Context / Problem Statement

One of the challenging things about doing data analysis of NFL Football (and sports data analysis in general) is that since the sport is highly situational, it makes much of the basic game data hard to value without proper context. Certainly, scoring a touchdown is generally going to be a positive statistic for a team, as it scores points for that team, and the object of the game is to score more than other team.

However, consider a player running for 4 yards (one of the more common plays of a football game). Is this considered a positive or negative play for that player's team? The answer completely depends on the context. Is the player's team facing a 3rd down and 1 yard to go, where four yards will bring a new set of downs? In that case, it very likely is a positive play. However, if the player's team is facing a 3rd and 20, they will likely have to punt after a 4 yard run, thus giving the ball away, and it is a negative play. Therefore, the data that says a player ran for four yards is impossible to assign a proper value without additional, contextual data so that we can truly know if that play brought the team close to the ultimate goal, which is simply to win the game.

Despite this truth, the way the NFL community has typically analyzed game statistics ignores context, out of either a lack of availability, ignorance, or both. After all, it is much easier to calculate the total amount of yards a player has gained running, than to go play by play and manually assign each play a value of "positive" or "negative". And generally speaking, gaining yards is an indicator of success, since it states that the ball was moved toward the opponent's goal, thus moving the team closer to scoring, and scoring more than the other team and winning is the object of the game. But if we stop there, we run the risk of treating all 4 yard runs as equal, when any football coach, player, or even fan could tell you that this clearly not the case.

Background Research

In recent years, solid progress has been made addressing this problem via a variety of efforts to encapsulate critical "game state" information that can be used to contextualize game data, and by doing so taking a step toward weighing plays by their importance. One of the first was Defense-adjusted Value Over Average (DVOA) by the football analytics website Football Outsiders (http://www.footballoutsiders.com/info/methods) , which manually encoded each play according to a proprietary scoring scale, and summed the results by player, game, or season. DVOA offers value as a simple heuristic, but lacks some of the statistical rigor of more modern approaches.

A more recent attempt has been to incorporate the idea of Win Probability (WP), and its derivative, Win Probability Added (WPA). The idea of Win Probability goes back to mid 20th century baseball, but was developed and widely introduced to NFL analysis by Brian Burke at his former site Advanced Football Analytics (http://www.advancedfootballanalytics.com/index.php/home/stats/stats-explained/win-probability-and-wpa). Since then, a variety of other approaches have emerged, summarized by Michael Lopez at his personal blog (https://statsbylopez.com/2017/03/08/all-win-probability-models-are-wrong-some-are-useful/).

Although the models surveyed by Lopez include a variety of implementations with regard to both data and model selection, most follow a similar approach. A sample of recent NFL plays are collected (typically no older than from the 2000 season, as that seems to be an inflection point where NFL data became widely available via the internet). These data sets will include play by play game logs that include game state data encoded, such as (but not limited to):

- Home Score

- Away Score
- Score Differential
- Total Points Scored
- Down
- Yards to go (for first Down)
- Time Remaining
- Las Vegas Point Spread
- Timeouts Remaining

Once play data was selected and encoded, a model was fit on the data. Varying levels of transparency have been provided, but among the approaches were random forest (Lock and Nettleton), ensemble methods (Burke), logistic regression, and others.

Theory

My goal for this analysis was to draw on the aforementioned research to fit a well performing Win Probability estimator model. The model was to use game state information as input, and output a response variable between 0 and 1. 0 was to indicate the team on offense had no chance to win the game given current game state variables, while 1 was to indicate the team on offense was sure to win. However, analytically, neither is hardly ever the case, so generally the value would be continuous in the range of 0..1. Next, my plan was to fit that model to a series of plays, and use the difference in Win Probability for consecutive plays to derive Win Probability Added.

From there, my theory was that a positive aggregate WPA score over a given subset of plays (game/player/season) would have a stronger relationship with wins than traditional statistics. Specifically, I chose to compare total WPA for QB during a game on passing plays to

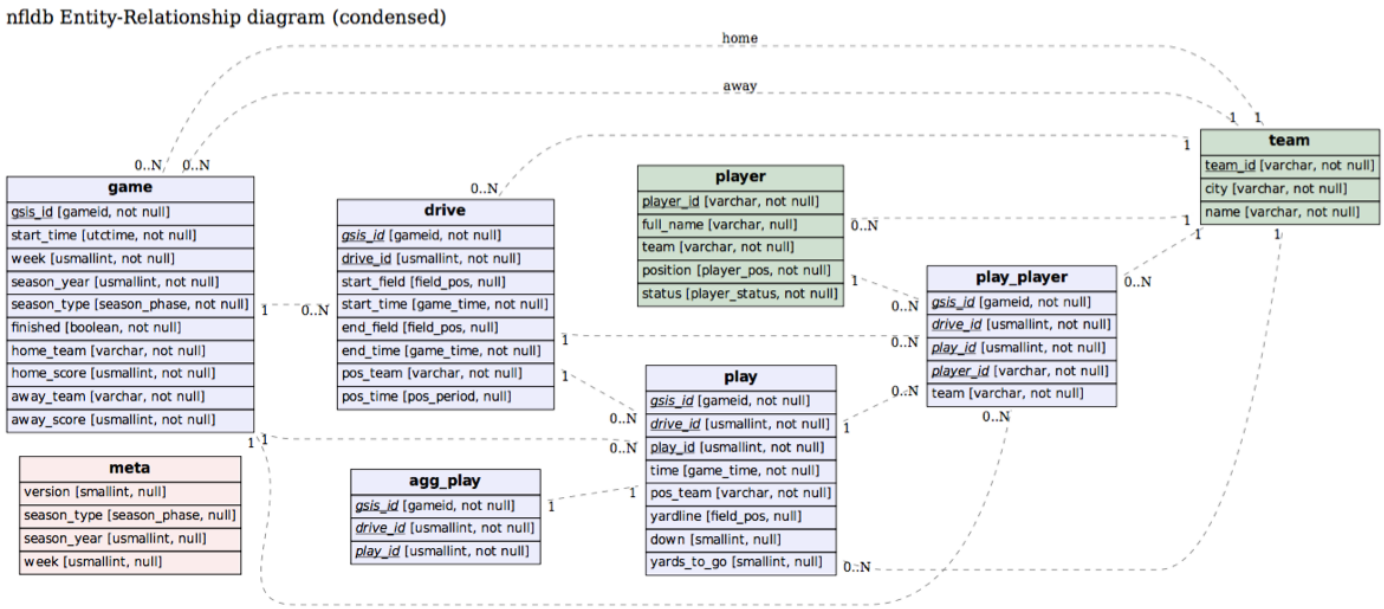
the NFL's Passer Rating statistic, and measured which one had a stronger relationship with the given team winning.

