CS221 Final Paper: Fantasy Football Draft Bot

Jose Suarez-Rodriguez, Oluwasanya Awe

December 16, 2016

Abstract

This paper explores the details of a Fantasy Football draft bot. The bot was created using Machine Learning and Stochastic Gradient Descent to first predict the estimated FanDuel points for each player in the league and then modeled the draft selection as a Constraint Satisfaction Problem (CSP). The driving algorithm behind solving the CSP was backtracking with some pruning to reduce search space of the factor graph. The draft picker was able to match our Oracle (human draft picker). On average, the draft bot returned a value of 105.7 points with a standard deviation of 20.0. This is a decent result given that a score of 100 points is sufficient to make monetary gains on the Fantasy Football game.

1 Fantasy Football Primer

A daily fantasy football contest consists of a player matched up against the number of players allowed by the contest (e.g. 999 other players). At the start of each week during the NFL season, each player selects players that will make up her/his fantasy football lineup. There are 9 total positions, 1 Quarterback, 2 Running-backs, 3 Wide Receivers, 1 Tight End, 1 Kicker and 1 Defense/Special Teams.

Each fantasy site uses its own specific scoring system, but they are fairly similar. We used the FanDuel scoring system to evaluate our team picker. Fantasy football players gain points based on how they perform in real life. For example, if Andrew Luck throws a touch down pass, his fantasy player will get 6 points. The objective of the fantasy contest is to score as many points as possible by identifying which players will have high scores for a given week. Participants must consider factors such as if the fantasy player is at home or away, whether he has played well in the recent weeks and the quality of the opponent the fantasy player is facing. Furthermore, a fantasy football team must stay under the set salary cap. A player's salary is usually determined by the favorability of their matchup that week.

2 Problem Definition

The aim of this project is to use artificial intelligence techniques to predict the optimum fantasy football lineup. On most daily fantasy football sites, each player has a set salary and there is a team salary cap. The sum of the salaries of the players on a given team must stay below the salary cap. The objective of a fantasy football game is to score more points than your opponent. Each category of player (e.g. Quarterback, Running Back...) receives points based on a different scoring function. This project will predict the team of players that will score the highest number of points while staying under the salary cap.

3 Literature Review

The problem of Fantasy Football draft pickers is a fairly popular one and a variety of papers exploring different approaches have been written. The primary paper that influenced our work was by Abdulhammid et.al.[1]. The project on which this paper was written made use of a Constraint Satisfaction Solver as well as a neat trick in using player-point predictions.

Additional Papers which stimulated thought but were more technical in implementation were Becker and Sun's *An analytical approach for fantasy football draft and lineup management*[2] while

[1] http://adamabdulhamid.com/CS221.pdf

[2] https://www.degruyter.com/downloadpdf/j/jqas.2016.12.issue-1/jqas-2013-0009/jqas-2013-0009.xml

others discuss heuristics for the player selection like by *A Player Selection Heuristic for a Sports League Draft* Fry et. al[3].

In addition to these, Fantasy Football Draft selection problem has a similar structure as Constrained Portfolio Optimization where there are certain constraints on assets and the objective is to choose an optimal selection of assets under these constraints. The material involved in tackling this problem was beyond the scope of the project.

4 Setup

4.1 Input-Output Behavior

Our system takes in the following input parameters and return a list of players that will maximize the predicted score for the next match:

4.2 Input

1. Players (**Set**): This is a set of player objects where each object contains all the information necessary for the system to uniquely identify a player. This object will consist of the following fields:
   - Player ID (**String**): This ID is generated by some function that maps each player to a unique ID e.g. jwint01.
   - Position (**String**): This gives the position of the player as a 2-character string e.g. Running Back [RB], Quarter Backs [QB] etc.
   - Salary (**Double**): The current salary of the player.
   - Stats (**Dictionary**): Stats of the player from the beginning of the year.
2. Salary Cap (**Double**): This is a salary cap of the sum total of players in our chosen.
3. Player Positions (**Dictionary**): This is a dictionary of the form (position, number) that gives the number of players expected for each position in our output. For example.

```
{  players: [(Jameis Winston,jwinst01,QB,8200),(Andrew Luck,aluckx11,QB,8000),
   (Antonio Brown,abrown11,WR,9500),(Doug Baldwin,dbaldw13,WR,8600)],
  salary_cap: 20000, positions: {QB:1,WR:1} }
```

4.3 Output

This returns a list of the players of the optimum team. For example, given the input above, our system could return `[Jameis Winston", "Antonio Brown"]` as an optimum lineup.

