|  |
| --- |
| Project: Fantasy Football Team Optimizer  /Users/roderickhead/Documents/Indiana University/Fall 2015/INFO I590 Big Data Applications and Analytics/Fantasy Football Project/ff.jpeg |
| Brett Steele, Matt Morris, & Roderick HeadINFO 590 - 34717December 18, 2015 |

# Define Problem

The most challenging aspect of owning and operating a daily fantasy football team is not having a great deal of football knowledge; it is determining the best lineup to use for the upcoming week. Many experts weigh in with their weekly projections based on historical player performance and how they perceive the player will do given their current health and their opponent. Many of these projections rely on player and football knowledge. Player and football knowledge is still necessary when determining lineups, but we wanted to pursue a more data-driven approach to building a weekly lineup. Our team sought to tackle two main problems when creating the optimal lineup:

1. Create a data-driven weekly player performance projection model based on historical data available.
2. Optimize the weekly lineup by choosing players with the highest performance projections while keeping the team under a given salary cap.

The first problem is a classic regression problem using historical data to predict future performance. The output of this model, predicted fantasy points, is a necessary input into the second part of the problem.

The second problem is necessary because weekly fantasy football-like sites institute a team salary cap that must not be exceeded when building a team. They also have a requirement on number of players at each position. This is a problem with combinatorial optimization often referred to as the “knapsack problem with multiple constraints”. Our team followed the FanDuel.com format. Each week, every player is given a salary. Better players have higher salaries. The crux of the second problem is to take the projections from the first model and create a team that produces the highest number of projected fantasy points while remaining within the salary cap and position constraints.

# Define Software

This project used both R (version 3.2.2) and Python (version 2.7.9) for the analysis. Below is a list of packages from each that are required to be installed in order to run the analysis:

* *R*: dplyr, Rglpk
* *Python*: numpy, pandas, math, requests, re, io, sklearn, BeautifulSoup, datetime

Python is used for web scraping, data wrangling, and machine learning. Data is pulled from the web using the BeautifulSoup package, and that data is manipulated using Python lists, dictionaries, pandas dataframes, and numpy arrays. The sklearn machine learning package is then used to train several regression models on the data to create player performance projections. Regression metrics such as mean squared error, mean absolute error, median squared error and R-squared, are used to evaluate which model is best to use to create the projections.

R is used to do the linear optimization problem to find the optimal lineup given the projections and lineup constraints. The projections from Python are pulled in and the constraints are defined. The Rglpk package is then used to run the linear optimization and output the optimal lineup.

# Documentation

Github Project Location: <https://github.com/brttstl/proj-fantasy>

*Part 1: Web Scraping*

Script Location: <https://github.com/brttstl/proj-fantasy/blob/master/1.projections/projections.py>

This script scrapes several web pages for data input into the regression model. Below are the sources it pulls from:

* <http://espn.go.com/nfl/statistics/team/_/stat/>: Historical (from current season) team aggregate data. Used for opponent team statistics.
* <http://www.foxsports.com/fantasy/football/commissioner/Players/DepthCharts.aspx>: Current week depth charts. Used to determine where a player fits on the team’s roster. Below shows which players are considered in this model:
  + Quarterbacks: only starters
  + Running backs: starters and backups
  + Wide Receivers: WR1, WR2, WR3, and reserves
  + Tight Ends: starters and reserves
  + Kickers: only starters
* <http://rotoguru1.com/cgi-bin/fyday.pl>: FanDuel.com historical player performance and salary. Player performance used as target variable in projection models.

*Part 2: Projections Model*

Script Location: <https://github.com/brttstl/proj-fantasy/blob/master/1.projections/projections.py>

External data inputs:

* NFL stadium attributes: <https://github.com/brttstl/proj-fantasy/blob/master/data/stadiums.csv>
* List of the next week’s available players (week to project for). In this case, it is week 15 of the 2015 NFL season: <https://github.com/brttstl/proj-fantasy/blob/master/data/FanDuel-NFL-2015-12-20-13996-players-list.csv>

Output: Next week’s projections in a csv format to folder set by user in one of the first lines of code within the script.

