# Capstone 2

**Predicting Fantasy Points for NFL Quarterbacks**

***Using regression techniques to build winning fantasy football lineups***

## Introduction

The fantasy football industry has seen a massive spike in both revenue and the number of fantasy players in the past five years. One of the main reasons for this spike is how the concept of daily fantasy sports (DFS) has revolutionized the industry. Daily fantasy players no longer need to deal with the risk of season-ending injuries to their star players, drafting player for a season that underperform, making trades that end up in favor of the other team-owner, etc. Also, daily fantasy players are able to win money and get paid immediately after the final contest in a slate has been completed, instead of having to wait until the end of the season to collect their winnings. Players have the ability to draft a new fantasy team each week, or every day for certain sports like baseball or basketball. Contests between players who participate in DFS may include anywhere from 2 participants (1 versus 1) all the way up to 500,000+. Entry Fees for participants may range from no cost (usually a promotional contest with a small reward for playing) up to $10,000, and prizes may reach up to $2,000,000 for winning a tournament with a high volume of participants. Daily fantasy sites, such as DraftKings and FanDuel are more popular and lucrative than ever. The motivation of this project is to explore ways to implement machine learning to predict points scored for players, which will help us build profitable daily fantasy sports lineups. Those who enjoy playing NFL fantasy football could use our predictions as recommendations for using players in their lineups to help them make profitable decisions. For the purpose of this project, we will be focusing on predicting fantasy points for the quarterback position, as it is arguably the most important position to predict in fantasy football. We will, however, be collecting data for all positions for future work.

***How it Works:***

FanDuel and DraftKings both host DFS games in nearly every major sport, which include NFL, MLB, NBA, NHL, and NCAAF among others. For our purposes, we will be exploring professional football (NFL), and using FanDuel's format for building a team. For each NFL contest using FanDuel, the participant is allowed a $60,000 salary cap to draft his or her players. For each team, the participant may choose one quarterback (QB), two running backs (RB), three wide receivers (WR), one tight-end (TE), one flex (a choice of an extra RB, WR, or TE), and one team defense/special teams (DST). The participant chooses players for his/her team, and pays a fee to enter a contest. The site, or host (FanDuel in this case), of the DFS contest automatically takes 10% of the participant's entry fee upon entering the contest. This is known as the "rake", and is how the site makes its profit. Once a participant has entered a contest, in order for his or her entry to win money, it must place higher than a certain percentage of entries in the contest. The amount of money a single entry wins depends on what type of contest it is, as well as how many entries are in it. For example, if someone entered a two-person contest, and paid $1 to enter, that person's lineup must score higher than the opponent to win $1.80, since the host keeps 10% of each entry. Another example is if someone entered a tournament that had 20,000 entries for $5, the host keeps 10% of $100,000, and the remaining $90,000 would be up for grabs among the entries. The top prize might win $40,000, second place might win $15,000, third might win $5,000, and the remaining $30,000 is distributed in a decreasing manner among the remaining top 10-15% of entries in this contest.

## Data

### *Attribute Selection*

The attributes that will be used for this project will be mainly individual player statistics based on a player's overall history, and that player's recent history. We will also be using matchup attributes for an instance. Every position within a lineup is different, and thus different attributes will be important. Since we are focusing mainly on the quarterback position, we will need to take FanDuel's scoring system for quarterbacks into account, and will want to use individual overall history attributes, such as pass yards per game, rush yards per game, pass TD's per game, rush TD's per game, pass attempts per game, average fantasy points per game, and others. For current matchup attributes, we will be using defensive points allowed per game, whether the match is home or away, defensive yards allowed per game (pass, rushing, receiving), etc.

### *Obtaining the Data*

For all of the attributes described, there are many sources from which to obtain the data. One source is pro-football-reference.com, which has a database that contains nearly every relevant football statistic for every NFL game played dating back to 1920, which includes weather, venue, and referee data. Another source for obtaining the needed data will be fantasydata.com, which has defensive and offensive rankings for every team in each contest, and data from sports books, which includes the totals for each game. Additionally, fantasydata.com has every relevant fantasy stat needed for individual players.

