How to Beat Vegas at Spread-Setting

Project by: Jake Hill, Christopher Kreke, Christopher Steeves, and Matthew Zlotnik

Background/Motivation

Sports betting has existed in the United States since the 1600s, when patrons of races would place bets on horses. Since its early days, sports betting has evolved to include nearly every sport (and other competition) imaginable, and for a while was largely shunned from public eye due to several large scandals involving athletes throwing games on purpose, funded by wealthy bettors with large stakes in the game. Since its shaming in the early 1900s, sports betting has slowly worked its way back into the public eye. Sports betting was legal only in Las Vegas until 2011, when New Jersey passed a law allowing betting at the state's racetracks and Atlantic City casinos. New Jersey then lost in two consecutive attempts to overturn the Professional and Amateur Sports Protection Act (PASPA) in district courts before finally convincing the United States Supreme Court to hear its plea. In the most impactful ruling in sports betting history, the Supreme Court overturned the decision of the lower courts, repealed PASPA, and gave full authority to state governments to legalize sports betting[1]. Since this landmark case, many states, including Delaware, New Mexico, Pennsylvania, and Arkansas have legalized sports betting. Investors project that the sports betting industry alone is valued at nearly $75bn[2].

Legal sports betting involves a patron placing a wager on one or more outcomes during a contest, including but not limited to wagering on who will win, how many points will be scored, or how individual competitors will perform. The exact details and the return of this wager used to be determined by skilled oddsmakers who would gather all available information leading up to a competition. Now, complex algorithms and models crunch hundreds of millions of possible inputs to determined exactly what the terms of the wager will be. For this project, we will dive into exclusively betting on the NFL.

To fully dive into sports betting, several key terms must be introduced. Firstly, on any wager, a bettor will received "odds", which are defined as the return of the wager should the bettor correctly predict the outcome. There are two possible ways to denote the odds of a wager, but we will only use the American Odds system. American Odds can either be positive or negative. Negative odds (for example, -110) denote the amount of money required to wager to win $100 from a bet. In this case, the bettor would have to place a $110 bet to receive a return of $100. Positive odds (such as +110) indicate the return from winning a bet placed at $100 (in this case, the bettor would win $110 in addition to receiving their $100 wager back).

The most simple wager a bettor can place is called a "Money Line" wager. In this wager, a bettor will simply choose a team to win the game and receive odds based on how evenly the teams are matched. Betting money line on a team that is heavily favored to win typically yields large negative odds, while betting money line on a heavy underdog typically yields large positive odds. The most commonly placed bet is called "The Spread." The spread is the amount by which the favorite is expected to win the game, as determined by the oddsmaker's algorithms. For example, betting on

[1] https://www.legalsportsreport.com/sports-betting/timeline/
[2] sportsbettingdime.com/guides/finance/global-sports-betting-market/

the Cowboys with a spread of Cowboys -5.5 means that the Cowboys must win the game by 6 or more points to win the wager. While several other types of wagers exist, the final wager we will discuss is the “Point Total”. A bet on the point total involves choosing whether the two teams combined will score over or under the number of points set by oddsmakers.

One more key concept in the realm of sports betting is the “edge”. The edge is the white rabbit that all professional sports bettors (or “sharps”) chase. Betting with an edge means betting with odds of winning that are higher than those determined by oddsmakers. It is common knowledge that “the house always wins” in regards to all types of gambling. In sports betting, the house maintains its advantage by setting odds in such a way that bettors must win, on average, 52.4% of bets to turn a profit. As such, the primary goal of this project is to create an algorithm powerful enough to achieve an edge of 3% or higher, but simple enough to be run between the time odds are released on Tuesday to the kickoff of the first NFL game of the week on Thursday.

Data Collection

The first step for us was finding a site that could reliably provide the spread, MoneyLine, and point totals for games for both past and future games in a way that time spent building a scraper would be minimized. We were fortunate enough to find sportsbookreview.com. It conveniently organizes all NFL games by date and used a uniform URL where the data was interchangeable. Using Selenium, we scraped all Vegas betting lines from 2017 to the current NFL week. An example of a few games is shown below. As you can see, it conveniently organizes the betting lines as well as the odds right next to each other in a data table.

