NFL Project Report

Maurice Rankin

November 19, 2019

Data Science: Capstone "Choose Your Own!"

Contents

Introduction

For my “Choose Your Own” project” portion of my Capstone Project, I decided to create, train and test a learning algorithm that will predict the probabilities of winning outcome of a regular season NFL game. My algorithm will utilize historical data obtained from all NFL games played during the years 2009 to 2019 to generate winning probabilities over time for the home team.

Objectives

1. Scrap NFL data for years 2009 to 2019 from NFL.com.
2. Create a general logic function to estimate the probability of a NFL game Win.
3. Fit a model that will best represent this logical function.
4. Create an algorithm that projects the win probabilities of the home NFL team.

Data Analysis

The first challenge I found was getting the proper data sets I needed for this NFL Project. After testing various online data sets and sports-related websites, I decided to use the nflscrapR; which allows me to download the most complete and up to date NFL play-by-play statistics I will need to complete this project.

nflscrapR Installation and Library Setup

```
#Installed nflscrapR to support scrapping data from NFL.com
library(devtools)
devtools::install_github(repo <- "maksimhorowitz/nflscrapR")
devtools::install_github("dgrtwo/gganimate")
install.packages('caTools')

#Loaded proper libraries needed
library(nflscrapR)
library(tidyverse)
```

Next was the process of creating a web scrapping process that pulled the data I needed into R Studio. This process took quite a while to complete. NFL.com online website has a lower yearly limit of for game data of 2009, to 2019. The upper year limit is our current year 2019.

By using the saveRDS function in R Studio, I was able to save play-by-play data into individual files for each year. I then used the bind_rows function to combine each individual data file into one complete data frame named “play”.

Web Scrapping Season Play by Play Data

```
#Web Scrapping Data and Binding all seasons for years 2009 to 2019
play1 <- season_play_by_play(2019)
saveRDS(play1, "play_data_2019.rds")

play2 <- season_play_by_play(2018)
saveRDS(play2, "play_data_2018.rds")

play3 <- season_play_by_play(2017)
saveRDS(play3, "play_data_2017.rds")

play4 <- season_play_by_play(2016)
saveRDS(play4, "play_data_2016.rds")

play5 <- season_play_by_play(2015)
saveRDS(play5, "play_data_2015.rds")

play6 <- season_play_by_play(2014)
saveRDS(play6, "play_data_2014.rds")

play7 <- season_play_by_play(2013)
saveRDS(play7, "play_data_2013.rds")

play8 <- season_play_by_play(2012)
saveRDS(play8, "play_data_2012.rds")

play9 <- season_play_by_play(2011)
saveRDS(play9, "play_data_2011.rds")

play10 <- season_play_by_play(2010)
saveRDS(play10, "play_data_2010.rds")

play11 <- season_play_by_play(2009)
saveRDS(play11, "play_data_2009.rds")
```

Data Binding Season Play by Play Data

```
play <- bind_rows(play1,play2,play3,play4,play5,play6,play7,play8,play9,play10, play11)
saveRDS(play, "play_data.rds")
```

Now I required a way to scrape NFL.com for the gameID and the specific game info needed to create and gather features needed for my project. I was able to accomplish this task by using the following code in which I targeting the gameID's (years 2009 to 2019). I saved the individual files, that represented each years, with saveRDS (). This combined all individual file into on data frame, which I named "games".

Web Scrapping for game IDs

```
games2019 <-  scrape_game_ids(2019)
games2018 <-  scrape_game_ids(2018)
games2017 <-  scrape_game_ids(2017)
games2016 <-  scrape_game_ids(2016)
games2015 <-  scrape_game_ids(2015)
games2014 <-  scrape_game_ids(2014)
games2013 <-  scrape_game_ids(2013)
games2012 <-  scrape_game_ids(2012)
games2011 <-  scrape_game_ids(2011)
games2010 <-  scrape_game_ids(2010)
games2009 <-  scrape_game_ids(2009)
```

Data Binding game IDs

```
games <-  bind_rows(games2019, games2018, game2017, games2016, games2015, games2014,
games2013, games2012, games2011, games2010, games2009)
saveRDS(games, "games_data.rds")
```

The final stage of preparing my data for analysis was to bind both data frames into one large database of complete football data. I used the full_join function to combine both data frames on gameID. The resulting data frame (play_final) contained 481629 observations and 112 variables.

