VANDERBILT
UNIVERSITY

Big Data Final Project:

NBA All-Stars Predictions

Presented by: Bryce Crawford & Matt Owens
Date: April 30, 2016

Overview

1 Overview

1.1 Problem Statement

All-star selection in the NBA is a two-pronged process: the stars are selected by the votes of the general public while the bench is filled out with the votes of NBA coaches. By these measures, we expected popularity to drive fan selection while statistical output would drive coach selection. Because of the dual nature of NBA all-star selection, we felt motivated to quantify how effective measures of player popularity and on court skill are at predicting whether or not a player will be an all-star in a given year. We pulled data from a few sources, Wikipedia and Google Trends to measure player popularity, and basketball-reference.com to measure player statistics and generate a dependent data set of player information. To conduct the analysis we used Python for data preprocessing and aggregation, PySpark and Hadoop FS to analyze Wikipedia data and Scikit-Learn to generate predictions.

1.2 Team Members

Matt Owens was responsible for the NBA statistics and Wikipedia data sources. Bryce Crawford was responsible for the Google Trends data source, and both team members were responsible for combining the data sources in our prediction.

1.3 Installation & Running

The instructions for installation and how to run our predictions can be found in the README.md file of our repository (https://github.com/vu-bigdata-2016/final-project.git).

1.4 Evaluation Metrics

Because our model is a binary prediction (i.e. all-star or not, starter or not) with labels, we used three supervised machine-learning classifiers: Support Vector Machine (SVM), Gaussian Naïve Bayes, and Random Forest. To produce the greatest effectiveness, we used the test_train_split module of Scikit Learn to test and train our classifiers, as well as perform cross validation.

To measure the effectiveness of each of these classifiers in our predictions, we used the following metrics: Area Under the Curve (AUC), accuracy score, precision score, recall score, and F1 score. AUC is based on the rate of true positives versus false positives found and is an effective measurement of the overall performance of a classifier in a prediction. The accuracy score measures the accuracy of classifier. The precision score measures the ratio tp / (tp + fp), where tp is the number of true positives and fp is the number of false positives. In essence, precision is the ability of a classifier to not label a sample as positive that is actually negative. The recall score measures the ratio tp / (tp + fn), where fn is the number of false negatives. Essentially, recall is the ability of a classifier to find all positives samples. The F1 score is basically a weighted average of precision and recall, providing another

helpful indicator of the overall effectiveness of a classifier. All of these scores reach their best value at 1 and their worst value at 0.

2 Dependent Data Source

We could not find any single data sources that would give us all the information we needed for NBA all-stars over the past 11 years, so we had to manually aggregate the data from multiple sources. For each season since 2006, we looked up the players in the All-Star game, as well as information for each of them, including whether they were a starter, were injured and could not play, replaced an injured player, and what conference they belonged to. Next, we manually inputted this data into a CSV file and executed a command in Vim to reformat it into a Python-readable text file.

Altogether, the aggregated dependent data source is 288 rows with 5 features: Player, Year, Conference, Injured, Replacement, and Starter. Of these features, we ended up using Player, Year, and Starter in our predictions. For each prediction, we loaded this data into our Python program by opening the CSV file, reading and parsing the lines into a list, and creating a dictionary where each key is a tuple of a player's name and the year and each value is a 1, indicating that the player was an all-star during that year. For the starters prediction, we also checked whether the player started in the All-Star game and only included these players in the dictionary.

3 Independent Data Sources

3.1 NBA Player Statistics

3.1.1 Data

For this independent data source, we took advanced player statistics for the last 11 seasons (2006-2016). The statistics for each season could be downloaded as a CSV file, which we then copied onto the cluster. Each of these files was between 540 and 677 rows with 27 features, for a total data size of 881KB.

3.1.2 Process

To load the data into our Python program, we iterated over each file, reading each line and splitting the features by comma, deleted unnecessary lines and empty columns, and aggregated the processed data into a single list.

For our baseline prediction, we used the Player Efficiency Rating (PER) as our feature. This seemed like the natural choice because this rating is meant to represent a player's effectiveness on a minute-by-minute basis. We constructed our input matrix by initializing a one-dimensional array with zeros, iterating over the list of players, and assigning each cell the player's PER.

