 # NFL Plays Analytics - Final

Predicting offensive football plays

The NFL is the most profitable major sports league in the US with over $12 billion in revenue last season. However, NFL coaches and general managers are woefully behind many other leagues in their utilization of advanced analytics to enhance in-game decisions and player acquisitions. With this project I would seek to give teams a leg up on the competition by building a model to predict the opponent's next offensive play, or at least narrow down the possibilities. Thus, coaches of NFL teams are the directed client. With these results computed historically and in real-time, coaches could call better defensive plays, adjust their personnel, and improve their strategy - giving them a greater likelihood of hindering the opponent's success.

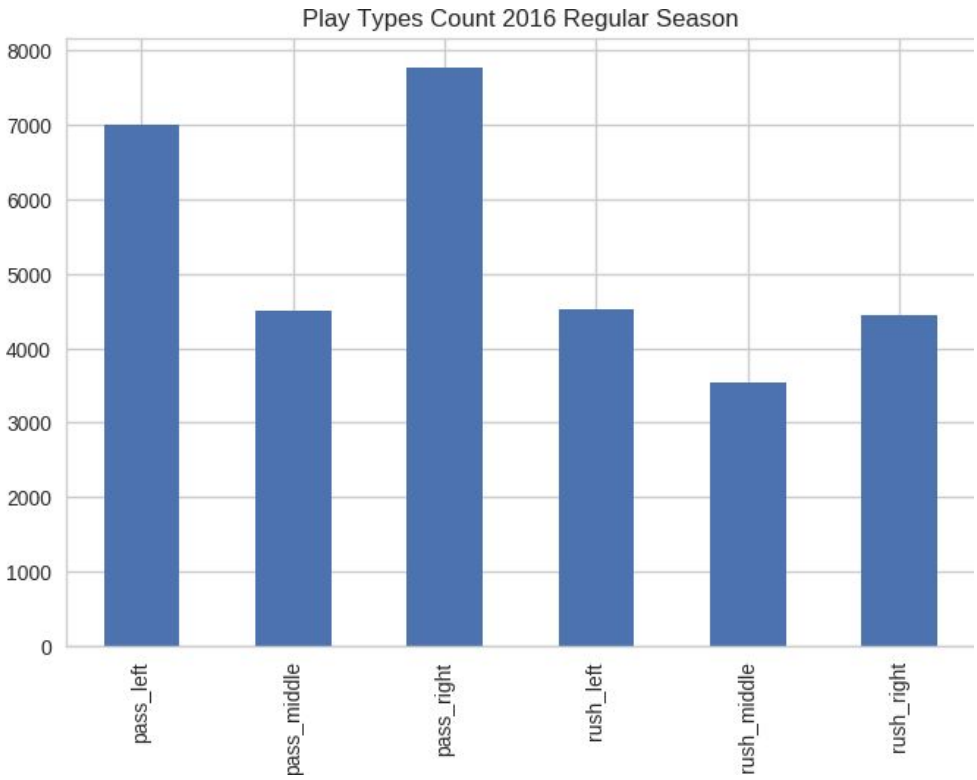
The main open-source dataset can be found here, nfldb; it is a Postgresql relational database containing play-by-play, game scores, player stats, schedules, and much more from the 2009-2016 seasons and playoffs. There are 8 tables in total in the database (team, game, drive, play, player, agg_play, agg_player).

 # Data Wrangling (see plays_wrangle.ipynb)

I explored the various tables with SQL queries to better understand the scope of the data available, and determined that I would directly query data from 3 tables - plays, games, and drives table - which contained object pointers to the other tables in order to access team, play statistics, and players. The database includes a small querying module, based on SQL, which I used to efficiently retrieve tables, returned as objects. I wrote custom functions utilizing Python list comprehensions to select non-overlapping attributes, accessed with the dir() and getattr() Python functions, from each object within each table. I collected the selected attribute data into a Python list of lists and stored the queried tables in separate pandas dataframes which shared game and drive keys, useful for merging later. To reduce the influence of season to season team changes, I focused on the 2016 Regular season data for every team.

To obtain labels for play type and direction, I searched each plays text description for combinations of pass vs rush and left vs middle vs right, using string manipulation via regular expressions. By iterating over custom play type and direction string vectors, I constructed complex regular expressions to recognize phrases beginning with any player's name followed by a given play type and direction with any number of words and/or spaces in between. I then used pandas "str" and "contains" methods to create a

row mask matching the given regular expression. Using pandas ".loc" method along with the mask, I labeled each row with the appropriate target label through string concatenation of the current play type and direction. This wrangling was used to produce my target variable and provided ~32,000 plays. Resulting play count totals are plotted below.



