NFL RB Rushing Linear Regression Model

Ethan Xu

2023-09-22

Stage 0: Background

I started this project as a way to apply my knowledge from DS 1000 and challenge my abilities. As a lifelong fan of football, I decided that determining what factors affect an running back's total yardage would be an enjoyable and complex challenge. Since football culture revolves heavily around a player's statistics and discussion of said stats, it seemed fitting to apply my data science knowledge to this topic.

I chose a linear regression model, as I wanted a simple way to use continuous data (independent variables) to predict a continuous variable (our dependent variable). This approach led to me dividing this project into four stages: Setup, Training, Plotting, and Conclusions. Setup will consist of light data cleaning and determining which factors will be relevant for our regression. After that, Training will involve of the linear regression, alongside testing and analyzing its performance. Next, Plotting will consist of visualizing our errors and residuals, along with graphing our independent variables and seeing how they performed in our regression. Finally, Conclusions will consist of my interpretation of the model, and my closing thoughts.

So the question is, what predicts a player's rushing yards?

Stage 1: Setup

Firstly, I loaded all necessary packages and read in my CSV file, consisting of all NFL rushing data. After that, I filtered for just players designated as running backs (RB) and removed any special characters from the ends of their name for clarity. (The characters dictated honors and awards, which is irrelevant to our experiment).

```
# Setup: Load necessary libraries and suppress warnings
suppressWarnings({
  require(tidyverse)  # Load the tidyverse package for data manipulation and visualiz
ation
  require(caret)      # Load the caret package for machine learning tools
  require(ranger)     # Load the ranger package for random forest modeling

  # Read the rushing data from the provided URL
  rushing <- read.csv("https://github.com/ethan-j-xu/nfl-rushing-model/raw/main/NFL%2
02022-2023%20Rushing%20Data.csv")

  # Rename several columns
  rushing <- rushing %>%
    rename(Rank = Rk,
           Team = Tm,
           Position = Pos,
           Games.Played = G,
           Games.Started = GS,
           Rushing.Attempts = Att,
           Rushing.Yards = Yds,
           Rushing.TDs = TDs,
           First.Downs = X1D,
           Rushing.Success.Rate = Succ.,
           Longest.Rushing.Attempt = Lng,
           YPC = Y.A,
           YPG = Y.G,
           Fumbles = Fmb
           ) %>%

    # Filter the data to include only running backs (RB) and remove special character
s from the 'Player' column
    filter(Position == 'RB') %>%
    mutate(Player = gsub("[*+]", "", Player))  # Remove asterisks and plus signs from
the 'Player' column
})
```

```
## Loading required package: tidyverse
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
## Loading required package: caret
##
## Loading required package: lattice
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
## Loading required package: ranger
```

