Predicting NFL Coaching and Team Based Play Calls

By Austin Grosel

DePaul University
CSC 529 - Advanced Data Mining
Final Project

Introduction

In the National Football League (NFL), there are two types of calls every offense chooses before each play (aside from punting): run (also called rush) or pass. Both play types have their tradeoffs: passing is more efficient than rushing, but it can lead to more turnovers, making it a bigger play call risk. Rushing the ball is less efficient, but it's less risky than passing and a team is able to control the game clock. There are different situations that determine whether a team will rush or pass the ball: teams tend to pass on third downs or when they need to get a large chunk of yards and rush when they are winning and want to run out the game clock (Orr).

Because there are only two choices for every play, we are able to set up a classification analysis to predict if a team will run or pass based on the different game variables presented before the play happens. If NFL defenses are able to accurately predict an offensive play type, they can game plan with their alignment or personnel to combat the play type properly.

NFL games are divided into four 15 minute long quarters. Past studies on this topic have shown the most accurate results happen when there are four separate models for each quarter, and that the 2nd and 4th quarters are much easier to predict than the 1st and 3rd quarters (Jones). This is because teams tend to pass frequently at the end of the 2nd quarter before halftime, and point differential in the fourth quarter has a large effect on play calling (the team losing will likely pass). I believe that because point differential makes 2nd and 4th quarter play calling trivial, the most practical use for the model would be during the first quarter, when a team's point differential does not have such a large effect on the prediction.

Commonly, this classification analysis takes league wide play by play data and tries to find one machine learning method to get the highest out-of-sample accuracy (as was done in Case Study 1). I believe one of the problems for this setup is that teams and coaches tend to have different theories on how to execute their game plan based on the states presented to their team. A team with a prolific quarterback may be inclined to pass more. A coach may be more conservative in 3rd and long situations. When computing an overall classification model, I believe that we are losing nuanced information that could be captured if we split the data based on a coach or a team.

To test this theory, I will subset the data for each coach and choose the method with the highest accuracy. I will also do this same process with a team subset. I will finally classify play calling on the entire dataset, ignoring coaches or teams and compare and contrast the different methods. In the end of my analysis, I will demonstrate a use case where I pick a random team as an upcoming opponent and explore the different possible models to choose from if I were helping a defense game plan against their opponent.

For this project, the most important metric to compare against other methods is accuracy. Because there are only about 25 seconds before a play starts, a non-performance metric like

model scrutability is not necessary. We would not have enough time to explain to the players why the model outputs a pass or run. Also, recall and precision are not necessary in evaluating the model because there is no positive or negative class. Guessing wrong would be evenly detrimental for both play types. While some models will predict pass or rush more often than others, these two metrics will not be used in the final model selection. Ultimately, we want the most accurate model possible.

Brief Overview of American Football

American football has a diverse set of rules compared to other sports in the world. However, like many sports, there's an offensive team trying to score points and a defensive team trying to prevent the offense from doing so. When an offensive team receives possession of the ball, they start a drive, which is a series of plays linked together that end in a turnover or points scored. To score points, the offensive team must "drive" down the field with their series of plays and either kick the ball through the opponent's field goal posts or run across the opponent's endzone line with the ball. In the middle of the drive, the offense must continuously cumulate 10 yards in 4 plays. These sequences are called downs, and they range from 1st to 4th down. If the team fails to get past the 10 yard marker on 4th down, the ball is turned over to the defensive team. If an offense is able to get at least 10 yards, it is called a first down, and the down sequence resets until the ball is turned over or the offense scores points. You can also think of achieving a first down as a checkpoint in the drive sequence.

On offense, a play can either be a pass or a rush. A rush attempt is when the ball is handed off to a position player behind the line of scrimmage, while a pass attempt is when a player throws the ball past the line of scrimmage to another player. Typically, when the play is over, the offense gets about 40 seconds to call their next play. This gives the defense time to assess the different variables before the next play. Here is where my model will hypothetically take over. After evaluating the factors immediately following the previous play, we want to relay the model's output to the defensive team on whether the offense will pass or run the following play.

Data Description

There were a few different datasets used for the analysis. The main dataset used for this analysis is an NFL play-by-play set from the site ArmchairAnalysis.com (Erny) ranging from the 2000-2016 NFL seasons. Some of the data in the early seasons had flaws like attributing the play's minutes and seconds left to be the same as the first play of the drive. Each observation of this dataset is an NFL play, with over 740,000 plays. The target variable in the dataset is "type", which will be reduced to only include passing or rushing plays, and a boolean variable will be created to determine if it was a pass or not (pass_bool). The following table below shows the different columns:

Play-by-Play table

Variable name	Variable Type	Description

gid	Numeric	Game ID#
pid	Numeric	Play ID#
off	Categorical	Offensive team name, 32 levels
def	Categorical	Defensive team name, 32 levels
type	Categorical	Play type, 6 levels (will be reduced to 2 levels)
qtr	Ordinal	Quarter number, ordered 1-4
min	Numeric	Minutes left in the quarter
sec	Numeric	Seconds left in the quarter
ptso	Numeric	Cumulative points scored by the offensive team
ptsd	Numeric	Cumulative points scored by the defensive team
timo	Ordinal	Timeouts left for the offensive team, 0-3
timd	Ordinal	Timeouts left for the defensive team, 0-3
dwn	Ordinal	Down, ordered 1-4
ytg	Numeric	Yards to go to get a first down
yfog	Numeric	Yard line the ball is at, with 1 being one yard away from own end zone and 99 being one yard away from the opponent's end zone (most likely to score)
yds	Numeric	Yards gained on the play
succ	Categorical	Was the play classified as a success, 0 or 1 (Explained more in depth below)
sg	Categorical	Was the QB in shotgun, 0 or 1 (Explained more in depth below)
fd	Categorical	Did the play result in a first down, 0 or 1
nh	Categorical	Was the play called with no huddle, 0 or 1
pts	Numeric	The amount of points scored on the play

The variable "succ" is a boolean variable set by the database administrator to flag if the play was successful or not. The way we the DBA determines success is if the team gets at least 40%

of the yards to go on 1st down, at least 60% of the yards to go on 2nd down, or achieves a first down or 3rd or 4th down (Erny).

The variable “sg” is flagged if the quarterback (QB) of the team is in a shotgun formation. Quarterbacks are usually the first player that touches the ball and attempt the most passes on offense. Figure 1 below shows an example of a shotgun formation and a non-shotgun formation.

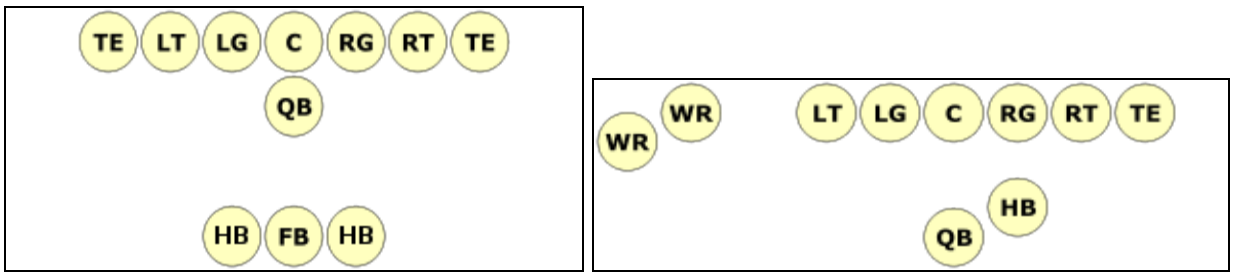

Figure 1. These two images show a formation of a QB under center, or not in shotgun (left) and a formation of a QB in shotgun (right).

The diagram on the left shows the QB right behind the center (C) (sg = 0), while the diagram on the right shows the QB behind the center but with some space in between them (sg = 1). This is the idea behind the shotgun offense. The vast space between the QB and the center gives him a much better view of the defense, and he doesn’t have to retreat as far back to attempt a clean pass over his linemen.

Another dataset that was included in this analysis was a table that held every game’s information from the 2000-2016 NFL seasons (Erny). The following table below shows the fields from this dataset:

Game Info table

Variable name	Variable Type	Description
gid	Numeric	Game ID#
seas	Ordinal	Year the game took place, from 2000-2016
wk	Ordinal	Week number, 1-21
day	Categorical	Day of the week, 4 levels
v	Categorical	Visiting team, 32 levels
h	Categorical	Home team, 32 levels
stad	Categorical	Stadium name

temp	Numeric	Temperature outside the stadium
humd	Numeric	Humidity
wspd	Numeric	Wind speed, in MPH
surf	Categorical	Surface material, 6 levels
ou	Numeric	Vegas over/under
sprv	Numeric	Vegas spread
ptsv	Numeric	Points scored by the visiting team
ptsh	Numeric	Points scored by the home team

The last dataset is crucial for the new extension from Case Study 1. To figure out if we can predict play calls based on which coach is calling the plays, we would have to get a database of each of coach in the last five or so years. Luckily, a site called “The Huddle” has a list of the different head coaches, offensive coaches (often called coordinators), and defensive coordinators (The Huddle). After building a scraping tool to import the different coaching staffs from each of the past five years, I was able to get a table that had the following variables below:

Coaches table

Variable name	**Variable Type**	**Description**
seas	Ordinal	Year the game took place, from 2012-2016
team	Categorical	NFL team, 32 levels
HC	Categorical	Head Coach, 57 levels (57 different head coaches)
OC	Categorical	Offensive Coordinator, 84 levels
DC	Categorical	Defensive Coordinator, 89 levels
WeekStart	Ordinal	Week of the season the three coaches started, ranked from 1 to 17
WeekChange	Ordinal	Week of when there was a coaching change between one of the old
HC_PlayCaller	Boolean	1 if the HC calls plays, 0 if the OC calls plays

Often, a team will switch their coaching staff around during the season if the organization feels like the team isn’t performing up to expectations. Thus, a WeekStart and WeekChange variable

were created to keep track of when the coaching staff changes. These two variables are vital when performing merges with the main dataset.

In the NFL, play calling duty can differ from team to team. Sometimes your head coach calls the plays, but when your head coach is more of a defensive minded instructor, the offensive coordinator tends to call plays. Thus, I created a boolean variable (HC_PlayCaller) to designate if the head coach of the trio of coaches in the table is the play caller. The process of determining which coach called the plays was a bit subjective. While it's not true for some organizations, I based the result on giving the head coach the play calling duties if he was an offensive coordinator at one point in his coaching career. While this does not explain every team's situation (an offensive minded head coach may not call the plays), I felt it generalized the rest of the league relatively well.

Data Cleaning

The first data cleaning step was merging the two tables mentioned above by their game ID#. That created a dataset of 740,000 plays and 35 variables. Because I only had coaching data since 2012, I cropped out all plays that happened before this year. I also removed all plays that did not have a "PASS" or "RUSH" type. The final number of observations fell to around 165,000.

