Choose your Own: Predicting NFL Games

Craig Barnes

7/25/2021

Contents

Intro / Project Overview

The game of American Football is a past time in this country that has grown in popularity over the past few decades. Kids and adults alike will gather on Sundays and even on Thanksgiving Day to cheer on their favorite team. Like many other sports, Las Vegas has fancied putting odds on the game of football, and because of this you can make a lot of money if you're good at guessing who will win the game.

My Choose-your-own Project for the HarvardX Data Science Capstone is built upon the question:

> "Can I predict who should win a game of football based on statistics from previous games?"

I built a front-to-back project that gathers the data, formats it, and then models/predicts on it. You can find this all on my github page. Script #1 (ScrapeNFL.R) is a webscraping script, designed to pull the past few years of football games that have occured. For this project, I pull in 5 years of football games (2015-2019) for the main data set (train and test). I also pull in 2020 to use as a true holdout test on the final models. The main script which can be considered my actual "project" for the purposes of grading (NFLCapstone.R) consists of Cleaning/Transforming, Visualizations/Motivation, Modeling/Results/Conclusions sections.

Part 1: The Data

Below you will see the structure of the original data. This was the result of the web scraping from the 1st Script in my github repository. The full scrape/cleaning takes about 15 minutes, so I decided to pre-run this part of my project and leave the code in my github so you could check it out if you'd like!

You'll see a number of columns, and I'll explain a few of them in some detail below. Please see the data dictionary in the github repository for further clarification as well. This data dictionary will break things down a bit more for those unfamiliar with American football, but it is not a necessity to read it to understand the analysis provided as you move through the report.

```
# download data from github
x <- getURL("https://raw.githubusercontent.com/CraigMBarnes/Capstone/main/NFLdf.csv")
df <- read.csv(text = x)
# take a look at first row of observations and their data types
glimpse(df[1:1,])
```

```
Rows: 1
Columns: 45
$ away                                    <chr> "Pittsburgh Steelers 21"
$ home                                    <chr> "New England Patriots 28"
$ date                                    <chr> "September 10th  2015  "
$ VegasLine                               <chr> "New England Patriots -7.0"
$ OverUnder                               <chr> "52.0 "
$ FirstDowns_Away                         <int> 23
$ FirstDowns_Home                         <int> 26
$ Rushes_RushYards_RushTDs_Away           <chr> "25-134-1"
$ Rushes_RushYards_RushTDs_Home           <chr> "24-80-0"
$ Completions_PassYards_PassTDs_INTs_Away <chr> "26-38-351-1-1"
$ Completions_PassYards_PassTDs_INTs_Home <chr> "25-32-288-4-0"
$ Sacks_Yards_Away                        <chr> "3-21"
$ Sacks_Yards_Home                        <chr> "2-7"
$ Fumbles_Lost_Away                       <chr> "0-0"
$ Fumbles_Lost_Home                       <chr> "1-0"
$ Turnovers_Away                          <int> 1
$ Turnovers_Home                          <int> 0
$ Penalties_Yards_Away                    <chr> "8-77"
$ Penalties_Yards_Home                    <chr> "7-64"
$ ThirdDowns_Away                         <chr> "7-15"
$ ThirdDowns_Home                         <chr> "7-11"
$ TOP_Away                                <chr> "32:05"
$ TOP_Home                                <chr> "27:55"
$ xpm_Away                                <int> 1
$ xpm_Home                                <int> 4
$ xpa_Away                                <int> 1
$ xpa_Home                                <int> 4
$ fgm_Away                                <int> 2
$ fgm_Home                                <int> 0
$ fga_Away                                <int> 4
$ fga_Home                                <int> 0
$ punt_Away                               <int> 2
$ punt_Home                               <int> 4
$ kr_Away                                 <int> 1
$ kr_Home                                 <int> 1
$ kryds_Away                              <int> 22
$ kryds_Home                              <int> 8
$ krtd_Away                               <int> 0
$ krtd_Home                               <int> 0
$ pr_Away                                 <int> 1
$ pr_Home                                 <int> 1
$ pryds_Away                              <int> 3
$ pryds_Home                              <int> 1
$ prtd_Away                               <int> 0
$ prtd_Home                               <int> 0
```

The first few columns above are pretty obvious. Here, I have the away team name and score, the home team name and score, the date of the football game, etc. **Keep in mind** that the main objective of my script is to predict if the home team or away team will win.

away	home	date
Pittsburgh Steelers 21	New England Patriots 28	September 10th 2015

The next two columns are the Las Vegas gambling odds. I decided not to use this feature, although one of my original ideas was to predict if Las Vegas made a "correct" prediction for the actual outcome based on my models. See the conclusion section for more on these variables. You can see below that New England was favored to win the game by 7 points, just as an example.

VegasLine	OverUnder
New England Patriots -7.0	52.0

The remaining columns that follow are what was used to generate the final data set. You will notice a suffix of "_Away" or "_Home" for each set of variables.

FirstDowns_Away	FirstDowns_Home
23	26

TOP_Away	TOP_Home
32:05	27:55

For example, TOP Home represents "Time of Possession" for the home team, or how long they were on offense (i.e. controlled the football). TOP Away represents the same metric, for the away team. I then combined these variables in the cleaning process so that TOP_Home and TOP_Away become d_TOP, and it represents the difference of the two variables or (TOP_Home - TOP_Away). I went on to use the difference between home and away for all variables.

In the above example, the TOP_HOME is 4 minutes and 10 seconds shorter then TOP_Away. You'll see this entry in the final dataset below. The value is -250 for this observation, which is the above difference converted to seconds.

In the next chunk, I will bring in the preprocessed dataset and you will notice the reduction in variable names. The prefix "d" stands for "delta" between home and away. **Homewin** is my target variable, 1 means that the Home team wins, 0 means that the Away team wins. Let's take a look at the final format of the data, and get into motivating figures and visualizations.

