Predicting Atlanta Falcons NFL Touchdowns with Regression Modelling

Bhargav Ashok

2025-05-13

0. Setup - Install Packages: tidyr, readxl, dplyr, car, glmnet, Metrics, 3dscatterplot

Loading our Atlanta Falcons Data

```
Atlanta_Falcons_data <- read_excel("Atlanta_Falcons_data.xlsx")
head(Atlanta_Falcons_data)
```

```
## # A tibble: 6 x 18
##      Rk Player      From    To     G Pos      AV   Tgt   Rec `Ctch%`   Yds `Y/R`
##   <dbl> <chr>      <dbl> <dbl> <dbl> <chr> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1     1 Julio Jon~  2011  2020   135 WR      119  1320   848   0.642 12896  15.2
## 2     2 Roddy Whi~  2005  2015   171 WR      107  1377   808   0.587 10863  13.4
## 3     3 Terance M~  1994  2001   126 WR       67   989   573   0.579  7349  12.8
## 4     4 Alfred Je~  1975  1983   110 WR       61    NA   360  NA      6267  17.4
## 5     5 Andre Ris~  1990  1994    78 WR       53   463   423  NA      5633  13.3
## 6     6 Jim Mitch~  1969  1979   155 TE       47    NA   305  NA      4358  14.3
## # i 6 more variables: TD <dbl>, Lng <dbl>, `Y/Tgt` <dbl>, `R/G` <dbl>,
## #   `Y/G` <dbl>, Fmb <dbl>
```

Preparing our data, cleaning and training

Here, we are cleaning our data with the tidyr package and then splitting our data into a trained and tested set.

```
vars_needed <- c("TD", "Tgt", "Rec", "Ctch%", "Yds")
clean_df    <- Atlanta_Falcons_data %>% drop_na(all_of(vars_needed))

set.seed(123)
n       <- nrow(clean_df)
train_i <- sample(n, 0.8*n)
train_df <- clean_df[ train_i, ]
test_df  <- clean_df[-train_i, ]
```

Fitting our Model (regular MLR)

Next, we fit our model using the equation: Y = (Beta)0 + (Beta)1X1 + (Beta)2X2 + … + (Beta)nXn + error (assuming normal distribution)

```r
mlr_fit <- lm((TD) ~ Tgt + Rec + `Ctch%` + Yds, data = train_df)
summary(mlr_fit)
```

```
##
## Call:
## lm(formula = (TD) ~ Tgt + Rec + `Ctch%` + Yds, data = train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4458  -0.7242  -0.1969   0.7605  10.6716
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.3606742  0.7621684  -0.473   0.6368
## Tgt          0.0312459  0.0135714   2.302   0.0228 *
## Rec         -0.0044023  0.0176458  -0.249   0.8034
## `Ctch%`      0.5479172  1.0636249   0.515   0.6073
## Yds          0.0020808  0.0008571   2.428   0.0165 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.423 on 138 degrees of freedom
## Multiple R-squared:  0.9028, Adjusted R-squared:  0.8999
## F-statistic: 320.3 on 4 and 138 DF,  p-value: < 2.2e-16
```

The summary statistics are displayed above. Note the coefficients for the MLR equation and the adjusted R-squared Value.

