NBA Matchup Prediction Application

Alex Brockman, Andrew Haisfield, Anish Prasanna, Carlos Samaniego

December 6,2019

1 Abstract

For this project, our goal was to predict the outcome of NBA games using prior statistics. We decided that the best way to do this is to project the Margin of Victory (projected points scored - projected points allowed) for each NBA team, and then given the scheduled matchups on a particular day, we would predict that the team with the higher projected Margin of Victory would win the game. We ended up using basketball-reference.com as the source for all of our historical basketball data, and were able to scrape the data from the website using the python library BeautifulSoup. Once we collected all of the required statistics, we then implemented our own Brute Force feature selection method by running each possible combination of features and using the combination with the most optimal R-Squared value. Next, we tested the performance of four different regression algorithms (Multi-Linear, KNN, Decision Tree, and Gradient Boosting) in order to select which one to use for our final model. After comparing the Time Series Average Cross Validated MSEs, we determined that the Gradient Boosting regressor was the ideal algorithm, since it had the lowest MSE. However, due to the significant increased time cost in comparison, we implemented the KNN regressor instead. Finally, we used this model to predict the NBA games for the current date and displayed them onto our Flask web application.

2 Background

The NBA is amongst the most difficult league to predict match victories. For this reason, it continually generates massive revenue from a dedicated fan base. The aim of our final project is to predict the margin of victory of a team winning its game before the match up.

For example, let's say that on Tuesday, November 19th, the Heat and the Bulls were to play each other. We would like our model to generate a prediction as to how many points the Heat would win by. This could then be mapped to a probability in order to provide an additional, more intuitive metric to exemplify a team's projected dominance.

This information could be practically used for numerous applications. Firstly, one use could be for TV broadcasters to use this information to make an educated decision as to which game to cover given several games are being televised simultaneously. This is of tremendous significance for the NBA, since optimizing the viewership of games is crucial in generating interest. If boring games are televised, at the expense of more interesting games, viewership may be impacted significantly. Therefore, it is of utmost importance for the NBA, and its broadcasters to understand this crucial factor, in order to maintain its business.

An additional application could be for gambling purposes. Over the course of a week or so, often times Sportsbooks and Casinos modify their betting lines to reflect the change in betting demands on the two teams. Therefore, this movement can be exploited by interested gamblers to gain value on the lines. To illustrate this idea, suppose at the inception of the week, the underdog has heavy betting interest, and so the Sportsbook designates them at a near even line, suggesting that the match-up is a close one. If a gambler wanted to bet on the underdog, the start of the week would not be a time that would present any inherent value. However, if the gambler were to use this application, he would be able to identify moments before the matchup where significant value could stand to be gained. Optimizing value effectively increases the expected profit long-term, which could be a more interesting monetarily related use of our application.

3 Prior Work

Predicting sports matchups is no new concept. Most popular sports books employ a similar computer enabled tactic in order to allow them to choose the best betting lines. It would be rare to find their methods publicly available, however, because it would conflict with their business interests. Those methods are trade secrets that they would not want to release. FiveThirtyEight [4] has a repository of projects that are posted on their website. One of their projects is almost identical to ours in its goal to display game predictions for each day during the season.

When comparing their model against ours, we observed that our predictions are within 10 percent of theirs. All of our predictions for the winners of each game matched theirs as well.