Part 2: Motivation

Note: The chunk hidden below is the cleaning up of data from the above set to the final set below. The main cleaning was the transformation of _home and _away suffixes and turning them into the delta between home and away. Additional cleaning involved some regex work, as well as other wrangling techniques we learned throughout the data science certificate coursework, such as functions like "separate", using ifelse statements, and casting data types. If you are interested in looking at the scraping script mentioned above, you'll find a lot of cleaning in that script as well.

```
>       date               HomeWin      d_FirstDowns       d_Rushes
>  Min.   :2015-09-10   Min.   :0.00   Min.   :-23.00   Min.   :-39.00
>  1st Qu.:2016-11-12   1st Qu.:0.00   1st Qu.: -4.00   1st Qu.: -9.00
>  Median :2018-09-07   Median :1.00   Median :  1.00   Median :  1.00
>  Mean   :2018-05-11   Mean   :0.55   Mean   :  0.86   Mean   :  0.74
>  3rd Qu.:2019-11-04   3rd Qu.:1.00   3rd Qu.:  6.00   3rd Qu.: 10.00
>  Max.   :2021-02-07   Max.   :1.00   Max.   : 24.00   Max.   : 38.00
>   d_RushYards       d_RushTDs      d_Completions    d_PassAttempts
>  Min.   :-263.0   Min.   :-6.000   Min.   :-30.00   Min.   :-40.00
```

```
>  1st Qu.: -46.0   1st Qu.:-1.000   1st Qu.: -6.00   1st Qu.: -9.00
>  Median :   4.0   Median : 0.000   Median :  0.00   Median :  0.00
>  Mean   :   5.5   Mean   : 0.126   Mean   :  0.24   Mean   : -0.13
>  3rd Qu.:  56.0   3rd Qu.: 1.000   3rd Qu.:  6.00   3rd Qu.:  9.00
>  Max.   : 246.0   Max.   : 5.000   Max.   : 29.00   Max.   : 52.00
>   d_PassYards       d_PassTDs       d_Sacks        d_Turnovers
>  Min.   :-339.0   Min.   :-6.0   Min.   :-11.00   Min.   :-7.000
>  1st Qu.: -56.0   1st Qu.:-1.0   1st Qu.: -1.00   1st Qu.:-1.000
>  Median :   8.0   Median : 0.0   Median :  0.00   Median : 0.000
>  Mean   :   6.6   Mean   : 0.1   Mean   :  0.14   Mean   : 0.001
>  3rd Qu.:  72.0   3rd Qu.: 1.0   3rd Qu.:  2.00   3rd Qu.: 1.000
>  Max.   : 313.0   Max.   : 5.0   Max.   :  9.00   Max.   : 6.000
>   d_Penalties      d_PenaltyYards    d_ThirdDownpercent  d_XPpercent
>  Min.   :-17.000   Min.   :-138.00   Min.   :-0.6059    Min.   :-1.0000
>  1st Qu.: -3.000   1st Qu.: -25.00   1st Qu.:-0.1076    1st Qu.: 0.0000
>  Median :  0.000   Median :  -5.00   Median : 0.0258    Median : 0.0000
>  Mean   : -0.406   Mean   :  -3.83   Mean   : 0.0183    Mean   : 0.0214
>  3rd Qu.:  2.000   3rd Qu.:  19.00   3rd Qu.: 0.1469    3rd Qu.: 0.0000
>  Max.   : 13.000   Max.   : 128.00   Max.   : 0.6273    Max.   : 1.0000
>  d_KickReturnYards d_KickReturnTDs   d_PuntReturnYards d_PuntReturnTDs
>  Min.   :-176.00   Min.   :-1.0000   Min.   :-145.00   Min.   :-1.0000
>  1st Qu.: -33.00   1st Qu.: 0.0000   1st Qu.: -11.00   1st Qu.: 0.0000
>  Median :   0.00   Median : 0.0000   Median :   1.00   Median : 0.0000
>  Mean   :  -0.14   Mean   : 0.0025   Mean   :   1.39   Mean   :-0.0031
>  3rd Qu.:  33.00   3rd Qu.: 0.0000   3rd Qu.:  14.00   3rd Qu.: 0.0000
>  Max.   : 180.00   Max.   : 1.0000   Max.   : 179.00   Max.   : 2.0000
>   d_FGpercent         d_Punts           d_TOP
>  Min.   :-1.0000   Min.   :-6.000   Min.   :-1598.0
>  1st Qu.:-0.2500   1st Qu.:-1.000   1st Qu.: -340.5
>  Median : 0.0000   Median : 0.000   Median :    9.0
>  Mean   : 0.0171   Mean   :-0.214   Mean   :    8.8
>  3rd Qu.: 0.3333   3rd Qu.: 1.000   3rd Qu.:  374.5
>  Max.   : 1.0000   Max.   : 6.000   Max.   : 1538.0
```

Below is the structure of the set we will focus on the rest of this report. This will be split into Train/Test.

```
df2020 <- dfy %>% filter(date>"2020-7-1") #holdout set of 2020 games
df <- dfy %>% filter(date<"2020-7-1") #main data set to be split
#
ncol(df)
```

```
> [1] 23
```

```
nrow(df)
```

```
> [1] 1335
```

```
glimpse(df[1:5,])
```

```
> Rows: 5
> Columns: 23
> $ date              <date> 2015-09-10, 2015-09-13, 2015-09-13, 2015-09-13,...
> $ HomeWin           <dbl> 1, 1, 0, 0, 0
> $ d_FirstDowns      <int> 3, -2, 4, 5, 1
> $ d_Rushes          <dbl> -1, -6, 3, -11, 19
> $ d_RushYards       <dbl> -54, -48, 56, 1, 87
> $ d_RushTDs         <dbl> -1, 2, 0, 0, 0
```

```
> $ d_Completions      <dbl> -1, -14, 0, 4, -1
> $ d_PassAttempts     <dbl> -6, -14, 13, 14, -3
> $ d_PassYards        <dbl> -63, 46, 36, 91, -30
> $ d_PassTDs          <dbl> 3, 0, -2, -1, 0
> $ d_Sacks            <dbl> 1, 4, -2, -3, 2
> $ d_Turnovers        <int> -1, 2, 1, 2, 1
> $ d_Penalties        <dbl> -1, -3, -4, 4, 5
> $ d_PenaltyYards     <dbl> -13, -16, -10, 14, 49
> $ d_ThirdDownpercent <dbl> 0.16970, 0.12440, 0.04706, -0.01648, 0.01190
> $ d_XPpercent        <dbl> 0, 0, 0, -1, 0
> $ d_KickReturnYards  <dbl> -14, 7, -43, -30, -18
> $ d_KickReturnTDs    <dbl> 0, 0, 0, 0, 0
> $ d_PuntReturnYards  <dbl> -2, 22, 11, -67, -51
> $ d_PuntReturnTDs    <dbl> 0, 0, 0, 0, -1
> $ d_FGpercent        <dbl> -0.5000, 0.0000, 0.0000, 0.3333, -0.5000
> $ d_Punts            <int> 2, -1, -1, 0, -1
> $ d_TOP              <dbl> -250, -536, 224, -638, 948
```

```
summary(df)
```

```
>       date                HomeWin       d_FirstDowns       d_Rushes
>  Min.   :2015-09-10   Min.   :0.000   Min.   :-22.00   Min.   :-39.00
>  1st Qu.:2016-10-09   1st Qu.:0.000   1st Qu.: -4.00   1st Qu.: -8.00
>  Median :2017-11-12   Median :1.000   Median :  1.00   Median :  1.00
>  Mean   :2017-11-08   Mean   :0.561   Mean   :  1.12   Mean   :  1.14
>  3rd Qu.:2018-12-09   3rd Qu.:1.000   3rd Qu.:  6.00   3rd Qu.: 11.00
>  Max.   :2020-02-02   Max.   :1.000   Max.   : 24.00   Max.   : 38.00
>   d_RushYards        d_RushTDs     d_Completions     d_PassAttempts
>  Min.   :-263.00   Min.   :-6.00   Min.   :-30.000   Min.   :-40.00
>  1st Qu.: -45.00   1st Qu.:-1.00   1st Qu.: -5.500   1st Qu.: -9.00
>  Median :   5.00   Median : 0.00   Median :  0.000   Median :  0.00
>  Mean   :   6.69   Mean   : 0.15   Mean   :  0.351   Mean   : -0.08
>  3rd Qu.:  57.00   3rd Qu.: 1.00   3rd Qu.:  6.000   3rd Qu.:  9.00
>  Max.   : 246.00   Max.   : 4.00   Max.   : 29.000   Max.   : 52.00
>   d_PassYards       d_PassTDs         d_Sacks         d_Turnovers
>  Min.   :-333.0   Min.   :-5.000   Min.   :-11.000   Min.   :-7.000
>  1st Qu.: -53.5   1st Qu.:-1.000   1st Qu.: -1.000   1st Qu.:-1.000
>  Median :  11.0   Median : 0.000   Median :  0.000   Median : 0.000
>  Mean   :   9.5   Mean   : 0.118   Mean   :  0.109   Mean   :-0.022
>  3rd Qu.:  76.0   3rd Qu.: 1.000   3rd Qu.:  2.000   3rd Qu.: 1.000
>  Max.   : 278.0   Max.   : 5.000   Max.   :  9.000   Max.   : 6.000
>   d_Penalties      d_PenaltyYards    d_ThirdDownpercent  d_XPpercent
>  Min.   :-17.000   Min.   :-138.00   Min.   :-0.6059    Min.   :-1.0000
>  1st Qu.: -3.000   1st Qu.: -26.00   1st Qu.:-0.1060    1st Qu.: 0.0000
>  Median :  0.000   Median :  -4.00   Median : 0.0321    Median : 0.0000
>  Mean   : -0.421   Mean   :  -4.03   Mean   : 0.0222    Mean   : 0.0143
>  3rd Qu.:  2.000   3rd Qu.:  19.00   3rd Qu.: 0.1552    3rd Qu.: 0.0000
>  Max.   : 13.000   Max.   : 128.00   Max.   : 0.5417    Max.   : 1.0000
>  d_KickReturnYards d_KickReturnTDs   d_PuntReturnYards d_PuntReturnTDs
>  Min.   :-176.0    Min.   :-1.0000   Min.   :-141.00   Min.   :-1.0000
>  1st Qu.: -33.0    1st Qu.: 0.0000   1st Qu.: -11.00   1st Qu.: 0.0000
>  Median :   0.0    Median : 0.0000   Median :   1.00   Median : 0.0000
>  Mean   :  -0.6    Mean   : 0.0022   Mean   :   1.53   Mean   :-0.0022
>  3rd Qu.:  33.0    3rd Qu.: 0.0000   3rd Qu.:  15.00   3rd Qu.: 0.0000
>  Max.   : 180.0    Max.   : 1.0000   Max.   : 179.00   Max.   : 2.0000
```

```
>   d_FGpercent         d_Punts           d_TOP
>  Min.   :-1.0000   Min.   :-6.000   Min.   :-1598
>  1st Qu.:-0.2500   1st Qu.:-2.000   1st Qu.: -318
>  Median : 0.0000   Median : 0.000   Median :   28
>  Mean   : 0.0151   Mean   :-0.258   Mean   :   29
>  3rd Qu.: 0.3333   3rd Qu.: 1.000   3rd Qu.:  386
>  Max.   : 1.0000   Max.   : 6.000   Max.   : 1538
```

