Fantasy Football AI:
Predicting Performances Via Regression Learning?

Mohammed S. Khan, Robert J. Parsons, Ramon G. Moreno
California State Polytechnic University, Pomona
3801 W Temple Ave
Pomona, CA 91768
mohammedk@cpp.edu, rjparsons@cpp.edu, rgmoreno@cpp.edu

December 12, 2018

Abstract

Fantasy Football has been played by American football enthusiast throughout the globe for the past 30+ years. It has become more prominent as of late ever since the digital age came about. The creation of fantasy football applications on smartphones was the final push to bringing this concept game into the fingertips of many citizens. So naturally, people have tried many attempts to get the optimal fantasy lineup in order to get an upper hand on their opponents. Using some sort of artificial intelligence is the next logical step most tech-friendly players have landed on. Hence, we attempted to create an AI to predict fantasy points given different data surrounding a player, team, and game conditions. In this paper, we start with a basic introduction to fantasy football, what it is and how its played, along with how AI plays into it. Next, we will cover a few recent attempts people have had, doing the same thing we did. Following that, we go over how we collected, modeled, and trained our data set to arrive at a prediction. Finally, we will discuss the results (the good and the bad), reflect on what caused our results to do good or bad, and conclude our paper by discussing future approaches and designs to further improve accuracy and accessibility.

Keywords

Machine Learning, Artificial Intelligence, Fantasy Football, Model, Over / Under, Vegas Line

Introduction

When we were searching for a problem domain for an Artificial Intelligence project we came up with many problems in which AI can pontentially help. Fortunately for us, it was Football season, something near and dear to us; furthermore, our minds are already occupied with Football, so naturally we chose Football as our problem domain. We furthered narrow our problem domain to Fantasy Football and if it was possible to accurately predict a players fantasy points. We saw that fantasy footballs statistics, conditions, team formations, and many other data sets can lend itself to a creation of an Artificial Intelligence that accurately creates predictions.

Before researching into which artificial intelligence approach will be implemented, we needed to make sure that data will be available for us to use. The data is a vital part of our Fantasy Football AI so we were critical in which data we chose. Needless to say, some data sets were unavailable to us, as it required a subscription, so we opt into data mining from websites. Data mining was an ongoing process throughout our work, as we needed to make sure we were getting what we needed and that data we are receiving can help with the predictions. For example, we excluded positions as it variables of type string was not liked by the

linear regression model. Once we were confident that collecting data was possible we moved to the actual implementation of our Artificial Intelligence approach.

Any approach that was uninformed or involved a Monte Carlo method did not seem feasible as we saw that previous data is crucial to accurately create a prediction for a given player. At first we thought of using a logistic regression approach, but with the advice of Dr. Adam Summerville, a logistic approach would not have gone well with our data set. Instead, we went with a Linear Regression Approach and with the help of PyTorch we able to implement a linear regression model: however, as we saw the results we saw that the accuracy of the predictions was not to our standards.

```
def calculateFantasyPoints(pyds,ptd,pint,ryds,rtd,fmb,recyds,rectd):

    temp_list_of_stats = [0,0,0,0,0,0,0,0]
    temp_list = [pyds,ptd,pint,ryds,rtd,fmb,recyds,rectd]
    for index in range(len(temp_list)):
        temp_list_of_stats[index] = float(temp_list[index])
    total = 0
    total += (float) (temp_list_of_stats[0]/25)
    total += (float)(temp_list_of_stats[1]*4)
    total -= (float)(temp_list_of_stats[2]*2)
    total += (float)(temp_list_of_stats[3]/10)
    total += (float)(temp_list_of_stats[4]*6)
    total -= (float)(temp_list_of_stats[5]*2)
    total += (float)(temp_list_of_stats[6]/10)
    total += (float)(temp_list_of_stats[7]*6)
    return total
```

Figure: Calculate Fantasy Points function from our Code

Data Mining

Collecting data is a vital part of any machine learning / artificial intelligence endeavour. For Fantasy Football it is no different. Early on in our search, finding data was a priority since that is the first thing you need before moving onto anything else. There were a few prime candidates to pull data from, that had great data sets and options, however, they were paid services that were just not feasible at that point and time. So we limited our search to open databases and free websites to cut our costs in this section. Looking a little harder, one website stood out to us in particular. This was because the data was, for the most part, nicely organized in tables and sections. The name of this site is (https://www.pro-football-reference.com/).