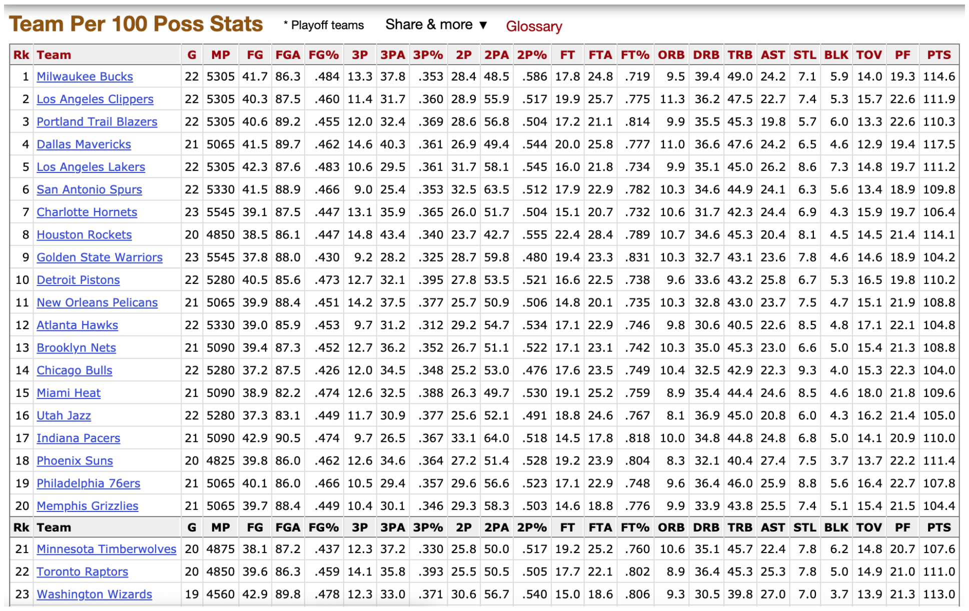
4 Data and Methods

Data:

In order to find relevant data, we needed to use a data source that was free, data rich and easily able to be web scraped. We settled on the website http://www.basketball-reference.com [3] [2] [1] for our data. The website contained simple tables that we were able to navigate and scrape without much difficulty (Figure 1,2). We used the python library BeautifulSoup to parse through the tables that were necessary for the project. In particular we scraped three different tables. First we scraped the NBA game schedules from the 2015 season to the 2020 season [3]. We then scraped the 2015-2020 all team stats per possession table [2] and the 2015-2020 all opponents stats per possession table [1]. We chose the first table because in order to compute our predictions, we needed to know the schedules for each team. Rather than simply reference stats for the entire team, we decided to choose the all team stats per possession tables because it would return better results overall. Some NBA teams possess the ball more than others and we wanted to remove that variable from our model. We decided to scrape every column's data within those tables so we could determine which features we needed for our model.

We struggled with the "Team Per 100 Possession" stats table when initially using BeautifulSoup. When scraping the page we noticed that the tables were missing from the output provided by the library. After much trial and error we discovered that BeautifulSoup does not parse comments within the web page source files. We found a work around that served our needs well. Our solution included using the find-all method to parse the entire file including comments before deleting the comment syntax from the code. The correct tables were then able to be parsed by BeautifulSoup. We suspect that the website performs this practice to discourage web scraping for current statistics as we were easily able to scrape tables from previous years.

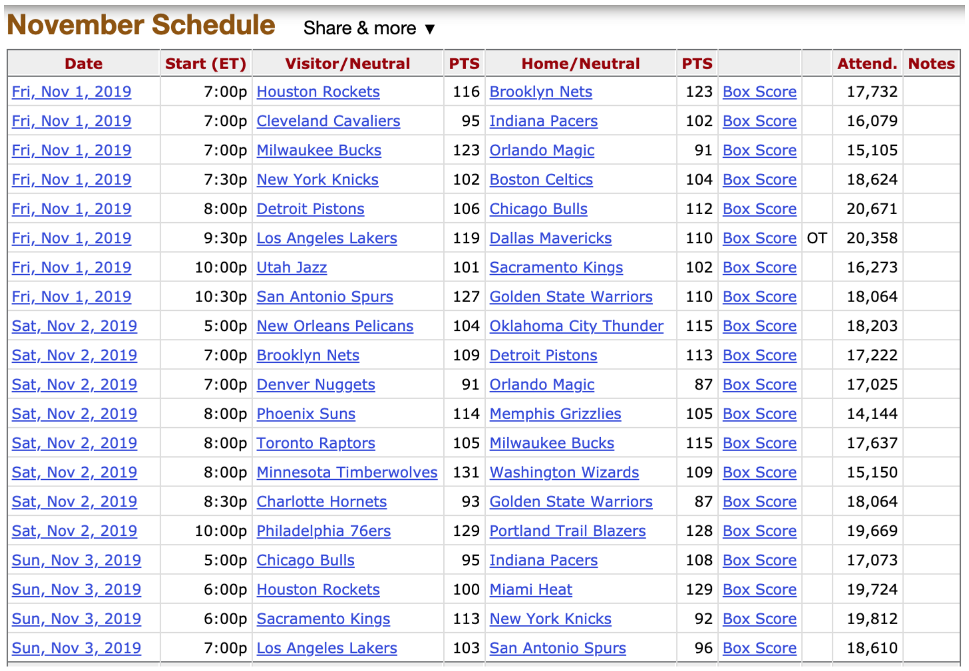
Below are two examples of the tables that we scraped to include in our model:



Team Per 100 Poss Stats * Playoff teams Share & more ▼ Glossary

Rk	Team	G	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
1	Milwaukee Bucks	22	5305	41.7	86.3	.484	13.3	37.8	.353	28.4	48.5	.586	17.8	24.8	.719	9.5	39.4	49.0	24.2	7.1	5.9	14.0	19.3	114.6
2	Los Angeles Clippers	22	5305	40.3	87.5	.460	11.4	31.7	.360	28.9	55.9	.517	19.9	25.7	.775	11.3	36.2	47.5	22.7	7.4	5.3	15.7	22.6	111.9
3	Portland Trail Blazers	22	5305	40.6	89.2	.455	12.0	32.4	.369	28.6	56.8	.504	17.2	21.1	.814	9.9	35.5	45.3	19.8	5.7	6.0	13.3	22.6	110.3
4	Dallas Mavericks	21	5065	41.5	89.7	.462	14.6	40.3	.361	26.9	49.4	.544	20.0	25.8	.777	11.0	36.6	47.6	24.2	6.5	4.6	12.9	19.4	117.5
5	Los Angeles Lakers	22	5305	42.3	87.6	.483	10.6	29.5	.361	31.7	58.1	.545	16.0	21.8	.734	9.9	35.1	45.0	26.2	8.6	7.3	14.8	19.7	111.2
6	San Antonio Spurs	22	5330	41.5	88.9	.466	9.0	25.4	.353	32.5	63.5	.512	17.9	22.9	.782	10.3	34.6	44.9	24.1	6.3	5.6	13.4	18.9	109.8
7	Charlotte Hornets	23	5545	39.1	87.5	.447	13.1	35.9	.365	26.0	51.7	.504	15.1	20.7	.732	10.6	31.7	42.3	24.4	6.9	4.3	15.9	19.7	106.4
8	Houston Rockets	20	4850	38.5	86.1	.447	14.8	43.4	.340	23.7	42.7	.555	22.4	28.4	.789	10.7	34.6	45.3	20.4	8.1	4.5	14.5	21.4	114.1
9	Golden State Warriors	23	5545	37.8	88.0	.430	9.2	28.2	.325	28.7	59.8	.480	19.4	23.3	.831	10.3	32.7	43.1	23.6	7.8	4.6	14.6	18.9	104.2
10	Detroit Pistons	22	5280	40.5	85.6	.473	12.7	32.1	.395	27.8	53.5	.521	16.6	22.5	.738	9.6	33.6	43.2	25.8	6.7	5.3	16.5	19.8	110.2
11	New Orleans Pelicans	21	5065	39.9	88.4	.451	14.2	37.5	.377	25.7	50.9	.506	14.8	20.1	.735	10.3	32.8	43.0	23.7	7.5	4.7	15.1	21.9	108.8
12	Atlanta Hawks	22	5330	39.0	85.9	.453	9.7	31.2	.312	29.2	54.7	.534	17.1	22.9	.746	9.8	30.6	40.5	22.6	8.5	4.8	17.1	22.1	104.8
13	Brooklyn Nets	21	5090	39.4	87.3	.452	12.7	36.2	.352	26.7	51.1	.522	17.1	23.1	.742	10.3	35.0	45.3	23.0	6.6	5.0	15.4	21.3	108.8
14	Chicago Bulls	22	5280	37.2	87.5	.426	12.0	34.5	.348	25.2	53.0	.476	17.6	23.5	.749	10.4	32.5	42.9	22.3	9.3	4.0	15.3	22.3	104.0
15	Miami Heat	21	5090	38.9	82.2	.474	12.6	32.5	.388	26.3	49.7	.530	19.1	25.2	.759	8.9	35.4	44.4	24.6	8.5	4.6	18.0	21.8	109.6
16	Utah Jazz	22	5280	37.3	83.1	.449	11.7	30.9	.377	25.6	52.1	.491	18.8	24.6	.767	8.1	36.9	45.0	20.8	6.0	4.3	16.2	21.4	105.0
17	Indiana Pacers	21	5090	42.9	90.5	.474	9.7	26.5	.367	33.1	64.0	.518	14.5	17.8	.818	10.0	34.8	44.8	24.8	6.8	5.0	14.1	20.9	110.0
18	Phoenix Suns	20	4825	39.8	86.0	.462	12.6	34.6	.364	27.2	51.4	.528	19.2	23.9	.804	8.3	32.1	40.4	27.4	7.5	3.7	13.7	22.2	111.4
19	Philadelphia 76ers	21	5065	40.1	86.0	.466	10.5	29.4	.357	29.6	56.6	.523	17.1	22.9	.748	9.6	36.4	46.0	25.9	8.8	5.6	16.4	22.7	107.8
20	Memphis Grizzlies	21	5065	39.7	88.4	.449	10.4	30.1	.346	29.3	58.3	.503	14.6	18.8	.776	9.9	33.9	43.8	25.5	7.4	5.1	15.4	21.5	104.4
Rk	Team	G	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
21	Minnesota Timberwolves	20	4875	38.1	87.2	.437	12.3	37.2	.330	25.8	50.0	.517	19.2	25.2	.760	10.6	35.1	45.7	22.4	7.8	6.2	14.8	20.7	107.6
22	Toronto Raptors	20	4850	39.6	86.3	.459	14.1	35.8	.393	25.5	50.5	.505	17.7	22.1	.802	8.9	36.4	45.3	25.3	7.8	5.0	14.9	21.0	111.0
23	Washington Wizards	19	4560	42.9	89.8	.478	12.3	33.0	.371	30.6	56.7	.540	15.0	18.6	.806	9.3	30.5	39.8	27.0	7.0	3.7	13.9	21.3	113.0