How it works:

1. *Gather user input.* In the first several lines of code within the script, the user inputs the week to create projections for and the file path of where the csv will be saved. It is worth noting that the projections can only be created for weeks that FanDuel has the player and salary list published. Typically, this is the upcoming week of the season. No data is available for weeks that are far off.
2. *Scrape the web.* The script scrapes ESPN.com for offense and defense season statistics for each team. These statistics are used as predictors for a player that is matched up against that team. For instance, a running back that is facing the Chicago Bears will have various Chicago Bears rushing defense statistics passed through the model when determining that player’s projected points for the week.
3. *Clean and merge datasets.* The datasets that were retrieved from scraping within this script and the datasets that were pulled from Github require some cleaning in order to remove columns that are not required in the model as predictors and to clean up variables that will be used to join the datasets. Additional columns are also appended. For instance, whether the player is at home or away is determined based on which team they are on and the location of the game.
4. *Prepare training and test sets for regression model.* A training set that matches 70% of the data available for each player is created. The remaining data is used on the test set.
5. *Run several different regression models.* In this analysis a gradient boosting machine and a generalized linear model were both tested. The parameters of the gradient boosting machine were tweaked for several different trials to optimize this model. A different model was created for each different player position. The models were run on a training set that was comprised of 70% of the available data for each player, and the test set was the remaining 30% of the available data.
6. *Evaluate models.* Metrics were used to determine which model performed the best at creating projections. The metrics reviewed were mean absolute error, mean squared error, median absolute error and R2. A chart with that compares the mean squared error between the different models can be found at the end of this report.
7. *Run best model on data to create projections.* Based on the metrics evaluating the different models, the gradient boosting machine with the following parameters was deemed the best model:
   1. n\_estimators = 250
   2. max\_depth = 3
   3. min\_samples\_split = 1
   4. learning\_rate = 0.01

This model was used on all of the available data to create player point projections for the next week. Once again, a different model was created for each player position.

1. *Append statistical measures of multiple iterations of model to output dataset.* In this analysis, the determined best model was run 10 times. The number of iterations can be manually changed by changing the value of the variable ‘iterations’ within the script, though 10 was determined to be sufficient. The projections of each iteration were included, and the mean, median, and standard deviation of projections for each player were appended. How these are used is detailed in the section about the lineup optimizer script.
2. *Write output projections to csv file.* The final dataset with players, projections, and statistical measures is written to a csv file at a location specified by the user. This csv file is then read into the lineup optimizer script.

*Part 3: Lineup Optimizer*

Script Location: <https://github.com/brttstl/proj-fantasy/blob/master/2.lineup_optimization/optimal_lineup.r>

External data inputs:

* List of the next week’s available players (week to project for). In this case, it is week 15 of the 2015 NFL season: <https://github.com/brttstl/proj-fantasy/blob/master/data/FanDuel-NFL-2015-12-20-13996-players-list.csv>
* Projections from projections.py: <https://github.com/brttstl/proj-fantasy/blob/master/data/week_15_projections.csv>

How it works:

1. *Set path for csv output.* This is a file location where an optimized lineup.csv will be generated.
2. *Import your league’s player pool.* This is an exportable csv of for eligible players in a given league and is available when you join a FanDuel league. For this exercise, the file is located here: <https://github.com/brttstl/proj-fantasy/blob/master/data/FanDuel-NFL-2015-12-20-13996-players-list.csv>
3. This script manually removes outliers such as backup players with high projections and sets the metric to be optimized against given the salary. For this program, we are optimizing on projected fantasy points, i.e., we are using $60,000 salary cap to produce the lineup that will score the most points.
4. We define constraints, which in the case are specific to position limits, number of total roster spots, and total salary available.
5. Using the R library, Rglpk, we call the function Rgplk\_solve\_KP which is a high level R interface for solving linear and MILP problems.
6. This includes the following arguments:
   1. obj - the objective, in this example, projected points
   2. mat - a matrix of constraint coefficients (positions / roster spots, salary)
   3. dir - character vector of constraint directions
   4. rhs - right hand side of constraints, i.e., the values
7. *Write output to csv based on path from step 1.* This will produce the optimal FanDuel lineup that contains a lineup of 1 QB, 2 RB, 3 WR, 1 TE, 1 PK, 1 DEF set to score the most points that week.

## Instructions

Below are instructions on how to run all stages of the program from start to finish. The end result will be: 1) an optimal fantasy football lineup based on FanDuel.com scoring and salary system and 2) a list of all of the players for the current week from FanDuel.com with projections based on output from the regression model.