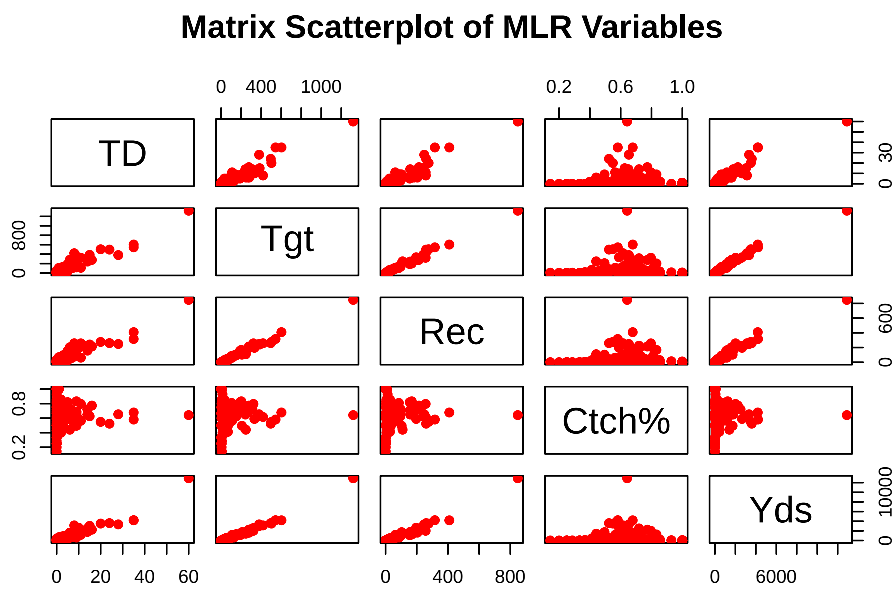
The MLR equation can be identified and written as: "TD = -0.36 + 0.03*Tgt - 0.004*Rec + 0.55*Ctch% + 0.002*Yds"

This equation suggests that Catch% % has the largest effect on Touchdowns by a factor of 0.548, with yards being a close second with an effect factor of 0.00208. The other factors(reception targets and receptions, meaning how many times the said player was targeted for the throw, and how many times they completed a given catch) seem to have a negative effect, suggesting over fitting. However, this is handled via cross-validation and Ridge regression in the improved MLR model later on. This means that all of these factors influence the number of touchdowns by varying rates based on test statistics performed in this module.

Data Visualization of the Matrix Scatterplot

Here we see that the scatter plot compares all the different variables together to check for linearity and how each variable influences each other. We may see non-linearity between some variables but that will be accounted for later on.

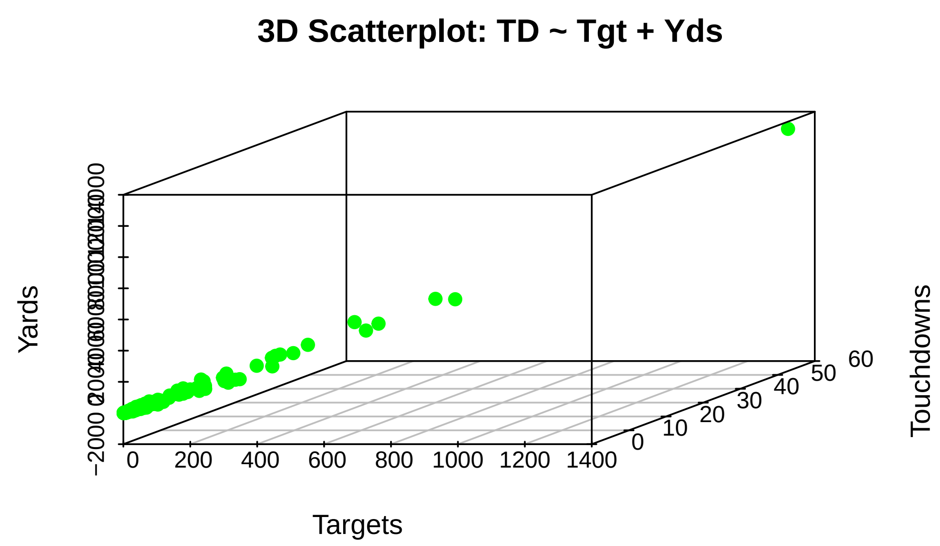
```r
pairs(
  ~ TD + Tgt + Rec + `Ctch%` + Yds,
  data = train_df,
  main = "Matrix Scatterplot of MLR Variables",
  col = "red",
  pch = 19
)
```



Data visualization of the 3d scatterplot

Here we are taking the top two linearly correlated variables and plotting them on a 3d scatterplot using the R 3d scatterplot package.

