The Application of Machine Learning to Predict NFL Running Back Performance

STATS-5405

Giovanni Lunetta[a,*], Sam Lutzel[a,*]

[a] *University of Connecticut, Statistics, 2075 Hillside Road, Storrs, 6269*

Abstract

This study employs an ensemble of machine learning techniques, including Multiple Linear Regression (MLR), Generalized Linear Models (GLM), and XGBoost to enhance predictive analytics in the National Football League (NFL). The research focuses on one of the most critical aspects of football offense - the running game. In order to predict the yards gained by NFL running backs following a handoff, the study analyzes a plethora of player tracking data from the 2017 to 2019 seasons. It integrates a variety of factors such as player positions, orientation, and the situational context of the game, including down, distance, and field position. This multifaceted approach aims to yield highly accurate models that can serve as valuable tools for coaching staffs to optimize play-calling, manage player workloads, and enhance game planning. The predictive insights derived from this research are intended to support teams in deploying their running backs more effectively, leading to potentially improved outcomes on the field. As the NFL continues to evolve with a greater emphasis on analytics, this study seeks to contribute significantly to the field of sports analytics by providing a model that underscores the importance of the running game in a predominantly pass-oriented league.

Keywords: Machine Learning, Predictive Analytics, Sports Analytics

1. Introduction

1.1. Background of the National Football League

The National Football League (NFL) is the most popular professional sports league in the United States, with an estimated 187 million fans. The league consists of 32 teams divided into two conferences, the American Football Conference (AFC) and the National Football Conference (NFC). Each conference is further divided into four divisions, with four teams in each division. The NFL season consists of 17 weeks, with each team playing 16 games and having one bye week. The regular season is followed by a 12-team playoff, with the winner of each conference advancing to the Super Bowl, the league's championship game.

The NFL game strategy primarily consists of two ways to advance the football down the field: running and passing. Running the ball is defined as a play in which the quarterback gives the ball to another player that is located behind them (a handoff or a small toss). The player then attempts to run the ball down the field as far as possible before being tackled to the ground. Passing the ball is defined as a play in which the quarterback throws the ball to another player down the field. The player then attempts to catch the ball and advance it down the field as far as possible before being tackled to the ground. The goal of each team is to score as many points as possible by advancing the ball down the field and into the end zone or by kicking the ball through the field goal posts. The team with the most points at the end of the game wins.

*Corresponding author

Email addresses: giovanni.lunetta@uconn.edu (Giovanni Lunetta), samuel.lutzel@uconn.edu (Sam Lutzel)

1.2. Literature Review of "Predicting NFL running back rushing yards using Hierarchical Bayesian Linear Regression"

In the realm of NFL predictive analytics, the study "Predicting NFL running back rushing yards using Hierarchical Bayesian Linear Regression" stands out for its innovative approach, aligning closely with our study's objectives. It specifically aimed to forecast NFL running backs' performance using Bayesian linear regression, drawing on the "NFL Offensive Stats 2019 – 2022" dataset. The study's methodology, employing a student t distribution to account for the positively skewed rushing yards data and integrating informative priors based on factors like team favorability, weather conditions, and game-specific variables, offers a valuable framework for our research. These elements, including the use of R for data processing and JAGS for model compilation, underscore the importance of considering individual player performance and contextual game factors. This approach is particularly relevant to our study, which also seeks to enhance predictive models in football using machine learning techniques. By focusing on player-specific and game-context data, the referenced study provides key methodological insights and underscores the significance of situational analysis in sports analytics, thereby informing our approach to leveraging machine learning in predicting NFL running back performance.

1.3. Motivation

The primary goal of this research is to develop cutting-edge predictive models capable of accurately estimating the yards a running back will gain after a handoff during NFL games. This objective is pivotal for formulating advanced offensive strategies and refining player evaluations. The running play is a fundamental aspect of the game that can dictate the tempo, control the clock, and establish physical dominance.

The motivation behind this study stems from the transformative impact that data analytics has had on sports, particularly in the NFL, where the fusion of technology and sports science has begun to redefine how the game is played and understood. In an era where marginal gains are increasingly sought after, the ability to predict the outcome of running plays with high precision can provide teams with a significant competitive advantage. It enables coaches to make informed decisions regarding play selection, player rotations, and game management, especially in critical moments of a match. Additionally, the insights from this study can empower front offices in their scouting and drafting processes by quantifying the expected value a running back adds to their team. Furthermore, there is also a tremendous opportunity to leverage these predictive models on the defensive end of the ball, allowing teams to better anticipate and defend against running plays. With better insights into the opponents running game, defenses can adjust their schemes and personnel to counter the opposing team's offensive strategy, such as by stacking the box or blitzing the quarterback. In addition to the benefits performance analytics provides to the teams, it also helps fans select better fantasy football teams and make more informed betting decisions. This leads to a more engaging and enjoyable experience for the fans, which is critical for the long-term success of the league.

Ultimately, the true beauty of this research lies in its ability to bridge the gap between complex player tracking data and practical on-field strategies. In specific, it's about enhancing the very essence of the game and enriching the broader discourse on sports performance analytics. By pioneering research in this domain, this study is set to propel the analytical capabilities of NFL teams to new heights, providing them with tools that were once considered unimaginable. Ultimately, it contributes to the ongoing evolution of the sport itself, marking a pivotal moment in the history of football and the broader world of sports analytics.

2. Methods

2.1. Data Cleaning and Preprocessing

Data cleaning and preprocessing are critical to ensuring the quality and integrity of machine learning models. The cleaned dataset was obtained by addressing any inconsistencies and handling missing data through imputation techniques. Preprocessing steps included encoding categorical variables using one-hot encoding or label encoding methods and scaling and normalizing numerical variables to ensure they are on comparable scales.

To begin, several variables were removed in order to simulate the beginning of each play. For instance, the dataset include the speed, acceleration, direction, and distance traveled of each player on the field at the beginning of each play. However, this information is not available to the coaching staff prior to the snap of the ball. Therefore, these variables were removed from the datset. Furthermore, some variables may introduce multicollinearity issues since they represent the same information. For instance, the dataset includes the name and id of each player on the field. However, these variables are highly correlated with each other. Therefore, only one of these variables was kept in the dataset.

When the dataset was first obtained, each row represented a single player on the field. However, the goal of this study is to predict the yards gained after a handoff for each play. Therefore, every 22 rows had to be merged into one row (11 players on each side). During this process, any duplicated information/columns were dropped from the dataset. One example of this is the weather during the play. When all 22 rows are merged, it included the weather 22 times, even though weather is only needed once.

In addition to the attempt at removing unnecessary variables and proactively removing potenital multicollienarity issues, variables such as weather, wind speed, and wind direction had multiple levels, some of which represented the same data. For instance, "rainy" and "showers" were considered the same weather type. There were multiple instances of relationships between inputs on the same variable that existed similar to the example above. Due to this, these weather types were grouped into one category - "rain." This helps to drastically reduce the complexity of the dataset. These variables were also converted to factors for the modeling process.

Variables such as offense personnel had to be converted to dummy variables. The input for each dummy variable was either 0 or 1 depending on whether or not the offense had that personnel on the field. For instance, if the offense had 2 running backs on the field, the input for the "RB" dummy variable would be 2. The input for the "WR" dummy variable would be 0 since there were no wide receivers on the field. This process was repeated for all personnel types.

One additional step that was taken was the randomization of all plays within the dataset. The reason for this randomization is to ensure that each row (play of a game) is independent of one another. Furthermore, the dataset only contains running plays. Therefore, passing plays that occured between the running plays were removed. In other words, all plays are not dependent on the previous outcome. In all, the randomization of the dataset in conjunction with the removal of passing plays ensures that each row is independent of one another.

2.2. Data Exploration

Prior to the development of the predictive models, a comprehensive exploratory data analysis (EDA) was conducted to understand the underlying structure and characteristics of the data. This step involved visualizing the distribution of key variables, identifying patterns and outliers, and exploring correlations and interactions between predictors. Data visualization, through techniques like scatter plots, histograms, and heatmaps, can offer an intuitive understanding of data distributions, correlations, and potential clusters within the dataset. EDA-informed feature selection and engineering strategies highlighted potential predictors that are most informative of yards gained after a handoff.

2.3. Feature Engineering

During preprocessing, we will also undertake feature engineering to create new variables that may have a stronger relationship with the target variable. This could include interaction terms that capture the combined effect of two predictors, polynomial features for capturing non-linear relationships, and domain-specific features that encapsulate strategic elements of the game.

3. Variable Selection and Explanation:

3.1. Dependent Variable:

- `Yards`: The yardage gained on the play (we are predicting this).

3.2. Game Context Variables:

- `Quarter`: The quarter of the game (1-5, 5 == overtime). Indicates the stage of the game. Player fatigue, strategic decisions, and urgency can vary significantly depending on the game quarter.
- `YardsFromOwnGoal`: The number of yards from the offense's own goal line. Provides context on field position. The closer to the end zone, the higher the potential pressure and change in play strategy.
- `Down`: The down (1-4). Critical for understanding the play strategy. Teams might take different risks or play types depending on the down.
- `Distance`: The yards needed for a first down. Directly impacts the play call. Short distances might favor running plays, while longer distances might necessitate passing.
- `OffenseFormation`: The formation of the offense. Influences the type of play likely to be called and the potential for yardage gain.

3.2.1. Environmental Variables:

- `GameWeather`, `Temperature`, `Humidity`, `WindSpeed`, `WindDirection`: The weather, temperature, humidity, windspeed and wind direction during that play of the game. Weather conditions can significantly affect gameplay, influencing factors like ball handling, player performance, and play-calling strategy. For instance, windy and rainy conditions might favor running plays, while games played in a dome might favor passing plays.

3.2.2. Player Composition Variables:

- `RB`, `TE`, `WR`: The number of running backs, tight ends, and wide recievers on the field during that play. The number of players in specific roles can indicate the likely type of play (running vs. passing) and the potential for yardage gain.
- `DL`, `LB`, `BL`: The number of defensive linemen, linebackers, and defensive backs on the field during that play. The defensive setup can provide insights into how the defense is preparing to counter the offense, impacting the offense's success in gaining yards.

3.2.3. Game State Variables:

- `ScoreDelta`: The difference in score between the two teams. The score difference can dictate the urgency and risk level of plays. Teams behind by a large margin might favor riskier, longer yardage plays. Teams ahead by a large margin might favor safer, shorter yardage plays, especially at the end of games when the clock is running out.
- `DefendersInTheBox`: The number of defensive players lined up near the line of scrimmage, inside an imaginary area known as "the box." This area extends laterally about the width of the offensive line and a few yards forward and backward from the line of scrimmage. Indicates the defensive focus on stopping the run. More defenders in the box typically suggest a greater emphasis on stopping running plays.
- `Week`: The week of the game (1-17). Can be an indicator of team development and adaptation over the season. Teams usually take a few weeks to find their rhythm and establish their identity but become more run down and fatigued as the season progresses, specifically running backs who take a lot of hits.
- `GameHour`: The hour of the game (1-24). Might correlate with environmental factors or player fatigue. For instance, night games might be colder, while afternoon games might be warmer. Additionally, late games such as prime-time games might be more physically and mentally taxing on players.

4. Model Development and Evaluation

With a clean and prepared dataset, we will then proceed to develop our MLR, GLM, and XGBoost models.

4.1. Multiple Linear Regression (MLR)

Multiple linear regression models predict a continuous repsonse variable using a linear combination of predictors. For our baseline MLR model, we will use the ordinary least squares (OLS) method to estimate the coefficients of our predictor variables. The model is specified as:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_j X_{ij} + \epsilon_i$$

where Y_i represents the yards gained after the handoff for the i^{th} observation, X_{ij} is the i^{th} observation on the j^{th} predictor variable (where $j = 1, ..., p$), β_j is the j^{th} coefficient to be estimated corresponding to the proper X_{ij} variable, and ϵ_i is the error term. The j^{th} coefficient, β_j, represents the change in the response variable for a unit change in the X_{ij} predictor variable, while holding all other predictors constant.

