# Project Proposal

The NFL's Next Gen Stats has partnered with Kaggle.com to see who can best predict how many yards an NFL player will gain after receiving a handoff. The NFL has been around since 1920, and over the last 100 years coaches, players, and fans have accumulated a treasure trove of conventional wisdom about what makes a team, player, or play successful. Recently, the NFL has begun to follow the footsteps of Major League Baseball by using sophisticated data analytics to find out where conventional wisdom is right, and where it might fall short.

There are two main use cases for a model that predicts rushing yards. The first is for the benefit of NFL teams who can take insights from the model and apply them to their own strategy to improve performance. The second is for use during the telecast or pre-game/post-game breakdowns. The NFL makes more than $6 billion annually from TV deals, and this viewing experience is enhanced by highly knowledgeable sport's casters, commentators, and analysts. If a predictive model can be used by these experts to further enhance the viewing experience, the potential value to the NFL could easily be in the tens of millions.

The data was collected by Next Gen Stats and contains over 400 features relevant to the game, play, and players. The data is made available via Kaggle's NFL Big Data Bowl competition. Because the goal is to predict the number of yards gained, the best approach will likely be to use a regression model.

Deliverables for this project will be the code required to train the model and make predictions given new data, as well as a slide deck presentation explaining the process and findings. I will also include a written report detailing the full process of building the model.

# Data Wrangling

I used data from the NFL Big Data Bowl Kaggle competition. The data was imported directly from Kaggle and contained 682,154 rows and 48 columns. Each row represents a player on a given play, and because there are 11 offensive and 11 defensive players per play, every batch of 22 rows represents a single play. Thus, there were 31,007 plays to be analyzed in the data set.

## Variations in spelling/misspelled words

Although the data was provided by Kaggle, there were still several cleaning steps needed before analysis. The first was with the StadiumType variable, which contained many misspellings or alternative spellings of the same word (ex. Outdoor, outdoors, outdor, Ourdoor, etc.) There were 33 unique variations which I manually recoded into 2 categories: indoor and outdoor.

A similar variable was Turf, which described the field of play. Again there were many variations of spellings, which I manually recoded into two categories: natural and artificial.

The team of the player was represented by a three letter abbreviation, however there were some inconsistencies. For example, the Arizona Cardinals were sometimes represented

as ARI and other times as ARZ, so I found the few offenders and recoded those to be consistent.

The WindSpeed variable sometimes contained a single number, and other times contained the 'mph' unit at the end, so I stripped 'mph' from every row. Other rows contained two numbers (ex. 12-22), so I took the average of the two numbers in those cases.

The WindDirection variable also had variations (ex. 'N', 'North', 'from the north', etc.), so I converted these to be consistent acronyms (ex. 'N', 'NW', 'SE', etc.). Because there are 8 possible directions I assigned values between ⅛ and 8/8 starting with 'N' and moving clockwise ('N' = ⅛, 'NE' = 2/8, 'E' = ⅜, etc.).

The Weather variable had many possible variations, so I looked at the most common words and decided to make a numbered scale that ranged from -3 being most inclement to +3 being most fair. Mention of snow was given -3, rain was -2, cloudy was -1, clear was 1, sunny was 2, and indoor stadiums or closed dome stadiums were given a 3. Any weather condition that could not be placed in one of these categories was given a 0.

Converting to common units

The GameClock variable shows how much time was left in the quarter and was in the format Minutes:Seconds:Hundredths of a second. Because the data was collected once per second, the hundredths were always zero. I converted this variable to be the total number of seconds remaining in the quarter. The height of players was in the format Feet-Inches, so I converted this variable to total inches. The TimeSnap, TimeHandoff, and PlayerBirthDate variables were in the form of a datetime object, but were imported as a string object so I converted them to datetime.

