NBA Player Awards and Team Performance Prediction

GROUP 19

Zhan Shu
UNI: zs2584

Yuan Dou
UNI:yd2676

Yingqi Ma
UNI:ym2926

Abstract

In this paper, given the performance and award data of players and teams, we analyze the dataframe to get the features and stats. With the feature correlation for candidate players and teams, we build 5 machine learning models to predict the awards of players and playoff teams, and make comparisons between different ML models for different tasks. When making predictions, we update the latest data at 7 am everyday.

1. INTRODUCTION

The basketball competition is one of the most watched sports activities in the world. With the advancement of technology, basketball games and player statistics can be more easily stored and therefore can be used for prediction using artificial intelligence techniques such as machine learning. The National Basketball Association (NBA), as the most prestigious basketball league, has the best basketball players and largest fan base in the world. As of 2022 season, the NBA ended the regular season with over 66 million followers on Instagram, making it the most-followed professional sports account. [1] Every fan has a favorite player and team. So prediction about the game brings a lot of fun to fans and improves their viewing experience.

Using the players and teams performance data from *basketball-reference.com*, we are able to predict the player awards and whether the team can enter the playoffs. In this project, we process the data and find the feature correlation for candidate players and teams. Then use different machine learning methods(logistic regression, gaussian naive bayes, random forest, decision tree and k-nearest neighbors) to predict the all-star selections, awards voting results (Rookie of the Year, Sixth Man of the Year, Most Valuable Player, etc), and teams that can enter the playoffs of current year. After that, we make comparisons between different ML models for different tasks and get the best result. We also use DAGs and NBA API to update the latest dataset everyday. Therefore, the up-to-date prediction is our best prediction and update daily. Eventually, we implement the results to a web application we built using D3 and show the users clearly the predictions.

2. RELATED WORKS

In *Predicting NBA Games Using Neural Networks*[2], this paper applies a machine learning approach, Support Vector Machine (SVM), to predict the playoff results of the NBA, using the features composed of the historical statistics of regular seasons. Compared to our project, this paper only uses one machine learning model (SVM) to train the training data with cross validation to build up the classifier. This may lead to a higher error rate because of unpredictable factors.

In *A Novel Approach to Predicting the Results of NBA Matches*[3], this paper's approach uses both the team's own stats, but also the data known about the opposing team to make that prediction. And the findings show that utilizing the information about the opponent improves the error rate observed in the predictions. As opposed to our project that merely takes into account the player's or team's own performance and statistics in predicting the result, this paper gets a more accurate prediction in specific matches.

In *Predicting the Outcome of NBA Playoffs Based on the Maximum Entropy Principle*[4], the author claims that they use maximum entropy and k-mean clustering to build the training model. They also compare these two models with the other four models' performance. What we've done better than

them is we use deep learning models to train the data instead of basic machine learning models.

3. DATA

We first decide the datasets we want to use. We choose the datasets from Kaggle[7] and the datasets we used are player_totals.csv, player_award_shares.csv, and team_ summaries.csv, team_totals.csv files. Team summaries csv file is the more depth analyzed data of teams for each season. Team total csv file shows the total performance data of all NBA teams for each season.

It was not until June 1979 when the NBA adopted the three-point line for the 1979-80 season. Since then, NBA games have become more similar to modern games today despite some minor rule changes each year. Therefore, team data from 1980-2022 is used for training.

The data above are the data we use to train our model. When making predictions, we use the NBA API to update the latest data for the most accurate prediction. Therefore, all data from NBA API are selected this season. We first utilize the built-in function get_teams and get the team_id and team_name of all 30 NBA teams. Next, using teamyearbyyearstats function which takes in team_id and returns all seasons teams performance data, we select the latest year data and stack them together to a preferable shape we want. Same procedure for players' data update. We use the get_active_players function to acquire all the current NBA players name and player_id. After that, we use the playercareerstats function which inputs play_id and returns player career data. As we mentioned before, we select the season_id of 2022-23 to make a prediction. Again, we stack the rows of data for each player together and get the ideal dataset.

3.1. Exploratory Data Analysis

The purpose of implementing Exploratory Data Analysis (EDA) before we start building the learning model is we need a clear layout of our data. Imagine when facing a large amount of data, one would not directly select out the features that are important and may be confused by those statistics. That is when we need the EDA to clear things out and get the clean data before furthermore complex analysis. Exploratory Data Analysis is a vital process that does initial investigation on data to show features performance relationship, hypothesis correctness and assumption accuracy.

