NBA Shot Prediction

Justin Hsiang and Adhitya Logan

I. Abstract

The NBA is a multi-billion dollar industry where the greatest financial rewards are given to the most successful teams. An efficient offense would help teams find success. In our project we sought to answer the question of whether or not we could predict if a player's shot will go in considering multiple statistics. Using our findings, teams could potentially reverse-engineer both offensive and defensive strategies to optimize their performances. The data we have chosen is a representation of the modern NBA and the recent rise of the 3-point shot in the past decade. We chose a season that had not yet been affected by COVID-19 to get as accurate results as we could.

The features we ended up choosing to predict if a shot would go in or not were the following: period, time. home/away, distance from the hoop and the average shooting percentage of the type of shot being taken. We chose a logistic regression with the distance from the hoop as our benchmark model while using all features in a Logistic Regression, Linear Support Vector Classification and a Random Forest for the rest of our modeling. We ended up choosing the Linear Support Vector Classification as our champion model as it predicted the most true positives for made shots. With an accuracy of 60.29% it is lower than the Logistic Regression and the Random Forest Model; however, we decided to prioritize the F1 score as the data has a lot more missed shots than made shots and predicting a true positive made shot is more important to us. Overall, our champion model seems to be mildly successful in predicting whether a shot will be made, but there is still room for improvement to optimize our findings.

II. Data and Methods

Background of Data

Shot data from the 2018-2019 NBA season was obtained from the NBA website using the nba_api Python package, which uses the NBA's API to gather officially measured statistics from NBA games. The 2018-2019 season was chosen for analysis because it is the most recent season to not be affected by the COVID-19 pandemic. This data was collected in the form of raw JSON, which we cleaned and exported into a Pandas DataFrame. Next, we converted this DataFrame to a Spark DataFrame using spark.createDataFrame(). The raw data contained 24 variables and 219,458 observations. Because the data was most easily transferable through a DataFrame, the majority of our project was conducted with the ML package. RDD's and the MLlib package was used to calculate the summary statistics using MulticlassMetrics() and BinaryClassificationMetrics().

Data Preprocessing/Cleaning

In order to construct machine learning models using the ML library, our data needed to be processed into a digestible format. Our data started off with these features:

GRID_TYPE	GAME_ID	GAME_EVENT_ID	PLAYER_ID	PLAYER_NAME	TEAM_ID
TEAM_NAME	PERIOD	MINUTES_REMAINING	SECONDS_REMAINING	EVENT_TYPE	ACTION_TYPE
SHOT_TYPE	SHOT_ZONE_BASIC	SHOT_ZONE_AREA	SHOT_ZONE_RANGE	SHOT_DISTANCE	LOC_X
LOC_Y	SHOT_ATTEMPTED_FLAG	SHOT_MADE_FLAG	GAME_DATE	HTM	VTM

First, we needed to quantify all features. To do this, all numeric columns were converted to float or numeric types using the cast() function to ensure that no errors would occur later in the construction process. To begin with, we needed to change **HTM** and **TEAM_NAME** to be the same format, as **HTM** would give the abbreviated team name while **TEAM_NAME** would give the full name. To do this we mapped **HTM** to **TEAM_NAME** and changed the variables to the full name of the team and with withColumn() we created a column called **HOME** that gave us which team the player taking the shot was on. With this we ran withColumn() and when() again to give us values of 1 if the player was at home and 0 if otherwise. Next we created a column using withColumn() called **DISTANCE_TRUE** that gave the euclidean distance from the hoop given the **LOC_X** and **LOC_Y** values. We then wanted to convert **ACTION_TYPE** to the percentage that the certain shot type would go in and to do this did a groupby() on **ACTION_TYPE** and agg() on the mean while renaming the column **SHOT_PERC**. With a join() to our main data frame we now had **SHOT_PERC** for each action type. The last feature we engineered was the time remaining in a quarter where we simply created a **TIME** column using withColumn() and converted **MINUTE_REMAINING** and **SECONDS_REMAINING** into a single time in seconds. And lastly using select, we chose our features to be the following:

PERIOD: whichever quarter the shot was taken in(including overtime periods)

TIME: the remaining time in the shots respective quarter

HOME: 1 for playing at home and 0 for playing at away

DISTANCE_TRUE: calculated the hypotenuse from the x and y values of where the shot was taken on the court to get the true distance as a float

SHOT_PERC: percentage values for each ACTION_TYPE seeing how likely they were to go in