Standardizing all plays to go from left to right

About half of the plays moved from left to right, while the other half moved from right to left. This is important because the X and Y positions, and the Orientation must be interpreted relative to the direction of the play. The field is 120 yards long (including the endzones) so if the play was going right to left I changed the X position to be (120-X). The field is 53 and ⅓ yards wide, so for plays moving right to left I changed the Y positions to be (53.33-Y). The orientation variable was in degrees from 1-360, so if the play was right to left I changed it to be (180-d).

Missing Values

The variables with missing values were FieldPosition, StadiumType, Temperature, Humidity, WindSpeed, and WindDirection. FieldPosition is irrelevant because that can be determined by the YardsLeft variable which has no missing values, so FieldPosition was dropped entirely. For StadiumType, I created a dictionary of every team and the type of the home stadium. For every missing row I used the HomeTeamAbbr variable to assign the StadiumType.

The Temperature, Humidity, WindSpeed and WindDirection variables were all missing for the indoor stadiums. I did some research and found that indoor football stadiums are usually set to 65 degrees F for games so I filled in all missing values with 65. I could not find the typical humidity of indoor football stadiums, but I did find that the typical humidity inside a home is

about 50%. The average Humidity in the data set was 54.76 so I used mean replacement for the missing values. There is no wind in an indoor stadium, so I set the missing WindSpeed and WindDirection variables equal to 0.

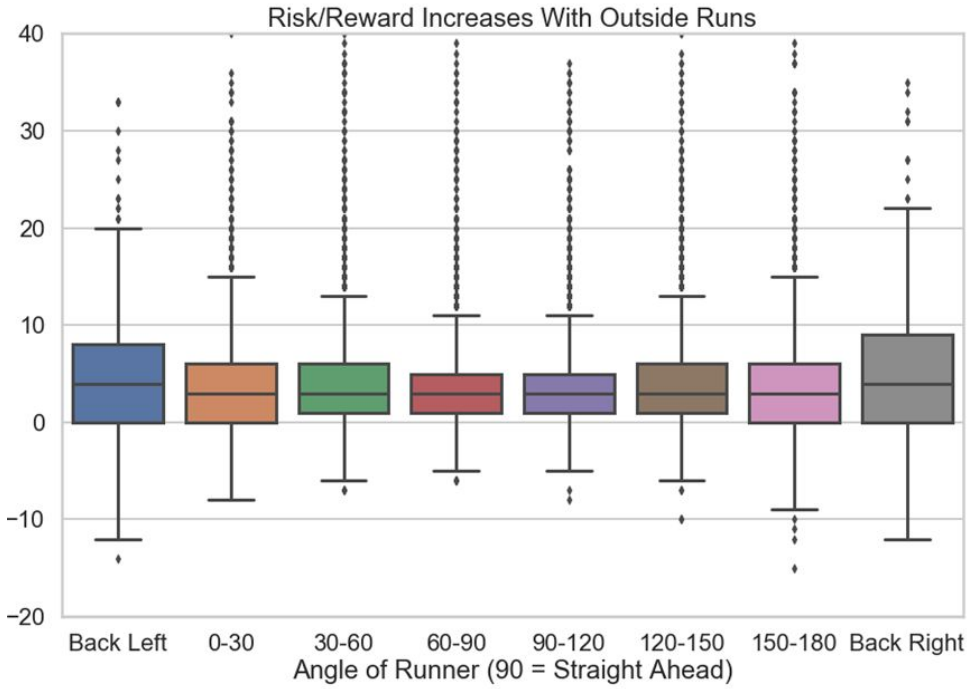
Outliers

I expect X, Y, and Orientation to be the most important predictors, and they are bound by the field of play and the possible 360 degrees of orientation, so I do not consider any of these to have outliers. I checked all of the numeric variables for outliers using a boxplot, and while some measurements were outside of 1.5x the IQR, they were all realistic values in a game of football and did not seem like they were measured incorrectly, so I did not remove any outliers from the data set.

