Predicting The NBA MVP

Mohammed Bukhattala, Justin Clifton, Maxwell Pitney, Graham Swain

05/04/2021

The NBA’s most valuable player award is given to one player every NBA season. There have been 65 different players who have won this award. In this paper, we offer several models that attempt to predict the NBA MVP. These models will be using individual player data tracked over the course of a season.

# Introduction

The most prized individual accolade for a player in the NBA is the Most Valuable Player award. Winning this award signifies that coaches, the media, and former players thought that the winner was the most important player in the NBA. There have been 65 different players who have won this award. These players played different positions, had different statistics, and come from different eras. What then, do they have in common? Given player statistics, can we accurately predict what player will win the MVP award in a given year? In this paper, we attempt to answer that very question. In order to do this, we frame the problem in terms of classification. Thus, we will create binary classifiers that, when given player statistics can classify that player as an MVP or not. Some models we will be using include LDA, KNN, and decision trees.

# Data Source and Variable Description

We will be considering data about player statistics tracked over the course of a season. Our data is a combination of two other datasets. The first is a dataset of NBA season statistics from the years 1950-2017 contained at <https://www.kaggle.com/drgilermo/nba-players-stats>. The data in this dataset was scraped from www.basketball-reference.com, a website dedicated to tracking basketball statistics. In order to get the players that won the MVP award, more data was scraped from basketball-reference using the beautifulSoup python package. These two datasets were then combined to form the one that is used in this project.

### Variable Description

Since the data was pulled from basketball-reference, descriptions of variables can be found on their website at: <https://www.basketball-reference.com/about/glossary.html>

# Methods

### Scoring Our Models

Before we can discuss the actual modeling of the problem, it is important to understand how the accuracy of models will be assessed. In a classification problem, there a few metrics that can be considered. Some of the metrics that will be considered are standard to all classification problems such as accuracy, true positive rate, false positive rate, true negative rate, and false negative rate. In the context of our problem domain, a true positive happens when a player was predicted as the MVP and actually won the MVP. A true negative happens when a player was predicted as a non-MVP and was not the MVP. A false positive occurs when a player was predicted as the MVP but did not win the MVP. A false negative occurs when a player was predicted as a non-MVP but actually won the MVP. The metric of most interest for this problem is the true positive rate. It is important to note that many of the models contained in this paper will have very high accuracy. This is due to the fact that there are so few players who win the MVP and so many that do not win that any model can simply guess non-MVP for every player and achieve high accuracy. On this account, we will be using metrics other than accuracy to score our models. These metrics are the f1 score and Matthews Correlation Coefficient (MCC). The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal (scikit).

The MCC produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements in the dataset (Chicco, Gurman).

These metrics will be calculated using the following functions where table is the confusion matrix:

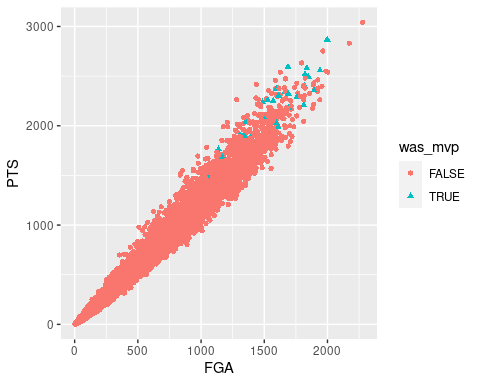
truePositive <- function(table) {  
 tp = table[2,2]  
 tn = table[1,1]  
 fp = table[1,2]  
 fn = table[2,1]  
 return(tp/(tp + fn))  
}  
  
falseNegative <- function(table) {  
 tp = table[2,2]  
 tn = table[1,1]  
 fp = table[1,2]  
 fn = table[2,1]  
 return(fn/(tp + fn))  
}  
  
falsePositive <- function(table) {  
 tp = table[2,2]  
 tn = table[1,1]  
 fp = table[1,2]  
 fn = table[2,1]  
 return(fp/(fp + tn))  
}  
  
f1Score <- function(table) {  
 tp = table[2,2]  
 tn = table[1,1]  
 fp = table[1,2]  
 fn = table[2,1]  
 return(2\*tp/(2 \* tp + fp + fn))  
}  
  
mcc <- function(table) {  
 tp = as.double(table[2,2])  
 tn = as.double(table[1,1])  
 fp = as.double(table[1,2])  
 fn = as.double(table[2,1])  
 divisor = sqrt((tp + fp) \* (tp + fn) \* (tn + fp) \* (tn + fn))  
 return(((tp \* tn) - (fp \* fn))/divisor)  
}