From there, I wanted to merge the play calling dataset with the coaching dataset. To do this, I matched up the offensive team (off) and the week variable (wk) so the coaching trio can map to the correct games that they've coached. When this was merged correctly, I added a "PlayCaller" variable to designate which coach was the main play caller based on the HC_PlayCaller variable.

The next step was to modify some of the pre-existing variables. Twenty-five different levels of weather conditions seemed too much, so I generalized the conditions into four different categories: Clear, Rain, Snow, or Dome. The Dome condition was given to any game that was played inside a stadium with a roof over it.

The other modification was made to the Vegas point spread (sprv) variable. For every game, the Las Vegas bookkeepers create a point spread number that predicts how many points one team will beat the other by. The larger the number, the larger discrepancy there is between the two teams. This variable was modified so that the number would match the correct spread value given which offense was on the field.

Finally, it was time to add new variables to the table. I first created a point differential variable (pts_diff) by subtracting the ptsd variable from the ptso variable. I also added two variables to keep track of how many fumbles and interceptions have been thrown (cum_fum and cum_int, respectively). These were added because I hypothesized that if a quarterback is throwing interceptions, a coach may stay away from passing and if a running back is fumbling often, a coach may stop running the ball.

Next, I created a seconds left in the quarter variable (time_rem) by adding the sec variable to the product of 60 and the minute variable. One hypothesis I had was to see if there was an effect on play calling when a team is about to score. In football, when a team is 20 yards away from their opponent's endzone, it's called the redzone. There is a high percentage chance that the offensive team will come away with points at the end of the drive. Thus, I'd expect play callers to be more conservative (rushing the ball more) so they're able to secure at least some points before the drive ends. This variable is called redzone_bool, and it's a 1 when the team is in the redzone and 0 when it is not.

Something else I wanted to see was if a team is having success running or passing the ball on their drive, will they continue to do so? I created a previous rush success (prev_rush_succ) and previous pass success (prev_pass_succ) by observing if the previous play was a success and what play type was called. The last variable that I added was an in game passing ratio. My thought process behind this is that we can find out if a team hasn't passed much, they're likely to pass more and vice versa. This variable would be called in_game_pass_ratio.

Data Analysis

I first wanted to calculate how many passes and rushes there were in the dataset. Taking the mean of our classifier (pass_bool), teams called a pass on 58.5% of plays the first quarter. In Case Study 1, which had data since 2009, this number is up from 53.9%. We can conclude that teams recently have been passing more than in the past. The 58.5% number is a good baseline for guessing if a play will be a pass or rush. If we were to guess a pass on every play regardless of the different pre-play factors, we would be correct 58.5% of the time.

Figure 2 below shows the correlation plot from Case Study 1 with pass_bool (the target variable) versus the other continuous variables.

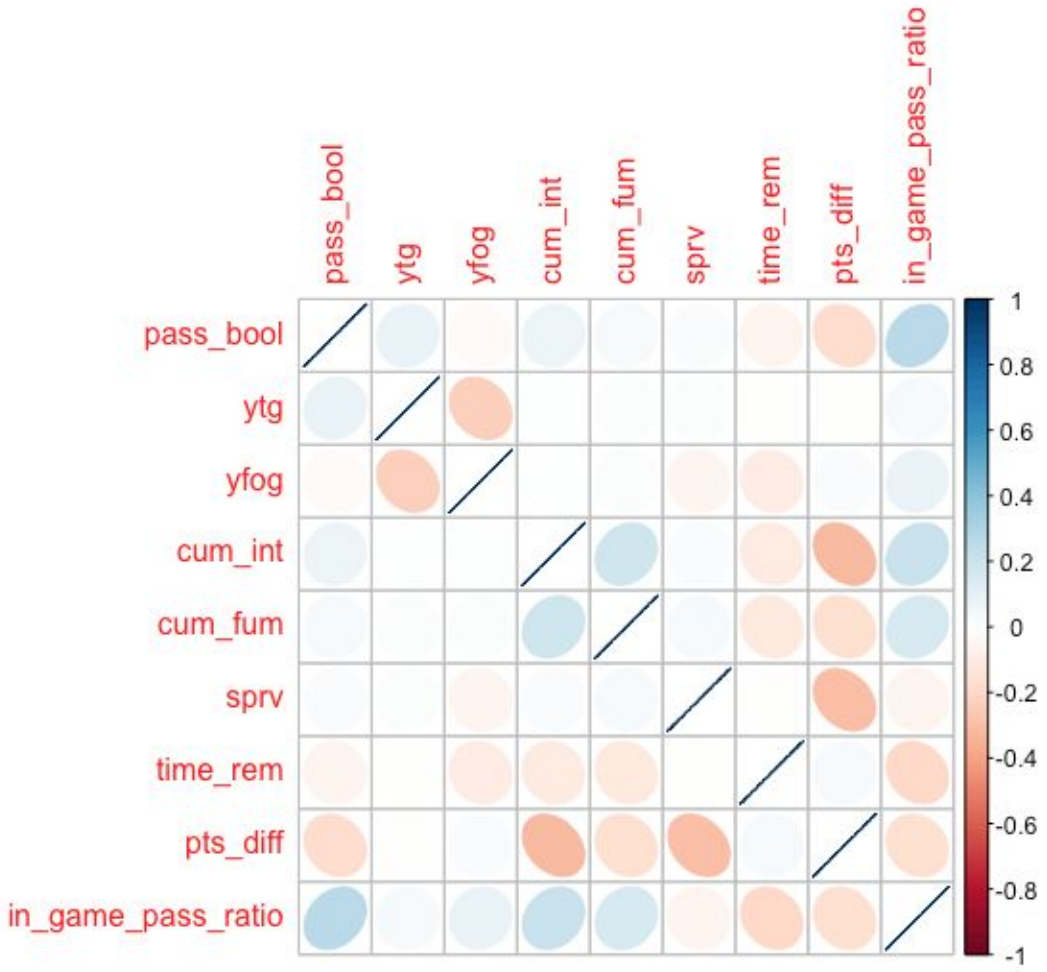

Figure 2. The correlation plot between the continuous variables of the data.

My hypothesis stated earlier was that a team that had been passing more in a quarter would start to pass less to try to get the pass/run ratio to 50/50. However, with the correlation being positive, that meant teams who are passing frequently tend to keep passing and teams that run frequently tend to keep running. I discovered this correlation may be biased to what happens on the first play of the game. When I filtered the data to only include observations that had a 0.00 in_game_pass_ratio (which happens at the start of every game because no passes have been thrown), there is a 63.0% chance that the play call will be a run. That's a huge difference from the 41.5% run plays in the entire dataset. Therefore, I believe the models will probably predict a run almost every time before a team throws a pass.

As for the other continuous variables, there aren't many variables that seem to be highly correlated with pass_bool. The only variable with slightly any significant correlation is ytg (yards to go) which is slightly positive. This makes sense because as a team gets farther away from the first down marker, they're more likely to pass and pick up large chunks of yards rather than running the ball, which is less efficient.

The next step is to view the relationship between pass_bool and the categorical variables. The bar graphs below show the distribution of pass_bool among the variables with different factor levels. Figure 3 shows the different barplot counts based on passing or rushing.

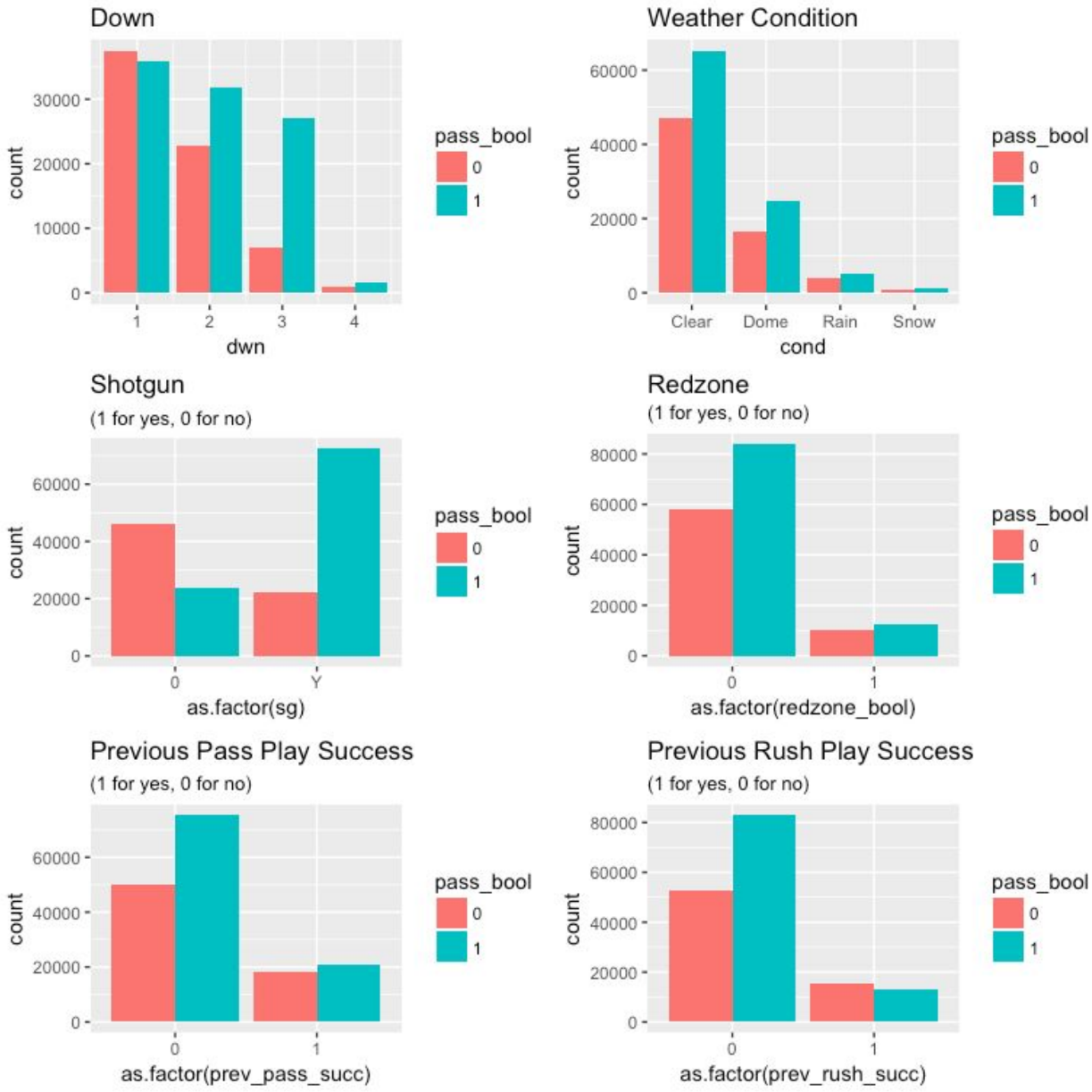

Figure 3. Barplots showing the relationship of our target variable pass_bool vs. each of the categorical variables.