The below set contains games from the current NFL season. This will be saved until the very end.

```
glimpse(df2020[1:5,])
```

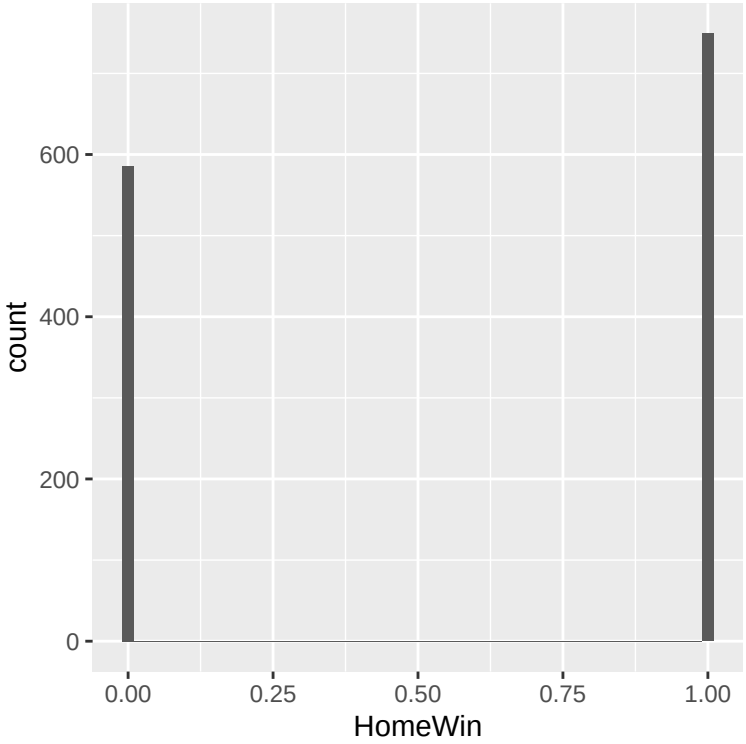
```
> Rows: 5
> Columns: 23
> $ date                <date> 2020-09-10, 2020-09-13, 2020-09-13, 2020-09-13,...
> $ HomeWin             <dbl> 1, 0, 1, 1, 0
> $ d_FirstDowns        <int> 7, 6, 16, -1, -6
> $ d_Rushes            <dbl> 12, 1, 17, 19, -10
> $ d_RushYards         <dbl> 48, -12, 46, 23, -24
> $ d_RushTDs           <dbl> -1, 0, 0, 2, 1
> $ d_Completions       <dbl> 4, 6, 12, -7, -13
> $ d_PassAttempts      <dbl> 0, 19, 11, -11, -19
> $ d_PassYards         <dbl> -42, 128, 97, -92, -105
> $ d_PassTDs           <dbl> 2, -2, 1, -1, -2
> $ d_Sacks             <dbl> 3, 1, 0, 5, -2
> $ d_Turnovers         <int> -1, 2, 0, -3, 1
> $ d_Penalties         <dbl> -4, 0, -2, 4, -1
> $ d_PenaltyYards      <dbl> -32, 26, -16, 35, -43
> $ d_ThirdDownpercent <dbl> 0.13846, 0.16667, 0.13636, -0.07937, -0.04545
> $ d_XPpercent         <dbl> 0.0, -0.5, 0.0, 0.0, 0.0
> $ d_KickReturnYards   <dbl> -18, -43, -1, 0, 46
> $ d_KickReturnTDs     <dbl> 0, 0, 0, 0, 0
> $ d_PuntReturnYards   <dbl> -19, -7, 68, 3, 0
> $ d_PuntReturnTDs     <dbl> 0, 0, 0, 0, 0
> $ d_FGpercent         <dbl> 1.0000, 0.0000, -0.5000, 0.1667, 0.0000
> $ d_Punts             <int> 0, -2, -5, 2, 1
> $ d_TOP               <dbl> 574, -72, 1354, -78, -1352
```

Judging by the data, we can get a baseline of 56% accuracy by simply picking the Home team to win every time. Clearly we can do better than that!

```
mean(df$HomeWin)
```

```
> [1] 0.561
```

