Predicting the 2023 NBA Champions

Shoumik Chaudhuri, Stephen Hwang, Jasmine Zhang, Ningyu Gao, Anna Sasagawa, Jason Vo

Background & Motivation:

Each NBA season culminates in the NBA Playoffs, which is an end of season tournament featuring the top 8 teams from each conference (Eastern and Western). During the regular season, all 30 teams each play 82 games over a period lasting approximately 6 months. Then, 16 teams qualify for the playoffs by being in the top 8 teams by win percentage in their respective conference at the end of the regular season (with the 7th and 8th seeds being selected via a Play-In tournament, which we won't get into here). There are a maximum of 4 rounds that a team can go through. Each round of the playoffs tournament involves winning a best-of-7 series (i.e first to 4 wins) against their opponent. The NBA finals involve the winner of the Western Conference facing off against the winner of the Eastern Conference in a best-of-7, and the winner of this series is crowned that year's NBA champion (having won 16 games total).

For sports fans and gamblers, the NBA playoffs is one of the most exciting times of the year, often riddled with an array of upsets, thrilling moments, and intense matchups. One of the biggest debates that ubiquitously arises amongst anyone who follows the sport is "who will be this year's NBA champion?" Using the power of past data and advanced statistics, we aim to provide a statistically-backed accurate prediction.

Our project will make use of common machine learning techniques such as multiple linear regression as well as random forests to predict the 2023 NBA champion (and those of prior years!) based on statistics collected during the preceding regular season.

Data

For the data, we will be scraping the last 25 years of regular season data from basketball-reference.com, consisting of 512 observations. This table is known as `NBA_playoff_contenders`. The variable our models will try to predict is `Champion Share Score`, which is the ratio of playoff games won by that team that season to the number of possible playoff wins that can occur (16 in recent seasons). For example, for the 2022 NBA champions, the Golden State Warriors, their `Champion Share Score` would equal 1.0, as they had to win 16 games in the playoffs to secure the championship.

Observe that `Champion Share Score` values can only be between 0 and 1, with 0 representing a sweep in the first round (no wins) and 1 representing a championship victory (16 wins).

Our data contains 26 features, with most of them consisting of rudimentary regular season stats such as "Wins", "Age" (average age of team), and "Seed" (the final seed of the team going into the playoffs). Other features consist of advanced analytics like "SRS" (simple rating system) and "eFG%" (effective field goal percentage). For detailed explanations on our features, please reference the glossary provided by our data source, as well as the table of features in the appendix. Additionally, via the data extraction process, we've aggregated the "Playoff

Experience" feature, which is the total number of playoff games played by the roster of a team before that year. Most casual NBA fans know that prior experience in the playoffs and/or having a championship pedigree is *huge* when it comes to playoff success, which is why it intuitively made sense to go the extra mile to extract this data (just look at the Golden State Warriors, for example).

Speaking of data extraction, a great deal of data engineering effort has been expended during the ETL process. This project does not make use of any pre-cleaned datasets. All data was manually scrapped, processed, and aggregated from scratch, largely thanks to the BeautifulSoup python library. Further detail on this can be found in "NBA_data_engineering_142final.ipynb", which we went to great lengths to comment/annotate in order to make it readable. Please look through it, it is as valuable as this report.

Preliminary Approach: Multiple Linear Regression (OLS)

We decided to create a rudimentary model to predict using "Championship Share Score" using multiple linear regression (OLS). Much of the same process used to build OLS models throughout the course was applied here. Our feature process was as follows:

1. Use 70/30 training/testing split. While the data is chronological, we decided to *not* split the data chronologically, as many NBA fans agree that style of play has changed drastically over the years. We want our model to be diverse with different "eras" of basketball incorporated in order to prevent overfitting.
2. Use VIF to eliminate features in order to prevent collinearity
3. Eliminate statistically insignificant features with p-value > 0.05.
4. Optimizing feature selection to maximize out-of-sample R2 (OSR2) as well as in-sample R2.

After iterating through this process several times, we finally came up with an optimal model:

OLS Regression Results

Dep. Variable:	Champion Share Score	R-squared:	0.542
Model:	OLS	Adj. R-squared:	0.534
Method:	Least Squares	F-statistic:	69.12
Date:	Sat, 06 May 2023	Prob (F-statistic):	1.63e-56
Time:	21:56:10	Log-Likelihood:	65.581
No. Observations:	358	AIC:	-117.2
Df Residuals:	351	BIC:	-90.00
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0356	0.385	-0.092	0.926	-0.793	0.721
Seed	-0.0727	0.007	-9.962	0.000	-0.087	-0.058
Playoff Experience	0.0002	5.54e-05	4.317	0.000	0.000	0.000
L	-0.0056	0.002	-2.526	0.012	-0.010	-0.001
Pace	0.0082	0.003	2.556	0.011	0.002	0.015
Opp TOV%	-0.0202	0.010	-2.017	0.044	-0.040	-0.001
Opp FT/FGA	1.2780	0.478	2.674	0.008	0.338	2.218