Using pandas "merge" function, I then left merged the dataframes plays and drives based on the drive ID key. With custom Python code, I performed integrity checks to find and remove any duplicate columns, ensure no plays contained multiple overlapping target labels, and examine play descriptions that did not receive a target label to ensure there was not a fundamental flaw in the constructed regular expressions. In the process, several plays and play types with insufficient data or outliers were excluded, including kicks, punts, two-point conversions, illegal forward passes, and quarterback scrambles. This reduced dataframe was created through boolean row selection, using pandas "notnull()" method on the play target column. The resulting dataframe was then left merged with the games dataframe based on the game ID key, producing a 100 column dataframe.

I created separate play type and direction label columns with pandas "str.split" method on the play label. Using pandas "get_dummies" function, I constructed dummy vectors

for play type and direction then combined them with the overall dataframe using pandas "concat" function. These vectors will be used for producing cumulative counts of play type and direction by team, game, and drive. Score tuples (i.e., home, away scores), game clock quarter and time, and variety of other categorical variables with string representations were split and converted to integers, category codes, dummy vectors, and/or time floats as necessary. In the above wrangling, I utilized pandas "datetime" methods and functionality. I first scaled yardlines from 1-99, where yardline 49-1 on one's own side of the field are 51-99, reflecting the fact that one is farther away from the endzone to be scored on. Using pandas "set_index()" and "sortlevel()" methods, I transformed the dataframe index into a multilevel index of game, offensive team, and drive. With list comprehensions on the dataframe column names, I created a list of the numeric column names. Then, using pandas "groupby()" and "cumsum()" methods, I accumulated these statistics separately across drive possessions (collections of plays) and games for each team and game. I shifted all statistics forward for each play to prevent leaking future information into the model for a given play. I zero-filled the resulting missing data for the beginning of drives and forward-filled stats for cumulative game stats. The newly concatenated dataframe contained 246 columns.

Following processing plays, I merged this dataframe with general game and drive data that was queried from the database. These merges provided additional data about home vs away teams, day of the week, week of the season, and summary stats from previous play drives. With the current wrangling setup, ~32,000 plays and ~200 features emerged.

Statistical Inference (for plots of results, see plays_compare_stats_infer.ipynb)

After wrangling the NFL plays data into a suitable form and exploring patterns in the data (see plays_explore_plots.ipynb), I performed inferential statistics to examine differences and relationships between and within the feature (predictor) and target (dependent) variables. The dependent variable type is categorical with 6-classes composed of play type and direction: rush vs. pass and left vs. middle vs. right.

Initially, I examined whether play call is dependent on the team's current location on field. I first scaled yardlines from 1-99, where yardline 49-1 on one's own side of the field are 51-99, reflecting the fact that one is farther away from the endzone to be scored on. With a one-way ANOVA comparing mean yardline across the play classes, I found an overall significant difference across plays (F=2.5, p=0.028, η^2=0.0004). Follow-up pairwise Tukey comparisons, found that only the mean yardline of rush middle significantly differed from pass middle and left, indicating rushes up the middle

have a lower average yardline or are slightly more likely as a team approaches the 1 yardline. Nonetheless, play types nor directions were dissociated across yardline.