# Validating the Data

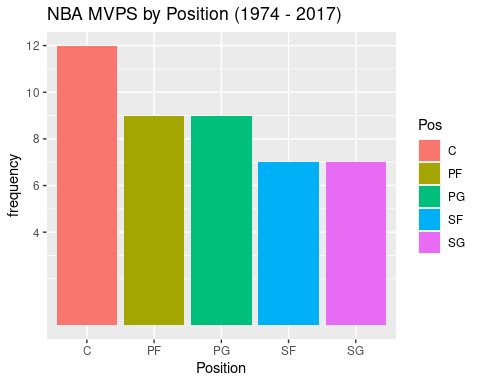
The first problem to consider is how to split the data into training and testing sets. Initially, we simply split the data randomly. However, while this method may be valid in most contexts, it is not the most appropriate method for our problem domain. As discussed previously, a very small amount of players have won the MVP award, and our data contains an even smaller subset of those players. Therefore, if random sampling is performed it is entirely possible that our training sets contain 0 MVP players. For this reason, we have instead decided in some cases to partition the data based on the year with training set given by years prior to 2000 and testing set given by years later than 2000.

# Plots and Data Analysis

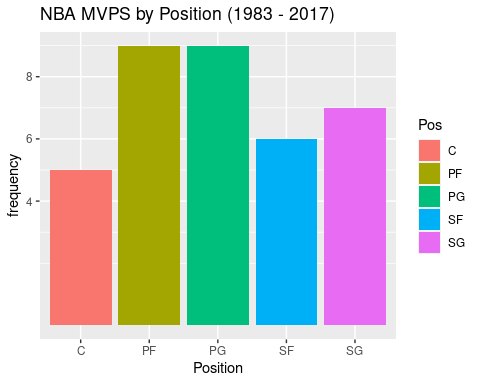
First, we will do some preliminary data analysis and consider a few plots of the data. In our first plot, we produce a scatter plot displaying field goal attempts (i.e. the number of shots taken) on our x axis, and the number of points scored on the y axis. Red circles indicate players who did not win the MVP award, and blue triangles correspond with players who did win. As can be seen from the plot, MVP players tend to score higher and take more shots.



Next, we consider a breakdown of all MVP players by their position. This data comes from seasons between 1974 - 2017. We see that the center position has the most MVP players between this time frame.

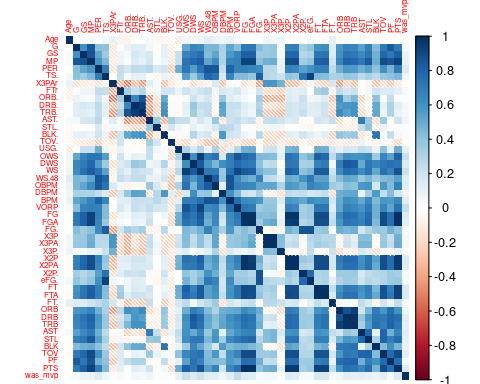


Unfortunately, statistics were not tracked properly prior to 1983, and therefor we omit this data from consideration. Below we display a histogram corresponding to all MVP players where accurate data was collected.

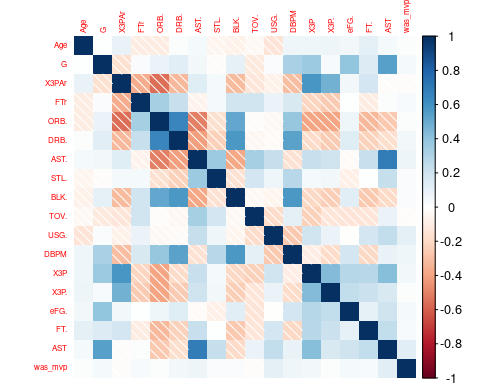


# Modeling and Results

### LDA

The first model that we will consider is an LDA model. It is important to remember that one of the assumptions of LDA is that none of the features are heavily correlated. We will first examine a correlation matrix of our data. 

It can be seen that there are many heavily correlated features. We will now drop features with correlation higher or lower than than |.7|.

