NBA Draft Companion Application

Nick Cantalupa, Pazin Tarasansombat, Sean Duffy, Aditya Saxena
Northeastern University
August 12th, 2024

Release Notes

Project Introduction

For the second project of DS5500, our team aims to deliver a product that can allow NBA teams to optimize their roster and make the correct draft pick that will most improve the team's chance in the next NBA season. We modeled around the 2024 Draft Picks and NBA team's roster at the end of 2023-24 NBA season. We developed a model that can, given a team's roster, predict the success of the NBA team as well as make suggestions on the draft pick that will most optimize the team's performance in terms of the Win/Loss Percentage (W/L%).

Our project consists of two main models. The first model projects an incoming draft pick using their data from their time in college playing in the NCAA, or data from international leagues for incoming draft picks from outside the country. The model will take their existing data to make predictions on their performance in the NBA. The second model will consider the team's roster and the player statistics of each player in each position. The model will then make predictions on NBA teams' W/L% given the roster information. We then use the projected performance of the draft candidate to optimize the W/L% prediction by performing permutations on the team's roster.

We also implemented a supplementary model that calculates the similarity between draft candidates and historical All-star NBA players. We aim to give the user more insights on the potential draft candidate's role and development.

Web Application

The first section of the application contains the output of the win prediction model. This section allows you to select the team that you would like to be drafting for. Selecting the

team and clicking the ‘Enter’ button will produce a 10x3 table. This table will show the 10 players that are predicted by the NBA Season Win Prediction Model to give you the highest win percentage for the 2024-25 NBA season if they were to be drafted and placed into your lineup. The first column of the table will be the draft prospects name. The second column will be the predicted win percentage with that player in your lineup, the output of the Win Prediction Model. The third column will give you the player’s playstyle All-star comparison, the output of the Nearest Neighbor Player Comparison Algorithm.

The second section of the application, labeled “Available Players”, is a table of players that have declared for the 2024 NBA draft and are able to be drafted. This list is composed of both players coming out of college, as well as players that are entering the draft after having played in other International Leagues. The table displays the player’s name, position, as well as their college/international stats. These stats are their averages across all the seasons that they played in college or international leagues. These stats are their “per game” stats and therefore show the average of how many of each stat they log in each individual game. There is also a dropdown list that allows you to select any of the draftable players. Clicking the ‘Remove Drafted Player’ button will remove the selected player from the “Available Players” table. This will also remove the player from the Win Prediction Model in section one. This means after the player has been removed from the table, the player will not be included in the Win Prediction Model[1] the next time the ‘Enter’ button is clicked to run the model. This allows the app user to keep an updated list of “Best Available Players” in real time as the draft unfolds.

The final section of the application, labeled “Player Radar Plot”, lets you show a Radar Plot of any of the draftable players to better understand their strengths and weaknesses. This contains a dropdown list to select any of the draftable players. Clicking the ‘Plot Stats’ button will show the radar plot of the player's stats.

Future Considerations

For future improvement on our final product, we plan to include considerations on outside factors that can potentially affect NBA team roster construction. These include factors such as salary caps, player contract lengths, and potential trades. By including these considerations during the roster construction we can give more meaningful suggestions for roster construction, including recommending trades or free agents.

We also consider the prediction target itself. The model currently targets NBA team’s Win/Loss Percentage to predict and optimize. To improve the model further, we could consider factors such as the strength of schedule for each NBA team that can affect the

W/L% outside of the team's roster construction. Furthermore, we also consider incorporating point differentials for each team over the season to have more insight into the team's performance beyond the W/L%.

We also consider projection of the draft candidate's growth over their NBA career. By setting the draft candidate's predicted performance to a given year in their NBA career (such as rookie year vs 5th season), we can give a more detailed timeline for a team's roster construction and predict which draft candidate is more likely to make an immediate impact.

Finally, we are considering improvements to the UI design of the application itself, including better explanation and visualizations for player statistics.

Maintenance

The repository is set up in a way that automated every step possible to make it as easy as possible to re-run and get updated. Most of the data files are collected when build.py is run as these will run Selenium automation scripts to scrape data from the database. Please credit Sports Reference for their clean and flexible database. To access this data, the only barrier to entry is signing up for their free month trial or paying $10 for any additional months. You can enter your username and password into the environment to allow access to the data. The only manual aspect of getting this app set up is collecting the names of the draft prospects for the current year. These can be found at https://www.nba.com/draft/2024/prospects for each year. Finally, collecting data on international player stats is also manual as there is no singular database to pull from. Otherwise, the script can be re-run each year to produce updated data files.

