Predicting NCAA FBS Success

Sports have a unique place in human history and modern society. Fans rabidly follow their teams. Over the last twenty-five years, an entire industry has sprouted around the identification of future talent for college sports, particularly football. High school recruits are identified, ranked, and tracked by sites such as 247sports, Rivals, and ESPN. Millions of dollars are spent by schools and fans before players even step on campus. But the money skyrockets once they reach campus. In 2015, Business Insider reported that college sports generated $9.15 billion in revenue. With so much money on the line, much like professional sports, it's not surprising that college teams are turning to data to help gain an edge in finding and developing talent.

Sports provide a unique opportunity for data science as substantial amounts of data are gathered form each game. But it does lead to the question, with all the data collected, what statistics and features are essential? What kind of insights can the data bring? Can you predict future success with reasonable certainty? In this project, I explore the data looking at armchair theories and applying machine learning to identify the most critical statistical features to predict if a team will win a game. Like any research study, the story begins by obtaining the data.

Data Acquisition

One would think that in the modern world frenzied with interest in sports, that gaining access to sports statistics would be easy and, in some ways, you would be correct. Indeed, you can find statistics on the TV, newspapers, magazines, and numerous websites. Unfortunately, these sources typically provide a limited number of statistics for a single game rather than all available data for all games. When you try and find and access large-scale data sources that aggregate all of the statistics in databases and APIs the problems continue. Although a few databases and APIs exist for college football statistics, gaining access is difficult. Typically, the data sources are geared towards companies and individuals with deep pockets. Luckily, there is one source that is pretty complete and relatively easy to use, the NCAA. The NCAA posts all game statistics on their website making the trick not finding the data, but crawling their site and extracting it efficiently. For this project, I created several spiders to scrape the NCAA's website using scrapy, a python-based web scraper. The spiders were designed to extract all college football game data from 2013 to 2017 and save the results into CSV files. I chose this date range due to the similarly of data structures across years, and the resulting data included all completed seasons at the time of scrapping. Games before 2013 were in differing formats, including PDF files, which are difficult to parse. I did not include the 2017-2018 season as it had not concluded. In addition to game-level data, I also extracted historical data on the team and head coach.

Scraping the data from the site provided its own unique set of challenges. In fact, the time required to build, debug, and run the web scraper probably exceeded the amount of time

needed to analyze the data. Since the purpose of this paper is to outline the analyses and findings, I will only briefly describe the scraping process. I completed the site scraping in four phases: Identify, Pilot, Scale, and Scrape.

Identify

To begin, I spent a great deal of time examining how the NCAA's website worked and the data provided. For example, I realized that, over the last several years, there might be a few colleges who made the leap to FBS (Football Bowl Subdivision) as well as some who have dropped out. Thus, for each year, I was going to need to find a single page that listed all the FBS schools for that year. I then needed to scrape the link for each team. The resulting links provided the next page that I needed to request, parse, extract, and ultimately save the data from. The final result was the identification of which pages I needed to scrape, how to get there, and the order in which to complete the process.

Pilot

During the pilot phase, I used strategies identified in the previous step to create rudimentary spiders using Jupyter Notebooks and scrapy. The pilot phase allowed me to test the process on a subset of teams, and tune the process, code and output. In total, I created eight spiders. Although it may have been possible to crawl the site in one pass, I broke the process into smaller components. I did this for several reasons. First, it allowed me to simplify the code base. I didn't need to worry about overly complicated crawling patterns. Second, breaking it into parts allowed me to run small segments. If errors did occur, I could modify the system without having to rerun large chunks, making it slightly more robust to errors and failures. The table below outlines the various spiders as well as their purposes and outputs.

Spider	**Purpose**
Teamlinks_spider	Generate team links by year
PeopleHistoryRosterStats_spider	Generate links to coach, team history, roster, stats
Coach_spider	Extract coaching history (e.g. wins, losses, years in business)
Roster_spider	Extract team roster by year (e.g. Name, Position, year in school)
History_spider	Extract team history (e.g. wins, losses)
Teamstats_spider	Extract aggregate team statistics by year (E.g. cumulative year passing yards, rushing yards, fumbles, interceptions)
GameByGame_spider	Extract team stats for each game for each year (E.g. cumulative year passing yards, rushing yards, fumbles, interceptions)
GameBygGameTeamName_spider	Extract the opponent team name (e.g. Nebraska Cornhuskers)