For our enhanced prediction, we added in the remaining 21 relevant features, including games played, minutes played, true shooting percentage, 3-point attempt rate, free throw attempt rate, offensive rebound percentage, defensive rebound

percentage, total rebound percentage, assist percentage, steal percentage, block percentage, turnover percentage, usage percentage, offensive win shares, defensive win shares, total win shares, win shares per 48 minutes, offensive box plus/minus, defensive box plus/minus, total box plus/minus, and value over replacement player. We excluded the index, player, position, age, and team features because they are merely descriptive. We then added these features to our input matrix, assigning zeros to cells with no statistics.

Finally, we combined our enhanced prediction with our PageRank prediction using Wikipedia data (described in Section 3.3). We constructed the input matrix in the same way as before, except that we added the PageRank feature by checking whether there was a corresponding PageRank value for each NBA player and adding it to the end of each row in the matrix. If a player did not have a Wikipedia page, we assigned 0 to the cell.

For each prediction, we used the same output matrix, which we created by initializing a one-dimensional array with zeros, iterating through the player list and checking whether each player/year combination existed in the all-stars dictionary, and assigning 1 to the corresponding cell in matrix if the player was an all-star that year.

3.1.3 Results

For our enhanced prediction with the rest of the features added in, the Random Forest classifier performed the best by a significant margin. Even for our baseline, this classifier produces rather impressive results:

```
aucs:  [0.91275333662876912]
aucs mean:  0.912753336629
accuracy_score:  0.960377358491
f1_score:  0.46835443038
precision_score:  0.606557377049
recall_score:  0.381443298969

classification report:
             precision    recall  f1-score   support

        0.0       0.97      0.99      0.98      2023
        1.0       0.61      0.38      0.47        97

avg / total       0.95      0.96      0.96      2120
```

Although this resulted in high AUC and accuracy scores, the precision and recall scores were still lower than we would have liked them to be. Fortunately, we vastly improved in these areas by adding in other features:

```
aucs:  [0.98005666790670187]
aucs mean:  0.980056667907
accuracy_score:  0.97641509434
f1_score:  0.709302325581
precision_score:  0.813333333333
recall_score:  0.628865979381
```

```
classification report:
             precision    recall  f1-score   support

        0.0       0.98      0.99      0.99      2023
        1.0       0.81      0.63      0.71        97


avg / total       0.97      0.98      0.97      2120
```

Not only did the F1 score increase from 0.47 to 0.71, but the AUC score increased to 0.98 as well. In fact, this was the best overall all-stars prediction out of all of our prediction models and classifiers. Interestingly, it performed slightly better than the same prediction combined with our PageRank prediction, which on its own performed very well also (see Section 3.3.3). Here are the results from the combined prediction:

```
aucs:  [0.96149181322013344]
aucs mean:  0.96149181322
accuracy_score:  0.97358490566
f1_score:  0.670588235294
precision_score:  0.780821917808
recall_score:  0.587628865979

classification report:
             precision    recall  f1-score   support

        0.0       0.98      0.99      0.99      2023
        1.0       0.78      0.59      0.67        97


avg / total       0.97      0.97      0.97      2120
```

It should be noted that we tested numerous Random State value when performing cross validation, and we found that random_state = 46 produced the best results.

Runtime: 1 minute, 32 seconds

3.2 Google Trends

3.2.1 Data

In order to gain data on player popularity we first turned to Google Trends. While it would have been much simpler to get statistics on keyword searches for the players we were examining over a given time frame, we found that Google keyword search data was not available for free. Instead, we used Google Trends, which uses a system of relative search popularity. For example, if Google Trends is asked for popularity of the terms “basketball” and “baseball” on a monthly basis since 2010, there would be a single month with a score of ‘100’ for exclusively either basketball or baseball. This single month score of ‘100’ represents the most popular point for a term over a given time frame, any score of ‘50’ would thus mean a term was half as popular as the most popular point over the same time frame.

We used Google Trends data from October 2015 to January 2016 to capture player popularity up to the 2016 all-star data. In total we analyzed the popularity of all the 2016 all-stars and the next top 48 players by PER. Unfortunately, the unofficial API for Google Trends was not functional due to a recent change in the Google Trends back end so we were unable to collect more than 72 players worth of data in a reasonable amount of time.