Data

The data I used for my analysis comes from Andrew Gallant's nfldb project (https://github.com/BurntSushi/nfldb). Nfldb reads NFL.com's (the official league website) JSON feed, parses, and stores the data in a relational Postgresql database.

A condensed view of the Postgresql data model is shown in the below nfldb Entity-Relationship Diagram **Figure 1.**

Figure 1: nfldb Data Model



The key tables for the purposes of this analysis are the **game**, **play**, and **play_player** tables. **game** contained contest information such as which year the game took place, who the participants were, and the score of the participants (particularly important for deriving the outcome independent variable). **play** included each situation, such as down & distance and time

remaining. **play_player** included player performance on each play. This was used to derive the score for each play, something that was not available "out of the box" from the database. Instead of having the score available for each play, I needed to check all associated player stats, determine if there were any scoring events, and if so add the score. This also needed to be done sequentially in order to obtain the cumulative score for any given play.

My variable set used for model fitting was as follows:

Offense Win (Outcome Variable): Indicated whether the team that possesses the ball (the offense) wound up winning or losing the game, with 1 for a win and 0 for a loss.

Offense Score: The cumulative point total for the possessing/offensive team at that play.

Defense Score: The cumulative point total for the non-possessing/defensive team at that play.

Down: 1, 2, 3, 4, or 0 for "non down" plays (such as kickoffs). If the requisite number of yards aren't gained by 4^{th} down, ball possession changes hands.

Yards to go: For down plays, how many yards are needed to either gain a first down or score.

Game Seconds: How many seconds have elapsed from the game clock. Each non-overtime NFL game has 3,600 game seconds, thus 0 seconds would indicate the beginning of the game, and 3,600 would indicate the end of the game.

Field Position: Where the ball was on the 100 yard long football field. -50 indicates a team on its own goal line, 0 indicates the ball at the 50 yard line (mid-field), and 50 indicates the ball close to the opponents goal, thus the possessing team is close to a score.

Initially, instead of Offense and Defense Score, I used Home and Away Score. These were used in conjunction with a third variable, Offensive Possession, which served as an interpreter variable of sorts to tell the model whether Home or Away score pertained to the offensive team, since the outcome variable, Offensive Win, was measured in terms of the

offensive team. I changed to Offense and Defense score to eliminate this extraneous variable and simplify the model, and also make the data work better for logistic regression.

While some of the mentioned variables were readily available in the nfldb database (such as time), some needed to be derived (such as score). Further, these variable did not all live in the same table, and instead needed to be joined and in some cases aggregated from several tables.

Along with the database, nfldb includes a python package that serves as an interface to the data model, but I decided it would be easier to query the database directly via SQL. My process for getting the data from the format in **Figure 1** to the previously discussed variable set was a combination of SQL queries in R (using the *RPostgreSQL* library) and *dplyr* package data manipulations. I wrote a series of functions to do these tasks, and a script that called these functions to build the R dataframe used for training.

As far as game selection, I used games for the 2009-2016 NFL seasons (the complete seasons offered by nfldb). I filtered out preseason games (where teams usually do not care if they win, since the game does not count), tie games, and games that went to overtime.