Scale

Piloting with Jupyter notebooks provided a simple way to quickly develop a code base and make sure things were working correctly. During the scaling process, I did two things. First, I tried to simplify and reuse code where possible without becoming overly complicated. A great example of this is the use of a single pipeline, which output to several different CSV files. By using the generic system and pipeline, I no longer had to shuffle things during runs at the command line nor did I have to maintain specialized pipes for each file. The second significant improvement was the migration of the scraper from Jupiter notebooks to the typical scrapy format run from the command line. One of the major reasons for this was that the output generated from the spiders created large output logs in the notebook, which crashed the browser. Using the command line-based approach simplified the code and helped avoid this issue. Once scaled, the spiders could be run from the command line and the results saved to a CSV. You can view the scrapy program on my GitHub repository.

Scrape

The final step in the process was to run the various spiders in the sequential order outlined above. Some of the spiders, such as the Team Links spider, had a simple job: read one page, parse the HTML return, and save the appropriate links. Spiders such as these ran quickly and finished in minutes. Other spiders, such as the game-by-game spider, had more complicated jobs such as reading each team's statistics and crawling varying depths of pages with different categories. Running this spider alone took over 24 hours with a request about every 1.5 seconds. In total, the spiders created 61 different files and over 93 mbs of data. All links and files were zipped and are available on the GitHub repository.

Dataset Creation

My goal was to end up with a single row of data for each game that averaged all previous statistics for the home and opposing team. This data format would allow me to use all historical and game data to predict success (i.e., a win) in the current game. The creation of this type of data set proved challenging in several ways. First, I had to create a dataset that only included all of the games. Second, I had to create a second dataset that included all of the game statistics, which I used to calculate the statistical averages. Third, I had to merge in additional data from other sources, such as the historical and coaching data, once I created the game data. Again, since the purpose of this write-up is to explore the analyses and the machine learning algorithms, I will provide a high-level overview here. All information and the replicable code are available in the GitHub repository.

To begin, let's examine how I created the averages. The first game of 2013 was thrown out of analyses since no prior data was available for prediction. For all other games, I included all prior games for that year. For example, game nine would consist of all of the previous eight

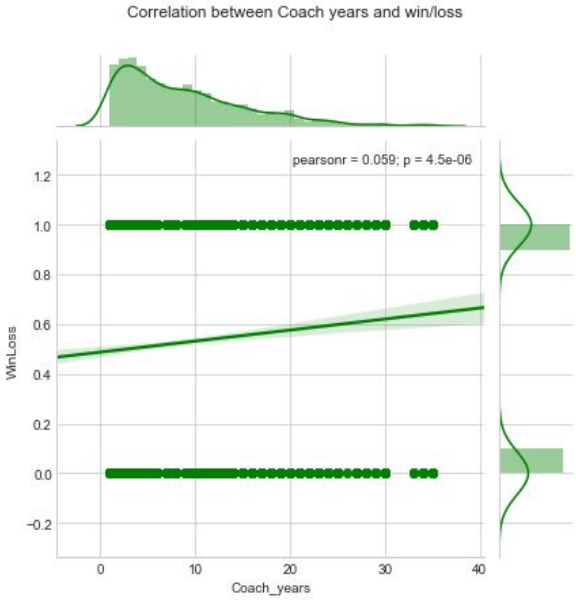
games; game eight would consist of the last seven games, etc. A problem occurs at the beginning of the year when little or no data is available for prediction. Since the beginning of the year has limited data, I decided to try and “smooth” the data by using the average of the previous year for the first three games of the target year?. For example, game one would be based on the average of the team’s success in the previous year. Game two would include game one plus the average of the previous year. Game three would consist of game one, game two, and the previous year’s average. After game three of the season, the average included only data from that same target year. I calculated all average football statistics in this manner.

After I created the averages for each game, I merged the data for the home team and the away team. I then added in historical data on the team's performance as well as the historical data on the head coach. The final result was one line of data used to predict whether the team experienced a win or a loss for each game. The data included the home team’s offense and defensive statistics and was mirrored by the offense and defense data of the opposing team. The hope here is that the machine learning algorithm would pick up the interaction of the home and away teams’ offensive and defensive tendencies.