Sunday, December 15, 2019

01:00 PM	305 **Tampa Bay**	55.00%	-4 -110
H2H	306 **Detroit**	45.00%	+4 -110
01:00 PM	307 **Philadelphia**	52.00%	-6 -110
H2H	308 **Washington**	48.00%	+6 -110

Arguably the most critical and difficult aspect of creating our NFL-predicting algorithm was finding data that could be used to predict the spreads of games. Not only did the data have to be clean and applicable, but somehow the data had to remain relevant and updated every week of the season. We primarily considered three sources for our data. First, we looked into Pro Football Reference, a site which shows player and team data on a weekly and seasonal time frame. Because of the prohibitively huge number and constant injuries/trades of players and over-simplistic nature of team statistics, we decided not to use Pro Football Reference. Next, we looked into the player rating system for the popular video game “Madden NFL 2020”. This game includes 1-100 ratings for every conceivable attribute for every NFL player. However, this data is not reliably updated weekly and as such would likely not provide an edge when applied to betting. Finally, we found Pro Football Focus (PFF), a website which contains a few characteristics which made it perfect for our objective.

Pro Football Focus contains 1-100 ratings for every player for every game. These ratings are created by people watching each individual play from every game and giving each player a

ARZ - Season Grades

KEY

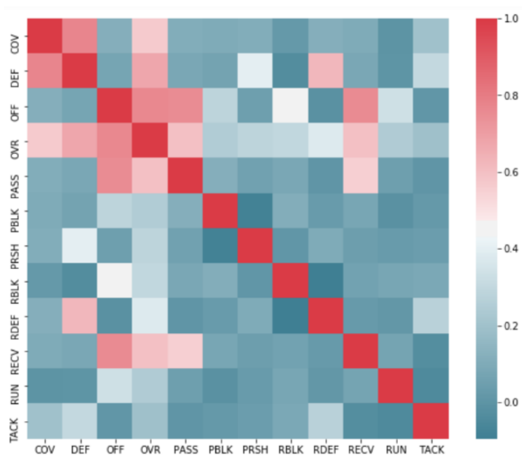
					RESULTS				OFFENSE						DEFENSE					SPECIAL	
WEEK	@	TEAM	DATE	TIME	W	PF	PA	OVER	OFF	PASS	PBLK	RECV	RUN	RBLK	DEF	RDEF	TACK	PRSH	COV	SPEC	
1		Detroit Lions	09/08	4:25pm	T	27	27	67.0	63.6	52.6	72.3	63.7	68.8	61.1	62.5	64.5	70.2	78.8	51.0	85.5	Game Reports
2	@	Baltimore Ravens	09/15	1:00pm	L	17	23	60.7	65.1	66.3	62.3	68.5	68.3	34.8	57.3	55.4	50.5	53.0	62.9	45.5	Game Reports
3		Carolina Panthers	09/22	4:05pm	L	20	38	59.6	59.2	56.4	67.5	55.5	83.2	45.1	62.6	70.2	70.1	65.6	53.7	49.9	Game Reports
4		Seattle Seahawks	09/29	4:05pm	L	10	27	62.0	61.6	59.3	64.3	60.3	76.1	46.1	64.7	47.9	30.0	79.1	69.0	67.1	Game Reports
5	@	Cincinnati Bengals	10/06	1:00pm	W	26	23	75.5	79.3	82.7	85.1	67.8	78.2	67.1	67.4	68.7	61.8	64.7	64.6	39.1	Game Reports