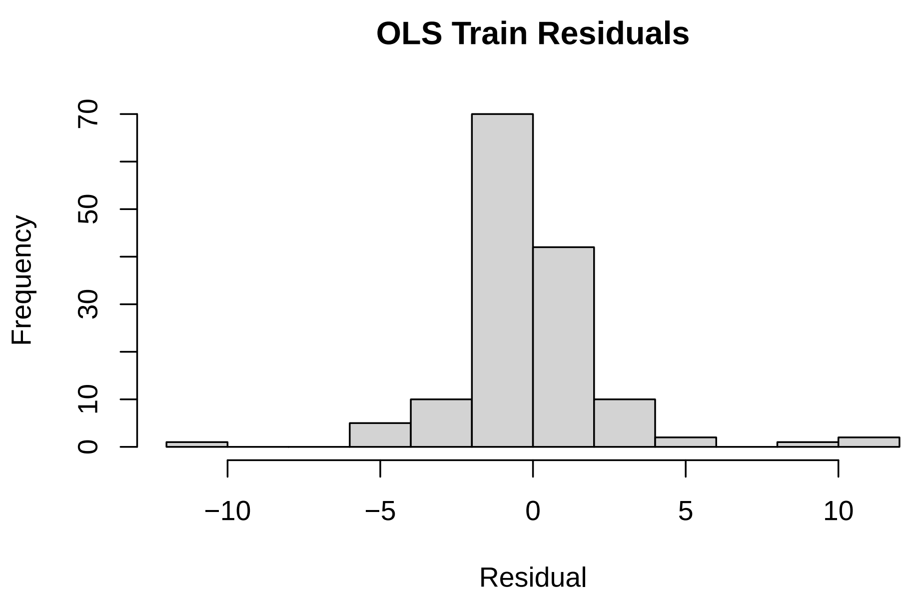
```
scatterplot3d(
  x = train_df$Tgt,
  y = train_df$TD,
  z = train_df$Yds,
  main = "3D Scatterplot: TD ~ Tgt + Yds",
  xlab = "Targets",
  ylab = "Touchdowns",
  zlab = "Yards",
  color = "green",
  pch = 19
)
```



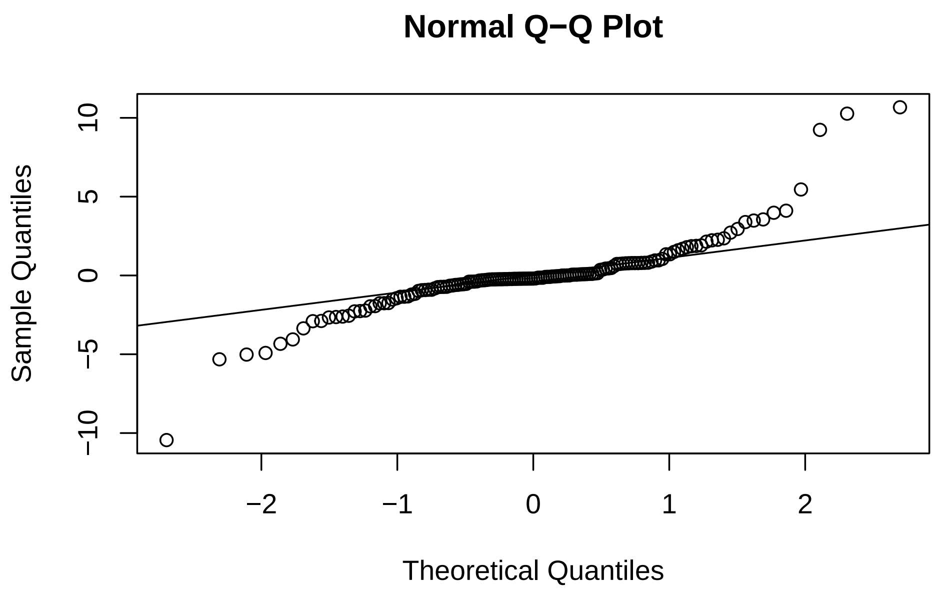
MLR fit data display (looking at model data)

