12/25/lions-jared-goff-rips-field-conditions-panthers-bank-america-stadium