MET CS777 – Project Assignment

Term Project Report

Brandon Bartol

1. **Introduction**

Both classification and regression models are powerful tools within data science. Classification models allow us to informed decisions by predicting different classes that data may fall into. Regression involves the prediction of real values rather than classes where data belongs. Regression allows the user to predict the numeric values. When combined with an application like “PySpark” which allows for parallel processing of large datasets, powerful insights can be found.

The primary question I would like to answer is “what makes a good shot in basketball?”. This model will be based on both shot location and the type of shot taken. From this model I hope to gain insight what makes for a successful shot attempt and what attempts should be avoided. To help I identify this I have split my dataset into shots taken by All-Stars and non-All-Stars. Splitting out the best players allows me to see if there is a difference in the type of shot they take that leads to their increased performance or if the difference comes from shot making skill. To help analyze this I am hoping to employ both a classification model in the form of logistic regression as well as a regression model in the form of linear regression.

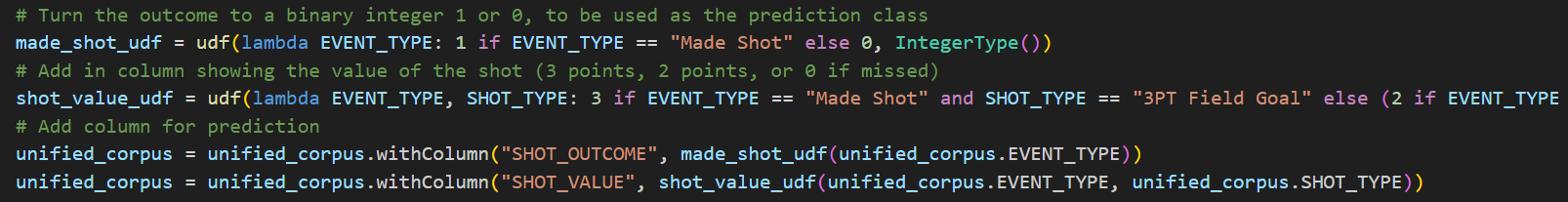
Following the creation of these models their performance will be tested and analyzed. For the classification model performance metrics, such as accuracy, precision, recall, F1-score will give me a holistic understanding of my model’s performance as well as strengths and weaknesses. For the linear regression model the coefficients that accompany the predictive features will be important in determining which areas are the best locations to score points. By comparing non-All-Stars to All-Stars, we can see if it is the shot selection that leads to a difference in the player performance or if the All-Stars are more efficient shooting similar shots.

In conclusion, this project endeavors to both predict if shots are made as well as output an expected point value for each shot. By comparing 2 different populations of players, I hope to see if the better performing players are shooting in different locations or are just more accurate taking the same shots.

1. **Data Import and Preprocessing**

Before building a data model the initial dataset must first be understood. To build my dataset I used the “NBA Shots” dataset on Kaggle ([⛹🏾‍♂️ NBA Shots (kaggle.com)](https://www.kaggle.com/datasets/mexwell/nba-shots/data)) and selected the 5 most recent years (2020-2024). This 5 year stretch was selected these because over a longer time period changes to the game such as rule changes may have an impact on the success of shots. These 5 datasets were combined through a union creating a “unified\_corpus” consisting of 1,031,742 total shots.

The next step taken is modifying my corpus to include 2 new columns that are used as the label columns for my models. The first was a “SHOT\_OUTCOME” column that returns a 1 if the shot was made and a 0 if the shot was missed. The second is a “SHOT\_VALUE” column this shows the overall value of the shot and is 0 if the shot was missed, 2 if the shot was made, and 3 if the shot was made from behind the 3 point line.



After creating my initial dataset, the preprocessing could begin. The dataset is evenly distributed with 482,499 made shots and 549,243 missed shots. Because of this even distribution the data was not weighted or resamples before creating models.