Then, we display plots for our fitted data and the VIFs using the car package in R.

```
hist(resid(mlr_fit), main="OLS Train Residuals", xlab="Residual")
```




```
qqnorm(resid(mlr_fit)); qqline(resid(mlr_fit))
```




```r
vif(mlr_fit)
```

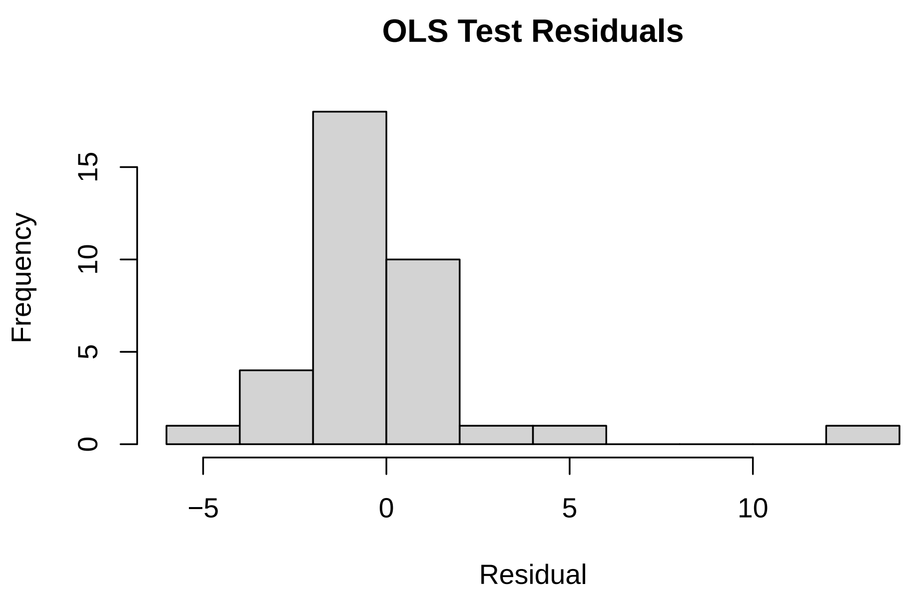
```
##        Tgt        Rec    `Ctch%`        Yds
## 110.938423  76.817866   1.126715  32.696953
```

From the plots, we see that our trained residual data is roughly normal along with the qqplot being roughly normal as well (with deviation in the extreme values or outliers).

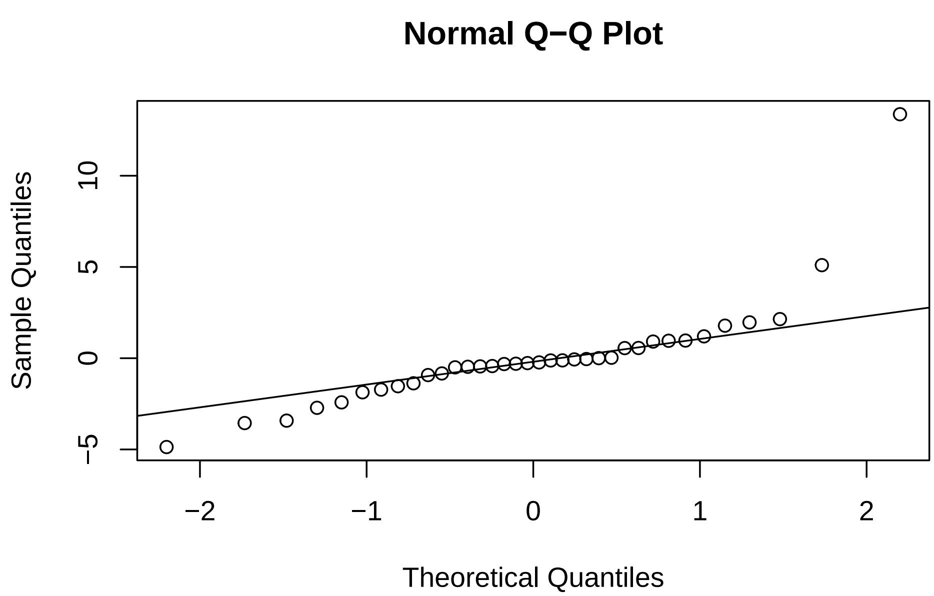
test data display (looking at unseen test data)

After that, we look at our unseen test data plots.