Now we needed an API to help us pull data from public websites. We choose BeautifulSoup4 because that was the most obvious choice to go with python. BeautifulSoup4 gave us the ability to pull data from a webpage, whole or just parts of it. However, we now needed an API to pull data precisely from tables on a webpage. This was quite tricky since some of the tables provided had multiple headings which would mess up the algorithm we formed to pull data from single header tables. Eventually we formed three different functions to pull data from different tables, the input would be the name of the table and the url of the webpage. Providing the webpage url was easy as you will see later on in this paper. However, finding the name of the table was a tad harder. A way to hard-code it would be to inspect element, on your browser, and check for the table = name string. That proved to be too tedious for the time being, so we created another function that takes a URL as an input and outputs a list of table names. With the junction of both those functions we were able to pull numerous tables with minimal to no problem.

Moving on, one of the most obvious data sets to look at while predicting a players fantasy points would be their average points. Since we pulled data before creating our model, we were not quite sure what data would be needed and what would be useless, so we pulled all the data that was given. This equated to nearly

18 different data points per each player. The way we decided to collect the different player data information was to create a class named Player. This class would have many different attributes that would be pulled from this table directly or that would be calculated through data scraped from the table.

Now, we needed a way to access the different players and their averages. We landed on the decision to create a dictionary with a key that would hold the string of the player name. One design issue we thought would come up with this choice of key, was that what if two players had the same exact first and last name? After further research, amongst active players, there are people who share first names, last names, and even middle names. However, no two players as of 2018 / 2019 season have the same first and last name in spelling. So we were able to get away with this design for the scope of our program. If there was ever an issue, one way to fix it would be to concatenate the string key with the teams name. Next the value of the dictionary would be simple, just the associated Player class object. These dictionary values would be created dynamically while we were scraping the data.

Passing *ProBowl, +1st-tm All-Pro Share & more ▾ Glossary

No.	Player	Age	Pos	G	GS	QBrec	Cmp	Att	Cmp%	Yds	TD	TD%	Int	Int%	Lng	Y/A	AY/A	Y/C	Y/G	Rate	QBR	Sk	Yds	NY/A	ANY/A	Sk%	4QC	GWD
16	Jared Goff	24	QB	12	12	11-1-0	275	414	66.4	3754	27	6.5	7	1.7	70	9.1	9.6	13.7	312.8	109.9	74.1	26	160	8.17	8.68	5.9	4	4
6	Johnny Hekker	28	P	12	0		1	2	50.0	12	0	0.0	0	0.0	12	6.0	6.0	12.0	1.0	68.7		0	0	6.00	6.00	0.0		
	Team Total	**26.0**		**12**		**11-1-0**	**276**	**416**	**66.3**	**3766**	**27**	**6.5**	**7**	**1.7**	**70**	**9.1**	**9.6**	**13.6**	**313.8**	**109.7**		**26**	**160**	**8.16**	**8.67**	**5.9**	**4**	**4**
	Opp Total			**12**			**266**	**408**	**65.2**	**3262**	**26**	**6.4**	**11**	**2.7**		**8.0**	**8.06**	**12.3**	**271.8**	**99.7**		**33**	**275**	**6.8**	**6.8**	**7.5**		

