**Steps**:

1. Write your capstone project 1 milestone report (Google Doc, 5-6 pages) and include the following:
   1. Problem statement: Why it’s a useful question to answer and for whom (get this from your proposal)
   2. Description of the dataset, how you obtained, cleaned, and wrangled it (get this from your data wrangling report)
   3. Initial findings from exploratory analysis (get this from your data story and inferential statistics reports)
      1. Summary of findings
      2. Visuals and statistics to support findings
2. Update your presentation slides.
3. Update your GitHub repository with the capstone project 1 code, milestone report, document, and slides .
4. Use the link below to share your report with your mentor for feedback, and update as needed.
5. Convert to .pdf and add to your repository. Share with your peer community.

Problem Statement:

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.

There are at least two types of consumers who could benefit from a solution to this problem: 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.

Concerning the data used to solve this problem, the initial step is to gather the fantasy score for each player in each game they have played in. Beyond this, it is a matter of determining what features are most predictive of a player’s fantasy score, and gathering those metrics; for example, data on how the player performed prior to this game. This may include the prior week, the prior three weeks, year-to-date, etc. Additionally, it is widely assumed that “quality of opponent” is an important factor in scoring fantasy points. “Quality of opponent” can be determined from the opponent’s W-L record, as well as from their defensive statistics against various positions (e.g. running back, wide receiver). Although there are a number of sources that have publicly available player data and game data, this project will rely on data gathered from ProFootballReference.com.

In solving the problem, we will first examine the factors believed to be strongly correlated with fantasy football performance (e.g. the player’s prior-week performance, the opponent’s win-percentage at the time of the game). Out of these, the features with the highest correlation will be kept. Regression analysis will be utilized to output a predicted fantasy football score for the player, ultimately resulting in a ranking of all players for the given week.

Description of the dataset:

**Steps for preparing the data:**

The process for preparing this dataset is outlined chronologically below. In the final tidy dataset, each observation is an NFL player in a given game, and the features are the player’s statistics for that game, as well as the opposing team’s average statistics up to that point in the current season.

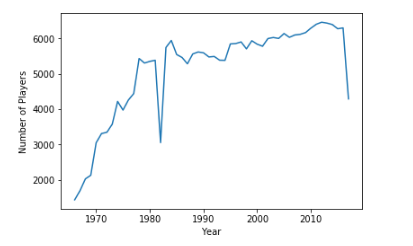
1. **Assemble player data**
   1. Initially, the player and game data resided in two different datasets found on Kaggle. The two datasets were merged into one. <https://www.kaggle.com/zynicide/nfl-football-player-stats/home>
   2. Since fantasy football does not involve individual players on defense or special teams, columns containing defense and special teams statistics were removed.
   3. For each statistic used in fantasy football, a new column was created converting the raw stat to the ESPN fantasy football point-value. After all these new columns were created, another column summing all the fantasy point columns was created, giving the total fantasy points accumulated by that player during that game. Fantasy point total is the target variable in this analysis.
   4. Players (rows) on defense and special teams were removed from the dataset.
   5. In this dataset, both the team the player plays for, and their opponent are abbreviated to three letters. In order to later join the opponent dataset with this dataset, the abbreviated opponent name needed to be converted to the full team name. The player’s own team name was also converted to its full name for convenience in reading the data.
   6. Certain statistic columns had too many null values, often due to the fact that the statistic started being tracked much later than 1966. In these cases, the columns were removed altogether.
2. **Cleaning the opponent data**
   1. Opponent statistics were obtained for each week of each season of the Super Bowl era (dating back to the 1966 season). <https://www.pro-football-reference.com/>
   2. Due to bye-weeks, and other various reasons for a team skipping a week, a column needed to be created for the *true* number game of the season for that team.
   3. 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).
   4. Missing values were found in a number of columns. A variety of imputation techniques were used, depending on the data in that column. In some cases, NaN values were filled with a zero (e.g. first-downs allowed). For the “home\_or\_away” column, the non-null rows contain the “@” symbol representing an away game. For the home games, the null values were filled with “vs”. The “overtime” column had null values for all games that did not go to overtime, and were thus rows were filled with the word “no”.
   5. For each numerical stat column, the column was changed to reflect a running average for that season (i.e. week 2 statistics were converted to an average of the first two weeks). This required writing a function that took into account the all the weeks up to the current week in a season, and then reset with each new season.
   6. A column needed to be created reflecting the win-percentage for each week of the season. This required a couple of steps. The “record” column contained a string in the format of “1-0” or “11-3-1” and needed to be made uniform, i.e., rows in which a tie had not occurred yet in that season would not have a third number, and so “-0” was appended to the end in these instances. Then the record needed to be separated into a “wins”, “losses” and “ties” column. This required an if-then statement that checked for single or double-digits in the wins and losses column, and sliced the corresponding indices of the string. Once the “W”, “L”, “T” columns were created and converted to integers, a “win-pct” column could then be created by dividing the number in the “W” column by the “game\_of\_season” column that was created in step 2-b.
   7. A “datetime” column needed to be created out of the unformatted dates in the dataset. To accomplish this, the datetime.strptime() function was used to convert dates that were initially in the format of “December 17” to ISO 8601 format. After the conversion, the month number and the day were sliced and separated into their own columns. Then the “year”, “month” and “day” columns were parsed using pd.to\_datetime().
   8. The way this dataset is set up, the statistics on a given date included the statistics of that day. The purpose of this data is to predict how a player will do based on the weeks *leading up to that game*. In other words, a row from the players dataset needed to be matched up with the opponent’s previous week so that it reflected the average of the statistic *leading up* to that game but *not including* that game. To accomplish this, the datetime column was shifted “up” one row, such that “1972-12-20” was now the datetime for the row that used to be “1972-12-13”. After the shift, the last row in the dataframe was removed, as well as the last row of each season, because it would be irrelevant when matching to the player dataset. Finally, the index was reset after deleting the rows.
3. **Combining the player and the opponent datasets.**
   1. The “players” and “opponent” datasets were merged.
   2. Irrelevant columns were removed.
   3. After the merge, rows were dropped for which there was data in the players dataset, but no data in the opponent dataset (the datetime of the first week of each season, which was lost in the shift mentioned in step 2-h). These rows were not necessary because there are no opponent statistics to go off of in week 1.