Data wrangling for this project was tricky given the lagged nature, the lack of data when a new season started, and the fact that there were offensive and defensive statistics for two teams within a single game. That’s to say nothing of the work required to get the data! There are a few data wrangling issues worth noting. First, in instances where teams played non-FBS teams, the data was dropped, as I did not scrape the data for non-FBS teams. In cases of missing data, I used data imputation. The data was imputed based on columnar means, which may help or hurt teams. Second, some data was lost due to broken or timed out requests during scraping. It’s also worth noting that during exploratory analyses and some initial runs of the algorithms, each game was included in the dataset twice. The game was added once for the home team and once for the away team. Subsequent analyses dropped this approach in favor of a single inclusion.

Exploratory Analyses

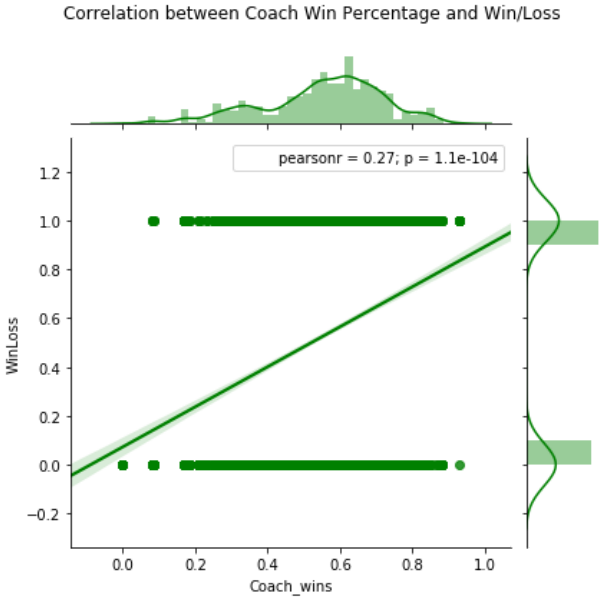
In the initial, exploratory analyses, I looked at different variables of interest. Being exploratory, they were not meant to create predictive models. They were intended to look at some of the typical “armchair” type analyses we might expect from fans and to make sure the dataset made sense. For these analyses, I used all available data, which included roughly 6,078 games.



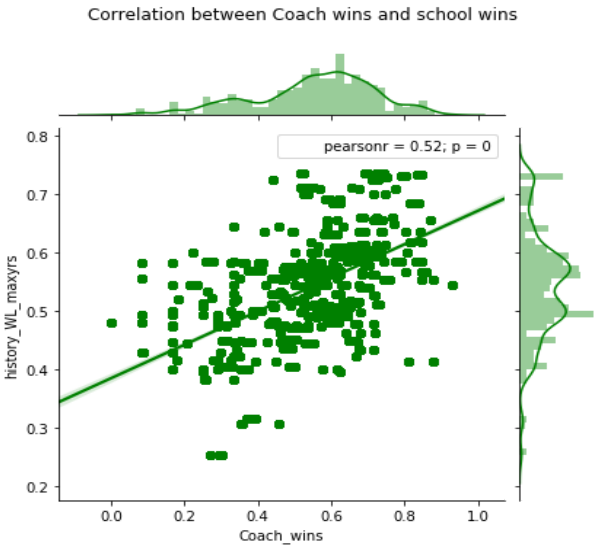
To begin, I looked at the relationship between the length of the coach’s career (in years) and game wins and losses, my hypothesis being that the

longer you survive in coaching, the more likely you are to win. Coaches who don't win games are unlikely to make it to 20 years as a head coach.

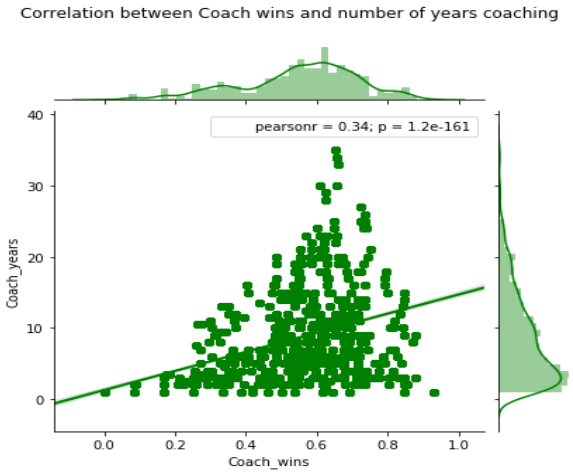
As seen in the chart above, there is a slight correlation between the number of years coaches have coached and the likelihood of winning the game. Interestingly, the correlation here (r = .06) is small. Therefore, coaching longer doesn't mean a coach will win many more games, although experience does seem to help. I think this makes intuitive sense as those coaches who lose probably won't survive long.