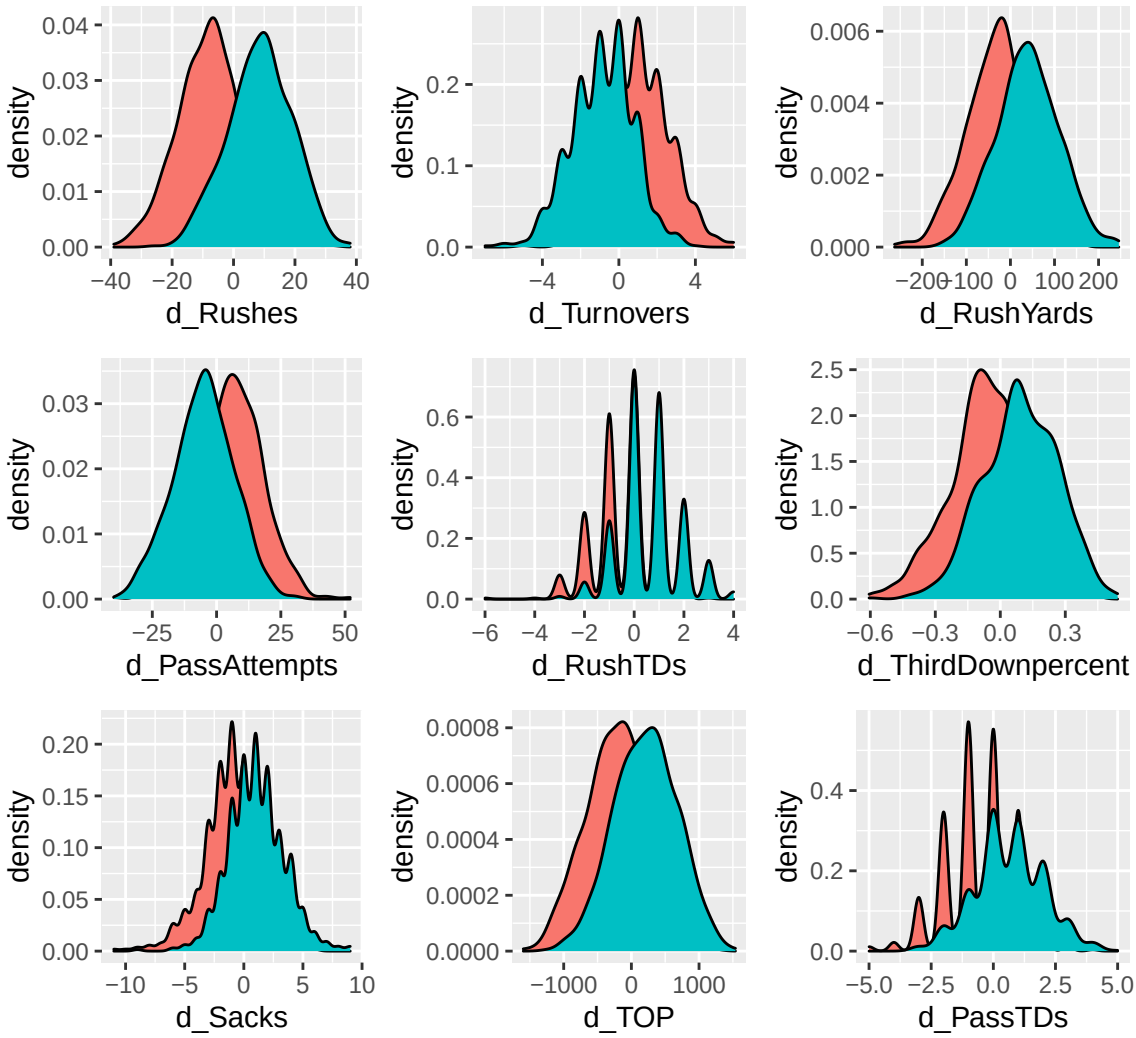
```
df %>% ggplot(aes(x=HomeWin)) + geom_histogram(bins = 50)
```



When I produce a correlation dataframe, I can see that there are some very strong predictors here. The below values show the absolute value of the r stat from the correlation on each feature to the HomeWin predictor.

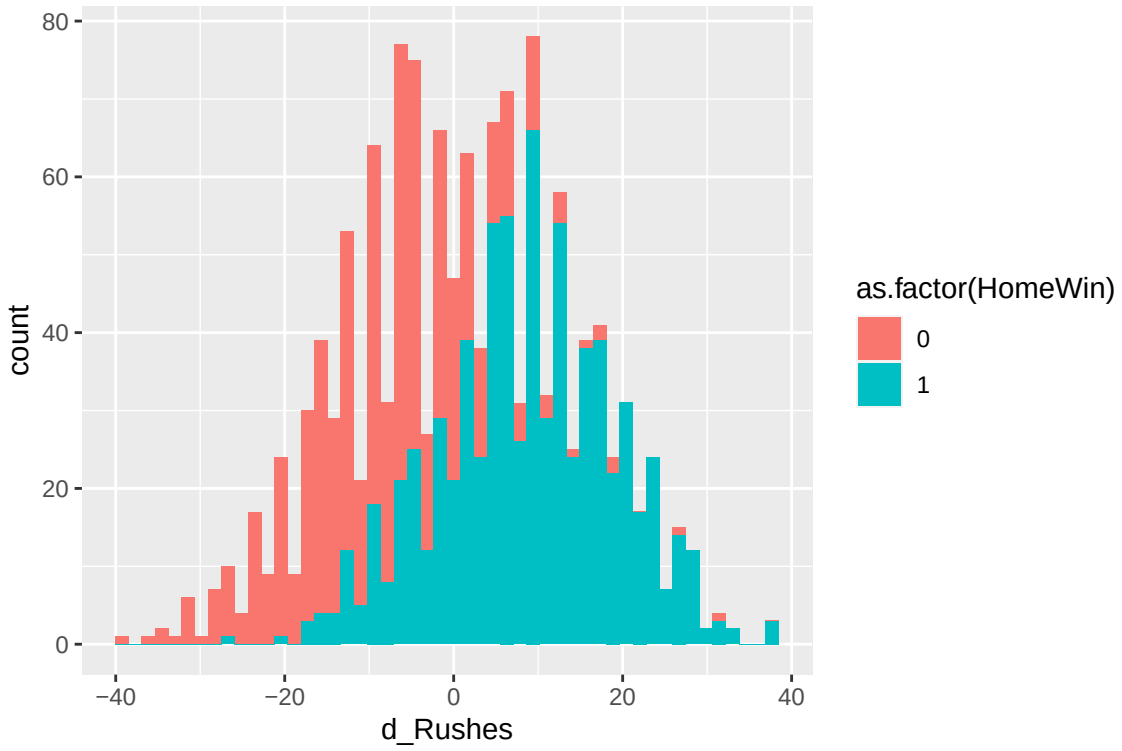
```
>                    x       r
> 1            d_Rushes 0.63331
> 2         d_Turnovers 0.51049
> 3         d_RushYards 0.47215
> 4      d_PassAttempts 0.43081
> 5           d_RushTDs 0.42529
> 6  d_ThirdDownpercent 0.41993
> 7             d_Sacks 0.41350
> 8               d_TOP 0.40464
> 9           d_PassTDs 0.39607
> 10  d_KickReturnYards 0.35928
> 11       d_FirstDowns 0.32266
> 12        d_XPpercent 0.29096
> 13            d_Punts 0.27751
> 14        d_FGpercent 0.24218
> 15  d_PuntReturnYards 0.18566
> 16      d_Completions 0.18504
> 17        d_Penalties 0.08792
> 18    d_PuntReturnTDs 0.06986
> 19     d_PenaltyYards 0.05680
> 20        d_PassYards 0.04309
> 21    d_KickReturnTDs 0.04145
```

Using a facet plot, we can investigate some of the highest correlated features and how large the separation is between the classes of HomeWin. . .