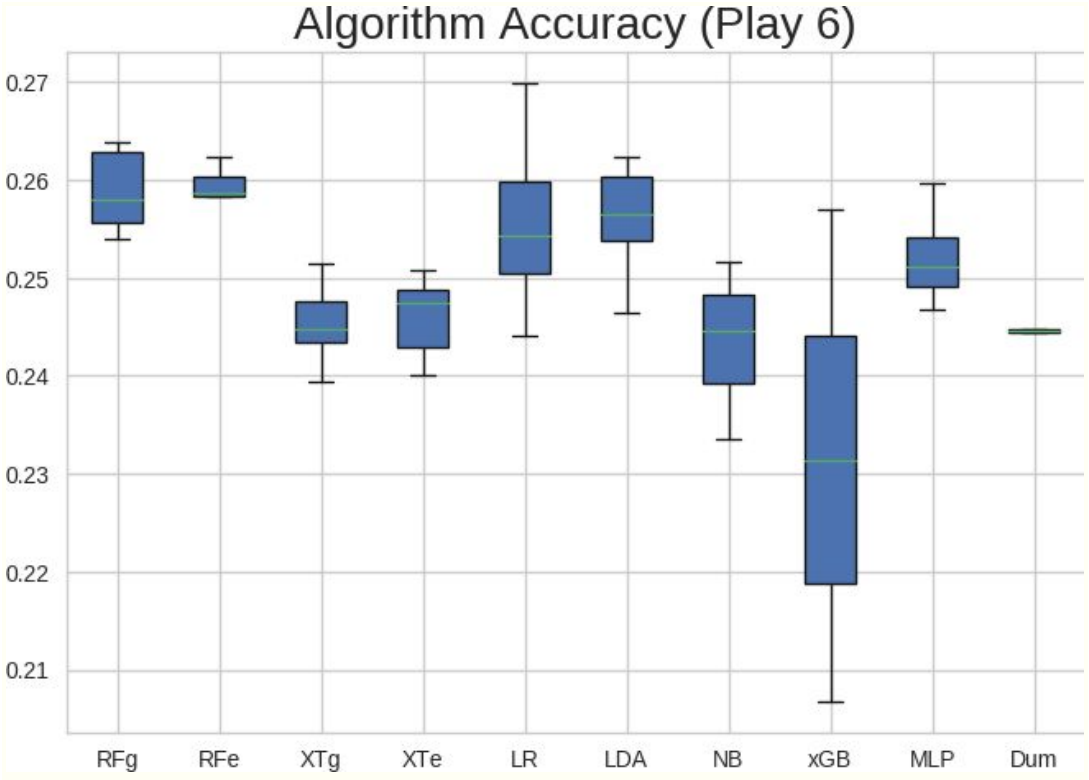
In football, plays are run on four separate downs, where the goal is to gain the desired number of yards in order to get a new 1st down before 4th down. First down begins with 10 yards to gain (or go) and the number of yards an offense gains or loses on each play is subtracted or added to the yards to go. Offenses typically punt the ball to the opponent on 4th down, so as 4th down approaches it becomes more imperative to gain the full yards to go. Accordingly, I tested whether play class depended on the current down. Because both of these variables are categorical I used a chi-square test of independence, which tests if the proportion of play classes is uniformly distributed across downs. I found play was significantly dependent upon down (χ^2=1997.95, p<0.001), with more pass plays called on 3rd down and fewer plays were directed over the middle generally.

Using a one-way ANOVA comparing mean yards to gain with play class, I found that play significantly depended upon yards to gain (F=58.34, p<0.001, η^2=0.009). Importantly, the eta-squared (η^2) value is larger than that for yardline, reflecting yards to go has a bigger impact on play than the current yardline. Follow-up comparisons found significant differences with pass > rush, pass middle > all except pass left, and rush middle < all except rush left with respect to mean yards to go. Thus, pass plays are more likely when the yards to gain is larger. Further, middle passes occurred more frequently with long yards to gain, whereas middle rushes occurred more with short yards to gain. In a subsequent one-way ANOVA, play class was significantly dependent upon the mean score difference between teams (F=166.1, p<0.001, η^2=0.025), with a larger effect size than either above. Follow-up comparisons found significant differences with rush > pass, pass middle < all except pass left, and rush left > all except rush right with respect to mean score difference. Note score difference scale ranges from negative to positive. Hence, a team is more likely to pass when losing, and middle passes and rushes have a lower mean score difference than their outside counterparts.

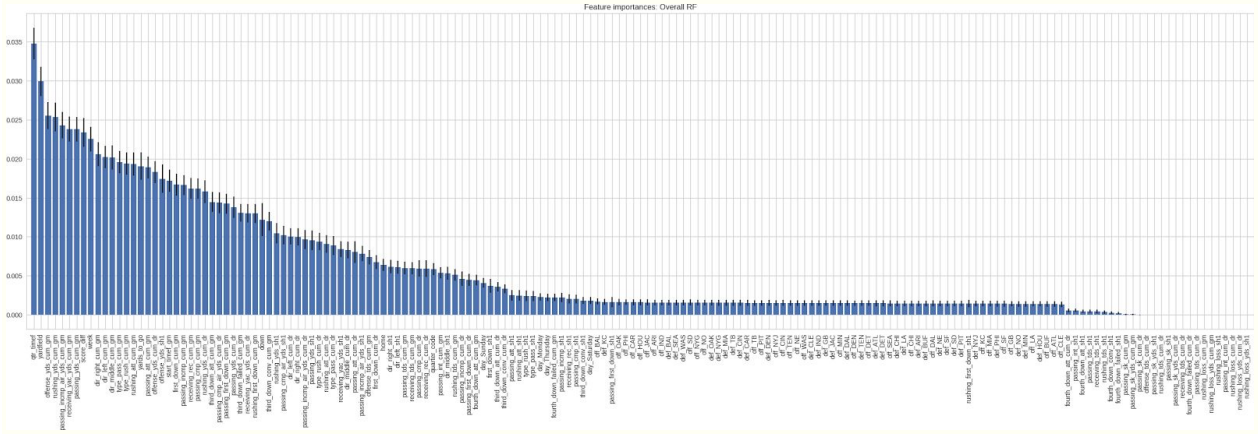
I computed a chi-square test to examine if the observed frequency of play types across teams differed from the expected frequencies based on the total play distribution in the NFL. The frequency distribution for each play type significantly depended upon team (pass left: χ^2=90.06, pass middle: χ^2=184.14, pass right: χ^2=54.12, rush left: χ^2=167.55, rush middle: χ^2=497.94, rush right: χ^2=164.30; all p's<0.001), indicating teams have different play calling strategies as expected.