4.4 Data

We retrieve data from sites like rotoguru.com and fantasydata.com. We also make use of the `nflgame`[4] Python library for additional player stats.

5 Approach

We initially tried to solve the problem using a Q-Learning policy but upon reading more, we realized it would be easier model the problem as a Constraint Satisfaction Problem (CSP). Necessary to this was being able to estimate the expected value of the players. We started off using a weighted prediction value from ESPN and then built a linear regression solver to estimate a player's value based on their previous stats.

5.1 Modeling

We modeled our fantasy football draft picker as a CSP. This was one of the more intuitive ways of modeling the problem since we wanted to find that lineup that maximized the expected number of points by each player while being under the salary cap constraint of $60,000. We discuss the detail of the model below.

[3]https://people.emich.edu/aross15/math319/player-selection-heuristic-sports-league-draft.pdf
[4]https://github.com/BurntSushi/nflgame

5.1.1 Variables

Based on FanDuel's Fantasy Football rules, we needed to pick a team of 9 players and this corresponded to having a CSP with 9 total variables (one for each position). These positions are: **(Quarterback, 1), (Runningback, 1), (Runningback, 2), (Wide Receiver, 1), (Wide Receiver, 2), (Wide Receiver, 3), (Tight End, 1), (Defense/Special Teams, 1) and (Kicker, 1)**. From now on, we will refer to these variables as QB, RB1, RB2, WR1, WR2, WR3, TE, D/ST and K respectively.

5.1.2 Domains

The domain for each variable is a set of player names. For each variable we restricted the search for the optimal team to contain only players in a domain's specific position under a given 'Game Year' and 'Game Week', where 'Game Year' represents an NFL season and 'Game Week' represents a given week in that season. This is appropriate since we are only generating a projection for a given week and year. When selecting the team for the given year and week, we have a dictionary that takes in a $(year, week)$ tuple and returns all of the active players for that week.

Given the potentially large search space of the Constraint Satisfaction Problem ($O(9^{200})$), we decided to prune the domain sizes for each position to contain only the best $8 \leq k \leq 16$ players. This is actually representative of how human fantasy football players choose their optimal team – they only consider the $k-$best players for each position.

5.1.3 Weights

We initially set the weight for each player value as the score projection divided by the salary (which will be divided by some normalization constant). We obtained these projections from FantasyData.com which provides weekly fantasy football projections.

The reason for dividing the projection by the salary is to find a good balance between salary and projection. A high projection is good but not if the player has an excessive salary. It is much better to have a player with a lower salary at a given normalized projection than a more expensive one. This allows us to find a team that matches up cheap finds with some more expensive players in other places rather than simply going for a team of just expensive players. The score projection is the number of points that player is expected to score that week. This projection takes into account factors such as home vs away, opponent and past/recent results. This strategy of dividing the salary by a factor was inspired by a previous CS221 project involving fantasy football by Adam Abdulhamid[5].

Formally, let W_p be the weight of a given player, p. We have that

$$W_p = \frac{\text{Projected Score}}{k \cdot \text{Salary}}$$

where $800 \leq k \leq 1000$ is a normalization constant and Salary is the salary earned by player p for that given week.

The reasoning behind this is that we want to give an advantage to a player that is cheaper but still has a high projection. This saves additional room in the salary cap and allows us to get even more highly projected players. We will order our valid line ups by weight and choose the lineup with the highest weight.