There's more information to take away from the data's categorical variables than the numeric variables. Starting with the down variable, we see that most teams run the ball on 1st down (43.4% passing), however on 3rd downs, teams increase their passing tendencies to 81.4%. That projects like it could be a significant variable in our model later on. There doesn't seem to

be too much differentiation between the weather condition variables. Domes, or indoor stadiums, have the highest passing percentage (55.2%) and games played in snow have the lowest (47.9%).

Moving on to the shotgun variable, it seems like there's a significant effect in play calling if the quarterback is in a shotgun formation. Teams pass 74.8% of the time in the first quarter when the quarterback is in shotgun, and only 37.7% when he's not in shotgun. That's a giant discrepancy that I'm sure will show up in the model. The redzone variable doesn't have much of a difference, but we can see that teams tend to run more when they're nearing the opponent's endzone.

There's not much of a difference between previous successful rush plays, but there seems to be more rushing plays called after a previous successful pass. There could be a few explanations for this. First, when a team gets a first down, it's automatically deemed as a successful play, and we've shown teams tend to rush on first down. Second, coaches tend to want to stay balanced on their play calls. Coaches may think to call a rushing play immediately following a successful pass play to keep the defense off balance. Overall, I'm pleased with the exploratory analysis results. Going forward, I believe we'll see some combination of categorical variables in our models, with shotgun and 3rd down being the most prevalent.

The last part I wanted to explore is the difference between teams and coaching and the different passing ratios for themselves. It'd be interesting to look back in retrospect if I was much better at predicting a team that tended to run the ball or pass the ball more frequently. The stacked bar plot in Figure 4 displays the passing ratios between each of the 32 NFL teams.

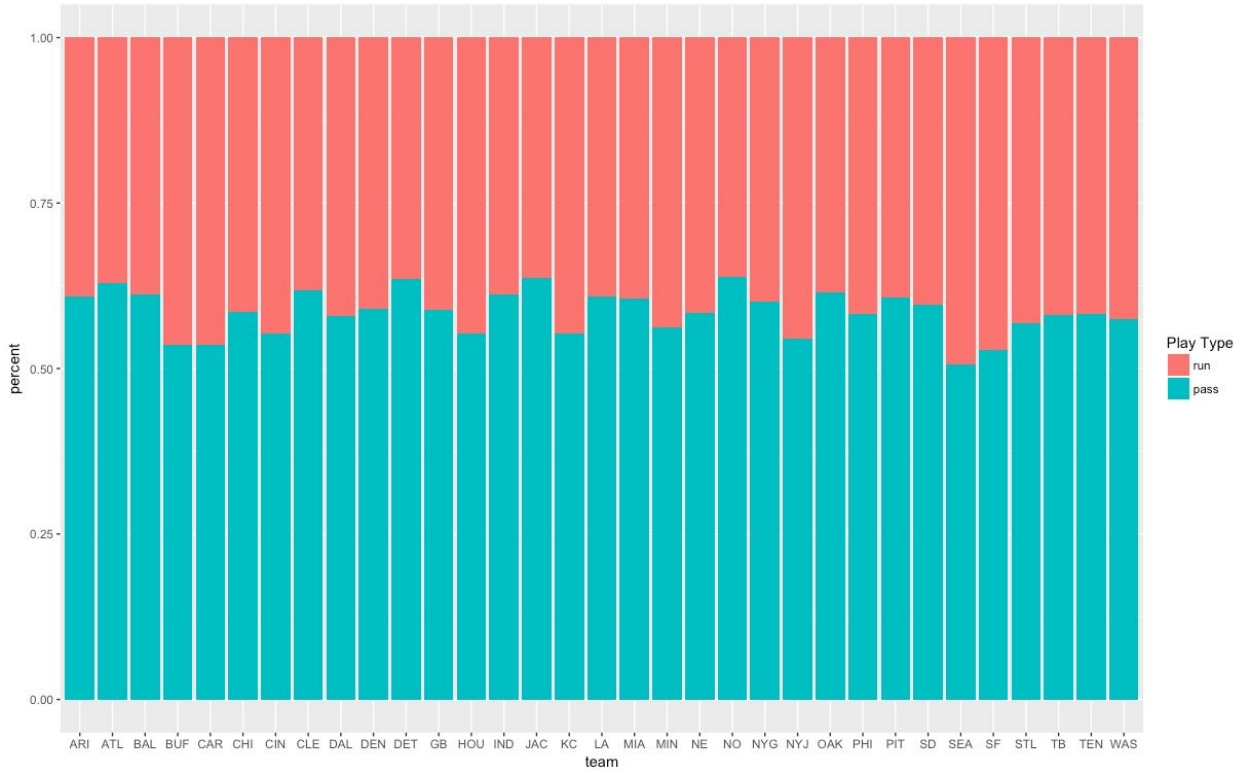

Figure 4. Stacked barplots for the pass to run ratio for each of the 32 NFL teams.

The green bars display the percentage for passing while the red bars show the percentage for run plays. In the past five years, we see the most pass happy teams are Jacksonville (JAC), New Orleans (NO), Atlanta (ATL), and Baltimore. The run heaviest teams seem to be Seattle (SEA), San Francisco (SF), Buffalo (BUF), and Carolina (CAR).

The stacked bar plot in Figure 5 looks at the passing ratios for the different coaches. Keep in mind, some of these coaches may have a low sample of play calling.

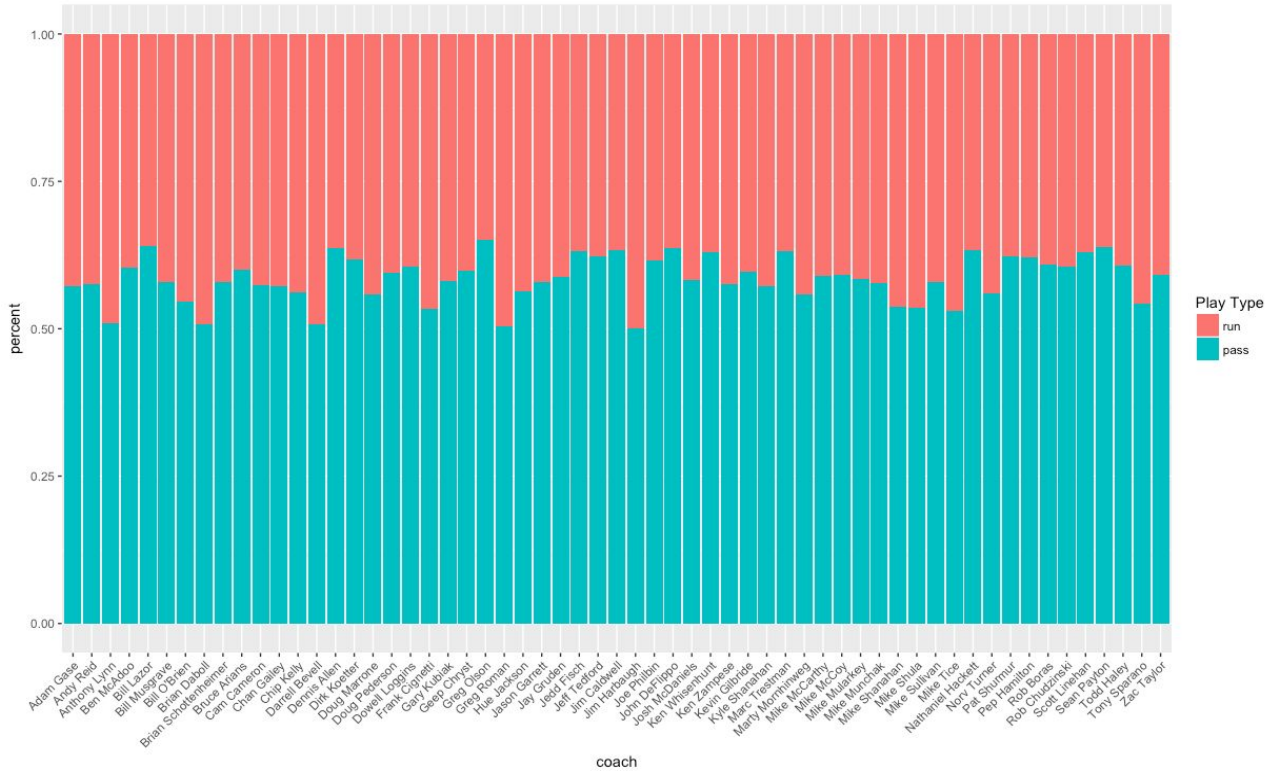

Figure 5. Stacked barplots for the pass to run ratio for each of the 40 NFL coaches.

While it may be tough to see in this document, the coaches that tend to pass the most are Greg Olson (former Jacksonville OC), Bill Lazor (former Miami OC), Dennis Allen (former Oakland HC), and Sean Payton (current New Orleans HC). The coaches that pass the least seem to be Jim Harbaugh (former San Francisco HC), Greg Roman (former Buffalo OC), Darrell Bevell (current Seattle OC), and Daboll (former Kansas City OC). Overall, we see that there seems to be some overlap between coaches that passed a lot along with teams that passed a lot.

Experimental Process

The experimental process of the model selection was modified from the first case study. In Case Study 1, I looked at four different machine learning methods using the entire dataset, computed a t-test to compare the four models, and chose the machine learning method that gave the highest accuracy. In this project because I will be doing this four method comparison between about forty different NFL coaches and thirty-two different NFL teams, I will simply take the highest accuracy value for the different coaches and teams and compute a t-test later if I want to put a coach's or team's model in production.

The steps of my process will be to calculate a first quarter prediction model for all coaches in the NFL that have coached at least one season in the NFL as their team's main play caller. I estimated that since there are about 65 offensive plays for a team in one game, there are about

16.25 plays for the first quarter. Multiplying that by the total amount of games in one season (16) gives us 260 plays. Thus, each coach had to have at least 260 plays in the dataset to qualify in my analysis. This resulted in 40 total coaches.

The four different methods I will be comparing for each of the coaches are logistic regressions (GLM), decision trees (DT), support vector machine (SVM), and random forests (RF). While scrutability is not important in this application, I do like to see which variables tend to come out on top when predicting a pass or rush, just in case I need to tell a coach or scout or player (which may not have as much machine learning knowledge as I do) what the model is saying. For GLM, DT, and RF, we can observe which variables are significant in the prediction process. I added a Gaussian kernel support vector machine because SVM's were not addressed in the first case study, and I wanted to see if SVM's could help the prediction accuracies for some coaches. I chose the Gaussian kernel since it gave the most accurate results in my Case Study 2; however, in a later project, it'd probably be wise to try a few different kernels as well.

Then I will complete a similar process for each of the 32 NFL teams. There is no need to filter out any of the teams because each of the teams have participated in about the same amount of first quarter plays for the past five years.

