Data-X Project: NBA Make-A-Team
- Final Report

Submitted to:

IEOR 290 Data-x

Ikhlaq Sidhu

Team Members:

Vikram Singh

Karl Walter

Dian Yu

Yuanzhe Yang

Dong-Eun Suh

Project Pitch

NBA Make-A-Team gives the NBA coach, general manager, and everyday fan the opportunity to select hypothetical starting lineups and get accurate feedback on their team: projected winning percentage, statistical strengths and weaknesses, and a suggested player swap that can help aid the team in its weakest categories. This simple-to-use tool empowers the user to assemble their dream team – from any year 2000 to 2017 – and compare that team against their friends' squad-of-choice. And, for the unduly rational intent on making their "dream team" a "realistic team", there is a salary cap option which will impose the league's spending limits on your suggested player swaps.

Are you ready to Make-Your-Team?

Problem Statement

Our goal is to provide a team performance prediction tool for the user to accurately predict the performance of their own makeup team. By working through the three major steps of data processing, backend algorithms modeling and user interface design and implementation, our team was able to develop an application that can use statistics associated with five selected NBA players to predict the team's winning percentage, statistical strengths and weaknesses. The model can also suggest a player swap and re-predict the winning percentage again. Users only need to input five player's names into the user interface and the program will run the appropriate backend machine learning models to predict and display the winning percentage.

Approaches

Data Processing:

Data collection

Our data source is from https://www.basketball-reference.com/. Originally we planned to use BeautifulSoup4 to get data from website, but for some reason, it didn't work very well. When we found out Panda can also read the website URL, so we took off BeautifulSoup4 and replaced with pd.read.html to generate the data frame. Data reflecting NBA player performance between 2000-2017 was collected.

Data preprocessing

Pandas and Numpy were the primary packages leveraged. Extensive data cleaning was done to standardize numerical and categorical data over the 17 seasons. We applied concatenation to unite statistics lists from all seasons, and dropped off columns that

don’t affect too much on winning percentage, which was evaluated by simulating the features as we will discussed below. Lastly we connected players statistics to team statistics over seasons, and fixed the minor issue related to the change of team names.

Features Engineering
We initially leveraged simplistic player statistics accumulated on a per game basis, such as rebounds, assists, blocks, etc. Model performance based on the features was subpar. We then introduced aggregate metrics that reflect a player’s efficiency, such as assists per 48 minutes, and the model improved. Lastly we have introduced features intended to reflect team dynamics, such as whether the players’ skills are equal across the team or one player is far better than others. All features described were included in the final model.

Modeling:
Our modeling was divided into two parts, the first one was predicting Winning Percentage Model, and second part was to recommend a swap player in the make-up team for Winning Percentage improvement. Linear Regression was used to optimize for interpretability of the model.

User Interface:
We tried other UI software Pyform, Tkinter, of which first one cannot be successfully installed, unstable to run, and second one was simply hard to manage. Then we chose PyQt, which allowed us to design the interface by dragging and adding components, and program the function by converting it to py script. The general procedure of UI was to get inputs, check for validity of inputs, refer to dataset, connect model to calculate the winning percentage, and display the output back to interface. The most difficult part was to show the error message, we tried amount of methods to compare the string vector with dataset. Also, we originally planned to make pop-up window for error message, but couldn’t succeed.

Solutions

Back End: The names of five NBA players and the season in which they played are input into the system via the user interface. The corresponding player statistics for the season are fetched from the data repository stored within the app and these players statistics are aggregated and normalized based on the season average for five player team performance. The aggregate statistics are then fed into a linear regression model which predicts the team’s win percentage over the course of a season. As a general manager of a professional basketball team, getting an accurate prediction of team

performance can inform several roster and business decisions. For example, the manager may be required to increase marketing spend for upcoming season to sell seats if the team is not good enough to sell seats organically.

Once the winning percentage is made, a player recommendation for improving the team is made. This is done by assessing z-scores per team statistic relative to the season average. The player who most improves the aggregate z-score summed across all statistics is recommended and a new winning percentage is given. A team manager is constantly trying to improve the team but isn't necessarily able to evaluate potential new players empirically - this statistics driven player recommendation system can change that.

User Interface (UI): UI was programmed by PyQt5 from Python libraries. Firstly we applied UI generator to create a window interface with components like textbox, push button, and so on. Then we started programming to make it functional. The program extracted a string vector of names and a value of year, then it checked the name with processed database with for loop. If name was spelled wrong or blank space, the error message at the right-lower corner will show up to guide users to correct their inputs. After correction, clicking "play" push button will return the predicted winning percentage shown in the output textbox. On the right hand side, there are three options of checkbox to ask if users want to apply the salary cap, know the strength and weakness of make-up team, or swap the most ineffective player with someone else. If users checked one or two of them, hitting "play" will run the model again with the constraints.

Future Developments

Fantasy Basketball is a popular e-sport that requires participants to create their own basketball team using the basketball players currently playing professionally. Participants then face off against fellow players to see which team is better. Over the course of the season, participants add and remove players from their team to optimize performance. The 'NBA-Make-a-Team' app could eventually be leveraged within this context to predict team performance and recommend players to add and remove over the course of the season. There are several rule-based constraints that exist within Fantasy Basketball - these constraints need to be developed into the app in order for the app to be useful in this context.

Team Members Contributions

Vikram Singh	Data Collection, Preprocessing, Features Engineering, Modeling Development of Winning Percentage Model - Main Models Integration
Karl Walter	Development of Player Swap Model Continuously brainstorming on adding more potential functions
Dian Yu	Project Management: Created and updated gantt chart to track project progress. Arranged team meetings Data Preprocessing: Feature correlation and comparison User Interface Design using pyqt5 and qt designer
Francis	User Interface Exploration: tried several packages to decide which is the best tool for us to make the UI. User Interface Functioning: connected UI to model and displayed the outputs.
Dong Eun Suh	Make and integrated presentation template and designs. Recorded and documented ideas and progression to update details for everyone. When subject was material recycling, reached out some tech firm CEOs to get technical advices.