The ordinary least squares (OLS) method will be deployed to minimize the sum of the squared differences between the observed and predicted values. The optimization problem can be represented as minimizing the sum of errors squared:

$$\Sigma_{i=0}^{n} \epsilon_i^2 = \Sigma_{i=0}^{n} (Y_i - \hat{Y}_i)^2 = \Sigma_{i=0}^{n} (Y_i - \beta_0 - \beta_1 X_{i1} - ... - \beta_j X_{ij})^2$$

Here, $\Sigma_{i=0}^{n} \epsilon_i^2$ represents the sum of the squared residuals, which we aim to minimize. The variable $\hat{Y}_i$ represents the predicted outcome of the model for the i^{th} observation. This approach assumes that the relationship between the independent variables and the dependent variable is linear. To ensure the robustness of our model, we will conduct a series of diagnostic tests:

1. Linearity: We will use scatter plots and residual plots to verify that the relationship between the predictors and the response is linear.
2. Homoscedasticity: We will inspect the residuals to confirm constant variance across all levels of the independent variables. This can be assessed visually using a residual vs. fitted values plot.
3. Independence: The Durbin-Watson test will help in detecting the presence of autocorrelation in the residuals, which should not be present in the data.
4. Normality of Residuals: Normality will be checked using Q-Q plots and statistical tests like the Shapiro-Wilk test.

If any assumptions are violated, we may consider transformations of variables or the use of robust regression techniques.

4.2. Generalized Linear Models (GLM)

GLMs extend the linear model framework to allow for response variables that have error distribution models other than a normal distribution. They are particularly useful when dealing with non-normal response variables, such as count data or binary outcomes. In its general form, a GLM consists of three elements:

1. Random Component: Specifies the probability distribution of the response variable Y, such as normal, binomial, Poisson, or exponential.
2. Systematic Component: A linear predictor $\eta = X\beta$.
3. Link Function: A function g that relates the mean of the response variable $E(Y)$ to the linear predictor η.

The choice of link function is crucial and is typically selected based on the nature of the distribution of the response variable. For instance, a logit link function is used for a binomial distribution, and a log link function is often used for a Poisson distribution.

The likelihood function for a GLM can be written as:

$$L(\beta) = \prod_{i=1}^{n} f(y_i; \theta_i, \phi)$$

where $f(y_i; \theta_i, \phi)$ is the probability function for the i-th observation, θ_i is the parameter of interest (e.g., mean), and ϕ is the dispersion parameter. The goal is to find the values of β that maximize this likelihood function.

4.3. XGBoost

XGBoost stands for eXtreme Gradient Boosting and is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It is a powerful technique that can handle a variety of regression and classification problems. For regression, it can be configured to optimize for different loss functions; the most common for regression being the squared error loss:

$$L(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where y_i are the observed values, and $\hat{y}_i$ are the predicted values.

In XGBoost, each new tree is built to correct the errors made by the previous ones. The algorithm combines weak predictive models to form a strong predictor. The model's complexity is controlled by the regularization term $\Omega(\theta)$ which is a function of the tree structure and the number of leaves. The overall objective function to be minimized is:

$$\text{Obj}(\theta) = L(\theta) + \lambda \sum_{k} (w_k^2) + \gamma T$$

where w_k represents the leaf weights of the trees, T is the number of leaves, λ is the L2 regularization term on the weights, and γ is the complexity control on the number of leaves. For regression tasks, we can also utilize the quantile loss which is particularly useful for prediction intervals:

$$L_\tau(\theta) = \sum_{i=1}^{n} \left[\tau(y_i - \hat{y}_i) \mathbb{1}_{y_i \geq \hat{y}_i} + (1 - \tau)(\hat{y}_i - y_i) \mathbb{1}_{y_i < \hat{y}_i} \right]$$

Here, 1 is an indicator function, and τ is the quantile of interest, allowing us to model different parts of the conditional distribution of the response variable.

XGBoost also provides a feature importance score, which is a metric that quantifies the contribution of each feature to the model's predictive power. This is done by measuring the impact on the model's accuracy each time a feature is used to split the data across all trees.

4.4. Model Performance Metrics

For MLR and GLM, the model's performance will be evaluated using metrics such as the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). For the XGBoost model, along with MSE and MAE, we will assess performance using additional metrics like the R-squared for regression tasks and feature importance scores to understand which variables are most predictive.

4.5. Model Interpretation and Application

The final step will be to interpret the models in the context of NFL games. This will involve translating the statistical outputs into actionable insights for coaches and team strategists, providing recommendations on how to leverage the results for competitive advantage in play-calling and player evaluation.

4.6. Software and Tools

All analyses will be conducted in R, a statistical computing language that provides a wide array of packages for machine learning and data analysis. For MLR and GLM, we will utilize the `stats` package that comes with the R base installation. For our XGBoost model, we will use the `xgboost` package, which is specifically designed for speed and performance. Data manipulation and cleansing will be managed with packages like `dplyr` and `tidyr`, while `ggplot2` will be employed for data visualization to facilitate understanding and interpretation of the data and model outputs. For feature engineering and preprocessing, we will take advantage of `caret` or `recipes`. Hyperparameter tuning can be optimized using the `tune` package, and for cross-validation, the `rsample` package will be employed. The `broom` and `modelr` packages will be useful for tidying model outputs and working with models in a pipeline, respectively. This suite of packages will enable a comprehensive workflow within R for developing, evaluating, and interpreting the predictive models.

5. Model Development

To commence our analytical journey, we initiate with Multiple Linear Regression (MLR) as our foundational modeling technique. This approach is not just a stepping stone, but a critical phase in our analysis, offering valuable insights into the relationships between various features of the game and the yards gained. MLR helps discern the linear associations and relative importance of different variables, setting the stage for more complex modeling. From there, we transition to the Generalized Linear Model with a Gamma Response. Generalized Linear Models (GLMs) are a powerful extension of the linear model framework, allowing for response variables that have error distribution models other than a normal distribution. GLMs are particularly useful when dealing with non-normal response variables, such as count data or binary outcomes. In its general form, a GLM consists of three elements: a random component, a systematic component, and a link function. The choice of link function is crucial and is typically selected based on the nature of the distribution of the response variable. For instance, a logit link function is used for a binomial distribution, and a log link function is often used for a Poisson distribution. In our specific case, the Gamma distribution is chosen as the response variable, as it is a continuous distribution that can handle non-negative values.

Subsequent to this exploratory analysis, we transition to XGBoost, an advanced machine learning technique renowned for its predictive power and efficiency. XGBoost, a gradient boosting framework, is chosen for its ability to handle the dataset's complexity, non-linear relationships, and interactions among variables more adeptly. This shift from MLR to XGBoost embodies our methodological progression, from understanding the foundational relationships in our data to harnessing advanced computational techniques for more accurate and robust predictions in the dynamic and unpredictable context of NFL games.

5.1. GLM with Gamma Response

First, lets start by importing the completely clean and ready to use dataset. This dataset was the result of procedures outlined in the data cleaning and preprocessing section.

```
# Import the dataset
football_data <- read.csv("/Users/samlutz10/Desktop/STAT5405/Final Project/train_ready.csv")

str(football_data)
```

```
'data.frame':   31007 obs. of  21 variables:
 $ Quarter          : int  2 2 4 2 2 1 2 4 4 4 ...
 $ Down             : int  1 1 1 2 1 1 1 1 1 1 ...
 $ Distance         : int  10 10 10 10 10 10 10 10 10 10 ...
 $ OffenseFormation : chr  "SINGLEBACK" "I_FORM" "I_FORM" "SINGLEBACK" ...
 $ RB               : int  2 2 2 1 1 1 1 1 2 1 ...
 $ TE               : int  1 1 2 1 1 2 1 1 1 1 ...
```

```
$ WR                : int  2 2 1 3 3 2 3 3 1 3 ...
$ DefendersInTheBox: int  8 7 8 6 7 8 6 7 7 7 ...
$ Yards             : int  1 12 3 -2 14 4 3 7 7 3 ...
$ Week              : int  3 9 14 2 2 2 17 11 17 3 ...
$ GameWeather       : chr  "clear" "overcast" "clear" "clear" ...
$ Temperature       : int  84 77 39 81 79 79 19 68 71 68 ...
$ Humidity          : int  25 62 55 45 48 48 36 54 66 42 ...
$ WindSpeed         : num  5 16 11 5 5 5 12 4 9 13 ...
$ WindDirection     : chr  "west" "south east" "none" "east" ...
$ GameHour          : int  15 9 3 3 13 12 2 6 15 12 ...
$ DL                : int  4 3 4 4 2 4 2 4 4 4 ...
$ LB                : int  3 4 3 2 4 3 4 2 3 2 ...
$ BL                : int  4 4 4 5 5 4 5 5 4 5 ...
$ YardsFromOwnGoal : int  40 84 79 21 86 25 49 25 65 54 ...
$ ScoreDelta        : int  -7 0 -13 -11 0 7 -11 41 -16 -8 ...
```

Since we are only focused on the most common plays and not breakout runs, we will remove any data of 10 yards or more. This is due to the fact that plays resulting in 10 yards are considered to be just as successful are play resulting in more than 10 yards. In addition, any plays resulting in 0 or negative yards will be removed. This is due to the fact that these plays are considered to be unsuccessful.

```
# Remove any rows whose yards column is greater than 30
football_data <- football_data[football_data$Yards <= 10 & football_data$Yards > 0, ]
```

The next model that will be synthesized is a GLM with a gamma response. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data. The gamma distribution is a two-parameter family of continuous probability distributions. The gamma distribution is a generalization of the exponential distribution. The gamma distribution is frequently used to model right-skewed data.

Lets start by removing any rows whose yards column is negative. This is because the gamma distribution cannot have negative values.

```
# Remove any rows whose yards column is negative
football_data <- football_data[football_data$Yards >= 0.1, ]
football_data <- football_data[football_data$Yards <= 10, ]
```

Lets start by fitting the null glm model. This model will be used as a baseline for comparison to the other models.

```r
# Fit the model
glm_model_gamma <- glm(Yards ~ 1, data = football_data, family = Gamma(link = "log"))
```

Now we can run the full model.

```r
# Fit the model
glm_model_gamma <- glm(Yards ~ ., data = football_data, family = Gamma(link = "log"))
```

Now lets use the step function to find the best model for the data.

```r
glm_model_step <- step(glm_model_gamma, direction = "backward")
summary(glm_model_step)
```

Running this reduced model yields the following results:

```r
reduced_model <- glm(Yards ~ YardsFromOwnGoal + Distance + RB + WR + DefendersInTheBox + BL, data=f
summary(reduced_model)
```

```
Call:
glm(formula = Yards ~ YardsFromOwnGoal + Distance + RB + WR +
    DefendersInTheBox + BL, family = Gamma(link = "log"), data = football_data)

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.8459028  0.0763143  24.188  < 2e-16 ***
YardsFromOwnGoal  -0.0015976  0.0001702  -9.389  < 2e-16 ***
Distance           0.0122438  0.0011600  10.555  < 2e-16 ***
RB                 0.0008051  0.0111905   0.072 0.942643
WR                -0.0146274  0.0082184  -1.780 0.075118 .
DefendersInTheBox -0.0938411  0.0060793 -15.436  < 2e-16 ***
BL                 0.0334060  0.0094814   3.523 0.000427 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 0.3704999)

    Null deviance: 9177.3  on 21900  degrees of freedom
Residual deviance: 8838.8  on 21894  degrees of freedom
AIC: 93929

Number of Fisher Scoring iterations: 5
```

The regression model was fitted to predict Yards using several predictor variables. The intercept, representing the expected value of Yards when all other predictors are zero, is highly significant (t = 24.188, p < 2e-16), indicating that there is a substantial baseline value of Yards. This intercept is consistent with the notion that even in the absence of other factors, some yards are expected.

Moving to individual predictors, YardsFromOwnGoal exhibits a negative coefficient (-0.0015976) with high significance (t = -9.389, p < 2e-16), implying that as a team moves away from their own goal, the expected yards tend to increase. Distance, with a positive coefficient of 0.0122438 and high significance (t = 10.555, p < 2e-16), suggests that longer distances on the field are associated with greater yards gained. These results

align with intuitive expectations in football, where advancing towards the opponent's goal typically results in more yards gained.

However, not all predictors are equally influential. RB, representing a certain player position, lacks significance (t = 0.072, p = 0.942643), indicating that the presence of RB does not significantly contribute to the variation in Yards. WR, on the other hand, has a marginally significant negative impact (t = -1.780, p = 0.075118), suggesting that having more Wide Receivers on the field might be associated with fewer yards gained. This result should be interpreted cautiously due to its marginal significance.

DefendersInTheBox, with a negative coefficient of -0.0938411 and high significance (t = -15.436, p < 2e-16), implies that having more defenders close to the line of scrimmage is associated with a decrease in yards gained. BL (perhaps denoting Blockers), with a positive coefficient of 0.0334060 and significance (t = 3.523, p = 0.000427), suggests that a strong blocking presence contributes positively to the yards gained.