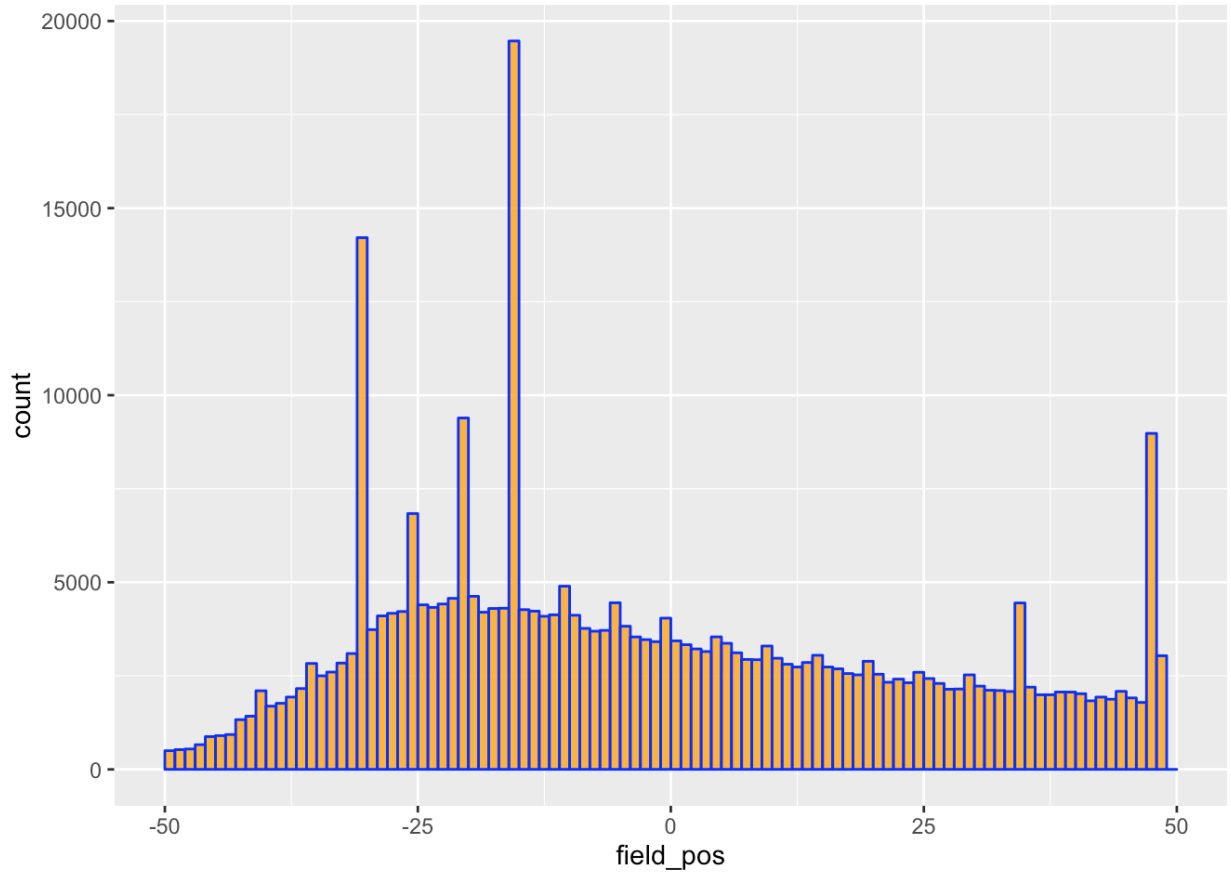
Distribution Checking

Distribution checking was performed in order to both check that data had been pulled accurately and to become aware of any interesting aspects that may be lurking.

A look at the field position distribution in **Figure 2** showed a right skewed unimodal distribution, with several field positions showing large spikes. At first this appeared alarming, until I realized that the field positions spiked at yard markers associated with rule based starting points. For instance, a touch back would result in ball placement at the 20 yardline (or -30 on my scale) according to the rules until 2016, when the ball was placed on the 25. We see a spike at

both spots, albeit a smaller one for the 25 yard line since the rule has been in place for only 1 year.

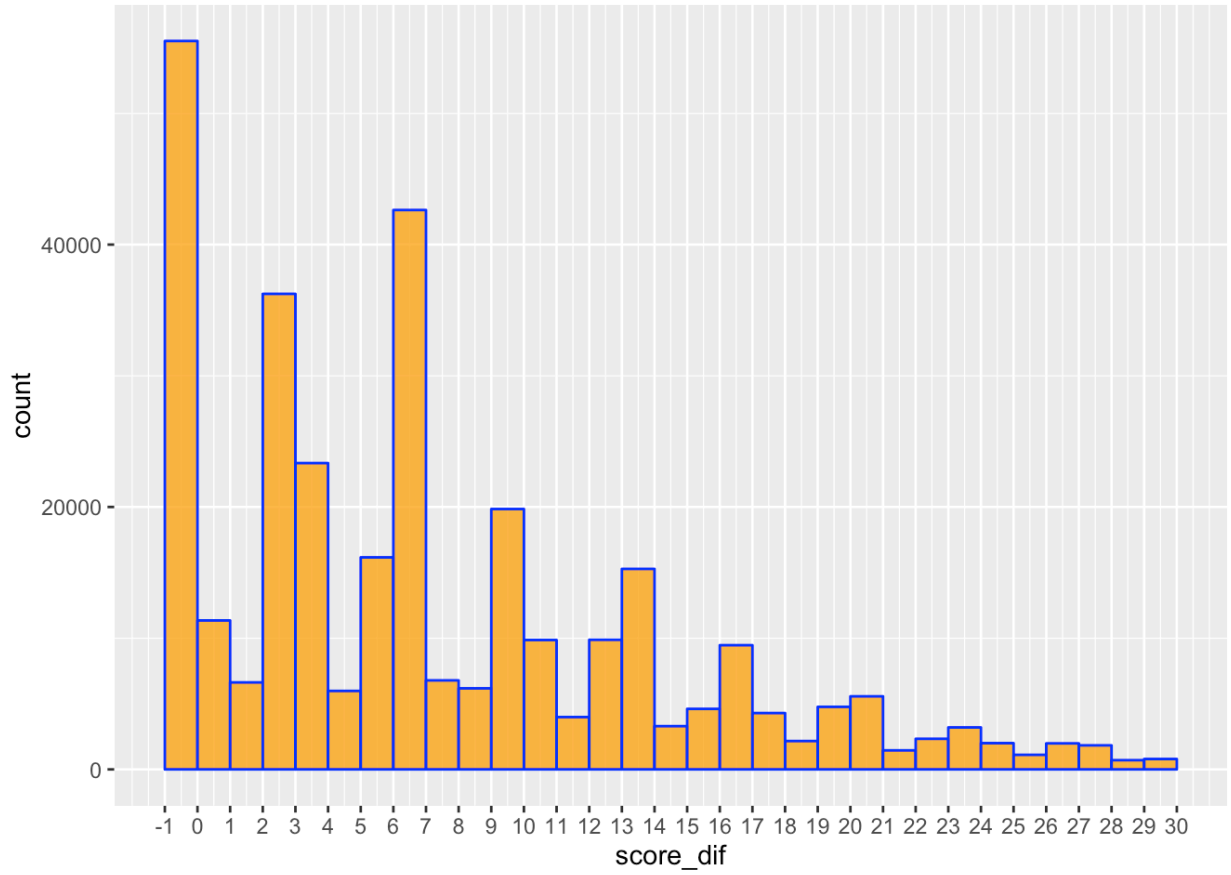
Figure 2: Data Field Position Distribution