Next, I looked at the relationship between coaching win percentage and wins/losses of games. I hypothesized that coaches with high winning percentages are more likely to win the current game. The results indicate that there is a moderate correlation between a coach's win/loss percentage and the likelihood of winning the current game. With a correlation of *r* = .27, this is a medium-sized effect.



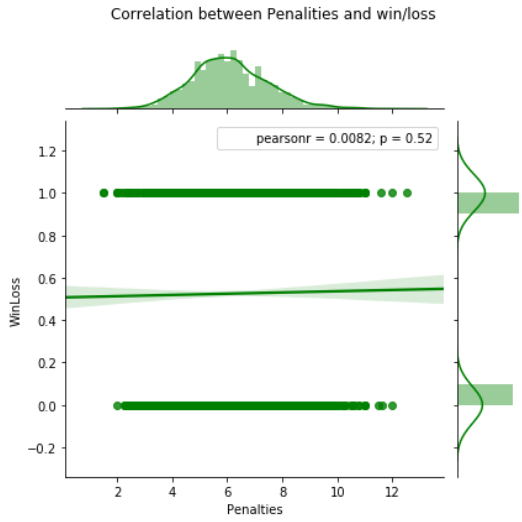
The next variables I examined were the relationship between coaching win percentage and school win percentage. I hypothesized that schools hire coaches that are at or above their average win percentage. Results of the analysis support this hypothesis. There is a healthy relationship (*r* = .52) between a coach's win/loss percentage and the history of the school. Put another way; most coaches mirror the average win/loss ratio of the entire history of the school.



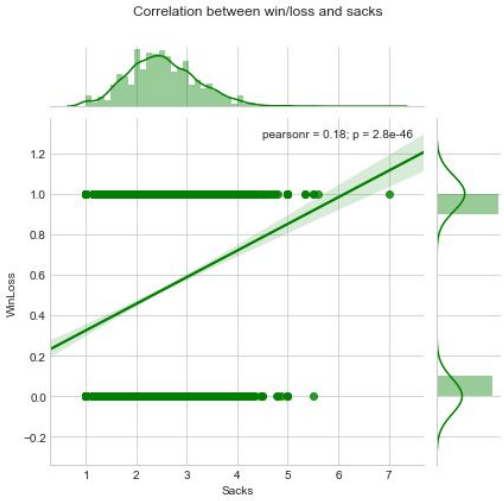
What about the relationship between the coaching win percentage and coaching years? Again, results indicate that there is a correlation between how long a coach has been coaching and his/her (or just his) overall win percentage.

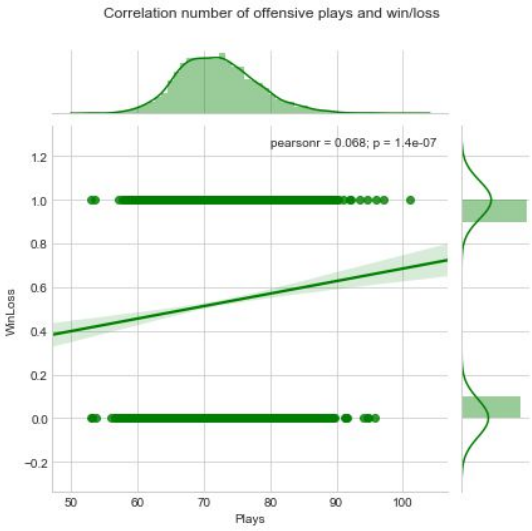
If we look at the median, it's .58 wins (58%, right? .58 sounds like less than 1 win). Looking at the graph, coaches who make it past 20 years tend to have higher win percentages. If you look at the extreme scores of 30+ years, they are almost all over the median. Coaches who win a lot of games can stay in business for a long time while those who don't often win don't make it to 20-30 years easily.

We also always hear a lot about offensive penalties during the game. Do offensive penalties matter in the outcome of the game? A review of the data suggests that there is a very small positive correlation, but the results were not statistically significant. Thus, teams that win usually have no more or less offensive penalties than those who lose. In fact, if anything, winning teams tend to have slightly more penalties than teams who lose, perhaps because they are more aggressive.



Another metric that is commonly believed to identify "good" teams is those who have a large number of sacks. Data in this sample show there is a small correlation between the number of sacks and winning (*r* = .18). Those with more sacks tend to win. Perhaps more aggressive style defenses win more games.