## Data Collection

To collect the data, I built two different data scrapers using Python's BeuatifulSoup4 library. The first scraper was for pro-football-reference.com. This scraper collects data from each individual game played since 2010. The variables collected in this scraper include the game ID, date, teams, metrics that show how strong a team's offensive, defensive, and special teams performance was, and team statistics such as first downs, turnovers, passing yards, time of possession, etc. The second data scraper collects odds data from fantasydata.com. The odds data includes data from every game dating back to 2010. The variables collected include the data of the game, who is favored to win the game, how many points the favorite is favored to win the game by (spread), who the underdog in the game is, the total number of points expected to be scored by both teams combined (total), the moneyline for both the home and away team, the season, and the week. We also will be using fantasy statistics for every player from every game dating back to 2010 using fantasydata.com. Additionally, we will be using data that we collected in the first capstone project, which includes team grades from every game from profootballfocus.com, venue data, and footballoutsiders.com data which measures teams' offensive, defensive, and overall efficiency.

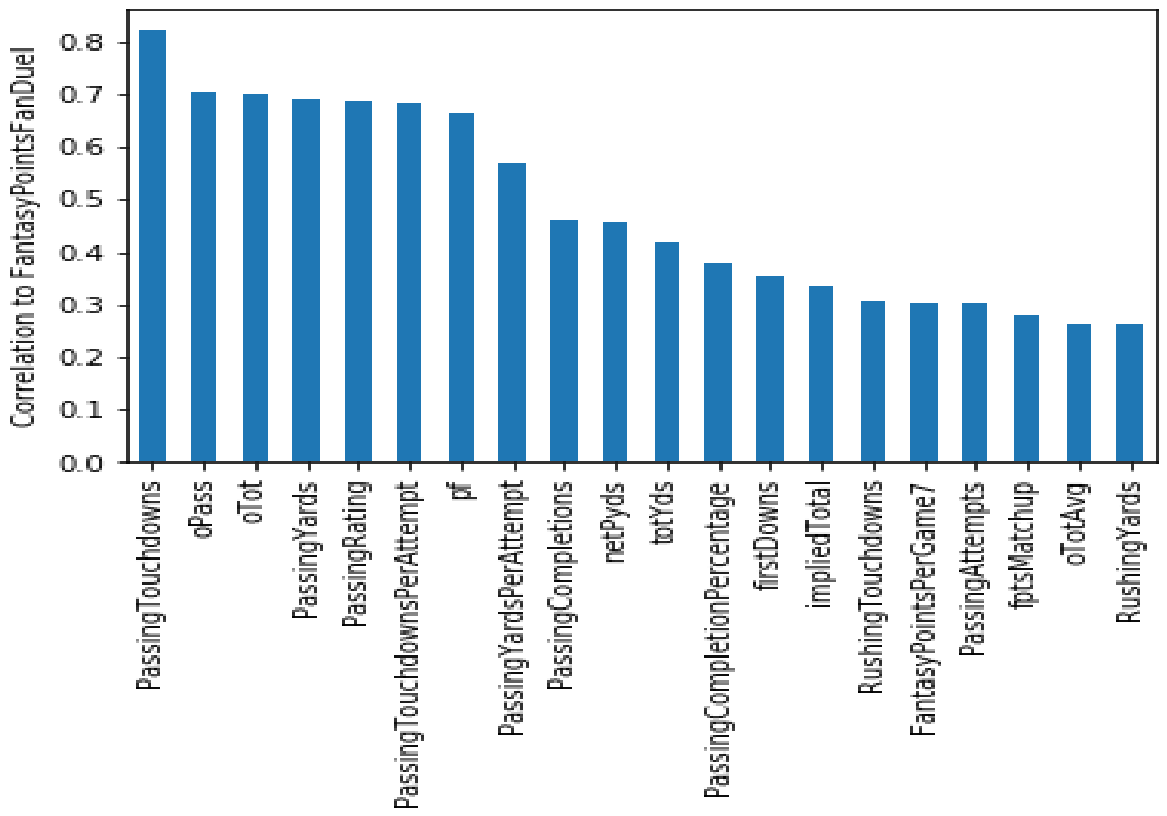
## Data Wrangling

For data wrangling, many techniques were used to get the data in a usable format for analysis and modeling. I used Python's Pandas library for a lot of this process. Using read_csv(), the data that was scraped from the .csv files were put into a Pandas dataframe. Column operations were performed on the dataframe to create new variables that could be useful for analysis and modeling. For example, a variable named FantasyPointsPerGame7 was created by averaging each quarterback's fantasy points over his last 7 games going into his upcoming contest. This was done by setting the index for the dataframe by date and name, grouping the dataframe by name, and using a rolling average method for each quarterback in the dataframe. This process was completed for many different variables in which a moving average would be useful, such as passing touchdowns over the last seven games, passer rating over the last seven games, etc.