The dataset itself is then reviewed to identify columns that will be valuable to my models:

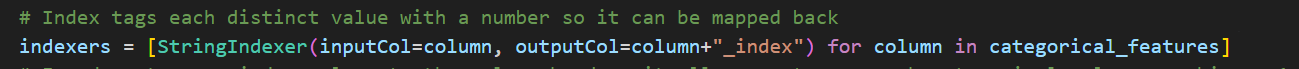
|  |  |  |
| --- | --- | --- |
| **Column Name** | **Description** | **Encoding/Handling Strategy** |
| SEASON\_1 | Season indicator variables | Dropped – Out of Players control |
| SEASON\_2 | Season indicator variables | Dropped – Out of Players control |
| TEAM\_ID | NBA's unique ID variable of that specific team | Dropped – Out of Players control |
| PLAYER\_ID | NBA's unique ID variable of that specific player | Use to split into All-Stars and non-All\_Stars |
| GAME\_DATE | Date of the game (M-D-Y // Month-Date-Year) | Dropped – Out of Players control |
| GAME\_ID | NBA's unique ID variable of that specific game | Dropped – Out of Players control |
| EVENT\_TYPE | Character variable denoting a shot outcome | Changed to “SHOT\_OUTCOME” |
| SHOT\_MADE | True/False variable denoting a shot outcome | Changed to “SHOT\_OUTCOME” |
| ACTION\_TYPE | Description of shot type (layup, dunk, etc.) | Use as categorical predictor |
| SHOT\_TYPE | Type of shot (2PT or 3PT) | Used to create “SHOT\_VALUE” |
| BASIC\_ZONE | Name of the court zone the shot took place in | Dropped as this highly correlates with the ZONE\_NAME |
| ZONE\_NAME | Side of court name | Use as a categorical predictor |
| ZONE\_ABB | Abbreviation of the side of court | Redundant with “ZONE\_NAME” |
| ZONE\_RANGE | Distance range of shot by zones | Use as a categorical predictor |
| LOC\_X | X coordinate of the shot | Dropped – “ZONE\_NAME” covers this |
| LOC\_Y | Y coordinate of the shot | Dropped – “ZONE\_NAME” covers this |
| SHOT\_DISTANCE | Distance of the shot | Dropped – “ZONE\_RANGE” covers this |
| QUARTER | Quarter of the game | Dropped – Out of Players control |
| MINS\_LEFT | Minutes remaining in the quarter | Dropped – Out of Players control |
| SECS\_LEFT | Seconds remaining in minute of the quarter | Dropped – Out of Players control |

From this review 3 columns containing categorical data were identified as containing information that the player can control about the shot taken. These 3 fields used as predictors can be summarized as follows:

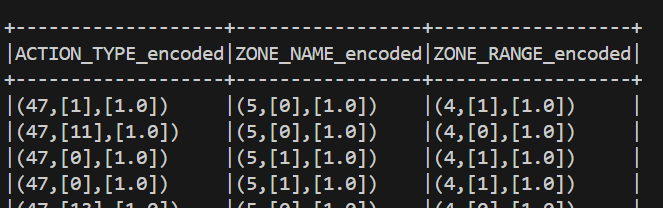
* ACTION\_TYPE– There are 48 different categorical descriptions that describe the shot type such as 'Jump Shot', 'Pullup Jump shot', and 'Driving Layup Shot'
* ZONE\_NAME – There are 6 different zones that a player can shoot from: ['Center', 'Left Side Center', 'Right Side Center', 'Left Side', 'Right Side', 'Back Court']
* ZONE\_RANGE – There are 5 different ranges a player can shoot from: ['Less Than 8 ft.', '24+ ft.', '8-16 ft.', '16-24 ft.', 'Back Court Shot']

These are all categorical fields (non numeric, non-ordinal) and in the Regression models these will need to be transformed into numeric values to uses. To do this each of these will be transformed into their own column so that a unique coefficient can be created for each feature. To do this 3 steps were performed.

The first step in establishing my features is to create an index for each set of categorical predictors. This means for a like “ACTION\_TYPE” a unique index number is assigned to each value; for the values "Jump Shot", "Layup", etc., it will assign categories numerical values (0, 1, 2, etc.). This is output into 3 new columns that have “\_index” in their name. This prepares the data for the one-hot encoding step.

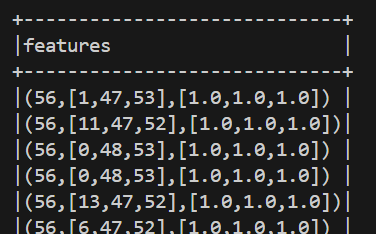


The second step is the one-hot encoder (OneHotEncoder in mlib). This works by taking the index values and transforming them into their own binary column. Continuing with the index example "Jump Shot" becomes [1, 0, 0, …], "Layup" becomes [0, 1, 0, …], etc. This allows for each value to be treated as a separate feature in the model. In my dataset the output is written into columns with “\_encoded” in the name. The data appears as a set of 3 values where the first is the index size of each column, the second is the index number that had a value, and the 3rd is the value of that index feature (it will always be 1 because it is a binary result).