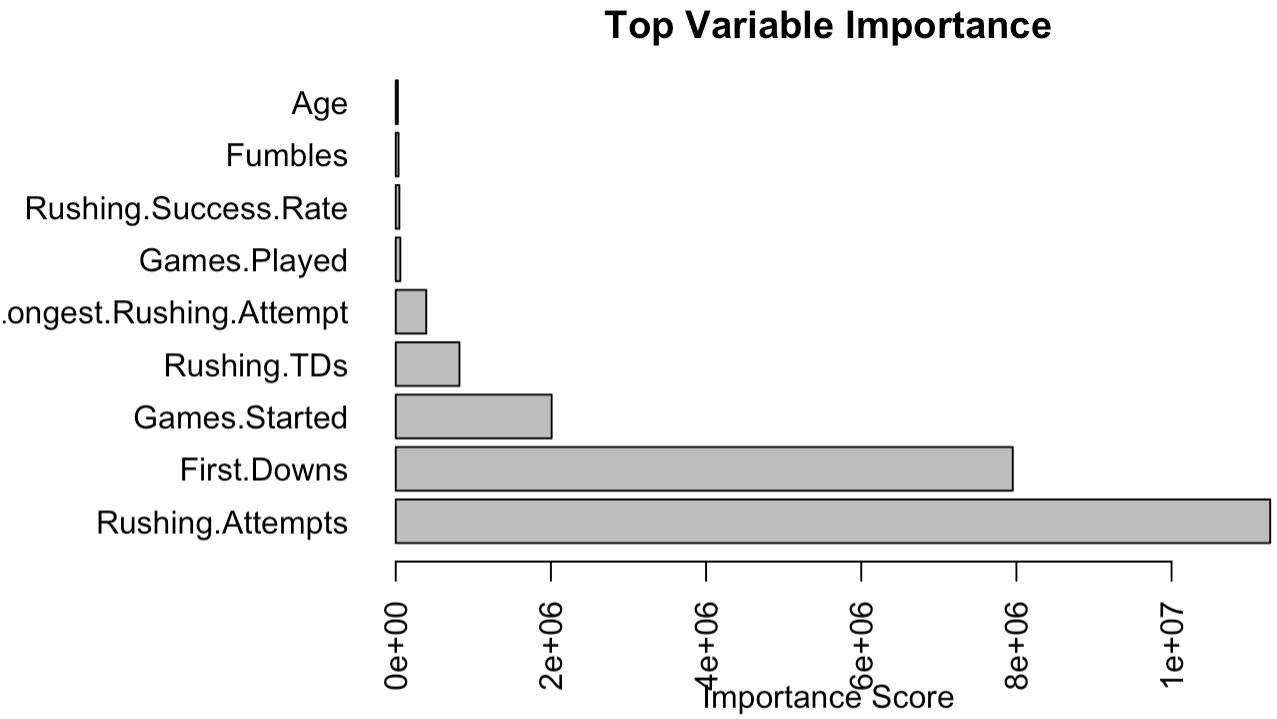
Finally, I checked the first ten rows of the data set to ensure that my cleaning was successful.

```
head(rushing, 10) # Check the first ten rows of the data set
```

```
##    Rank              Player Team Age Position Games.Played Games.Started
## 1     1       Derrick Henry  TEN  28       RB           16            16
## 2     2         Josh Jacobs  LVR  24       RB           17            17
## 3     3          Nick Chubb  CLE  27       RB           17            17
## 4     4      Saquon Barkley  NYG  25       RB           16            16
## 5     5        Najee Harris  PIT  24       RB           17            17
## 6     6         Dalvin Cook  MIN  27       RB           17            17
## 7     7     Jamaal Williams  DET  27       RB           17             9
## 8     8       Miles Sanders  PHI  25       RB           17            15
## 9     9 Christian McCaffrey  2TM  26       RB           17            16
## 10   10     Ezekiel Elliott  DAL  27       RB           15            14
##    Rushing.Attempts Rushing.Yards Rushing.TDs First.Downs Rushing.Success.Rate
## 1               349          1538          13          65                 46.7
## 2               340          1653          12          93                 57.4
## 3               302          1525          12          69                 50.0
## 4               295          1312          10          62                 47.5
## 5               272          1034           7          45                 46.0
## 6               264          1173           8          50                 47.3
## 7               262          1066          17          65                 53.1
## 8               259          1269          11          62                 56.8
## 9               244          1139           8          59                 48.4
## 10              231           876          12          52                 47.6
##    Longest.Rushing.Attempt YPC  YPG Fumbles
## 1                       56 4.4 96.1       6
## 2                       86 4.9 97.2       3
## 3                       41 5.0 89.7       1
## 4                       68 4.4 82.0       1
## 5                       36 3.8 60.8       3
## 6                       81 4.4 69.0       4
## 7                       58 4.1 62.7       3
## 8                       40 4.9 74.6       2
## 9                       49 4.7 67.0       1
## 10                      27 3.8 58.4       0
```

I opted to use a random forest in order to determine the best factors for my linear regression. I took out irrelevant variables such as the player's team, and factors that directly correlated with rushing yards such as yards per game.

```
suppressWarnings({
rf_model <- ranger(Rushing.Yards ~ . -Team - Rank - Player - Position -YPC - YPG, dat
a = rushing, num.trees = 100, mtry = 5, importance = 'impurity')

variable_importance <- importance(rf_model)

sorted_importance <- sort(variable_importance, decreasing = TRUE)
par(mar = c(10, 10, 2, 2))
n_top_variables <- 10  # Adjust this to the number of top variables you want to displ
ay
top_variables <- head(sorted_importance, n_top_variables)
barplot(top_variables, main = "Top Variable Importance", xlab = "Importance Score", h
oriz = TRUE, las = 2)
{}
})
```




```
## NULL
```