Omnibus:	10.318	Durbin-Watson:	1.960
Prob(Omnibus):	0.006	Jarque-Bera (JB):	11.171
Skew:	0.334	Prob(JB):	0.00375
Kurtosis:	3.551	Cond. No.	2.16e+04

```
Mean Squared Error: 0.047952606705470774

Training (in-sample) R-squared: 0.5416225693844531
Testing (out-of-sample) R-squared: 0.42554189426146416
```

As you can see, with an in-sample R2 of 0.542 and OSR2 of 0.425, it is clear that some degree of overfitting is occuring in our model. An OSR2 of 0.425 isn't particularly great either; meaning that less than half of the variability in predicted "Champion Share Score" can be explained by our model. Our group also found it fascinating that "Seed" and "Playoff Experience" were the most statistically significant features, with p-values of 0. Intuitively speaking, this tracks with NBA fan knowledge quite strongly, so it was reassuring to see this be reflected in the model.

Advanced Approach: Random Forest Regression

After creating our preliminary model, which gave us some confidence that our data was indeed viable for our modeling use case, we decided to take a more advanced approach with Random Forest Regression. Random Forests seemed like a great choice for our use case, considering the fact that our data likely has non-linear relationships that can't be modeled as well by MLR. Additionally, Random Forests do the feature selection for us, eliminating the guesswork. Finally, Random Forests can reduce overfitting by aggregating predictions across multiple trees, which leads to a more robust model.

We created an initial random forest model, with largely default hyperparameters (explanation of hyperparameters can be found in appendix):

```
0] # Creating the random forest
   rf = RandomForestRegressor(max_features = 4, min_samples_leaf = 2,
                              n_estimators = 500, random_state=88, verbose = 0)

   rf.fit(X_train, y_train)
```

```
▾                          RandomForestRegressor
RandomForestRegressor(max_features=4, min_samples_leaf=2, n_estimators=500,
                      random_state=88)
```

```
Mean Squared Error: 0.007798093312048643

Training (in-sample) R-squared: 0.9119353745558948
Testing (out-of-sample) R-squared: 0.5930482906877905
```

As you can see from the increased OSR2 to 0.6, this random forest model is a significant step up from our preliminary OLS model. Over half of the variability in the predicted "Champion Share Score" can be attributed to this model, and we think this is pretty good for a dataset completely extracted and cleaned from scratch.

From here, we wanted to track down the optimal hyperparameters for our Random Forest model. We employed Grid Search Cross-Validation (GSCV), which is a technique used to find the optimal hyperparameters for a machine learning model. In this technique, a grid of

hyperparameters is defined, and the model is trained and evaluated using each combination of hyperparameters in the grid. We then perform k-fold cross-validation for each combination of hyperparameters in the grid, and compute the mean squared error (MSE) on the training set. The optimal set of hyperparameters is the one that yields the lowest MSE. For our project, we focused on finding the optimal value for the "max_features" hyperparameter (only 1 chosen for computational/time purposes), which ended up being 14 (see figure in appendix).

After this hyperparameter optimization, we noticed our OSR2 approximately remained the same at 0.58, but the hope is that due to k-folds cross validation, this model will suffer less from overfitting and be more accurate on future data (which is certainly seen in the smaller discrepancy between in-sample R2 and OSR2). We also observed that "Seed", "W", "Playoff Experience", and "MOV" (average margin of victory) were the most important features, as ranked by their ability to decrease entropy after each split. For more information on feature importance as well as a list of the top 10 most important features, see appendix.

Ok enough ML, how well does it actually predict NBA champions?

Our optimal model (tuned with GSCV), was **able to correctly predict the NBA champion over the past 15 years (2008-2022) 12 out of 15 times** (see appendix)**!!!** Of the 3 times that it mispredicted the champion, the next highest predicted Champion Share Score ended up being the champion in 2 of those three times. The mispredicted years were 2008, 2016, and 2021. 2008 predicted the Lakers while the champions ended up being the Celtics, who beat the Lakers in an intense 6 game final series. 2016 predicted the Warriors, and the champions ended up being the Cavaliers, who beat them in the finals. This was somewhat of an anomaly year because the Cavaliers came back from a historic 1-3 deficit to win the finals. Additionally, the 2016 Warriors recorded the greatest regular season record (73-9) of all time, making them clear favorites to win. Altogether, 2016 is regarded as one of the greatest NBA finals of all time. Therefore, we can't really fault the model for not being able to make a history-defying prediction. 2021 is the only truly incorrect prediction of our model. The Milwaukee Bucks ended up winning, but our model predicted the 76ers, Clippers, and Suns to have the highest CSS, in that order.