With the completion of the encoder the features are ready to be assembled.

The final step of the preprocessing pipeline is the assembler. The assembler uses the “VectorAssembler” function from “mlib” and creates a “features” output column. This output column is a single matrix consisting of the 3 previous encoded matrices that were assembled before. The first value is the total size (56) meaning there are 57 unique categorical values that will be used in the model. The second set of 3 values are the index locations of the value that was present in the row. There will always be 3 values as they come from the ACTION\_TYPE, ZONE\_NAME, and ZONE\_RANGE columns. The final set will always be 3 “1.0”s as these represent the binary value of the categorical field.

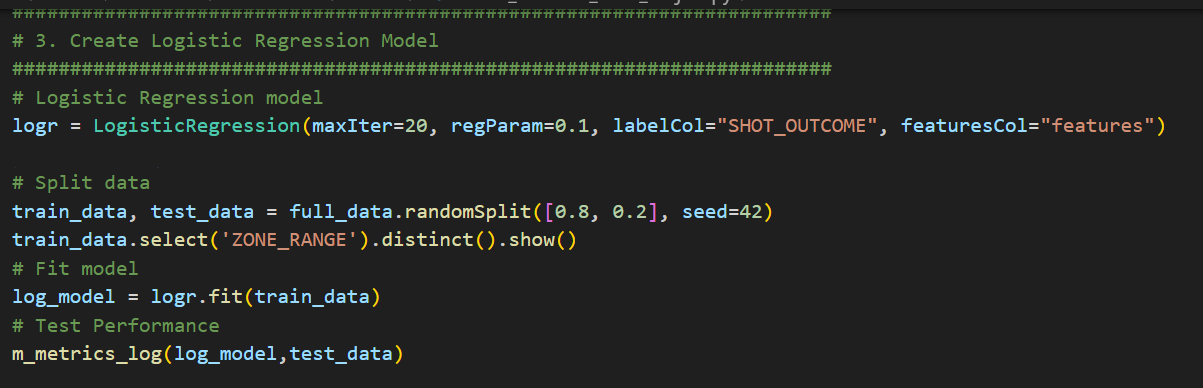


With the features now in a numeric matrix form the data is ready to be input into my models.

1. **Model Building**
   1. Logistic Regression

The first model utilized in this project is the logistic regression algorithm, more specifically “LogisticRegression” in the “pyspark.ml.regression” package. The parameters chosen for this were a maximum of 20 iterations to run, with a regularization parameter of 0.1. The features column created in preprocessing was used to classify the “SHOT\_OUTCOME” column that was also created during preprocessing.

The input dataset was split into 80% training and 20% testing data and the logistic regression model was fit to the training dataset. The model was then tested on the test dataset where it predicted if the shot was made using the information from the features column regarding the type of shot and where it was taken from. The performance will be discussed in the “Model Performance” section of this paper.