We now offer the new correlation matrix for potential features of the LDA model. 

With heavily correlated features removed, an LDA model can be considered. This model was fit on a random split of the data. All non correlated variables were used to predict if the MVP.

#### First LDA Results

ldaTable

##   
## FALSE TRUE  
## FALSE 7224 8  
## TRUE 49 12

accuracy

## [1] 0.9921843

truePositive(ldaTable)

## [1] 0.1967213

falseNegative(ldaTable)

## [1] 0.8032787

falsePositive(ldaTable)

## [1] 0.001106195

f1Score(ldaTable)

## [1] 0.2962963

mcc(ldaTable)

## [1] 0.3406628

The first model had many high false positives. The low true positive rate and MCC reflects that.

#### Second LDA Results

This model was fit on years prior to 2000 and tested on data from years greater than 2000.

ldaTable

##   
## FALSE TRUE  
## FALSE 7798 9  
## TRUE 35 8

accuracy

## [1] 0.9943949

truePositive(ldaTable)

## [1] 0.1860465

falseNegative(ldaTable)

## [1] 0.8139535

falsePositive(ldaTable)

## [1] 0.001152812

f1Score(ldaTable)

## [1] 0.2666667

mcc(ldaTable)

## [1] 0.2935688

The second model performed marginally worse. The new training and testing set did improve the false negatives, however, the true positive and MCC were slightly lowered.

### LOOCV LDA

We will now perform an LDA using leave one out cross validation on the original training and testing set.

ldaTable

##   
## FALSE TRUE  
## FALSE 14460 20  
## TRUE 89 16

accuracy

## [1] 0.9925266

truePositive(ldaTable)

## [1] 0.152381

falseNegative(ldaTable)

## [1] 0.847619

falsePositive(ldaTable)

## [1] 0.001381215

f1Score(ldaTable)

## [1] 0.2269504

mcc(ldaTable)

## [1] 0.2572689

Leave one out cross validation further worsened performance.

### KNN

We will now perform KNN using the randomly split training and testing sets.

knnTable1

## testY  
## knnPred1 FALSE TRUE  
## FALSE 7263 19  
## TRUE 8 3

accuracy

## [1] 0.9962978

truePositive(knnTable1)

## [1] 0.2727273

falseNegative(knnTable1)

## [1] 0.7272727

falsePositive(knnTable1)

## [1] 0.002609173

f1Score(knnTable1)

## [1] 0.1818182

mcc(knnTable1)

## [1] 0.1911468

knnTable2

## testY  
## knnPred2 FALSE TRUE  
## FALSE 7265 17  
## TRUE 6 5

accuracy

## [1] 0.9968463

truePositive(knnTable2)

## [1] 0.4545455

falseNegative(knnTable2)

## [1] 0.5454545

falsePositive(knnTable2)

## [1] 0.002334523

f1Score(knnTable2)

## [1] 0.3030303

mcc(knnTable2)

## [1] 0.3200032

knnTable3

## testY  
## knnPred3 FALSE TRUE  
## FALSE 7270 21  
## TRUE 1 1

accuracy

## [1] 0.9969834

truePositive(knnTable3)

## [1] 0.5

falseNegative(knnTable3)

## [1] 0.5

falsePositive(knnTable3)

## [1] 0.002880263

f1Score(knnTable3)

## [1] 0.08333333

mcc(knnTable3)

## [1] 0.1500932

knnTable4

## testY  
## knnPred4 FALSE TRUE  
## FALSE 7270 22  
## TRUE 1 0

accuracy

## [1] 0.9968463

truePositive(knnTable4)

## [1] 0

falseNegative(knnTable4)

## [1] 1

falsePositive(knnTable4)

## [1] 0.003017005

f1Score(knnTable4)

## [1] 0

mcc(knnTable4)

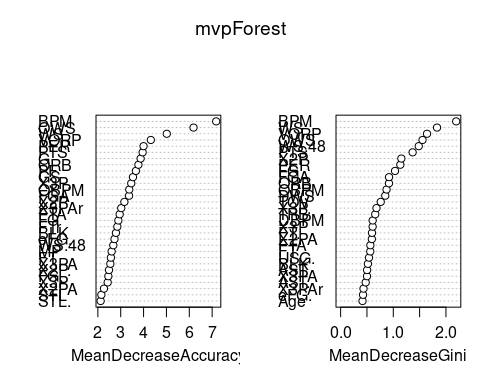
## [1] -0.0006441558

We find that for knn, a k = 2 is the most effective model with an mcc score of . That being said, none of these models are particularly effective.