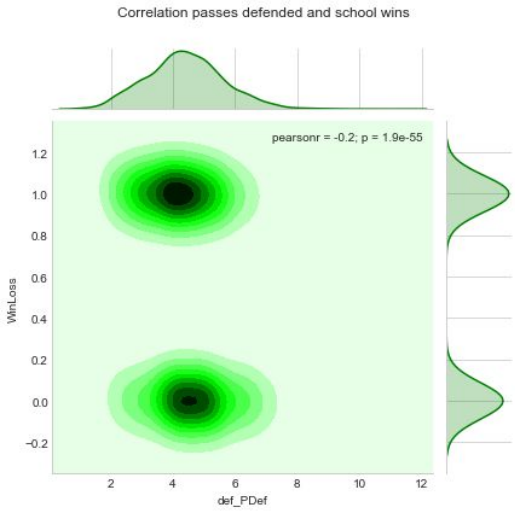




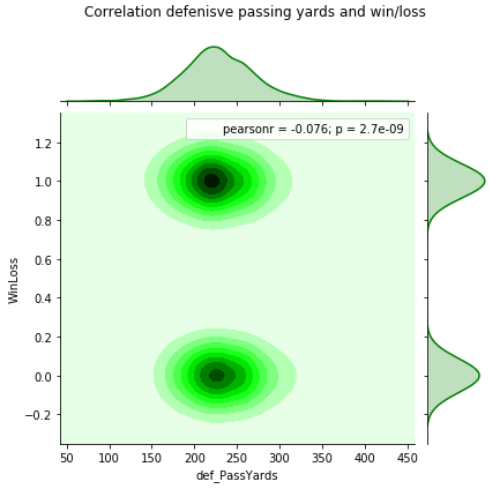
Another interesting statistic that has begun to pop-up over the last several years is the number of offensive plays. A lot of teams these days try to run a lot of plays, which, in theory, wear down the other team, put them in a better position and limit substitutions. In the day of these up-tempo offenses, do the number of plays make a difference for winning?

Data from this sample indicate a small positive correlation with more plays leading to more wins (*r* = ?). However, these results could be confounded by teams who merely run more running plays at the end of the game to run out the clock.

Another armchair-type statistic is the number of defensive yards given up. It's often said that the team that gives up the lowest amount of rushing yards is more likely to win. In the plot to the right, we can see that this tends to play out. Results here show that there is a negative correlation (r = ?). The fewer yards they give up, the more likely a team is to win. Using a kernel density plot, we can see there is a very dense population of around 150-180 yards or so given up for teams that win.

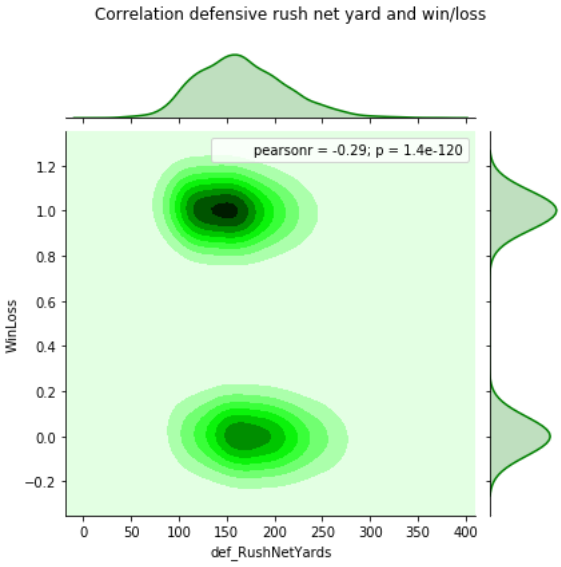


Do the same trends exist for passing yards? Again, there is a slight negative correlation, where winning teams give up fewer passing yards (r = ?). However, if you look at the KDE plot, you'll see that there is a lot of overlap between the graphs. It seems like winning teams clamp down on the rushing yards and passing yards are roughly the same for winning/losing teams and have similar ranges.