Figure 1: Team Per 100 Poss Stats Table



November Schedule Share & more ▼

Date	Start (ET)	Visitor/Neutral	PTS	Home/Neutral	PTS			Attend.	Notes
Fri, Nov 1, 2019	7:00p	Houston Rockets	116	Brooklyn Nets	123	Box Score		17,732	
Fri, Nov 1, 2019	7:00p	Cleveland Cavaliers	95	Indiana Pacers	102	Box Score		16,079	
Fri, Nov 1, 2019	7:00p	Milwaukee Bucks	123	Orlando Magic	91	Box Score		15,105	
Fri, Nov 1, 2019	7:30p	New York Knicks	102	Boston Celtics	104	Box Score		18,624	
Fri, Nov 1, 2019	8:00p	Detroit Pistons	106	Chicago Bulls	112	Box Score		20,671	
Fri, Nov 1, 2019	9:30p	Los Angeles Lakers	119	Dallas Mavericks	110	Box Score	OT	20,358	
Fri, Nov 1, 2019	10:00p	Utah Jazz	101	Sacramento Kings	102	Box Score		16,273	
Fri, Nov 1, 2019	10:30p	San Antonio Spurs	127	Golden State Warriors	110	Box Score		18,064	
Sat, Nov 2, 2019	5:00p	New Orleans Pelicans	104	Oklahoma City Thunder	115	Box Score		18,203	
Sat, Nov 2, 2019	7:00p	Brooklyn Nets	109	Detroit Pistons	113	Box Score		17,222	
Sat, Nov 2, 2019	7:00p	Denver Nuggets	91	Orlando Magic	87	Box Score		17,025	
Sat, Nov 2, 2019	8:00p	Phoenix Suns	114	Memphis Grizzlies	105	Box Score		14,144	
Sat, Nov 2, 2019	8:00p	Toronto Raptors	105	Milwaukee Bucks	115	Box Score		17,637	
Sat, Nov 2, 2019	8:00p	Minnesota Timberwolves	131	Washington Wizards	109	Box Score		15,150	
Sat, Nov 2, 2019	8:30p	Charlotte Hornets	93	Golden State Warriors	87	Box Score		18,064	
Sat, Nov 2, 2019	10:00p	Philadelphia 76ers	129	Portland Trail Blazers	128	Box Score		19,669	
Sun, Nov 3, 2019	5:00p	Chicago Bulls	95	Indiana Pacers	108	Box Score		17,073	
Sun, Nov 3, 2019	6:00p	Houston Rockets	100	Miami Heat	129	Box Score		19,724	
Sun, Nov 3, 2019	6:00p	Sacramento Kings	113	New York Knicks	92	Box Score		19,812	
Sun, Nov 3, 2019	7:00p	Los Angeles Lakers	103	San Antonio Spurs	96	Box Score		18,610	