Machine Learning (see [ml_algorithm_search_and_compare.ipynb])

Multiple machine learning algorithms were trained and tested on the 6 class target data with 10-fold, stratified cross validation. Observed performance did not significantly exceed simply labeling every example with the most prevalent class (Dum classifier) and most features were minimally informative. All feature importances were computed with the scikit-learn default method.

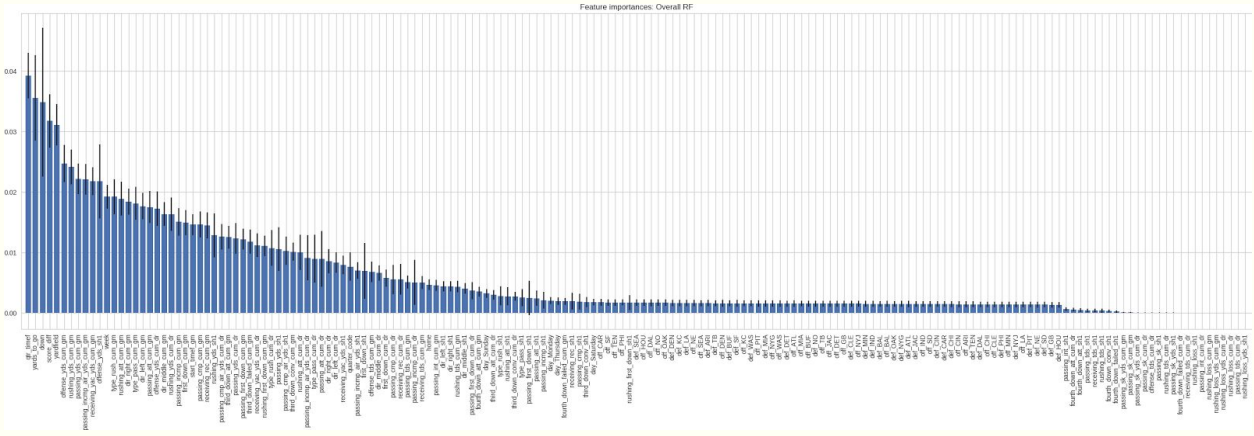


Random Forest Feature Importances: Play 6 class

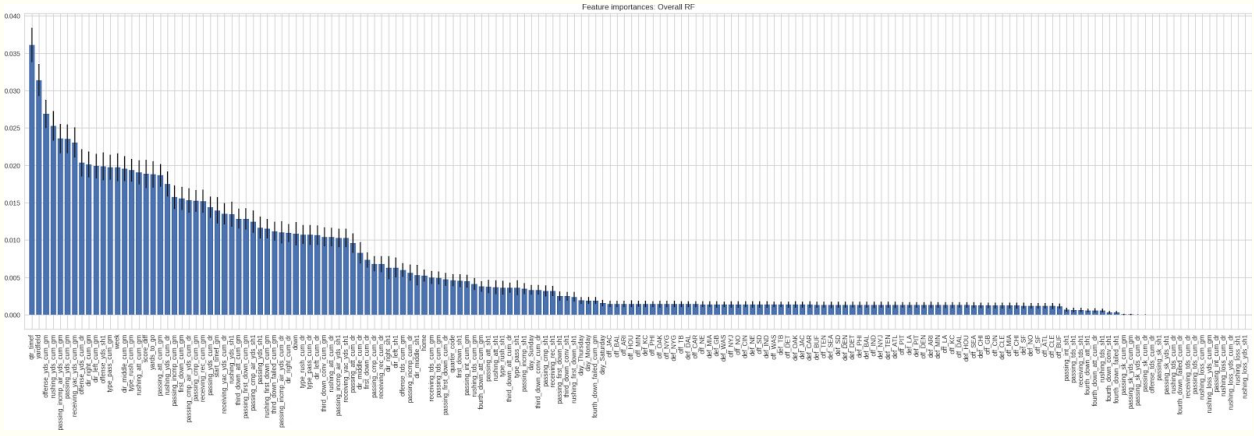


Thus, I decided to split the problem into a binary classification of play type (pass vs. rush) followed by a multiclass classification of play direction (left vs. middle vs. right). To gain an initial understanding of the predictive power of the 171 features, I trained a random forest classifier on the full dataset and plotted the feature importances and their standard deviation across decision trees.