We can see there is a very large spread between HomeWin = 1 and HomeWin = 0 for the d_Rushes feature.

```
df %>% ggplot(aes(x=d_Rushes, fill = as.factor(HomeWin)))+geom_histogram(bins=50)
```




```
df %>% filter(HomeWin==1) %>% summarise(m=mean(d_Rushes))
```

```
>       m
> 1 8.569
```

```
df %>% filter(HomeWin==0) %>% summarise(m=mean(d_Rushes))
```

```
>        m
> 1 -8.355
```

From the above, we can see that when the home team wins the football game (`HomeWin == 1`), the mean of d_Rushes is approximately 8.57, while when the home team loses (`HomeWin == 0`) this mean is -8.35. This spread of almost 17 rushes is quite remarkable. This feature is the primary driver of my research, and sparked the curiosity in me that led to my choosing of this project.

Part 3: Modeling

Note that this section will include substantially more on screen R-code, as I want to convey the different methods and configurations that I experimented with.

Below I will split train and test.

```
df$HomeWin <- ifelse(df$HomeWin==1,2,1)
df$HomeWin <- as.factor(df$HomeWin)
df <- df %>% select(-date)
#
set.seed(33, sample.kind = "Rounding")
test_index <- createDataPartition(df$HomeWin, times = 1, p = 0.15, list = FALSE)
test_set <- df[test_index, ]
testx <- test_set %>% select(-HomeWin)
train_set <- df[-test_index, ]
trainx <- train_set %>% select(-HomeWin)
```

```
# Below is the test set structure
str(test_set)
```

```
> 'data.frame': 201 obs. of  22 variables:
>  $ HomeWin           : Factor w/ 2 levels "1","2": 2 1 2 1 2 2 1 1 2 1 ...
>  $ d_FirstDowns      : int  -6 -6 -3 8 18 10 3 8 -1 10 ...
>  $ d_Rushes          : num  -6 1 3 4 8 2 0 -3 23 -8 ...
>  $ d_RushYards       : num  -27 41 73 -45 36 61 -13 8 128 -103 ...
>  $ d_RushTDs         : num  3 -2 3 0 1 0 0 1 2 -1 ...
>  $ d_Completions     : num  -12 -3 -13 9 11 11 0 15 -10 22 ...
>  $ d_PassAttempts    : num  -19 -6 -16 14 13 13 3 21 -14 22 ...
>  $ d_PassYards       : num  34 -71 -188 107 244 172 -42 163 -94 162 ...
>  $ d_PassTDs         : num  1 1 -2 0 2 1 -1 0 -3 1 ...
>  $ d_Sacks           : num  5 0 4 0 1 -2 -3 0 4 1 ...
>  $ d_Turnovers       : int  -1 1 -1 0 -3 0 1 4 -2 1 ...
>  $ d_Penalties       : num  0 0 -4 -5 -2 -1 0 0 7 -1 ...
>  $ d_PenaltyYards    : num  12 -30 -35 -16 -21 1 -19 3 60 -32 ...
>  $ d_ThirdDownpercent: num  0.1882 -0.0588 -0.1373 -0.0559 0.1282 ...
>  $ d_XPpercent       : num  -0.25 0 0 0 0 ...
>  $ d_KickReturnYards : num  -58 46 34 0 -71 78 0 66 -93 -32 ...
>  $ d_KickReturnTDs   : num  0 0 0 0 0 1 0 0 0 0 ...
>  $ d_PuntReturnYards : num  5 56 38 -1 -23 73 15 10 77 -22 ...
>  $ d_PuntReturnTDs   : num  0 0 0 0 0 0 0 0 0 0 ...
>  $ d_FGpercent       : num  -1 0 1 1 1 ...
>  $ d_Punts           : int  -1 -1 -2 0 -2 -6 0 -2 -1 -3 ...
>  $ d_TOP             : num  -838 -190 -418 598 784 312 136 260 562 446 ...
```

```
# Below is the train set structure
str(train_set)
```

```
> 'data.frame': 1134 obs. of  22 variables:
>  $ HomeWin           : Factor w/ 2 levels "1","2": 2 2 1 1 1 2 1 2 2 2 ...
>  $ d_FirstDowns      : int  3 -2 4 5 1 -8 1 3 7 12 ...
>  $ d_Rushes          : num  -1 -6 3 -11 19 19 -14 8 5 15 ...
>  $ d_RushYards       : num  -54 -48 56 1 87 83 -9 50 66 27 ...
>  $ d_RushTDs         : num  -1 2 0 0 0 2 0 2 1 1 ...
>  $ d_Completions     : num  -1 -14 0 4 -1 -12 4 -3 -11 15 ...
>  $ d_PassAttempts    : num  -6 -14 13 14 -3 -30 9 -8 -16 11 ...
>  $ d_PassYards       : num  -63 46 36 91 -30 -48 8 -52 -48 157 ...
>  $ d_PassTDs         : num  3 0 -2 -1 0 -1 0 1 2 0 ...
>  $ d_Sacks           : num  1 4 -2 -3 2 2 -3 3 2 -1 ...
>  $ d_Turnovers       : int  -1 2 1 2 1 -3 2 -4 0 1 ...
>  $ d_Penalties       : num  -1 -3 -4 4 5 6 0 -8 -2 1 ...
>  $ d_PenaltyYards    : num  -13 -16 -10 14 49 64 3 -79 -43 11 ...
>  $ d_ThirdDownpercent: num  0.1697 0.1244 0.0471 -0.0165 0.0119 ...
>  $ d_XPpercent       : num  0 0 0 -1 0 1 -1 0 0 -0.25 ...
>  $ d_KickReturnYards : num  -14 7 -43 -30 -18 -58 40 -14 33 -81 ...
>  $ d_KickReturnTDs   : num  0 0 0 0 0 0 0 0 0 0 ...
>  $ d_PuntReturnYards : num  -2 22 11 -67 -51 5 -37 3 13 -34 ...
>  $ d_PuntReturnTDs   : num  0 0 0 0 -1 0 0 0 0 0 ...
>  $ d_FGpercent       : num  -0.5 0 0 0.333 -0.5 ...
>  $ d_Punts           : int  2 -1 -1 0 -1 1 1 1 0 -3 ...
>  $ d_TOP             : num  -250 -536 224 -638 948 112 -492 -200 -408 984 ...
```

```
#
# Below is a confusion matrix of predicting the home team always wins
```

```
Homealwayswins <- test_set %>% mutate(yhat=as.factor(2))
confusionMatrix(data = Homealwayswins$yhat, reference = Homealwayswins$HomeWin)
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1   0   0
>          2  88 113
>
>                Accuracy : 0.562
>                  95% CI : (0.491, 0.632)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : 0.529
>
>                   Kappa : 0
>
>  Mcnemar's Test P-Value : <0.0000000000000002
>
>             Sensitivity : 0.000
>             Specificity : 1.000
>          Pos Pred Value :   NaN
>          Neg Pred Value : 0.562
>              Prevalence : 0.438
>          Detection Rate : 0.000
>    Detection Prevalence : 0.000
>       Balanced Accuracy : 0.500
>
>        'Positive' Class : 1
>
```

We can see that the method of simply saying the home team should win every game shows an accuracy of ~56% here as well (of course!).