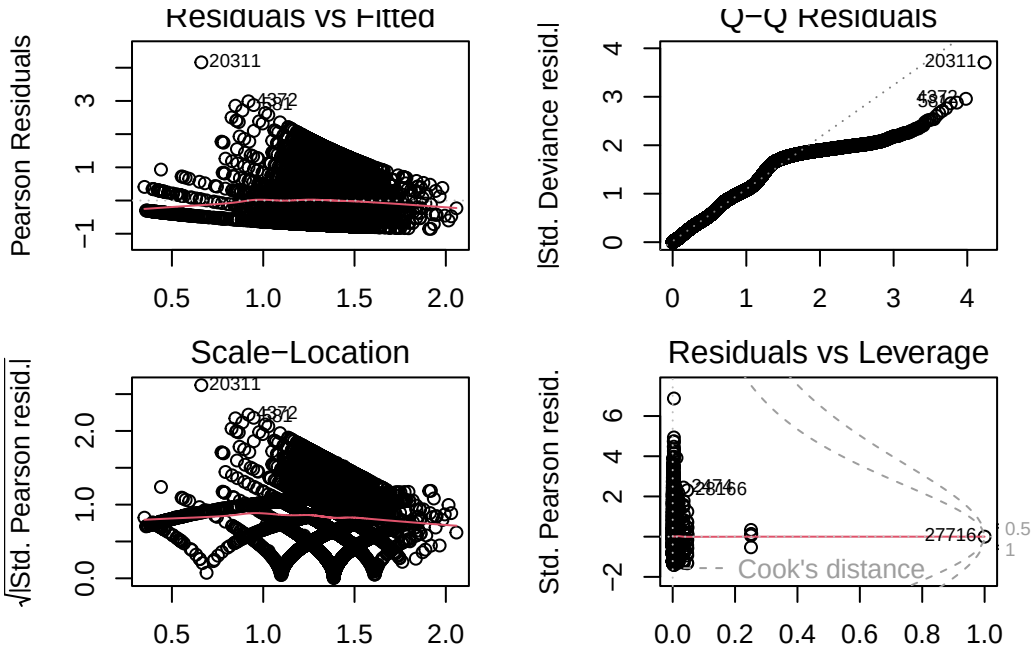
Turning attention to additional model diagnostics, the dispersion parameter for the Gamma family is 0.3704999. A lower dispersion parameter indicates that the model is well-fitted to the data. The null deviance (9177.3) and residual deviance (8838.8) provide insights into model improvement. The decrease in deviance suggests that the predictors collectively enhance the model's explanatory power. The Akaike Information Criterion (AIC) of 93929, while relatively high, serves as a trade-off between goodness of fit and model complexity. Model improvement might be achieved through further refinement or the consideration of interaction terms.

In conclusion, this regression model, while generally well-fitted and insightful, warrants careful consideration of the significance and practical implications of each predictor. Adjustments may be made to enhance interpretability and simplify the model without compromising its explanatory power.

By creating a Residuals vs Fitted plot, a Q-Q Residuals plot, a Scale-Location plot, and a Residuals vs Leverage plot, the assumptions of the model can be checked.

```
par(mfrow = c(2, 2))
old.par = par(mar = c(3, 4, 1, 2))
plot(glm_model_gamma)
```

```
Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



Above, we have four different plots called Residuals vs Fitted, Q-Q Residuals, Scale-Location, and Residuals vs Leverage. The Residuals vs Fitted plot shows that the residuals are randomly distributed around the 0 line. This indicates that the residuals are normally distributed. Although, there are some outliers that are far away from the 0 line.

The Q-Q Residuals plot shows that the residuals are, for the most part, normally distributed. However, there are some outliers that are far away from the 0 line. In addition, the line begins to veer away from the line of normality on the right side of the plot.

The Scale-Location plot does show a strange pattern in the residuals. It appears to be a bouncing ball pattern where the residuals fall to zero and then increase again, only to eventually come back down to zero. The line can be most aptly desrcibed by the path a ball bouncing a long a table would make.

The presence of a bouncing ball pattern in the Scale-Location plot of residuals may be indicative of systematic errors or trends in the data that the model is not capturing adequately.

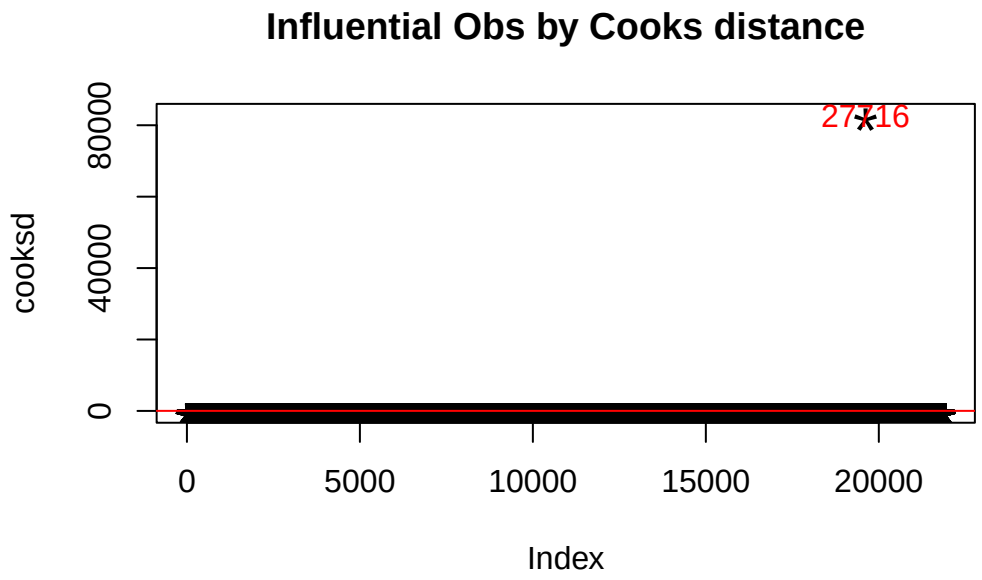
The bouncing ball pattern could be a signal of non-linearity in the relationship between the predictors and the response variable. It indicates that as the predicted values of the response variable change, the variability of the residuals also changes. The residuals first decrease, then increase, and finally decrease again, forming a pattern resembling a bouncing ball.

In addition, the presence of outliers or influential data points could lead to the observed pattern. These points might have a disproportionate impact on the model's fit, causing the residuals to exhibit a bouncing ball behavior.

The Residuals vs Leverage plot shows that there are some outliers that are far away from the 0 line. These outliers could be the cause of the bouncing ball pattern in the Scale-Location plot.

Lets now rerun a new model without the outliers. Lets first identify the outliers by calculating the Cook's distance for each observation.

```r
# Identify the outliers
cooksd <- cooks.distance(glm_model_gamma)
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd, na.rm=T),names(cooksd),""))
```



Based on the Cook's distance plot, there are several outliers that are far away from the 0 line. These outliers will be removed from the dataset.

```r
# Remove these outliers based on cooks cooks
extpts <- which(abs(residuals(glm_model_gamma)) > 3*sd(residuals(glm_model_gamma)))
football_data <- football_data[-extpts,]

#Now lets split the data into a training and testing set
set.seed(123)
train_index <- sample(1:nrow(football_data), 0.8*nrow(football_data))
train_data <- football_data[train_index,]
test_data <- football_data[-train_index,]
```

In order to have another fair comparison, lets run the null model again without outliers.

```r
# Fit the model
glm_model_gamma <- glm(Yards ~ 1, data = train_data, family = Gamma(link = "log"))
```

Now lets rerun the model without the outliers.

```r
# Fit the model
reduced_model <- glm(Yards ~ YardsFromOwnGoal + Distance + RB + WR + DefendersInTheBox + BL, data =
summary(reduced_model)
```

```
Call:
glm(formula = Yards ~ YardsFromOwnGoal + Distance + RB + WR +
    DefendersInTheBox + BL, family = Gamma(link = "log"), data = train_data)

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.846691   0.085400  21.624  < 2e-16 ***
YardsFromOwnGoal  -0.001445   0.000190  -7.606 2.98e-14 ***
Distance           0.013320   0.001304  10.217  < 2e-16 ***
RB                 0.007757   0.012486   0.621   0.5344
WR                -0.007886   0.009142  -0.863   0.3883
DefendersInTheBox -0.095197   0.006802 -13.996  < 2e-16 ***
BL                 0.026601   0.010603   2.509   0.0121 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Gamma family taken to be 0.3698381)

    Null deviance: 7311.3  on 17519  degrees of freedom
Residual deviance: 7038.5  on 17513  degrees of freedom
AIC: 75079

Number of Fisher Scoring iterations: 5
```

Running this reduced model yields the following results:

The regression model was fitted to predict Yards using several predictor variables. The intercept, representing the expected value of Yards when all other predictors are zero, is highly significant (t = 21.624, p < 2e-16), indicating that there is a substantial baseline value of Yards. This intercept is consistent with the notion that even in the absence of other factors, some yards are expected.

Moving to individual predictors, YardsFromOwnGoal exhibits a negative coefficient (-0.001445), implying that as a team moves away from their own goal, the expected yards tend to increase. Distance, with a positive coefficient of 0.0.013320 and high significance (t = 10.217, p < 2e-16), suggests that longer distances on the field are associated with greater yards gained. These results align with intuitive expectations in football, where advancing towards the opponent's goal typically results in more yards gained.

However, not all predictors are equally influential. RB, representing a certain player position, lacks significance (t = 0.621, p = 0.5344), indicating that the presence of RB does not significantly contribute to the variation in Yards. WR, on the other hand, has a marginally significant negative impact (t = -0.863, p = 0.3883), suggesting that having more Wide Receivers on the field might be associated with fewer yards gained. This result should be interpreted cautiously due to its marginal significance.

DefendersInTheBox, with a negative coefficient of -0.095197 and high significance (t = -13.996, p < 2e-16), implies that having more defenders close to the line of scrimmage is associated with a decrease in yards gained. BL (denoting backline), with a positive coefficient of 0.026601 and significance (t = 2.509, p = 0.0121), suggests that a strong backline presence contributes positively to the yards gained.

Now we can test for model adequacy. This will be done by comparing the fitted full model with the null model. We do this to see whether all the predictors taken together are useful for explaining yards above the intercept alone. The null hypothesis is that the null model is just as adequate as the reduced model. On the other hand, the alternative hypothesis is that the reduced model is more adequate than the null model. The confidence level for this test, α, is 0.05.

```r
with(reduced_model, cbind(deviance = null.deviance - deviance,
                          df = df.null - df.residual,
                          p = pchisq(null.deviance - deviance,
                          df.null - df.residual,
                          lower.tail = FALSE)))
```

```
     deviance df            p
[1,] 272.7759  6 5.525254e-56
```

With a p-value of $ 5.525e^(-56) $, which is less than the confidence level of 0.05, we reject the null hypothesis. This means that the reduced model is more adequate than the null model. In other words, the predictors in the reduced model are useful for explaining yards above the intercept alone.

We can also compare the Akaike Information Criterion (AIC) from the fitted reduced model and the null model. The better model is the one which gives the smaller AIC:

```r
AIC(glm_model_gamma, reduced_model)
```

```
                df      AIC
glm_model_gamma  2 75777.76
reduced_model    8 75078.85
```

The AIC for the null model is 75777.76. The AIC for the reduced model is 75078.85. The reduced model has a smaller AIC than the null model. This means that the reduced model is a better fit for the data than the null model.

Testing for overdispersion, we can calculate the dispersion parameter for the fitted model. The dispersion parameter is the ratio of the residual deviance to the residual degrees of freedom. The dispersion parameter should be close to 1. If it is much larger than 1, then the model is overdispersed. If it is much smaller than 1, then the model is underdispersed.

```r
(disp.est <- reduced_model$deviance/reduced_model$df.residual)
```

```
[1] 0.4019009
```

With a dispersion parameter of 0.4019009, which is less than 1, the model is underdispersed. This means that the model is not accounting for all the variability in the data.

Turning attention to the performance of the test set on the model, we will predict the yards gained for the test set using the model that was fit on the training set.

```r
# Predict the yards gained for the test set
predicted_yards <- predict(reduced_model, newdata = test_data, type = "response")
```

Now lets calculate the mean squared error and the mean absolute error for the test set.