As a group, we were extremely impressed with these results. We find it especially fascinating that the model was able to correctly predict the champion for the year 2020, during the pandemic-induced NBA playoff bubble, which was full of external factors. Being able to correctly predict 80% of the last 15 NBA champions is something we're really proud of, especially because there is still room for growth with this model.

What about this year's NBA Championship?

We're currently in exciting times, with this year's 2023 NBA Playoffs in full-swing at the time of writing this. Our model predicts the following 3 teams to have the highest CSS for 2023, in order: Milwaukee Bucks, Boston Celtics, and Philadelphia 76ers. Unfortunately, despite being the best regular season team, the Milwaukee Bucks were eliminated in a stunning first round upset, ruining our chances of our model predicting this year's NBA champion. Therefore, we're

going to go with our model's second pick, the **Boston Celtics, to be the 2023 NBA Champion.** The Celtics had a predicted CSS less than 0.03 behind the Milwaukee Bucks.

Conclusion and Impact

Overall, we believe that we were able to come up with a pretty accurate winner-predicting model, and largely consider this experiment to be a success. Our random forest model was better than our OLS model in pretty much every way imaginable (as expected). However, there are a few things that we'd tweak to get an even better model. We could have made our random forest regression model more accurate by using Grid Search Cross Validation on *all* of the hyperparameters, not just one. This would have given us a model with more optimized parameters, and likely yielded a model with higher OSR2. For the sake of compute time and complexity we chose to optimize just one hyperparameter. Furthermore, another way we could have improved our model would be to include some sort of feature based on injury/player health. There's no question that injuries to star players plague every sports team, and the NBA playoffs are no different. An ideal model would figure out some way to incorporate injury data and player health/status into this model. Furthermore, an important inflection point during the NBA regular season is the trade deadline, which is usually in mid-February. After this date, some NBA teams might have *completely* different rosters and ability to win, and teams usually play harder as the race to qualify for the playoffs comes to an end. Therefore, we feel like an improved model should include features that highlight this sub-section of the season, such as "Post trade-deadline record", etc.

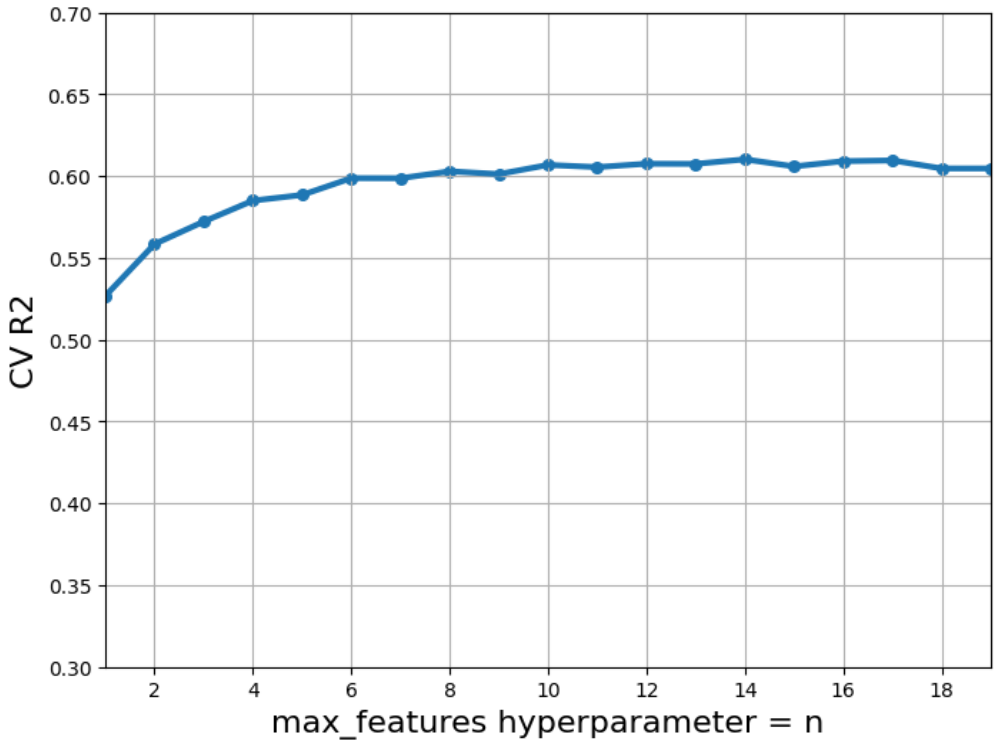
Our model has a broad application impact. Of course, it can trivially be enjoyed by fans looking to make fun predictions against their friends, but we could also see this model being employed by team executives in order to understand which features of a team are most integral for championship success. Furthermore, we could see both gamblers and casinos employing this model to make wagers or set casino odds for sports bets in the pursuit of financial gain. Within this subpopulation, the model could even be expanded to include features from a sports betting casino (such as, the odds on each team to win the playoffs). From here, insights on the accuracy of the odds could be drawn from the model to be exploited by gamblers or improved upon by the casino in order to create financial gain.