A similar look at the score difference distribution (measured as the absolute value of the difference between the offense's score and defense's score) showed expected results. The most frequent difference was at 0, the score differential for the start of the game until the first team scored, as well as time games. There were also spikes at the point values of the most commonly occurring scoring plays, 3 and 7 (6 points for the touchdown and 1 point for the highly probable extra point kick following). The next three most common score differentials occurred at 4, 6, and 10, which would result from a combination of multiple 3 and 7 point scores by either team. Of

course, almost any other score difference to an upper limit was possible, but it was nonetheless reassuring to see this expected distribution.

Figure 3: Data Score Differential Distribution



A final tripping point worth exploring was the outcome variable (offense win for this study, which was 1 for true, and 0 for false). If the outcome was strongly imbalanced, then a model could become "lazy" and simply predict the most frequent outcome for each play. Intuitively, this would not likely be an issue since teams possess the ball for a relatively comparable amount of time due to the rules of the game. A team that performs well enough to score ends up giving the ball to the other team, allowing them to possess the ball for an amount of plays that may equal or even exceed the scoring teams plays, despite the fact they are losing. Nonetheless, it was worth a check. As expected, the offensive team won the game 51.7% percent

of the time, both standing up to intuitive reason and guarding against any potentially lazy models.

Model Building

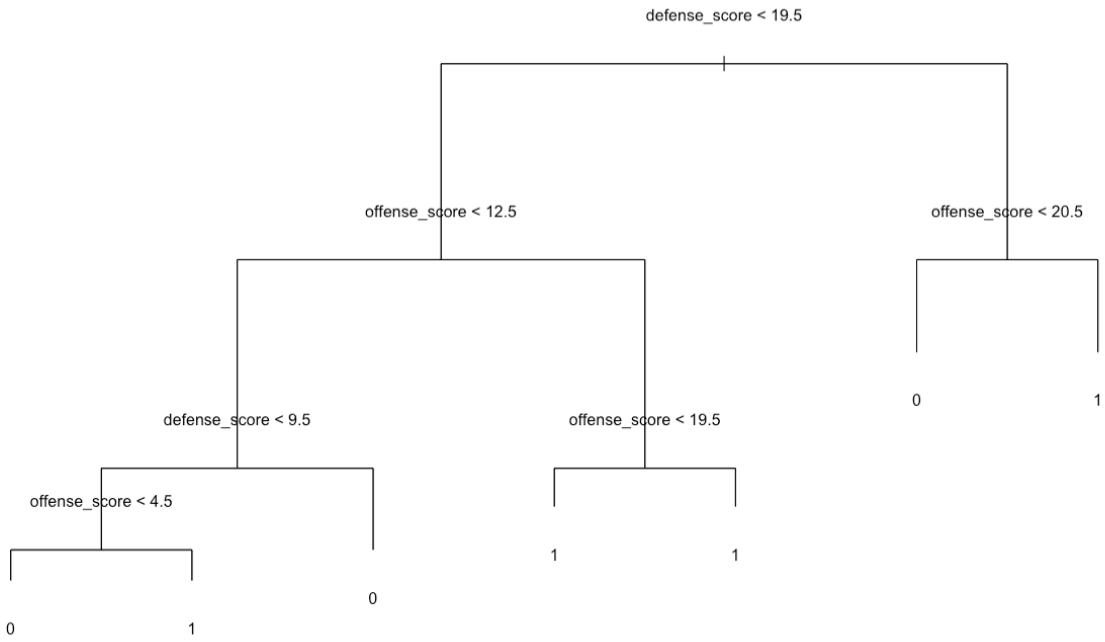
I followed Lock and Nettleton's approach to Train-Test splitting by holding out a single season for a test set (2016), and using the rest (2009-2015) for training data.

Before building the actual Win Probability models, which required regression due to the fact that I wanted a continuous value between 0 and 1, I built some classification models. The justification for doing so was to ensure that the win probability could beat a baseline model. The baseline I chose was to simply always predict the team that had a higher score at a given point to win. If the game was tied, then the baseline would flip a coin and randomly pick a team. This yielded a 74% classification accuracy rate, or a 26% error rate. This was the standard that classification models would need to beat, and any value would be the difference between these two rates.

In order to build intuition, I began with a decision tree. The reason for this was that in my own experience of observing football, my own mental process worked something like a decision tree. For instance, "Predict the higher scoring team to win, unless the score differential is low and the lower scoring team has field position that makes them very likely to score…" etc.

It turned that fitting a decision tree with recursive binary partitioning did not work as complexly as that, and simply used the score for each cutpoint **Figure 4.**

Figure 4: Simple Decision Tree



While underwhelming, this point was to be expected, due to the tradeoff of interpretability for simplicity (and subsequent predictive power).

