4/22/2021

NBA Award Prediction - Final Report

Tomer Zur

The code for my project can be found on Github, at this link: https://github.com/tomerzur/NBA-Award-Prediction. I also have a website with instructions and my code that shows how to replicate and run the models that I created. The link to this website is https://tomerzur.github.io/NBA-Award-Prediction/.

Introduction

For my project, I am predicting each of the individual NBA award winners for a given NBA season. These awards are voted on each year by a panel of NBA media members, and they include Most Valuable Player (MVP), Defensive Player of the Year (DPOY), Sixth Man of the Year (6MOY), Most Improved Player (MIP), and Rookie of the Year (ROY). I think that this project is interesting because it adds a new perspective to the discussion about who should win these awards. Currently, that discussion includes the opinions of the media, NBA executives, coaches, players, and fans; but by using the machine learning model that I will create, people could also take into account the model's predictions when considering who they think will win each award.

Methods

What I'm Predicting

I will predict how many points each player will receive for each award, and the player with the most points will receive the award. This would model the way that the awards are given out in real life, as voters submit their top three, and a player's total score is calculated as a weighted sum of the first, second, and third place votes they received.

The inputs (aka the features) to the models that I will create to predict each award will be the player stats each season. Here is an example of one row of input:

player	season	age	team	position	g	…	pf	pts	Max Award Pts.
LeBron James	2013	28	MIA	PF	76	…	110	2036	1210

*many statistics are omitted from this table for the purpose of readability

Once it is given these inputs, the model will then predict the weighted sum of voting points that each player will receive in a given season for a given award. The player with the highest vote point total will then be the model's predicted award winner.

In the example above, the correct output for the MVP prediction model would be 1207 voting points, as that is the number of points LeBron James received for MVP voting in 2013. The model would also hopefully not predict that anyone else got more points than LeBron did that year, as he was the MVP in that season.

Feature Engineering

One thing I realized after exploring my data is that I could extract some features that would be very helpful for my models. For example, I added net change features for each statistic to my most improved player award data, as it would be important for my models to know how much a player improved at each metric in a given season. The other features I added to my data were season number (e.g. 4 if it is the player's fourth season in the NBA) and % of games started in a season. I added these features to my datasets for each award.

Linear Regression

The first method that I am using to predict the NBA award winners is a linear regression. This regression will take in all of my features and then use them to make predictions.

For this linear regression, there are missing data values for players from older seasons. This happens because there are some statistics that weren't kept track of back in the day, such as blocks and steals. To deal with this, I am also using linear regression models to predict these missing values. Then, I am passing in my updated dataset with filled-in missing values into my main linear regression model (which is predicting the award points each player will get).

Neural Network

The other method that I am using to predict the NBA award winners is a neural network. The neural network also takes in all of my features and uses them to train the model. However, as it is a neural network, there is a non-linearity applied to the data in order to make the final predictions.

To prepare my data for the neural model, I scale it in two different ways. I scale them in order to reduce the skew in my award points values, and so that my model can make predictions from 0 to 1. First, I scale each award points value logarithmically. This reduces the skew between the award winners/high point receivers and the players who receive no award points. Then, I apply a min max scaler to each feature and to the award points so that their ranges of values are all from 0 to 1.

The type of neural model that I am using is a multi-layer perceptron. I have two hidden layers in the perceptron, and they each have 40 nodes. I tested a bunch of different hyperparameters, and these are the ones that I ended up using:

- Activation function (hidden layers): relu
- Activation function (output layer): sigmoid
- Regularization (input layer): L2 regularization (l=0.6)
- Optimizer: SGD
- Learning Rate: 0.01
- Clipnorm (clipping gradients between 0 to 1): Yes
- Epochs: 50
- Batch Size: 300

Train/Test Split

Before creating any models, I had to figure out how to split up my data. What I decided to do was to split my data into three groups: train, validation (development), and test. I made this split by season, as players from the same season had to end up in the same group. My train dataset is players from seasons earlier than 2011, my validation dataset is players from the 2011 season to the 2015 season, and my test dataset is players from the 2016 season to the 2020 season.

As I work on my models, I am using the validation dataset to judge how well the models are performing. Once I finish developing and tweaking the models, I will see what my final model's accuracy is on my test data.

Results

Here is how each of my models have performed at predicting each award.

