Predicting NBA MVP Using Machine Learning

Louis Rabinowitz, Clark Davis, Qiaofeng (Marco) Liu, Nishit Bade, Thomas Boyd

May 2019

1 Introduction

The most prestigious award that can be given to any individual NBA player each season is the Most Valuable Player award, better known as the MVP award. The award is given, as its name suggests, to the player who is perceived as most valuable. We would like to predict the winner of this award using various NBA statistics. In surveying other related works, we found our model may fall short as it only knows statistics and can't account for other factors such as popularity or narrative. Given this, our prediction will be evaluating who has the most MVP like statistics. We used a mix of box score statistics (points, rebounds, assists, steals, etc.) and combined statistics (combinations of box score statistics to increase number of features).Data for most of these statistics was available for each player starting with the 1950 NBA season on www.basketball-reference.com. Some statistics were not available over the whole time range (ex. Blocks were not recorded until the 1973-1974 NBA season.) Obviously, no linear combination of these statistics would be able to predict the winner of the MVP award with absolute certainty. Therefore, we trained various models to predict the outcome of the MVP award including logistic regression, support vector machine, and neural network. We compared the results of our multiple models by looking not only at how many MVPs they correctly predict, but also which models reduce the number of false positives. Our approach is useful for fields outside of just sports awards predictions. It can be used for any binary classification problem where the majority class is much more numerous than the minority class (for example, medical diagnosis). Additionally, the method can be expanded to problems with more than two classes.

2 Approach

2.1 Data Collection and Normalization

All of our data was gathered from basketball-reference.com in the form of a CSV file. The data was initially a mix of counting statistics, occurrence rate statistics, and percentages. A single point represents one player's statistics for a single year. To convert all of these into a uniform usable format, we first divided the counting stats by season length, then removed all data points from before the year 1980; the three point line was added during the 1979-1980 NBA season. Additionally, players with less than 5.0 minutes played per game were removed from the data set. After data removal we had about 12,000 remaining data points. Finally, we transformed the data from $[-0.5, 0.5]$ according to the following function:

$$x^*_{i,j} = \frac{x_{i,j} - \min(x_{:,j})}{\max(x_{:,j}) - \max(x_{:,j})} - 0.5, \tag{1}$$

where $x^*_{i,j}$ refers to a specific transformed feature of a data point, and $x_{:,j}$ refers to a whole feature.

2.2 Class Size Discrepancy

Nearly 500 players played in the NBA last season, but only one was selected as the most valuable player. This creates a sampling issue because the sizes of our two classes are vastly different and a model would get very good results from just predicting there are zero MVPs.

We fix this problem by introducing synthetic samples of MVP data. To generate the synthetic data we first calculate the five nearest neighbors to each existing MVP and create pairs. Next, we randomly selected N of these pairs and select a random point along the line connecting the two points. These N points are then added to our set of possible training data. The merits of this approach is discussed in Chawla, 2002.

3 Results and Implementation

We need to be able to compare the results of our different methods. Unfortunately, we can't simply look purely at what percentage of predictions a model gets correctly. With only 44 MVPs and 11,801 total data entries, we would get 99.63% of predictions correct if the model were to guess that there are zero MVPs. So, we will primarily look at three metrics in order

to judge the "best" model. The first two are easy to calculate: percentage of MVPs correctly guessed and the number of false positives. The third we have named "true score." "True score" is calculated by breaking the data up into years. Then the player within each year who had the prediction closest to 1.0 will be selected as the models true prediction. Finally, the number of true predictions that are correct is divided by the number of years of data. This gives us the percentage of true MVPs who were determined to be the most likely recipient of the award for their respective year.

3.1 Logistic Regression

For our logistic regression model we used the scikit-learn python package. We used logistic regression on data sets that both excluded and included the synthetic MVP data. We did this to analyze the usefulness of synthetic data and to determine how much synthetic data to use in later models. The 'lbfgs' solver was used for gradient descent and 30% of our data set was randomly selected for training.

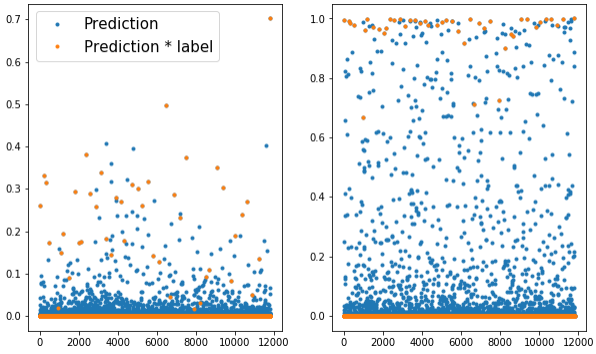

Figure 1: Scatter plot of the prediction probabilities from our logistic regression model. The left does not include the synthetic data; the right includes about equal parts synthetic data and real data.