Rushing & Receiving *ProBowl, +1st-tm All-Pro Share & more ▾ Glossary

				Games		Rushing							Receiving								Total Yds				
No.	Player	Age	Pos	G	GS	Att	Yds	TD	Lng	Y/A	Y/G	A/G	Tgt	Rec	Yds	Y/R	TD	Lng	R/G	Y/G	Ctch%	Touch	Y/Tch	YScm	RRTD	Fmb
30	Todd Gurley	24	RB	12	12	233	1175	15	36	5.0	97.9	19.4	61	46	474	10.3	4	56	3.8	39.5	75.4%	279	5.9	1649	19	1
34	Malcolm Brown	25		12	0	43	212	0	19	4.9	17.7	3.6	7	5	52	10.4	1	18	0.4	4.3	71.4%	48	5.5	264	1	0
16	Jared Goff	24	QB	12	12	37	86	1	16	2.3	7.2	3.1										37	2.3	86	1	7
17	Robert Woods	26	WR	12	12	15	122	0	56	8.1	10.2	1.3	99	64	971	15.2	5	36	5.3	80.9	64.6%	79	13.8	1093	5	0
12	Brandin Cooks	25	WR	12	12	7	41	1	10	5.9	3.4	0.6	90	63	1026	16.3	3	57	5.3	85.5	70.0%	70	15.2	1067	4	1
18	Cooper Kupp	25	WR	8	8	4	25	0	12	6.3	3.1	0.5	56	40	566	14.2	6	70	5.0	70.8	71.4%	44	13.4	591	6	0
14	Sean Mannion	26		1	0	3	1	0	3	0.3	1.0	3.0										3	0.3	1	0	0
83	Josh Reynolds	23	wr	12	4	2	8	0	10	4.0	0.7	0.2	25	15	197	13.1	3	27	1.3	16.4	60.0%	17	12.1	205	3	0
6	Johnny Hekker	28	P	12	0	1	3	0	3	3.0	0.3	0.1										1	3.0	3	0	0
81	Gerald Everett	24		12	0	0	0	0	0		0.0	0.0	30	19	217	11.4	3	40	1.6	18.1	63.3%	19	11.4	217	3	0
89	Tyler Higbee	25	TE	12	12	0	0	0	0		0.0	0.0	28	19	217	11.4	2	33	1.6	18.1	67.9%	19	11.4	217	2	1
11	KhaDarel Hodge	23		10	0	0	0	0	0		0.0	0.0	2	2	17	8.5	0	14	0.2	1.7	100.0%	2	8.5	17	0	0
15	Nick Williams	28		2	0	0	0	0	0		0.0	0.0	3	2	17	8.5	0	11	1.0	8.5	66.7%	2	8.5	17	0	0
37	Sam Shields	31	cb/rcb	12	2	0	0	0	0		0.0	0.0	1	1	12	12.0	0	12	0.1	1.0	100.0%	1	12.0	12	0	0
	Team Total	**26.0**		**12**		**345**	**1673**	**17**	**56**	**4.8**	**139.4**	**28.8**	**403**	**276**	**3766**	**13.6**	**27**	**70**	**23.0**	**313.8**		**621**	**8.8**	**5439**	**44**	**10**
	Opp Total			**12**		**281**	**1421**	**8**		**5.1**	**118.4**	**23.4**		**266**	**2987**	**11.2**	**26**		**22.2**	**248.9**				**4408**	**34**	**13**

Figure: Averages at (https://www.pro-football-reference.com/teams/ram/2018.htm)

The figure above contains the player averages for the players on the Los Angeles Rams. Each team has a separate webpage with similar tables. So now, we just needed to find a way to loop through the different webpages. Luckily, the reason we choose this website was because it was structured really well for data scrappers. A sample webpage for the above table would be something like this (https://www.pro-football-reference.com/teams/ram/2018.htm). We can loop through the web pages by just finding the abbreviations for the 32 teams in the league, and concatenating the url string accordingly. Another nice feature you can see in this link is the 2018 part. If for some reason you would like to scrape old data for a bigger data set, you would be able to just change the 2018 to whatever season you want to.

The next set of data we would need is game info. Game info refers to stuff like weather, vegas line, over/under, and other auxiliary data points specific to a certain game. The reason we wanted to pull this data in the first place was because a common connection people make in the NFL is that the colder the weather is the less points they are likely to score. This seems quite logical because, some games are played in the snow where the ball cant travel further than 10 yards and the players cant run without trying to break their slip.

Game Info	
Won Toss	Jaguars (deferred)
Roof	outdoors
Surface	grass
Weather	44 degrees, wind 5 mph
Vegas Line	Tennessee Titans -5.5
Over/Under	37.5 **(over)**

Figure: Game Info at (https://www.pro-football-reference.com/boxscores/201812060oti.htm)

The figure above, Game Info, contains the different data sets we would need for our game info section. So we pulled the data from this table and stored it in temporary variables for that certain game we are trying to predict.