The natural next step was to turn to a Random Forest/Bagging approach. Random Forest and Bagging both work by training large decision trees repeatedly through the bootstrapping technique. Bootstrapping is advantageous in this scenario because it reduces the variance of the model score by averaging the variance by the number of trees trained. Therefore, by training a large number of trees, we can reduce the variance.

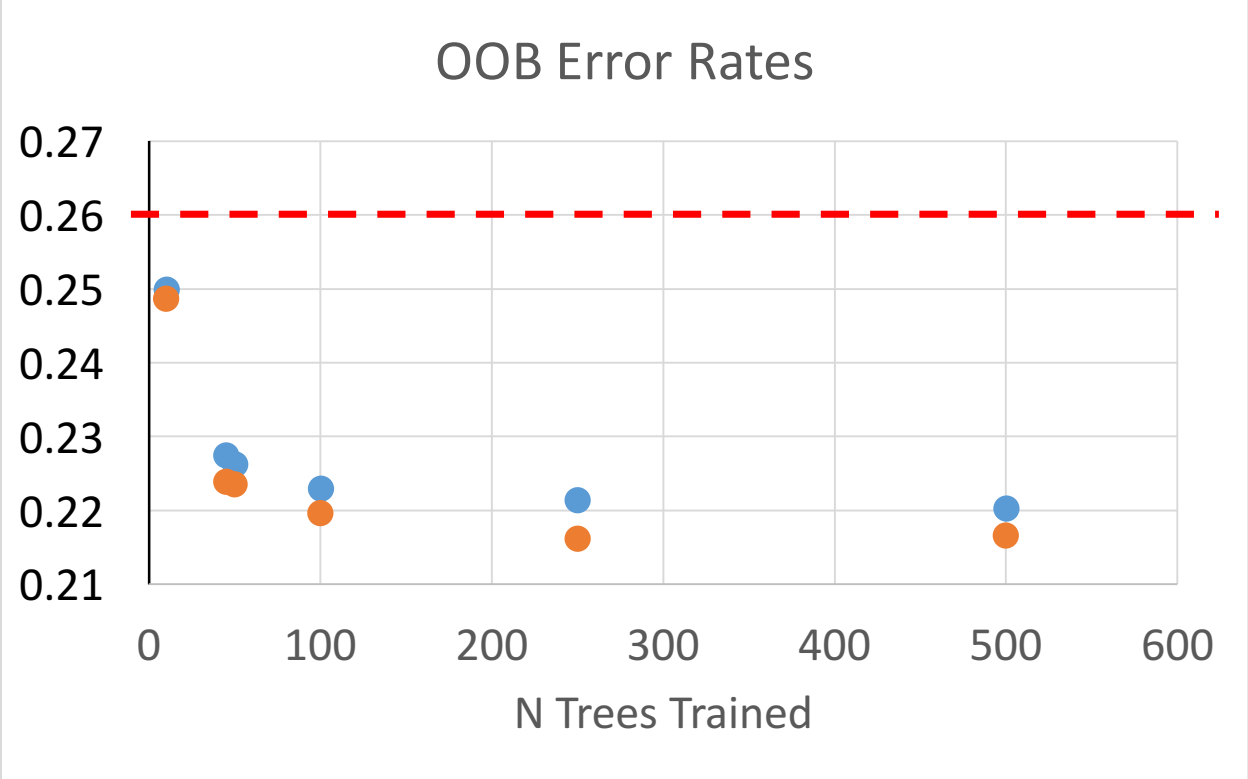
Bagging differs from random forest only in that bagging has access to all the available predictors at each cut point, while a random forest restricts itself to a given number of randomly selected predictors. The reason for doing this is to eliminate covariance between the trees, since often times bagging will rely on the same set of most powerful predictors.

For my classification check models using bagging and random forests, I used the *randomForest* library in R. For p predictors = 6, Random Forest models set the available predictors to p/3, or 2. For bagging, all p predictors were available at each node.

Due to the combination of size of my training set (n=285,050 plays) and limited computational performance availability, I was forced to limit the amount of trees that I trained to the recommended default (500) as an upper limit. Additionally, I ran each random forest and bagging model for 10, 45, 50, and 100 trees.

The performance metric I used for these models was Out of Bag error rate estimate, or OOB. This metric measures the tree at each given time on samples that were not chosen in the bootstrap. In **Figure 5**, a graph of the OOB rate for the Random Forest method (in blue), the bagging method (in orange), and the baseline predictor (red line) are plotted.

Figure 5



Both bagging and random forests perform better than the baseline model after only 10 trees, and both continue to improve until about a 22% OOB error rate. Furthermore, bagging performs better than the random forest model at each step, although only marginally and the two models tend to be more similar than different. In either case, this justifies the improvement these learning algorithms offer over simply predicting the winning team to win and flipping a coin the rest of the time.

