## NBA Prediction Model Based on Player Lineups

By Brian Ramm

## Summary:

For my final project in CS777, I explored making a model to predict betting NBA games. I’ve found a dataset on Kaggle that includes NBA game information, player statistics, and betting lines. By combining these together, I created a model that predicts NBA game results and incorporates Vegas odds to select a side to place betting wise. This data only relies on information that is known prior to the game such as the players in the lineup and the season the game takes place. The results were quite successful and resulted in profit if a user placed identical bets based off the final model’s prediction on every game.

## Data Preparation

The data preprocessing section begins with the acquisition of two datasets: “nba\_betting\_money\_line.csv" and "nba\_games\_all.csv." The first step involves correcting column names in the "bets" DataFrame for improved clarity, ensuring proper alignment of team identifiers and pricing information. I identified an error in the dataset where the home and away teams were mislabeled and needed to be corrected. Subsequently, a single book, "Bovada," is selected from the betting data, and irrelevant columns are dropped to streamline the analysis. An additional important note is that each row is a single game with the result of **home team**. This means that positive results indicate home wins and negative results indicate away team wins. This means that home-court advantage is baked into the model analysis.

To enhance the dataset, relevant columns from the "games" DataFrame are merged with the betting information, creating a consolidated DataFrame named "bets\_games." American odds are then converted to decimal odds, and the 'wl' column, representing game outcomes, is transformed into binary values (0 or 1) for ease of analysis. Exploration of player information follows, with the loading and filtering of player statistics from "nba\_players\_game\_stats.csv." The focus is on regular-season games occurring between 2007 and 2016. A DataFrame, "team\_game\_lineup," is constructed to represent player lineups in each game, highlighting the top six players based on playing time.

The subsequent step involves combining player and betting/game data. A new DataFrame, "merge\_df," is created, capturing player lineup information for each game. This information is then merged with the "bets\_games" DataFrame, resulting in an augmented dataset, "game\_df." Additionally, a new column, "season\_num," is introduced to represent the relative season number, aiding in the analysis of historical trends.

Finally, the dataset is split into training and testing sets using the scikit-learn library's train-test split functionality. The split is stratified by the season year to ensure an even distribution of data in both sets. This comprehensive data preprocessing pipeline sets the stage for subsequent exploratory data analysis and predictive modeling.

## Modeling Process

The first piece I wanted to explore was if a model could predict games just based on players in the game. To accomplish this, I came up with a creative solution based on Term-Frequency vectors to establish indicators when the most common players were playing in an NBA game. I created a dictionary of the 300 most frequently appearing players and then used that to indicate when one of those players was in the lineup. This took some manual trial and error to come up with the correct number of players to use.

When I started out using 1000 players, the model was overfitting to players that were only appearing in a few NBA games and over indicating on the results of their appearances. To evaluate this component, I printed the players with the highest coefficients in terms of win prediction to prove it lined up with mostly known all-star level players. These coefficients are across the players entire career regardless of season, so it tended to favor younger players in their prime. A simple logistic regression model of the player vector as features and the win or loss as the result to predict. The logistic regression model achieved an accuracy of about 60% win/loss prediction accuracy when test across test data.

|  |  |
| --- | --- |
| Player | Win Coefficient |
| LeBron James | 1.80 |
| Rudy Gobert | 1.24 |
| Chris Paul | 1.13 |
| Jimmy Butler | 1.12 |
| Chauncey Billups | 1.09 |
| Otto Porter Jr. | 1.07 |
| CJ McCollum | 1.05 |
| Jae Crowder | 1.05 |
| Paul George | 0.86 |
| Dirk Nowitzki | 0.86 |

The full PySpark script then intended to create a multi-category logistic regression model for predicting NBA game outcomes. The script utilizes PySpark's distributed computing capabilities to handle large-scale data. The main functionality involves reading and preprocessing NBA game data, transforming it into Spark DataFrames, and creating features for player lineups and game-related information.

The full model built off the findings from our player analysis and incorporated that vector strategy again. There were two key components added that improved performance though, adding the year the game was played and they opponent lineup. These were all combined via Milibs vector assembler to create the features set for model evaluation.

Two models were used off this dataset to predict the results of games an SVM model and a logistic regression model. Both models utilized the exact same training set and features to inference based on. They both resulted in very similar performance of about 65% of games predicted correctly.

The subsequent modeling approach involves applying a logistic regression model to raw feature values, combining player vectors and season information. Finally, by adding the logistic regression probability to the test output we can compare our models confidence in results to odds maker values and make smart betting decisions.

## Results

**SVM Model Win/Loss Prediction Results:**

--Contingency matrix--

TP: 698 FP: 331

FN: 240 TN: 386

----------------------

Accuracy = 0.655

F1 = 0.710

**Logistic Regression Model Win/Loss Prediction Results:**

--Contingency matrix--

TP: 693 FP: 326

FN: 245 TN: 391

----------------------

Accuracy = 0.655

F1 = 0.708

|  |  |  |
| --- | --- | --- |
|  | Profit By Season (units bet) | |
| **Season** | **Simple Predict Bet** | **Probability vs. Odds Betting** |
| 2007 | 6.23 | 0.85 |
| 2008 | 10.78 | 8.45 |
| 2009 | -7.90 | 1.77 |
| 2010 | 0.73 | 2.52 |
| 2011 | 20.67 | 4.87 |
| 2012 | -3.72 | -2.95 |
| 2013 | -1.92 | -0.59 |
| 2014 | -7.59 | 6.28 |
| 2015 | 1.34 | -7.14 |
| 2016 | 13.30 | 19.10 |
| **Total** | **31.92** | **33.16** |
| **Stan. Dev.** | **9.41** | **7.15** |

Simple Winner Prediction Profit by Season:

A graph of blue rectangular bars

Description automatically generated with medium confidence

Probability Based Model Profit by Season:

A graph of blue squares

Description automatically generated

## Conclusion:

The model was overall very successful at predicting games with limited features provided by just focusing the player tokens and the year played. Both the simple betting and the probability betting-based betting resulted in profiting over time. There was lots of variation from run to run depending on the train/test split, but they were almost always profitable in the end. The probability-based method was more consistent with less variation from season to season which would likely be desirable if someone was using their actual money.

Given the limited scope of this lab I was very delighted that this novel method for predicting NBA games showed excellent results and would be a great foundation for building a more advanced model to turn a profit over sports books. With additional time in this lab, I’d perform k-fold cross-validation on this data with multiple train-test split to fine tune the modeling and optimize for performance. Some additional features might be useful such as adding the records of the teams that season leading up to the game or the location of the game. An additional feature that is discussed with NBA analysis is the amount of rest a team has leading up to the game.