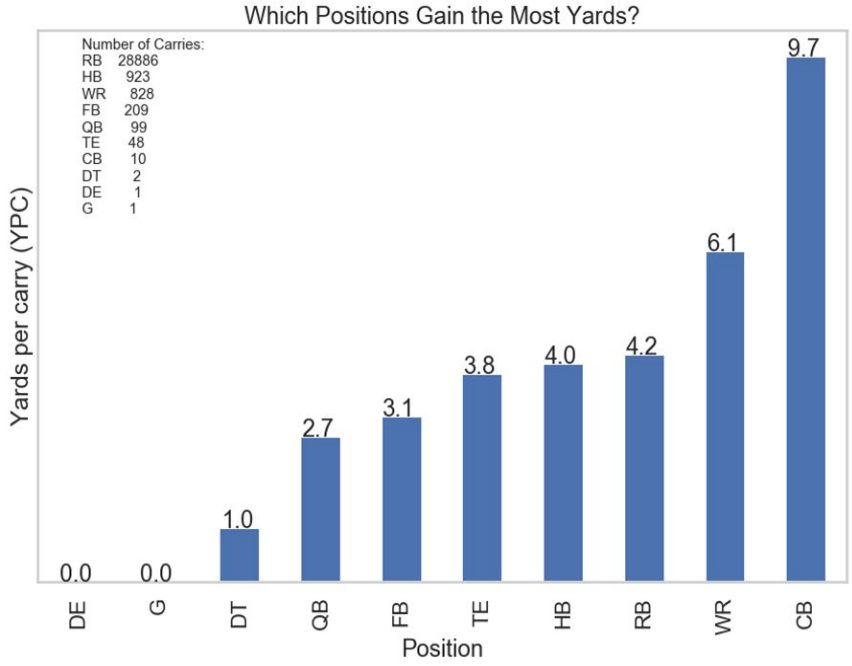
## Graphical EDA Findings

Most of the exploratory data analysis involved identifying variables that are correlated with yards gained. Many of these findings agree with conventional wisdom about football. For example, teams tend to gain more yards when there are fewer defenders “in the box”. “Getting the ball in space” is a common phrase in football that refers to putting players in position to utilize their athleticism to juke and outrun defenders, and it appears that the data corroborates this idea. Runners who get the ball with more space between themselves and the nearest defender tend to gain more yards.

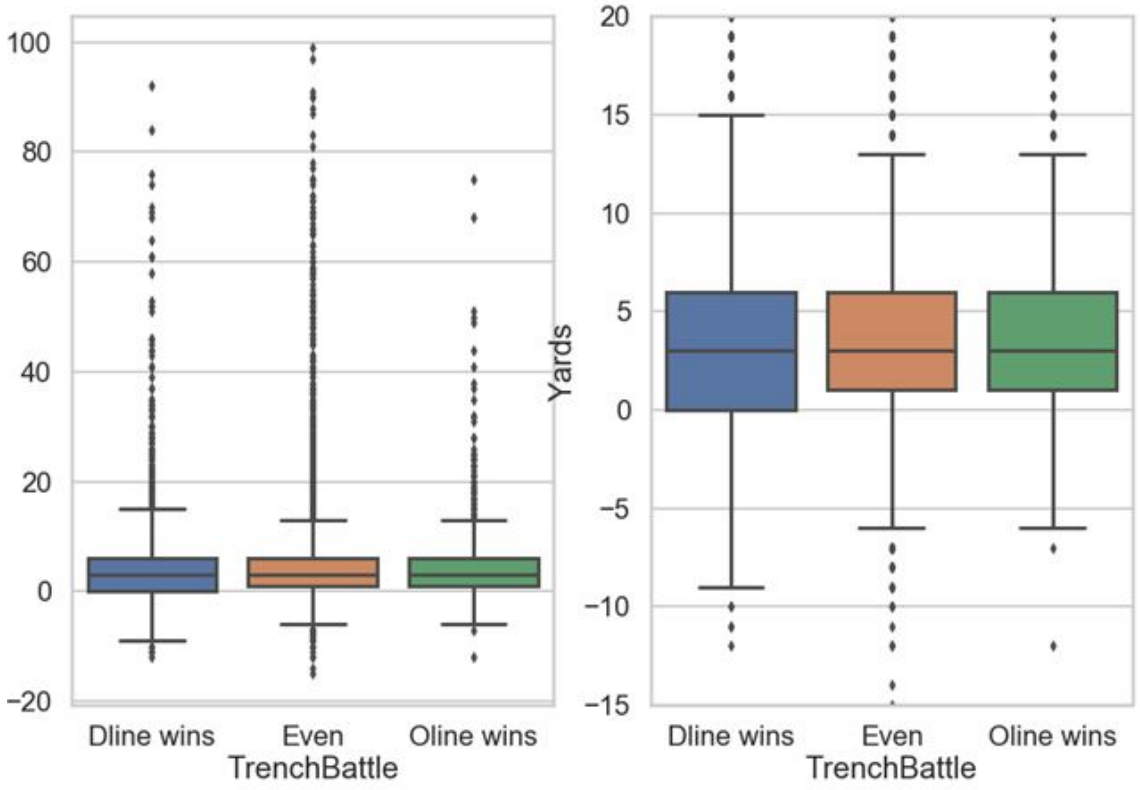
Another interesting finding was the relationship between the runners direction at the handoff and the yards gained. NFL commentators often express a preference for “North-South” runners (those that run directly towards the opponent’s end zone with little deviation left or right). The idea is that running “North-South” is safer, while dancing “East and West” has a higher risk/reward ratio. Again, the data seem to agree. The more a player is headed toward the sideline when receiving the handoff, the higher the variance in yards gained.