### Random Forest

Next, a random forest is fit to the data. This random forest serves two purposes. First, it will act as a sufficient model but it will also be used to select important features.

varImpPlot(mvpForest, sort=TRUE)



forestTable

##   
## predTest FALSE TRUE  
## FALSE 7271 16  
## TRUE 1 4

accuracy

## [1] 0.9976687

truePositive(forestTable)

## [1] 0.8

falseNegative(forestTable)

## [1] 0.2

falsePositive(forestTable)

## [1] 0.002195691

f1Score(forestTable)

## [1] 0.32

mcc(forestTable)

## [1] 0.3993134

In the plot above, we see some of the important features are the advanced statistics BPM, Win Shares and Value over replacement. We will use these and some of the other important features to fit more models.

We now consider logistical regression models.

### Logistical Regression

The first model will use all uncorrelated features.

logTable

##   
## predLog FALSE TRUE  
## FALSE 7265 14  
## TRUE 8 6

accuracy

## [1] 0.9976687

truePositive(logTable)

## [1] 0.4285714

falseNegative(logTable)

## [1] 0.5714286

falsePositive(logTable)

## [1] 0.001923341

f1Score(logTable)

## [1] 0.3529412

mcc(logTable)

## [1] 0.3571066

Now, we will begin using features that were indicated as important by the random forest.

This model uses VORP + WS + BPM + PER + PTS + AST + TRB + BLK + STL + TOV + GS.

logTable

##   
## predLog2 FALSE TRUE  
## FALSE 7256 14  
## TRUE 15 8

accuracy

## [1] 0.9960236

truePositive(logTable)

## [1] 0.3478261

falseNegative(logTable)

## [1] 0.6521739

falsePositive(logTable)

## [1] 0.001925722

f1Score(logTable)

## [1] 0.3555556

mcc(logTable)

## [1] 0.3536501

This model uses VORP + WS + BPM + PER + PTS + AST + TRB + BLK + STL + TOV + GS and is fit on the data split by year.

logTable

##   
## predLog2 FALSE TRUE  
## FALSE 7831 11  
## TRUE 2 6

accuracy

## [1] 0.9983439

truePositive(logTable)

## [1] 0.75

falseNegative(logTable)

## [1] 0.25

falsePositive(logTable)

## [1] 0.001402703

f1Score(logTable)

## [1] 0.48

mcc(logTable)

## [1] 0.5138284

This model will be making use of interaction between some of the features and uses VORP + WS \* OWS \* WS.48 + BPM + PER + PTS \* FG. + AST + TRB + BLK + STL + TOV + GS.

logTable

##   
## predLog2 FALSE TRUE  
## FALSE 7829 8  
## TRUE 4 9

accuracy

## [1] 0.9984713

truePositive(logTable)

## [1] 0.6923077

falseNegative(logTable)

## [1] 0.3076923

falsePositive(logTable)

## [1] 0.001020799

f1Score(logTable)

## [1] 0.6

mcc(logTable)

## [1] 0.6046672

# Conclusion

This research compared different classification machine learning algorithms with the goal to predict the NBA MVP. This was done using data scraped from basketball-reference.com. Through the course of building and testing various classification models, we offer models that can accurately predict non-MVP players, and predict MVP winners moderately well.

The nature of the MVP award means That only one player per season gets chosen. This introduces issues as there is an inherently limited number of MVP players. Being such a rare event led to all models having high accuracy, but low true to middle true positive rates. Various data splitting methods and multiple models were explored, however, none were particularly successful. Logistical regression using the features found by random forests as well as including interaction terms offered the best model with an MCC of 0.60.

It is the speculation of the authors that superior models could be found if the data contained features involving team success (games won by the team) and popularity of the players (Twitter followers). Another change that could yield more accurate results would be to, instead of classifying players as MVP or not, consider using regression to predict the number of MVP votes that each player received. These speculations serve as recommendations for any future research.

# References

Chicco, D., Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 6 (2020). <https://doi.org/10.1186/s12864-019-6413-7>

“Sklearn.metrics.f1\_score.” Scikit, scikit-learn.org/stable/modules/generated/sklearn.metrics.f1\_score.html.