```r
# Calculate the mean squared error
mse <- mean((test_data$Yards - predicted_yards)^2)
mse
```

```
[1] 5.515282
```

```r
# Calculate the mean absolute error
mae <- mean(abs(test_data$Yards - predicted_yards))
mae
```

```
[1] 1.909325
```

The mean squared error for the test set is 5.515282 while the mean absolute error for the test set is 1.909325. These values are relatively low, indicating that the model is a good fit for the data.

6. Summary and Conclusion

In this project, we developed a predictive model to forecast the yards gained after a handoff in NFL games. We began by exploring the data and conducting a comprehensive exploratory data analysis (EDA) to understand the underlying structure and characteristics of the data. This step involved visualizing the distribution of key variables, identifying patterns and outliers, and exploring correlations and interactions between predictors. Data visualization, through techniques like scatter plots, histograms, and heatmaps, offered an intuitive understanding of data distributions, correlations, and potential clusters within the dataset. EDA-informed feature selection and engineering strategies highlighted potential predictors that are most informative of yards gained after a handoff.

During preprocessing, we also undertook feature engineering to create new variables that may have a stronger relationship with the target variable. This included interaction terms that capture the combined effect of two predictors, polynomial features for capturing non-linear relationships, and domain-specific features that encapsulate strategic elements of the game.

With a clean and prepared dataset, we then proceeded to develop our MLR, GLM, and XGBoost models. For MLR and GLM, the model's performance was evaluated using metrics such as the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). For the XGBoost model, along with MSE and MAE, we assessed performance using additional metrics like the R-squared for regression tasks and feature importance scores to understand which variables are most predictive.

The final step was to interpret the models in the context of NFL games. This involved translating the statistical outputs into actionable insights for coaches and team strategists, providing recommendations on how to leverage the results for competitive advantage in play-calling and player evaluation.

The model picked as the best model for determining yards per carry was selected based on both performance and interpretability.

Interpretability is critical in the context of NFL games, where the model's outputs must be translated into actionable insights for coaches and team strategists. The GLM with a Gamma response was selected as the best model for its interpretability and performance. The GLM with a Gamma response is a powerful extension of the linear model framework, allowing for response variables that have error distribution models other than a normal distribution. GLMs are particularly useful when dealing with non-normal response variables. With the GLM model, we can directly see the affects each predictor has on the outcome of the play. For instance, the coefficient for each predictor can be interpreted as the change in the response variable for a unit change in the predictor variable, while holding all other predictors constant. A negative coefficient indicates that as the predictor variable increases, the response variable decreases. On the other hand, a positive coefficient indicates that as the predictor variable increases, the response variable increases. Therefore, predictors with a positive coefficient contribute to the yards per carry increasing, while the predictors with a negative coefficient contribute to the yards per carry decreasing.

As for performance, the GLM with a Gamma response had a mean squared error of 5.515282 and a mean absolute error of 1.909325. These values are relatively low, indicating that the model is a good fit for the data. Compared to the linear model, the GLM with a Gamma response had a lower mean squared error and a lower mean absolute error. This indicates that the GLM with a Gamma response is a better fit for

the data than the linear model. However, the XGBoost model had a lower mean squared error and a lower mean absolute error than the GLM with a Gamma response. This indicates that the XGBoost model is a better fit for the data than the GLM with a Gamma response. The issue of the XGBost model is that it is not as interpretable as the GLM with a Gamma response. Therefore, the GLM with a Gamma response was selected as the best model for determining yards per carry.

Based on the data gathered above, the GLM model states that the higher number of defenders that are in the box (the number defenders employed in an area close enough to the line of scrimmage where they can directly impact a rushing attempt by the offense) increases, the yards gained per carry decreases. Therefore, running the ball should not be attempted when there are a high number of defenders in the box. Any play with more than 6 defenders in the box will result in a play with less than average number of yards. In other words, the offense should only run the ball when there are 6 or more defenders in the box when only attempting to gain a yard or less.

Furthermore, as the offense gets closer to the endzone, the yards gained per carry decreases. This is due to the fact that there is less yardage for the offense to gain. In addition, as the distance to the endzone decreases, the defense will have more defenders in a smaller area. This means that safeties and cornerbacks will be closer to the line of scrimmage, making it more difficult for the offense to gain yards, or for breakout plays to occur.

7. Appendix

7.1. Multiple Linear Regression

First, lets start by importing the completely clean and ready to use dataset. This dataset was the result of procedures outlined in the data cleaning and preprocessing section.

```
# Import the dataset
football_data <- read.csv("/Users/samlutz10/Desktop/STAT5405/Final Project/train_ready.csv")

str(football_data)
```

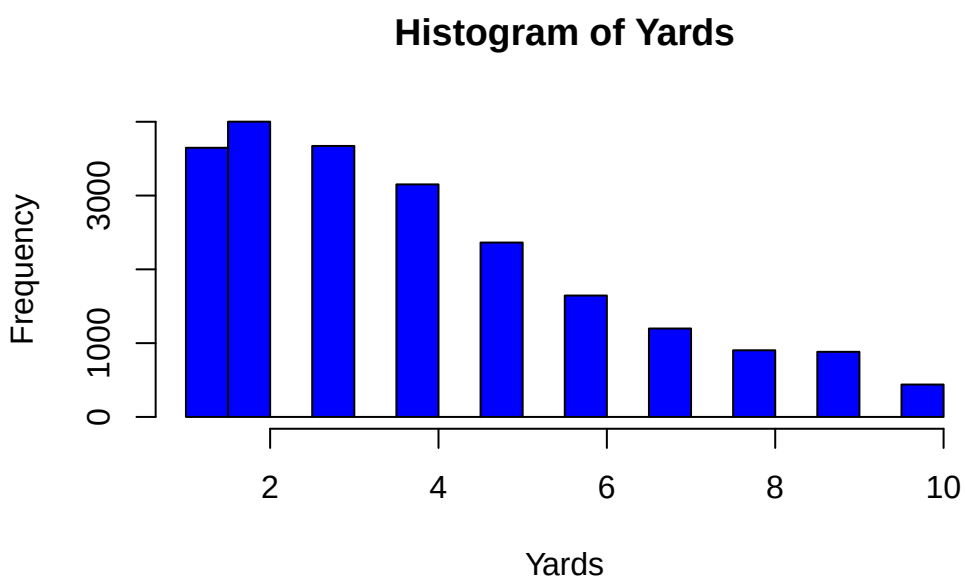
```
'data.frame':   31007 obs. of  21 variables:
 $ Quarter          : int  2 2 4 2 2 1 2 4 4 4 ...
 $ Down             : int  1 1 1 2 1 1 1 1 1 1 ...
 $ Distance         : int  10 10 10 10 10 10 10 10 10 10 ...
 $ OffenseFormation : chr  "SINGLEBACK" "I_FORM" "I_FORM" "SINGLEBACK" ...
 $ RB               : int  2 2 2 1 1 1 1 1 2 1 ...
 $ TE               : int  1 1 2 1 1 2 1 1 1 1 ...
 $ WR               : int  2 2 1 3 3 2 3 3 1 3 ...
 $ DefendersInTheBox: int  8 7 8 6 7 8 6 7 7 7 ...
 $ Yards            : int  1 12 3 -2 14 4 3 7 7 3 ...
 $ Week             : int  3 9 14 2 2 2 17 11 17 3 ...
 $ GameWeather      : chr  "clear" "overcast" "clear" "clear" ...
 $ Temperature      : int  84 77 39 81 79 79 19 68 71 68 ...
 $ Humidity         : int  25 62 55 45 48 48 36 54 66 42 ...
 $ WindSpeed        : num  5 16 11 5 5 5 12 4 9 13 ...
 $ WindDirection    : chr  "west" "south east" "none" "east" ...
 $ GameHour         : int  15 9 3 3 13 12 2 6 15 12 ...
 $ DL               : int  4 3 4 4 2 4 2 4 4 4 ...
 $ LB               : int  3 4 3 2 4 3 4 2 3 2 ...
 $ BL               : int  4 4 4 5 5 4 5 5 4 5 ...
 $ YardsFromOwnGoal : int  40 84 79 21 86 25 49 25 65 54 ...
 $ ScoreDelta       : int  -7 0 -13 -11 0 7 -11 41 -16 -8 ...
```

Since we are only focused on the most common plays and not breakout runs, we will remove any data of 10 yards or more. This is due to the fact that plays resulting in 10 yards are considered to be just as successful are play resulting in more than 10 yards. In addition, any plays resulting in 0 or negative yards will be removed. This is due to the fact that these plays are considered to be unsuccessful.

```
# Remove any rows whose yards column is greater than 30
football_data <- football_data[football_data$Yards <= 10 & football_data$Yards > 0, ]
```

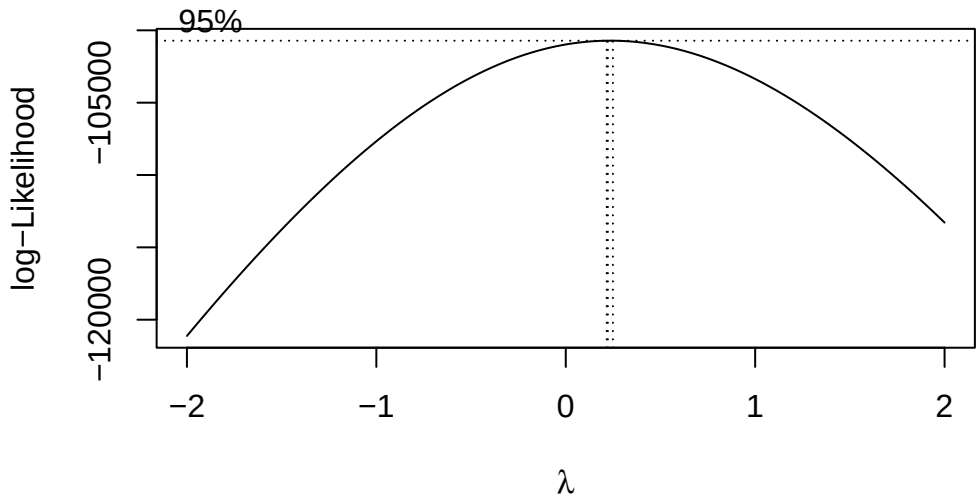
Lets now look at the histogram of Yards to get a better understanding of the distribution of the response variable.

```
# Histogram of Yards
hist(football_data$Yards, main = "Histogram of Yards", xlab = "Yards", ylab = "Frequency", col = "b
```



Based on the histogram above, it is apparent that the number of yards is skewed to the right. In order to address this issue, the Box-Cox transformation will be utilized. This will help to normalize the distribution of the response variable. The Box-Cox transformation can transform non-normal data into normal data.

```
library(MASS)
lambda_null <- boxcox(lm(Yards ~ 1, data = football_data))
```



The best lambda value for the Box-Cox tranformation is as follows:

```
lambda_null$x[which.max(lambda_null$y)]
```

```
[1] 0.2222222
```

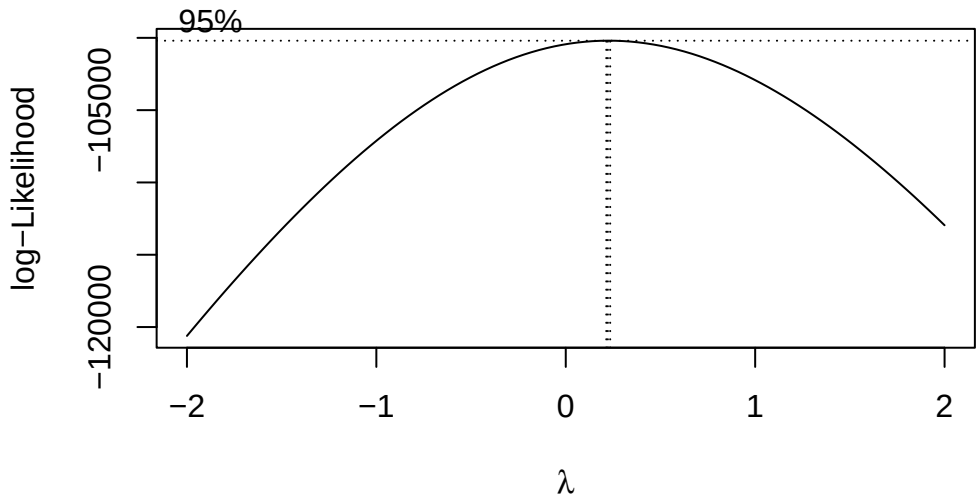
Now the Box-Cox tranformation will be applied to the variable Yards with the best lambda value.

```
lm_model_null <- lm((Yards^(lambda_null$x[which.max(lambda_null$y)])) ~ 1, data = football_data)
summary(lm_model_null)
```

```
Call:
lm(formula = (Yards^(lambda_null$x[which.max(lambda_null$y)])) ~
    1, data = football_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.30438 -0.13785 -0.02786  0.12559  0.36373

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3044     0.0013    1003   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1924 on 21900 degrees of freedom
```

Now the full model using all of the predictors will be created. In this full model, there will be 0 interactions between the predictors. The same process using the Box-Cox transformation will be used to normalize the distribution of the response variable.