Finally, I will repeat my first case study. I will ignore team and coaching distinction and focus on the entire dataset at hand. From there, I will use a t-test to find which machine learning method will give us the highest accuracy. This will be called my non-discriminatory model.

My last step will be to analyze the difference of accuracies between the coaches, teams, and the non-discriminatory model on a case-by-case basis. I will randomly select a team as an opponent and decide which of the models I would use if we were playing that team in an upcoming game.

Experimental Analysis and Results

I first created a for loop that would cycle through each of the different coaches in the full dataset. As I mentioned before, if a coach called less than 260 first quarter plays, they were skipped for this analysis. Loading the caret library in R, I used the train function to flip through the four methods that were talked about in the previous section. The validation set was tested using a repeated bootstrap method with n = 100.

I also created a tuning grid for the decision tree and support vector machine methods (GLM's do not take in modified parameters in caret's train function). The decision tree's complexity parameter (cp) is a sequence from 0.01 to 0.05. For decision trees, the smaller the cp value, the more likely the model will overfit. For the support vector machines, there were two parameters: sigma and C. Sigma is the Gaussian standard deviation. The higher the sigma value, the more generalized and flexible the classifier boundary will be. C values are somewhat opposite of

sigma. The more flexible boundary comes when C is lower. This can cause the training accuracy to decrease, but it generalizes the model well for our test data.

In the for loop, I used an 80/20 split to divide a training and test set for each coach. The train function finds which parameters give the highest accuracy metric on the validation set for each method. From there, it uses those parameters to calculate the accuracy on the test set. These test set accuracies are stored in a table with the coach, the method used, and the test accuracy. The highest accuracy among of the methods for each coach is also recorded in a different table. An example for the coach Adam Gase is shown below:

Coach	Method	Mean_Accuracy
Adam Gase	GLM	0.7663
Adam Gase	Random Forest	0.7282
Adam Gase	Decision Tree	0.7174
Adam Gase	SVM	0.5380

In this example, the highest prediction accuracy for Adam Gase's plays was achieved using a GLM, so that accuracy and the method was placed in a separate table as well.

At the end of the for loop, two tables will be generated. One was a table of the 40 coaches, the highest prediction accuracy recorded, and the method used to get that highest accuracy. This will be referred to as the Coach's Best Prediction table. There was also a table like the one shown above, with 160 entries of different method accuracies (four methods for the 40 coaches). This will be called the Coach's Total Predictions table moving forward.

Let's first dive into the Coach's Best Prediction table. The two coaches with the highest accuracy recorded were Jason Garrett at 81.0% and Dirk Koetter at 80.9%. These two accuracy metrics were much higher than the 70% accuracy measured in Case Study 1 when we ignored coaching. A decision tree computed Garrett's highest accuracy while a GLM computed Koetter's highest accuracy. The two coaches with the lowest accuracy values were Doug Marrone at 64.2% using a GLM and Dennis Allen at 64.6% using a decision tree.

The distribution of the Coach's Best Prediction dataset is shown in the table below:

Method	Count	Mean_Accuracy
GLM	21	0.7408
DECISION TREE	13	0.7317

RANDOM FOREST	3	0.7166
SVM	3	0.7130

From this table alone, it seems like the GLM method had a lot of success when computing the best accuracy metric for each of the different coaches, making up 52.5% of the best predictions and having the highest mean accuracy of the top predictions. This mean accuracy does not tell the whole story however. There's a possibility that the GLM method does very well with some coaches and very poorly with others.

To explore the differences in method accuracies, we will want to use the Coach's Total Predictions table. Below is the distribution of methods for the total predictions table:

Method	**Count**	**Mean_Accuracy**
GLM	40	0.7215
DECISION TREE	40	0.7100
RANDOM FOREST	40	0.6978
SVM	40	0.6141

Interestingly enough, the total predictions table has the same order of accuracy as the best predictions table, but there is a much bigger difference for the support vector machines. Looking at the density plot of the different methods on the total predictions table in Figure 6, we can see that the support vector machine accuracy distribution is bimodal: for some coaches it does really poorly, but for other coaches, it does about as well as the other methods.

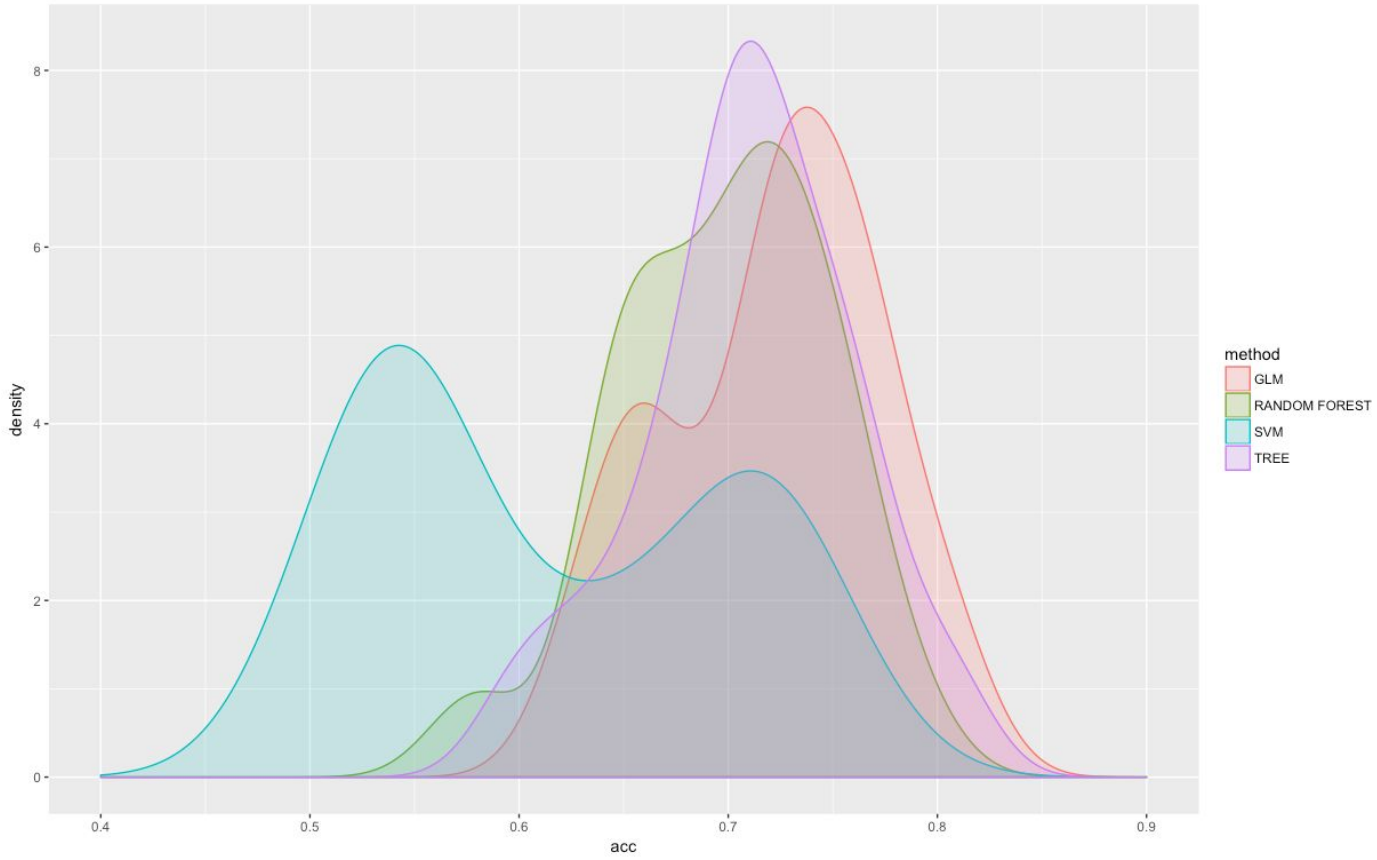

Figure 6. Density plots for the coaching accuracies based on the different machine learning methods.

Overall, when subsetting the entire dataset based on coaches, we see that generally logistic regressions had the most success in achieving the highest accuracy.

The next part of my process was to analyze the accuracies when subsetting the dataset using NFL teams rather than coaches. This process was very similar to coaching part explained above, using a for loop to generate two tables: a Team's Best Predictions table and a Team's Total Predictions table. The Team's Best Predictions table had 32 entries in it, while the Team's Total Predictions table had 128 entries in it. Each entry contains the team, the method used, and the accuracy generated on the test set.

In the Team's Best Predictions table, the highest accuracy was at 81.0% from a decision tree. This is identical to the decision tree used for Dallas's coach, Jason Garrett, because Garrett has been the main play caller for the team during the past five years. The lowest accuracy in the best predictions table is Buffalo with an accuracy of 65.5% using an SVM. The table below shows the distribution of the methods in the Team's Best Predictions table:

Method	**Count**	**Mean_Accuracy**

RANDOM FOREST	2	0.7291
DECISION TREE	14	0.7289
GLM	11	0.7248
SVM	5	0.7072

Comparing this to the Coach's Best Predictions table, decision trees seem to be the most popular methods for achieving the best accuracy. It's also interesting that the random forest had the highest mean accuracy, but only two methods had it as the best method.

Looking at the distribution of the Team's Total Predictions table, we can see that there is in fact a difference between the method accuracies in the Coach's Total Predictions table.

Method	**Count**	**Mean_Accuracy**
DECISION TREE	32	0.7125
GLM	32	0.7097
RANDOM FOREST	32	0.6924
SVM	32	0.6304

Decision trees seem to be the most accurate on a team level while SVM's again seem to struggle. Figure 7 shows the familiar bimodal distribution that we saw with SVM's in the coach's tables.