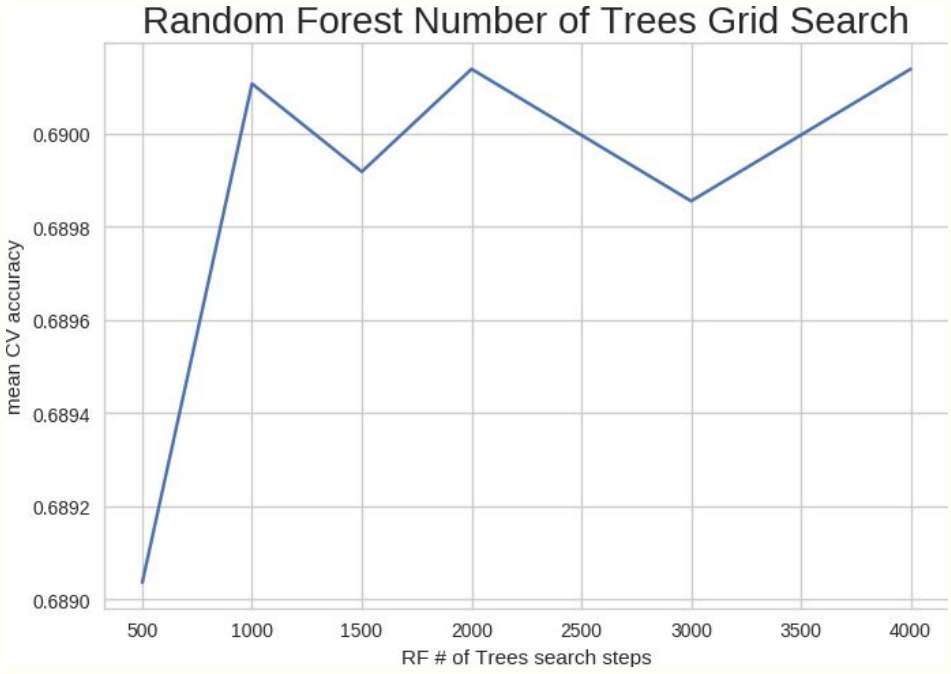
Random Forest Feature Importances: Play Type (Pass vs Rush)

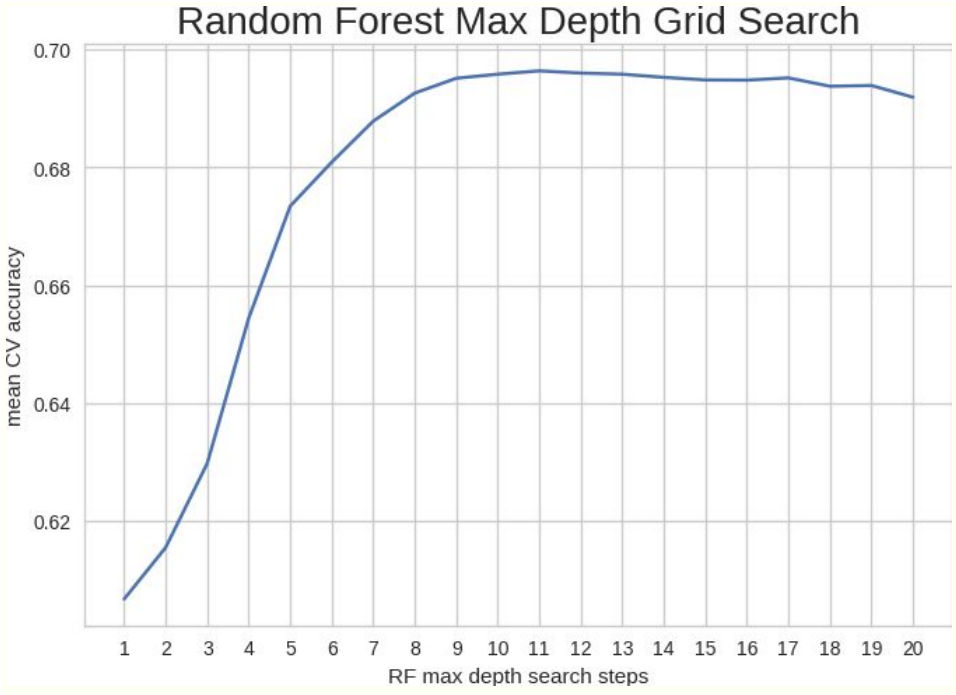


Random Forest Feature Importances: Play Dir (Left vs Mid vs Right)