```
lambda_full <- boxcox(lm(Yards ~ ., data = football_data))
```



The best lambda value for the Box-Cox tranformation is as follows:

```
lambda_full$x[which.max(lambda_full$y)]
```

```
[1] 0.2222222
```

Now the Box-Cox tranformation will be applied to the variable Yards with the best lambda value.

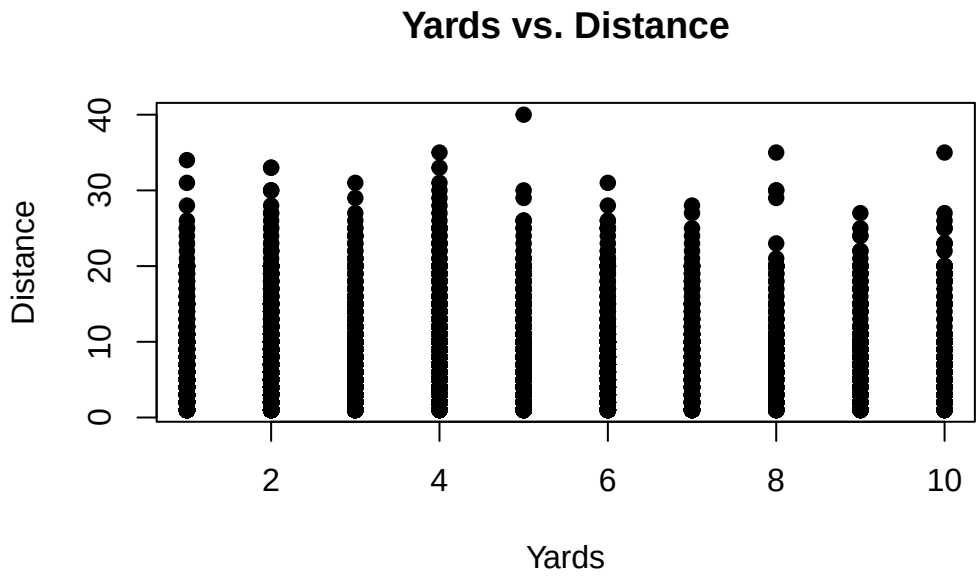
```
lm_model_full <- lm((Yards^(lambda_full$x[which.max(lambda_full$y)])) ~ ., data = football_data)
summary(lm_model_full)
```

```
Call:
lm(formula = (Yards^(lambda_full$x[which.max(lambda_full$y)])) ~
    ., data = football_data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.4747 -0.1367  0.0041  0.1391  0.5492

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                1.533e+00  4.252e-01   3.605 0.000313 ***
Quarter                    2.045e-04  1.118e-03   0.183 0.854945
Down                       5.107e-03  2.244e-03   2.276 0.022858 *
Distance                   3.834e-03  4.131e-04   9.281  < 2e-16 ***
OffenseFormationEMPTY      2.448e-02  1.924e-01   0.127 0.898746
OffenseFormationI_FORM    -1.747e-02  1.882e-01  -0.093 0.926043
OffenseFormationJUMBO     -1.027e-01  1.885e-01  -0.545 0.585728
OffenseFormationPISTOL    -1.521e-02  1.883e-01  -0.081 0.935626
OffenseFormationSHOTGUN   -2.954e-02  1.882e-01  -0.157 0.875255
OffenseFormationSINGLEBACK -1.847e-02  1.882e-01  -0.098 0.921801
OffenseFormationUNKNOWN   -4.269e-02  2.104e-01  -0.203 0.839222
OffenseFormationWILDCAT   -1.580e-04  1.898e-01  -0.001 0.999336
RB                        -3.485e-03  5.875e-03  -0.593 0.553054
```

```
TE                          -1.421e-03  4.898e-03  -0.290 0.771792
WR                          -6.871e-03  4.992e-03  -1.377 0.168679
DefendersInTheBox           -2.749e-02  1.945e-03 -14.135  < 2e-16 ***
Week                        -2.282e-04  3.435e-04  -0.664 0.506415
GameWeathernone              7.253e-03  4.231e-03   1.714 0.086506 .
GameWeatherovercast         -3.271e-04  3.110e-03  -0.105 0.916235
GameWeatherrain             -4.791e-03  6.303e-03  -0.760 0.447222
GameWeathersnow             -8.656e-03  1.815e-02  -0.477 0.633395
Temperature                 -3.019e-04  1.007e-04  -2.997 0.002725 **
Humidity                    -2.216e-05  6.295e-05  -0.352 0.724800
WindSpeed                    1.854e-05  3.094e-04   0.060 0.952216
WindDirectionnone           -9.389e-03  8.146e-03  -1.153 0.249086
WindDirectionnorth          -2.802e-03  9.503e-03  -0.295 0.768153
WindDirectionnorth east     -8.067e-03  8.197e-03  -0.984 0.325065
WindDirectionnorth west     -4.240e-03  8.276e-03  -0.512 0.608408
WindDirectionsouth          -7.687e-03  9.009e-03  -0.853 0.393538
WindDirectionsouth east     -4.907e-03  8.529e-03  -0.575 0.565104
WindDirectionsouth west     -5.235e-03  8.041e-03  -0.651 0.515082
WindDirectionwest           -5.371e-03  9.581e-03  -0.561 0.575079
GameHour                     5.345e-04  2.913e-04   1.835 0.066531 .
DL                          -2.015e-03  3.461e-02  -0.058 0.953559
LB                          -3.061e-03  3.469e-02  -0.088 0.929677
BL                           5.420e-03  3.469e-02   0.156 0.875840
YardsFromOwnGoal            -4.017e-04  5.336e-05  -7.529 5.33e-14 ***
ScoreDelta                  -2.177e-04  1.202e-04  -1.812 0.070023 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1881 on 21863 degrees of freedom
Multiple R-squared:  0.04641,   Adjusted R-squared:  0.04479
F-statistic: 28.76 on 37 and 21863 DF,  p-value: < 2.2e-16
```

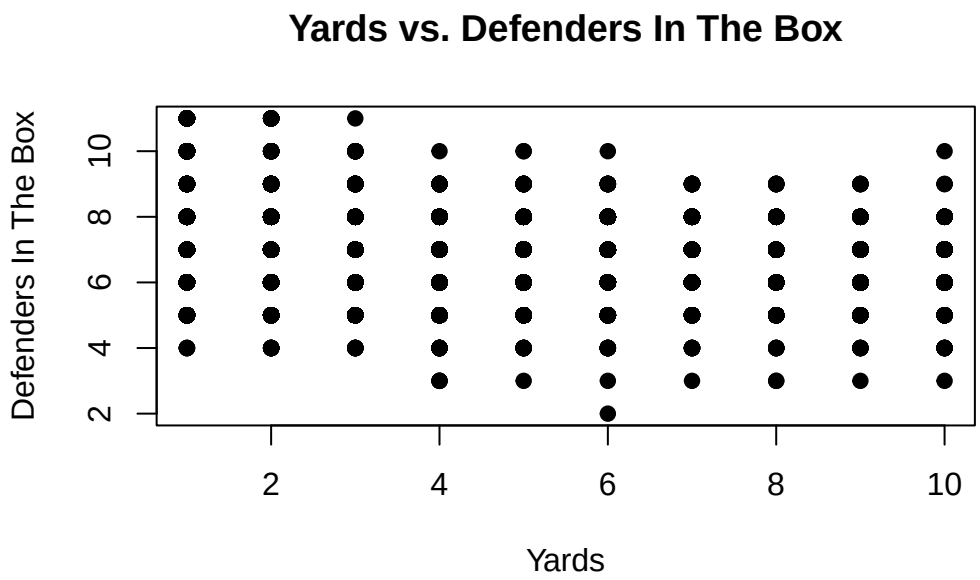
Based on the results of the full model, the model shows which predictors are statistically significant. The most significant predictors are Distance, DefendersInTheBox, and YardsFromOwnGoal. Creating scatter plots of the response variable versus each of these predictors will help to gain a better understanding of how these variables affect the predicted yards.

```r
# Scatter plot of yards vs. Distance
plot(football_data$Yards, football_data$Distance, main = "Yards vs. Distance", xlab = "Yards", ylab
```



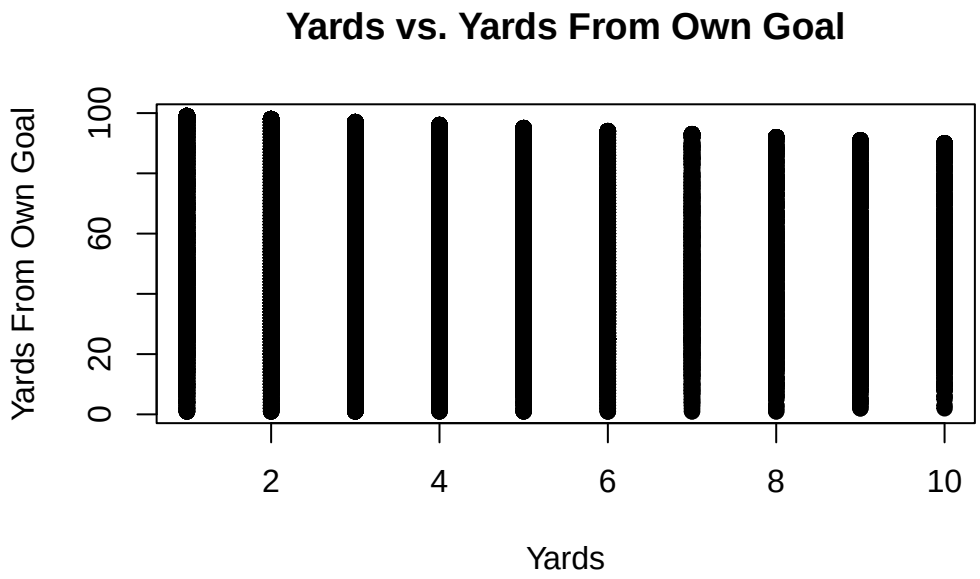
Looking at this scatter plot of Yards vs Distance, there appears to only be a sinusoial relationship between the two variables. Therefore, when a reduced model is created, a sine function will be used to model the relationship between the two variables.

```
# Scatter plot of yards vs. Defenders In The Box
plot(football_data$Yards, football_data$DefendersInTheBox, main = "Yards vs. Defenders In The Box",
```



Looking at this scatter plot of Yards vs Defenders In The Box, there appears to be a quadratic function relationship between the two variables. Therefore, when a reduced model is created, the Defenders In The Box variable will be squared in order to capture this relationship

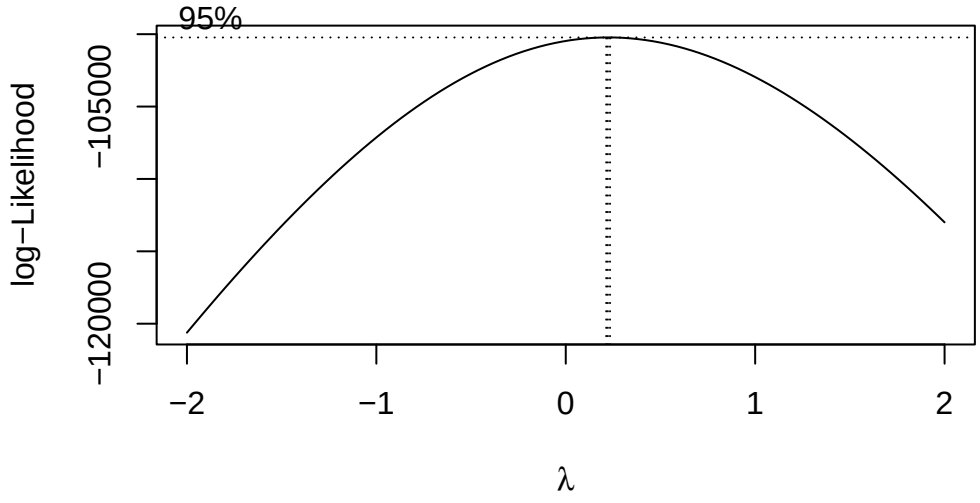
```r
# Scatter plot of yards vs. YardsFromOwnGoal
plot(football_data$Yards, football_data$YardsFromOwnGoal, main = "Yards vs. Yards From Own Goal", x
```



The scatterplot of Yards vs Yards From Own Goal shows that there is a linear relationship between the two variables. Therefore, when a reduced model is created, the Yards From Own Goal variable will not be changed since the model is already capturing this linear relationship.

In order to capture these relationships between variables, a new, full model will be synthesized. This model will include the Distance variable with a sine function and the Defenders In The Box variable squared. However, the Box-Cox transformation will need to be run again.