1. Install all necessary Python and R packages (see package list near beginning of report)
2. Download FanDuel.com’s latest player list: FanDuel.com → login (or create username if necessary) → Click first contest ($1.75M NFL Sunday Million) → Enter this contest → Click “Download players list” link near the top middle of page.
3. Upload the players list downloaded in step 2 to Github.
4. Open python file <https://github.com/brttstl/proj-fantasy/blob/master/1.projections/projections.py>. Go to lines 5-8 of the code and update these user inputs:
   1. Line 5: NFL week to create projection for (week 15 in example)
   2. Line 6: NFL season (2015 in example)
   3. Line 7: Input file: This is the Github location of the file uploaded in step 3.
   4. Line 8: Filepath: directory location of where output csv should be saved to.
   5. Run script.
5. Save output file to Github.
6. Open R file <https://github.com/brttstl/proj-fantasy/blob/master/2.lineup_optimization/optimal_lineup.r>. User inputs are required on 2 lines of the code:
   1. Line 9: Output path: Designate directory of output csv file.
   2. Line 12: Change location of file being read to be the location of the file saved in step 5.
   3. Run script.
7. The optimal lineup based on projections is now saved as a csv file to the location specified in step 6a.

## Future Improvements

While the model is a useful tool for the problem at hand, there are ways that it can be improved, given more time and effort. Most of the ways to improve the model would be to build additional knowledge into the projections model that aren’t currently being captured.

First, the model does not take into account the number of games a player has played. For instance, a player might be averaging a high number of points per game, but they may have only played 2 games, while another player is averaging slightly less points but has done it over 12 games. The model currently treats these situations equally.

The model also does not account for the recency of a player’s performance. For instance, it might make sense to assume that a player that has been performing well recently will continue to perform well as opposed to a player that had a hot start but has cooled down more recently. The model treats all weeks equally. Given more time to work on the model, it would make sense to build in some sort of adjustment for the recency of performance.

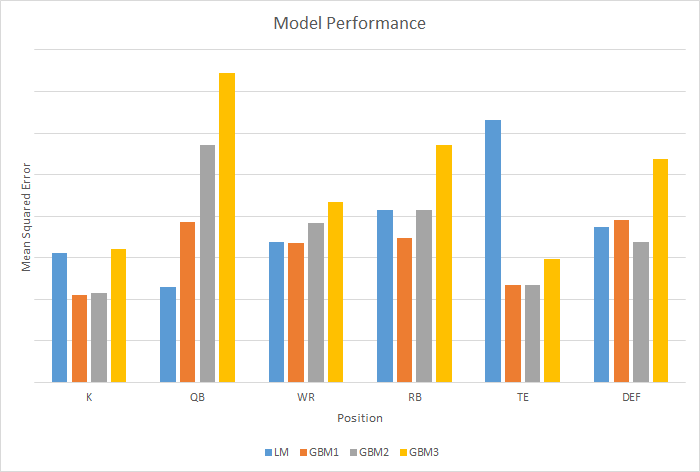
The model does not attempt to hedge the manager’s risk like one would when choosing stocks. For instance, several players from the same team or a wide receiver/quarterback combination could be suggested by the optimizer. In reality, it may be risky to choose several players on the same team. The correlation between their performance may be strong if it is a wide receiver/quarterback combination. A future iteration could attempt to diversify players to hedge the manager’s risk.

There are a number of ways to look at this problem numerically, and no model is perfect. However, it would be interesting to take into account the variance of player performance when creating performance projections. This might be choosing a player that is projected at 1 less point than another because that player’s historical point variance is so much lower than the player with the higher projection. The model currently uses point estimates, but in reality these are confidence intervals. A Monte Carlo simulation could be used to create confidence intervals and propose the best lineup based on these.

Lastly, the perfect model would take football knowledge into account. This would include; a player is still playing but they have been nursing an injury, a quarterback is expected to leave the game early to rest for the playoffs, two quarterbacks are battling for the starting job, and more. Even after several iterations, it would be very hard to model this. At the end of the day, this model can get a manager a long way, but they still may have to use their football knowledge to make adjustments.

# Figures

1. Model Performance



Below is a summary of each model that is used in the above chart:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Description | n\_estimators | max\_depth | min\_samples\_split | learning\_rate |
| LM | Generalized Linear Model | NA | NA | NA | NA |
| GBM1 | Gradient Boosting Machine | 250 | 4 | 1 | 0.005 |
| GBM2 | Gradient Boosting Machine | 500 | 4 | 1 | 0.01 |
| GMB3 | Gradient Boosting Machine | 1,500 | 10 | 1 | 0.1 |

II. Process Flowchart