Figure 2: Example Schedule Table

Model Creation:

Once we collected all of the data, we put together a model that predicted the outcome of each NBA game for a given day. Initially we tried predicting the total points scored for each team, but what we quickly realized was that this model over-estimated performance for teams with a good offense but a bad defense, while under-estimating teams with bad offenses but good defenses. Therefore, we decided to update our model to predict margin of victory (Points Scored - Points Allowed), as this accounted for defensive-minded teams much better.

After settling in on what we wanted to predict, the next step was to determine which subset of statistics would most-accurately project the margin of victory for a given team. With that being the case, we designed our model to evaluate the R-Squared value for each possible combination of predictor-variables (starting from length=1 up through length=total number of variables) in a Brute Force manner. From there, we found the subset of variables for each evaluated-length that had the highest R-Squared value and grouped them together. By looking at a plot of the R-Squared values for these selected subsets, we noted the point of highest-curvature (where the plot begins to plateau) on the graph to determine the model we want to use. We selected the subset with 6 features as our model (Figure 3).

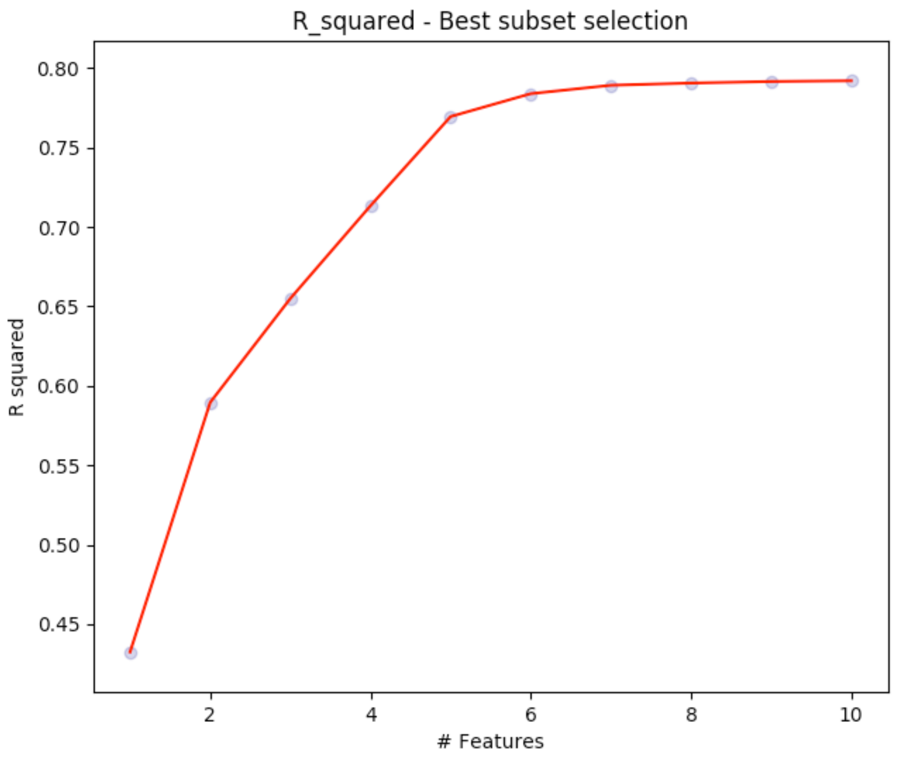
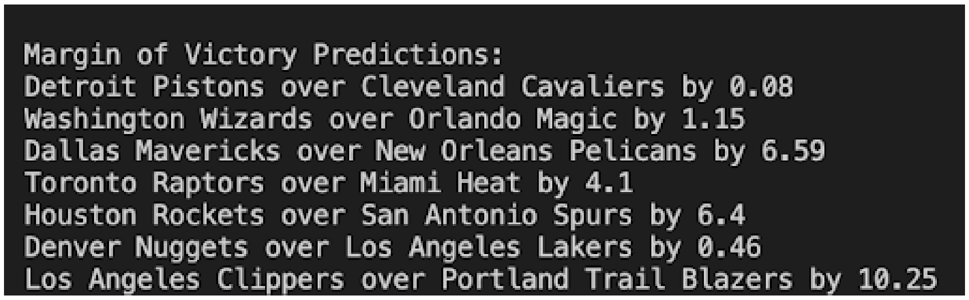