In the player total dataset, we have every player's total statistical data (including points, field goal percentage, assistant, rebounds, etc.) for each season from 1980 till 2022. Player award shows each year's awards candidate voting result and basic performance data. Team statistics shows the team statistical performance data for each year and indicate whether this is a playoff team.

1) Player award prediction data

When visualizing the datasets, we first need to preprocess the data into a preferable format. Therefore, we use the player_award_shares.csv file as the template and stack those candidates' statistical performance data to the template.

We want to plot out each year's comparison between candidates' statistical performance using a bar chart. As shown below is an example of the 2022 and 2021 DPOY award. The top figure 1 is the total blocks data of each DPOY candidate in 2022 season showing as y-axis. Figure 2 is the total blocks data of each DPOY candidate in 2021 season. Both figures have the leftmost candidate on the x-axis as the winner of DPOY. From the plot we can see that the winner of 2021 has especially higher blocks than other candidates, whereas the winner of 2022 has the least. But if we give more data from other years, we can see that block number is a high-related data with DPOY award.

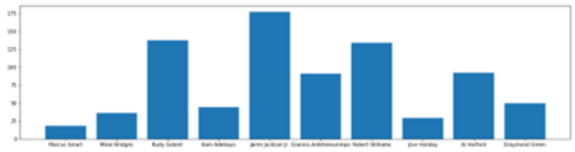


Figure 1. DPOY candidates block stats. 2022

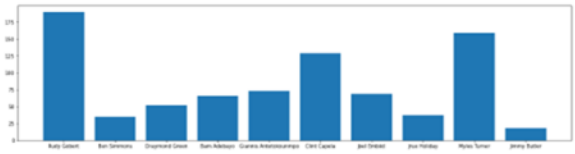


Figure 2. DPOY candidates block stats. 2021

2). Team statistics visualization

We then visualize data of all the playoff teams and non-playoff team performance data. Below are two examples that are typical. Figure 3 shows the points per game distribution of pf team and non-pf team. It is clear that the playoff team (orange, right) has higher values. Therefore, this is the kind of feature that we want to keep. Figure 4 shows the opposite way. Two teams have the same rebound per game. Therefore, we can consider dropping this feature for reducing complexity of data.

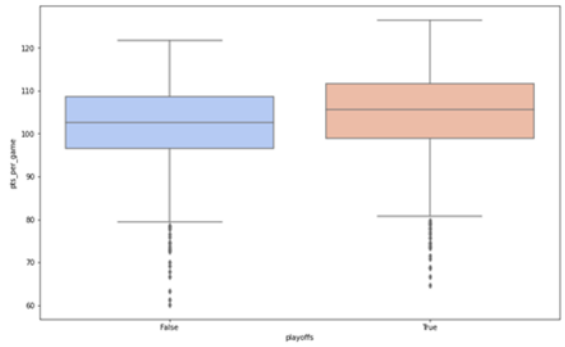


Figure 3. Playoff teams vs. non-pf team points per game

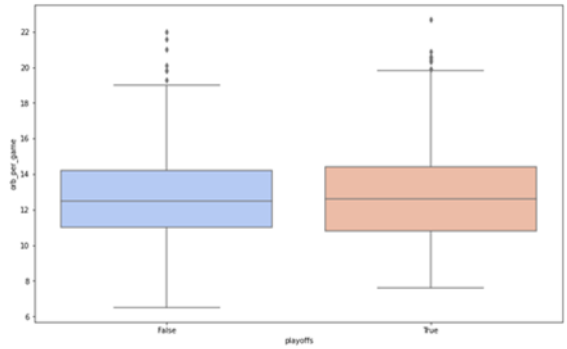


Figure 4. Playoff teams vs. non-pf team rebound per game

3.2. Data Preprocessing

The purpose of data preprocessing is to simplify data density and increase accessibility for our data shape. For instance, we want some input output of the same shape for stacking data and modeling them. Data preprocessing has four important steps: data quality check, data cleaning, data transformation and data reduction. After these processes, we would have the desired data that could enter the building model and training system.

