For those who play fantasy football, it can be difficult to know which players to add to a roster, and of those players, which ones to put in a starting lineup. Fantasy football hosting sites like ESPN do a reasonable job of projecting each player's score for the coming week, but the actual score can deviate sharply from these projections. Players who are relative-unknowns may unexpectedly have stand-out performances, while players who are expected to perform well, may not. Additionally, fantasy owners often must choose between two or more players on their roster that have similar projected scores.

A solution to this problem would benefit both fantasy football owners and hosting sites. Owners will be able to make better-informed decisions about the best players to draft, as well as reduce ambiguity between seemingly equal players on their roster. Fantasy football hosting sites can also benefit by providing their patrons more accurate week-to-week projections of players.

The data for this problem was gathered from two sources. First, we used a dataset comprised of scraped player statistics from each game they played in, and joined it with that player's profile information in order to have each players name, height, weight, etc. The data can be accessed at https://www.kaggle.com/zynicide/nfl-football-player-stats/home. We later combined this with statistics for the opponent that the player faced each week, obtained from https://www.pro-football-reference.com/.

Ultimately the model will be able to take in new player data each week, and predict player performance in their upcoming game based on 50 years worth of historical data.

## Data Preprocessing

Wrangling and Cleaning Player Statistics

Player statistics for each game were joined with their profile information. Non-fantasy football positions were filtered out, leaving only the positions: quarterback, running back, wide receiver, tight end, and kicker. Columns for non-fantasy football statistics were also removed. The scraped data contained team name abbreviations, which needed to be converted to the full team name so that it could later be joined to the opponent data. Some statistics had too much missing data due to the fact that they started being recorded long after 1966.

Wrangling and Cleaning Opponent Data

Opponent statistics were obtained for each week of each season of the Super Bowl era (dating back to 1966). Due to bye-weeks, and other reasons for a team skipping a week, a column needed to be created for the *true* number game of the season for that team. This column would be essential for creating cumulative season averages in various statistics. As with the player dataset, some statistics started being tracked much later than 1966. Columns that did not have enough data were dropped (e.g. third-down conversions). Null values were imputed as appropriate. For example, if a team had zero turnovers during a game, NaN would be filled with 0. The date also needed to be parsed so that it could be joined with the players dataset.

Feature Engineering

It is assumed that recent player performance is a strong indicator of success in the upcoming week. To give the model an understanding of how the player has performed in recent weeks, player stat columns were converted to a mean of the previous three weeks. Beyond this, the bulk of the feature engineering took place in the opponent dataset. For the model to understand how strong of an opponent a player is up against, it needed to know how the opponent had faired up to that point in the season. With that in mind, the weekly stats needed to be converted to a cumulative average of each stat, resetting with each new season. Another indicator of an opponent's strength is their winning percentage. This feature was created with the help of the "record" column.

EDA

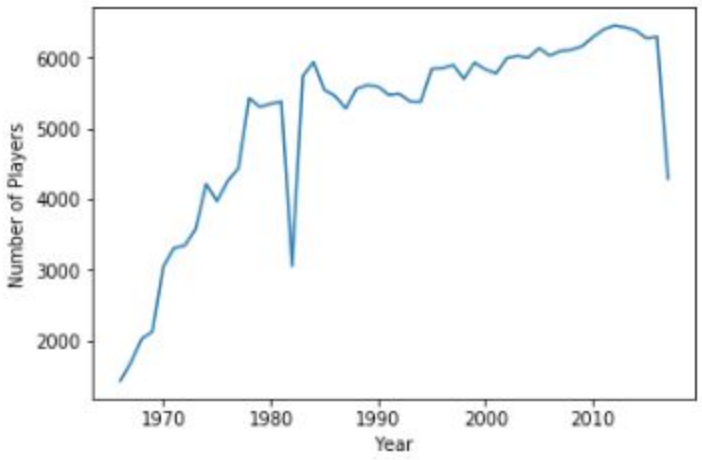
Following are some findings, having joined the DataFrames and removed extra columns.

*How many times did a player score more than 40 fantasy points?*
Only 211 out of 266,762 possible games.

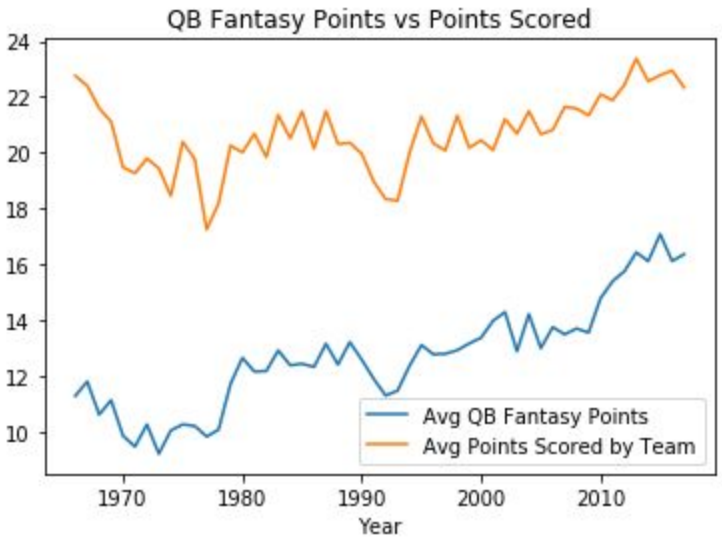
*The number of rows by position are as follows:*

- Wide Receiver: 84,038
- Running Back: 80,325
- Tight End: 49,318
- Quarterback: 31,031
- Kicker: 22,050