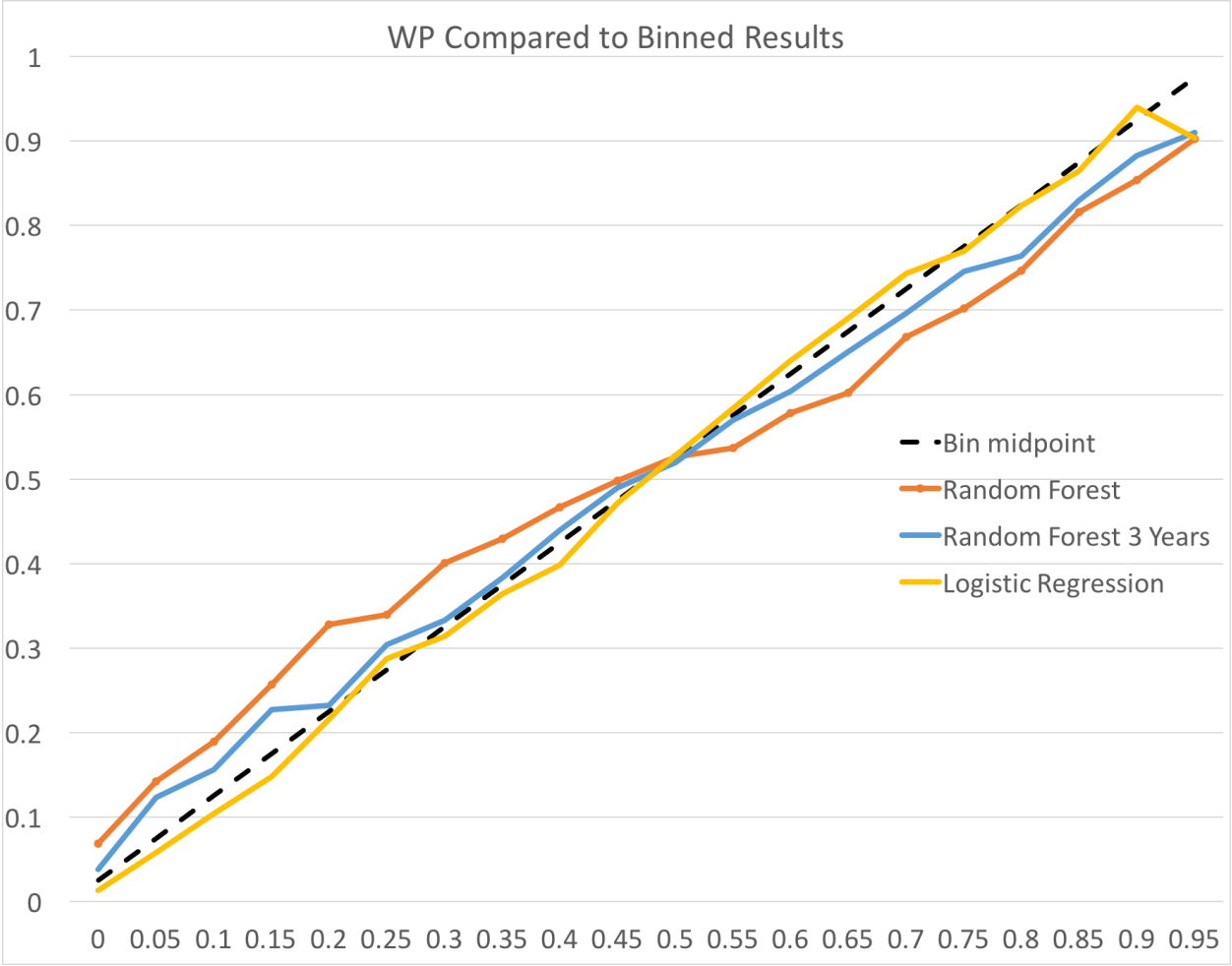
While these models have value, they are classification models that simply predict 1 or 0. For win probability, we need to fit regression models. To whatever degree classification trees are computationally expensive, my experience was that regression trees were much more so, limiting the amount of models I'd be able to fit. I also needed to trim the data in order to make it work. I ended up training a 50 tree random forest model on 1 and 3 years of data. I also fit logistic regression on the 3 year training data.

Evaluating these models is considerably more difficult, as the outcome variable they are trained against remains 1 or 0, discrete, definite outcomes. However, by definition, Win Probability is continuous and measures the uncertainty associated with a given situation. A 60% win probability, while predicting the given team to win, would also expect to be wrong 40% of the time. But this statement leads to a method that can be used for evaluation.

If a 60% win probability expects to be wrong 40% of the time, why not measure exactly that? I could take all the plays with a 60% win probability, and measure what rate the actual outcome was. But due to the continuous nature of probabilities, there are technically 0 that have *precisely* a 60% WP.

To overcome this, I binned each model's predicted WP range in 5% increments, and plotted them against actual results, seen below in **Figure 6.**

Figure 6



All 3 models performed relatively consistently with the actual outcomes of the plays they predicted. The random forest models had an issue where they over estimated low win probabilities and under estimated higher win probabilities than actual outcomes. The example trained with more data clearly performed better, leaving one to wonder if random forest would have performed better had computational power allowed for using more data and training more trees.

In the current case, the most accurate model turned out to be one trained using logistic regression, from both a visual and MSE standpoint, despite having an issue assigning slightly

lower WPs to teams that were nearly certain to win (though not underperforming the random forest counterparts in this regard). Below, in **Figure 7**, are the coefficients for the logistic regression model output.

Figure 7

```
Call:
glm(formula = offense_win_regression, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.5514  -0.8101   0.1357   0.8338   3.1685

Coefficients:
               Estimate Std. Error  z value Pr(>|z|)
(Intercept)    2.063e-01  1.362e-02   15.149  < 2e-16 ***
down          -3.255e-02  4.228e-03   -7.700 1.36e-14 ***
yards_to_go    1.027e-02  1.052e-03    9.758  < 2e-16 ***
game_secs      2.985e-05  7.974e-06    3.743 0.000182 ***
field_pos      8.655e-03  1.920e-04   45.073  < 2e-16 ***
offense_score  1.740e-01  9.236e-04  188.435  < 2e-16 ***
defense_score -1.787e-01  8.963e-04 -199.341  < 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: 394808  on 285049  degrees of freedom
Residual deviance: 276622  on 285043  degrees of freedom
AIC: 276636

Number of Fisher Scoring iterations: 5
```

All 6 predictor coefficients and the intercept were highly significant using a 0.001 threshold. This was encouraging that the intuitively chose game state variables were indeed variables that impacted WP. On the other hand, I may have been too restrictive and been able to improve my model even further by attempting to use more variables.