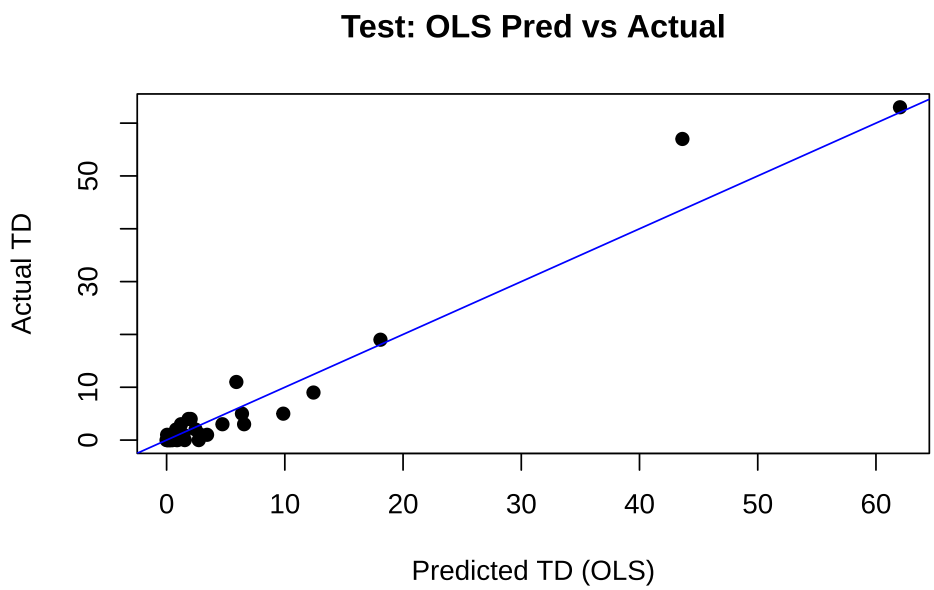
```r
test_df$pred_ols  <- predict(mlr_fit, newdata = test_df)
resid_test_ols    <- test_df$TD - test_df$pred_ols

# Plots
hist(resid_test_ols, main="OLS Test Residuals", xlab="Residual")
```




```r
qqnorm(resid_test_ols); qqline(resid_test_ols)
```




```r
plot(
  test_df$pred_ols, test_df$TD,
  xlab="Predicted TD (OLS)", ylab="Actual TD",
  main="Test: OLS Pred vs Actual", pch=19
)
abline(0,1,col="blue")
```



We see the linear regression line on our predicted test data above. We don't need to assume normality for our test residuals in a Machine Learning Model (unless a hypothesis test is conducted within a controlled experiment).

Using Cross validation + Ridge/Lasso Regression

After looking at our models and our VIF (Variance Inflation Factor, which shows how standardized our data is in terms of multicollinearity), we see that the VIF > 10, which indicates high multicollinearity. This is not ideal for this scenario because the variables need to be standardized to account for the number inflation in multicollinearity (since they are supposed to be statistically significant, not reflected in the regular MLR p-value) . Due to this, we need to switch to a new type of regression model for even more accurate results. We use a technique called K-folds cross validation, where the data is split into multiple subsets and is iterated more than once in order to account for the multicollinearity inflation which is indicated above, as well improving the model to see how accurately it can predict unseen data points. We use the glmnet package for this.

```
# Prepare matrices
x_train <- model.matrix(TD~Tgt+Rec+`Ctch%`+Yds, train_df)[,-1]
y_train <- train_df$TD
x_test  <- model.matrix(TD~Tgt+Rec+`Ctch%`+Yds, test_df)[,-1]
y_test  <- test_df$TD

# Ridge
cv_ridge  <- cv.glmnet(x_train,y_train,alpha=0)
best_ridge<- cv_ridge$lambda.min
ridge_mod <- glmnet(x_train,y_train,alpha=0,lambda=best_ridge)
test_df$pred_ridge <- as.numeric(predict(ridge_mod,x_test))
summary(cv_ridge)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
```

```
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         4    -none- call
## name         1    -none- character
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```r
# Lasso
cv_lasso  <- cv.glmnet(x_train,y_train,alpha=1)
best_lasso<- cv_lasso$lambda.min
lasso_mod <- glmnet(x_train,y_train,alpha=1,lambda=best_lasso)
test_df$pred_lasso <- as.numeric(predict(lasso_mod,x_test))
summary(cv_lasso)
```

```
##            Length Class  Mode
## lambda     59     -none- numeric
## cvm        59     -none- numeric
## cvsd       59     -none- numeric
## cvup       59     -none- numeric
## cvlo       59     -none- numeric
## nzero      59     -none- numeric
## call        4     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
## index       2     -none- numeric
```

```r
# RMSE
cat("Ridge RMSE:",  rmse(y_test, test_df$pred_ridge), "\n")
```

```
## Ridge RMSE: 3.52247
```