Full Joining Data frames

```
play_final <-  full_join(games, play_raw, by <- "GameID")
saveRDS(play_final, "play_final.rds")
```

Data Conversion

I wanted to make my model simple so I created a binary-in/binary out data conversion function. I created a response variable (poswins) that creates and stores the name of the team that ultimately wins the game (possession at the time of win).

```
        play_final <-  play_final %>% mutate(winner <- ifelse(home_score > away_score, home_team,
away_team))
        play_final <- play_final %>% mutate(poswins <- ifelse(winner == posteam, "Yes","No"))
        play_final$qtr <- as.factor(play_final$qtr)
        play_final$down <- as.factor(play_final$down)
        play_final$poswins <- as.factor(play_final$poswins)
```

The following code is used to filter plays that did not actually occur before I created the training and test sets (this can be considered and noise that could cause errors in my prediction).

```
noplay_data_filter = play_final %>% filter(PlayType != "No Play" & qtr != 5 & down != "NA") %>%
select(game_id, Date, posteam, HomeTeam, AwayTeam, winner, qtr, down, ydstogo, TimeSecs,
yrdline100, ScoreDiff, poswins)
```

Test and Training Sets

I used the following code to split the noplay_data_filter into training and test sets. To do this I used the sample.split function (caTools package) and set the seed to 123. Next, I created the training and test sets and removed any duplicate columns from the training gets.

```
 set.seed(123)
split <- sample.split(noplay_data_filter$poswins, SplitRatio <- 0.8)
train <- noplay_data_filter %>% filter(split == TRUE)
test <- noplay_data_filter %>% filter(split == FALSE)
train = cbind(noplay_data_filter, noplay_data_filter)
train = test[,!duplicated(names(test))]
```

Methods

Logical Regression Approach

After doing very in-depth research, I decided to use a simple logical regression model to create my algorithm. I used Beta to represent my estimated coefficients and the probability P of a binary event occurring. This binary event is the categorical outcome of either a Win or a Loss. Below is a breakdown of the logical function I used to build my model.

$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Logical Function

Variables and Features

X = predictors variables

B = estimated Coefficient

P = probability

Features

qtr = quarter

down = down (1^{st}, 2^{nd}, 3^{rd} or 4^{th})

ydstogo = yards to go

TimeSecs = time remaining for the game (in seconds)

ScoreDiff = difference in score calculated score in possession minus opponent's score

Poswins = response (binary variable/Shows which team wins the game)

Logistics Regression Model

Now the time had come to start building my logical regression model. Using the glm () function, I set the value for family = “binomial”. I used the following to estimate poswins (which is the win probability invoked on my training set).

```
model_lg = glm(poswins ~ qtr + down + ydstogo + TimeSecs + yrdline100 + ScoreDiff, train, family = "binomial")

summary(model_lg)
```

General Logistics Model for NFL Wins

Results/Summary

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.97345  -0.83774   0.08863   0.86917   2.97058