The first modeling method I tried was GLM.

```
> [1] "The accuracy of GLM model is: 0.9353"
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  83   8
>          2   5 105
>
>                Accuracy : 0.935
>                  95% CI : (0.892, 0.965)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
>
>                   Kappa : 0.869
>
>  Mcnemar's Test P-Value : 0.579
>
>             Sensitivity : 0.943
>             Specificity : 0.929
```

```
>          Pos Pred Value : 0.912
>          Neg Pred Value : 0.955
>              Prevalence : 0.438
>          Detection Rate : 0.413
>    Detection Prevalence : 0.453
>       Balanced Accuracy : 0.936
>
>        'Positive' Class : 1
>
```

Next Up was LOESS

```
> [1] "The accuracy of LOESS model is: 0.9154"
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  80   9
>          2   8 104
>
>                Accuracy : 0.915
>                  95% CI : (0.868, 0.95)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
>
>                   Kappa : 0.828
>
>  Mcnemar's Test P-Value : 1
>
>             Sensitivity : 0.909
>             Specificity : 0.920
>          Pos Pred Value : 0.899
>          Neg Pred Value : 0.929
>              Prevalence : 0.438
>          Detection Rate : 0.398
>    Detection Prevalence : 0.443
>       Balanced Accuracy : 0.915
>
>        'Positive' Class : 1
>
```

Next Up was SVM

```
> [1] "The accuracy of SVM (classification, linear kernel) is: 0.9154"
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  79   8
>          2   9 105
>
>                Accuracy : 0.915
>                  95% CI : (0.868, 0.95)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
```

```
>
>                   Kappa : 0.828
>
>  Mcnemar's Test P-Value : 1
>
>             Sensitivity : 0.898
>             Specificity : 0.929
>          Pos Pred Value : 0.908
>          Neg Pred Value : 0.921
>              Prevalence : 0.438
>          Detection Rate : 0.393
>    Detection Prevalence : 0.433
>       Balanced Accuracy : 0.913
>
>        'Positive' Class : 1
>
```

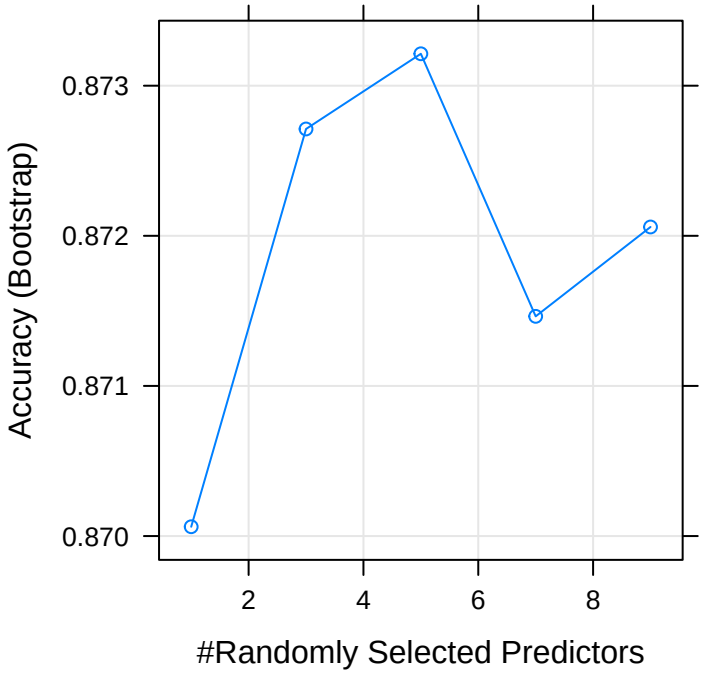
Next up is Random Forest, and Random Forest with Cross-Validation

```
> [1] "The accuracy of Random Forest is: 0.9055"

> [1] "The best value of mtry for RF is: 5"

> [1] "a look at variable importance..."

>                       Overall
> d_Rushes           100.00000
> d_Turnovers         48.51897
> d_PassAttempts      44.31690
> d_RushYards         31.78123
> d_PassTDs           31.62446
> d_ThirdDownpercent  29.63766
> d_Sacks             25.92749
> d_KickReturnYards   23.20331
> d_RushTDs           18.96907
> d_TOP               18.91801
> d_PassYards         15.09293
> d_PuntReturnYards   12.67678
> d_FirstDowns        11.84514
> d_PenaltyYards      11.63965
> d_Completions       10.57826
> d_FGpercent          9.69316
> d_Punts              8.93356
> d_Penalties          7.49400
> d_XPpercent          7.30880
> d_PuntReturnTDs      0.02346
> d_KickReturnTDs      0.00000
```




```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  79  10
>          2   9 103
>
>                Accuracy : 0.905
>                  95% CI : (0.856, 0.942)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
>
>                   Kappa : 0.808
>
>  Mcnemar's Test P-Value : 1
>
>             Sensitivity : 0.898
>             Specificity : 0.912
>          Pos Pred Value : 0.888
>          Neg Pred Value : 0.920
>              Prevalence : 0.438
>          Detection Rate : 0.393
>    Detection Prevalence : 0.443
>       Balanced Accuracy : 0.905
>
>        'Positive' Class : 1
>

> [1] "and with Cross-Validation..."

> [1] "The accuracy of Random Forest w/CV is: 0.9154"

> [1] "The best value of mtry for RF w/CV is: 5"
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  82  11
>          2   6 102
>
>                Accuracy : 0.915
>                  95% CI : (0.868, 0.95)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
>
>                   Kappa : 0.829
>
>  Mcnemar's Test P-Value : 0.332
>
>             Sensitivity : 0.932
>             Specificity : 0.903
>          Pos Pred Value : 0.882
>          Neg Pred Value : 0.944
>              Prevalence : 0.438
>          Detection Rate : 0.408
>    Detection Prevalence : 0.463
>       Balanced Accuracy : 0.917
>
>        'Positive' Class : 1
>
```

Lastly is XGBoost for classification.

```
> [1] "The accuracy of XGBoost for Classification is: 0.9055"
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  81  12
>          2   7 101
>
>                Accuracy : 0.905
>                  95% CI : (0.856, 0.942)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
>
>                   Kappa : 0.809
>
>  Mcnemar's Test P-Value : 0.359
>
>             Sensitivity : 0.920
>             Specificity : 0.894
>          Pos Pred Value : 0.871
>          Neg Pred Value : 0.935
>              Prevalence : 0.438
>          Detection Rate : 0.403
>    Detection Prevalence : 0.463
```

```
>       Balanced Accuracy : 0.907
>
>        'Positive' Class : 1
>
```

To recap:

```
> [1] "GLM: 0.9353"
```

```
> [1] "LOESS: 0.9154"
```

```
> [1] "SVM: 0.9154"
```

```
> [1] "Random Forest: 0.9055"
```

```
> [1] "Random Forest w/CV: 0.9154"
```

```
> [1] "XGBoost: 0.9055"
```

Now, to highlight another skill we learned in the machine learning course, I try an Ensemble method. I use all the models above since they are all above 90% accuracy, and then go with majority. There are 6 models, so if it is a true "tie" and 3 models predict `HomeWin==2` and the other 3 predict `HomeWin==1`, then that row will predict `HomeWin==2`.

```
>      glm_preds loess_preds SVM_preds rf_preds rf_cv_preds XGB_preds yhat truey
> 40           2           2         2        2           2         2    2     1
> 103          2           1         2        1           1         1    1     2
> 443          2           2         2        2           2         2    2     1
> 675          2           2         2        2           2         2    2     1
> 720          2           1         2        2           1         2    2     1
> 782          1           1         1        1           1         2    1     2
> 920          2           1         2        1           1         1    1     2
> 975          2           1         2        1           1         1    1     2
> 1034         1           1         1        1           1         1    1     2
> 1130         1           1         1        2           2         1    1     2
> 1137         2           2         2        1           1         2    2     1
> 1179         1           1         1        2           2         2    2     1
> 1197         1           1         1        1           1         1    1     2
> 1204         1           1         1        1           1         1    1     2
```