Instead of using the Projected Data, we also considered the alternative of building a predictor that projects the expected weights of a player. For a more detailed explanation of the features chosen for the feature extractor, consult the appendix. Based on player stats, we extracted features such as *Rushing Yards and Touchdowns, Receiving Yards and Touchdowns, Interceptions, Opponents Faced, Location (home/away) etc.*. We used Stochastic Gradient Descent (SGD) to learn the optimal weight, $\mathbf{w}$ such that for any player x and feature vector $\phi(x)$

$$\mathbf{w} \cdot \phi(x) = \text{Expected Player Score}$$

As a refresher, the Stochastic Gradient Descent algorithm, given a loss function goes as follows. For $(x, y) \in D_{train}$:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} Loss(x, y, \mathbf{w})$$

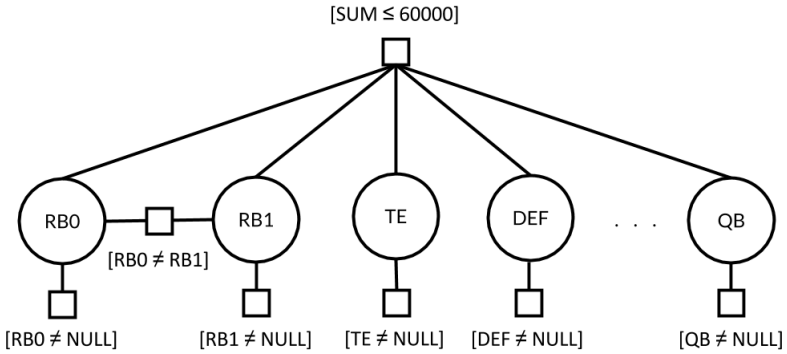
We will discuss the results of either choice of weight later.

[5] http://adamabdulhamid.com/CS221.pdf

5.1.4 Constraints

We had the following binary constraints on our variables:

1. $(RB1 \neq RB2)$ – These two variables cannot be the same player
2. $(WR1 \neq WR2)$ – These two variables cannot be the same player
3. $(WR2 \neq WR3)$ – These two variables cannot be the same player
4. $(WR3 \neq WR1)$ – These two variables cannot be the same player
5. Sum Constraint – The sum of the individual player salaries in the lineup does not exceed the FanDuel salary cap of \$60,000



A visual representation of the Constraint Satisfaction Problem showing some of the variables and constraints.

5.2 Algorithm

Algorithm: backtracking search

Backtrack$(x, w, \text{Domains})$:

- If x is complete assignment: update best and return
- Choose unassigned **VARIABLE** X_i
- Order **VALUES** Domain_i of chosen X_i
- For each value v in that order:
  - $\delta \leftarrow \prod_{f_j \in D(x, X_i)} f_j(x \cup \{X_i : v\})$
  - If $\delta = 0$: continue
  - $\text{Domains}' \leftarrow$ Domains via **LOOKAHEAD**
  - Backtrack$(x \cup \{X_i : v\}, w\delta, \text{Domains}')$

Generic Backtracking Algorithm presented by Percy Liang in CS221 Class.

The primary algorithm involved in choosing a lineup for the described factor graph is **backtracking search**. The backtracking algorithm attempts to find the top performing lineup. That is, this is where we are finding the lineup that we think will perform the best for a given week.

5.2.1 Variable Ordering

Now, we will describe the steps in the algorithm as they relate to the specific fantasy football factor graph. For this algorithm, we will assume a variable ordering of K, D/ST , TE, WR3, WR2, WR1, RB2, RB1, QB. The ordering of QB, RB1, RB2, WR1, WR2, WR3, TE, D/ST, K roughly represents the importance of a position to the outcome. That is, a quarterback has a much higher effect on a team than a kicker as they will produce much more points. This ordering is designed so that backtracking will try a wider range of players that will have large impacts on the score. Since backtracking won't be able to examine all of the possible lineups (state space is much too large) then it is better to try out more quarterbacks and runningbacks than it is to have the same quarterback and try out many different kickers and defenses.

5.2.2 Algorithm Details

We will now discuss the algorithm as well as provide examples at each step.

1. First, the backtrack call will check to see if the provided assignment is a complete assignment. That is, it will check to see if we have exactly one quarterback, two running back, three wide receivers, one tight end, one defense/special teams and one kicker.

2. If we do have a complete lineup assignment, we will add this lineup into a potential line up set which orders a team based on its total weight. That is, the sum of all player weights for each player in the lineup.

3. If the lineup is not complete, we choose a variable to assign. For example, we could choose to assign the kicker variable.

4. We then loop through all potential kickers that we want to consider. We could, for example choose Stephen Gostkowski of the New England Patriots and assign him to the kicker variable.