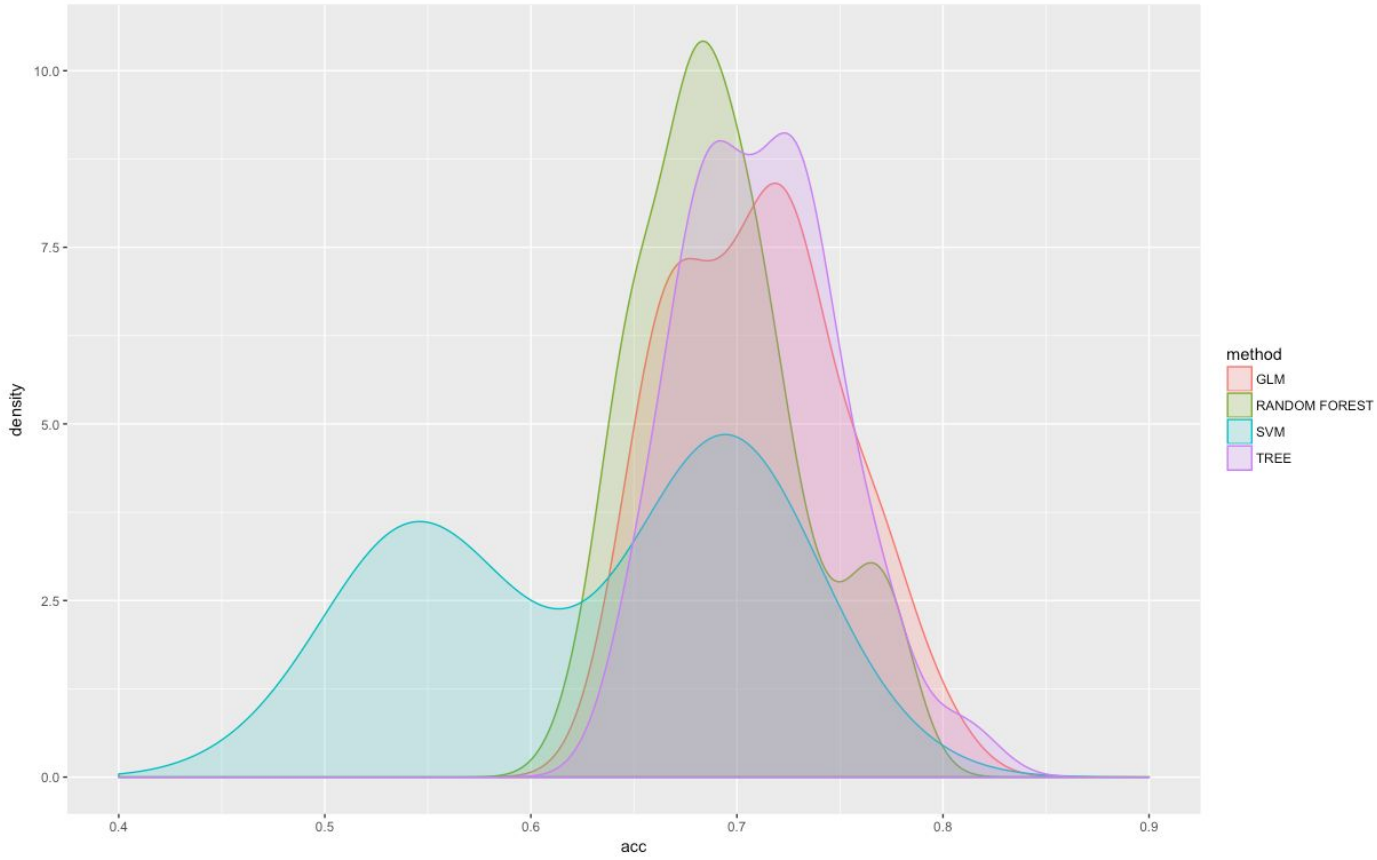

Figure 7. Density plots for the team accuracies based on the different machine learning methods.

These SVM distributions could be telling us something about the method. To inspect the SVM method, I took the lowest accuracy recorded for a team using an SVM (Tampa Bay - 47.4% accuracy), and ran the SVM kernel independently to observe if there overfitting was happening. After running the validation tests, the highest validation accuracy was about 50.2%, which doesn't seem like an overfitting problem. My assumption for the bimodal distribution is that some teams cannot be classified well using a support vector machine with the Gaussian kernel. As noted before, if there was more time and I was actually working for a team, it may be wise to look at the different kernels rather than choosing the default kernel in a project like this.

Overall, the datasets on the team level and coach level provide similar accuracy metrics. In some cases, using the team's model can give better accuracy and in other cases, the coach's model will have better accuracy. There are also cases, like Dallas and their play caller Jason Garrett as an example, that have identical models.

The final part of my analysis was to ignore both the team and coaches and finding a generalized model using the entire dataset. Again, this will be called the non-discriminatory model. I will be

using the same four methods I have used in the other two parts and will be comparing the models using t-tests to end up with one final non-discriminatory model.

First, I divided the entire dataset into a 80/20 split for training and testing purposes. I used the training data to conduct a validation test using the repeated bootstrap method, as I mentioned in the earlier part of the study. I implemented a GLM using the train function to get a validation accuracy of 72.4% (there is no tuning grid option for a GLM). I took the model created from the validation set to predict the test set and received a 72.0% out of sample accuracy with a 95% confidence interval of (71.0%, 73.0%).

The next method I used was a decision tree. I cycled through five different cp values for my tuning grid and used the rpart package to calculate my validation accuracy. The highest validation accuracy was achieved when cp = 0.01, which was used in the model to predict the test set. The out of sample accuracy from this decision tree came out to 72.1% with a 95% confidence interval of (71.1%, 73.1%).

The third method was the support vector machine with a Gaussian kernel. The tuning grid used for this method went through sigma values of 0.01 and 0.02 and C values of 0.75, 1, and 1.25. The parameters that achieved the highest validation accuracy was when sigma = 0.01 and C = 0.75. The out of sample accuracy for the support vector machine equaled 73.0% with a 95% confidence interval of (71.9%, 74.1%).

The final method I used was a random forest. The random forest generated the highest out of sample accuracy at 74.4% with a 95% confidence interval of (73.4%, 75.4%). Because the random forest's lowest accuracy interval is higher than both the decision tree and GLM's highest accuracy interval, we could assume that the random forest is a more accurate model than both of those methods. This is not the case for the SVM, however. Therefore, we need to do a t-test to compare the accuracies. Using an alpha = 0.05, the p value when comparing the two accuracies was 0.047; thus, we can reject the null hypothesis and conclude the random forest is a more accurate model than the SVM. The random forest will now be the non-discriminatory model.

The random forest in the project generated a much higher accuracy than the final model in Case Study 1 (74% to 70%). There are a few explanations for this situation. The first is that coaches may be getting more predictable with their play calling in the last few years. Recall that in the first case study, I used play-by-play data since 2009, but I'm only using data since 2012 in this project. Another reason for higher accuracy is the new variable of in_game_pass_ratio. According to the random forest, this is a very significant predictor in my overall model, and it was not used in my first case study. The variable importance plot is shown in Figure 8.

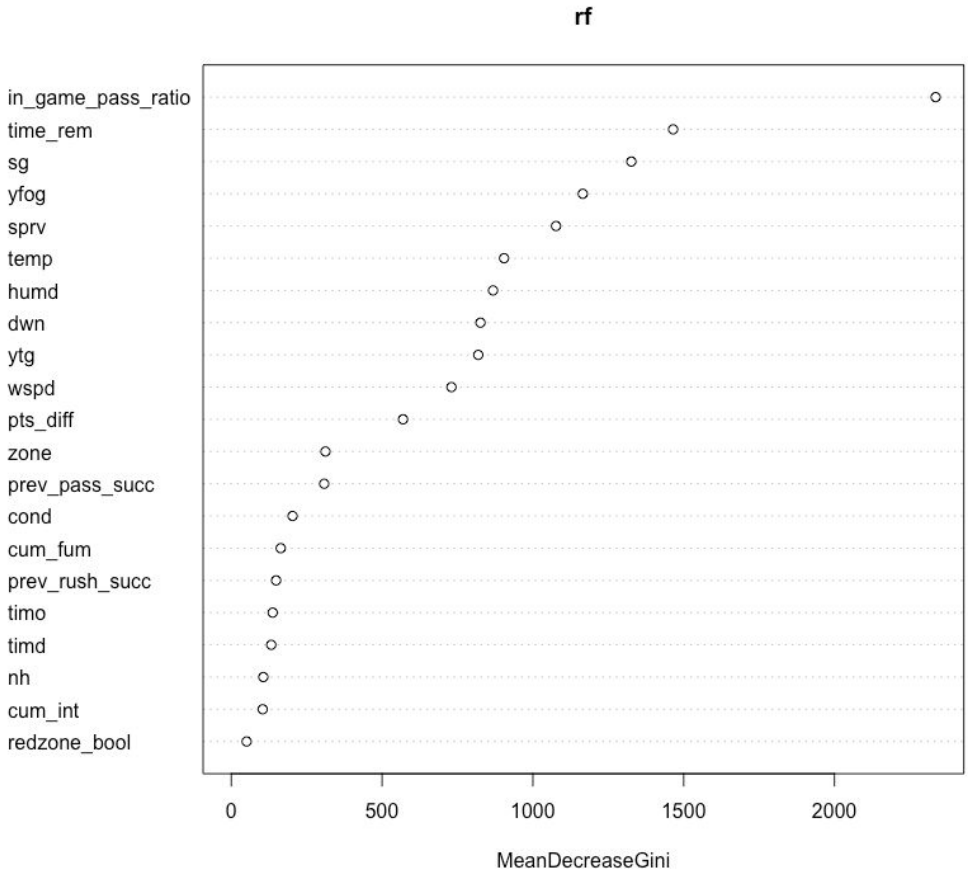

Figure 8. The variable importance plot for the overall random forest model.

We can see that some of the important variables from Case Study 1 like shotgun (sg), yards to go (ytg) and field position (yfog) are once again very important predictors for our model, but not nearly as important as in_game_pass_ratio.

Now that we have our non-discriminatory model along with each of the coach and team models, we can implement a use case of deciding which model we should use depending on the team we are facing in the upcoming week. Let's say we will be opening the 2017 season playing the San Francisco 49ers with their new head coach Kyle Shanahan. According to our coaching dataset, Kyle Shanahan has been a main play caller for the past three seasons, one season with the Cleveland Browns and two with the Atlanta Falcons. Since he is an offensive coach, we will assume he will also be the main play caller for the 49ers. Looking at the Coach's Best Prediction table, we can see the best model predicting Kyle Shanahan's play calls was a GLM that achieved a 73.7% accuracy. Figure 9 shows the output of the significant variables for Shanahan's GLM model:

```
> summary(glm_fit)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4033  -0.8563   0.3249   0.8359   2.0837

Coefficients: (1 not defined because of singularities)
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -3.1101129  1.6949434  -1.835  0.06652 .
timo               0.3993812  0.2973696   1.343  0.17926
timd              -0.2208125  0.3549491  -0.622  0.53388
dwn                0.2684744  0.1998485   1.343  0.17915
ytg                0.1413603  0.0441648   3.201  0.00137 **
yfog              -0.0112159  0.0189457  -0.592  0.55385
zone              -0.0284565  0.3602371  -0.079  0.93704
sgY                1.8675860  0.2660747   7.019 2.23e-12 ***
nhY                0.2413626  0.4346058   0.555  0.57865
cum_int           -0.2210668  0.5350667  -0.413  0.67949
cum_fum           -0.1525932  0.3928613  -0.388  0.69771
prev_pass_succ    -0.5188375  0.2975705  -1.744  0.08123 .
prev_rush_succ     0.7309685  0.3346024   2.185  0.02892 *
temp              -0.0072666  0.0093574  -0.777  0.43742
humd               0.0001369  0.0073837   0.019  0.98521
wspd               0.0022097  0.0265318   0.083  0.93362
condDome          -0.3926962  0.6561203  -0.599  0.54950
condRain           0.4301218  0.5641281   0.762  0.44579
condSnow                  NA         NA      NA       NA
sprv              -0.0308497  0.0243258  -1.268  0.20473
time_rem           0.0005484  0.0005158   1.063  0.28771
pts_diff          -0.0243574  0.0235817  -1.033  0.30165
redzone_bool      -0.3072351  0.4404215  -0.698  0.48543
in_game_pass_ratio 3.2313364  0.5614351   5.755 8.64e-09 ***
```

Figure 9. The output showing the summary of the GLM predicting Kyle Shanahan's play calls.