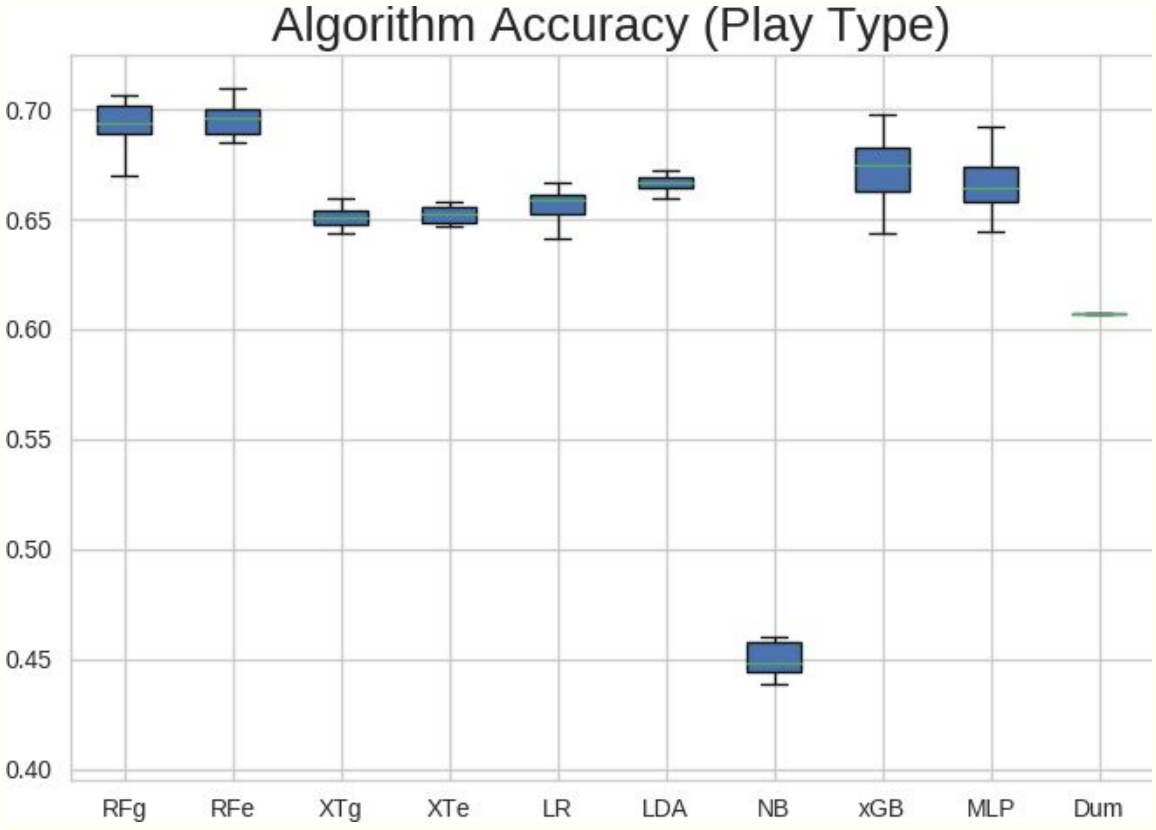
To optimize the number of trees in the random forest algorithm, I then performed a grid search of trees (500, 1000, 1500, 2000, 3000, 4000) with 3-fold cross-validation (CV), using scikit-learn's "GridSearchCV" class. Mean out-of-fold CV accuracy was nearly identical for 500 trees and the best performing 2000 trees (difference of 0.001), so I favored the less complex and speedier 500 trees in the final model. Using the resulting best estimator, I performed a grid search for maximum depth, varying depth from 1-20 levels. Mean out-of-fold CV accuracy was best with a max depth of 13, but performance plateaued at 9, which is the max depth used in the final model. The figures below represent results from the grid search on the play type target classification, but results were comparable for the play direction target.