First, we stack those two csv files together and check their features. We can see some feature columns have high correlation. Therefore, we drop them and keep the else feature performance data and get the desired dataset. We also observe the data have a row called League Average which shows the average statistical data of the whole season. We don't need this row showing each year, so we remove those rows as well for simplicity.

The following figure 5 and Figure 6 are the heatmaps we plotted with python matplot. It shows the correlation of each feature of teams and players and helps to make the dropout decision.

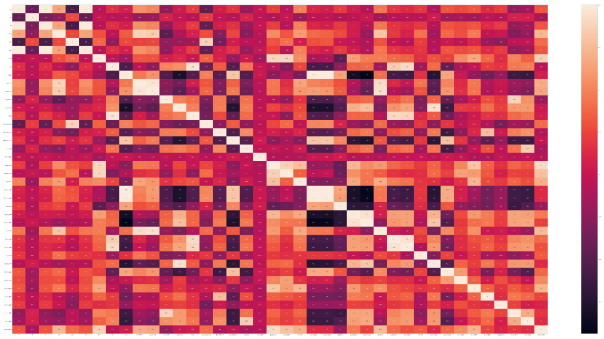


Figure 5. Heatmap of team statistic features

By observing the above heatmap, the following conclusions can be made:
“mov”, “srs” and “n_rtg” have a high positive correlation with each other.
“pace” has a high positive correlation with “fg_per_game” and “fga_per_game”.
“f_tr” has a high positive correlation with “fta_per_game” and “ft_fga”.
“x3p_ar” has a high negative correlation with “x2pa_per_game” and “x2p_per_game”, and a high positive correlation with “x3pa_per_game” and “x3p_per_game”“x3p_per_game”.
“ts_percent” has a high positive correlation with “x2p_percent” and “efg_percent”.
“x3p_percent” has a high positive correlation with “x3pa_percent”.
“ft_per_game” , “fta_per_game”, “ft_fga” have a high positive correlation with each other.
“fg_per_game” has a high positive correlation with “fga_per_game”.
Therefore, we decided to drop “mov”, “srs”, “pace”, “f_tr”, “x3p_ar”, “ts_percent”, “x3pa_percent”, “fga_per_game”, “ft_fga”, “fga_per_game”.

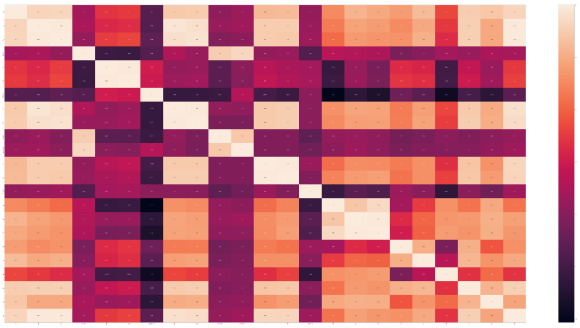


Figure 6. Heatmap of player statistic features

By observing the above heatmap, the following conclusions can be made:
“mp” is highly correlated with “fg”, “fga”, “tov”, “pf”, “pts”.
“fg” and “fga” are highly correlated.
“x2p” and “x2pa” are highly correlated.
“x3p” and “x3pa” are highly correlated.
“ft” and “fta” are highly correlated.
“drb” and “trb” are highly correlated.
“pts” is highly correlated with “mp”, “fg”, “fga”, “x2p”, “x2pa”, “ft”, “fta”, “tov”.
Therefore, we decided to drop “mp”, “fga”, “x2pa”, “x3pa”, “fta”, “trb”, “pts”.

4. METHODS

4.1 Comparison of Machine Learning Models

There are a total of 5 prediction tasks, playoffs prediction, MVP prediction, DPOY prediction, ROY prediction and SMOY prediction. We would like to determine which classification model performs best on each task. For each task, we split the data into 70% training data and 30% validation data, then train 6 models, logistic regression, gaussian naive bayes, random forest, multi-layer perceptron, k-nearest neighbors and support vector classifier. Then we evaluate each model using the validation accuracy, which is calculated by the number of correct predictions / size of the validation data. By comparing the validation accuracy, we are able to select the best model for each task. The details of the experiment process and results will be covered in the next section.

4.2 DAG

In order to provide the most accurate and most up to date prediction results, we have built two DAGs which are triggered at 7 a.m. We used nba_api to get the latest team statistics and player statistics.

