Superteam

This project aimed to identify sets of players that collectively form strong basketball teams using machine learning and advanced statistics. Such a tool would allow general managers to make more informed decisions in building their rosters. By collecting a wide variety of individual player and collective team performance data from over 10000 NBA games dating back to 2015, the goal was to identify a mapping from teams' combined player performance data to the outcome of the corresponding game. This mapping then allows for building custom teams by aggregating individual player performances.

Assumptions

A team's performance can only be judged relative to another group, and it is crucial to include both competing teams' player performances when predicting the outcome of a game. Moreover, instead of simply mapping to a binary win-loss condition, I used the team's plus-minus score. This score is purely relative and allows the model to pick up on more detailed correlations suggesting why a set of players may have been better than another set of players.

Initially, I assumed that mapping from collective player performances to team performance and then mapping from team performance to the plus-minus results would yield the best results. However, it turned out that while second mapping had a very high accuracy, the formed did not. Moreover, the mapping from individual player performance to team performance in large parts is trivial. For example, the number of team assists would simply be the sum of all player's assists. Instead of using two models, I decided to simply use a single model which maps directly from a set of individual player performances for both teams to a plus-minus score.

A given player's performance data is simply averaged over a given period, such as the current season, and is then grouped with the average performances of other players.

Limitations

We require a fixed number of players on each team, and we provide various team-sized models ranging from one player to 13 players.

Data

As mentioned, the input data our model was trained on was a set of individual player performance statistics. The output data of our model was the associated games plus-minus result. To collect all this data, we used the python nba_api (https://github.com/swar/nba_api), which allows programmatic access to all statistics stored on NBA.com. We then collect more than 100 statistics (see models.py) for every player who played that game and the entire team's statistics and store them in a separate database. By stacking this player performance data into a single row of data for each game

and then mapping it to the plus-minus score of the corresponding team, we build up our training dataset. For the whole team model with 13 players per team, we then have 26*105=2730 features for each game where the first 1365 features correspond to the first team's player features and the latter half corresponds to the other team's features. The plus-minus score is then always given with respect to the first team. This way, we can double the size of our dataset by using each game twice and switching the order of the two groups and the sign of the plus-minus score, as this is symmetric under a switch of teams.

Model

We then train a regression model using the extreme gradient boosting library xgboost (https://github.com/dmlc/xgboost) and achieve remarkable accuracy of 98% in predicting which of the two teams won. Moreover, the root means square error in predicting the plus-minus score by which a team won or lost is approximately equal to 2, indicating that we can predict the number of points by which a team wins or loses with an error of roughly 2 points. As we decrease the number of players on each team, our model's rmse increases.

Now that we have trained our models, which can predict the outcome of a matchup between two sets of player performances, we calculate the average performance data for each active player throughout a season and use this as our input data. Now let's build some cool apps using our model.

Apps

First up, let's get all individual player performance data for the 2021-22 NBA season and calculate the average values for each player. As you can see from the dataset above, we have 542 different players with 105 average features each as our input data.

Matchup

Now we can use this data to simulate a game between two teams. Let's match up the Boston Celtics (BOS) and the Pheonix Suns (PHX) using a team size of 13 players per team.

Our model predicts that the Celtics will win by 4.73 points over the Suns when playing in Boston and that the Suns will lose by -0.43 points when playing in Pheonix. The winner of this matchup will therefore be the Boston Celtics.

We can also match up imaginary teams and see who wins. As an example, we can randomly sample two sets of 13 players.

Regular Season Standings

Now that we can simulate any matchup let's simulate all possible matchups in the NBA and find out who the best current teams are.

These win-loss results show that Boston is the best team in the NBA, winning against all other teams, while Utah and Pheonix are their strongest competition this year. On the other end, Detroit, Orlando, and the L.A. Lakers have the lowest win-to-loss ratios, with Detriot unable to win against any team. By comparing this with the actual standings of the current NBA season, we see a clear correlation indicating the model's predictive power.

Tournament

One of our primary goals is to identify strong teams by combining different players. Therefore I designed an elimination tournament system similar to the playoffs. Initially, teams are randomly sampled from the player pool, and the team that wins the tournament then goes on to another tournament where new teams are sampled. Eventually, a team will emerge which remains unbeaten.

We can then test any individual team by matching it up against many randomly sampled teams and recording the win-loss statistics. Let us take the last team to win the tournament above.

Super Team Finder

Another way to identify a strong team would be to find a team that beats another team and then continue until a given team cannot be beaten by any arbitrary team.

Next, we will build a trade finder that compares a team's performance to its performance if it makes a player trade. By iteratively doing this and keeping track of the win-loss statistics for each potential trade, the best trade is found by maximizing the win-loss ratio. However, to build a realistic trade finder, we first need to calculate an approximate trade value for each player, as some players are worth more than others, and trades need to remain fair for both parties.

Trade Value Calculation

We propose a straightforward calculation to rank all players in our player pool. Simply normalize all of our average data features overall players and then sum them for each player. Doing so, we get the following ranking.

Interestingly, 3 out of the top 5 players (Jokic, Giannis, and Embiid) in our ranking are considered contenders for this season's MVP Award. By plotting the distribution of scores of all players, we see an interesting trend illustrating how difficult it is to be at the very top of our ranking.

Trade Finder

We can use this ranking to build our trade finder by only allowing trades between players with similar rankings. Let's look at possible beneficial trades for the Charlotte Hornets by matching them against other NBA teams.

From the above trade, we can learn that while Jaylen Brown may be better than LaMelo Ball, LaMelo possibly has a much higher potential as he's still a very young player. This is something our trade finder can't measure as of now. One possible way of measuring potential would be to predict future player performance by looking at trends in past player performance.

We can also do the same by matching up the team against a set of randomly selected teams instead of actual NBA teams.

Building a Team around a given Player

Another possible application of our model would be to build a team around a given player. For example, let's create a team around Giannis Antetokounmpo by finding the team with the highest win-loss ratio. As of now have imposed no restrictions on the total trade value of the group of players (similar to a salary cap)

These are just some of the potential applications built using the models presented. Feel free to create more.

****Further Extensions****

These are just some of the potential applications built using the models presented. Feel free to create more. For instance, we could add salary caps to all teams using the trade value data we calculated to build realistic teams.

It would also be essential to understand why some teams are better than others. All that can be used as clues are the feature importance of the individual model.

As we can see, a player's plus-minus score and net rating are considered most important in predicting the overall team plus-minus score (here, the number suffix of the feature name indicates the player's importance in the team measure in terms of minutes played going from 1 to 13 for the home team and 14-26 for the away team).

If we sort our average season data by plus-minus scores, we see why our model may think the Boston Celtics will win this year's Playoffs and why Jayson Tatum is this year's MVP.

****Limitations****

Our model relies heavily on the individual plus-minus scores of individual players. It may be a limitation of our model as it would favor players currently playing on winning teams while unfairly penalizing players playing on losing teams. Nevertheless, keeping the individual

plus-minus scores of players as features makes sense as they provide valuable information about a player's performance.

****Conclusion****

We have to build a set of models helpful in predicting who wins a basketball game based on the individual player statistics of players on both teams. The models can then be used in various applications, for example, a trade finder that can suggest trades for a specific NBA team. Our analysis predicts that the Boston Celtics will win this year's playoffs.