For both the linear regression and the neural network, I dealt with missing (NaN) data in three different ways. The first way (Method 1) was by filling all of my NA values with zeros. I attempted this way first because it was very easy to implement and would be a good baseline for my linear regression.

The second way I dealt with missing data (Method 2) was by using a separate linear regression to predict what each of these values was. For example, I created a linear regression that predicted how many blocks each player got in each season before 1974 (the year that the NBA started keeping track of blocks). I did this to fill in missing values for three pointers made and attempted, games started, offensive and defensive rebounds, steals, blocks, and turnovers.

The third way I dealt with these values (Method 3) was by filling them in with the mean value for that feature. For example, if a steals value was NA, I would fill it in with the average of all of the steal values that aren't NA.

The results of both the linear regression and the neural network when dealing with NA values in each of these three ways are shown in the tables below:

Linear Regression

Results on train dataset:

Award	Method of dealing w/ missing values	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	1	2531	43.64%	24	0.625
	2	2541	43.64%	24	0.619
	3	2534	41.82%	23	0.614
DPOY	1	205	28.57%	8	0.416
	2	205	28.57%	8	0.416
	3	205	28.57%	8	0.417
ROY	1	1051	65.96%	31	0.651
	2	1030	65.96%	31	0.651
	3	1030	63.83%	30	0.634
MIP	1	209	28.00%	7	0.416
	2	209	28.00%	7	0.416
	3	209	28.00%	7	0.416
SMOY	1	372	51.85%	14	0.537
	2	372	51.85%	14	0.537
	3	372	51.85%	14	0.535

Results on validation dataset:

Award	Method of dealing w/ missing values	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	1	4042	60%	3	0.650
	2	4001	40%	2	0.629
	3	4011	60%	3	0.645
DPOY	1	523	20%	1	0.345
	2	523	20%	1	0.345
	3	523	20%	1	0.344
ROY	1	4518	40%	2	0.591
	2	4477	60%	3	0.645
	3	4464	60%	3	0.645
MIP	1	534	0%	0	0.271
	2	534	0%	0	0.271
	3	534	0%	0	0.269
SMOY	1	1071	60%	3	0.562
	2	1071	60%	3	0.562
	3	1071	60%	3	0.562

I ended up using method 2 to deal with missing values (predicting my missing values using linear regression), so for my test dataset, my results only are calculated with this method. Here are the results for my test dataset:

Award	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	2892	20%	1	0.621
DPOY	577	20%	1	0.367
ROY	2818	20%	1	0.617
MIP	626	0%	0	0.297
SMOY	732	40%	2	0.457

Neural Network

Results on train dataset:

Award	Method of dealing w/ missing values	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	1	3455	38.18%	21	0.566
	2	3455	36.36%	20	0.562
	3	3455	36.36%	20	0.561
DPOY	1	227	14.29%	4	0.266
	2	227	14.29%	4	0.258
	3	227	14.29%	4	0.257
ROY	1	1366	57.45%	27	0.726
	2	1366	55.32%	26	0.702
	3	1366	57.45%	27	0.715
MIP	1	222	0%	0	0.179
	2	222	0%	0	0.155
	3	222	0%	0	0.174
SMOY	1	437	40.74%	11	0.478
	2	437	37.04%	10	0.476
	3	437	33.33%	9	0.469

Results on validation dataset:

Award	Method of dealing w/ missing values	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	1	5249	20%	1	0.492
	2	5249	20%	1	0.536
	3	5249	20%	1	0.515
DPOY	1	582	20%	1	0.233
	2	582	20%	1	0.230
	3	582	20%	1	0.223
ROY	1	6098	80%	4	0.688
	2	6099	80%	4	0.695
	3	6099	80%	4	0.698
MIP	1	568	0%	0	0.233
	2	568	0%	0	0.208
	3	568	0%	0	0.229
SMOY	1	1241	60%	3	0.530
	2	1241	60%	3	0.529
	3	1241	60%	3	0.526

I ended up using method 2 to deal with missing values (predicting my missing values using linear regression), so for my test dataset, my results only are calculated with this method. Here are the results for my test dataset:

Award	Mean Squared Error	% of correct winners predicted	# of correct winners predicted	Average Rank-Biased Overlap (RBO)
MVP	3766	20%	1	0.508
DPOY	617	0%	0	0.246
ROY	3685	60%	3	0.726
MIP	660	0%	0	0.175
SMOY	824	40%	2	0.470

Rank Biased Overlap (RBO)

After working with my two linear regression models, I realized that I needed a better way to measure the accuracy of the rankings that my model came up with. I looked at using some traditional rank correlation metrics (such as Spearman's rank correlation coefficient or the Kendall rank correlation coefficient), but there were a few problems with them. For example, it is possible for there to be items on one of my lists that is not on my other list at all. In addition, I had to find a metric that also weighted 1st place rankings as being more important than 5th place rankings.