A side benefit of logistic regression being the selected model is that it is more readily interpretable by coefficient analysis. The predictors with strongest effects were offense_score and defense_score, as expected. Additionally, the coefficients were nearly equal, which resonates in the sense that at the end of the game, points are in fact worth the same.

There is however, a small difference, which suggests that there are situations where the offensive team, despite winning or losing, may have the opposite expected win probability based on the other predictors. And the signs of all these other predictors make sense in the same way. For instance, as the offense gains better field position, they are more likely to score points. This is validated by the positive sign on the field_pos coefficient.

Win Probability Added

Beyond serving as a tool for evaluating individual game situations, Win Probability provides value via a derivative metric, Win Probability Added (WPA). Simply, WPA is the difference in win probability between two plays, and the sought after contextual play performance evaluator.

Calculating WPA is relatively simple, but there were still two major concerns. Since my WP is estimated in terms of the offensive team, the WP flips on plays that result in a change of possession. Also, the plays needed to be ordered sequentially in order for WP to make sense.

To evaluate WPA, I returned to the original problem that was the cause for this research: traditional NFL statistics. There exists a plethora of stats that WPA could be measured against, but one particularly interesting metric was the notorious NFL Passer Rating.

The NFL Passer Rating was an early attempt to address the issue of simple yards being inadequate to measure passer (i.e., the QB and most important player) performance. It used a fixed, and some might say arbitrary, scale for essentially weighing passer statistics. The formula is available at https://en.wikipedia.org/wiki/Passer_rating#NFL and is given below in **Figure 8.**

Figure 8

$$a = \left(\frac{\text{COMP}}{\text{ATT}} - .3\right) \times 5$$

$$b = \left(\frac{\text{YDS}}{\text{ATT}} - 3\right) \times .25$$

$$c = \left(\frac{\text{TD}}{\text{ATT}}\right) \times 20$$

$$d = 2.375 - \left(\frac{\text{INT}}{\text{ATT}} \times 25\right)$$

where

ATT = Number of passing attempts
COMP = Number of completions
YDS = Passing yards
TD = Touchdown passes
INT = Interceptions

If the result of any calculation is greater than 2.375, it is set to 2.375. If the result is a negative number, it is set to zero.

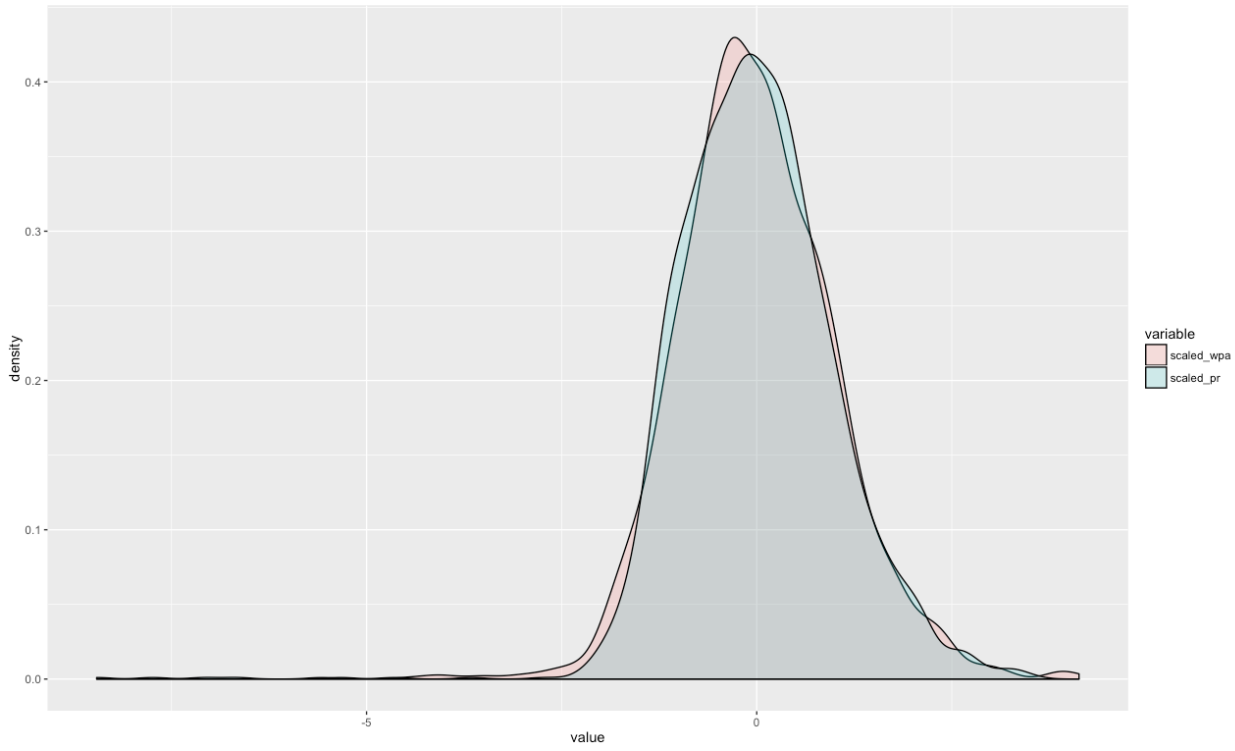
Then, the above calculations are used to complete the passer rating:

$$\text{Passer Rating} = \left(\frac{(a+b+c+d)}{6}\right) \times 100$$

To compare WPA to passer rating, I took all games from the 2016 season and rated each game's passer according to the above formula (games were limited to the 2016 season due to nfldb player table limitations). Next, I totaled WPA for the same plays that were used to provide stats for passer rating. Not all plays in a game contribute to a QB's passer rating, such as defensive plays, running plays, or special teams plays, so by limiting WPA totals to those plays I could be sure than each metric was measuring the same sample. I wanted to compare how strongly each was associated with a win.