Figure 3: R-Squared Graph

After our optimal Multiple Linear Regression model was generated, we then predicted the margin of victory for each of the 30 teams in the NBA. Following this, the next step was to pull the NBA schedule for a given day, and based on what the matchups were for that day, we predicted that the team with the higher-projected margin of victory would win the matchup as seen in Figure 4.



```
Margin of Victory Predictions:
Detroit Pistons over Cleveland Cavaliers by 0.08
Washington Wizards over Orlando Magic by 1.15
Dallas Mavericks over New Orleans Pelicans by 6.59
Toronto Raptors over Miami Heat by 4.1
Houston Rockets over San Antonio Spurs by 6.4
Denver Nuggets over Los Angeles Lakers by 0.46
Los Angeles Clippers over Portland Trail Blazers by 10.25
```

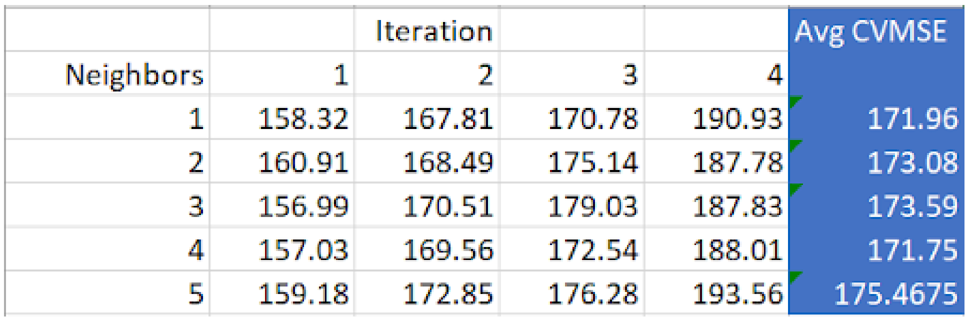
Figure 4: Margin of Victory Output

Methods Used:

After our baseline Multiple Linear Regression model was generated, we sought to identify additional Machine Learning algorithms (provided by the SciKit Library) [6] [7] [5] to determine which was best at predicting Margin of Victory(MoV) for a given matchup. The regression algorithms used were KNN Regression [6], Decision Tree Regression [7], and Gradient Boosting Regression [5]. To differentiate between the algorithms, we used Mean Squared Error(MSE). This metric was selected, due to its consideration of the discrepancy between the MoV our model predicts, and the actual MoV. For this reason, MSE lends itself particularly well for assessing regression results. Additionally, we needed to identify the optimal settings for each algorithm. These were the N-neighbors for KNN, and the max-depth for Decision Tree. We treated Gradient Boost as a black-box, and simply ran it under numerous settings to understand which would work best for our problem. In addition, we recorded the average time overall among all iterations to determine if there was a significant time cost associated with the algorithms. After comparison of the average time series cross validation of each respective regression algorithm, the one with the best overall MSE and time cost combination was selected to be implemented in the final website.