After the random forest, I checked the correlation of the top ten factors. Unfortunately, many of the variables had a high correlation with each other, which I opted to define as being higher than 0.5. Given my domain knowledge of football, I focused on keeping the Rushing.Attempts factor, along with taking factors which appeared to be independent of it, namely Rushing.Success.Rate and Age.

```
cor(rushing %>% select(Rushing.Attempts, First.Downs, Games.Started, Rushing.TDs, Lon
gest.Rushing.Attempt, Fumbles, Rushing.Success.Rate, Games.Played, Age)) # Check corr
elation
```

```
##                          Rushing.Attempts  First.Downs Games.Started Rushing.TDs
## Rushing.Attempts             1.000000000 0.9812562290    0.90560276  0.86897563
## First.Downs                  0.981256229 1.0000000000    0.87716167  0.87244388
## Games.Started                0.905602761 0.8771616672    1.00000000  0.78130642
## Rushing.TDs                  0.868975630 0.8724438850    0.78130642  1.00000000
## Longest.Rushing.Attempt      0.749177581 0.7545213057    0.64893288  0.68110172
## Fumbles                      0.656287138 0.6193605291    0.67482225  0.49900543
## Rushing.Success.Rate         0.214926304 0.2638171651    0.16518094  0.19789259
## Games.Played                 0.550837645 0.5438331926    0.44520271  0.47670708
## Age                          0.003618058 0.0009358233    0.04613664  0.04352939
##                          Longest.Rushing.Attempt    Fumbles Rushing.Success.Rate
## Rushing.Attempts                      0.74917758 0.65628714           0.21492630
## First.Downs                           0.75452131 0.61936053           0.26381717
## Games.Started                         0.64893288 0.67482225           0.16518094
## Rushing.TDs                           0.68110172 0.49900543           0.19789259
## Longest.Rushing.Attempt               1.00000000 0.47858264           0.26168294
## Fumbles                               0.47858264 1.00000000           0.11891812
## Rushing.Success.Rate                  0.26168294 0.11891812           1.00000000
## Games.Played                          0.47206490 0.37302032           0.27441761
## Age                                  -0.09700319 0.03706224          -0.01144786
##                          Games.Played           Age
## Rushing.Attempts           0.55083765  0.0036180578
## First.Downs                0.54383319  0.0009358233
## Games.Started              0.44520271  0.0461366406
## Rushing.TDs                0.47670708  0.0435293877
## Longest.Rushing.Attempt    0.47206490 -0.0970031879
## Fumbles                    0.37302032  0.0370622416
## Rushing.Success.Rate       0.27441761 -0.0114478595
## Games.Played               1.00000000  0.0278854171
## Age                        0.02788542  1.0000000000
```

Stage 2: Training

My linear regression returned an R-squared of 0.9767, indicating that my three factors performed well at predicting Rushing.Yards. Of the three, Rushing.Attempts was by far the most impactful (4.50941), with Rushing.Success.Rate having a positive relationship (0.73891) as Rushing.Yards increased and Age having a negative relationship (-1.39998).

```
yardsPrediction = lm (Rushing.Yards ~ Rushing.Attempts + Rushing.Success.Rate + Age,
data = rushing)
summary(yardsPrediction) # Run the linear regression
```

```
##
## Call:
## lm(formula = Rushing.Yards ~ Rushing.Attempts + Rushing.Success.Rate +
##     Age, data = rushing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -186.101  -24.048   -1.521   28.497  170.861
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -6.84978   53.82118  -0.127   0.8989
## Rushing.Attempts      4.50941    0.06139  73.456   <2e-16 ***
## Rushing.Success.Rate  0.73891    0.33445   2.209   0.0288 *
## Age                  -1.39998    2.02099  -0.693   0.4897
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.98 on 137 degrees of freedom
## Multiple R-squared:  0.9767, Adjusted R-squared:  0.9762
## F-statistic:  1912 on 3 and 137 DF,  p-value: < 2.2e-16
```

However, Age had an extremely high p-value at 0.4897, so I opted to remove it.

```
yardsPrediction = lm (Rushing.Yards ~ Rushing.Attempts + Rushing.Success.Rate, data =
rushing) # Remove 'Age' from the regression
summary(yardsPrediction) #Summarise the regression
```

```
##
## Call:
## lm(formula = Rushing.Yards ~ Rushing.Attempts + Rushing.Success.Rate,
##     data = rushing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -184.145  -23.577   -3.279   30.366  168.613
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -42.46624   15.88135  -2.674   0.0084 **
## Rushing.Attempts       4.50915    0.06127  73.592   <2e-16 ***
## Rushing.Success.Rate   0.74181    0.33379   2.222   0.0279 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61.86 on 138 degrees of freedom
## Multiple R-squared:  0.9766, Adjusted R-squared:  0.9763
## F-statistic:  2878 on 2 and 138 DF,  p-value: < 2.2e-16
```

