**Building prediction models and players analysis for NBA**

Author: Chun-Cheng Chang, Anthony Chu, Phyllis Chen

1. **Summary of Research Questions:**
   1. **How to efficiently visualize players’ shooting data:**

Typically, it is difficult to directly visualize and understand players’ shooting data because plotting an x-y coordinate plot abstractly in your head is very hard to formulate. Therefore, we found it beneficial to build a user-friendly program that can efficiently plot various individual player shooting data.

* 1. **How salary of each player affects each team’s performance:**

We believe that salary is a crucial indicator of a basketball team’s performance, so we searched for a method to affirm how salary can affect a team’s success. We accomplish this by building a model that predicts the winning rate when provided with necessary information for the prediction.

* 1. **Can we predict the winning rate of each team?:**

It is common for people to judge a team’s success based on their winning rate; this factor includes the season and play-off winning rate. By using player data such as points scored, rebounds, assists, all ranked within the league for enhanced comparison, we can predict the winning rate of each team.

1. **Motivation:**

My friends and I are fascinated by the NBA, on top of our long-lasting interest in playing the sport. We often debate over which team will take home the championship, which of us has the best bracket, and which of our top teams or players are most dependable. While many online reports and news sources display their own selective statistics, they typically prioritize publishing eye-catching news of the most talented all-star players. Thus, we transformed our passion towards basketball into constructing a program that can easily analyze and visualize player data to predict their future performances.

1. **Dataset:**

We focused on the following three datasets:

* 1. **Nba-api | An API Client package to access the APIs for NBA.com:** [**https://github.com/swar/nba\_api**](https://github.com/swar/nba_api)
  2. **NBA Player Salaries | HoopsHype:** <https://hoopshype.com/salaries/players/>
  3. **bttmly/nba: Node.js client for nba.com API endpoints (github.com):**

json file for teams:

<https://raw.githubusercontent.com/bttmly/nba/master/data/teams.json>

Column ‘teamName’ refer to the teams’ names.

json file for players

<https://raw.githubusercontent.com/bttmly/nba/master/data/players.json>

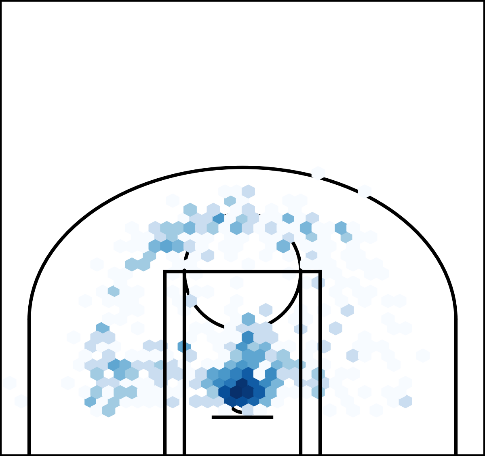
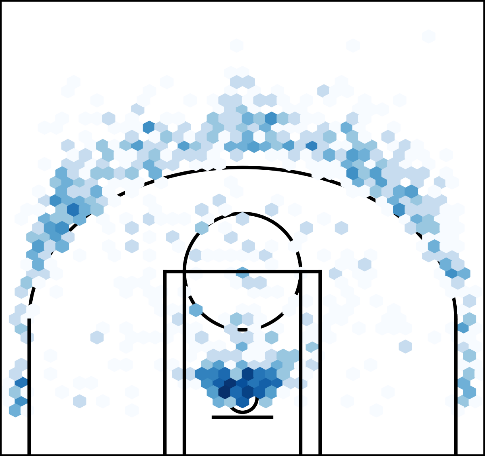
Column ‘firstName’, ‘lastName’ refer to the players’ names.

1. **Method:**
   1. **My environment:**

Using Anaconda to manage the package.

* + 1. Including NumPy, BeautifulSoup, pandas, matplotlib, requests, nba-api, SciPy, TensorFlow.
    2. Using git to manage version.
  1. **Data Preparation:**
     1. For each team’s past winning rate:
        1. Using endpoint LeagueDashTeamStats in nba\_api to access all teams’ record of winning percentage and other information.
        2. Using json and pandas modules to filter data until we get each team’s id, name, game played in a season, and winning rate.
        3. Save the filtered data into csv and category them by season.
     2. For each player’s data:
        1. Using endpoint LeagueDashPlayerStats in nba\_api to access each player’s data.
        2. Using json and pandas modules to clean the raw data.
        3. Save the prepared data into csv and category them by season.
     3. For each player’s salary:
        1. Using requests and BeautifulSoup modules to parse the HTML information on [NBA Player Salaries | HoopsHype](https://hoopshype.com/salaries/players/).
        2. Using pandas modules to transform dictionary-like data into csv file and category them by season.
  2. **Methods for Question A:**
     1. Use requests and json modules to obtain the data of each player and each team.
     2. Define a class of methods to quickly access data of player’s name, id and team’s name, id.
     3. Using endpoint ShotChartDetail in nba\_api to access target player’s shot location and the field goal made percentage.
     4. Generate easy-visualized map of players using matplotlib.
  3. **Methods for Question B:**
     1. Using all the data above and turn them to feature df and label df by pandas.
     2. Use keras.Sequential to build up the model.
     3. Save the best model based on the accuracy score of evaluating df.
  4. **Methods for Question C:**
     1. Using all the data above and turn them to feature df and label df by pandas and numpy.
     2. Use keras.Sequential to build up the model.
     3. Save the best model based on the accuracy score of evaluating df.