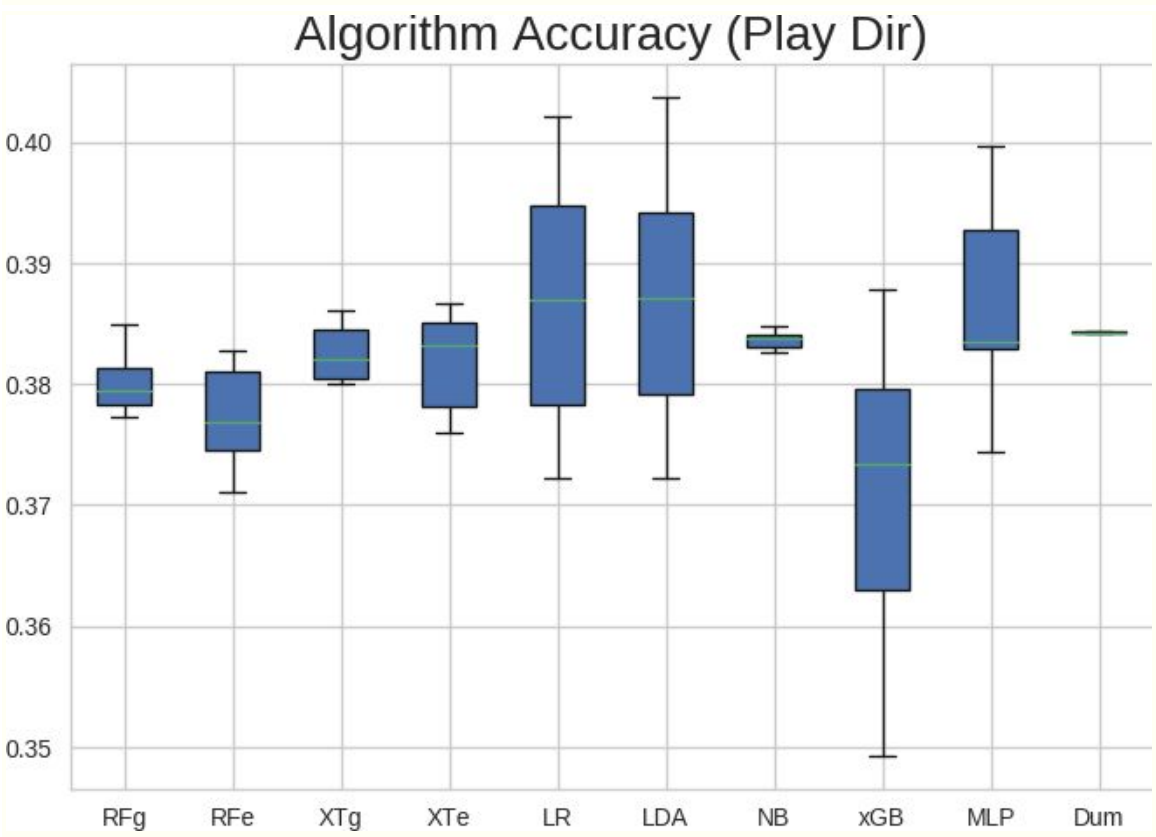




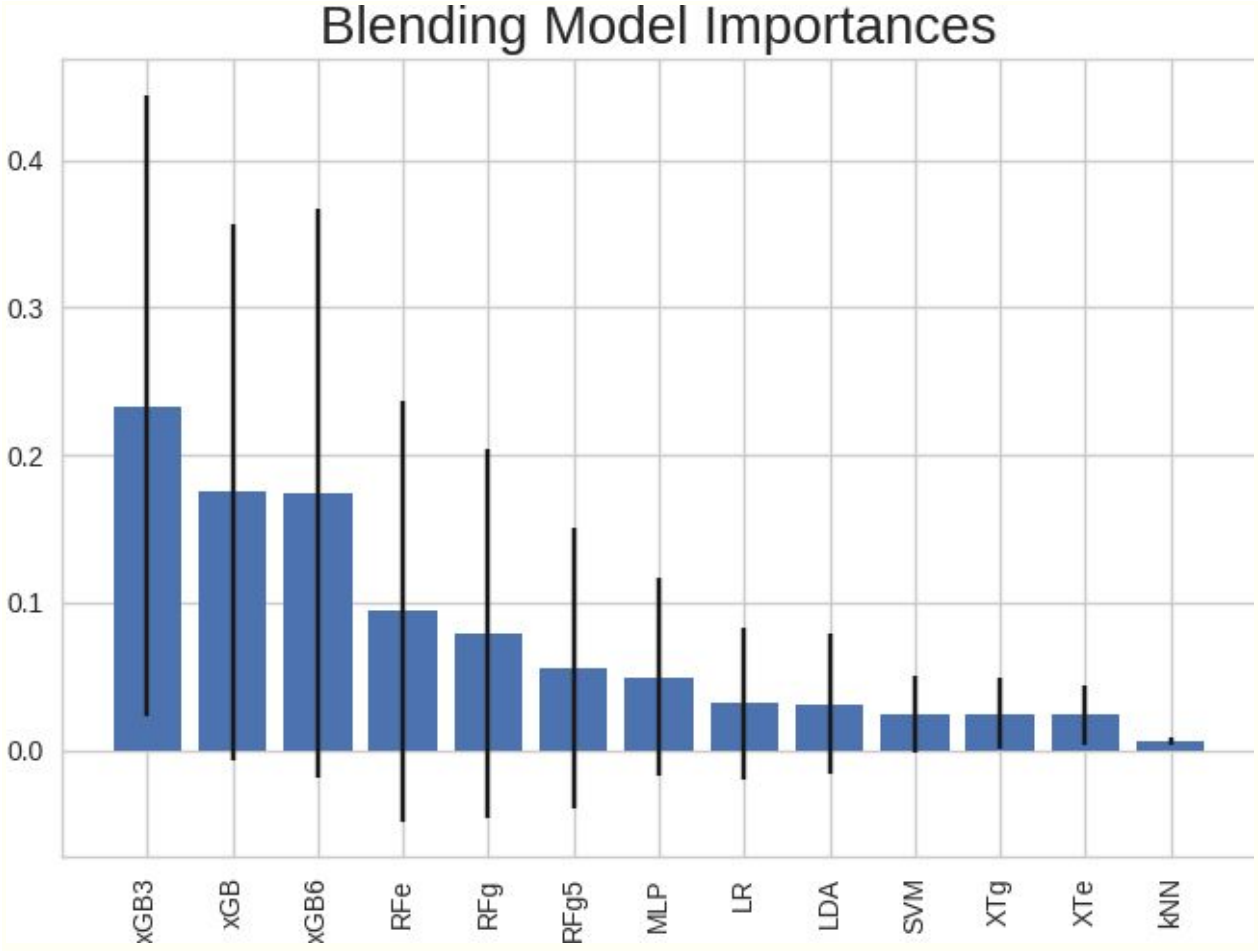
This deep level of trees indicates there is a high amount of interaction among the features, revealing that the predictive power of the features depends on the values of many other features. Several of the algorithms tested assume minimal such collinearity among features, thus I executed a greedy feature selection based on logistic regression coefficients for these algorithms. Using scikit-learn's "RFECV" class, features were recursively pruned from the full feature set based on absolute beta coefficients across 10-fold CV. This procedure selected 45 and 23 features for the play type and play direction models, respectively. These features were used for the logistic regression, linear discriminant analysis, and naive bayes algorithm training. The top 80 informative features from the random forest feature importances above were used for training all other algorithms in the final model. I experimented with using the top 20 or all features, but using 80 features offered the best balance between performance and parsimony.