There are some variables that overlap with our non-discriminatory model like in_game_pass_ratio, shotgun (sgY), and yards to go (ytg). But one key difference is that if there was a previous successful rushing play (prev_rush_succ), Shanahan tends to dial up a pass play. This variable ranked relatively low in importance in the non-discriminatory model.

When inspecting the Team's Best Predictions table, we see a low out of sample accuracy value of 68.0% from San Francisco. This may be due to the fact that the 49ers have had two different regimes and three different coaching staffs in the past five years. Thus, we can disregard San Francisco's team based model because it doesn't seem to be as accurate as the model predicting Kyle Shanahan's plays or the non-discriminatory model.

Finally, we come to the final decision. Should we choose Kyle Shanahan's GLM model or our non-discriminatory random forest? The mean accuracy of the random forest is higher than the mean accuracy of the GLM. Conducting a t-test and using an alpha = 0.05, we get a p value = 0.685, which means we cannot reject the null hypothesis. Since we cannot conclude that the random forest is statistically more accurate than the GLM, we must use a subjective decision on which model to will use for the upcoming game. Because Shanahan has coached for two different teams and we were still able to get a relatively high accuracy for his play calls, I lean to using the coach based model over the non-discriminatory model. If Shanahan only was the main play caller for one team, I'd be much more hesitant to say that we will continue to coach the same exact way for another team. However, since he seems to be relatively predictable (at least very similar to our non-discriminatory model in accuracy), I believe we should use the GLM for the upcoming game against San Francisco.

Conclusion

The goal of this project was to observe a coach and team's play calling models and compare them against a non-discriminatory model, similar to one that was created in Case Study 1. In our analysis, we came up with a use case where a data scientist gathered the different models based on the upcoming coach and team and compared them to the overall non-discriminatory model. Since there wasn't a large difference in accuracy, it took a subjective judgment from the data scientist or possibly another person with domain knowledge.

For coaches, we saw that GLM's were much more accurate than the rest of the methods tested, but for teams, decision trees tended to be the most accurate method. Support vector machines tended to be hit or miss for both coaches and teams. One can conclude that since we only used the Gaussian kernel, there would be some teams and coaches where a different kernel could've registered a higher accuracy.

For the non-discriminatory model, we see a large jump in accuracy (about 74%) over the accuracy in Case Study 1 (about 70%). This can be attributed to both the inclusion of the new variable in_game_pass_ratio and the reduction of seasons in the dataset. The league seems to be getting a bit more predictable in the last five years compared to the last eight years. A random forest ended up with the highest accuracy due to significant predictors like in_game_pass_ratio, shotgun formation, yards to go, and time left in the quarter.

Something I hypothesized earlier was that some coaches and teams that had a relatively high or low pass ratio would be easier to predict. This does not seem to be the case. Dennis Allen, who was in the top 3 coaches of passing rate, had one of the worst predicted accuracies of the 40 qualified coaches. Another example is Buffalo had one of the highest rushing rates, but also one of the worst predicted accuracies among the 32 teams.

A useful test in the future for this project would be to import the 2017 play by play data after the season and go game by game to see which of these models would be the most accurate during the 2017 season. In the use case for the San Francisco 49ers, I chose to use Kyle Shanahan's model over the team model and the non-discriminatory model, but it's possible that just because the model generalizes well out-of-sample, it doesn't mean it will yield the highest accuracy with the inclusion of a new season.

Code

CreateCoachDB.R

```
library(rvest)

# The following loop takes into account the last five seasons from thehuddle.com
# It imports the head, offensive, and defensive coaches and puts them into coach_db
season = c()
team = c()
head_coach = c()
off_coach = c()
def_coach = c()
for(seas in c(2016:2012)) {
  coach <- read_html(paste0("http://thehuddle.com/",seas,"/nfl/coaching-changes.php"))
  coach_text = coach %>% html_nodes(xpath = "//tr/td") %>% html_text()

  offset = 1
  if(seas == 2016) {
    offset = 2
  } else {
    offset =1
  }

  i = 1
  if(seas < 2014) {
    i = 9
    coach_text = coach_text[1:136]
  }
  while(i < length(coach_text)) {
    season = c(season, seas)
    team = c(team, coach_text[i])
    head_coach = c(head_coach, coach_text[i+offset])
    off_coach = c(off_coach, coach_text[i+2*offset])
    def_coach = c(def_coach, coach_text[i+3*offset])
    i = i + 3*offset + 1
  }
}

coach_db = data.frame(
  seas = season,
  team = team,
  HC = head_coach,
  OC = off_coach,
  DC = def_coach,
  WeekStart = 1,
  WeekChange = NA
```

```
)

coach_db[is.na(coach_db$WeekChange),]$WeekChange = 18
```

CoachDBEdits.R

```
### The following file walks through errors that were present in the coaching database.

coach_db$HC = as.character(coach_db$HC)
coach_db$DC = as.character(coach_db$DC)
coach_db$OC = as.character(coach_db$OC)

### 2016 midsesason coaching changes
# BAL
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "BAL",
                                      HC = "John Harbaugh", OC = "Marc Trestman",
                                      DC = "Dean Pees", WeekStart = 1, WeekChange = 5))
coach_db[coach_db$seas == 2016 & coach_db$team == "BAL" & coach_db$OC == "Marty
Morinwheg",]$WeekStart = 5

# BUF
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "BUF",
                                      HC = "Rex Ryan", OC = "Greg Roman",
                                      DC = "Dennis Thurman", WeekStart = 1, WeekChange = 3))
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "BUF",
                                      HC = "Rex Ryan", OC = "Anthony Lynn",
                                      DC = "Dennis Thurman", WeekStart = 3, WeekChange = 17))
coach_db[coach_db$seas == 2016 & coach_db$team == "BUF" & coach_db$HC == "TBD",]$OC = "Chris
Palmer"
coach_db[coach_db$seas == 2016 & coach_db$team == "BUF" & coach_db$HC == "TBD",]$WeekStart =
17
coach_db[coach_db$seas == 2016 & coach_db$team == "BUF" & coach_db$HC == "TBD",]$DC = "Dennis
Thurman"
coach_db[coach_db$seas == 2016 & coach_db$team == "BUF" & coach_db$HC == "TBD",]$HC = "Anthony
Lynn"

# JAC
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "JAC",
                                      HC = "Gus Bradley", OC = "Nathaniel Hackett",
                                      DC = "Todd Wash", WeekStart = 1, WeekChange = 16))
coach_db[coach_db$seas == 2016 & coach_db$team == "JAC" & grepl("Doug Marone",
coach_db$HC),]$WeekStart = 16

# LA
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "LA",
                                      HC = "Jeff Fisher", OC = "Rob Boras",
                                      DC = "Gregg Williams", WeekStart = 1, WeekChange = 13))
```

```
coach_db[coach_db$seas == 2016 & coach_db$team == "LA" & grepl("Jim Fassel",
coach_db$HC),]$WeekStart = 13

# MIN
coach_db = rbind(coach_db, data.frame(seas = 2016, team = "MIN",
                                      HC = "Mike Zimmer", OC = "Norv Turner",
                                      DC = "George Edwards", WeekStart = 1, WeekChange = 9))
coach_db[coach_db$seas == 2016 & coach_db$team == "MIN" & grepl("Pat Shurmur",
coach_db$OC),]$WeekStart = 9

### 2015 Midseason coaching changes

# DET
coach_db = rbind(coach_db, data.frame(seas = 2015, team = "DET",
                                      HC = "Jim Caldwell", OC = "Jim Bob Cooter",
                                      DC = "Teryl Austin", WeekStart = 8, WeekChange = NA))
coach_db[coach_db$seas == 2015 & coach_db$team == "DET" & grepl("Joe Lombardi",
coach_db$OC),]$WeekChange = 8

# IND
coach_db = rbind(coach_db, data.frame(seas = 2015, team = "IND",
                                      HC = "Chuck Pagano", OC = "Rob Chudzinski",
                                      DC = "Greg Manusky", WeekStart = 9, WeekChange = NA))
coach_db[coach_db$seas == 2015 & coach_db$team == "IND" & grepl("Pep Hamilton",
coach_db$OC),]$WeekChange = 9

# MIA
coach_db = rbind(coach_db, data.frame(seas = 2015, team = "MIA",
                                      HC = "Dan Campbell", OC = "Bill Lazor",
                                      DC = "Kevin Coyle", WeekStart = 5, WeekChange = 13))
coach_db = rbind(coach_db, data.frame(seas = 2015, team = "MIA",
                                      HC = "Dan Campbell", OC = "Zac Taylor",
                                      DC = "Kevin Coyle", WeekStart = 13, WeekChange = NA))
coach_db[coach_db$seas == 2015 & coach_db$team == "MIA" & grepl("Joe Philbin",
coach_db$HC),]$WeekChange = 5

# PHI
coach_db = rbind(coach_db, data.frame(seas = 2015, team = "PHI",
                                      HC = "Pat Shurmur", OC = "Pat Shurmur",
                                      DC = "Bill Davis", WeekStart = 17, WeekChange = NA))
coach_db[coach_db$seas == 2015 & coach_db$team == "PHI" & grepl("Chip Kelly",
coach_db$HC),]$WeekChange = 17


### 2014 Midseason Coaching Changes
# OAK
coach_db = rbind(coach_db, data.frame(seas = 2013, team = "OAK",
                                      HC = "Tony Sparano", OC = "Greg Olson",
                                      DC = "Jason Tarver", WeekStart = 5, WeekChange = NA))
```

```
coach_db[coach_db$seas == 2013 & coach_db$team == "OAK" & grepl("Dennis Allen",
coach_db$HC),]$WeekChange = 5

### 2013 Midseason Coaching Changes
# HOU
coach_db = rbind(coach_db, data.frame(seas = 2013, team = "HOU",
                                      HC = "Wade Phillips", OC = "Rick Dennison",
                                      DC = "Wade Phillips", WeekStart = 14, WeekChange = NA))
coach_db[coach_db$seas == 2013 & coach_db$team == "HOU" & grepl("Wade Phillips",
coach_db$DC),]$HC = "Gary Kubiak"
coach_db[coach_db$seas == 2013 & coach_db$team == "HOU" & grepl("Gary Kubiak",
coach_db$HC),]$WeekChange = 14

coach_db = coach_db[order(coach_db$seas, coach_db$team, coach_db$WeekStart),]

coach_db$HC = ifelse(regexpr('\\(', coach_db$HC) == -1, coach_db$HC, substr(coach_db$HC, 1,
regexpr('\\(', coach_db$HC)-2))
coach_db$OC = ifelse(regexpr('\\(', coach_db$OC) == -1, coach_db$OC, substr(coach_db$OC, 1,
regexpr('\\(', coach_db$OC)-2))
coach_db$DC = ifelse(regexpr('\\(', coach_db$DC) == -1, coach_db$DC, substr(coach_db$DC, 1,
regexpr('\\(', coach_db$DC)-2))

coach_db[coach_db$seas == 2016 & coach_db$team == "CLE" & grepl("Hue Jackson",
coach_db$HC),]$OC = "Pep Hamilton"
coach_db[coach_db$seas == 2014 & coach_db$team == "HOU" & grepl("Bill O'Brien",
coach_db$HC),]$OC = "Bill O'Brien"

# MISSPELLINGS
coach_db[coach_db$HC == "Doug Marone",]$HC = "Doug Marrone"
coach_db[coach_db$HC == "Doug Maronne",]$HC = "Doug Marrone"
coach_db[grepl("Marty", coach_db$OC),]$OC = "Marty Mornhinweg"
coach_db[coach_db$DC == "open",]$DC = "Romeo Crennel"
coach_db[coach_db$HC == "Jim Fassel", ]$HC = "John Fassel"

coach_db[is.na(coach_db$WeekChange),]$WeekChange = 18
```