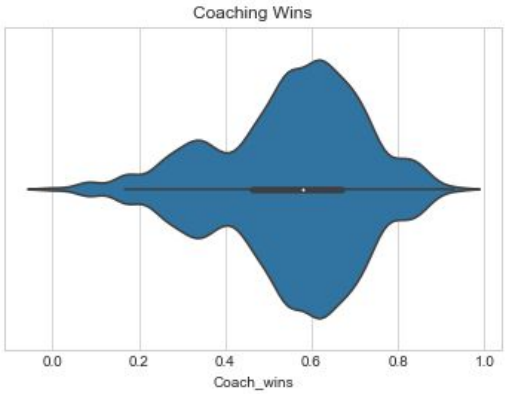
Is there a relationship between passes defended and games won? Similar to previous results, we again we see negative correlations with winning teams having more passes defended. If you look at the KDE plot, the number of passes defended is also much denser. The density could be because winning teams mount a more aggressive defense or because teams that are losing are forced to pass at the end of the game.

I also examined the correlation between coaching win percentage, historical wins at a school, and the likelihood? Of winning or losing a game. Results show that these variables are all positively related. Coaches with higher win percentages work at schools with higher win percentages and are more likely to win games. Granted, these are not mutually exclusive events, but the relationships are nonetheless interesting.

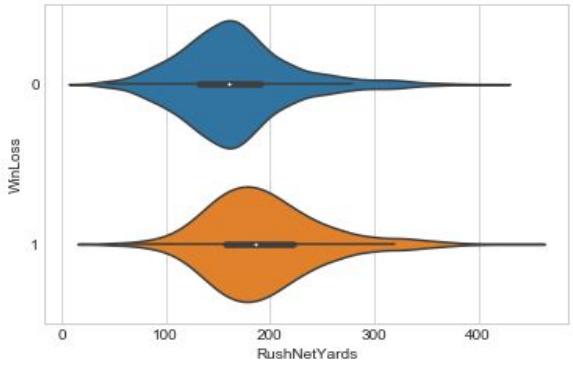


	Coach_wins	history_WL_maxyrs	WinLoss
Coach_wins	1	.52	.27
history_WL_maxyrs	.52	1	.21
WinLoss	.27	0	1

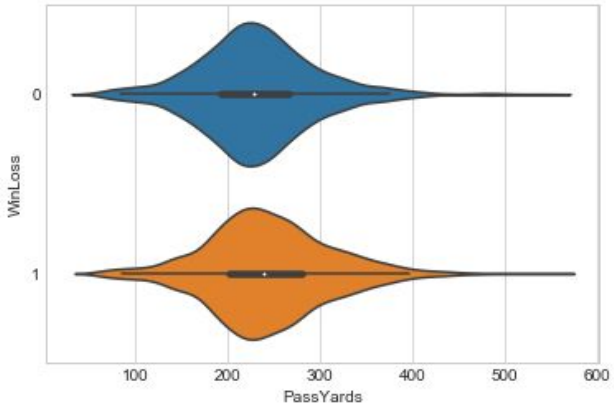
The results above promoted further exploration into some of the variables I had already explored. For example, the violin plot to right outlines the coaching win percentages. As seen in the graph, most coaches win approximately 60% of the time. There is a definite drop off of coaches who win more than 70% of their games and very few who win 80% or more.



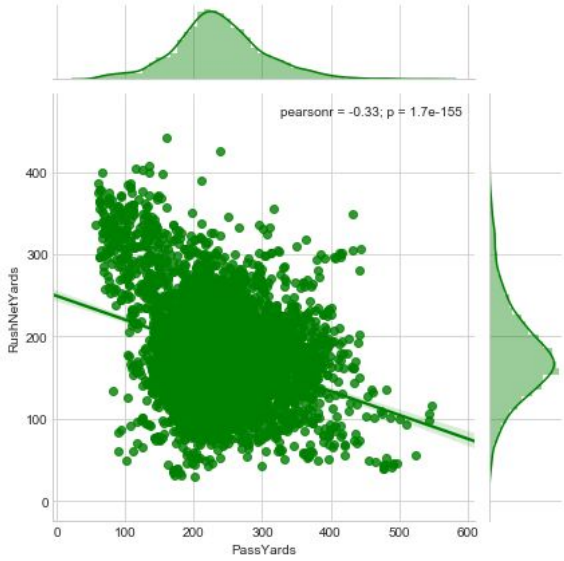
Similarly, with respect to the violin plot of rushing net yards and wins/losses shown on the right, the results show a pretty distinct difference between those who win and those who don't as. Those who have a more substantial number of rushing yards tend to win more games than those who do not. The average number of yards for teams who win is pushing nearly 200 yards.