Below we can see that the accuracy of the ensemble is good, but not higher than the single GLM model. (The last time I ran it I saw ~93% accuracy.) I printed the observations with incorrect predictions above. The confusion matrix is below.

```
mean(ensemble$yhat==test_set$HomeWin)
```

```
> [1] 0.9303
```

```
confusionMatrix(data = ensemble$yhat, reference = test_set$HomeWin)
```

```
> Confusion Matrix and Statistics
>
>           Reference
> Prediction   1   2
>          1  82   8
>          2   6 105
>
>                Accuracy : 0.93
>                  95% CI : (0.886, 0.961)
>     No Information Rate : 0.562
>     P-Value [Acc > NIR] : <0.0000000000000002
```

```
>
>                  Kappa : 0.859
>
>  Mcnemar's Test P-Value : 0.789
>
>             Sensitivity : 0.932
>             Specificity : 0.929
>          Pos Pred Value : 0.911
>          Neg Pred Value : 0.946
>              Prevalence : 0.438
>          Detection Rate : 0.408
>    Detection Prevalence : 0.448
>       Balanced Accuracy : 0.931
>
>        'Positive' Class : 1
>
```

Last but not least... the holdout set! I attempted each model individually, and then the ensemble method.

```
> [1] "GLM model on holdout:  0.9368"

> [1] "LOESS model on holdout:  0.948"

> [1] "SVM model on holdout:  0.9257"

> [1] "Random Forest model on holdout:  0.881"

> [1] "RF w/Cross-Validation model on holdout:  0.881"

> [1] "XGBoost model on holdout:  0.9145"

> [1] "Ensemble model on holdout:  0.9294"
```

```
>     hoglm holoess hosvm horf horfcv hoxgb yhat truey
> 13      1       1     2    2      2     2    2     1
> 30      2       2     2    2      2     2    2     1
> 34      2       2     2    2      2     2    2     1
> 35      1       1     1    1      1     1    1     2
> 36      1       1     1    2      2     2    2     1
> 37      2       2     2    2      2     1    2     1
> 64      2       2     1    1      1     1    1     2
> 75      1       1     1    1      1     1    1     2
> 76      1       1     1    1      1     1    1     2
> 81      1       2     1    1      1     1    1     2
> 93      2       1     2    2      2     2    2     1
> 109     2       2     2    2      2     2    2     1
> 115     1       1     1    1      1     1    1     2
> 130     2       2     2    2      2     2    2     1
> 160     2       2     2    2      2     2    2     1
> 179     1       1     1    2      2     2    2     1
> 180     2       1     1    2      2     2    2     1
> 229     1       1     1    1      1     1    1     2
> 254     2       2     2    2      2     2    2     1
```

The incorrect predictions on the holdout are above.

Part 4: Results

The accuracy percentages I was able to achieve were higher than my expectation when I started this project. Out of all models, the GLM method was the most accurate on the test set and a number of models performed very well on the holdout. If I were to build a model to implement for a real-life use case, I would probably explore something resembling the ensemble method to avoid overfitting, but simplifying it a bit to something like GLM+SVM+XGBoost.

Also, it is noteworthy and encouraging that both GLM and SVM was able to hit ~93% accuracy on both the test set and holdout set!

Lastly, after substituting the hold out data into the model, the Ensemble performed at a ~93% accuracy. In the real world, if I were to predict winners at 93% accuracy, I would be extremely successful at gambling in Las Vegas.

Let's take a look at the holdout set with predictions brought in to understand what was causing incorrect predictions:

You can see that the average d_Rushes between correct and incorrect predictions is ~.5 off...

```
> [1] "Incorrect Predictions: 1.158"
```

```
> [1] "Correct Predictions: -1.404"
```

The average d_Rushes between correct and incorrect predictions if the Home Team wins...

```
> [1] "Incorrect Predictions: -10.429"
```

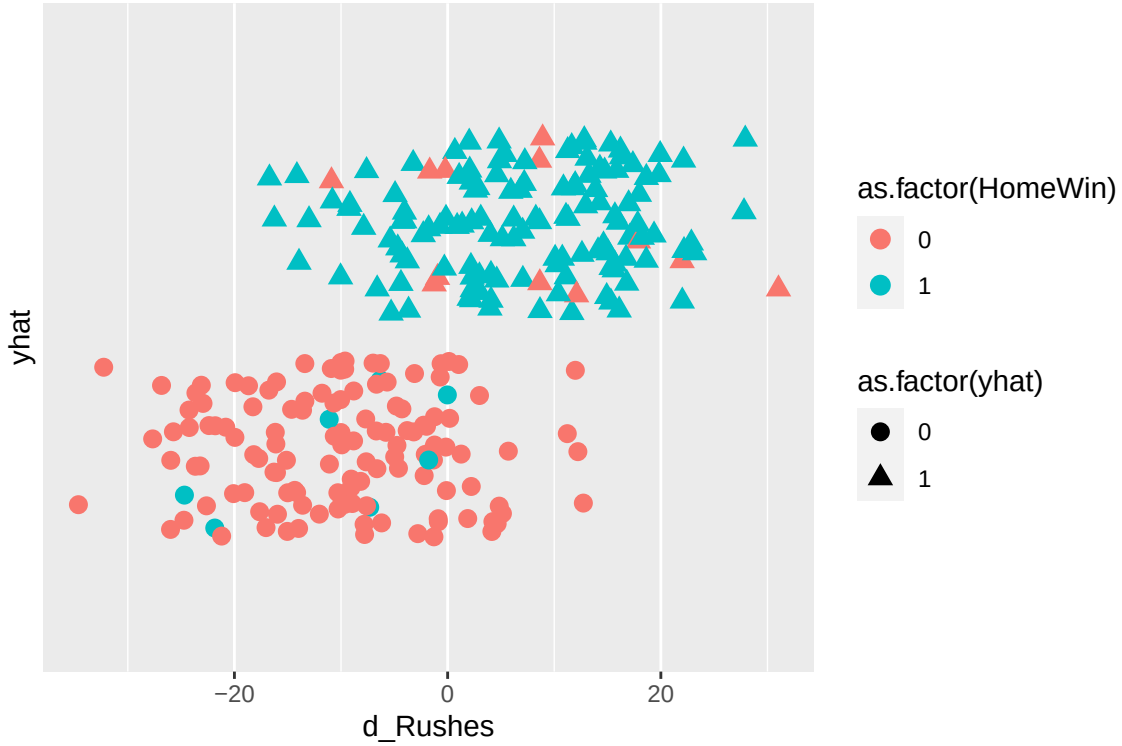
```
> [1] "Correct Predictions: 7.024"
```