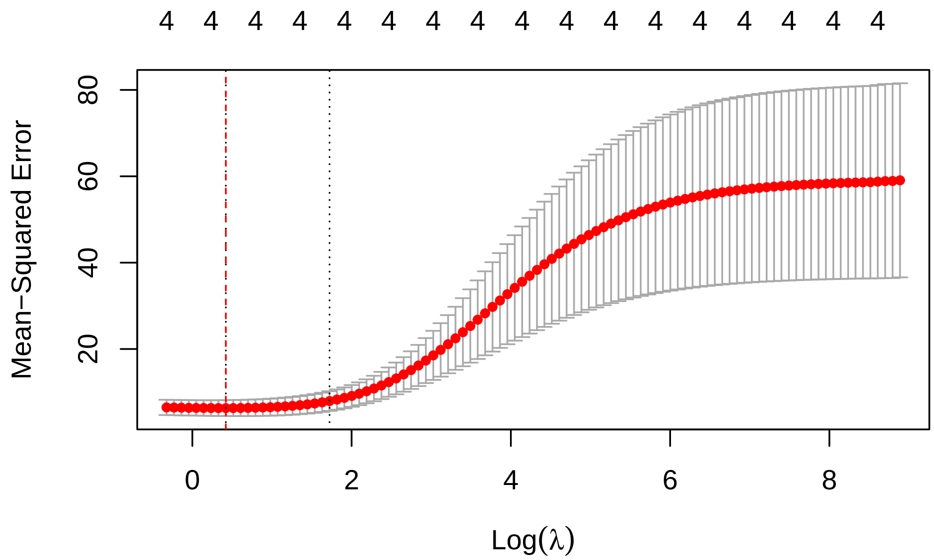
```r
cat("Lasso RMSE:",  rmse(y_test, test_df$pred_lasso), "\n")
```

```
## Lasso RMSE: 3.975591
```

We do a Cross Validation of Ridge and Lasso regression to see which one is more accurate. As we can see, Ridge regression has a lower RMSE which is more accurate for our model, so we will plot the Cross validation curve. We get the RSME score from the Metrics Package.