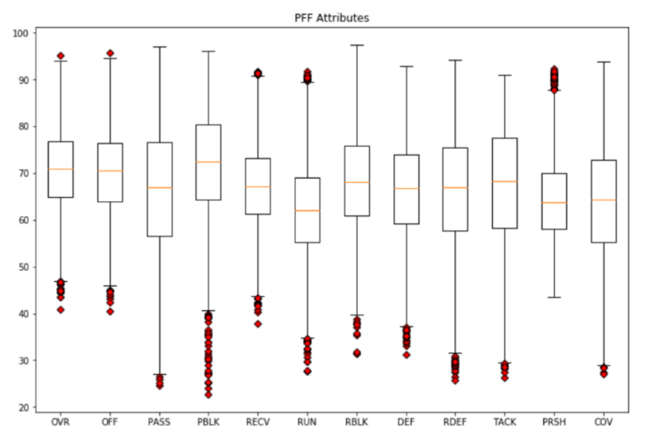
score from -2 to +2 based on individual performance in the context of the play. PFF releases their weekly grades on Tuesday morning, meaning that the data could be re-scraped weekly and remain up-to-date. Another unique characteristic of PFF is that they aggregate player grades for every game into positional “unit” grades. A sample for positional unit grades over a team’s first five games of the 2019-2020 season are shown below. These positional unit grades proved perfect for our model because positional groups remain consistent week-to-week, and they offer more insight into a team’s performance than simply the countable statistics that could be obtained from Pro Football Reference. We decided that in order to accurately gain an edge over Vegas, we would have to try and predict NFL spreads in a way that nobody had likely ever tried before. Ultimately, we decided that predicting PFF scores for each week and using these predicted scores as a proxy to predict the spreads of games could give us the edge we needed.

To obtain the data from PFF, we had to build a scraper that utilized Selenium’s ability to interact with websites, including pressing “show more” and scrolling as it scrapes data. Because PFF stores every team’s game data from every year on a separate webpage, our scraper had to manually input URL’s for each combination of team and year that we needed. Thankfully, Pro Football Focus uses a standardized URL format so embedded for loops with lists of every team name and every year were sufficient to allow Selenium to scrape all the required data to fit our model. Using this process, we scraped the game-by-game grades for each of the 32 NFL teams for every year since 2015. Even though PFF is a website designed by data scientists, the data obtained by scraping the website still required a fair amount of cleaning.

Data Pre-Processing and EDA

Upon scraping the data, we noticed that PFF marks any games which have not yet been played with “-“ or “Na” in all fields. This allowed us a convenient way to split out our holdout set from the rest of the data. Once we split the data, we filled all numeric fields in our holdout set with zeros. Also, all playoff games were denoted in the week column as “WC, DP, CC, SB” for Wild Card, Divisional Playoff, Conference Championship, Super Bowl. To ensure that our models could process a team’s season in order, we replaced any instances of these letters with numbers corresponding to that week in the season. Next, we ordered all of the rows in our remaining data by team, season, and week so that we could create a time series and easily create lag variables which could help aid our predictions. Additionally, we elected to drop the “Overall”, “Offense”, and “Defense” grades for each team, as we assumed these would be highly colinear with the various specific grades within these categories.





Next, we looked into the distributions of each of the PFF attributes for each team over our time frame to see if any specific trends appeared. Of note, the distributions for most of the statistics were very similar, with means in the upper-sixties, lower-seventies area with large IQR and many outliers below the distribution. Ultimately, we realized that in order to predict these possible outlier games, we would have to keep these outliers in the data and hope that there was a specific trend to when these outliers occurred.

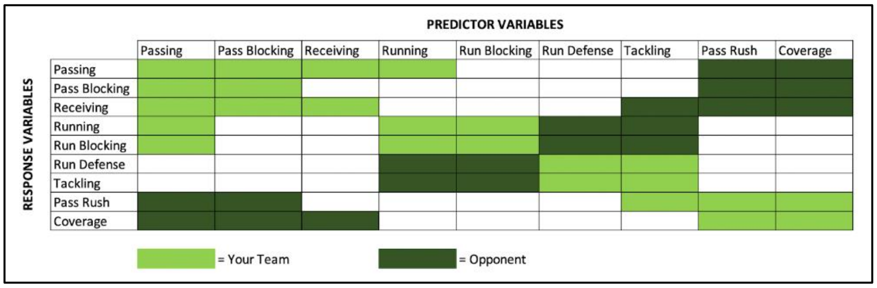
Feature Selection and Engineering

Before we dove too deeply into exactly which features needed to be included, what new features had to be created, or what the best way to predict PFF scores was, we wanted to ensure that accurately predicting PFF data would allow us to gain an edge on Vegas. As such, we created a dataset consisting of only games that had already occurred that we could test our theory on. We then fit a simple random forest regression model that took inputs of both team's scores for each of the past sixteen weeks as well as their scores from the game which we were trying to predict and output our predicted final score for the game. From this predicted final score, we took the difference in scores and called that our predicted spread.