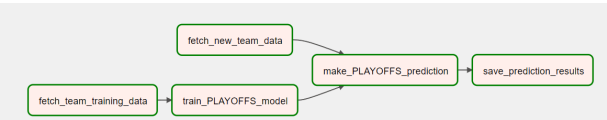


Figure 7: DAG for playoffs prediction

Figure 7 shows the overall DAG structure of playoffs prediction. In the task “fetch_new_team_data”, we first use the function get_teams() to get the id and team names of all current NBA teams. Then

we utilize another class provided by nba_api, teamyearbyyearstats, which takes a team id and returns all the team statistics of each year. We take only the team statistics of the year 2022-23, stack the data from all 30 NBA teams and return the data frame as the testing data. Another version of the data frame with less features is also saved to a csv file for visualization in our web application. In the task "fetch_team_training_ data", we select team statistics of season 1980-2022 from the dataset we downloaded from kaggle, and return the data frame as the training data. In the task, "train_PLAYOFFS_model", we train a logistic regression model using the training data and return the model. In the task "make_PLAYOFFS_ prediction", we use the logistic regression model which we have trained in the previous task and make predictions based on the testing data. Instead of a binary output, we utilized predict_proba of the scikit_learn library to output the probability of each team getting into the playoffs. Finally, we save our predicted probabilities and their corresponding team names as a csv file in the final task.

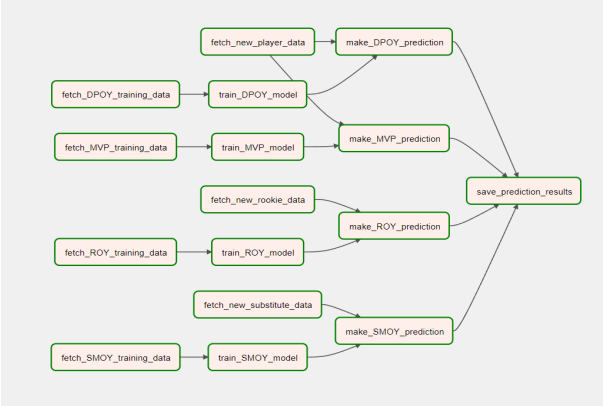


Figure 8. DAG for player awards prediction

Figure 8 shows the overall structure of various player awards predictions. In each of the four fetch training data tasks, we select only players who are candidates of the award each year. We add an additional column representing the true labels of whether the player has won the award that year or not. We set the winner of each award to true, and labels of other players to false, and return the data frame as training data for each prediction task. In the task "fetch_new_player_data", we first use the function get_active_players() to get the ids and names of all the players who are playing in the current season. Then we utilize another class provided by nba_api, playercareerstats, which takes a player id and returns the player statistics of each year. For each player, we take player data with season_id = 2022-23, stack the data together, and return the data frame as testing data for MVP model and DPOY model. Additionally, we saved a copy of data with less features for visualization of our web application. As for the task "fetch_new_ rookie data" and "fetch_new_substitute_data", they follow the same process except that for rookie data, we pick only the players who have less or equal to 1 year of experience in the league, and for substitute player data, we pick only players who have played less than half of the games as starters. For MVP model, DPOY model and ROY model, we train logistic regression models using the corresponding training dataset, while for SMOY model, we train a random forest model with a setting of max_depth = 10. In each of the four make predictions task, we use the model trained in the previous task and the corresponding testing data to predict the probability of each player winning a certain award. In the final task, we stack all the player names and four arrays of prediction results. For players who are not rookies or substitute players, we fill the entries of p(ROY) or p(SMOY) with 0.

4.3 Web Application

We have developed a web application for better visualization of the predicted results as well as displaying the most up to date team and player statistics. Details of how the web application is built and instructions of how to use the web application will be covered in the System Overview section.

5. EXPERIMENTS

5.1. Team Performance Prediction

For the team performance prediction task, we would like to predict the probability of each team entering the playoffs based on its current performance. We have conducted two sets of experiments using different datasets.

The first experiment we did uses a dataset containing only the performance of the team in all the games they played in the past seasons. Some of the

features in the dataset are total number of field goals, assists, rebounds, and average number of points, turnovers, steals. This is a low level analysis of the team performance, which is easy to obtain.