3.1.1 No Synthetic Data

The logistic regression gets more than 99% of predictions correct. Unfortunately, this metric is meaningless when assessing the quality of our predictions. Our model only predicts two MVPs in our entire data set (both are correct). Using "true score" as defined above the model predicts 57.4% of MVPs correctly on average over 1,000 trials. A scatter plot of results appears in Figure 1.

3.1.2 Synthetic Data

We chose to test our logistic regression on 10 different test sizes of synthetic data. Each amount of synthetic data was run for 10,000 iterations. Plots of our three main metrics appear in Figure 2.

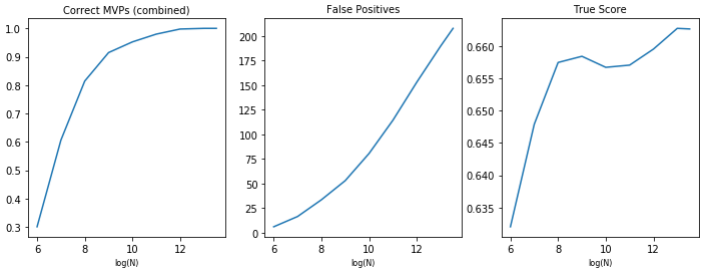

Figure 2: Plots of the percentage of correct predictions, the number of false positives, and the "true score" of the data. Each metric is plotted against $\log_2(N)$.

3.2 Support Vector Machine

We also used the scikit-learn package for training our SVM model. Instead of training a linear SVM classifier, we selected the polynomial kernel with degree of three (Figure 3) to be in use. We also set a penalty term with the penalty parameter 1.0. We still trained the classifier on 30% of our combined data, and then tested all three measures on the original data. The model gives us an average 77.1% true score over 1,000 trials. A scatter plot of results is shown in Figure 4.

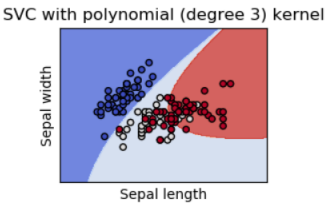

Figure 3: SVM kernel

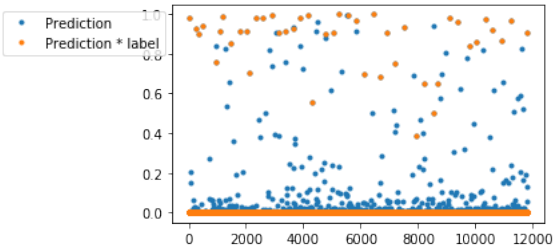

Figure 4: Scatter plot of the prediction probabilities from our SVM model (on combined data).

3.3 Neural Network

We used the pytorch library in order to construct our neural network, which provides functions for common neural network layer types and functionalities. The network consists of four layers - first three LeakyReLU layers, then a sigmoid layer in order to center the values around zero. Our first layer includes 37 features, while the inner layers only have 16. The output of the network is one value between 0 and 1 for each player indicating how likely the network believes it is that that player is an MVP. Unlike our other tests, for the neural network we used 5-fold cross-validation in order to train our neural network, training on one half at a time and testing on the other. We used SGD optimizer as the parameter optimizer and manually set the learning rate to be 0.1, the momentum parameter to be 0.9, and the batch size to be 128. For each run, we ran back-propagation for 5000 epochs to train a network. Finally, we ran 10 times for evaluating performance. (Figure 6)

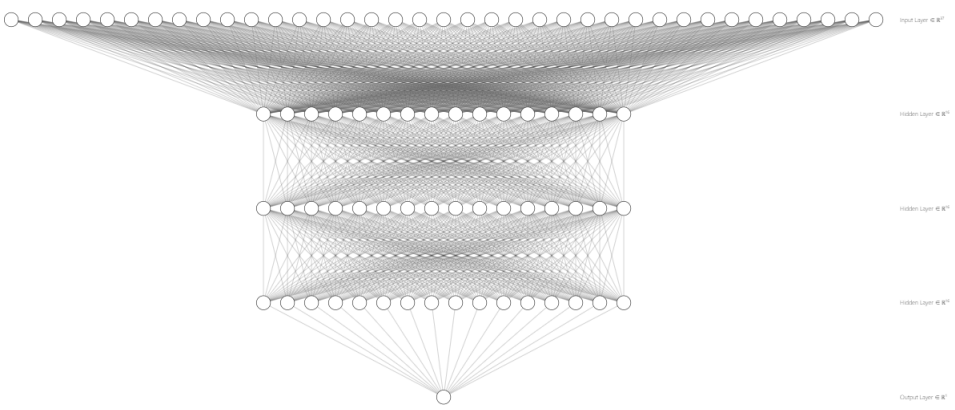

Figure 5: Structure of our neural network

4 Conclusions

4.1 Synthetic Data

Using increasing amounts of synthetic data has a few major trade offs. Looking at Figure 2 we can see that as the percentage of correct predictions and the "true score" go up the number of false positives increases at a faster rate. Additionally, there are run time considerations when generating and using more data.