We then compared our spread to the official spread of the game obtained from sportsbookreview. We decided to test the results if we had theoretically placed wagers according to the output of our model. As such, we compared our model's predicted spreads to the official spread of the game and bet whichever side of the official spread that our predicted spread fell. For example, if Houston and Dallas had an official spread of Houston -6.5 and our model predicted Houston -8.5, then our model had more confidence in Houston than Vegas did, and we would "bet" on Houston -6.5. Alternatively, if our model had predicted Houston -2.5, then we would "bet" on Dallas +6.5 as our model had more confidence in Dallas than did Vegas. We will discuss more about specific betting rules when we describe our final model. We were thrilled to find out that, depending on the exact betting rule being utilized, our model was able to predict between 65-80% of games correctly. We were well aware that this extreme winning percentage would be unattainable without using the PFF grades to predict the outcome of the same game, but it gave validity to the strategy of trying to accurately predict the PFF scores. With the newfound confidence in the predictive power of PFF scores, we decided that we did not need to gather any extra data so that our data pipeline could remain streamlined.

Next, we created sixteen weeks of lag variables for each game and appended them to our data frame, increasing the number of columns from 27 to 232. Now for every game we had a thorough representation of a team's performance from each of the previous sixteen games. We then merged this data frame with itself based on the index of game number, so that we then had two copies of each game in our table, one representing each team's point of view. This new data table had 447 columns. Knowing that it would be nearly impossible to build an effective model that could use 447 different input variables and only 160 different games from which to pull data, we decided to apply our own knowledge of football to decide which predictors could be predictive of which PFF grades. The table below shows how we determined certain scores interacted with a team's own as well as their opponent's past scores. We decided that, given the extreme depth of data we had, we would employ all sixteen lags of a team's own data, but only the average of the opponent's last three games to predict the scores.

To interpret which variables we used to predict a team's passing grade, we used each of the team's own grades from the last sixteen games of passing, pass blocking, receiving and running as well as the opponent's average of their last three games' scores in pass rushing and coverage. In this scenario, we decided that neither a team's own run defense nor an opponent's run defense has any predictive power on the effectiveness of a team's own passing score. With our predictors now set, we moved on to the model-building phase of our project.



Model Building

From the beginning of our modeling phase, we knew that every iteration would be extremely computationally expensive. We knew that the power of our potential model stood in being able to accurately predict each PFF score extremely well. To drive that point home, we decided to fit a separate model to each PFF statistic for each team, for a total of 288 different models. After finding many different models that we believed could be effective, we decided to run a simple script with no cross validation of models to test the untrained average predictive accuracies of each of our possible models. The results below show the negative mean squared error of each of the models we tried against the actual PFF grades they were predicting. After considering these results as well as the implications of running each of the models, we settled on taking three different types of models further: Gradient Boosting Regression, Bagging, and Ensembling GBR + Bagging.

	score
Ada Boost	-73.052030
Gradient Boosting	-76.677400
Bagging	-77.407197
Random Forest	-84.719755
Extra Trees	-87.760120
Ridge	-93.853058
Linear Regression	-96.472557
Lasso	-147.328444
Decision Tree	-155.112762

```
if attribute == 'RBLK':
    for i in range(1, 17):
        columns.append('RBLK_lag' + str(i))
        columns.append('PASS_lag' + str(i))
        columns.append('RUN_lag' + str(i))
    columns.append('TACK_avg3_opposing')
    columns.append('RDEF_avg3_opposing')
```

```
# N
gri
gri

pri

gbr

)
```

```
, scoring='neg_mean_squared_error', n_jobs=-1, verbose=3)

,
tors'],
_rate']
```

Because we could only scrape betting lines from 2017 onwards, we decided to use the entirety of the dataset pre-2017 to train our models, and 2017/2018 data as our test set. We then set out to cross-validate 288 different models for two different types of models for upwards of 11 different options for each of five different parameters. These models were fit using some of the snippets below. For each model being fit, the program would check which attribute was trying to be predicted, and would append the appropriate predictors into a list of column names. Then the model would fit a gradient boosting regressor and random forest model grid search to find the best possible model parameters.