Next, with a custom function "testAlgorithms", I tested and plotted accuracy for a diverse set of algorithms on each classification task. Accuracy was chosen as the performance metric in order to have a comparable metric across the multi-class and binary problems.To account for the imbalanced classes and provide a baseline, each model's accuracy was compared to a dummy classifier that simply labeled all examples as the most frequent class. For play direction, no algorithm performed significantly above labeling each example with the most frequent class (Dum), so the remainder of analyses will focus on play type.


Algorithm Accuracy (Play Type)
0.70
0.65
0.60
0.55
0.50
0.45
0.40
RFg
RFe
XTg
XTe
LR
LDA
NB
xGB
MLP
Dum


Algorithm Accuracy (Play Dir)
0.40
0.39
0.38
0.37
0.36
0.35
RFg
RFe
XTg
XTe
LR
LDA
NB
xGB
MLP
Dum

I then, with a custom function named “blend_proba”, ensembled the individual models together to enhance predictive power by capitalizing on the diverse residuals across the models. After testing a simple majority vote ensemble that selected the most common prediction across the models, I designed the winning model based an ensemble technique called blending. To begin, I randomly held out 10% of the plays to use for testing on the final blender model. With the remaining 90% of the plays I trained and tested the base machine learning algorithms, using stratified 10-fold CV which ensures equal proportions of target classes across folds. I obtained each algorithm’s average predicted probabilities across CV folds, which represents that algorithms predicted likelihood that the given play is a rush/pass. Those probabilities are then input as the training features for the blender model which was a Random Forest that selected features to split on using maximum entropy or information gain. The resulting feature importances, plotted below, depict the relative contribution of each base model to the final blending ensemble.



Finally, predictive probabilities were obtained from the Random Forest blender based on testing with new plays. The winning blender model boosted performance ~3% above the best base model and offered relatively strong predictive power with an accuracy of 73% on new play samples.

Recommendations and Future Analyses

The above analyses revealed a number of interesting patterns. Here are my recommendations for NFL coaches. First, scheme your passing defense to be strongest outside, as plays over the middle are less frequent for both passes and rushes. For example, spread out the linebackers slightly more to give them better coverage of outside. Second, do not let the previous play overly dictate your defensive calls because overall game patterns are much more predictive of the next play, according to the random forest feature importances. Third, on 2nd and 3rd use pass defenses far more frequently, as offenses are significantly more likely to call pass plays on later downs.

A fruitful avenue for future analysis would be to train and test a recurrent neural network (RNN) with long short term memory (LSTM) and an attentional component to not only capture the most recent plays but also plays from previous drives and games in similar situations. The sequence of plays is likely important in predicting the next play, and using a RNN may better capture this predictive information. Collecting additional data that includes features which are related to play direction would be helpful. Some examples include gameday weather, stadium cardinal direction, field surface, offensive line players and rankings, defensive and offensive team rankings, player rankings, and coaches tendencies.

As a result of this project, I identified and elucidated factors that are predictive of the opponent's upcoming play. I built a model that can give coaches data-driven information to guide their defensive play calling and strategy.