Do we see the same types of trends for passing teams? Not really. Teams who win and teams who lose seem to have similar distributions. Between rushing and passing, it looks favorable to err on the side of rushing.



What are the relationships between passing yards and net rushing yards? Results here show that there is a negative relationship, which makes sense. Teams with high rushing yards do not tend to pass much. Based on the previous results, I would surmise that it is better to be a rush-heavy team than a pass-heavy team.



Machine Learning Model

The goal of this project was to create a model using machine learning that successfully predicted game success on the binary level. There are several different types of models which can predict binary outcomes with continuous features. These include models like Logistic Regression, Decision Trees, Naive Bayes, or a Random Forest Classifier. For this project, I chose a Random Forest Classifier. I selected a random forest rather than the Logistic Regression or Decision Tree as it is less likely to overfit the model to the training set. Using typical machine learning approaches, I used a train test split of the available data. Using a train test split creates a holdout sample of data not used to train the model. In this case, 33% of the data was held old for the testing set. By then applying the model to the test data, data the model hasn't seen before, we can get a rough idea of how the model will perform in unseen samples. Only cases marked as a "home" game were used so that no data was duplicated during training. Given a large number of variables, as well as missing data, I also decided to use a Pipeline which I describe below.

A machine learning pipeline is used to create a series of steps that can be applied to data sequentially. The sequence can include code to clean data; imputers to fill in missing values; feature selection, used to identify the most predictive variables; and modeling techniques such as the Random Forest Classifier used here. By using the pipeline, data scientists can build sophisticated models that contain several steps. For this project, I used a pipeline consisting of three stages: imputation for missing values, feature selection, and the Random Forest Classifier. Imputation for missing data was done using Scikit Learn's Imputer with mean

replacement of the column. As mentioned previously in this paper, by using the columnar mean, some teams may have been slightly advantaged or disadvantaged if their actual value differed significantly from their mean value for that feature?.

The next step in the pipeline was feature selection using a Random Forest Classifier. Feature selection is the process of empirically identifying the best predictive features. For this process, I used a Random Forest Classifier with a small number of estimators (100), a short depth (3), a minimal sample (20), and a variable selection threshold of .03. I chose a random forest to help avoid the overfitting of a single decision tree. I selected conservative hyperparameters to avoid overfitting the model and to make sure the most predictive features were identified. I set a threshold of .03, given the smaller correlations of the features and the outcomes. I used a random state value, which allowed differing runs of the algorithm, so all results were comparable.

For the final prediction, I again used a Random Forest Classifier. Since the final selection for the model was made using grid search, I provided a wide variety of hyperparameters. I used both gini and entropy criteria; a maximum tree depth of 1 to 5; number of estimators of 100, 500, 750, 1,000, and 1,250; and, finally, I set the minimum number of samples between 10 and 50, increasing by 5. I set the cross-validation fold number to three, and I again set a random state for both the forest and the randomized grid search for replicable results.

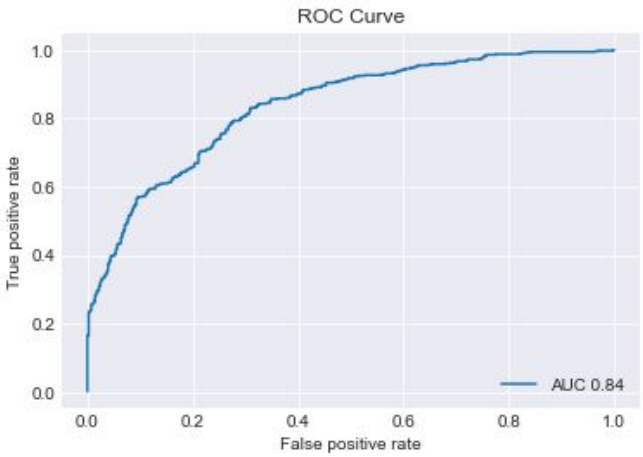
Results

After performing the grid-search, sci-kit learn returns the best fit model. The best fit model used an entropy criterion; maximum tree depth of 5; minimum sample split of 15; and 1,000 estimators. The predictive accuracy was .77. I list the most predictive features and their relative importance in the table below.

Feature	Importance
Rushing - Net Yards	.15
Punt Return Yards	.15
Fumble Return Yards	.14
All-Purpose Yards	.14
Receiving Yards	.12
Interception Return Yards	.11
Kickoff Return Yards	.11
Rumbles Recovered	.07

A review of the test score data indicated a predictive accuracy of .76, mirroring that of the trained model. The Receiver Operating Characteristic (ROC) Curve and the Area Under the Curve (AUC) show good results (AUC = .84).