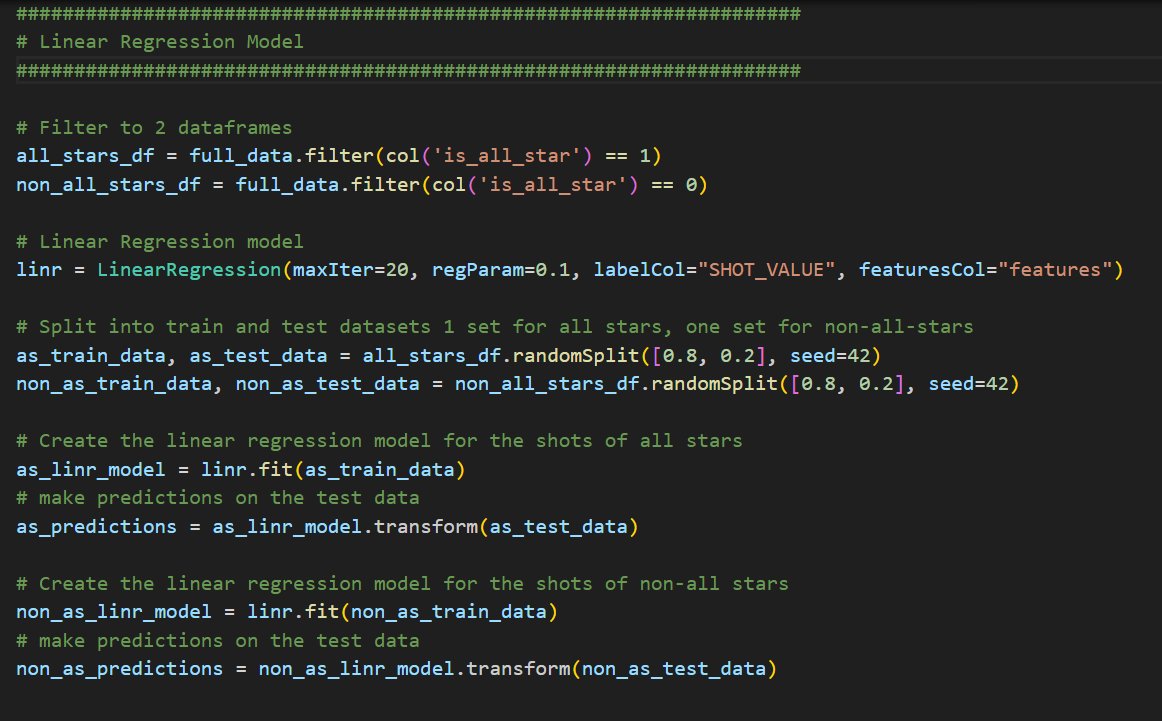


* 1. Linear Regression

The next model used is linear regression as provided by the “LinearRegression” function in the “pyspark.ml.regression” package. Like the logistic regression model, the parameters chosen for this were a maximum of 20 iterations with a regularization parameter of 0.1. The features column was again used but this time to predict the value of the “SHOT\_VALUE” column that was also created during preprocessing.

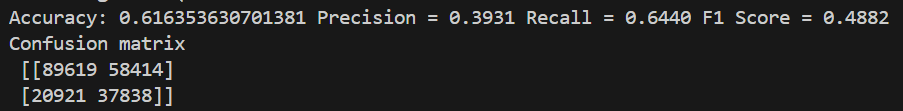
Unlike the previous model where the full dataset was split into test and train data, we the dataset is first split into shot data from All-Stars and shot data from non-All-Stars. Each of these subpopulations of data was then split into 80% train data 20% test data, producing a total 4 datasets: as\_train\_data, as\_test\_data, non\_as\_train\_data, non\_as\_test\_data.

The linear regression model is then run on both the as\_train\_data and the non\_as\_test\_data. This creates data models with an intercept and a coefficient value for each of the features. To test the performance these were run against the test data sets and the results of these models will be further discussed in the next section. The model was run using a parameter grid in order to find the optimal parameters and a regularization parameter of 0.1 was found to be best.



1. **Model Performance and Interpretation**
   1. Logistic Model

The logistic regression model can be summarized by the following performance metrics and confusion matrix.



The logistic regression model sought to predict if a shot was made or not and for this it performed poorly. While its accuracy of 61.6% is higher a random guess would be, as 46.8% of all shots in the dataset were made, this is not particularly strong accuracy and suffers from a few errors. The low precision shows it predicts a made shot only 39.3% of the meaning it suffers from false positives and is bias towards assuming a shot was made. The recall show it was able to correctly predict 64.4% of the made shot attempts, but as the f-score shows it sacrifices precision from this increase in recall.

These results lead me to reevaluate my model and its purpose. There is a fundamental uncertainty in if a shot will go in and because of this it seems unlikely through the features I selected alone that made shots will be predicted at a high accuracy. Also using the binary outcome does not account for that fact that shots are worth different values. Trying to correctly predict a made shot can be misleading if it is not indicated that the made shot is worth more. To try to account for this I hoped a linear regression model could better show the expected number of points.