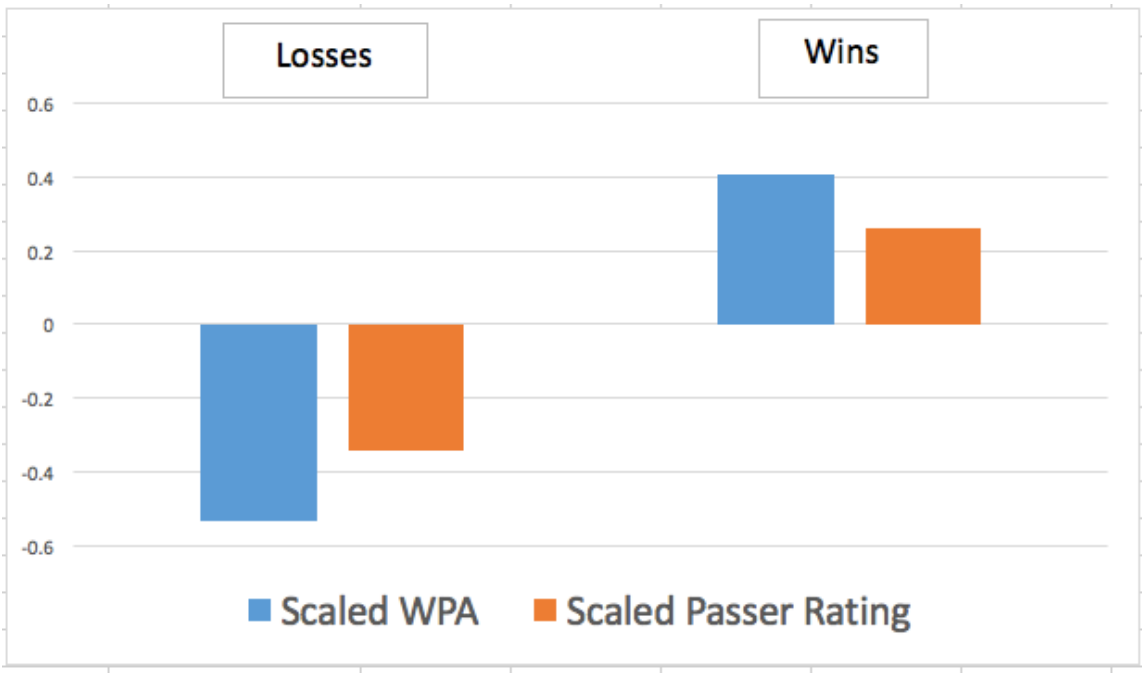
This presented a problem in that WPA and Passer Rating are on completely different scales. Total WPA ranged from -2.5 to 1.6, while Passer Rating ranges from 0 to 158.3. To deal with this, I scaled and centered both sets of values, resulting in the below density plots in **Figure 9.**

Figure 9



With the data scaled and centered, they could now be compared. To do this, I measured the average scaled metric for wins and for losses, resulting in the below table in **Figure 10.**

Figure 10



For both Losses and Wins, the scaled value of WPA was much higher than the Scaled Passer Rating. This indicates that WPA does a better job assessing whether a passer contributes to his team's success.

Result

The result of my research is confirmation of the idea that Win Probability is theoretically an easy to fit model that produces accurate results. I achieved accurate, predictive results using a simple logistic regression model with 6 predictor variables. Though my method differed from some of the other, previous research, I achieved similar results. Further, I was able to show Win Probability Added is a metric that can outperform the NFL Passer Rating on a per pay basis as it relates to wins, which are the ultimate goal of any NFL team and player.

Discussion

Even though my model produced satisfactory and usable results, it could possibly be improved several ways. One method would be simply to obtain a more powerful machine to run larger random forest models with more data. A distributed cloud computing solution could be suited to such a task.

In any event, the availability and predictive power of Win Probability is not in question, despite the 2016 Season Super Bowl between the Patriots and Falcons (in which the Patriots came back from a low win probability to win the game). The next milestone for WP/WPA is to take a bigger place amongst popular football metrics. Watch any NFL telecast, and most statistics displayed or repeated are the same metrics that have been used for years. The NFL is getting a little better in this regard, but there is still further to go. WPA could be key metric to elevate the sophistication and quality of general NFL discussion.

That said, neither WP nor WPA is a silver bullet solution. WPA measures the result of the play in terms of the team, not individual players. As an example, a QB could make a good throw down the field, but have the ball bounce off the hands of the receiver, leading to an interception. This would result in a negative WPA on that play for the QB, despite no fault of his own. Additionally, for offensive linemen WPA would be less specific to their performance, and lack value.

Nevertheless, both Win Probability and Win Probability Added have value. They are accurate, interpretable, and are metrics that properly align with the goals of their team, something that is rare for any sort of performance metric.