```r
lambda_full <- boxcox(lm(Yards ~ Quarter + Down + sin(Distance) + OffenseFormation + RB + TE + WR +
```



```
summary(lambda_full)
```

```
  Length Class  Mode
x 100    -none- numeric
y 100    -none- numeric
```

The best lambda value for the Box-Cox tranformation is as follows:

```
lambda_full$x[which.max(lambda_full$y)]
```

```
[1] 0.2222222
```

Now the Box-Cox tranformation will be applied to the variable Yards with the best lambda value.

```
lm_model_full <- lm((Yards^(lambda_full$x[which.max(lambda_full$y)])) ~ Quarter + Down + sin(Distan
summary(lm_model_full)
```

```
Call:
lm(formula = (Yards^(lambda_full$x[which.max(lambda_full$y)])) ~
    Quarter + Down + sin(Distance) + OffenseFormation + RB +
        TE + WR + DefendersInTheBox^2 + Week + GameWeather +
        Temperature + Humidity + WindSpeed + WindDirection +
        GameHour + DL + LB + BL + YardsFromOwnGoal + ScoreDelta,
    data = football_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43047 -0.13761  0.00579  0.13964  0.54723

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                1.572e+00  4.257e-01   3.693 0.000222 ***
Quarter                    4.560e-04  1.119e-03   0.407 0.683760
Down                       6.823e-04  2.260e-03   0.302 0.762764
sin(Distance)             -1.309e-02  2.557e-03  -5.118 3.11e-07 ***
OffenseFormationEMPTY      1.810e-02  1.926e-01   0.094 0.925163
OffenseFormationI_FORM    -2.044e-02  1.885e-01  -0.108 0.913627
OffenseFormationJUMBO     -1.057e-01  1.888e-01  -0.560 0.575549
OffenseFormationPISTOL    -1.896e-02  1.886e-01  -0.101 0.919903
OffenseFormationSHOTGUN   -3.184e-02  1.884e-01  -0.169 0.865822
OffenseFormationSINGLEBACK -2.233e-02  1.884e-01  -0.119 0.905657
OffenseFormationUNKNOWN   -4.517e-02  2.107e-01  -0.214 0.830241
OffenseFormationWILDCAT   -3.616e-03  1.901e-01  -0.019 0.984823
RB                        -3.657e-03  5.883e-03  -0.622 0.534153
TE                        -9.796e-04  4.904e-03  -0.200 0.841678
WR                        -7.130e-03  4.999e-03  -1.426 0.153781
DefendersInTheBox         -2.953e-02  1.930e-03 -15.301  < 2e-16 ***
Week                      -1.953e-04  3.439e-04  -0.568 0.570043
GameWeathernone            7.612e-03  4.237e-03   1.797 0.072415 .
GameWeatherovercast       -2.761e-04  3.114e-03  -0.089 0.929343
GameWeatherrain           -4.145e-03  6.312e-03  -0.657 0.511335
```

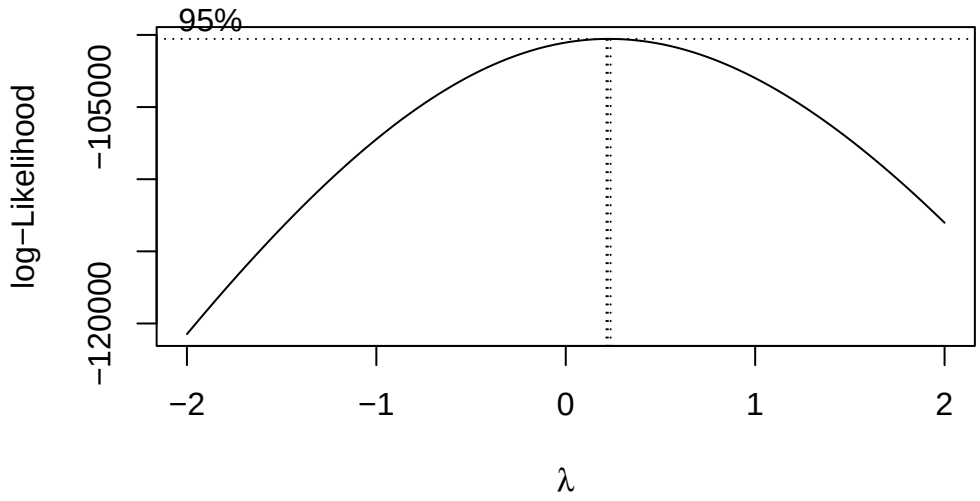
```
GameWeathersnow            -7.028e-03  1.817e-02  -0.387 0.698943
Temperature                -2.819e-04  1.008e-04  -2.795 0.005193 **
Humidity                   -3.439e-05  6.302e-05  -0.546 0.585250
WindSpeed                   7.867e-05  3.097e-04   0.254 0.799487
WindDirectionnone          -8.918e-03  8.157e-03  -1.093 0.274260
WindDirectionnorth         -2.268e-03  9.516e-03  -0.238 0.811644
WindDirectionnorth east    -7.485e-03  8.208e-03  -0.912 0.361843
WindDirectionnorth west    -3.958e-03  8.287e-03  -0.478 0.632975
WindDirectionsouth         -7.636e-03  9.021e-03  -0.846 0.397306
WindDirectionsouth east    -4.302e-03  8.541e-03  -0.504 0.614508
WindDirectionsouth west    -5.156e-03  8.052e-03  -0.640 0.521962
WindDirectionwest          -4.050e-03  9.593e-03  -0.422 0.672902
GameHour                    4.157e-04  2.914e-04   1.427 0.153660
DL                         -1.108e-03  3.465e-02  -0.032 0.974487
LB                         -1.580e-03  3.473e-02  -0.045 0.963712
BL                          7.304e-03  3.473e-02   0.210 0.833457
YardsFromOwnGoal           -4.871e-04  5.229e-05  -9.315  < 2e-16 ***
ScoreDelta                 -2.299e-04  1.203e-04  -1.911 0.055994 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1883 on 21863 degrees of freedom
Multiple R-squared:  0.0438,    Adjusted R-squared:  0.04218
F-statistic: 27.06 on 37 and 21863 DF,  p-value: < 2.2e-16
```

Now lets use the step function to find the best model for the data.

```
lm_model_step <- step(lm_model_full, direction = "backward")
summary(lm_model_step)
```

This new reduced model needs to be run similar to that of the previous models utilizing the Box-Cox transformation.

```
lambda_reduced <- boxcox(lm(Yards ~ sin(Distance) + WR + DefendersInTheBox^2 + Temperature + BL + Y
```



The best lambda value for the Box-Cox tranformation is as follows:

```
lambda_reduced$x[which.max(lambda_reduced$y)]
```

```
[1] 0.2222222
```

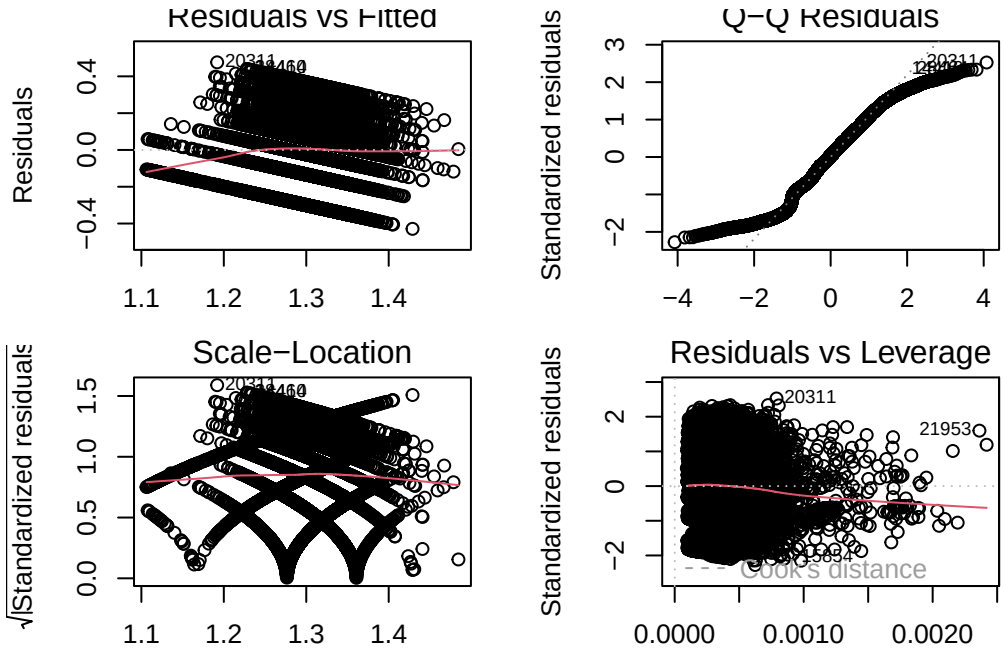
Now the Box-Cox tranformation will be applied to the variable Yards with the best lambda value.

```
lm_model_reduced <- lm((Yards^(lambda_reduced$x[which.max(lambda_reduced$y)])) ~ sin(Distance) + WR
summary(lm_model_reduced)
```

```
Call:
lm(formula = (Yards^(lambda_reduced$x[which.max(lambda_reduced$y)])) ~
    sin(Distance) + WR + DefendersInTheBox^2 + Temperature +
        BL + YardsFromOwnGoal, data = football_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.42892 -0.13844  0.00668  0.14027  0.47618

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.518e+00  2.254e-02  67.351  < 2e-16 ***
sin(Distance)     -1.536e-02  2.198e-03  -6.989 2.85e-12 ***
WR                -5.522e-03  2.406e-03  -2.295 0.021750 *
DefendersInTheBox -3.049e-02  1.865e-03 -16.345  < 2e-16 ***
Temperature       -2.167e-04  7.435e-05  -2.914 0.003574 **
BL                 9.724e-03  2.939e-03   3.309 0.000939 ***
YardsFromOwnGoal  -5.382e-04  5.167e-05 -10.417  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1886 on 21894 degrees of freedom
Multiple R-squared:  0.0394,    Adjusted R-squared:  0.03914
F-statistic: 149.7 on 6 and 21894 DF,  p-value: < 2.2e-16
```

By creating a Residuals vs Fitted plot, a Q-Q Residuals plot, a Scale-Location plot, and a Residuals vs Leverage plot, the assumptions of the model can be checked.

```
par(mfrow = c(2, 2))
old.par = par(mar = c(3, 4, 1, 2))
plot(lm_model_reduced)
```

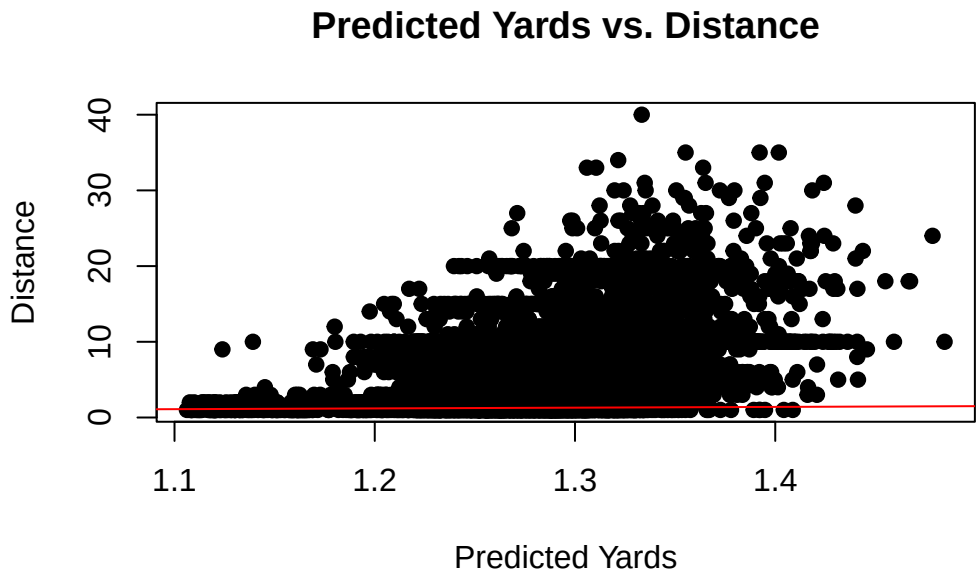


Based on the plots above, the normality assumption is not violated, as the Q-Q Residuals plot shows that the residuals are mainly normally distributed.

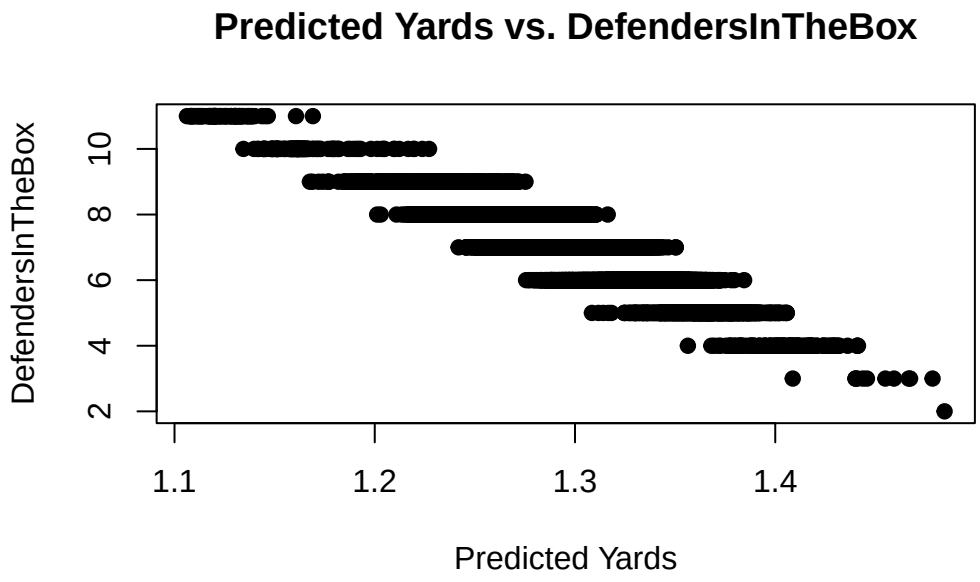
In order to gain a better understanding of how these variables affect the predicted yards, scatter plots of the predicted yards versus each of the predictors will be created. However, the only scatter plots that will be synthesized are those that have the three highest t values: Distance, DefendersInTheBox, and YardsFromOwnGoal.