Since the features are not on the same scale, we first scale each feature by its standard deviation, so each scaled feature has a mean of zero, using the equation:

$$X_{scaled} = \frac{x - \mu(x)}{\sigma(x)}$$

In order to determine which machine learning algorithm performs better on the dataset, the training data is split into 70% training data and 30% validation data. Then the data is used to train 5 classification models, logistic regression, gaussian naive bayes, random forest classifier, multi-layer perceptron, k-nearest neighbors and support vector classifier. After training, the models are evaluated using the validation data by number of correct predictions / size of the validation data.

Table 1: Experiment 1 results

Classifiers	Validation Accuracy
Logistic Regression	0.79944
Gaussian Naive Bayes	0.64903
MLP	0.80501
Random Forest	0.73537
KNN	0.70195
SVC	0.78552

From the above result, we can see that logistic regression and multi-layer perceptrons achieved a relatively higher accuracy compared to other classifiers. However, the highest accuracy, achieved by multi-layer perceptrons, is only 80.5%, which is relatively low if we want to make an accurate prediction.

The second experiment we did uses a dataset containing high level analysis of the team performance. Other than basic statistics like field goal percentage, average points per game, etc, the dataset also contains the features of their average opponent performance when played against the team, as well as the strength of future schedule.

Similarly, we standardized the data and split the dataset into training data and validation data. Then we trained six classification models and compared their validation accuracy. The results are shown in Table 2.

Table 2: Experiment 2 results

Classifiers	Validation Accuracy
Logistic Regression	0.92119
Gaussian Naive Bayes	0.91576
MLP	0.88043
Random Forest	0.91848
KNN	0.87228
SVC	0.89130

From the above results, we can see that logistic regression achieved the highest accuracy, which is 92.1%. Compared to the results we obtained in the first experiment, we can notice that all the classifiers achieved a higher validation accuracy in the second experiment. Thus, we can conclude that with high level analysis of the team performance, the models perform better and we are able to get a more accurate prediction.

Below is the predicted probability of the logistic regression model with input of the latest team statistics.

Table 3. Predicted probability of each team entering the playoffs

team	playoffs
Atlanta Hawks	0.918265
Boston Celtics	0.992032
Brooklyn Nets	0.871761
Chicago Bulls	0.534361
Charlotte Hornets	0.591622
Cleveland Cavaliers	0.999653
Dallas Mavericks	0.773474
Denver Nuggets	0.911193
Detroit Pistons	0.119427
Golden State Warriors	0.734514
Houston Rockets	0.21314
Indiana Pacers	0.503872
Los Angeles Clippers	0.708626
Los Angeles Lakers	0.851453
Memphis Grizzlies	0.72601
Miami Heat	0.325904
Milwaukee Bucks	0.997425
Minnesota Timberwolves	0.540314
New Orleans Pelicans	0.999323
New York Knicks	0.498536
Oklahoma City Thunder	0.499137
Orlando Magic	0.245366
Philadelphia 76ers	0.964326
Phoenix Suns	0.999703
Portland Trail Blazers	0.149002
Sacramento Kings	0.88145
San Antonio Spurs	0.166523
Toronto Raptors	0.883631
Utah Jazz	0.549811
Washington Wizards	0.581965

5.2. Player Awards Prediction

For player awards prediction, there are a total of four tasks, MVP prediction, DPOY prediction, ROY prediction, SMOY prediction. For each task, we filter the data by choosing the players who are candidates of the award each year. Then we add an additional column representing the true labels. The winner of the award each year is set to true, while the rest are set to false. Then we pick the player data of season 1980-2022. In terms of NaN values in the dataset, we replace them with 0 instead of dropping them since these NaN values are caused by no such performance of the data throughout the season rather than missing values.

To evaluate the performance of each classifier on each task, we first standardized the data and split the data into 70% training data and 30% validation data. Then we trained the six classification models on each task using the training data and compared the validation accuracy. The results are shown in Table 4.

Table 4: validation accuracy of different types of classifiers on each task.