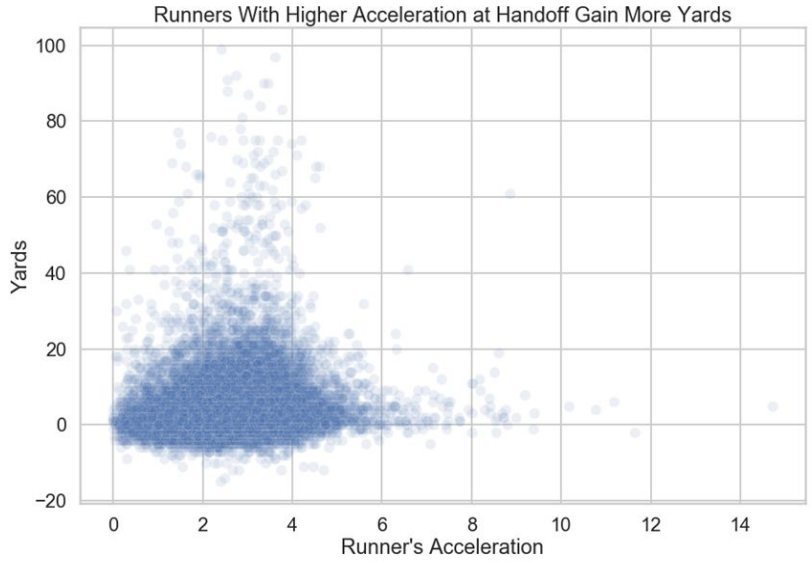

Running backs and half backs (a position nearly identical to running backs) carry the ball on 96% of running plays. Wide receivers make up most of the remaining 4% (828 total), but gain nearly two extra yards per carry (4.2 vs. 6.1). This is an enormous discrepancy in football; for context the highest career YPC for a running back in the history of the league is Jamaal Charles with 5.4. While this may seem to indicate that teams should hand the ball to their wide receivers more often, some of the success of wide receivers is likely due to the surprise factor from how rarely they run the ball.