CoachPredict.R

```
library(readr)
library(plyr)
library(dplyr)
library(sqldf)
library(party)
library(caret)
library(randomForest)

# Imports the datasets from Armchair Analysis
PBP <- read_csv("~/Desktop/nfl_00-16/PBP.csv")
```

```
game <- read_csv("~/Desktop/Football/nfl_00-16/GAME.csv")

# Edits the values of fumbles/interceptions
PBP$ints[!is.na(PBP$ints)] = 1
PBP$ints[is.na(PBP$ints)] = 0
PBP$fum[!is.na(PBP$fum)] = 1
PBP$fum[is.na(PBP$fum)] = 0

# Merges the new variables onto pbp
PBP = PBP %>% group_by(gid, off) %>% mutate(cum_int = cumsum(ints), cum_fum = cumsum(fum))
pbp = merge(PBP[,c(1:2,4:25,85:86)], game[,1:3], by = "gid")
pbp = pbp[,c(1:2,27:28,3:26)]

# Crops the pbp dataset into rush/pass
pbp = pbp[pbp$type == "RUSH" | pbp$type == "PASS",]

# Merges the coach dataset on the play-by-play set
pbp <- sqldf("SELECT coach_db.HC, coach_db.OC, coach_db.HC_playcaller, pbp.* FROM pbp,
coach_db WHERE pbp.off = coach_db.team AND pbp.seas = coach_db.seas AND pbp.wk <
coach_db.WeekChange AND pbp.wk >= coach_db.WeekStart")

# Edits some of the coaching formats
pbp[is.na(pbp)] = 0
pbp$PlayCaller = ifelse(pbp$HC_playcaller == 1, pbp$HC, pbp$OC)
pbp$HC = trim(pbp$HC)
pbp$OC = trim(pbp$OC)
pbp$HC_playcaller = NULL
pbp$PlayCaller = trim(pbp$PlayCaller)
pbp = pbp[,c(ncol(pbp), 1:(ncol(pbp)-1))]

# Edits the condition variable to only have 4 factors
game = game[,c(1:3, 6, 8:10, 12, 15)]
levels(as.factor(game$cond))
game$cond = ifelse(grepl("Roof", game$cond), "Dome", game$cond)
game$cond = ifelse(grepl("Rain", game$cond) | grepl("Showers", game$cond) |
grepl("Thunderstorms", game$cond), "Rain", game$cond)
game$cond = ifelse(grepl("Snow", game$cond) | grepl("Flurries", game$cond), "Snow", game$cond)
game$cond = ifelse(game$cond == "Dome" | game$cond == "Rain" | game$cond == "Snow", game$cond,
"Clear")
game$cond[is.na(game$cond)] = "Clear"
game$cond = as.factor(game$cond)

# Creates the successful plays
pbp_mod = pbp
pbp_mod$prev_succ <- c(0, pbp_mod$succ[-nrow(pbp_mod)])
pbp_mod$prev_succ = ifelse(pbp_mod$dseq == 1, 0, pbp_mod$prev_succ)
pbp_mod$prev_type = c("PASS", pbp_mod$type[-nrow(pbp_mod)])
pbp_mod$prev_pass_succ = ifelse(pbp_mod$prev_type == "PASS" & pbp_mod$prev_succ == "Y", 1, 0)
pbp_mod$prev_rush_succ = ifelse(pbp_mod$prev_type == "RUSH" & pbp_mod$prev_succ == "Y", 1, 0)
pbp_mod$prev_succ = NULL
```

```
pbp_mod$prev_type = NULL

# Joins the game dataset onto the play-by-play dataset, removes some variables
pbp_game_info = left_join(pbp_mod, game, by = c("gid", "seas", "wk"))
pbp_game_info$sprv = ifelse(pbp_game_info$off == pbp_game_info$h, pbp_game_info$sprv * -1,
pbp_game_info$sprv)
pbp_game_info$h = NULL
pbp_game_info$pts = NULL
pbp_game_info$fd = NULL
pbp_game_info$yds = NULL
pbp_game_info$succ = NULL
pbp_game_info$time_rem = pbp_game_info$min * 60 + pbp_game_info$sec
pbp_game_info$min = NULL
pbp_game_info$sec = NULL

# Creates the target variable.
pbp_game_info$pass_bool = ifelse(pbp_game_info$type == "PASS", 1, 0)
pbp_game_info$pts_diff = pbp_game_info$ptso - pbp_game_info$ptsd
pbp_game_info$redzone_bool = ifelse(pbp_game_info$zone == 5, 1, 0)
pbp_game_info$type = NULL
pbp_game_info$ptsd = NULL
pbp_game_info$ptso = NULL
pbp_game_info$PlayCaller = as.factor(pbp_game_info$PlayCaller)

# Creates an in-game passing ratio
pbp_game_info$pass = ifelse(pbp_game_info$passnum == 0, 0, pbp_game_info$passnum - 1)
pbp_game_info$play = ifelse(pbp_game_info$playnum == 1, 1, pbp_game_info$playnum - 1)
pbp_game_info$in_game_pass_ratio = round(pbp_game_info$pass/pbp_game_info$play, 3)

game_pass_ratios = pbp_game_info %>% group_by(seas, gid, off) %>% summarise(ratio =
mean(pass_bool))
game_pass_ratios = game_pass_ratios %>% group_by(off, seas) %>% mutate(gnum = row_number())
game_pass_ratios = sqldf('select a.*, b.ratio as prev_game_pr from game_pass_ratios a,
game_pass_ratios b where a.seas = b.seas and a.off = b.off and a.gnum = b.gnum + 1')
game_pass_ratios = game_pass_ratios[,c(2:3,6)]

pbp_game_info$passnum = NULL
pbp_game_info$playnum = NULL
pbp_game_info$pass = NULL
pbp_game_info$play = NULL

### For this for loop, we will go track GLM, DT's, RF's, and SVM's for each of the different
coaches
### From there, we add them into two separate dataframes: accs_fac and coach_pred_df
accs_fac = data.frame()
coaches = c()
coach_pred_df = data.frame(qtr = c(), coach = c(), accuracy = c(), n = c(), method = c())
for(coach in levels(pbp_game_info$PlayCaller)) {
  if (coach %in% non_coaches) {
    next
```

```
  }
  for(quarter in 1:1) {
    coach_pbp = pbp_game_info[pbp_game_info$PlayCaller == coach & pbp_game_info$qtr ==
1,c(13:ncol(pbp_game_info))]
    coach_pbp[is.na(coach_pbp)] = 0
    coach_pbp$sg = as.factor(coach_pbp$sg)
    coach_pbp$nh = as.factor(coach_pbp$nh)

    set.seed(123)
    train_ind = sample(seq_len(nrow(coach_pbp)), size = 0.8 * nrow(coach_pbp))
    train = coach_pbp[train_ind,]
    test = coach_pbp[-train_ind,]
    train_control = trainControl(method = "boot", number = 100)
    print(paste(coach, nrow(coach_pbp)))
    if(nrow(coach_pbp) >= 260) {
      print("GLM")
      glm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control, method
= "glm")
      pred_glm = predict(glm_fit, test)
      tbl_glm = table(test$pass_bool, pred_glm)
      acc_glm = sum(diag(tbl_glm))/sum(tbl_glm)

      tree_grid = expand.grid(cp = c(0.01, 0.02, 0.03, 0.04, 0.05))
      print("Tree")
      tree_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control,
tuneGrid = tree_grid, method = "rpart")
      pred_tree = predict(tree_fit, test)
      tbl_tree = table(test$pass_bool, pred_tree)
      acc_tree = sum(diag(tbl_tree))/sum(tbl_tree)

      svm_grid = expand.grid(sigma = c(0.01, 0.15, 0.02),
                             C = c(0.75, 1, 1.25, 1.5))
      print("SVM")
      svm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control,
tuneGrid = svm_grid, method = "svmRadial")
      pred_svm = predict(svm_fit, test)
      tbl_svm = table(test$pass_bool, pred_svm)
      acc_svm = sum(diag(tbl_svm))/sum(tbl_svm)

      print("RF")
      rf = randomForest(as.factor(pass_bool) ~ ., data = train,
controls=cforest_unbiased(ntree=1000, mtry=3))
      pred_mod_rf = predict(rf, test)
      s = table(test$pass_bool, pred_mod_rf)
      acc_rf = sum(diag(s))/sum(s)

      accuracies = data.frame(method = c("GLM", "SVM", "TREE", "RANDOM FOREST"),
                              acc = c(acc_glm, acc_svm, acc_tree, acc_rf))
      accuracies = accuracies[order(-accuracies$acc),]
```

```
      accs_fac = rbind(accs_fac, data.frame(acc = accuracies$acc, method = accuracies$method,
coach = coach))
      print(paste(coach, quarter, accuracies$method[1], accuracies$acc[1], nrow(coach_pbp)))
      coaches = c(coaches, coach)
      coach_pred_df = rbind(coach_pred_df, data.frame(qtr = quarter, coach = coach, accuracy =
accuracies$acc[1], n = nrow(coach_pbp), method = accuracies$method[1]))
    }
  }
}
```