Furthermore, date columns were dealt with by transforming them into datetime objects, string methods were used on columns to clean string variables, team name columns needed to be cleaned for consistency among all tables, so string methods and dictionaries were used to create one consistent variable. Joins and merges were used on tables to prepare the data for analysis and modeling.

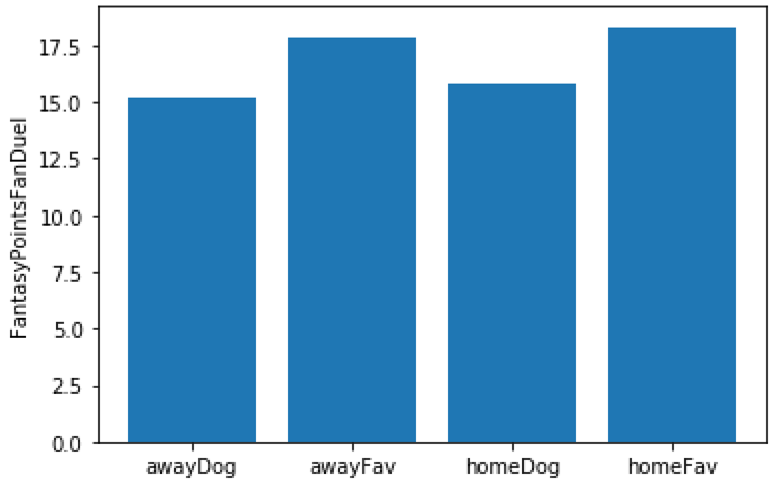
## Exploratory Analysis

Now that we have the wrangling process out of the way, we can begin digging into the data to find trends and patterns. We need to keep in mind our main goal, which is to predict fantasy points for quarterbacks. With this in mind, let's first take a look at some correlations to a quarterback's fantasy points.



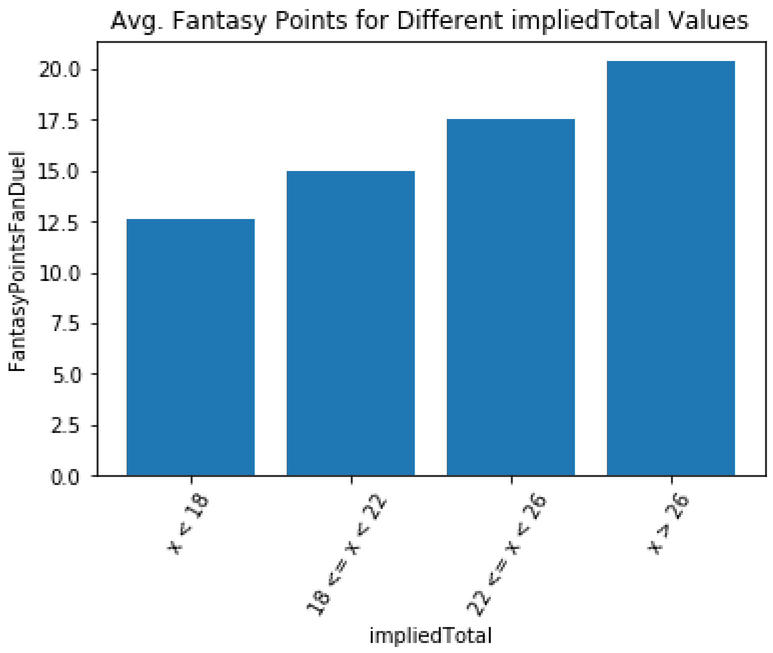
Looking at the bar graph above, we can see that passing touchdowns have the strongest correlation to fantasy points for quarterbacks. The next tier of correlations includes oPass, which is a measure of the quarterback's team's offensive points gained via passing plays, oTot, which is a measure of a qb's team's offensive points gained by the team's offense as a whole, passing yards, passer rating, touchdowns per attempt, and pf, which is a team's total points scored. We notice that PassingAttempts doesn't crack the top 15 correlations to fantasy points. A common point of emphasis among the fantasy community is that quarterbacks who throw the ball lots of times will score more points. While there is some truth to this, it may be a better strategy to target quarterbacks who are projected to be more efficient, rather than targeting qb's who are projected to throw a lot.