* 1. Linear Regression

To test the performance of the Linear Regression model values for the root mean squared error (RMSE) and R^2 were calculated. The RMSE shows the average difference between the predicted and actual value for my model and the R^2 shows what percentage of the variance in the shot data my model explains. Unfortunately even after parameter tuning and cross validation I was only able to achieve the following metrics:

All Star Root Mean Squared Error (RMSE) = 1.137889521954311

All Star R2 = 0.0379838837198786

non-All Star Root Mean Squared Error (RMSE) = 1.176578956345913

non-All Star R2 = 0.0379838837198786

This means that for both the population of All-Stars and non-All-Stars the average prediction was 1.1 points away which is very bad for a prediction that should be between 0 and 3. The R^2 value of .04 is also concerning as that means only 4% of the variation can be explained by my model. This means I am missing major fields within my model. These will be further discussed in the conclusion.

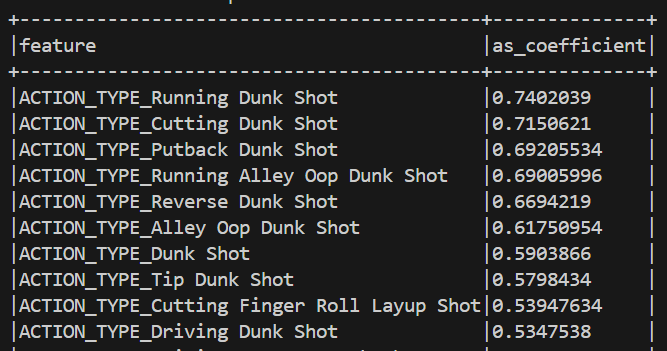
Although my performance is concerning, I went ahead with viewing the model intercept and coefficient values that are found with my linear regression model.

All-Star Intercept: 1.0267403313354737

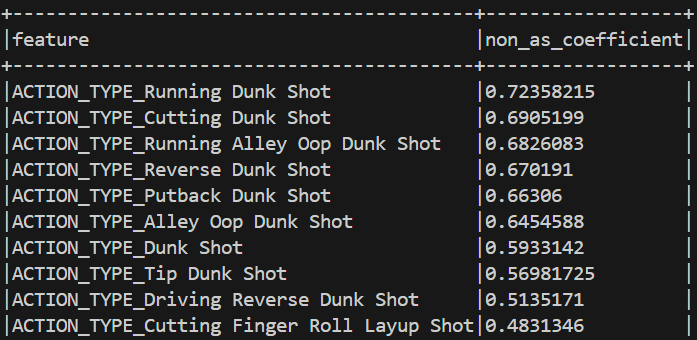
non All-Star Intercept: 1.0004058862652132

I was encouraged to find nearly identical intercepts for the models of the two populations that appears to show that before the features are applied the underlying expected points is very similar but slightly higher for All-Stars.

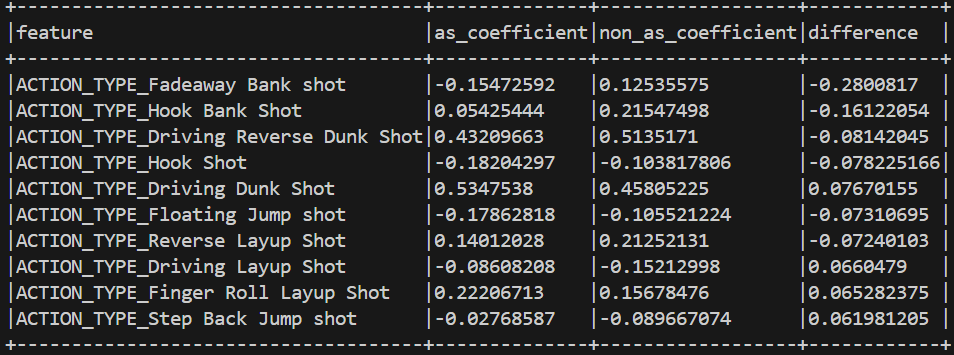
The below features resulted in the 10 highest increased in predicted points for All-Stars:



This summary was then repeated for non-All-Stars:



This summary shows how shots like dunks and layups resulted in the highest increase in predicted points. The values being higher for All-Stars also lends to the idea that the they have a higher conversion rate for turning these shots into points, but this is not always the case. The specific features for All-Stars and non-All-Stars was compared and the following are the 10 features with the highest differences between the two populations:

Surprisingly, this shows that the shot that produced the largest difference in predicted points was a Fadeaway Bank Shot. This shot produced an expected 0.125 points for non-All-Stars while the same feature cause a decrease in the expected points of an All-Star by 0.15. Overall the All-Stars did seem to be better at producing points as the coefficient values of the features were higher for 36 of the 57 features.

