I first began with the idea of using NBA player data to see how stats correlate with position. Last year, I created a project in which I ran a KMeans algorithm on 2022-23 NBA player data to group players based on their stats. This wound up creating four distinct groups of players, with varying levels of insight into each group. Building off of this idea, I decided to create an end to end machine learning project in which I see how well various models can predict the listed positions of NBA players, this time for the 2023-24 season. I imagined it would not work very well, given the increasingly “positionless” nature of the modern NBA, in which players of all sizes and skill levels fill out widely varied roles. If it could, however, successfully predict players’ positions based just on basic box score statistics, this would appear to disprove the claim that the modern NBA is positionless.

I began by looking at my previous KMeans model. In this model, I cleaned up the data by removing duplicate players, first. Duplicates occurred because some players played for multiple teams in the season, so I kept only the overall stats for that player rather than stats by team. I then weaned the data down to only players who played at least half the games in the season. Then, finally, I changed the values of positions to the traditional listing of 1-5 for each (point guard = 1, center =5). Any players listed with multiple positions were assigned to the first position listed, since that is typically their most commonly played position.

I then visualized the data with histograms. Much of the data was not useful, such as age and games started. From there, I checked correlations of each statistic with position. The highest correlation was offensive rebounds, meaning players with lots of ORBs were more likely to play 5, and the lowest (negative) correlation was 3 point attempts, meaning players who shot many threes were more likely to play 1. Some non correlated variables included age and games started, as I would expect. Then, I created some new variables such as rebounds per minute and assist to turnover ratio, knowing these are usually good indicators of position. I again checked the correlation, and rebounds per minute was indeed highly correlative of a center (5) and high assist ratio correlated to a PG (1).

To ensure I did not overcorrect for minutes played and categorize players by raw counting stats, I split the testing and training sets by a stratified minutes category. I used 6 categories of minutes to ensure bench players saw as much representation as stars did. Then, I reduced the number of variables down to only those relevant to position and reduced redundancy, using 2P% and 3P% and dropping overall FG% (a combination of the two), for example. I scaled the data using min-max scaling given the difference in ranges for each.

I then created 4 models, similar to those used in homework 4- an AdaBoost Tree, Decision Tree, Random Forest Classifier, and a 3 layer neural network with a softmax output. To train the tree classifiers, I used RandomizedSearchCV instead of GridSearchCV in the interest of time.

As for results, all performed similarly, with the exception of the AdaBoost tree performing at about 40% accuracy as compared to over 50% for all the other models. This may have been due to poor hyperparameter tuning. If I were to retry, I would attempt a GridSearchCV or use a broader range for a RandomSearchCV to better tune the trees. However, these metrics of 50% are somewhat misleading because they do not account for the closeness of position. For example, the models would often predict 2s in place of 1s or 4s instead of 5s, which is a relatively small difference in position. Both 2s and 1s handle the ball and shoot threes, and both 4s and 5s shoot high 2P% and get many rebounds. If I accounted for this error, it is possible that the accuracy would read much higher. The neural network worked at a rate of slightly over 50% as well, but that may also be misleading NN model for the same reasons listed above. If I were to improve on this in the future, I would likely opt for a float value as the output of position rather than an integer. A float could help represent how close the predictor was to predicting the correct category.

All in all, I would argue that the models did a very good job on both the training and test data. They each output over 50% accuracy for both training and testing, indicating that it was not overfit and I do not believe it was underfit, either. Given the difficulty in actually assigning an NBA player a position, each model did terrifically well, especially accounting for similarities across nearby positions. This would seem to indicate that the NBA is not as positionless as coaches and media pundits have argued recently- positions are fairly well defined, even well enough for simple box score stats to be able to successfully predict them.

Resources

Sam Kenney. “KMeans on NBA Data”- <https://github.com/srken32/Python-KMeans-NBA-Project/blob/main/KMeans%20NBA%20Data.pdf>

Kaggle 2023-2024 NBA Player CSV download

<https://www.kaggle.com/datasets/vivovinco/2023-2024-nba-player-stats?resource=download>