Appendix:

Description of Hyperparameters used in Random Forest Regression

Hyperparameter	Description
n_estimators	The number of decision trees in the forest. Increasing the number of trees can improve performance, but can also lead to overfitting.
max_features	The number of features to consider when splitting a node in each decision tree. A smaller value can help prevent overfitting, while a larger value can lead to improved performance.
min_samples_leaf	The minimum number of samples required to be at a leaf node. A larger value can help prevent overfitting, while a smaller value can lead to improved performance.
random_state	The seed used by the random number generator. Setting a seed ensures that the results are reproducible.

Optimizing max_features for the highest R2 (optimal value = 14)



R2 Metrics and Most Important Features of Our Optimal Random Forest Model:

```
[55] print('Cross-validated R2:', round(rf_cv.best_score_, 5))
     print('OSR2:', round(OSR2(rf_cv, X_test, y_test, y_train), 5))

     Cross-validated R2: 0.6103
     OSR2: 0.58037
```

Similarly to our initial model, we can also check for feature importance.

```
pd.DataFrame({'Feature' : X_train.columns,
              'Importance score': 100 * rf_cv.best_estimator_.fe
```

	Feature	Importance score
0	Seed	39.5
1	W	9.7
2	MOV	6.2
3	Playoff Experience	5.8
4	L	5.6
5	NRtg	4.2
6	SRS	4.1
7	TS%	2.0
8	Pace	1.8
9	Opp FT/FGA	1.8
10	ORB%	1.7

These scores represent the relative importance of each feature in predicting the target variable, based on the decrease in impurity that each feature causes when it is used to split a node in the decision trees of the random forest. The feature importance scores are calculated as the average of the impurity decrease over all the trees in the forest, weighted by the number of samples that are split on each node.

Championship Predictions Over the Past 15 Years

```
def predict_champion(year):
    if year == 2023:
        last_year = nba_playoff_contenders_2023.drop(10)
    else:
        last_year = nba_playoff_contenders[nba_playoff_contenders["Year"] == year]
    last_year_dropped = last_year.drop(columns = ["Team", "Champion Share Score"])
    last_year["Predicted CSS"] = rf_cv.predict(last_year_dropped)
    last_year = last_year.sort_values("Predicted CSS", ascending = False)
    return last_year["Team"].iloc[0]
```

```
[133] # Predicting the champion for each year!
      predictions_by_year = [(year, predict_champion(year)) for year in range(2008, 2023)]
      predictions_by_year
```

```
[(2008, 'Los Angeles Lakers*'),
 (2009, 'Los Angeles Lakers*'),
 (2010, 'Los Angeles Lakers*'),
 (2011, 'Dallas Mavericks*'),
 (2012, 'Miami Heat*'),
 (2013, 'Miami Heat*'),
 (2014, 'San Antonio Spurs*'),
 (2015, 'Golden State Warriors*'),
 (2016, 'Golden State Warriors*'),
 (2017, 'Golden State Warriors*'),
 (2018, 'Golden State Warriors*'),
 (2019, 'Toronto Raptors*'),
 (2020, 'Los Angeles Lakers*'),
 (2021, 'Philadelphia 76ers*'),
 (2022, 'Golden State Warriors*')]
```

Dataset Features and their descriptions:

Feature Name	**Description**
Seed	The playoff seeding of the team (between 1 and 8)
Champion Share Score	# of Playoff games won / 16 (feature to predict)
Playoff Experience	Sum of roster's prior playoff games.
MOV	The team's margin of victory in the regular season
SRS	The team's simple rating system in the regular season
eFG%	The team's effective field goal percentage in the regular season

TOV%	The team's turnover percentage in the regular season
ORB%	The team's offensive rebound percentage in the regular season
FT/FGA	The team's free throws made per field goal attempt in the regular season
Opp eFG%	The opponent's effective field goal percentage in the regular season
Opp TOV%	The opponent's turnover percentage in the regular season
DRB%	The team's defensive rebound percentage in the regular season
Opp FT/FGA	The opponent's free throws made per field goal attempt in the regular season