5. We then call backtrack with the domains reflecting that the kicker variable has been assigned. In the next level of backtracking, we choose a Defense, say the Minnesota Vikings Defense. Again, we call back tracking with the updated domains. This process continues for the Tight End variable, the three wide receiver variables and the two running back variables.

6. At the point where we get to the quarterback variable, a potential assignment might look like

```
{K:Stephen Gostkowski,
D/ST: Minnesota Vikings,
TE:   Greg Olsen,
WR3:  Julian Edelman,
WR2:  Doug Baldwin,
WR1:  Antonio Brown,
RB2:  Adrian Peterson,
RB1:  Devonta Freeman,
QB:   (Unassigned)}
```

7. At this point, we will loop over all possible values in the domain of the quarterback variable. Consider the following limited domain of {Tom Brady, Aaron Rodgers, Ben Roethlisberger}. First we would try Tom Brady and call backtracking with the assignment

```
{K:Stephen Gostkowski,
D/ST: Minnesota Vikings,
TE:   Greg Olsen,
WR3:  Julian Edelman,
WR2:  Doug Baldwin,
WR1:  Antonio Brown,
RB2:  Adrian Peterson,
RB1:  Devonta Freeman,
QB:   Tom Brady}
```

8. Since this assignment is complete, backtracking will calculate the total weight and store this in a potential teams set. Back tracking will then return, and we will try the next quarterback in the list. For example, we would then call back tracking with the assignment

```
{K:Stephen Gostkowski,
D/ST: Minnesota Vikings,
TE:   Greg Olsen,
WR3:  Julian Edelman,
WR2:  Doug Baldwin,
WR1:  Antonio Brown,
RB2:  Adrian Peterson,
RB1:  Devonta Freeman,
QB:   Aaron Rodgers}
```

9. This process would continue until we exhaust the list of potential quarterbacks. After we do this, backtracking will 'backtrack' up a level and choose a new RB1. We will then backtrack down and exhaustively search all quarterbacks, backtrack up and choose a new RB1 and so on until we have exhausted the RB1 domain. After exhausting that list, it will 'backtrack' up and try all RB2's in a similar style, searching all possible combinations at the lower levels of the backtrack. This process will continue 'backtracking' up until we get all the way back to the K variable. It will then choose a new kicker and restart the backtracking process all the way down the chain.

Note: Given the time constraint, we limit the search till we found the best 30 results

6 Results and Experiments

6.1 Evaluation

We used the FanDuel scoring system to evaluate our system. For a given week, we generate the optimum team and see how it performs given the FanDuel scoring protocols. If the team generates revenue in a paid contest, we will consider this a success. Our goal is to get above a **50%** success rate. In order to generate revenue in a paid contest, the team should score around **100** points or more.

6.2 Oracle-Trivial Comparison

6.2.1 Oracle Implementation

Given our collective football experience and knowledge of the players, we picked a set of teams for the 2014 season and compared our results against that baseline's for 2015 games.

6.2.2 Trivial Implementation

For the trivial implementation, we simplified the problem by considering only the players whose salary, $s_p \leq \frac{S_{team}}{N_{team}}$ where s_{team} is the salary cap for the whole team and N_{team} is the number of players in the draft team. We assigned the score based on their statistics and rules defined by FanDuel[6]. We trained our baseline model on data from 2014 and tested our model on the data for 2015. We used a greedy algorithm to choose the best player at each step of our selection. The results are put forward below.

6.2.3 Comparison

Week	1	2	3	4	5	6	7	8
Trivial	130.42	104.6	78.4	46.8	64.7	79.18	60.32	71.94
Oracle	170.76	187	135.5	94	132.2	150.46	128.84	149.12