Plotting Ridge regression and summary metrics

```r
plot(cv_ridge)
abline(v=log(best_ridge),col="red",lty=2)
```




```r
summary(train_df)
```

```
##        Rk           Player               From           To
##  Min.   :  1.0   Length:143         Min.   :1992   Min.   :1992
##  1st Qu.: 79.0   Class :character   1st Qu.:2001   1st Qu.:2004
##  Median :156.0   Mode  :character   Median :2009   Median :2012
##  Mean   :166.7                      Mean   :2009   Mean   :2011
##  3rd Qu.:263.0                      3rd Qu.:2018   3rd Qu.:2020
##  Max.   :326.0                      Max.   :2024   Max.   :2024
##        G              Pos                 AV              Tgt
##  Min.   :  1.0   Length:143         Min.   :  0.00   Min.   :   1.00
##  1st Qu.: 12.0   Class :character   1st Qu.:  1.00   1st Qu.:   4.50
##  Median : 22.0   Mode  :character   Median :  3.00   Median :  28.00
##  Mean   : 35.8                      Mean   : 11.43   Mean   :  82.94
##  3rd Qu.: 52.5                      3rd Qu.: 11.00   3rd Qu.:  96.50
##  Max.   :222.0                      Max.   :203.00   Max.   :1320.00
##       Rec              Ctch%             Yds              Y/R
##  Min.   :  1.00   Min.   :0.1430   Min.   :   -7.0   Min.   :-7.000
##  1st Qu.:  2.00   1st Qu.:0.5240   1st Qu.:   19.5   1st Qu.: 7.100
##  Median : 16.00   Median :0.6670   Median :  181.0   Median :10.000
##  Mean   : 53.17   Mean   :0.6708   Mean   :  618.0   Mean   : 9.727
##  3rd Qu.: 60.50   3rd Qu.:0.7830   3rd Qu.:  622.0   3rd Qu.:13.300
##  Max.   :848.00   Max.   :1.0000   Max.   :12896.0   Max.   :21.000
##        TD             Lng           Y/Tgt             R/G
##  Min.   : 0.00   Min.   :-5.0   Min.   :-7.000   Min.   :0.000
##  1st Qu.: 0.00   1st Qu.:12.5   1st Qu.: 4.700   1st Qu.:0.250
##  Median : 1.00   Median :28.0   Median : 6.300   Median :0.800
##  Mean   : 3.65   Mean   :33.3   Mean   : 5.987   Mean   :1.262
##  3rd Qu.: 3.00   3rd Qu.:53.0   3rd Qu.: 8.000   3rd Qu.:1.800
##  Max.   :60.00   Max.   :94.0   Max.   :14.500   Max.   :6.300
##       Y/G             Fmb
```

```
##  Min.   :-0.80   Min.   : 0.0
##  1st Qu.: 1.90   1st Qu.: 0.0
##  Median : 8.40   Median : 0.0
##  Mean   :14.51   Mean   : 3.0
##  3rd Qu.:19.65   3rd Qu.: 2.5
##  Max.   :95.50   Max.   :89.0
```

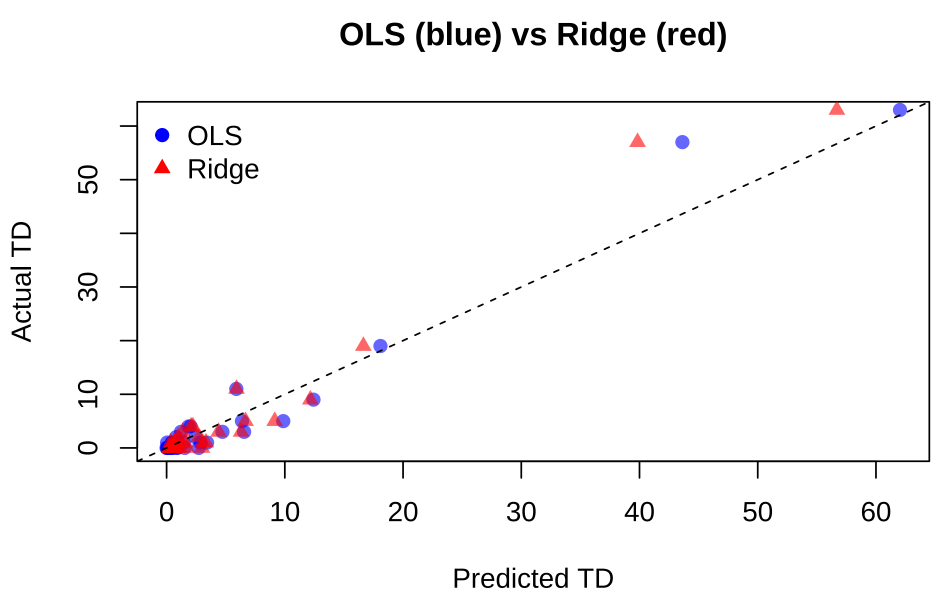
The cross validation ridge plot is shown above. According to "https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/choosing-lambda.html":