This cross-validation helps ensure the quality of my regression. In the context of predicting NFL running back

rushing yards, an RMSE of 51.81 means that, on average, the model's predictions differ from the actual values by approximately 51.81 yards.

```
# Set seed for reproducibility
set.seed(123)

# Train a linear regression model using cross-validation
# Predict Rushing.Yards using Rushing.Attempts and Rushing.Success.Rate
# Perform 50-fold cross-validation
cv_results <- train(
  y = rushing$Rushing.Yards,
  x = rushing[, c("Rushing.Attempts", "Rushing.Success.Rate")],
  method = "lm",
  trControl = trainControl(method = "cv", number = 50)
)

# Print the results of the cross-validated linear regression model
print(cv_results)
```

```
## Linear Regression
##
## 141 samples
##   2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (50 fold)
## Summary of sample sizes: 138, 138, 138, 138, 138, 138, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   51.81315  0.9548759  42.55331
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

To check for overfitting, I calculated the RMSE on the full data, and then on a 80% sample of the dataframe.

```
#RMSE Whole Data
set.seed(123)
rushing %>%
  mutate(errors = Rushing.Yards - predict(yardsPrediction)) %>%
  mutate(sq_error = errors^2) %>% # Calculate the squared errors
  summarise(mean_sq = mean(sq_error)) %>% # Calculate the mean squared errors
  mutate(sq_of_the_mean_sq = sqrt(mean_sq))
```

```
##    mean_sq sq_of_the_mean_sq
## 1 3745.066          61.19695
```

```
#RMSE 100-fold CV
cvRes <- NULL # Instantiate an empty object to store data from the loop
for(i in 1:100) { # Loop 100 times
  inds <- sample(x = 1:nrow(rushing), # Sample from the row numbers of the rushing da
taframe
                 size = round(nrow(rushing)*0.8), # Set the size to be 80% of the tot
al rows (don't forget to round()!)
                 replace = FALSE) # Sample WITHOUT replacement

  train <- rushing %>% slice(inds) # Use the 80% to get the training data
  test <- rushing %>% slice(-inds) # Drop the 80% to get the test data

  m <- lm(Rushing.Yards ~ Rushing.Attempts + Rushing.Success.Rate, data = train) # Tr
ain the model on the train data

  test$errors <- predict(m, newdata = test) # Generate predicted values from the mode
l

  e <- test$Rushing.Yards - test$errors # Calculate the errors as the true Y minus th
e predicted Y
  se <- e^2 # Square the errors
  mse <- mean(se) # Take the mean of the squared errors
  rmse <- sqrt(mse) # Take the square root of the mean of the squared errors
  cvRes <- c(cvRes, rmse) # Append the rmse to the cvRes object
}