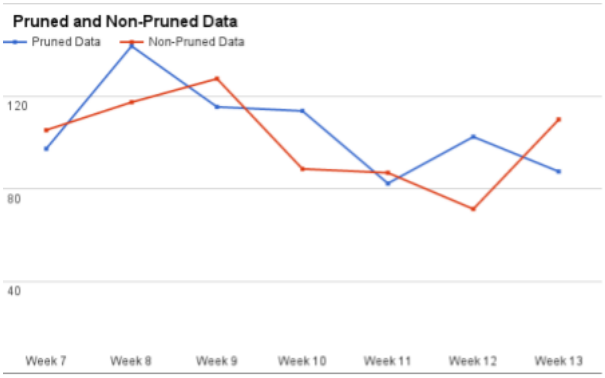
Table 1: Table showing first results of Oracle-Trivial Comparison. Higher score is better. The baseline would have earned revenue in two of the eight, since in weeks one and two the optimum team scored higher than the score cutoff for that week. Teams that score higher than the cutoff earn the prize. The oracle would have earned revenue in seven of the eight weeks for the same reason. The gap in success exists since the baseline is very simplistic and chooses a team once. It doesn't update the weights and make a new choice each week. The oracle considers matchups and trends to make an informed decision.

6.3 Draft Picker Results

The results below show the results of different experiments we performed in determining the best settings and optimization for our bot.

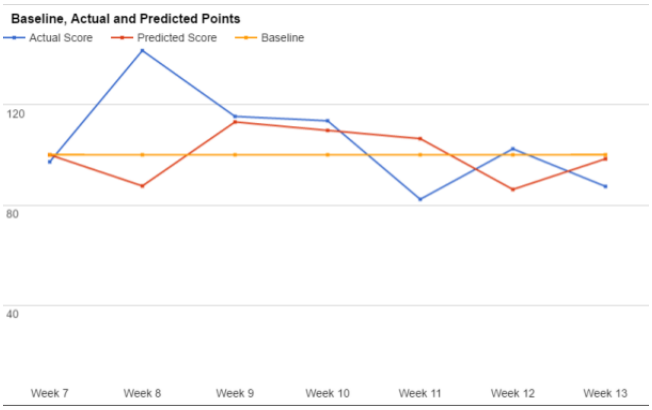
[6]https://www.fanduel.com/rules

6.3.1 Domain Pruning



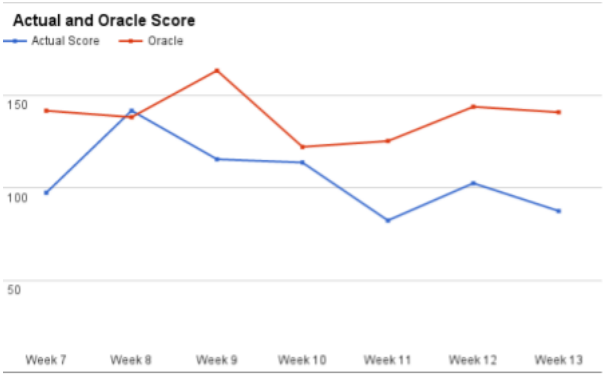
Graph showing the results from pruning the domains of each variable. Note that it took twice as long for the draft picker with an unpruned domain size (red) twice as large as the pruned domain size (blue).

6.3.2 Predicted vs. Actual Result



Graph showing the baseline, actual (blue) and predicted (red) scores after training on the first 6 weeks of 2015 season and testing on weeks 7 to 13. The baseline score of 100 (yellow) is the minimum needed to gain some monetary reward.

6.3.3 Actual vs. Oracle Result



Graph showing the relationship between an oracle result (red) where the team is chosen with no salary constraint.

7 Analysis

7.1 Domain Pruning

As we can see in the domain pruning set of experiments, the results from pruning the domain and significantly better than those from attempting to explore the whole tree. We noticed that, given the large state space, this CSP takes very long to run. These teams were the optimum teams, at

the time of the memory error that resulted. Even though only a small fraction of the state space was examined, the CSP produced some pretty good results.

7.2 Predicted vs. Actual Result

As we can see from the results between the Predicted and Actual result, our numbers are not too far off. Except for Week 8, the predicted value was representative of reality. The reason for Week 8 being an outlier was due to the out-of-ordinary performances put in by Charcandrick West and Odell Beckham Jr.

7.2.1 Charcandrick West's Performance

Very close to this game, the starting running back for the Kansas City Chiefs, Jamaal Charles, tore his anterior cruciate ligament (ACL), sidelining him for the rest of the season. The Chiefs, were one of the most run-heavy teams in the National Football League. A sudden and unexpected rise in the workload of the backup to Jamaal Charles, Charcandrick West, led to a dramatic rise in his fantasy production. This sort of increase is hard to predict using our models.