Methodological Appendix

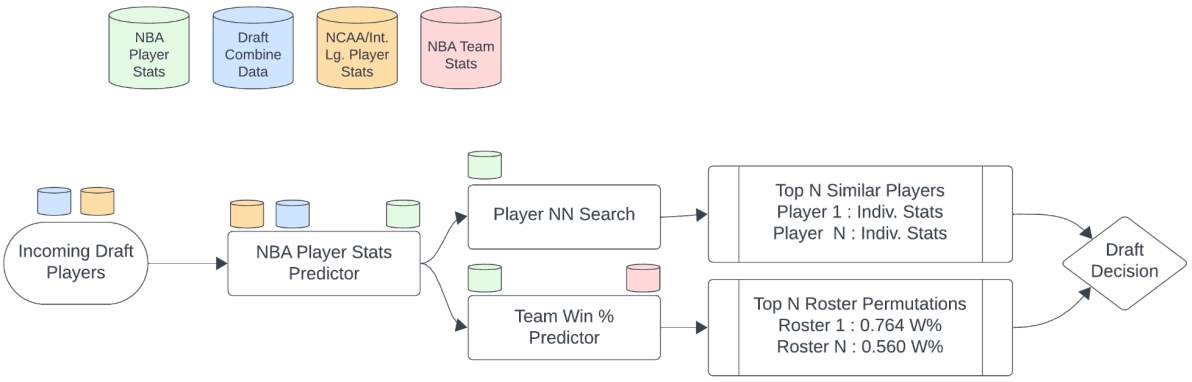

Figure 1. System Design

Data Collection and Processing

We collect the data for our project using two main sources of data. Our first data-source is the Stathead Basketball database, which has an extensive collection of basketball data and statistics. Our second main data-source is from NBA.com which contains data on players' physical characteristics such as weight and height. We used Selenium to automate the data collection process. For international players, the data is manually collected from data-source of their relevant leagues.

The main data that we used for our project is as follows

- Team Data
    - Team's performance for each season including Win/Loss Ratio
- Player Data
    - Position
    - Physical characteristics (heights, weights, strength)
    - Season statistics (e.g. shooting percentages)
    - Prospect college and international stats

The aim for our model is to be able to understand roster construction and make predictions on a team's Win/Loss percentage. To that end, outside of the routine data cleaning step, we process the NBA team and player data into a format that reflects our goal. We look at each team's roster as constructed of the five main NBA positions (Point Guard, Shooting Guard, Power Forward, Small Forward, and Center). Thus, for each team's roster during each NBA season, we aggregate the players statistics by their playing positions and minutes played. The data that we train our W/L% Prediction model thus consists of the aggregated player statistics for each player position on their respective team. To prevent issues from data leakage, we use a player's statistics from the previous season for the prediction of a given NBA season.

Model Development

The draft companion is built on two models: 1. a nearest neighbor model that is used to project a draft prospect's NBA stats using their college or international stats and find the All-star players they are most similar to and 2. A regression model to predict a teams win percentage based on their rosters previous season stats

Win Percentage Regression

This model predicts a teams win percentage based on an aggregation of 59 player stats for each of the 5 positions on the roster, as described above in 'Data Processing'. By fitting a model that can predict a team's success from their roster, a potential draft picks impact can be predicted by inserting that player into a team's roster.

We attempted to fit a linear regression model as well as a boosted trees model. Our evaluation criteria were mean squared error of the win ratio (between 0 and 1) and R^2 (correlation between our models predicted win percentages and actual win percentages. We sought to maximize the R^2 value.

With a simple linear regression, we were seeing significant overfitting. Test R^2 was -0.312 while train R^2 was 0.823. To combat this, we fit models with regularization. The best L1 regularization (Lasso) resulted in an R^2 of 0.521 with an α of 0.001. The best L2 regularization (ridge regression) resulted in an R^2 of 0.495 with an α of 10,000. These results make sense as there is collinearity expected in our data (players' skills tend to be captured in many different statistics. The lasso model shrank our predictors to 83 from 295, which supports this idea. This lasso is the model that was used in the final application. It provided the most accurate predictions (lowest MSE, highest R

Figure 1. Correlation plot of actual vs. predicted win percentage. Red line represents perfect prediction

Table 1. Top five features that affect win percentage (negatively and positively)

Most Important Features	
Positive	**Negative**
C Box +/-	PF Turnovers
C Off. Win Share	C Def. Win Share
PF Def. Box +/-	PG Personal Fouls
PG Off. Rating	SF Heaves Attempted
PG 2pt FGs Made	PG Def. Rating

We attempted to fit a gradient boosted trees model but even after a large grid search for optimal hyperparameters, a better R^2 value (>0.521) was not achieved. This grid search

used 3-fold cross validation and tested different values for 7 different hyper parameters (shown below). The best model achieved a test R^2 of 0.454. There was still significant overfitting in this model, as the train R^2 was 0.923.