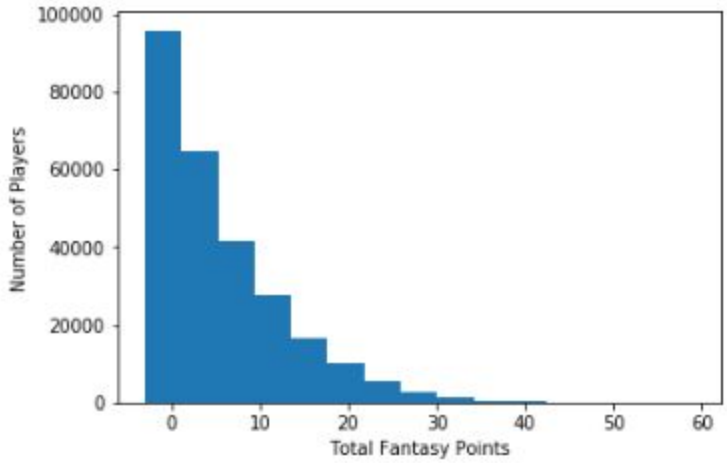
*A look at the number of players in the dataset by year. In 1982, there was a player strike, as seen by the sudden drop in the graph. The data was scraped mid-season, explaining the drop at the end.*




*Quarterbacks are scoring more fantasy points than they used to. Teams are also scoring more than they used to. Spectators like to see offensive games, so the league has encouraged more scoring through various modifications to the rules.*




*Although a significant number of players have zero fantasy points, this is not considered bad data. The presumption is that these games will aid the model in making predictions.*




## Machine Learning

Model Selection and Evaluation

Due to the moderate size and complexity of the data (266K rows, 88 columns), it was assumed that a linear regression model would not be ideal in this instance. To cater to the presumably non-linear relationships between the variables, a random forest was selected. Mean absolute error (MAE) was chosen as the metric of evaluation.

## Limitations of Random Forest

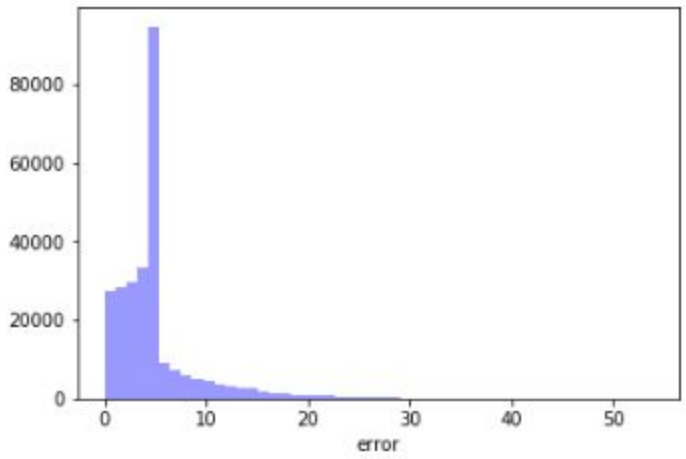
A key drawback to using a random forest is that it is computationally expensive. This became apparent as various hyperparameters were tested. It was only possible to test a handful of shallow “max_depths”. For educational purposes, a gridsearch was performed with tree depths ranging from 3 to 7, and then from 8 to 12. This is not ideal considering the number of features in the model (87). As few as 10, and as many as 100 trees were tested. In the first iteration, a mean absolute error (MAE) of 3.74 was recorded. In the second iteration, the MAE improved to 3.64. It is presumed that with larger tree depths, accuracy would continue to improve.

## Decision Tree

To avoid the limitations of a random forest, the data was then fit to a decision tree regressor. Although a decision tree is prone to overfitting, it was determined to be most optimal model, all things considered. For the decision tree, tree depths ranging between 5 and 50 were tested. The model produced a MAE of 3.73 and a RMSE of 5.34. R-squared was 0.41. The decision tree had similar performance to the limited random forest, and it was able to achieve this in a fraction of the time (roughly two orders of magnitude faster).

## Interpreting the results

The decision tree was able to predict a player’s fantasy score within six points 82% of the time. As noted in the distribution of the error below, many errors hovered around 5 points. This is likely due to the model taking the “safe” option when trying to predict a zero fantasy score. Most fantasy players would like to see the prediction be a little closer than six points. Nevertheless, this provides an acceptable baseline from which to improve.




Suggestions for Future Analysis

The player statistics were converted to an average of the past three weeks in an effort to take into account their recent performance. Through testing, it may be discovered that a different window is better. Different windows for the opponent's average statistics could similarly be tested. Additional feature engineering may also look at columns that were initially left out like "height." The height of a player matters more depending on the position, i.e., it's advantageous to be tall if you are a wide receiver, but not if you are a running back. To include a feature like height, it may be useful to create separate models for each position, and cater features accordingly. It may also be useful to place a weight on more recent players, since they will more closely match the performance of today.

The top 10 features from the decision tree can be seen below. An examination of the feature importances may also aid in future feature engineering.

*Note: A feature with an "x" suffix refers to the player's statistic (average of the previous three weeks), while a "y" suffix refers to the opponent's statistic (average for the season).*

| | feature | importance |
|---|---|---|
| 11 | passing_yards | 0.327963 |
| 18 | receiving_yards | 0.303752 |
| 21 | rushing_yards_x | 0.237516 |
| 14 | point_after_makes_pct | 0.067114 |
| 19 | rushing_attempts_x | 0.033936 |
| 16 | receiving_targets | 0.008929 |
| 7 | passing_rating | 0.004240 |
| 12 | player_team_score | 0.002187 |
| 22 | Season | 0.001913 |
| 23 | year_of_game | 0.001302 |