mean(cvRes)
```

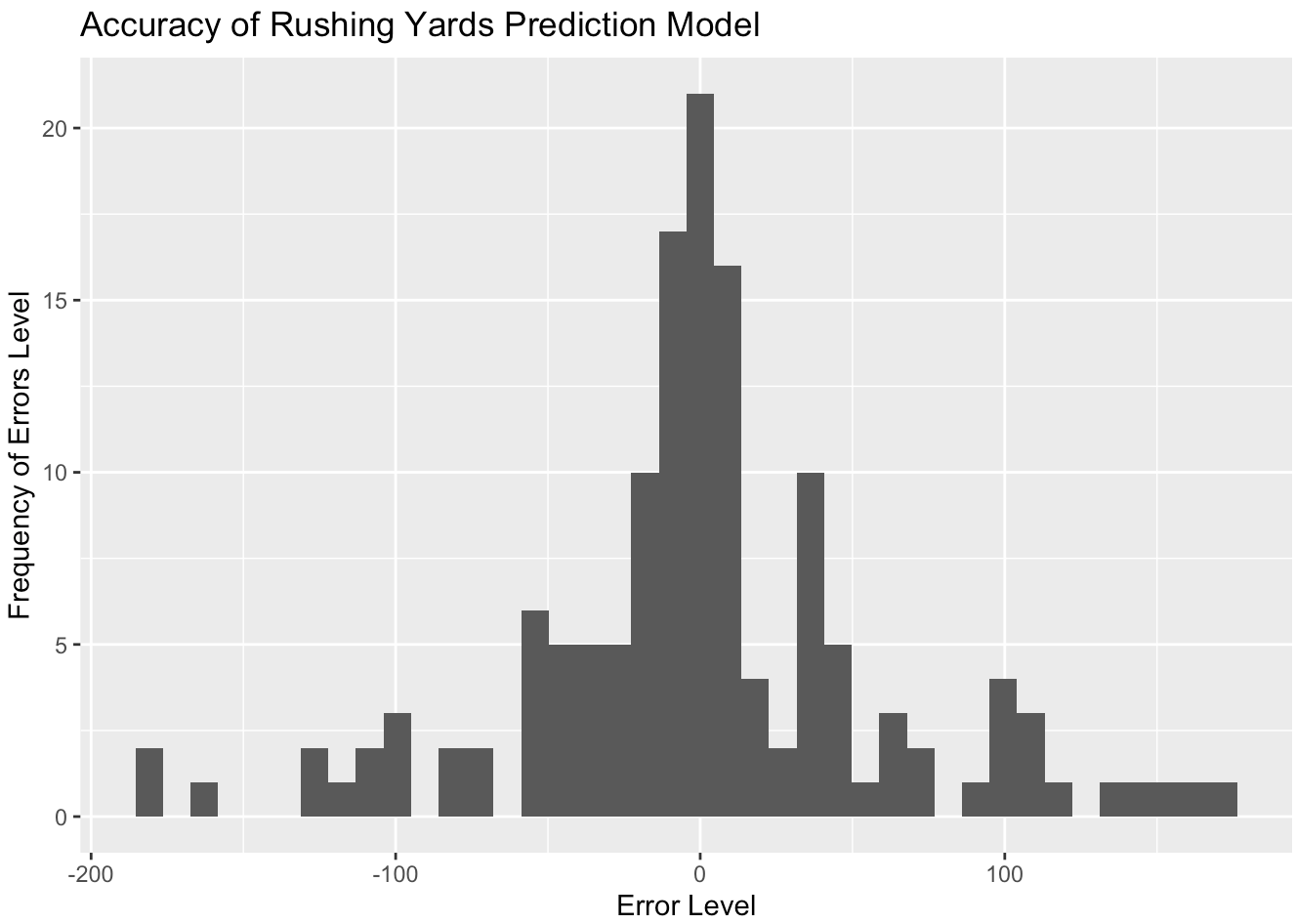
```
## [1] 60.58031
```

With the a different of 0.61664 between my full and test data, I concluded that my model was not overfitted.

Stage 3: Plotting