7.2.2 Odell Beckham, Jr.'s Performance

On the 8th week of the 2015 season, Odell Beckham, Jr. scored 35 fantasy points. While Odell Beckham, Jr. is among the elite NFL players, a 35 point fantasy performance is pretty rare for any player. Beckham caught 3 touchdown passes that day, including two from very short yardage. These short yardage touchdowns, even though they are not as impressive as long range catches, still count just the same in fantasy football.

These two players had unusually special days and they just happened to both be on our team. We can predict a general range of a fantasy player's score but it is hard to account for spectacular once-a-season performances such as these two.

7.3 Actual vs. Oracle Result

We decided to compare the results from our implementation with that of an Oracle implementation. The team chosen by the Oracle (a human expert) is not constrained by the salary cap and hence, should perform even better than the best teams. While this does not represent the trend of the data, our implementation was able to surpass the Oracle by a few points in Week 8 and came close in Week 10.

8 Conclusion

In conclusion, we have seen that through elementary techniques such as SGD, Constraint Satisfaction to perform at near-human levels in Fantasy Football. This involved translating domain-specific heuristics such as limiting the domain size to produce better results.

This sort of two-fold problem (determining the value of each player and choosing a combination of players under a salary cap) occurs in places beyond Fantasy Football such as Portfolio Optimization and some of the approaches used in this project could be applied in other areas as well.

9 References

- Abdulhamid, Adam, Tyler Fallon, and Adam Warmoth. "Fantasy Football Analysis." Web. Becker, Adrian, and Xu Andy Sun. "An Analytical Approach for Fantasy Football Draft and Lineup Management." De Gruyter. 2016. Web. 16 Dec. 2016.

  <https://www.degruyter.com/downloadpdf/j/jqas.2016.12.issue-1/jqas-2013-0009/jqas-2013-0009.xml>.

- BurntSushi. "BurntSushi/nflgame." GitHub. 13 Sept. 2016. Web. 16 Dec. 2016.

  <https://github.com/BurntSushi/nflgame>.

- FanDuel. "FanDuel Rules." FanDuel. Web. 16 Dec. 2016.

  <https://www.fanduel.com/rules>.

- "Fantasy Data." NFL Football Stats & League Leaders. Web. https://fantasydata.com/nfl-stats/nfl-fantasy-football-stats.aspx.
- Fry, Michael J., Andrew W. Lundberg, and Jeffrey W. Ohlmann. "A Player Selection Heuristic for a Sports League Draft." Journal of Quantitative Analysis in Sports. Emich.edu. 2007. Web. 16 Dec. 2016.
- "Weekly Football Points."Rotoguru.. Web. http://rotoguru1.com/cgi-bin/fyday.pl?game=fd.

10 Appendix

10.1 Drafted Teams and Results

Week	Actual Score	Prediction
7	97.2	99.985373
8	141.58	87.57996394
9	115.3	113.0864613
10	113.56	109.7276293
11	82.26	106.4357749
12	102.38	86.1881397
13	87.38	98.38243353

Table 2: Team Projected Scores and Actual Scores

Week 7

'QB0': 'Josh McCown', 'PK0': 'Justin Tucker', 'Def0': 'Jacksonville Defense', 'RB2': 'Chris Ivory', 'RB1': 'Frank Gore', 'WR2': 'Allen Robinson', 'WR3': 'Odell Beckham Jr.', 'WR1': 'Travis Benjamin', 'TE0': 'Greg Olsen'

Week 8

'QB0': 'Ben Roethlisberger', 'PK0': 'Justin Tucker', 'Def0': 'Cincinnati Defense', 'RB2': 'Justin Forsett', 'RB1': 'Charcandrick West', 'WR2': 'Odell Beckham Jr.', 'WR3': 'Keenan Allen', 'WR1': 'Tyler Lockett', 'TE0': 'Greg Olsen'

Week 9

'QB0': 'Philip Rivers', 'PK0': 'Andrew Franks', 'Def0': 'Tennessee Defense', 'RB2': 'Lamar Miller', 'RB1': 'Jay Ajayi', 'WR2': 'Stefon Diggs', 'WR3': 'Brandon Marshall', 'WR1': 'Julio Jones', 'TE0': 'Greg Olsen'