3.2.2 Process

In order to generate our features, we wrote a Python script to extract relevant information from the Google Trends reports for each player. To generate the player "popularity index" feature, we first grabbed Google Trends reports for each player's relative popularity in comparison to the most popular player over the given time frame. In this case, the most popular player was Lebron James. To calculate the index, we simply summed the total weekly popularity scores of both the given player and Lebron James and then divided the total given player score by Lebron's total score. Because we found there to be a very large disparity in popularity index between the top 4 players and the rest of the players, we decided to include a second feature which was a simple ranking of player popularities where the least popular player had a score of 1 and the most popular player had a score equal to the total number of players (in this case, 72).

After considering that all star games selections tend to have a ratio of three forwards/center per two guards we also wanted to consider positional player popularity, that is, a player's popularity compared to other players at the same position. As such, we introduced a "positional player popularity index" and a "positional player popularity ranking" feature. We only needed to recalculate positional player popularity indexes for guards as Lebron James is considered a forward. The most popular guard for the 2015-16 season was Stephen Curry, so we pulled Google Trends data for each guard compared to him and ran a Python script to generate their new popularity indexes. We recognized that forwards and centers would have much higher positional popularity ranking scores as there were much more of them. In order to calculate for this disparity, we multiplied each forward/center's positional popularity ranking by the ratio of guards to forwards (0.70) in the data set.

3.2.3 Results

The results for all of our predictions using the Google Trends dataset can be found in Appendix A. There are a few important things from these results to note. First, we tested a number of different random_state values, and 65 produced the best results (and the results seen in Appendix A). In all cases, the Gaussian Naïve Bayes classifier produced the best results, so these are the results we have included in Appendix A.

Surprisingly, the additional features added to the baseline prediction did not improve the effectiveness of the prediction at all. However, combining it with the PageRank prediction increased both the precision score (from 0.75 to 0.80) and the recall score (from 0.33 to 0.44), although it did not affect the AUC score. In general, predicting only starters instead of all all-stars did not change results much at all.

Runtime: 48 seconds

3.3 Wikipedia PageRank

3.3.1 Data

For this independent data source, we used a subset of the Wikipedia database, which consisted of all Wikipedia pages containing the word 'ball' somewhere in them. This includes virtually every page related to basketball, including pages for many NBA players and almost every all-star player. Because of this, we were able to use this data source to calculate the PageRank of each player, which actually turned out to be an effective alternative to Google Trends for predicting NBA all-stars based on popularity. The entire dataset was just over 5GB.

3.3.2 Process

The process for predicting all-stars with this dataset consisted of two parts: calculating the PageRank for the Wikipedia page of each player and using it as a feature in our machine-learning model.

First, we used Spark to calculate PageRank. For each Wikipedia page, we parsed through the body to find links, denoted by [[title]]. We saved each pair of page titles and links to a text file in Hadoop FS, which we then used with a PageRank algorithm to calculate the PageRank for each page.[3]

Next, we loaded the input for our machine-learning model by once again parsing through the CSV files of NBA player statistics and creating a dictionary where each key is a player's name and each value is his PageRank, which we initially set to 0. Then we iterated through the text file of page titles and PageRanks that we saved to Hadoop FS. For each line in the file, we used a third party library to convert it back into a tuple of page titles and ranks, removing all instances of '(basketball)' from titles in order to account for NBA players with the same names as other people with Wikipedia pages. Finally, we checked if a page title corresponded to an NBA player's name in our dictionary, and if so, we set the value for that key in the dictionary to the page's PageRank.

We created the input matrix for our prediction by creating a one-dimensional array with the same length as the player dictionary and initializing it with zeros. Then for each player in the dictionary, we set the corresponding entry in the input matrix to the player's PageRank. We created the output matrix in a similar way, constructing a one-dimensional array with zeros and iterating through the player dictionary and checking whether each player was in the all-stars dictionary.

It should be noted that we do not have a baseline for this prediction, since the only relevant feature from our data source is PageRank. This prediction was only meant to be an alternative and supplement to our popularity prediction using Google Trends, since our Google Trends prediction did not produce satisfactory results.