"What is plotted is the estimated CV MSE for each value of (log)lambda on the x-axis. The dotted line on the far left indicates the value of lambda which minimizes CV error. The dotted line roughly in the middle of the x-axis indicates the 1-standard-error lambda- recall that this is the maximum value that lambda can take while still falling within the on standard error interval of the minimum-CV lambda. The second line of code has manually added a dot-dash horizontal line at the upper end of the 1-standard deviation interval of the MSE at the minimum-CV lambda to illustrate this point further". These plots can change with randomization according to our seed number.

The summary shown by the trained data are regularized and explain the scale of the variables within the ridge regression.

Plotting our Comparison graph between MLR and Ridge Regression MLR

```r
plot(test_df$pred_ols, test_df$TD,
     xlim = range(c(test_df$pred_ols,test_df$pred_ridge)),
     ylim = range(c(test_df$pred_ols,test_df$pred_ridge)),
     xlab="Predicted TD", ylab="Actual TD",
     main="OLS (blue) vs Ridge (red)", pch=19, col=rgb(0,0,1,0.6))
points(test_df$pred_ridge, test_df$TD, pch=17, col=rgb(1,0,0,0.6))
abline(0,1,lty=2)
legend("topleft", legend=c("OLS","Ridge"), pch=c(19,17),
       col=c("blue","red"), bty="n")
```



Finally, we can compare our MLR Ordinary Least Squares Regression (Linear Regression) Model with our Cross-Validated, Ridge Regression Model visually.

Conclusion of Findings

We can now safely say that the Atlanta Falcons TDs can be predicted by multiple factors within a game such as catch percentage, receptions, yards, and potentially more.

So when identifying top talent on the Falcons, or potentially scouting out new talent that fits the Falcons scheme (looking at college production rates vs Falcons production rates) This model can be identified and used for the comparison and scouting of offensive talent. Outliers are special cases that may influence the linearity of the data set, however, depending on production value of the player (i.e., Superstar or Not superstar), appropriate coaching staff/scouts must factor this in to the model and make the best decision for the team.

Future Improvements for the Model

While this model gives us a deeper understanding of Touchdown prediction, there can be a handful improvements made in a real-life scenario:

1. Automation of roster data in future findings
2. A classification model detailing other external factors (behavior, team chemistry, etc.) can also be used in tandem with this model in order to make an even more accurate decision.
3. Expanding the model to look at more advanced offensive stats/metrics

References Used:

https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/choosing-lambda.html

https://www.pro-football-reference.com/teams/atl/career-receiving.htm

https://online.stat.psu.edu/stat462/node/180/

https://stats.stackexchange.com/questions/279300/how-to-interpret-cross-validation-plot-from-glmnet

https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

https://online.stat.psu.edu/stat462/node/131/

https://www.reddit.com/r/AskStatistics/comments/ycjoy4/what_threshold_is_used_to_assess_multicollinearity/

https://www.datacamp.com/doc/r/regression

https://www.geo.fu-berlin.de/en/v/soga-r/Basics-of-statistics/Linear-Regression/Polynomial-Regression/Polynomial-Regression---An-example/index.html

https://online.stat.psu.edu/stat462/node/177/

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

https://www.reddit.com/r/rstats/comments/oibm2w/how_to_deal_with_multicollinearity_in_linear/

https://www.datacamp.com/tutorial/multicollinearity

https://www.rdocumentation.org/packages/car/versions/3.1-3

https://www.rdocumentation.org/packages/Metrics/versions/0.1.4

https://www.r-bloggers.com/2021/10/lambda-min-lambda-1se-and-cross-validation-in-lasso-binomial-response/

https://www.techtarget.com/searchenterpriseai/definition/data-splitting#:~:text=Training%20sets%20are%20commonly%20used,the%20final%20model%20works%20correctly.

https://www.datacamp.com/doc/r/scatterplot-in-r

https://www.educba.com/multiple-linear-regression-in-r/