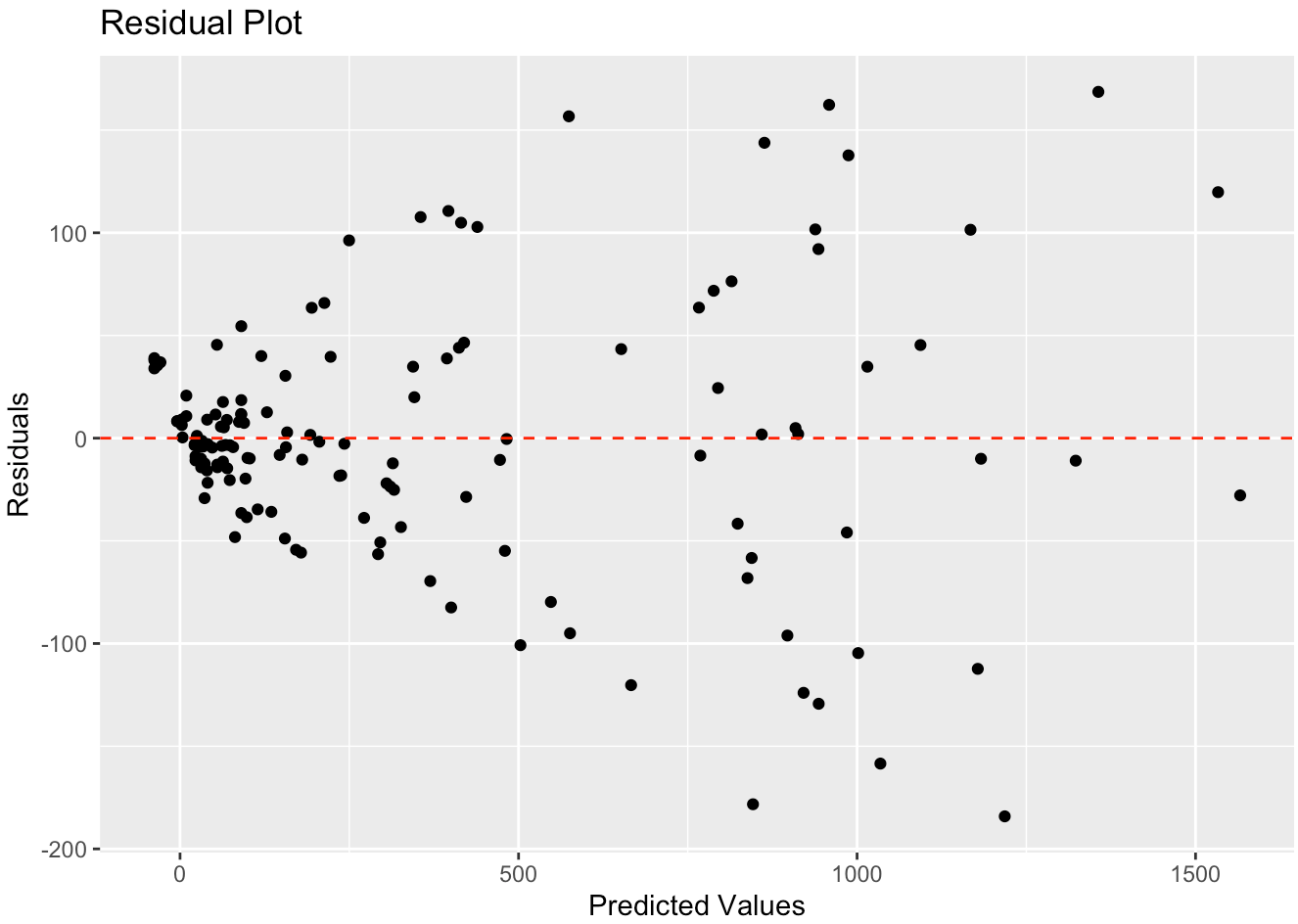
This graph shows all the errors in my regression. Overall, most predictions were within 100 yards, though there were some notable outliers that were close to -/+ yards.