Passing, Rushing, & Receiving

Share & more ▾ Glossary

		Passing									Rushing				Receiving					Fumbles	
Player	Tm	Cmp	Att	Yds	TD	Int	Sk	Yds	Lng	Rate	Att	Yds	TD	Lng	Tgt	Rec	Yds	TD	Lng	Fmb	FL
Cody Kessler	JAX	25	43	240	1	0	4	45	35	81.5	5	17	0	10	0	0	0	0	0	1	1
Leonard Fournette	JAX	0	0	0	0	0	0	0	0		14	36	0	7	3	2	5	0	9	0	0
T.J. Yeldon	JAX	0	0	0	0	0	0	0	0		1	7	0	7	1	1	10	0	10	0	0
Dede Westbrook	JAX	0	0	0	0	0	0	0	0		0	0	0	0	10	7	88	1	23	0	0
Keelan Cole	JAX	0	0	0	0	0	0	0	0		0	0	0	0	7	3	55	0	35	0	0
Donte Moncrief	JAX	0	0	0	0	0	0	0	0		0	0	0	0	10	5	47	0	12	0	0
Blake Bell	JAX	0	0	0	0	0	0	0	0		0	0	0	0	2	2	16	0	9	0	0
James O'Shaughnessy	JAX	0	0	0	0	0	0	0	0		0	0	0	0	4	3	12	0	7	0	0
Rashad Greene	JAX	0	0	0	0	0	0	0	0		0	0	0	0	2	1	10	0	10	0	0
Tommy Bohanon	JAX	0	0	0	0	0	0	0	0		0	0	0	0	1	0	0	0	0	0	0
Carlos Hyde	JAX	0	0	0	0	0	0	0	0		0	0	0	0	1	1	-3	0	-3	0	0
		Passing									Rushing				Receiving					Fumbles	
Player	Tm	Cmp	Att	Yds	TD	Int	Sk	Yds	Lng	Rate	Att	Yds	TD	Lng	Tgt	Rec	Yds	TD	Lng	Fmb	FL
Marcus Mariota	TEN	18	24	162	0	1	1	0	29	75.3	4	13	0	12	0	0	0	0	0	0	0
Derrick Henry	TEN	0	0	0	0	0	0	0	0		17	238	4	99	0	0	0	0	0	0	0
Dion Lewis	TEN	0	0	0	0	0	0	0	0		10	13	0	5	5	5	39	0	29	0	0
Taywan Taylor	TEN	0	0	0	0	0	0	0	0		0	0	0	0	7	6	59	0	13	0	0
Anthony Firkser	TEN	0	0	0	0	0	0	0	0		0	0	0	0	3	3	27	0	16	0	0
Corey Davis	TEN	0	0	0	0	0	0	0	0		1	0	0	0	3	2	21	0	13	0	0
MyCole Pruitt	TEN	0	0	0	0	0	0	0	0		0	0	0	0	1	1	9	0	9	0	0
Luke Stocker	TEN	0	0	0	0	0	0	0	0		0	0	0	0	1	1	7	0	7	0	0
Tajae Sharpe	TEN	0	0	0	0	0	0	0	0		0	0	0	0	4	0	0	0	0	0	0

Figure: Game Data at (https://www.pro-football-reference.com/boxscores/201812060oti.htm)

The next step was to get what we called Game Data. This was basically a box score that would display the different yardage, touchdowns, and attempts a player had that game. Luckily, the same webpage we found the game info at, below it it held a table for boxscores. Figure: Game Data at (https://www.pro-football-reference.com/boxscores/ 201812060oti.htm)

As you can see in the table, Game Data, above it displays different sections of data for each player that played in that game. Using the first column in this table, we are given the key to our dictionary so we could

pull the players averages and stats from the player averages table we pulled. Using the second column, we are able to access the Players team and the opponents team. Finally, columns 3 till the end we pulled the data and threw it into our function that would calculate fantasy points based on different stats (Figure: Calculate Fantasy Points). After that, we would end up with a floating number that would represent the number of fantasy points gained by player X in game Y with conditions A,B,C. In other words, this was our output that we were trying to achieve, the final column in our csv data sets.

So we just needed a way to loop through the different weeks and games to fill our data set in csv files. Looking at each weeks URL it is easy to spot how to accomplish this. The URLs look like this https://www.pro-football-reference.com/years/2018/week_14.htm. Very clearly, if we loop through week 1 to week x and concatenate x in our week_x.htm we would be able to easily accomplish this. This Webpage would display a list of games played that week and a link to their data. The game data was stored in a webpage like this https://www.pro-football-reference.com/boxscores/201812020atl.htm. The only changing part of this url is the few numbers and letters after 2018 however, it was quite simple to find the URL by looping through the tables. Finally, we just had to create a csv file by looping through week 1 to week current and an inner loop that would loop through the games played that week and we were able to create our data sets needed for the next step.

Training

To begin training, we organized our scavenged data (player_team_offense_ranking, player_fantasy_avg, weather, opp_team_defense_ranking, and f_points_earned) into two matrices, X and Y. The X matrix consisted of every data point except for fantasy points for each team, while Y consisted of the fantasy points for each team. The data was organized this way because our main objective with this project was to predict the potential fantasy points a team could earn based off the aforementioned statistics for that team. We then transformed both of the matrices into Pytorch tensors, so that the data could be used in our training function. For training function, the optimizer is Stochastic Gradient Descent function that takes a learning rate of 0.01 and our loss function is Mean Squared Error, which is a fairly common and basic setup for a training function for this sort of project.