1. **Result**
   1. **Question A:**
      1. By running the plotting program, we can now easily recognize the precise position and location where the target player makes a shot.
      2. We shorten the process of accessing target player’s shot chart by only required user to input player’s name, target season, and team name.
      3. The plot on the left side is the field goals made chart for Stephen Curry, Golden State Warriors, 2021-22 season; the plot on right side is the field goals made chart for Tim Duncan, San Antonio Spurs, 2011-12 season. The darker color the mark is, the higher numbers of field goals the player made.



* 1. **Question B:**
     1. After training, the neural network model we built has approximately greater than 50 percent of accuracy on predicting the salary level of each player.
     2. Feature we used for training: season, player age, team winning rate, total games played in season, real winning rate, minutes per game, points per game, rebounds per game, assists per game, steals per game, blocks per game, turnover per game, field goal percentage, 3-point field goal percentage, total plus minus
     3. Salary level: level 0 – annual salary below 1200000; level 1 – annual salary below 2000000; level 2 – annual salary below 5000000; level 3 – annual salary below 10000000; level 4 – annual salary below 20000000; level 5 – annual salary above 20000000
  2. **Question C:**
     1. After training, the neural network model we built has approximately greater than 33 percent of accuracy on predicting the winning rate of each team.
     2. For this question, we will focus on how to use players’ data to predict the result, but not using the overall team data to do so. With that not all players in a team have great influence on the standing, we only consider the top 12 influential players in the team based on the total minute played data.
     3. Since the players’ data value have great gaps between each other, we choose to use the ranking of each feature within the league to train our model.

1. **Impact and Limitation**

The target audience of our project is people who are interested in using players’ data to predict the game trajectory of entire season. People can use the shot chart visualizer we built to acquire a better idea of how each player makes shots on the real court; moreover, the models we trained can help people get a rough idea about the trend of a team performance and the salary level a player might get.

However, there are still lots of limitations about the results we got. First, for shot chart visualizing, we are only able to make a 2D plot that show the position of each shot, but we are not able to find data that show the real shooting heights, which is another influential factor of this dataset. Basketball is a sport that requires a good analysis of 3D space, while we are only eligible to visualize 2D plane.

Additionally, there are too many factors that can influence the players’ salary and teams’ winning rate, which makes it difficult to choose the correct feature for training models for prediction. We cannot prove that salary rate has direct correlation on performance. Also, there are some factors that we will never be able to include in our models, such as players’ injury. Although we can still get 50 percent accuracy on salary prediction, we are not sure of how much each feature weighs in depicting the accuracy of our model and evaluating a player’s ability. For winning rate prediction, the tactics of the coach and the team culture adaptability of each player may have great influences on team performance, but these data can’t be valued and predicted only by evaluating each player’s data.

As for the practicality of the results we obtained, I would only recommend people to use our result as a rough trend of future predictions. Nevertheless, our data should not be used as a prediction machine for any commercial activity or gambling, while our data is not accurate enough to put in real application, but it may be a great factor to consider during prediction.

1. **Challenge Goal**
   1. **Multiple Datasets and Messy Data:**
      1. Using datasets from different sources to compile a more detailed result; we use the api offered by the official website of NBA and some other datasets that cover the limits of nba\_api such as salary.
      2. Filtering out some irrelavant data and combine different dataset’s information into appropriate form for faster and easier used in future.
   2. **Machine Learning:**
      1. Training models that allow people to predict each player’s salary by given information.
      2. Use players data to predict each teams’ winning rate.
   3. **New Library:**
      1. Using requests and BeautifulSoup module to get data from each website.
      2. Using TensorFlow to train models for salary and winning rate prediction.
   4. **Web Scraper:** 
      1. Learning the structure of HTML to obtain the correct data from website.
      2. Learning methods to obtain data from website, like BeautifulSoup module.
2. **Work Plan Evaluation**

We were fairly accurate in our proposed work plan estimates. They were close to reality because our group individually understood our levels of comfort in different topics and were able to properly adhere to the plan to come up with the final results that we aimed to receive while formulating the work plan estimate. We divided different tasks and served different purposes in making this project whole.

1. **Testing**
   1. **Question B:**

We randomly split the data into 80% for training and 20% for evaluating the accuracy. Also, we have checked this model for 1000 times (approximately 15 hours) by keep feeding randomly split data and find the averages of the result accuracy.

* 1. **Question C:**

We randomly split the data into 80% for training and 20% for evaluating the accuracy. Also, we have checked this model for 500 times (approximately 19 hours) by keep feeding randomly split data and average the result accuracy.

1. **Collaboration**