```
football_data$predicted_yards <- predict(lm_model_reduced, football_data)
```

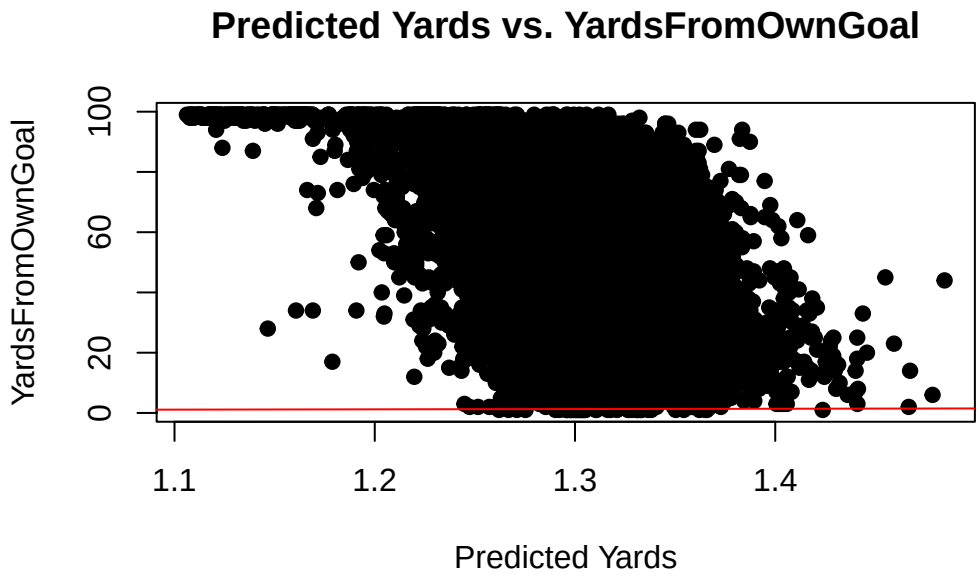
```
# Scatter plot of predicted yards vs. Distance
plot(football_data$predicted_yards, football_data$Distance, main = "Predicted Yards vs. Distance",
abline(0, 1, col = "red")  # Adds a 45-degree line
```




```r
# Scatter plot of predicted yards vs. DefendersInTheBox
plot(football_data$predicted_yards, football_data$DefendersInTheBox, main = "Predicted Yards vs. De
abline(0, 1, col = "red")  # Adds a 45-degree line
```




```r
# Scatter plot of predicted yards vs. YardsFromOwnGoal
plot(football_data$predicted_yards, football_data$YardsFromOwnGoal, main = "Predicted Yards vs. Yar
abline(0, 1, col = "red")  # Adds a 45-degree line
```



7.2. Decision-Tree Model

First, lets start by importing the completely clean and ready to use dataset. This dataset was the result of procedures outlined in the data cleaning and preprocessing section.

```
# Import the dataset
football_data <- read.csv("/Users/samlutz10/Desktop/STAT5405/Final Project/train_ready.csv")

str(football_data)
```

```
'data.frame':   31007 obs. of  21 variables:
 $ Quarter          : int  2 2 4 2 2 1 2 4 4 4 ...
 $ Down             : int  1 1 1 2 1 1 1 1 1 1 ...
 $ Distance         : int  10 10 10 10 10 10 10 10 10 10 ...
 $ OffenseFormation : chr  "SINGLEBACK" "I_FORM" "I_FORM" "SINGLEBACK" ...
 $ RB               : int  2 2 2 1 1 1 1 2 1 ...
 $ TE               : int  1 1 2 1 1 2 1 1 1 ...
 $ WR               : int  2 2 1 3 3 2 3 3 1 3 ...
 $ DefendersInTheBox: int  8 7 8 6 7 8 6 7 7 7 ...
 $ Yards            : int  1 12 3 -2 14 4 3 7 7 3 ...
 $ Week             : int  3 9 14 2 2 2 17 11 17 3 ...
 $ GameWeather      : chr  "clear" "overcast" "clear" "clear" ...
 $ Temperature      : int  84 77 39 81 79 79 19 68 71 68 ...
 $ Humidity         : int  25 62 55 45 48 48 36 54 66 42 ...
 $ WindSpeed        : num  5 16 11 5 5 5 12 4 9 13 ...
 $ WindDirection    : chr  "west" "south east" "none" "east" ...
 $ GameHour         : int  15 9 3 3 13 12 2 6 15 12 ...
 $ DL               : int  4 3 4 4 2 4 2 4 4 4 ...
 $ LB               : int  3 4 3 2 4 3 4 2 3 2 ...
 $ BL               : int  4 4 4 5 5 4 5 5 4 5 ...
 $ YardsFromOwnGoal : int  40 84 79 21 86 25 49 25 65 54 ...
 $ ScoreDelta       : int  -7 0 -13 -11 0 7 -11 41 -16 -8 ...
```

Since we are only focused on the most common plays and not breakout runs, we will remove any data of 10 yards or more. This is due to the fact that plays resulting in 10 yards are considered to be just as successful are play resulting in more than 10 yards. In addition, any plays resulting in 0 or negative yards will be removed. This is due to the fact that these plays are considered to be unsuccessful.

```
# Remove any rows whose yards column is greater than 30
football_data <- football_data[football_data$Yards <= 10 & football_data$Yards > 0, ]
```

Decision Trees, Random Forest and XGBoost models are all tree-based models used as part of the standard for sports analytics. Therefore, we will go through the process of building a decision tree model, then a random forest model, and finally an XGBoost model to get a better understanding of our data and help with our analysis.

Our initial step involves preparing the data for analysis. Recognizing the importance of having a representative sample for both training and testing, we employed stratified sampling. This technique ensures that our training and testing sets mirror the distribution of 'Yards' in the complete dataset. By maintaining this distribution, we aim to build models that are well-generalized and reflective of real-world scenarios.

```
library(rpart)
library(randomForest)
```

```
randomForest 4.7-1.1
```

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

```
library(xgboost)
library(caret)
```

```
Loading required package: ggplot2
```

```

Attaching package: 'ggplot2'
```

```
The following object is masked from 'package:randomForest':

    margin
```

```
Loading required package: lattice
```

```
library(rpart.plot)
library(fastDummies)
```

```
Thank you for using fastDummies!
```

```
To acknowledge our work, please cite the package:
```

```
Kaplan, J. & Schlegel, B. (2023). fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from C
```

```
# Convert specified factor columns to dummy variables
football_data <- dummy_cols(football_data,
                            select_columns = c("WindDirection", "GameWeather", "OffenseFormation"),
                            remove_selected_columns = TRUE)
```

```r
# Split the data into a training set and a testing set
set.seed(123457)
train.prop <- 0.80
strats <- football_data$Yards
rr <- split(1:length(strats), strats)
idx <- sort(as.numeric(unlist(sapply(rr,
        function(x) sample(x, length(x)*train.prop)))))
train.set <- football_data[idx, ]
test.set <- football_data[-idx, ]

names(train.set) <- make.names(names(train.set), unique = TRUE)
names(test.set) <- make.names(names(test.set), unique = TRUE)
```

Once the data is split, we delve into verifying the balance between our training and testing sets. It's crucial that the average 'Yards' in these sets closely align with the overall dataset's average. This alignment indicates that our stratified sampling was effective and that both sets are indeed representative, eliminating biases that might skew our model training and validation.

```r
# Check that the average of yards is about the same in both the training and testing sets
(ave_train <- mean(train.set$Yards))
```

```
[1] 3.875942
```

```r
(ave_test <- mean(test.set$Yards))
```

```
[1] 3.877081
```

```r
(ave_all <- mean(football_data$Yards))
```

```
[1] 3.87617
```

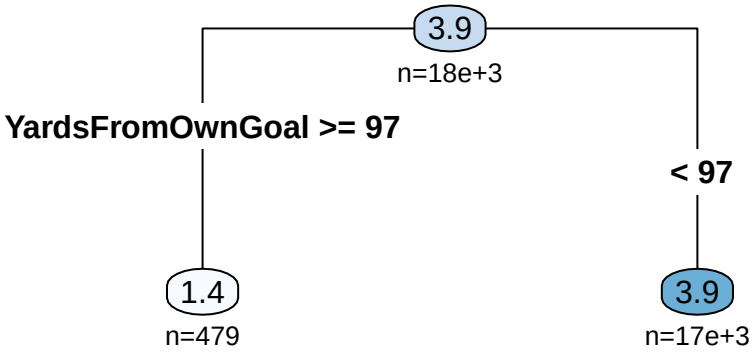
Moving forward, we construct a Decision Tree model. The beauty of a Decision Tree lies in its simplicity and interpretability. After training the model, we assess its performance using the Mean Squared Error (MSE) on the test set. The RMSE offers a clear metric to gauge the accuracy of our predictions, guiding us in refining our model further.

```r
# Decision Tree Model
dt_model <- rpart(Yards ~ ., data = train.set, method = "anova",
                  control = rpart.control(maxdepth = 5, cp = 0.01, minsplit = 20))
dt_predictions <- predict(dt_model, test.set)

# Evaluate the models
dt_rmse <- sqrt(mean((test.set$Yards - dt_predictions)^2))
dt_rmse
```

```
[1] 2.348833
```

To gain deeper insights into the decision-making logic of our model, we visualize the Decision Tree. This visualization not only aids in interpreting the model but also serves as a tool to communicate our findings effectively to stakeholders who might be less familiar with machine learning intricacies.

```r
# Plot the decision tree
rpart.plot(dt_model, type = 4, extra = 1, under = TRUE, faclen = 0)
```



Upon evaluating our Decision Tree model, we observed its simplicity - it comprises only one split. While simplicity can be a virtue, in this case, it indicates that our model may be overly simplistic to capture the complexities inherent in our dataset. This simplicity could stem from the nuanced nature of the 'Yards' variable in the NFL Big Data Bowl dataset, which might require a more sophisticated approach to uncover the underlying patterns effectively.

Given this situation, we turn our attention to the Random Forest model. A Random Forest is essentially an ensemble of Decision Trees. This ensemble approach amalgamates the predictions from multiple trees, thereby enhancing the model's ability to capture complex relationships and reducing the risk of overfitting, which is a common challenge with individual Decision Trees. By leveraging the collective wisdom of multiple trees, a Random Forest can often achieve higher accuracy and better generalization to new, unseen data.

We proceed to build and evaluate a Random Forest model on our dataset, hoping to achieve better predictive performance than our initial simplistic Decision Tree.