A good way to target efficient quarterbacks is to look at the odds data. The 'total' variable indicates how many combined points two teams are projected to score via the Las Vegas sports books. The 'spread' variable indicates how many points the team who is favored in a game is expected to win by multiplied by -1. The spread will be negative for a team who is favored to win because this is essentially a 'handicap' for the better team. If a team is favored to win by 8, the spread variable will be -8. These sports books are incredibly good at setting the totals of games, as well as spreads, taking bets from both recreational bettors and professionals on the totals and spreads, and turning a profit. We can leverage this information to our advantage when targeting quarterbacks for our lineups.



As we can see from the above image, quarterbacks who are favored to win average more fantasy points than quarterbacks who are underdogs, regardless of whether they are playing a home or away game. Being a favorite means that the spread < 0 for a quarterback. In this scenario, quarterbacks average approximately 17.6 fantasy points per game, as opposed to underdog quarterbacks, who average only around 15 fantasy points per game.

The impliedTotal variable takes into account both the spread and the total, and has an even higher correlation to fantasy scoring than the total. Implied total for a quarterback is calculated by taking (total - spread)/2. For example, if a team is favored to win by 7 in a game with a total of 48, the implied total for that team's quarterback is (48 - (-7))/2 = 55/2 = 27.5. This team is expected to score 27.5 points in this particular game. The underdog in this game will be expected to score 20.5, since they are 7-point underdogs. Notice that adding 27.5 and 20.5 gives us our 48-point total. Now that we have a better understanding of impliedTotal, let's look at how we can use it.



Looking at the graph above, we notice that as impliedTotal increases, a quarterback's fantasy output also increases. When a quarterback's implied total is < 18, he should not be a target in our fantasy football lineups, as quarterbacks in this scenario only average 12.5 points per game. Conversely, quarterbacks with an impliedTotal of $> 26$ should be targeted regularly, as they average over 20 fantasy points per game. This information is valuable, since most fantasy football players target quarterbacks in games with a high total instead of taking both total and spread into account. The 'total' variable should be considered, but it can be argued that 'impliedTotal' should be weighed more heavily.

The impliedTotal variable is the highest correlated variable to fantasy points among variables that are known before the game begins. Passing touchdowns correlate highly to fantasy points, but we don't know how many passing touchdowns a QB throws in a game until after the game is over. The top five correlations to fantasy points among variables we do know before the game is over are impliedTotal, FantasyPointsPerGame7, fptsMatchup, oTotAvg, and oPassAvg. The Pearson Correlation Coefficients for the variables with respect to fantasy points are .333, .304, .277, .263, and .258, respectively.

**Model Building**

Exploratory analysis was helpful in getting a feel for which variables could be important when proceeding with the model building process. Now that we have a better understanding of our data, we will proceed with building models. For this project, we will be using three different machine learning algorithms to predict fantasy points for quarterbacks: XGBoost, Random Forest Regressor, and Lasso Regression. Once we have built each of the three models, we will be comparing our results to projections from FantasyData.com. Many people subscribe to their website and rely on their projections for making their lineups. If we can outperform their projections, we can build a solid case for why people should use our projections instead.

There are a number of packages we will need to complete the model building and evaluation process. The used packages include Pandas, numpy, matplotlib, sklearn, random, and xgboost. We will split our data into training and test sets using a 75/25 split. We have 78 total input variables, and our target variable is FantasyPointsFanDuel. We have 3,646 instances in our dataset. Due to the nature of this project, we will not use a random split of the data, but will instead use a split that ranges from the 2011 season to the end of the 2016 season for training, and will use the beginning of the 2017 season to the end of the 2018 season for testing. The reasoning behind this decision is that we will be able to compare our projections for a given week of NFL data in 2017 or 2018 to fantasydata.com's projections, which will allow us to compare weekly results to see if we would have made correct decisions for our fantasy lineups. Given this decision, we end up with around 912 instances for our test set, and 2,734 training instances.