We then decided to write a basic Pytorch Sequential model that consisted of one Linear layer that took 4 as its input space and 1 as its output space, since we are only trying to predict a single value. After training over 10,000 epochs, the loss settled at 33.57. Our second model was similar to the first model, but now with 100 hidden layers. Training over 10,000 epochs resulted in a loss of 32.07, which is not much better than the results of the first model. The third model we wrote was similar to the second model, but with an ReLU layer, which resulted in a loss of 29.40 after 10,000 epochs. While the loss was getting better, it was still not close to where we needed it to be to make accurate fantasy point predictions.

So we decided to write our own custom linear regression model that consisted of 4 linear input layers, 100 hidden layers, an ReLU layer, and 1 linear output layer, similar to the previous Sequential model. However, in the forward pass, we decided to use the clamp function with a min parameter of 0 (which is similar to an ReLU activation function) in hopes of accounting for any nonlinearities as much as possible in our data. Additionally, we updated the training functions optimizer to an Adam optimizer, as this has become a popular optimizer in neural network training in recent years (Source 1). However, with these changes, the loss was only marginally improved over the previous model, with it leveling off at 29.10 after 10,000 epochs of training.

Results

To see how well the custom model performed, we loaded team data (including fantasy points) from a CSV file into an array and then passed that to a Pytorch tensor. This newly created tensor object was then passed to the custom models forward function so that a fantasy point prediction could be outputted for that team. This was done for each team and an average percent difference between the actual fantasy point value versus

the predicted fantasy point value was around 81%. Obviously, these are not the results that we had hoped for, as a prediction using this model would most likely be very inaccurate for a person partaking in fantasy football and trying to predict team fantasy points.

In order to improve the results of this project, we had a few ideas in mind but did not have time to implement. Some of these ideas include training the custom model on a GPU with CUDA acceleration to see if the loss could be improved, trying different optimizers or loss functions, creating a new model that is optimized to reduce noise, have the model train on additional data from previous seasons, and create an entirely new model that predicts something else, such as betting line or wins (which could use logistic regression rather than linear regression). Though we may not have had time to implement these ideas into our project before the deadline, we are still very interested in testing these ideas and are committed in pursuing the implementation of them going forward.

Reflections

Looking back at our results, it was definitely not the most accurate. However, we can reason why that may have been. First off, we made an assumption that aspects of the game like weather, opponent defense rank, and other elements would affect a player to play better or worse. However, after further investigation, in our relatively small data set it was hard to draw that conclusion. One reason this could be was because the data we collected was unbiased and whole. Normally in fantasy football, you would look at only the highest performers but we took even the benchwarmers to gain a larger data set. Another reason, would be that our data set is relatively small as mentioned earlier. We only pulled data from this season which turned out to be around 4k points of data. If we pulled from more seasons we could have easily surpassed 50-100k data points which would give us a bigger field. Lastly, we needed to create a better fitted model to package the data in. Right now we have two separate models that are testable however, we feel like there are different models that might specify aspects like rushing, passing, and catching yards / touchdowns. If we bring more position specific data from opposing defenses it could prove to help us gain better results.

The positives we can take back from this are plentiful. You never really make mistakes without learning as the saying goes. We learned a lot about Python, Machine Learning, and big data in our attempt. These are skills we are proud to own after creating this project. In the future if we ever have to create any AI for some application, the starting learning curve will definitely be more smooth than it was for us this time around. Additionally, we also feel we learned a lot about neural networks and how they work. From activation functions to printing results, we feel that we have a much deeper understanding of this subset of machine learning than we had previously.

Conclusion

Although our results, and a loss function at around 29.1, from our Fantasy Football AI was not as accurate as we liked it to be, the experience proved helpful to better understand Artificial Intelligence. We were required to look into different approaches that could prove useful for our problem domain. This in itself proved useful into learning into Artificial Intelligence and how its a growing field. Furthermore, the we learned the difficulties of not only acquiring data for our linear regression model, but manipulating it in a meaningful way.

Along with the creation of our Fantasy Football Artificial Intelligence we also learned much about Python as we had little to no experience in it. We learned how to use PyTorch and matplotlib, along with how to write general code in Python.

It is needless to say that there are better Fantasy Football predictions out there, but with our amount of knowledge and resources going into this project we found ourselves satisfied with the results.

Sources

https://arxiv.org/abs/1705.08292