Week 10

'QB0': 'Jameis Winston', 'PK0': 'Justin Tucker', 'Def0': 'Cincinnati Defense', 'RB2': 'Darren McFadden', 'RB1': 'Doug Martin', 'WR2': 'Mike Evans', 'WR3': 'Antonio Brown', 'WR1': 'Stefon Diggs', 'TE0': 'Greg Olsen'

Week 11

'QB0': 'Derek Carr', 'PK0': 'Justin Tucker', 'Def0': 'Jacksonville Defense', 'RB2': 'James Starks', 'RB1': 'Jeremy Langford', 'WR2': 'Michael Crabtree', 'WR3': 'Mike Evans', 'WR1': 'Eric Decker', 'TE0': 'Rob Gronkowski'

Week 12

'QB0': 'Josh McCown', 'PK0': 'Phil Dawson', 'Def0': 'Jacksonville Defense', 'RB2': 'Thomas Rawls', 'RB1': 'T.J. Yeldon', 'WR2': 'Allen Robinson', 'WR3': 'Larry Fitzgerald', 'WR1': 'Mike Evans', 'TE0': 'Rob Gronkowski'

10.2 Features and Feature Extractor

We chose our features for each player (for the purposes of training) to be the statistical categories of rushing yards, rushing touchdowns, passing yards, passing touchdowns, interceptions, receiving yards, receiving touchdowns, receptions, home, away and the opponent being played that week. For the statistical categories, a three week average (or fewer if no more than 4 games had been played at the time of the prediction) was taken. The reasoning behind this was to determine if the player was playing well as of late. We didn't want a 35 point week 1 score followed by 4 to 5 point performances to be chosen over a player who scored 12 points consistently. More clearly, we did this to account for outliers. The home and away features represent whether the game was played

at a player's home field or at the opponent's home field. These features were added to account for home field advantage and tough road venues. The opponent feature represents what opponent the player's team was facing that week. This feature was added to account for the difficulty of the matchup. For example, a game against the Denver Broncos is seen as much tougher than a game against the Cleveland Browns. This feature should reflect that in the weight vector. The weight vector after training on 5 weeks (2-7) of the 2015 season is provided.

Feature	Trained Weight
Rushing Yards	0.07620522804957748
Rushing Touchdowns	0.00048762295618415876
Passing Yards	0.07172144073578218
Passing Touchdowns	0.0006001392915056799
Interception	0.00021623305467964376
Receiving Yards	0.1319862575547108
Receiving Touchdowns	0.0007969401413051872
Receptions	0.01259985096364802
Home	0.003566772691116201
Away	0.0033124924262109556
Tennessee Titans	0.0001648264713758584
New England Patriots	0.00019334015870976824
Seattle Seahawks	0.00020291105109381853
Indianapolis Colts	0.00029061388411977804
Arizona Cardinals	0.00014767099312409482
Pittsburgh Steelers	0.00018430147203998408
Minnesota Vikings	0.000147861074378851
Miami Dolphins	0.00022748367306629433
Cleveland Browns	0.00022849706221273703
Oakland Raiders	0.00027195301768490065
New York Jets	8.253890363094083e-05
Houston Texans	0.00021966448365878715
Washington Redskins	0.00021710213993918892
Jacksonville Jaguars	0.00028651952664815065
Tampa Bay Buccaneers	0.00016091571686077322
Philadelphia Eagles	0.00016432672802958381

Feature	Trained Weight
Dallas Cowboys	0.0002262533313573566
Kansas City Chiefs	0.0003128941872588689
San Francisco 49ers	0.00030627605739460477
San Diego Chargers	0.0001963816399700253
Buffalo Bills	0.00025280091280880296
New Orleans Saints	0.00027539070791547134
Carolina Panthers	0.0001628736953184543
Atlanta Falcons	0.000194415759635737
Detroit Lions	0.0002301437041553649
Cincinnati Bengals	0.00026019245604845927
New York Giants	0.0002471651431896192
Green Bay Packers	0.0002014034221747619
St. Louis Rams (Los Angeles)	0.00011617810829872708
Denver Broncos	0.00011122307874470214
Chicago Bears	0.0002316530062561659 5
Baltimore Ravens	0.00035237423031295157