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.018e+00  5.451e-02  18.676  < 2e-16 ***
qtr2        -1.658e-02  1.968e-02  -0.843  0.39943
qtr3        -5.383e-02  3.256e-02  -1.653  0.09831 .
qtr4        -1.121e-01  4.728e-02  -2.371  0.01774 *
down2       -8.387e-02  1.096e-02  -7.653 1.96e-14 ***
down3       -1.912e-01  1.280e-02 -14.943  < 2e-16 ***
down4       -3.674e-01  1.562e-02 -23.514  < 2e-16 ***
ydstogo     -9.312e-03  1.164e-03  -8.003 1.21e-15 ***
TimeSecs    -4.825e-05  1.649e-05  -2.927  0.00343 **
yrdline100  -8.685e-03  1.881e-04 -46.176  < 2e-16 ***
ScoreDiff    1.769e-01  7.173e-04 246.624  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 421744  on 304258  degrees of freedom
Residual deviance: 302028  on 304248  degrees of freedom
AIC: 302050
```

Logistic Model Results/Summary

To estimate the win probabilities for each play in my training dataset, I can use my logical regression model. I used the following code to predict the estimate win probabilities for plays in my training set.

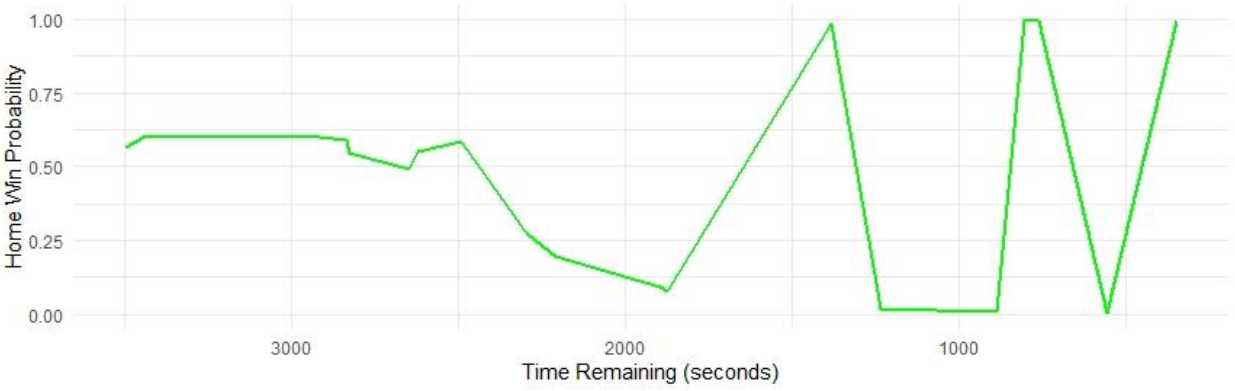
```
pred <- predict(model_lg, train, type <- "response")
train <- cbind(train,pred)
train <- mutate(train, pred1h = ifelse(posteam == HomeTeam, pred, 1 - pred))
```

Plotting In-Game Win Probability

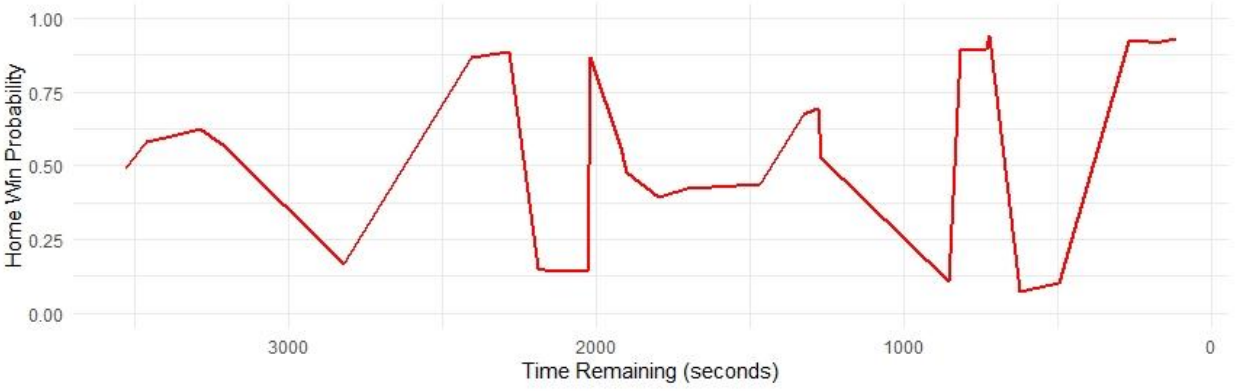
To plot the win probability, I used the ggplot function. The plot will graph probabilities between the binary rang of 0 to 1. The midpoint of 0.5 represents a 50 percent win/loose probability (win represents 1 and 0 represents a loss). The control variable, which is game_id, plot the probability of the home team win/lose probability of the home team over time counting down to the end of the NFL game. The prediction acts as a function that generations the plot with the ggplot () below.

```
ggplot(data=(filter(train, game_id == "2016090800")), aes(x <- TimeSecs,y <- pred)) + geom_line(size = 1, colour = "green") + scale_x_reverse() + ylim(c(0,1)) + theme_minimal() + xlab("Time Remaining (seconds)") + ylab("Home Win Probability")
```

Key: **Green Line** = Home Team Win **Red Line** = Home Team Loss



Houston Texans (vs @ **Baltimore Ravens**) In-Game Win Probability (November 17th, 2019)



Seattle Seahawks (vs @ **San Francisco 49ers**) In-Game Win Probability (November 11th, 2019)

Conclusion

In conclusion, this final project really helped me gain a deeper understanding in all the different research, analysis, development and testing that is required to complete an A.I. project such as this one. I truly feel much more comfortable building model and fitting them to the requirements of the project. The most difficult part of this project was finding the best way to get the data sets. The next difficult process was the data wrangling and manipulation of the test and training sets. I also note that in order for my model to become effective, I had to convert the estimated win probabilities from a logistic regression model into a continuous prediction of win or loss.

Limitations

I feel the biggest limitation to this project was the lower limit of my web crapping to get consistent data before the year 2009. The second limitation of this project was the fact that it only predicted the win and loss probability of the home team.

Future work

This project can continued by adding research from Las Vegas betting odds. It would be interesting to learn about the process of betting organizations in Vegas collect their sports data. Another future work would be to modify this model to include future game predictions that benchmark against Las Vegas betting models.