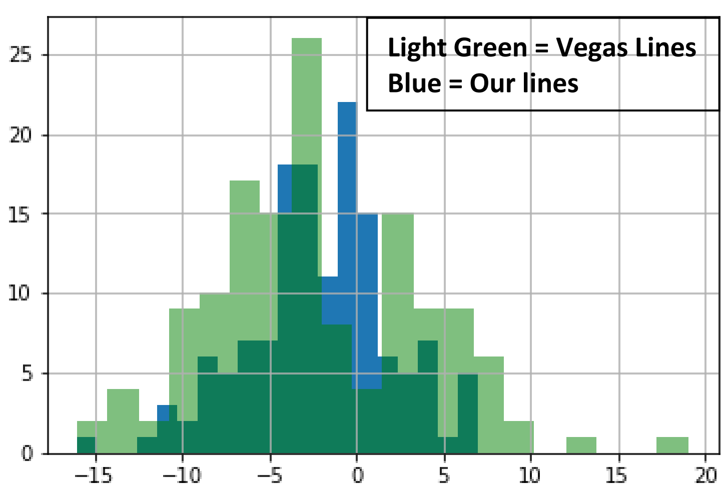
These grid searches took several hours to run, but were ultimately highly effective at creating 288 individual models, each of which was specialized for a specific statistic for a specific team. Ultimately, our gradient boosting regressor was deemed the most effective of the three modeling choices, as the bagging was not as powerful and the ensembling of the two methods added little predictive power. The final outputs of these models were predictions of PFF grades for every team for every grade for the 2017 and 2018 seasons. Shown below are a small chunk of the predicted data as well as the variances of both the actual data and the predicted data from our models.

Next, we had to fit a model which could predict the spread for a game given the two teams' predicted PFF grades for that week. Because our models were not able to reduce the variance of the data, we knew that simply using this week's predicted PFF grades to predict the spread would not be an effective way to generate a betting model.

	PASS	Team	GameNum	PBLK	RECV	RUN	RBLK	RDEF	TACK	PRSH	COV
0	63.38676	Arizona Cardinals	185	63.85601	66.63105	57.815	77.53052	65.75775	60.7227	60.85624	68.96456
1	59.51567	Arizona Cardinals	186	65.56962	66.63105	58.962	70.36435	63.72692	66.81987	61.7762	66.5358
2	56.25564	Arizona Cardinals	187	63.72601	73.74705	57.823	66.53644	63.2065	60.7227	65.21091	62.49828
3	52.63088	Arizona Cardinals	188	63.72601	62.43853	58.984	62.3658	64.9405	70.8238	61.76174	63.90465
4	63.14845	Arizona Cardinals	189	64.28832	66.63105	58.949	69.57866	64.93517	65.72403	61.84979	64.46532
5	56.90291	Arizona Cardinals	190	63.72601	59.69364	57.473	69.71624	61.56125	64.42201	59.84795	63.63268

	Actual Variation	GBR	Bagging	Ensemble GBR+Bag
OVR	8.686815	8.946790	8.534510	8.508605
OFF	9.150033	9.290589	10.178938	9.450092
PASS	12.650637	13.488054	12.838122	12.508993
PBLK	12.553985	13.984841	12.193415	12.656218
RECV	7.268326	8.191492	7.368853	7.584415
RUN	10.061331	13.314773	11.617901	12.033162
RBLK	10.026115	12.542467	13.296712	12.622753
DEF	9.824371	13.312134	10.477246	11.669067
RDEF	12.738064	14.242311	14.013929	13.816338
TACK	13.483721	15.516653	14.757172	14.577531
PRSH	8.145606	8.609999	8.151528	8.153898
COV	12.077438	14.024670	12.990558	13.150194