	Logistic Regression	Gaussian Naive Bayes	RandomForest	MLP	KNN	SVC
MVP	0.93333	0.88888	0.93016	0.92063	0.92381	0.93333
DPOY	0.91477	0.85227	0.90909	0.91477	0.90909	0.91477
ROY	0.87368	0.84211	0.83158	0.86316	0.78947	0.81053
SMOY	0.92258	0.86452	0.94839	0.93548	0.94194	0.93548

From the above table, we can see that the logistic regression model achieved the highest accuracy on the task of predicting MVP, DPOY and ROY, while the random forest classifier achieved the highest accuracy on the task of predicting SMOY.

6. SYSTEM OVERVIEW

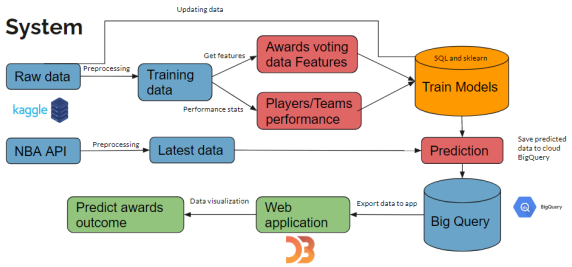


Figure 9: System Diagram

The above figure 9 shows our team system architecture. Our system is based on Python. We first get data from Kaggle and update the data from the NBA API when making predictions. Next, we preprocess the data and get the training data. After doing EDA with the python pandas package, we get the correlations between features and get the preferable dataset.

After getting the dataset, the training model was built using the scikit-learn library. This model

system takes the training data of teams' performance and players' statistics to predict the awards of the end of the season and the percentage of a team that could enter playoffs. We then save the output predictions to GCP Big Query for the next visualization and application system.

The end port for users is the web application we built using D3 and html. Users can easily overview the prediction result of four different prediction tasks. The web app also visualizes the data in table form which could be sorted in customized criteria.

6.1 Web Application

We use jquery.min.js to import the jQuery JavaScript library so that we can simplify the implementation of our web application. First, we use $('#demo').html to implement a sample web.

With the 4 csv files stored in the BigQuery, we use d3.csv to read the csv files into an array for each of the csv files. Then we push the title name of the csv file to the array named columnsDats. After that, we push the values of the csv file to the array named dataSet. Now with two arrays of title and content of the table, we use $('#example').dataTable to visualize the table on the web.

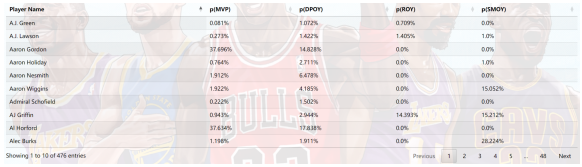


Figure 10: View of the Table

After we finish the first html file, we just repeat the steps for the rest 3 csv files. And now we get our 4 task html. To display all of them on the index, we create 4 href links on the index, each corresponding to one task. In this way, we can go to our desired web page of task from every html webpage.

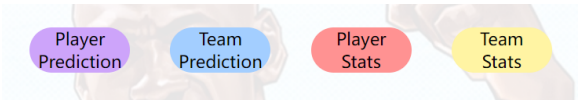


Figure 11: Four links of task

We also add a dropdown menu so that the users can change how many records of the table they want to see on one page, from 10~100.

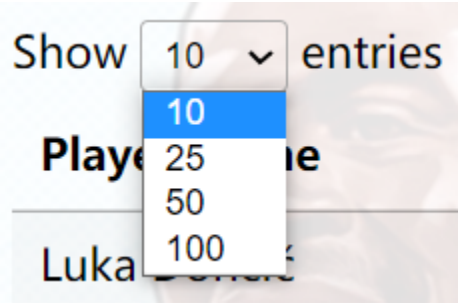
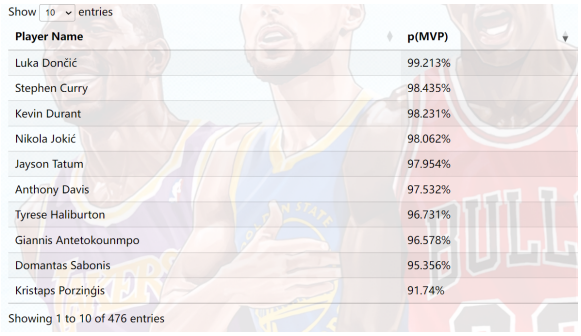


Figure 12: Dropdown Menu

Also, we create a search bar on the top right of the table, so that users can search their favorite player or team.