3.3.3 Results

With random_state = 33, all of our classifiers produced surprisingly good results:

```
SVC:

aucs:  [0.94285714285714284]
aucs mean:  0.942857142857
```

```
accuracy_score:  0.898734177215
f1_score:  0.272727272727
precision_score:  0.75
recall_score:  0.166666666667

classification report:
             precision    recall  f1-score   support

        0.0       0.90      0.99      0.95       140
        1.0       0.75      0.17      0.27        18

avg / total       0.89      0.90      0.87       158

GaussianNB:

aucs:  [0.96646825396825398]
aucs mean:  0.966468253968
accuracy_score:  0.955696202532
f1_score:  0.774193548387
precision_score:  0.923076923077
recall_score:  0.666666666667

classification report:
             precision    recall  f1-score   support

        0.0       0.96      0.99      0.98       140
        1.0       0.92      0.67      0.77        18

avg / total       0.95      0.96      0.95       158

RandomForestClassifier:

aucs:  [0.90634920634920635]
aucs mean:  0.906349206349
accuracy_score:  0.873417721519
f1_score:  0.52380952381
precision_score:  0.458333333333
recall_score:  0.611111111111

classification report:
             precision    recall  f1-score   support

        0.0       0.95      0.91      0.93       140
        1.0       0.46      0.61      0.52        18


avg / total       0.89      0.87      0.88       158
```

Although Gaussian Naïve Bayes did not perform as well as other classifiers in our other predictions, it actually produced the best results for our PageRank prediction, with an AUC score of 0.96, a precision score of 0.92. It actually produced the best F1 score (0.77) out of all of our predictions, arguably making this prediction the best of all.

Runtime: 3 minutes, 26 seconds

4 Future Work

In the future we believe that we can improve our prediction models through the incorporation of other data sets and expanding the collections of current ones. We were limited by the lack of API for Google Trends; given additional time we could have collected a larger Trends data set that might have had an impact on predictions. Additionally we would like to expand the model to include seasons predating the 2015-16 season for our Google Trends prediction and the 2005-06 season for our NBA stats prediction.

5 Sources

1. http://www.basketball-reference.com/leagues/
2. http://www.nba.com/allstar/
3. https://github.com/apache/spark/blob/master/examples/src/main/python/pagerank.py
4. https://www.google.com/trends/

6 Appendices

6.1 Appendix A

```
Predicting all allstars with baseline:

aucs:  [0.91269841269841268]
aucs mean:  0.912698412698
accuracy_score:  0.695652173913
f1_score:  0.461538461538
precision_score:  0.75
recall_score:  0.333333333333

classification report:
             precision    recall  f1-score   support

        0.0       0.68      0.93      0.79        14
        1.0       0.75      0.33      0.46         9

avg / total       0.71      0.70      0.66        23


Predicting all allstars with enhanced features:

aucs:  [0.91269841269841268]
aucs mean:  0.912698412698
accuracy_score:  0.695652173913
f1_score:  0.461538461538
precision_score:  0.75
recall_score:  0.333333333333

classification report:
             precision    recall  f1-score   support

        0.0       0.68      0.93      0.79        14
        1.0       0.75      0.33      0.46         9

avg / total       0.71      0.70      0.66        23


Predicting all allstars with enhanced features combined with
pagerank:

aucs:  [0.90476190476190477]
aucs mean:  0.904761904762
accuracy_score:  0.739130434783
f1_score:  0.571428571429
precision_score:  0.8
recall_score:  0.444444444444

classification report:
```

```
             precision    recall  f1-score   support

        0.0       0.72      0.93      0.81        14
        1.0       0.80      0.44      0.57         9

avg / total       0.75      0.74      0.72        23


Predicting only starters with enhanced features:

aucs:  [0.91269841269841268]
aucs mean:  0.912698412698
accuracy_score:  0.695652173913
f1_score:  0.461538461538
precision_score:  0.75
recall_score:  0.333333333333

classification report:
             precision    recall  f1-score   support

        0.0       0.68      0.93      0.79        14
        1.0       0.75      0.33      0.46         9

avg / total       0.71      0.70      0.66        23


Predicting only starters with enhanced features combined with
PageRank:

aucs:  [0.90476190476190477]
aucs mean:  0.904761904762
accuracy_score:  0.739130434783
f1_score:  0.571428571429
precision_score:  0.8
recall_score:  0.444444444444

classification report:
             precision    recall  f1-score   support

        0.0       0.72      0.93      0.81        14
        1.0       0.80      0.44      0.57         9

avg / total       0.75      0.74      0.72        23
```