A potentially surprising finding has to do with how the battle between the offensive and defensive lines affects a running play. When the ball is snapped, the offensive line tries to push the defensive line backwards to give more space for the runner to run. Conversely, the defensive line tries to push or penetrate the offensive line in order to get to the runner. To get an idea of which line was successful in that objective on a given play, I calculated the average position of all linemen relative to the line of scrimmage. Surprisingly, on plays where the average position was behind the line of scrimmage (success for the defense) the yards gained had a higher variance. The risk/reward ratio was much higher on these plays. This presents an opportunity to further investigate what separates the big gain plays from the big loss plays when the defensive line gets in the backfield.




The most direct and significant correlations found had mostly to do with the physics of the runner at the handoff. Runner's with a higher speed and acceleration at the handoff tend to gain more yards.



## Inferential Statistics

I started my statistical investigation by looking at the Pearson correlation between every numeric independent variable and the dependent variable: yards. I created a data frame containing four columns: variable, Pearson's correlation coefficient, absolute value of Pearson's correlation coefficient, and p-value. I then selected only rows with a p-value less than 0.05 and sorted by absolute correlation coefficient. From here I could quickly see which variables had the strongest relationships with yards. Runner's acceleration, runner's distance to the defense's centroid, and yards left to the goal line were the three variables with the strongest relationships with yards. Runners with higher acceleration and runners further from the centroid of the defense gain more yards. Runners further away from the goal line also gain more yards, however this is expected because the runner can only gain as many yards as are left to the goal line. Some other variables that had a strong relationship with yards gained were runner's speed, distance to first down, runner's distance to closest defender, and combined weight of the defensive line.

I also conducted tests to see if average yards gained differed for some categories. First I looked at whether average yards gained differed by the down of the play (first down, second down, etc.). The Levene test showed that the variances of the categories were not homogenous, and I know that yards gained are not normally distributed so I used the Kruskal-Wallis H-test to determine whether median yards gained differed by category. This test was highly significant, and pos-hoc analysis shows that runs on fourth down gain significantly fewer yards than those on first, second, or third downs. Additionally, runs on third down gain fewer yards than those on first or second.

Another test was to check whether runs to the weak side of the field gained more yards than those to the strong side. The strong side of the field is defined as the side (left or right) with more offensive linemen. Although runs to the strong side of the field gain more yards on average, an independent samples t-test shows that this difference is not significant.

## In-Depth Model Analysis

For this problem I decided to use a Light Gradient Boosting Machine (LGBM) model. This model is a gradient boosting decision tree algorithm that grows trees leaf-wise instead of level-wise. I chose LGBM because it handles large data sets quickly, while being equally or more accurate than other boosting algorithms such as XGBOOST. This allows me to tune my hyperparameters over a large parameter space in a short amount of time. I use the LGBMRegressor object from the lgbm package to perform this analysis.

The first step was to convert any categorical variables in the data to dummy variables if I wished to use them in analysis. Most of the categorical variables, such as stadium, showed no relationship with the outcome during exploratory data analysis so I simply removed them from the data. PossessionTeam and runner's position were found to be important, so they were converted to dummy variables. Lgbm can handle categorical variables internally when they are predefined, however there are reports that feature importance of these variables can be inflated when choosing this route, so I decided to create dummies manually.

I then separated the outcome variable, Yards, into a separate array and created an 80/20 train/test split on the data. Because the outcome variable is extremely skewed, I use the StandardScaler to standardize Yards in both the train and test data. I then used the training data to perform 10-fold cross validation to tune the hyperparameters.

*Feature Engineering*

The raw data came with many features describing every player on the field as well as characteristics of the specific play and of the game in which the play occured. Other features, however, needed to be engineered from the available data. The following is a list of engineered features from the data:

DlineWeight - The combined weight of defensive linemen
OlineWeight - The combined weight of offensive linemen
OlineDlineRatio - The ratio of OlineWeigh to DlineWeight
DlineAvgX - The average X position of the defensive line relative to line of scrimmage
OlineAvgX - The average X position of the offensive line relative to line of scrimmage
ClosestDefender - the distance from the runner to the closest defender
DefenseCentroid_Y - The average y position of all defenders
DefenseCentroid_X - The average x position of all defenders
DistanceToDefenseCentroid - The distance from the runner to the centroid of defense
Vertical/HorizontalSpeed - Separately, the vertical and horizontal component of runner's speed
Vertical/HorizontalAcceleration - Separately, the vertical and horizontal component of runner's acceleration
StrongSideRun - Boolean, whether the runner was running toward the strongside of the field
PlayerBMI - The body mass index of each player
DL/LB/DB/WR/TE/RB/OL - A column for each position on the field giving the total number of players with that position on the play
TimeDelta - The elapsed time between the snap of the ball and the handoff
LOS_Abs - The absolute line of scrimmage ranging from 1-99 (99 represents the opponent's 1 yard line

*Parameter Tuning*

One of the biggest risks of lgbm is the possibility of overfitting. Because the trees grow leaf-wise, they can become quite complex, which is the source of both the high accuracy and the risk of overfitting. The best way to reduce overfitting is to limit the number of leaves per tree. I conducted 10 fold cross validation over the space of 2 to 100 leaves in increments of 5 and saved the validation score (scoring metric discussed later) for each iteration. The best validation score was achieved with 90 leaves.

Another parameter that needs to be tuned is the learning rate. A lower learning rate during training will reduce the risk of overfitting, but may also require a large number of trees (estimators) to improve accuracy. For this reason, these two parameters are often tuned in

conjunction with one another. To improve the speed of cross validation I reduced the number of folds to 5. I then trained models for every combination of learning rate between 0.0001 and 0.1 in log10 space and number of trees ranging from 200 to 500 in increments of 100. The best validation score was achieved with a learning rate of 0.01 and 300 trees.

*Scoring Metric*

The scoring metric used in this competition is Continuous Ranked Probability Score (CRPS). CRPS is an analog of Mean Absolute Error for a Cumulative Density Function. This is useful for scoring probabilistic forecasts rather than a single predicted value. Although the regression model does predict a single value, it is then converted to an array of probabilities that represent the probability that the runner will gain fewer than a certain amount of yards. The prediction array has a length of 199 because there are 199 possible outcomes ranging from losing 99 yards to gaining 99 yards. The predicted array will have probabilities that will increase from 0 to 1 as the number of yards increases, whereas the actual answer array will be all zeroes until the point of the actual yards gained at which point it will be all ones.

*Training and Predict*

Once the parameters were tuned, actually training the model was very simple. I first instantiated a LGBMRegressor object with the selected parameters, and then called the fit method on the training data. Then the trained model was used to predict the test data. Using CRPS to compare the predicted values with the actual test values yielded a score of 0.01479, which would place my model in the middle of the Kaggle leaderboard.

*Feature Importance*

The top 4 most important features had to do with the runner's speed and acceleration. Raw acceleration and speed were the top two, followed by the vertical component of acceleration and speed. After this the runner's distance to the centroid of the defense was next most important, followed by the average position of the defensive line relative to the line of scrimmage. Also important was the runner's distance to the nearest defender.

*Conclusions*

I think the most interesting takeaway from this exercise is that basic physical properties like speed and position is much more important to the success of a play compared to more intricate features like formation and situation within the game. The relative importance of speed and acceleration at the handoff indicates that draw plays are less likely to be successful compared to other running plays. In terms of spacing, it appears to be very important that the runner receive the handoff with a lot of space around himself. This model would suggest that misdirection plays in which the defense is tricked into moving one direction, while the runner gets the ball moving very fast in the other direction are likely to be successful.

The primary use case for a model like this would likely be used during the broadcast or post-game analysis. The commentators might show an image of a certain play and say that the analytics predicts that this play has an X% of gaining Y yards or less. Usually these examples are used to show how a certain player defied the odds and did something beyond what they

were expected to, such as in this commercial. These interesting examples of elite players making extraordinary plays can capture the excitement of fans and really add to the viewing experience.