TeamPredict.R

```
team_accs = data.frame()
teams = c()
team_pred_df = data.frame(qtr = c(), team = c(), accuracy = c(), n = c(), method = c())

pbp_game_info$off = ifelse(pbp_game_info$off == "STL", "LA", pbp_game_info$off)

### Similar for loop as in CoachPredict.R, however it creates two different dataframes:
team_accs and team_pred_df
for(team in levels(as.factor(pbp_game_info$off))) {
  for(quarter in 1:1) {
    team_pbp = pbp_game_info[pbp_game_info$off == team & pbp_game_info$qtr ==
1,c(13:ncol(pbp_game_info))]
    team_pbp[is.na(team_pbp)] = 0
    team_pbp$sg = as.factor(team_pbp$sg)
    team_pbp$nh = as.factor(team_pbp$nh)

    set.seed(123)
    train_ind = sample(seq_len(nrow(team_pbp)), size = 0.8 * nrow(team_pbp))
    train = team_pbp[train_ind,]
    test = team_pbp[-train_ind,]
    train_control = trainControl(method = "boot", number = 100)
    print(paste(team, nrow(team_pbp)))
    if(nrow(team_pbp) >= 260) {
      print("GLM")
      glm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control, method
= "glm")
      pred_glm = predict(glm_fit, test)
      tbl_glm = table(test$pass_bool, pred_glm)
      acc_glm = sum(diag(tbl_glm))/sum(tbl_glm)

      tree_grid = expand.grid(cp = c(0.01, 0.02, 0.03, 0.04, 0.05))
      print("Tree")
      tree_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control,
tuneGrid = tree_grid, method = "rpart")
      pred_tree = predict(tree_fit, test)
      tbl_tree = table(test$pass_bool, pred_tree)
      acc_tree = sum(diag(tbl_tree))/sum(tbl_tree)
```

```
      svm_grid = expand.grid(sigma = c(0.01, 0.15, 0.02),
                             C = c(0.75, 1, 1.25, 1.5))
      print("SVM")
      svm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control,
tuneGrid = svm_grid, method = "svmRadial")
      pred_svm = predict(svm_fit, test)
      tbl_svm = table(test$pass_bool, pred_svm)
      acc_svm = sum(diag(tbl_svm))/sum(tbl_svm)

      print("RF")
      rf = randomForest(as.factor(pass_bool) ~ ., data = train,
controls=cforest_unbiased(ntree=1000, mtry=3))
      pred_mod_rf = predict(rf, test)
      s = table(test$pass_bool, pred_mod_rf)
      acc_rf = sum(diag(s))/sum(s)

      accuracies = data.frame(method = c("GLM", "SVM", "TREE", "RANDOM FOREST"),
                              acc = c(acc_glm, acc_svm, acc_tree, acc_rf))
      accuracies = accuracies[order(-accuracies$acc),]

      team_accs = rbind(team_accs, data.frame(acc = accuracies$acc, method =
accuracies$method, team = team))
      print(paste(team, quarter, accuracies$method[1], accuracies$acc[1], nrow(team_pbp)))
      teams = c(teams, team)
      team_pred_df = rbind(team_pred_df, data.frame(qtr = quarter, team = team, accuracy =
accuracies$acc[1], n = nrow(team_pbp), method = accuracies$method[1]))
    }
  }
}
```

OverallPredict.R

```
### This is a similar process as the other two predict files except taking in the entire
dataset.
ovr_pbp = pbp_game_info[pbp_game_info$qtr == 1,c(13:ncol(pbp_game_info))]
ovr_pbp[is.na(ovr_pbp)] = 0
ovr_pbp$sg = as.factor(ovr_pbp$sg)
ovr_pbp$nh = as.factor(ovr_pbp$nh)

set.seed(123)
train_ind = sample(seq_len(nrow(ovr_pbp)), size = 0.8 * nrow(ovr_pbp))
train = ovr_pbp[train_ind,]
test = ovr_pbp[-train_ind,]
train_control = trainControl(method = "boot", number = 100)

print("GLM")
glm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control, method =
"glm")
```

```
glm_fit
pred_glm = predict(glm_fit, test)
tbl_glm = table(test$pass_bool, pred_glm)
acc_glm = sum(diag(tbl_glm))/sum(tbl_glm)
s_glm = sqrt(sum((as.numeric(pred_glm)-1 - mean(as.numeric(pred_glm)-1))^2)/(nrow(test) - 1))

tree_grid = expand.grid(cp = c(0.01, 0.02, 0.03, 0.04, 0.05))
print("Tree")
tree_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control, tuneGrid =
tree_grid, method = "rpart")
pred_tree = predict(tree_fit, test)
tbl_tree = table(test$pass_bool, pred_tree)
acc_tree = sum(diag(tbl_tree))/sum(tbl_tree)
s_tree = sqrt(sum((as.numeric(pred_tree)-1 - mean(as.numeric(pred_tree)-1))^2)/(nrow(test) -
1))

svm_grid = expand.grid(sigma = c(0.01, 0.02),
                       C = c(0.75, 1, 1.25))
print("SVM")
svm_fit = train(as.factor(pass_bool)~., data = train, trControl = train_control, method =
"svmLinear")
pred_svm = predict(svm_fit, test)
tbl_svm = table(test$pass_bool, pred_svm)
acc_svm = sum(diag(tbl_svm))/sum(tbl_svm)

print("RF")
rf = randomForest(as.factor(pass_bool) ~ ., data = train,
controls=cforest_unbiased(ntree=1000, mtry=3))
pred_mod_rf = predict(rf, test)
s = table(test$pass_bool, pred_mod_rf)
confusionMatrix(s)
acc_rf = sum(diag(s))/sum(s)
s_rf = sqrt(sum((as.numeric(pred_mod_rf)-1 - mean(as.numeric(pred_mod_rf)-1))^2)/(nrow(test) -
1))

t_test = function(a1, a2, s1, s2, N) {
  t = (as.numeric(a1) - as.numeric(a2))/(sqrt((s1^2)/N+(s2^2)/N))
  return(t)
}

varImpPlot(rf)
accuracies = data.frame(method = c("GLM", "SVM", "TREE", "RANDOM FOREST"),
                        acc = c(acc_glm, acc_svm, acc_tree, acc_rf))
accuracies = accuracies[order(-accuracies$acc),]
```

Graphs.R

```
library(ggplot2)
```

```
### File for my exploratory analysis and process graphs.

off_perc = pbp_game_info %>% group_by(off) %>% summarise(pass = sum(pass_bool == 1), run =
sum(pass_bool == 0))
off_perc_t = t(off_perc[,-1])
colnames(off_perc_t) = off_perc$off

off_perc_melt = melt(cbind(off_perc_t), ind = rownames(off_perc_t), id.vars = c('ind'))

colnames(off_perc_melt) = c('type', 'team', 'percent')

ggplot(off_perc_melt, aes(x = team, y = percent, fill = factor(type, levels = c('run',
'pass')))) + geom_bar(position = 'fill', stat = 'identity') + labs(fill = "Play Type")

coach_perc = pbp_game_info %>% group_by(PlayCaller) %>% summarise(pass = sum(pass_bool == 1),
run = sum(pass_bool == 0))
coach_perc_t = t(coach_perc[,-1])
colnames(coach_perc_t) = coach_perc$PlayCaller

coach_perc_melt = melt(cbind(coach_perc_t), ind = rownames(coach_perc_t), id.vars = c('ind'))

colnames(coach_perc_melt) = c('type', 'coach', 'percent')

pbp_game_info %>% filter(in_game_pass_ratio <= 0) %>% group_by(pass_bool) %>% summarise(n =
n()) %>% mutate(ratio = n/sum(n))

ggplot(coach_perc_melt, aes(x = coach, y = percent, fill = factor(type, levels = c('run',
'pass')))) + geom_bar(position = 'fill', stat = 'identity') + theme(axis.text.x =
element_text(angle = 45, hjust = 1)) + labs(fill = "Play Type")

library(corrplot)
colnames(pbp_game_info)

for(i in 1:ncol(pbp_game_info)) {
  print(paste(i, colnames(pbp_game_info)[i], class(pbp_game_info[,i])))
}

lapply(pbp_game_info, class)

corrplot(cor(pbp_game_info[,c(30,16,17,21,22,28:29,31,33)]), method = 'ellipse')

library(grid)
library(gridExtra)

g1 = ggplot(pbp_game_info, aes(dwn, ..count..)) + geom_bar(aes(fill = as.factor(pass_bool)),
position = "dodge") + labs(title = "Down", fill = "pass_bool")

g2 = ggplot(pbp_game_info, aes(cond, ..count..)) + geom_bar(aes(fill = as.factor(pass_bool)),
position = "dodge") + labs(title = "Weather Condition", fill = "pass_bool")
```

```
g3 = ggplot(pbp_game_info, aes(as.factor(sg), ..count..)) + geom_bar(aes(fill =
as.factor(pass_bool)), position = "dodge") + labs(title = "Shotgun", subtitle = "(1 for yes, 0
for no)", fill = "pass_bool")

g4 = ggplot(pbp_game_info, aes(as.factor(redzone_bool), ..count..)) + geom_bar(aes(fill =
as.factor(pass_bool)), position = "dodge") + labs(title = "Redzone", subtitle = "(1 for yes, 0
for no)", fill = "pass_bool")

g5 = ggplot(pbp_game_info, aes(as.factor(prev_pass_succ), ..count..)) + geom_bar(aes(fill =
as.factor(pass_bool)), position = "dodge") + labs(title = "Previous Pass Play Success",
subtitle = "(1 for yes, 0 for no)", fill = "pass_bool")

g6 = ggplot(pbp_game_info, aes(as.factor(prev_rush_succ), ..count..)) + geom_bar(aes(fill =
as.factor(pass_bool)), position = "dodge") + labs(title = "Previous Rush Play Success",
subtitle = "(1 for yes, 0 for no)", fill = "pass_bool")

grid.arrange(g1,g2,g3,g4,g5,g6,nrow = 3)

ggplot(data = accs_fac, aes(acc, fill = method, color = method)) + geom_density(alpha = 0.2) +
xlim(0.4, 0.9)

ggplot(data = team_accs, aes(acc, fill = method, color = method)) + geom_density(alpha = 0.2)
+ xlim(0.4, 0.9)
```

Works Cited

Orr, Connor. “NFL's Most, Least Balanced Offensive Play-Callers.” *NFL.com*, www.nfl.com/news/story/0ap3000000816875/article/nfls-most-least-balanced-offensive-play callers.

Erny, Dennis. “Armchair Analysis.com.” *NFL Play Data. 750,000+ Plays. Daily Updates. Armchair Analysis.com*, armchairanalysis.com/data.php.

Jones, Willie D. “Model Predicts Whether NFL Teams Will Run or Pass.” *IEEE Spectrum: Technology, Engineering, and Science News*, 13 Aug. 2015, spectrum.ieee.org/tech-talk/geek-life/tools-toys/statistical-model-predicts-whether-nfl-teams-will-run-or-pass.

The Huddle, 19 Nov. 2017, http://thehuddle.com/2017/01/18/2017-coaching-change-tracker/