The confusion matrix for the ROC curve is shown below.



		Actual	
		Loss	Win
Predicted	Loss	246	162
	Win	78	509

Finally, I also examined a classification report, which outlines the precision, recall, and F1-score of the prediction. My/the precision rate of .76 indicates a low false positive rate. The recall rate of .76 indicates that the model labeled the majority correctly. The F1 score is the weighted average of Precision and Recall. As we can see from this measure, as well as from the precision and recall metrics, the model does as well predicting wins as it does losses.

	Precision	Recall	F1-Score	Support
Loss	.76	.60	.67	408
Win	.76	.87	.81	587
avg/total	.76	.76	.75	995

Overall, these results indicate that the model produced by the Random Grid Search with cross-validation is predictive and stable in the training and test sets.

Discussion

Results from this study illustrate the utility of using machine learning techniques for predicting national football league? game success. In fact, the final model successfully predicted game wins/losses 76% of the time. The most predictive features were selected using machine learning techniques. The exciting thing about this approach is that the empirically chosen features were not the ones I would have picked based on everyday intuition. A theoretical approach would have pointed me towards coaching win percentage and team history. Instead, the most predictive features were related to metrics of yards within the game. To me, this was an unexpected finding, but, as I have thought about it after the modeling, it makes sense. The historical slant is probably more of a human predisposition. Our experiences create heuristics that help us make predictions and history of a team is an example of this. The machine learning and exploratory analyses showed us that, while these do have small correlations, other, more basic, statistics through the year account for much more variance, explaining why they were selected by the model. Similarly, I expected time of possession and penalties to be a driver in predicting success, but they were not. Again, the exploratory analyses showed that the relationships here were small and unlikely to determine a large number of game outcomes. Based on the mantra of “defense wins championships,” I expected that defense-related features would also be essential predictors, but they were not. Defensive variables did not make it into the final model. My hunch is that some of these features were not predictive because they are co-linear with other variables or may be masked by interactions. For example, defensively dominant teams will have more offensive opportunities and thus have more opportunities to gain yards, factors that emerged as key predictive features in the model. Taken as a whole, I feel that these analyses were quite enlightening, not only for the techniques and outcomes, but also as a re-evaluation of what matters for teams that win. History and experience are great but what teams do on the field is much more predictive of their success.

Limitations and Future Directions

There were several different limitations of this study. First, scraping the data from the website was not without hurdles. Several of the links within the sites returned error messages, resulting in missing data. Future research and scraping should create a list of failures to get a better idea of how much data is lost and if there are better ways to recover it. As it stands, I do not have a metric for the overall amount of failures due to these errors, a factor which could bias the results.

Similar to bad requests, another problem is that of missing data. Some fields within the site were scraped but contained no data. The treatment of these null pieces of data was troublesome. Ultimately, I set a large number of these variables to zero because they were low base rate variables. For example, no fumbles, no passes broken up, no returned fumbles, etc.

Other variables with more substantial variance may have been influenced more significantly because of this.

Imputation of some of the missing data could have caused over- or underestimation of results. For this study, I imputed the missing values based on the average value of all data available for all teams. In my opinion, a better approach would be to impute the value based on the average of the individual team's data instead. Future studies may want to take a look at this procedure and see if it significantly alters the results.

Another potential problem with this study is the data chosen for analysis. For model creation, the data used was only for those cases marked as playing a home game. I selected home games to avoid using the same data twice. Choosing the home team may or may not have caused problems with this process. Future studies may wish to include both pieces of data into the data sets or randomly select either the home or away team.

Features included for feature selection is another area for review. In this study, I used a fraction of the scraped data. Additional data on the game experience of the team, individual contributors, etc. are also available. Being able to include how many games players have played, injuries to key players and the depth at positions may also make a difference within the resulting models.

Future research should also review the methodology used for feature selection. The parameters used for feature selection may be further tuned, and alternative methods such as Lasso or KBest may also be more appropriate. Spending more time on this step may produce better results.

Finally, the pipeline and final model should be reviewed. As I mentioned in the paper, alternative machine learning methods may be more predictive. By using the pipeline approach, future research is set up to easily incorporate and test new techniques such as Bayes or Logistic Regression.