Figure 13: Search Bar

Furthermore, we add a sort button next to each column name. This can let the users sort the table by the column values. This will help users to sort the table according to statistics they are most interested in. For example, if a user wants to check who is most likely to get the MVP award this year, he can just click on the column of p(MVP). And the table will show the top 10 players with the highest possibility to win MVP. And in the figure below, we can see that Luka Dončić is the top contender to win MVP award this year.



Show 10 entries

Player Name	p(MVP)
Luka Dončić	99.213%
Stephen Curry	98.435%
Kevin Durant	98.231%
Nikola Jokić	98.062%
Jayson Tatum	97.954%
Anthony Davis	97.532%
Tyrese Haliburton	96.731%
Giannis Antetokounmpo	96.578%
Domantas Sabonis	95.356%
Kristaps Porziņģis	91.74%

Showing 1 to 10 of 476 entries

Figure 14: Table sorted by p(MVP)

Finally, we use style.css to beautify our web application by formalizing the format of table and head. Also we add a background image to make it more attractive. So below is the overview of our web application.

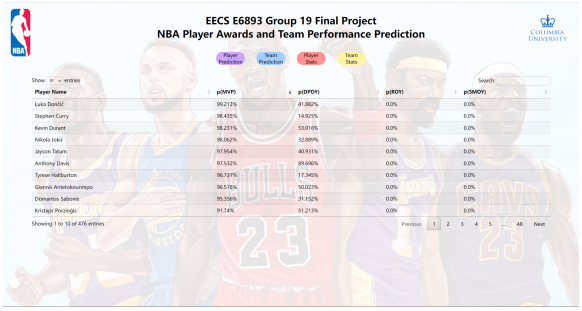


Figure 15: Overview of Web Application

6.2 Instructions on running the application

- open the demo folder
- download html folder and open with vscode
- right click on any html files and select Open with Live Server

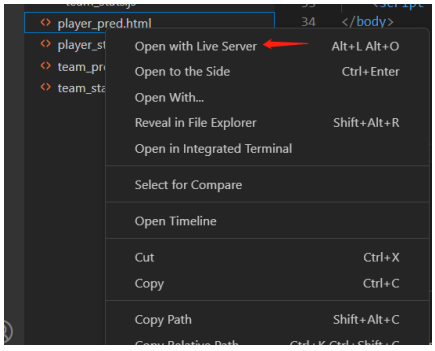


Figure 16: Instruction for Opening Web Application

Detailed instructions and descriptions can be found on the homepage of our github repository.

6.3 Bottlenecks in this project

Throughout the project, we discover a potential bottleneck about the model. When updating the team performance prediction, the dataset of NBA API does not include the factor of opponent strength. However, when we do the experiment, we take the opponent's performance as a factor. Therefore, when implementing DAG, we did not use the opponent performance factor. In the future, this could be an interesting aspect for improvement.

7. CONCLUSION

In this project, we implement a learning model that takes input of our preprocessed data and outputs four predictions of MVP, DPOY, ROY, and SMOY for player awards and playoff percentage for teams statistics. Finally, we successfully built a web application to show the predicted results. Our result is predicted as our expectation and the model performs with high accuracy. Therefore, Our model is an ideal model for the prediction.

However, there are some unforeseen problems in this project for example some off-court issues and player's injury. These are the factors that doing pure data processing are not eligible to take them into account.

Yet, we did discover some problems during the project. Initially, we did the comparison of the winner of each award with all the other players' performance. However, each year there is only one winner, and all the labels of the other players are False. This will make the dataset with lots of False labels and very few True labels. Although the accuracy is very high, we did not see this as an ideal model since all the predictions of correctness are those predictions of false. That is, the data is highly biased. So instead of comparing the winner with the whole dataset of other players, we select the top 10 candidates of the award that season and compare with the winner. Although this dataset is still a little bit biased, it much better represents the relation between the winner and similar players.