**Baseline Score**

We first want to set a goal for our models to outperform. We will use mean squared error to evaluate how our regression models perform, since our target variable is fantasy points, and any outliers in the data (e.g. a very high-scoring quarterback on a particular week, or a very low one) are values we still care about. We want our evaluation metric to penalize outliers, since the cost of playing a quarterback who scored a low amount of points basically guarantees that we will have a losing lineup on most weeks. Conversely, not playing a quarterback who scores a very high amount of points is something we also want to penalize. Mean squared error is the preferred metric for the goals of this project.

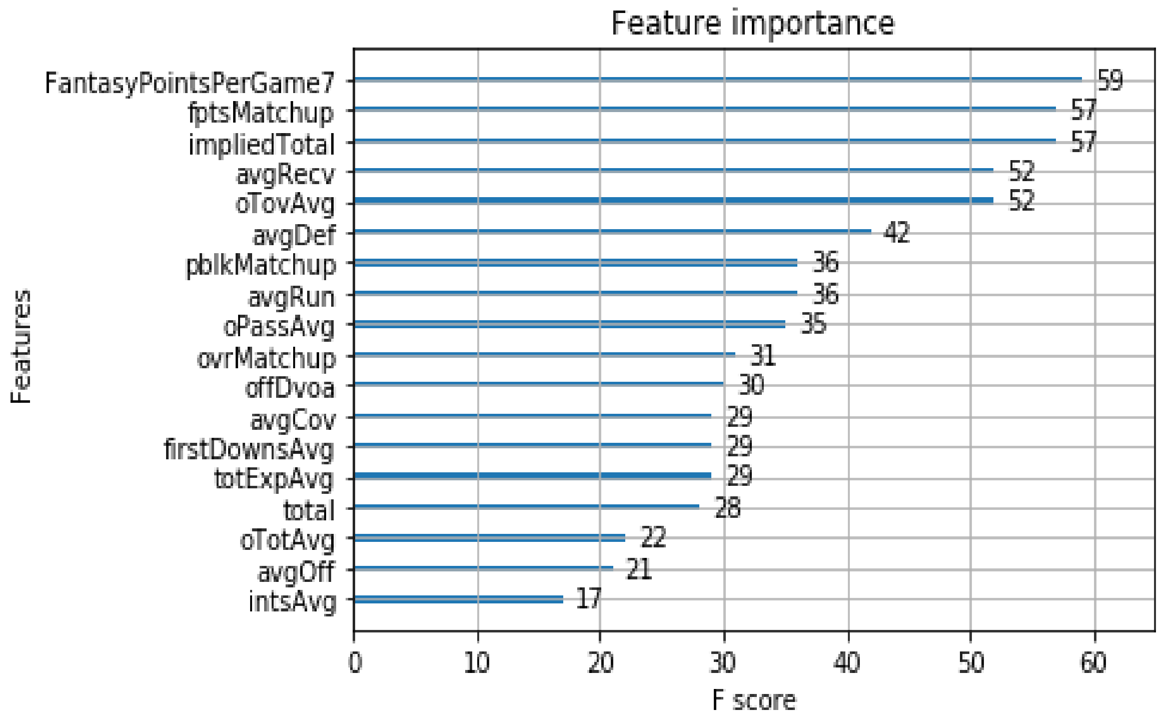
Using sklearn.metrics, we import the mean_squared_error method, and merge the projections data we collected from fantasydata.com with our test dataframe. We use the mean_squared_error method on y_test and the projections to achieve a MSE score of 56.456. This is the score we will aim to beat moving forward.

**XGBoost**

The first model we will build will use the XGBoost algorithm. We begin by fitting the training and testing data and using the XGBResgressor method. Once the model is fitted, we cross validate our training data using the model model to check for overfitting. Our MSE scores from 10-fold cross-validation range from 36.720 to 56.873, so we have no overfitting issues. Using our test data and predictions from the model yields an MSE score of 54.68. This beats the benchmark score of 56.456 from the fantasydata.com projections.