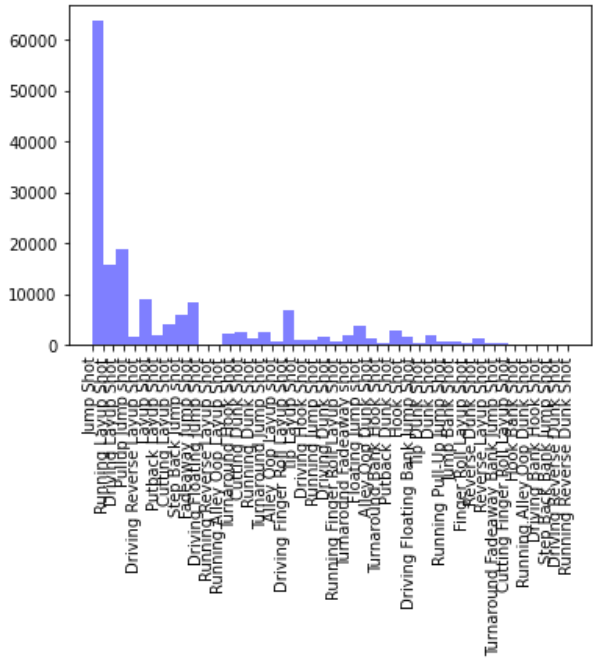
Our response variable was defined as **SHOT_MADE**. After we had selected all of these variables to keep for our modeling we dropped all null values to avoid any errors in classification. The number of observations after dropping came out to be 170,560 meaning we lost 48,898 columns to null values. In cleaning our data, we chose variables that we thought would be relevant in a prediction to if the shot was made or not. The columns we ended up removing were irrelevant such as the player's name or the team name.

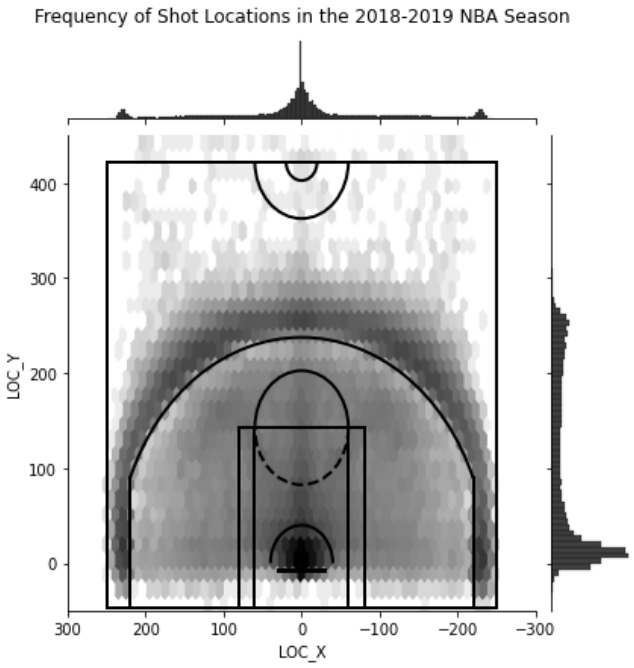
Exploratory Data Analysis

Understanding the Shot Distribution

Once we found the predictor variables that we had decided on we wanted to see how heavily they influence our response in SHOT_MADE. First, we wanted to see a distribution of the type of shots being taken in the league to see the most common of them all. For this we made a histogram of all the different types of shots and how often they are taken just to examine the data we were working with and how it would affect our classification.

From this table we can see that jump shots are by far the most taken shot amongst the league by a large margin.

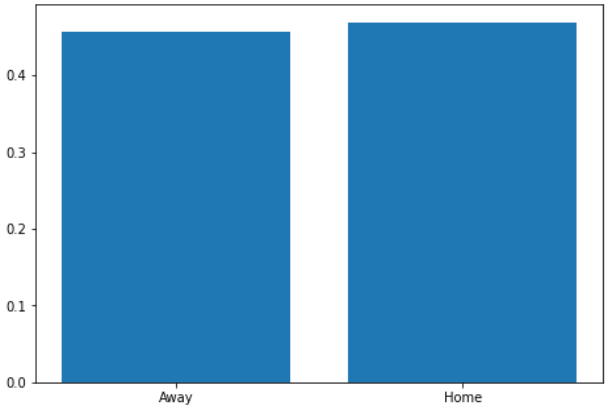




It should be taken into consideration that jump shots will have the most deviation as they rely heavily on range. With this we decided to create a shot chart to show the impact of range. From here the frequency of a shot on a court is shown and it is clear that the most taken shots in a specific area are in the paint while most of the jump shots are taken from behind the 3 point line. We see that shots are normally distributed on the x-axis centered at 0 while shots seem two reach two peaks on the y-axis in the paint around 0 and at the top of the three point line at around 250 on the y-axis. It should be noted that the majority of our observations come from jump shots, however this should be taken into account with our parameter **DISTANCE_TRUE** when performing our modeling as it will differentiate the jump shots from each other.