As an alternative, we decided to import all of our predicted PFF scores for both teams for each of the past sixteen weeks. Then, we trained and fit another gradient boosting regressor with gridsearchcv using the home team of each game as a reference. The model was fit on a random sample of each team's set of predicted PFF scores to generate an expected points for and against the home team, the difference of which created an estimated spread, and the sum of which created an estimated point total. We then ran this process through 1000 simulations, each time selecting a different random combination of PFF scores from the past sixteen games. We then compared these 1000 generated spreads and point totals to the actual Vegas line, and if over 55% of the simulated predictions fell to one side of the Vegas line, then we would say that our model recommended betting that side. Because we often predicted the same spread as Vegas, there were many occasions in which this 55% threshold was not met, which, after consulting a few experts on the matter, we deemed a positive. According to the experts, no betting model should recommend a bet on every single game. Shown below is the distribution of our predicted spreads against those from Las Vegas. As you can see, our model is much more conservative when looking at teams with a high skill

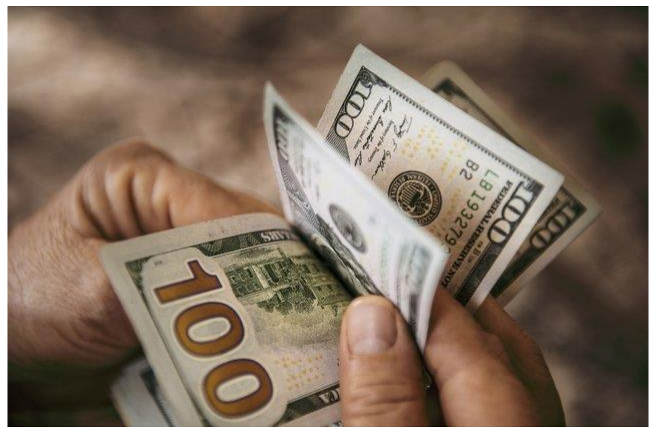


disparity, with many fewer lines occurring outside of a magnitude of ten. Another observation is that this distribution is not symmetrical. This is an encouraging sign, because it means that our model was able to pick up the fact that home field advantage exists in the NFL, so spreads are more likely to favor the home team (our point of reference in modeling).

Results

By far the most important aspect of this, or any, betting model: does it correctly predict games? As we mentioned above, we know that the theoretical maximum win rate we could expect from our PFF-based model was between 65-80%, depending on which betting rule we used. For this analysis, we did not dive into exactly which scenario to bet to achieve the highest success rate, but rather just measured the overall accuracy of the model on a holdout set. We also decided to just test our model's accuracy in betting against the spread. For our holdout set, we used 2019 games, games which had not been introduced to the data in any capacity. Our resulting win rate:

57%



From the very start, we knew the only figure that would ultimately matter was the edge we could create over Vegas. Using a holdout set of over 150 games, we achieved a 7% edge! We had to curb our own enthusiasm because it is common knowledge that a model has not proven itself as effective until it has a full season of predictions completed. Since we finished tuning the model a few weeks ago, we have continued to create predictions. Predicting for games that had not even happened, the model has accumulated a 18-12-2 (W-L-D) record over the past two weeks of NFL data, for a win rate of 56%. While we acknowledge that this validation comes from a sample size that is entirely too small, the results are nonetheless encouraging so far. We will continue to monitor the success of this model through the remainder of the 2019 NFL regular season.

Conclusion

We learned many lessons throughout this process, including time management, process delegation, and leaning on each other's strengths and weaknesses. Because of the intensity of this project in conjunction with the extreme workload of this program in general, we were forced to lean on one another and trust that each of us was capable of completing the process that was assigned to us. In training so many different models, we learned the value of computational efficiency and patience. While setting n_jobs to -1 in every model may have made our computers virtually useless during the model-training process, it allowed us to significantly chop time off of our training process.

Finally, even though we achieved an edge that we thought impossible before the project, there is still much more work that can be done on this model. We have a list of the six different betting scenarios, shown above, and we could run analyses on what level of confidence we should require to bet on each individual scenario. Additionally, our model does not account for weather or recent player injuries. In future iterations, we could add a field that denotes if a major injury has occurred in the past few weeks or if inclement weather is expected during the game. Because our model is already sitting at an incredible 7% edge, we do not expect that adding any extra predictors will likely have any improvement on the predictive accuracy.

Link to GitHub containing code: https://github.com/chris-steeves/Advanced-Predictive-Modeling-Project