8. REFERENCE

[1]https://theathletic.com/3500551/2022/04/13/nba-average-viewership-up-19-for-2021-22-vs-last-season-across-abc-tnt-espn/
[2]Loeffelholz, B., Bednar, E. & Bauer, K. (2009). Predicting NBA Games Using Neural Networks. *Journal of Quantitative Analysis in Sports*, *5*(1). https://doi.org/10.2202/1559-0410.1156
[3]Jackie B. Yang and Ching-Heng Lu .(2012). PREDICTING NBA CHAMPIONSHIP BY LEARNING FROM HISTORY DATA.
[4]Cheng, Ge & Zhang, Zhenyu & Kyebambe, Moses & Nasser, Kimbugwe. (2016). Predicting the Outcome of NBA Playoffs Based on the Maximum Entropy Principle. https://www.researchgate.net/publication/312236952_Predicting_the_Outcome_of_NBA_Playoffs_Based_on_the_Maximum_Entropy_Principle
[5]Basketball Reference https://www.basketball-reference.com/about/glossary.html
[6]NBA Glossary https://www.nba.com/stats/help/glossary
[7]Kaggle.com https://www.kaggle.com/datasets/sumitrodatta/nba-aba-baa-stats?select=Team+Summaries.csv

9. APPENDIX

9.1 Contributions

Zhan Shu - 40%

Main contribution: Implement comparison of ML models, DAGs

Dou Yuan - 30%

Main contribution: Design web application

Yingqi Ma - 30%

Main contribution: EDA, Data Preprocessing

9.2 Other Materials

Definition of features of team statistics:
sos- Strength Of Schedule, which represents a team's average schedule difficulty faced by each team in the games that it's played so far or for all season. [5]
o_rtg - Offensive Rating, representing points scored per 100 possessions. [5]
d_rtgDefensive Rating, representing points allowed per 100 possessions. [5]
n_rtg - Net Rating, measuring a team's point differential per 100 possessions. [6]
e_fg_percent - Effective Field Goal Percentage, measuring field goal percentage adjusting for made 3-point field goals being 1.5 times more valuable than made 2-point field goals. [6]
tov_percent - Turnover
orb_percent - Offensive Rebounding Percentage, representing the percentage of available offensive rebounds a player or team obtains while on the floor. [6]
opp_e_fg_percent - Opponent Effective Field Goal Percentage, representing an opponent's field goal percentage that is adjusted for made 3 pointers being 1.5 times more valuable than a 2 point shot. [6]
opp_tov_percent - Opponent Turnover Percentage, representing the number of turnovers an opponent averages per 100 of their own possessions. [6]
opp_drb_percent - Opponent Defensive Rebounds, representing the percentage of opponent's number of defensive rebounds collected. [6]
opp_ft_fga - Opponent free throws per field-goal attempt
fg_per_game - Field Goal per Game
fg_percent - Field Goal Percentage
x3p_per_game - 3 Point Field Goals per Game
x3p_percent - Percent of 3 Point Field Goals Attempted. Representing the percentage of a team's 3 point field goals attempted. [6]
x2p_per_game - 2 Point Field Goals per Game
x2p_percent - Percent of Field Goals Attempted (2 Pointers). Representing the percentage of field goals attempted by a team that are 2 pointers. [6]
ft_per_game - Free throw per Game
ft_percent - Percent of Team's Free Throws
orb_per_game - Offensive Rebounding per Game
drb_per_game - Defensive Rebounding per Game
trb_per_game - Total Rebounding per Game
ast_per_game - Assists per Game
stl_per_game - Steal per Game
blk_per_game - Block per Game
tov_per_game - Turnover per Game
pf_per_game - Personal Fouls per Game
pts_per_game - Points per Game

Definition of features of player statistics:
fg - Field Goal.
fg_percent - Field Goal Percentage. Representing the percentage of field goal attempts that a player makes.[6]
x3p - 3 Point Field Goals
x3p_percent - Percent of 3 Point Field Goals Attempted. Representing the percentage of a team's 3 point field goals attempted. [6]
x2p - 2 Point Field Goals
x2p_percent - Percent of Field Goals Attempted (2 Pointers). Representing the percentage of field goals attempted by a team that are 2 pointers
e_fg_percent - Effective Field Goal Percentage, measuring field goal percentage adjusting for made 3-point field goals being 1.5 times more valuable than made 2-point field goals. [6]
ft - Free throw
ft_percent - Percent of Team's Free Throws Attempted. Representing the percentage of a team's free throws attempted. [6]
orb - Offensive Rebounding
drb - Defensive Rebounding
ast - Assists
stl - Steal
blk - Block
tov - Turnover
pf - personal fouls