Home/Away Shot Percentages

Next we wanted to see how factors such as whether or not the team was playing at home or not would influence the shot percentage:

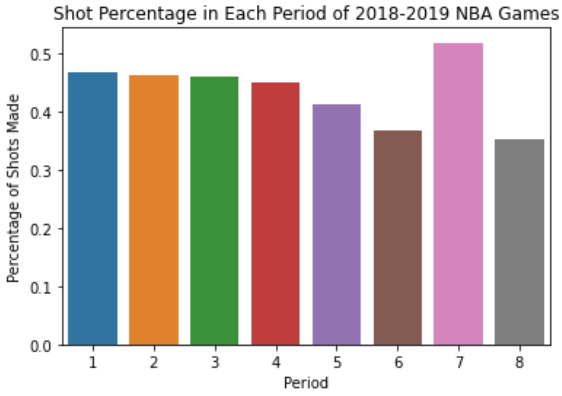


As we can see from this bar graph, the shooting percentage for players taking shots at their home court is slightly higher than those that are taking their shots at an away game. The home court

advantage shooting percentage is 46.94% while the away shooting percentage is about 45.66% with a 1.28% difference.

Period Shot Percentages

The last metric we looked at before conducting our machine learning was the shot percentage in each period of the game. We wanted to see if the period of which the game was had any affect on the chances for a shot to go in. With this, we created a bar chart of the shooting percentages in each period:



As we can see, the shooting percentage does minimally decrease throughout the periods but is still evident in its downwards trend. Outliers include the shooting percentages from periods 6,7 and 9 as the amount of data for that is drastically smaller than the first four periods of a normal game. Ignoring these outliers, the bar chart still demonstrates a decreasing trend.

III. Results

After coming to the dataframe that we wanted to perform supervised learning classification on, we split our data to 70% for the training set and reserved the remaining 30% for the test set. The machine learning models that we chose include: Logistic Regression, Linear Support Vector Classification and Random Forest. We were looking to predict **SHOT_MADE** as our response variable from the predictor variables **PERIOD, TIME, HOME, DISTANCE_TRUE** and **SHOT_PRC.**

Modeling

For our benchmark model, we chose to apply a logistic regression to a single feature, **DISTANCE_TRUE**, from our training data. **DISTANCE_TRUE** was selected for the benchmark model because, by our judgement, the distance from the hoop was the most important factor to determine whether a shot was made or not. We chose a logistic regression as it is simple to apply, understand and it is frequently used. We used MultiClassMetrics and BinaryClassificationMetrics to obtain performance indicators for the model. Without cross-validation and hyperparameter optimization, our initial accuracy score was 59.9%. To achieve maximum accuracy, we used TrainValidationSplit and ParamGridBuilder to cross validate and optimize parameters and ended up with the same accuracy and confusion matrix.

Other supervised learning models we considered were Logistic Regression with all of the features, Linear Support Vector Classification and Random Forest. We applied the same classification methods to the other models and optimized our results with cross-validation and

hyperparameter optimization. These were the resulting accuracy, precision, recall, F1 score and AUC values:

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC**
Benchmark	0.5989	0.5648	0.5525	0.5586	0.5955
Logistic Regression	0.6277	0.6458	0.4194	0.5086	0.6120
Linear SVC	0.6029	0.5691	0.5575	0.5632	0.5981
Random Forest	0.6400	0.6844	0.4012	0.5058	0.5981

To begin with, comparing the models off our first look in accuracy we see that our Benchmark Model performs the worst and our Random Forest Model gives us the best accuracy. However, with the data we are examining there are a lot more missed shots than made shots in general and we thought that accuracy was not the best way to evaluate our best model. Given this, we thought to consider precision and recall the most as they are measures of true positives. With the F1 score being the harmonic mean of the precision and recall scores we found this to be the best measure of success given our data. It is clear that the Linear Support Vector Classification had the highest F1 score so we chose this as our champion model as we value true positives the most. Through our hyperparameter optimization, we were able to quantify the sensitivity of each of our models by measuring the change in F1 score from the default model to the tuned model. All of our models demonstrate low sensitivity apart from the Linear SVC, which had its F1 score increase by 0.16 after its parameters were tuned. As a result, despite its superior performance, future work needs to take into consideration the potential volatility of the Linear SVC model.

IV. Conclusions

As previously mentioned, our champion model could be improved in various ways. Even though its accuracy was above 50%, it is not high enough to provide clear insights into the probability of a shot being made. In the future, we could include more seasons of data in order to obtain more information about made shots, as most of our models were not effective at predicting made shots. Additionally, we could subset and sample our data to include equivalent amounts of makes and misses to prevent any biases or false predictions. We noticed that the PERIOD and HOME features had similar shot percentages for each value. For instance, the overall shot percentage for the 1st, 2nd, 3rd, and 4th periods were similar. Consequently, we could choose features that lead to larger differences in shot percentage. Google's March Madness dataset contains many more variables and data points, and could be considered for use in additional research. Since our benchmark model was not the most successful, it seems as though adding more variables and constructing more complex models could lead to higher accuracy levels. Finally, the parameter grids that we constructed for the hyperparameter tuning could be expanded to include more possible permutations of parameters. This would allow us to more rigorously test different models and their performances.