***XGBoost Improvements and Feature Importance***

To improve the model, we take the most important features from the already built model, and rerun the model. We use the top half of the features from the previous model to build the next model, and repeat this process until we get the lowest MSE possible. Once we find the model with the lowest MSE, we can use GridSearch for parameter tuning to improve the model further. Proceeding with this process, we find that the model with the lowest MSE has a total of 18 features. We use GridSearch to find that the parameters we should use are n_estimators: 100, max_depth: 3, min_child_weight: 4. Using these parameters, we are able to improve our model, as we get an MSE of 54.301.



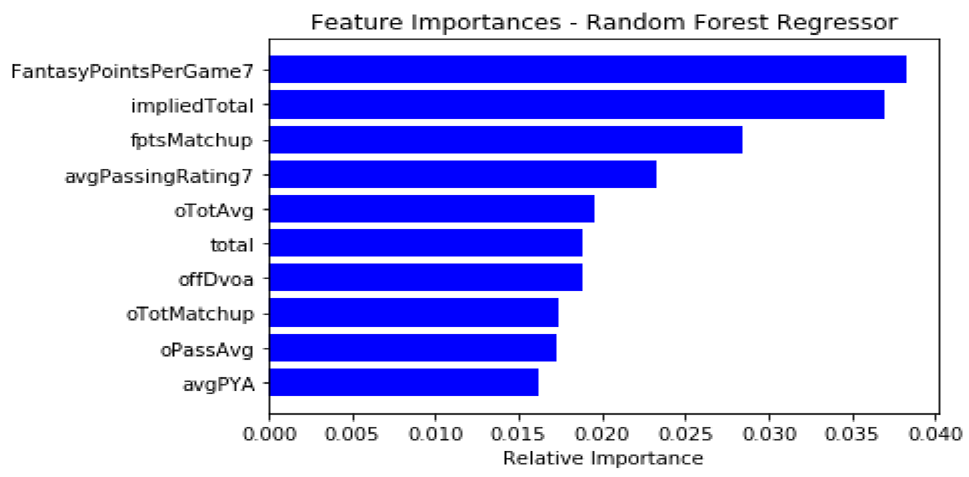
Using the plot_importance method from the xgboost library, we can take a look at the Feature Importance chart, which is shown above. FantasyPointsPerGame7, fptsMatchup, and impliedTotal rank at the top of the list. Recall that FantasyPointsPerGame7 is a quarterback's average fantasy points scored over his last 7 games, and impliedTotal is a QB's team's projected number of points scored for a game via the Vegas odds data. The fptsMatchup variable is how many fantasy points the opposing defense allows to QB's per game. The next two variables, avgRecv and oTovAvg, also high relatively high importance. The avgRecv variable is a measure of how skilled a QB's wide receivers are, and oTovAvg is how many turnovers a QB's team is averaging per game.  The importance of these variables suggests that a quarterback needs to have the backing of a strong Vegas impliedTotal, a good matchup for fantasy quarterbacks, and should average at least a decent amount of fantasy points per game for us to consider playing him in our lineups. Furthermore, a quarterback needs to avoid turnovers, and have strong receiving weapons at his disposal.

## Random Forest Regressor

Next, we will be trying the Random Forest Regressor algorithm. We set up our test/train split in the same way, and use parameter grid for the GridSearch method. The parameters we tune and values they may take on are n_estimators: [1, 10, 100, 1000], max_features: ['auto', 'sqrt', 'log2'], min_samples_split: [2,4,8], bootstrap: [True, False]. We run grid search with this parameter grid on a Random Forest Regressor, and find that our best parameters for the model are n_estimators: 1000, max_features: 'sqrt', min_samples_split: 2, bootstrap: True. Running a model with these parameters yields MSE of 53.224, which outperforms both fantasydata's projections and our XGBoost model.

### ***Random Forest Regressor Feature Importances***

After using GridSearch to find optimal parameters and model, we plot feature importances for the model, as shown below:



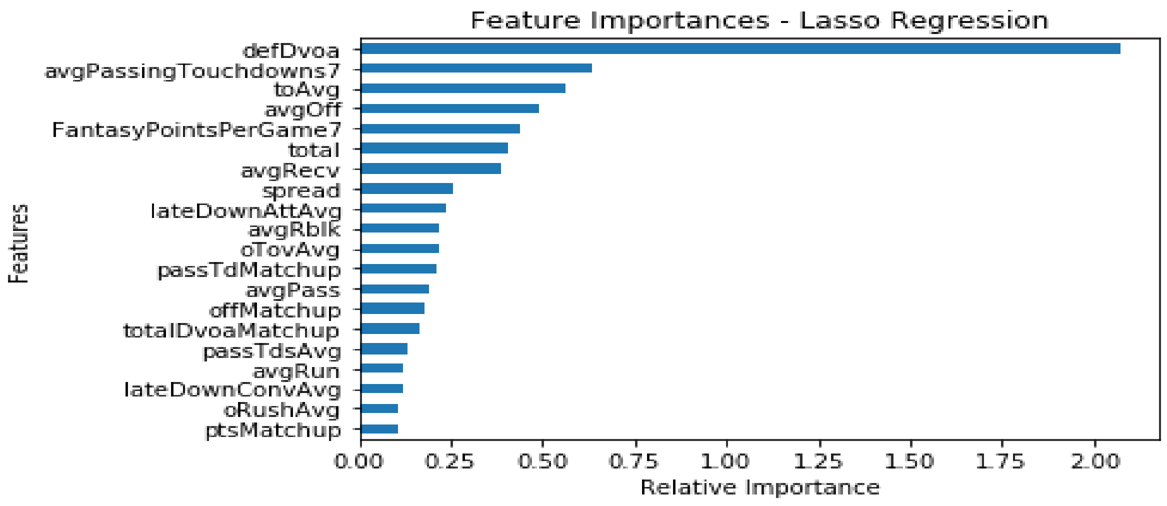
We notice that the same top three variables show up again: FantasyPointsPerGame7, impliedTotal, and fptsMatchup. Average passer rating, which is a statistic used to measure a quarterback's efficiency over his last 7 games, and oTotAvg, which is a metric that measures a QB's team's overall offensive performance over the last 7 games, also show up in the top five variables in terms of importance.

**Lasso Regression**

For the final model, we will use Lasso Regression. We begin by fitting the training data to a lasso model. We will use different values for the alpha parameter, and choose the model that yields the lowest MSE. After testing for alpha = 1, 0.01, and 0.0001, we find that the best model is when alpha = 0.01. The MSE for this model is 51.296, which gives us our best performing model of all.

***Lasso Regression Feature Importances***

The lasso regression model we built ge=ot rid of all the features it did not see as important. The model uses 48 variables, which means it got rid of 30 variables. Of the features the model used, we plot the absolute value of the feature importances, since some will be negative. The results are as follows:



We can see that defDvoa, which is a measure of the strength of a QB's opponent's defense, is by far the most important feature in the Lasso model. Variables that come up in the top importances again include toAvg, FantasyPointsPerGame7, and avgRecv. Some other variables that show up in the top importances include avgPassingTouchdowns7, avgOff, which is a measure of how well a QB's offense performs as a whole, and total, which is the vegas total for the game, as mentioned earlier.

**Key Takeaways**

After building three different models for predicting fantasy points, we found that using Lasso Regression gave us the lowest MSE of 51.296. Each one of the models that we built was able to outperform fantasydata's projection system. Using our projections, particularly those created from using our Lasso model, will yield better long-term results than using fantasydata's. From looking at feature importance scores of all of the models, it is clear that using QBs' average fantasy points over his past 7 games is critical in projecting future performance. This variable showed up in the top 5 most important features for all three models. Also, using Vegas data is very beneficial in creating projections, as some combination of spread, impliedTotal, and total popped up in all three models as important features. A QB's fantasy matchup, or how bad the opposing defense is at defending against QB's is usually a good indicator, as well as how good his wide receivers are.

**Future Work**

Predicting how quarterbacks will perform in daily fantasy football contests is arguably the most important part of a winning player's lineup. If your quarterback doesn't perform, your lineup will likely be losing you money for the week. Picking a quarterback usually leads you to pick a wide receiver on the same team, since the two positions are highly correlated. For example, if a quarterback throws a touchdown pass to a wide receiver on his team, both the quarterback and the receiver will receive fantasy points for your lineup. Therefore, picking the "right" quarterback will often lead to picking the "right" receivers, and other positions that correlate. For our future work, we will explore the data for running backs, receivers, tight ends, and defense/special teams, and build models to predict fantasy points for all of these positions. We hope to have similar success in outperforming paid-for projection systems as we did for the quarterback position.