Initial findings from EDA:

The number of rows by position is as follows:

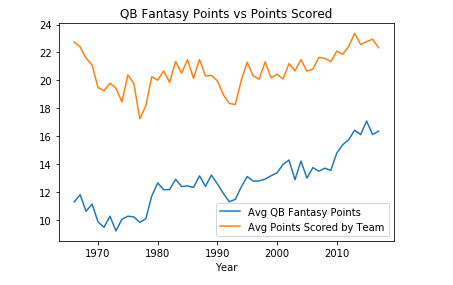
* Wide Receiver: 84,038
* Running Back: 80,325
* Tight End: 49,318
* Quarterback: 31,031
* Kicker: 22,050

A look at the number of players in the dataset by year:

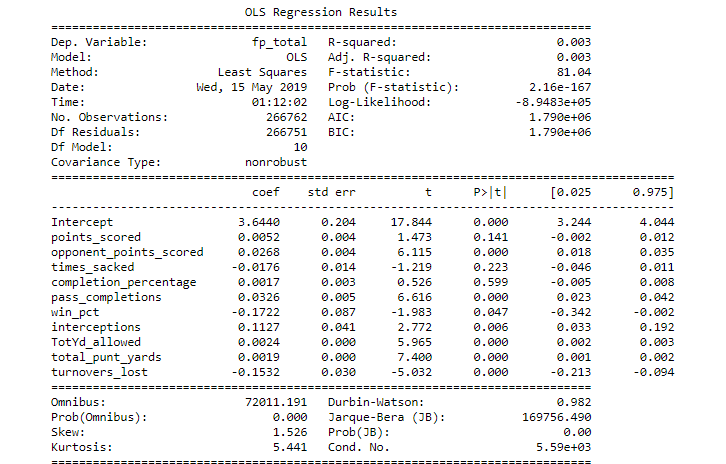


*Mean: 5,130*

*Standard Deviation: 1,326*



Initial regression analysis found the majority of the coefficients to be lower than expected (see below). R-squared and adjusted r-squared were also unusually low.



It was later discovered that a significant proportion of players had zero total fantasy points (29%). This fact likely played a significant role in the initial analysis. Removal of some, if not all of the samples with zero fantasy points is expected to lead to a stronger correlation between the features and the target.