We had to choose how much synthetic data to include with our future models. We can see that more synthetic data causes diminishing returns and exponentially increasing false positives. So, we decided to use just 500 synthetic samples for

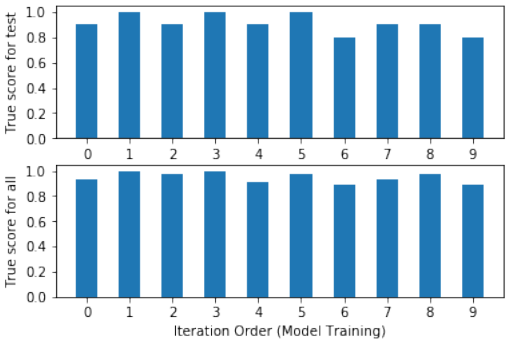

Figure 6: The top plot shows the "true score" associated with the test data for each of our 10 runs of the neural network. The lower plot shows the "true score" for the entire data set.

all models after our initial logistic regressions. Using 500 synthetic samples greatly reduces the number of false positives and lies near where both the MVP prediction percentage and "true score" curves begin to plateau.

4.2 Comparing Models

Our three models have drastically varying amounts of complexity. Our linear regression model has just 1 coefficient per feature; 37 in our case. The SVM has a coefficient for each support vector; 144 for our model. Finally, for each layer of the neural network there is $(n_{input} + 1)n_{nodes}$. Our neural network has a total of 1,169 coefficients. It is important to keep in mind the complexity of our network when comparing our results.

Below is a comparison of different values given by our different models when trained on the same dataset. The SVM and neural network used 500 samples of synthetic data.

	LR w/o synthetic data	LR w/ synthetic data	SVM	Neural Network
Accuracy	1.9%	88.7%	95.3%	100%
True Score	57.4%	65.6%	77.1%	94.8%
False Positives	0.21	54.4	52.48	15.3

As you can see in the table, the neural network performs the best - guessing all MVPs correctly as well as having fewer false positives than the SVM or logistic regression models. This makes sense, because the neural network requires many more stored weights and adjustments than the other methods.

The SVM is the runner-up, performing almost as well as the neural network. Our choice of kernel likely greatly impacted the performance of the SVM due to the fact that our data is not easily separable by a simple surface in 37-d space. However, even though a non-linear kernel was used, the relatively few coefficients could still limit the SVM's ability to classify data, and this may be part of the reason why it performs worse than the neural network. But if less computational resources are available, SVM could be a better choice.

The logistic regression model performs the worst, as it can only represent data as the sigmoid of a linear function derived from the parameters. The model performs particularly badly when using no synthetic data because the model is rewarded for predicting nearly no MVPs. The addition of synthetic data fixes this problem at the cost of false positives. With 500 points of synthetic data included the model predicts more false MVPs than there are actual MVP winners. The logistic model is useful for determining what features are valuable in predicting MVP and is easier to visualize than our other models.

The neural network's 100% accuracy on MVP's prediction could mean that some over-fitting happened during training. Though we did only train the neural network on half the original data set, over-fitting could still be possible because we didn't use any regularization techniques like L2-regularization and weight decay to carefully avoid it. Besides, an alternative explanation of 100% accuracy could be that we set 0.5 as a threshold for deciding MVP, though it turned out that the accuracy was 100%, 0.5 could be too intuitive to be a practical and reasonable threshold. More importantly, please also note the difference between true score and accuracy - accuracy checks the overall percentage correct, comparing all selected MVPs to actual MVPs, while the true score selects one MVP per year and compares it to the actual MVP for that year.

For our purposes, we'd likely choose the neural network for this task due to its superior performance. This doesn't apply in our case, but usually the choice of the model would depend on other factors as well, like the computational space and time available. The large number of parameters and difficult computation required for the neural network might make it unrealistic to use in some cases.

Our code repository is https://github.com/Lrab/COMP562_Final.

5 References

1. Chawla, Nitesh. SMOTE: Synthetic Minority Over-Sampling Technique, www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/ch
   html/chawla2002.html.
2. "Documentation of Scikit-Learn 0.20.3." Documentation of Scikit-Learn, scikit-learn.org/stable/documentation.html.
3. "PyTorch Documentation." PyTorch Documentation, pytorch.org/docs/stable/index.html.
4. "Matplotlib 3.0.3 Documentation." Matplotlib Version 3.0.3, matplotlib.org/contents.html.
5. Jain, Yashvardhan. "Create a Neural Network in PyTorch - And Make Your Life Simpler." Medium, Coinmonks, 27 June 2018, medium.com/coinmonks/create-a-neural-network-in-pytorch-and-make-your-life-simpler-ec5367895199.