From these metrics I found that the linear regression model does not do a good job of predicting the point value of an individual shot taken. This is because there is always a chance a shot misses. Take a 3 point attempt for example, a great shooter makes about 40% of these attempts. This means the expected point value for this shot for a great shooter is 1.2 points. If my model predicts a similar number then over the course of the season the total points will be similar however the RMSE will be large because the shot will either be worth 0 or 3 points meaning if I am predicting 3 points then my error will be large. When I was creating this model I did not consider how have discrete rather than a continuous actual result will affect my prediction. When I look at the total points predicted by my model compared to the total actual points scored then I see that it was accurate.

The total points predicted by my model were only 0.7% different than the total actual point value for All-Stars. For non-All-Stars the point value was even closer with a 0.08% difference between the predicted total points and the actual total points scored in my test dataset. This shows how because the output values in my prediction were continuous rather than the discrete value that real life models, it appeared that the models were inaccurate but they were very accurate in predicting the total number of points scored.

1. **Conclusion**
   1. Summary of key findings and insights from the project

I found that the logistic regression model was able to predict if a shot was made with an accuracy of around 61%, but suffered from both type 1 and type 2 error producing an F-score of only 0.49. Because of this inability to reliably classify a made and missed shot I found that the model was not very successful at predicting shots.

The linear regression model also proved to struggle with predicting the points value of each shot taken with a very large RMSE of over 1 for both the All-Star Population and non-All-Star population. However, linear regression was very good at predicting the expected point value for each shot. When the data was aggregated to show the total amount of points scored in the test data versus the predicted number of points the values were within 1% of each other meaning while it could not accurately predict an individual shot’s points it was able to find the expected value of the shot very well.

With my linear regression model, I was able to see some systemic differences in the expected value of shots taken by All-Stars versus non-All-Stars. The coefficient values of the features were higher for the All-Stars model in 36 of the 57 features. This supports the idea that in general they are better at making the same shots than non-All-Stars.

* 1. Suggestions for future research or improvements in methodology

Based on my models to evaluate on a shot-by-shot level additional features would be necessary to make the prediction accurate. I tried to predict based off the types of shots and where they are on the court however this does not get the whole picture for if a shot will go in. On major factor I overlooked was defense. There is a defender trying to stop the player from scoring if I had addition features describing how close the defenders are then this could lead to a better model. In addition, there is a level of randomness to taking a shot in basketball, not all shots taken by a player are the same. To predict if the shot will go in the hoop, I would likely need information on the ball itself such as speed and launch angle. These addition features would help to account for the fact that not every open shot and even open shots are not all taken the same way.

In terms of the models used, I used a linear regression model because I wanted to predict an integer point value. After performing this project, I would likely try a different model like a decision tree or random forest because my resulting value is discrete. Using one of these models may do a better job of classifying with either a 0, 2, or 3 value however they would still suffer from the issue of randomness where a shot that appears the same based on my features can have different outcomes. The only way to get over this would be to add in the proposed features to my model.

* 1. Closing remarks

I was able to not able to successfully predict the value of a shot taken but was able to accurately find the expected value of the shot taken. There seemed to be evidence that All-Stars were more efficient when shooting the same shot but there was not a clear difference in the selection of shots found between the two populations. This project had varied success but could be improved with additional predictive features and using models like a decision tree or random forest that can output discrete values.

1. **VII. References**

* Class Notes
* library(pyspark.ml)
* library(pandas)
* library(pyspark.sql)
* library(pyspark.mllib)
* [⛹🏾‍♂️ NBA Shots (kaggle.com)](https://www.kaggle.com/datasets/mexwell/nba-shots/data?select=NBA_2024_Shots.csv)
* Wikipedia (for All Star lists: [2024 NBA All-Star Game - Wikipedia](https://en.wikipedia.org/wiki/2024_NBA_All-Star_Game), <https://en.wikipedia.org/wiki/2023_NBA_All-Star_Game>, <https://en.wikipedia.org/wiki/2022_NBA_All-Star_Game>, <https://en.wikipedia.org/wiki/2021_NBA_All-Star_Game>, https://en.wikipedia.org/wiki/2020\_NBA\_All-Star\_Game)