```r
# Building the Random Forest model
rf_model <- randomForest(Yards ~ ., data = train.set, ntree = 300., mtry = 3)

# Predicting on the test set
rf_predictions <- predict(rf_model, test.set)

# Evaluating the model using Root Mean Squared Error
rf_rmse <- sqrt(mean((test.set$Yards - rf_predictions)^2))

# Output the Root Mean Squared Error
rf_rmse
```

```
[1] 2.339031
```

```r
# Getting the importance of each variable based on IncNodePurity
feature_importance <- importance(rf_model)

# Sorting features by IncNodePurity and selecting the top 3
top_features <- sort(feature_importance[, 'IncNodePurity'], decreasing = TRUE)[1:3]
```

```r
# Output the top 3 important features
top_features
```

```
YardsFromOwnGoal       ScoreDelta      Temperature
        5887.599         3891.366         3719.057
```

Our models top 3 import predictors are YardsFromOwnGoal, Temperature and Humidity. Before commenting on these results, we will build an XGBoost model to see if we can improve our MSE.

Next, we turn our focus to preparing the data for XGBoost, a powerful gradient boosting framework. XGBoost requires the data to be in a matrix format. Thus, we convert our training and testing datasets accordingly, excluding the target variable 'Yards'. This conversion is a critical step in harnessing the full potential of XGBoost's capabilities.

```r
# Prepare the data for XGBoost
# Convert data to matrix format as xgboost works with the matrix
train_matrix <- as.matrix(train.set[, -which(names(train.set) == "Yards")])
test_matrix <- as.matrix(test.set[, -which(names(test.set) == "Yards")])

# Create DMatrices
dtrain <- xgb.DMatrix(data = train_matrix, label = train.set$Yards)
dtest <- xgb.DMatrix(data = test_matrix, label = test.set$Yards)
```

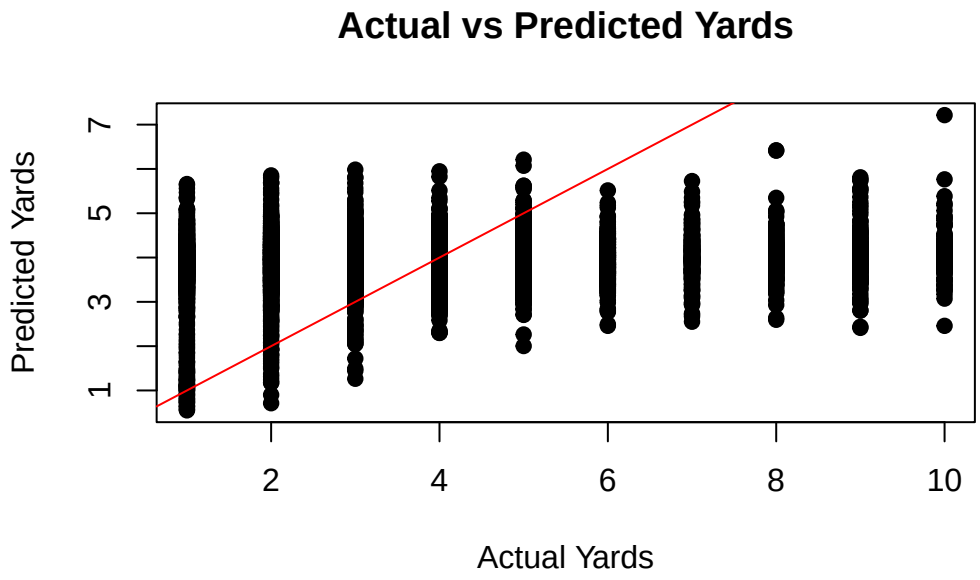
Nowe we can run the model:

```r
# Set XGBoost parameters
params <- list(
    booster = "gbtree",
    objective = "reg:squarederror",  # Objective function for regression
    eta = 0.1,                       # Learning rate
    max_depth = 6,                   # Depth of trees
    subsample = 0.5,                 # Subsampling of the training data
    colsample_bytree = 0.5           # Subsampling of features
)

# Number of boosting rounds
nrounds <- 100

# Train the model
xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds)

# Predicting
xgb_predictions <- predict(xgb_model, dtest)

# Evaluate the model
# For example, using Root Mean Squared Error (RMSE)
true_values <- test.set$Yards
rmse <- sqrt(mean((true_values - xgb_predictions)^2))
print(paste("RMSE:", rmse))
```

```
[1] "RMSE: 2.35443715824306"
```

Next we can plot a scatter plot of the actual vs predicted values:

```
# Scatter plot of actual vs. predicted values
plot(true_values, xgb_predictions, main = "Actual vs Predicted Yards", xlab = "Actual Yards", ylab
abline(0, 1, col = "red")  # Adds a 45-degree line
```



As we can see, there are many outliers from plays where the running backs break out and gain a lot of yards and also some outliers from where thee running backs lose more than 5 yards. Similar to the approach used in the previous modeling techniques, we will remove all plays where the running back gained more than 10 yards and lost more than 5. This will help us get a better understanding of the model's performance on the majority of plays. The intuition behind this is two fold. Domain knowledge helps us claim that not many plays end in either breakout or major loss plays, but we can also look at the summary statistics of the dataset to confirm this.

```
summary(football_data$Yards)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.876   5.000  10.000
```

When we check these statistics we can see that 75% of the data results in a gain of 6 yards or less and 75% of the data results in a gain of 1 yard or more. This confirms our intuition, and helps justify the tresholds we set for the outliers.

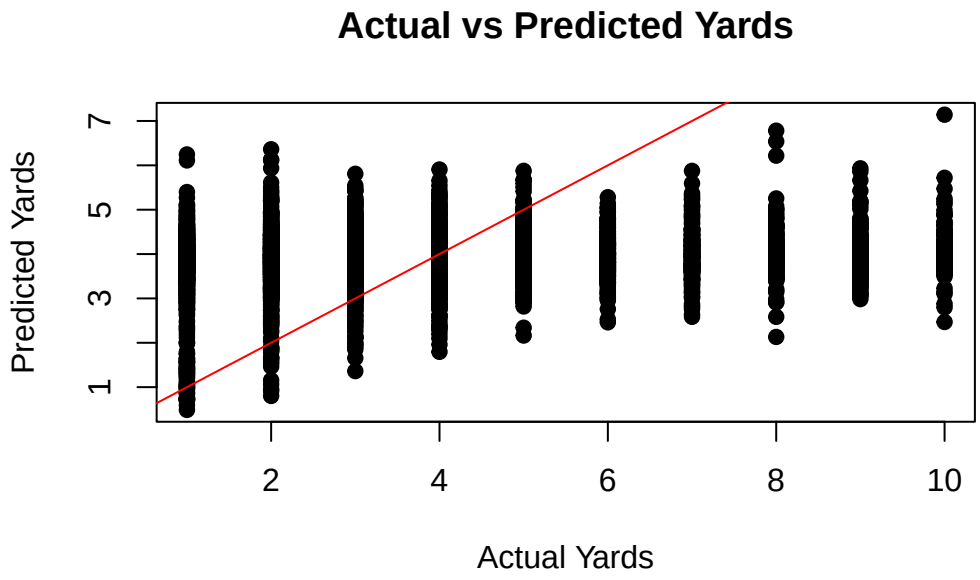
```
# Split the data into a training set and a testing set
set.seed(123457)
train.prop <- 0.80
strats <- football_data$Yards
rr <- split(1:length(strats), strats)
idx <- sort(as.numeric(unlist(sapply(rr,
        function(x) sample(x, length(x)*train.prop)))))
train.set <- football_data[idx, ]
```

```r
test.set <- football_data[-idx, ]

# Prepare the data for XGBoost
# Convert data to matrix format as xgboost works with the matrix
train_matrix <- as.matrix(train.set[, -which(names(train.set) == "Yards")])
test_matrix <- as.matrix(test.set[, -which(names(test.set) == "Yards")])

# Create DMatrices
dtrain <- xgb.DMatrix(data = train_matrix, label = train.set$Yards)
dtest <- xgb.DMatrix(data = test_matrix, label = test.set$Yards)

# Set XGBoost parameters
params <- list(
    booster = "gbtree",
    objective = "reg:squarederror",  # Objective function for regression
    eta = 0.1,                       # Learning rate
    max_depth = 6,                   # Depth of trees
    subsample = 0.5,                 # Subsampling of the training data
    colsample_bytree = 0.5           # Subsampling of features
)

# Number of boosting rounds
nrounds <- 100

# Train the model
xgb_model <- xgb.train(params = params, data = dtrain, nrounds = nrounds)

# Predicting
xgb_predictions <- predict(xgb_model, dtest)

# Evaluate the model
# For example, using Root Mean Squared Error (RMSE)
true_values <- test.set$Yards
rmse <- sqrt(mean((true_values - xgb_predictions)^2))
print(paste("RMSE:", rmse))
```

```
[1] "RMSE: 2.35698973331392"
```

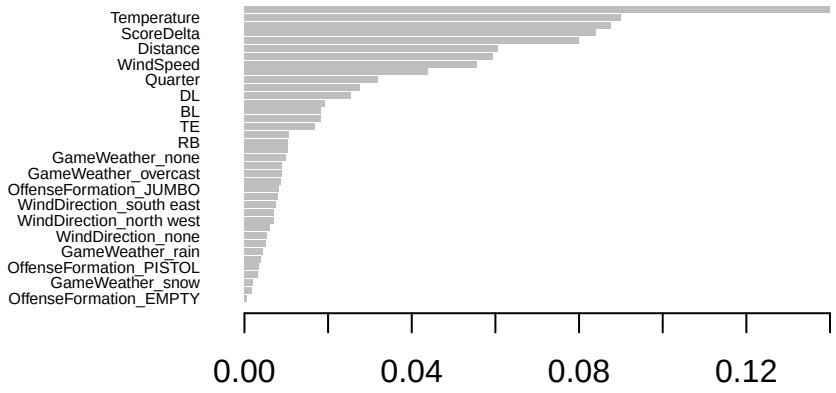
Again we can plot the actual vs predicted values:

```r
# Scatter plot of actual vs. predicted values
plot(true_values, xgb_predictions, main = "Actual vs Predicted Yards", xlab = "Actual Yards", ylab
abline(0, 1, col = "red")  # Adds a 45-degree line
```



We can also check feature importance, mean absolute error and the residuals plot:

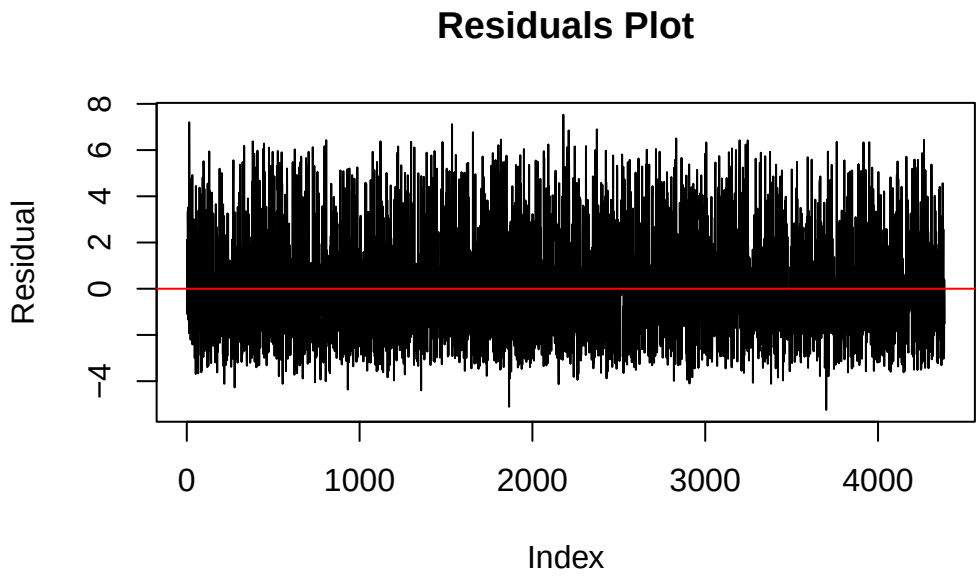
```r
# Variable importance plot
importance_matrix <- xgb.importance(feature_names = colnames(train_matrix), model = xgb_model)
xgb.plot.importance(importance_matrix)
```




```r
# Mean Absolute Error
mae <- mean(abs(true_values - xgb_predictions))
print(paste("MAE:", mae))
```

```
[1] "MAE: 1.90598015938817"
```

```r
# Residuals plot
residuals <- true_values - xgb_predictions
plot(residuals, type = "l", main = "Residuals Plot", xlab = "Index", ylab = "Residual")
abline(h = 0, col = "red")
```



References