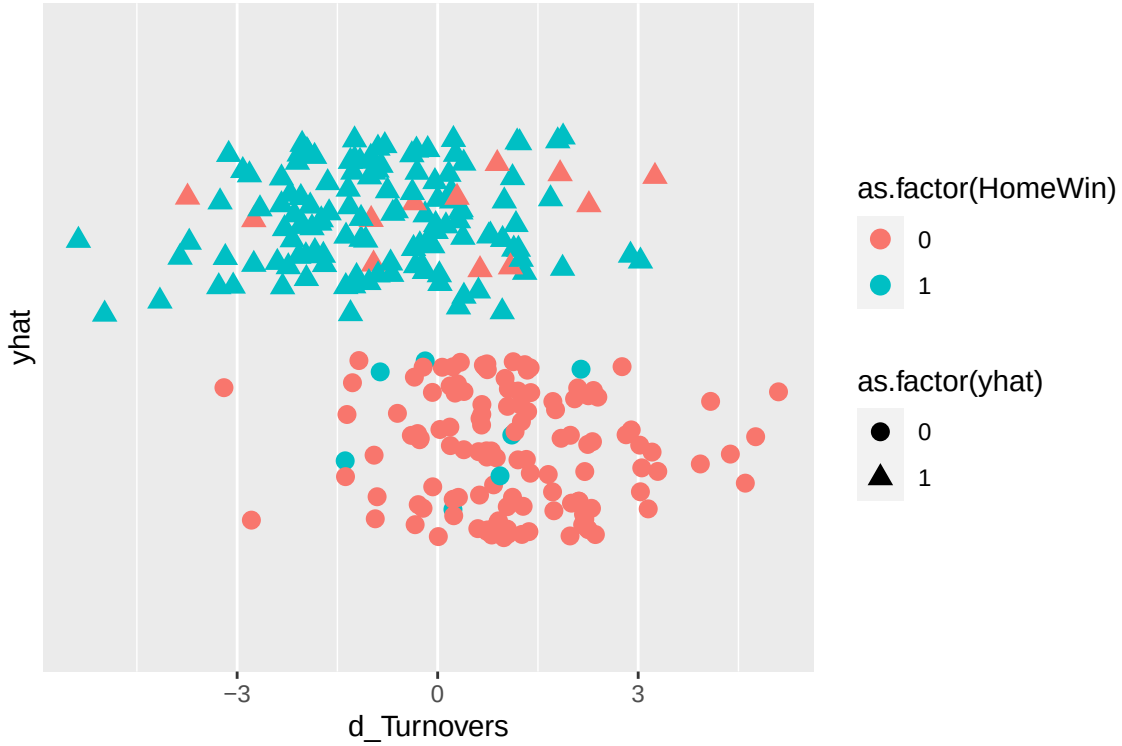
The average d_Rushes between correct and incorrect predictions if the Home Team loses...

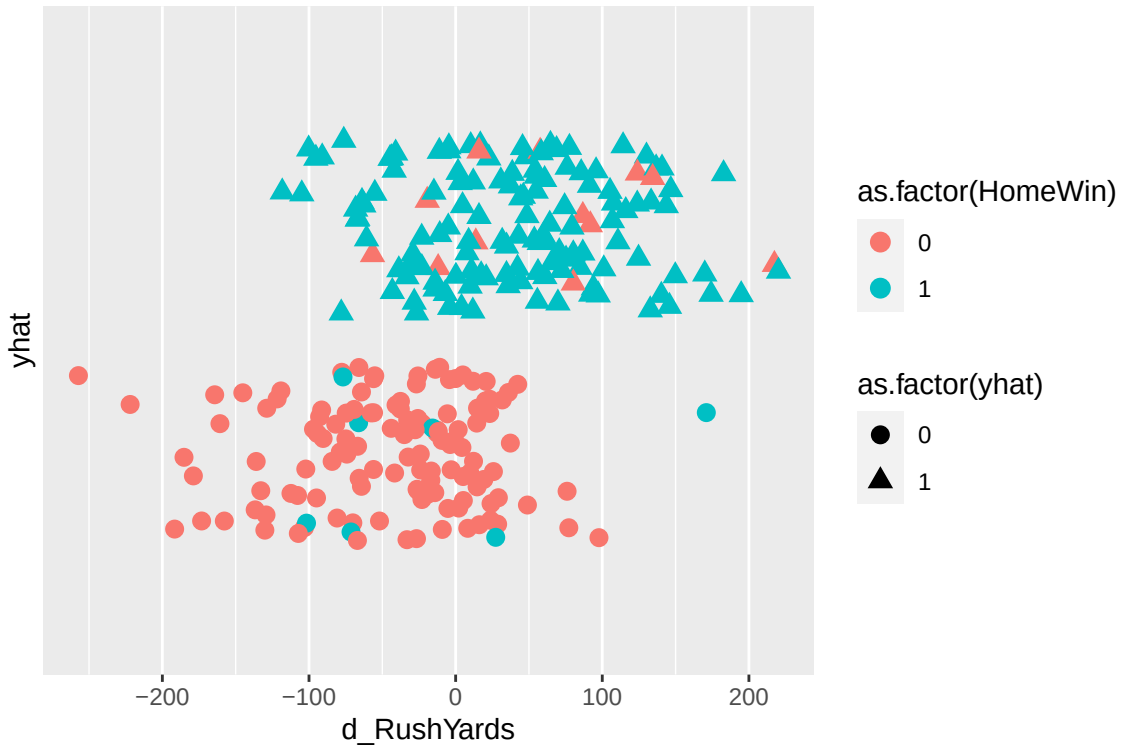
```
> [1] "Incorrect Predictions: 7.917"
```

```
> [1] "Correct Predictions: -9.968"
```

We can investigate predictions vs actuals. Below you will see a few visualizations – the triangles are a Home Team Wins prediction, and Red color means the Home Team lost... Plotting out the top correlated features, you will find d_Rushes, d_Turnovers, and d_RushYards on the x-axis of the following vizzes. We can get a sense of where the inaccurate predictions fall for these values. There are definitely some outliers noticeable in all three graphs.


yhat
−20
0
20
d_Rushes
as.factor(HomeWin)
0
1
as.factor(yhat)
0
1


yhat
−3
0
3
d_Turnovers
as.factor(HomeWin)
0
1
as.factor(yhat)
0
1



Concluding Remarks

Overall, this exercise was extremely fun and reinforced so many of the skills I learned in this certificate program.

Some specific call-outs:

1. I only used accuracy as measurement. This is a binary exercise, and no need to measure error other than Correct/Incorrect predictions.
2. I attempted to use this data set as is, and also with scaled features. There was no value from scaling, so I decided for the sake of time and space to omit a whole section to show scaled vs unscaled models.
3. I did some feature selection exercises once I had gotten through a ton of modeling experimentation, but removing features didn't substantially change accuracy of HomeWin predictions. PCA and reverse-feature selection were considered... but the model did not improve enough to justify a section for this. To simplify data collection and implementation, I could consider that for a future model.
4. In my review of the Vegas odds column compared to the true winner, Vegas is only right ~60% of the time, which didn't seem like a good enough target. For reference, VegasLine refers to "the spread", which shows which team is favored to win and by how many points. (i.e. New England predicted to win by 7 points or more) There is also another Vegas metric called Over/Under, where they set a margin of how many points total would be scored by both teams combined.

Ultimately, I would want to use the modeling from this research to feed an algorithm that would look at the specific two teams within a contest. For example, if the New England Patriots and Pittsburgh Steelers are playing, I should know their average statistics from recent games in order to forecast what they will do in the matchup in question... then I can feed those recent average stats into the model and it would give me a prediction. That is slightly more advanced than this exercise called for, but something I might try independently when I have some time.

Thank you for following along, and I welcome any feedback that you have.

Craig Barnes | craig.michael.barnes@gmail.com