```
rushing <- rushing %>%
  mutate(errors = Rushing.Yards - predict(yardsPrediction))# Create errors column for
differences between actual and predicted yards

rushing %>%
  ggplot(aes(x = errors)) + #Plot our findings, and visualize how frequently our mode
l was off by a certain amount.
  geom_histogram(bins = 40) +
  labs(x = 'Error Level',        # Label for x-axis
       y = 'Frequency of Errors Level',               # Label for y-axis
       title = 'Accuracy of Rushing Yards Prediction Model')  # Title for the plot
```



Through plotting the residuals, we can see that as Rushing.Yards increased, our model became less accurate. Our model was best at predicting running backs with a lower rushing total, as we were able to predict almost every sub-500 rusher within 100 yards.

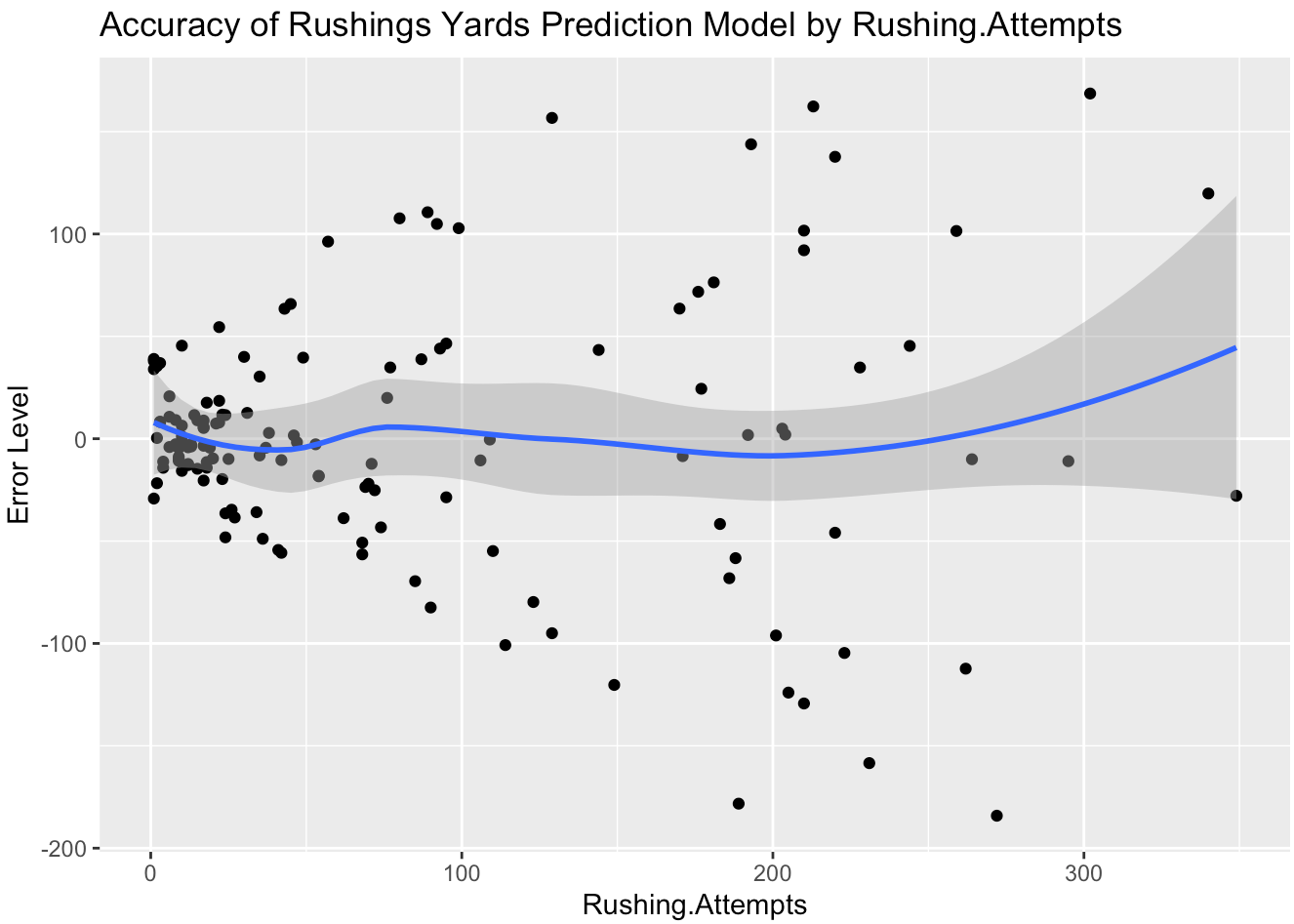
```
#Plots the predictions and residuals
residuals_rushing <- data.frame(Residuals = resid(yardsPrediction), Predicted = predi
ct(yardsPrediction))

ggplot(residuals_rushing, aes(x = Predicted, y = Residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +  # Add a dashed li
ne at y = 0
  labs(x = "Predicted Values", y = "Residuals") +
  ggtitle("Residual Plot")
```



I included two scatterplots to check how my model performed by variable. Rushing.Attempts performed best as lower levels and became more inaccurate, notably taper off around the 250 mark.