The metric I found that met these criteria for measuring my rankings is Rank Biased Overlap (RBO). The metric is described in detail in this paper, but it solves both of these problems I had. To calculate overlap, you incrementally look at your two lists as sets of the top 1 items, then as sets of the top 2 items, and so on until you look at the lists as sets of all of the items in the list. Then, you take the average of the percentage of overlap (number of items seen in both sets / size of set) seen in each of these sets. Rank biased overlap is built on this idea of calculating overlap.

The rank-biased overlap I am reporting is the average of the rank-biased overlap numbers that I get for each year that the model is predicting.

Conclusions

Now that the 2021 NBA regular season is less than a month from finishing, I would love to use my model to predict who will win the MVP this season. Unfortunately, when I tried to re-scrape my data to update my 2021 season data, I found out that basketball-reference.com recently updated their website to limit scraping. After I scraped a few hundred players, my requests to the website timed out, and after digging through the website, I found out that basketball-reference can block your IP address in response to excessive scraping activity. Because of this, I will not be predicting the 2021 MVP winner, as I do not have enough data from this current season.

However, for the 2020 season, I have a full dataset. The actual MVP ended up being Giannis Antetokounmpo, and unfortunately, my models didn't end up picking him to win the award. My neural model predicted James Harden as the MVP (he was actually the third-leading vote getter), while my linear regression model predicted Luka Doncic as the MVP (he was actually the fourth-leading vote getter).

In the end, my models performance was a little underwhelming. Although they did a great job filtering through the players and identifying solid candidates for each award, the ranking of these candidates did not end up being as accurate as I had hoped for.

However, I have a lot of ideas of possible improvements that I could make to my model in the future.

One improvement is adding a lot more statistical filters to the data in the preprocessing stage. Two of the awards that had the best accuracy rate were Rookie of the Year and Sixth Man of the Year, as those awards had much less candidates to choose from due to the filtering of non-rookies and starters from those awards' datasets. In order to add more filters, I would have to come up with statistical requirements that are extremely unlikely for an award winner to not meet. For example, some additional filters I could add would be a minimum minutes played requirement and a minimum games played requirement.

Another possible improvement would be adding team performance to each player's season statistics. Although this would require scraping more data from basketball-reference.com, it could make a big difference. For some awards – especially MVP – how well a team performs is a very important criteria for the award voters. When you take a closer look at the predicted winners versus actual winners for that award, you can see that a lot of the incorrect predictions were made by predicting a player with incredible individual stats whose teams didn't perform

that well. For example, my models love predicting Russell Westbrook as the MVP winner, as he has been averaging a near triple-double each season since 2017 while his teams have not been among the league's best teams. Although he won the MVP award in 2017, that was an exception to the rule of thumb that the MVP is the best player on one of the league's best teams.

One last thing that could improve the model's performance would be getting rid of some unnecessary features. For this project, I used almost all of the data that I scraped, even though some features are similar to each other and others don't intuitively affect whether a player would win an award. For example, a player should have an equal chance of winning an award no matter what team they happen to play for, so I could try removing that feature when making my predictions. I also use field goal %, two point field goal %, three point %, free throw %, and effective field goal % to make my predictions. However, I could retain all of the shooting percentage information even if I didn't pass in a player's field goal % and effective field goal %, as those two percentages are just functions of the other three percentages.

Ultimately, this project was a great learning experience for me. I scraped the dataset myself, made decisions about how to deal with the data I retrieved, and ended up fitting two different models to the data. Even though the predictions should currently be taken with a grain of salt, the work done to create this model is still useful, as I can be improve on it in the future and people can still use my models and my data to gain insight on which players are playing well in each season.