Grid Search:

```
{'learning_rate': [0.01, 0.1, 0.2],
 'depth': [4, 6, 8, 10],
 'iterations': [100, 200, 500],
 'l2_leaf_reg': [1, 3, 5, 7],
 'border_count': [32, 64, 128],
 'bagging_temperature': [0, 1, 2, 3],
 'random_strength': [1, 2, 5]
}
```

Best Parameters:

```
{'bagging_temperature': 0,
 'border_count': 32,
 'depth': 4,
 'iterations': 100,
 'l2_leaf_reg': 5,
 'learning_rate': 0.2,
 'random_strength': 5
}
```

Nearest Neighbor Model

Projecting NBA stats for draft prospects is difficult as we are attempting to project 59 different stats (used in the win prediction model) from only 12 stats regularly collected and readily available from college and international play. For this reason, a nearest neighbor model was used. Data is available for almost all NBA players from when they played NCAA or international basketball. By finding the nearest players in the college/international stats space, we can identify the NBA players a prospect is most likely to play similarly to. For the sake of projection, the top 5 nearest players are selected, and their career NBA stats are averaged. This average serves as the prospects projected career stats.

The Nearest Neighbor Model is also used to find an NBA All-star player who has similar stats and playstyle to each draft prospect. These comparisons were created using the players per game averages for their college/international careers. These stats were then scaled using a Standard scaler to preserve the distributions of the data. One hot encoding was used for the players positions, giving them integer dummy variables for if they are a Guard (G), Forward (F), or a Center (C). The same stats were collected for all the NBA All-stars since the 2000-01 season. This data was scaled using a Standard Scaler as well, and then had the same positional dummy variables added. This ensures that the comparisons were being made using two datasets that have the same structure, making them consistent across the college/international and All-star datasets. A Python scikit-learn Nearest Neighbor model was then trained on the NBA All-star dataset, and then each draft prospect was fed into the model to find their nearest neighbor out of all the NBA All-stars. The resulting comparison gives the user a comparison of a player with a

similar playstyle based on their stats and gives the application user a way to visualize what each draft prospect could turn out to play like with professional success. This makes it easier to decide which prospect would be the best fit for your team. This comparison is displayed in the “All-star Comparison” table in the first section of the application.

Project Takeaways

As a team, we had learned several lessons as we worked on the project throughout the semester. The first lesson that we learned early in the project is the importance of having a clear vision on the end product that we would deliver, and the intended user of the product. Early in the project we struggled with finding the right approach to the project and we had performed some initial EDAs and analysis that, while providing interesting insight, were not used in the final design of the project. This includes attempting to define teams based on playstyle using clustering algorithms. Once we have a clear vision of the end product and use-case, we can come up with a concrete system design, and necessary tasks to complete the project.

We also learned a lesson in working as a team. Over the eight weeks that we worked together on the project, we have come to understand the individual strength and expertise of each member. This allows us to effectively divide up the tasks and assign them to each team member. By having a concrete system design and data pipeline, we are able to smoothly integrate each system together and make sure each member’s efforts contributed to the final product as a whole. We also learned to appreciate the importance of maintaining a clean and readable collaboration environment as we work to ensure smooth integration, and creating documentations for each individual contribution so that they are understandable for the remaining team members.

On the technical side, this project has allowed us to experiment with and test out different models and methods when it comes to making the prediction model. By experimenting with different approaches and models, such as various regression and tree-based models, we gained more familiarity with the strengths and weaknesses of each approach. This allows us to choose the model that is most suitable for the project and achieve the best possible result. The project also gives us an opportunity to implement and experiment with StreamLit to create a front-end application for our system and give us experience with designing and implementing user interfaces.

Finally, we have learned the importance of collaboration and having different viewpoints and perspectives on the project. The group working sessions we held throughout the semester gives us the opportunity to receive feedback early as we work on the projects. The feedback allows us to consider issues and solutions that we had not considered beforehand, such as issues with data leakage on same year data, and aggregating statistics using weighted average. The group sessions also allow us to identify design choices and decisions that may be unclear immediately and thus require in-depth and explicit explanations during delivery.