```
rushing %>%
  ggplot(aes(x = Rushing.Attempts, y = errors)) + #Plot our findings, and visualize h
ow accurate our model was for each level of rushing attempts
  geom_point() +
  geom_smooth() + #Add line of best fit
  labs(x = 'Rushing.Attempts',        # Label for x-axis
       y = 'Error Level',                # Label for y-axis
       title = 'Accuracy of Rushings Yards Prediction Model by Rushing.Attempts')
```

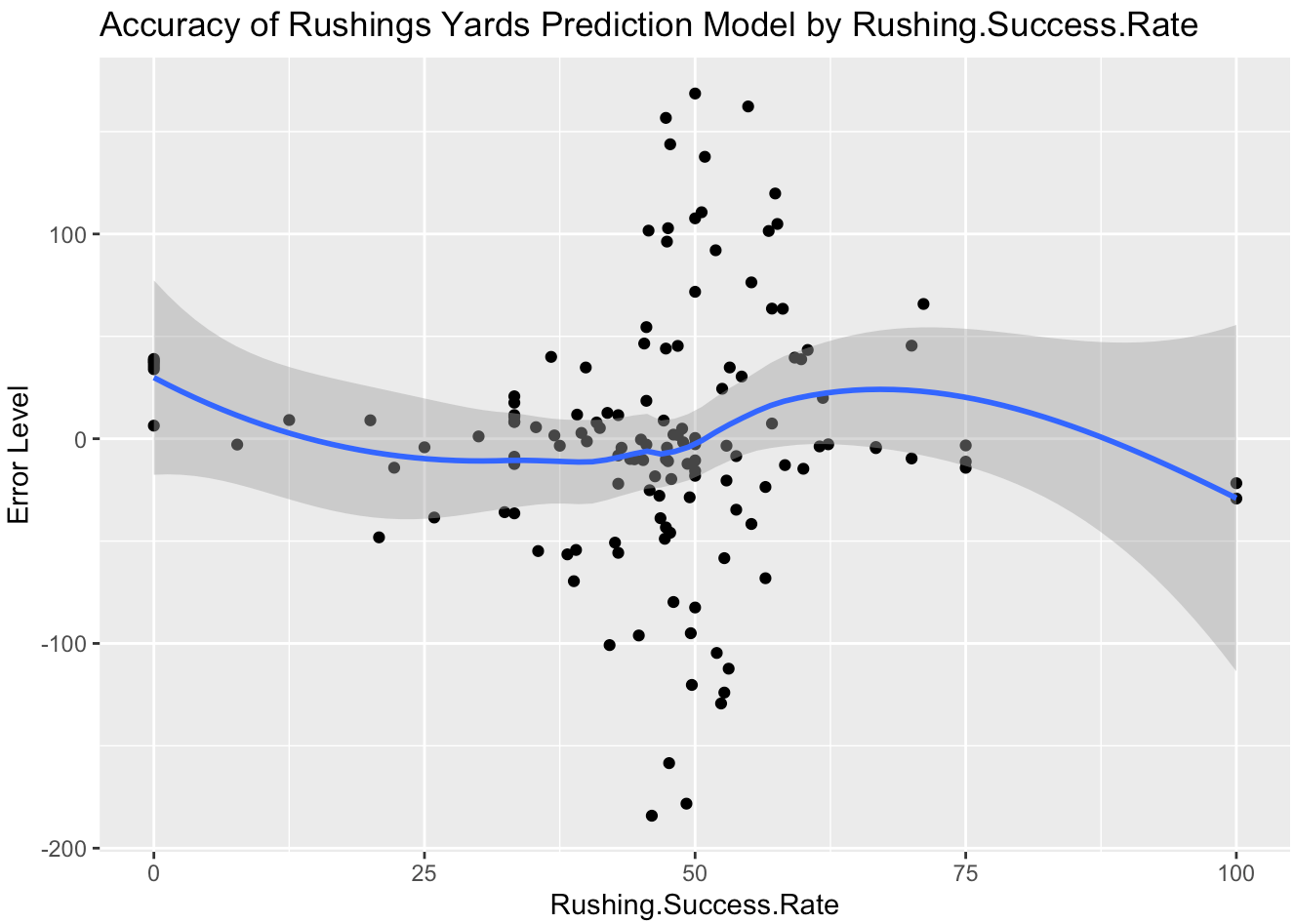
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Rushing.Success.Rate had a much different graph. Our lowest residuals came around the 40-60 mark, with both lower and higher levels of Rushing.Success.Rate having high residuals.

```
rushing %>%
  ggplot(aes(x = Rushing.Success.Rate, y = errors)) + #Plot our findings, and visuali
ze how accurate our model was for each level of rushing success
  geom_point() +
  geom_smooth() + #Add line of best fit
  labs(x = 'Rushing.Success.Rate',        # Label for x-axis
       y = 'Error Level',                # Label for y-axis
       title = 'Accuracy of Rushings Yards Prediction Model by Rushing.Success.Rate')
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Stage 4: Conclusions

Overall, all metrics indicated that my model was sound and accurate. Between a low RMSE and high R-squared, my model appears to successfully perform its goal of predicting a player's rushing yards, using the Rushing.Attempts and Rushing.Success.Rate factors. However, it is important to note that the model performed better on 'worse' (less productive players), while struggling more for 'better' (more productive players). This may indicate that I was missing a factor that was not present in my dataset, that would explain what distinguishes these high performing players from the rest of the league.

Additionally, the fact that I was only able to use two factors is an area of concern. A similar regression developed by a classmate regarding NFL receiving yards successfully integrated several more factors from their respective data set, including advanced metrics such as ADOT (Average Depth of Pass) and Broken Tackles. Between the limited data in my data set, the inherently less complicated role of running backs, and my own lack of data science experience, I cannot confidently attribute this issue to any one factor. It is entirely plausible that the question of what makes a productive NFL running back is quite simple: more quality attempts leads to more yards. Future work on this topic may benefit from the addition of advanced metrics, such as contact in the backfield (which could be indicative of the offensive line quality of a team), or broken tackles.