5 Results

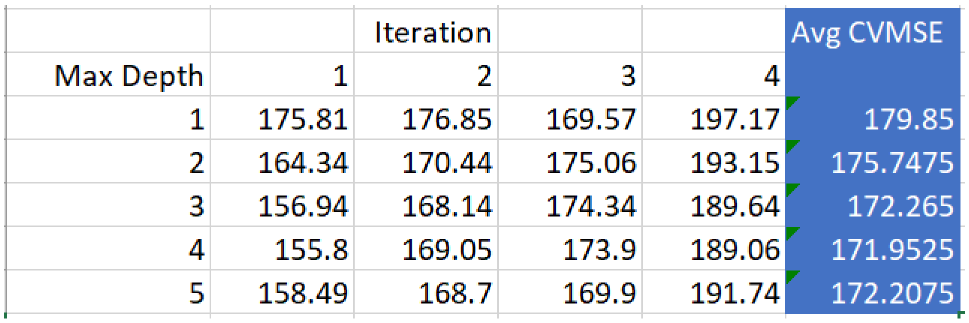
First, we sought to further understand the conditions at which each algorithm would perform best. For KNN, our group found the Average Cross Validated MSE (ACVMSE) for each N up to N=5. We used the least of these values (N=4) as our setting for the final algorithm comparison (Figure 5).



		Iteration			Avg CVMSE
Neighbors	1	2	3	4	
1	158.32	167.81	170.78	190.93	171.96
2	160.91	168.49	175.14	187.78	173.08
3	156.99	170.51	179.03	187.83	173.59
4	157.03	169.56	172.54	188.01	171.75
5	159.18	172.85	176.28	193.56	175.4675

Figure 5: Average Cross Validated MSE Table

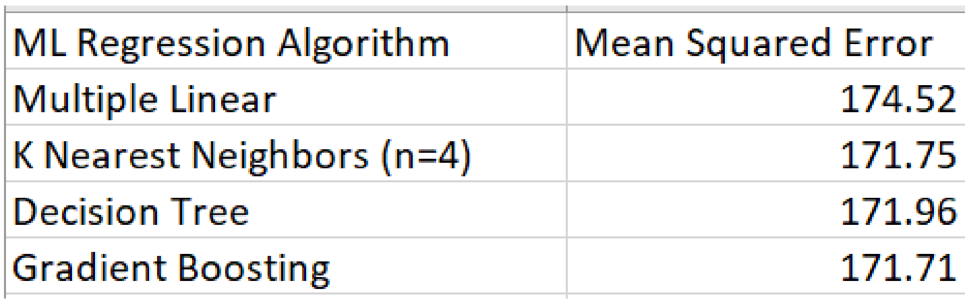
Next, we attempted to optimize the Decision Tree algorithm by identifying the best Max Depth of the tree. After comparison of the ACVMSE(Figure 6) between the various Max Depths, we determined that the ideal Max Depth was 4. We used this Max Depth for the purposes of our final algorithm comparison.



		Iteration			Avg CVMSE
Max Depth	1	2	3	4	
1	175.81	176.85	169.57	197.17	179.85
2	164.34	170.44	175.06	193.15	175.7475
3	156.94	168.14	174.34	189.64	172.265
4	155.8	169.05	173.9	189.06	171.9525
5	158.49	168.7	169.9	191.74	172.2075

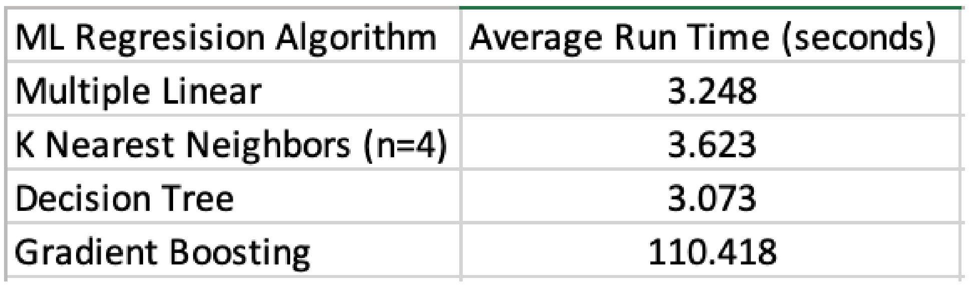
Figure 6: Decision Tree Table

Lastly, we attempted to optimize the Gradient Boosting regression algorithm. However, given that there were numerous parameters (n estimators, max depth, min sample split, learning rate, loss), we were unsure of the best method to approach this. After some time, we settled on the following parameters (n-estimators = 500, max depth = 3, min sample split = 2, learning rate = .01, loss = ls). These settings allowed for a ACVMSE of 171.71. This was our best ACVMSE(Figure 7). Although it was better than the other algorithms, it had a significant time cost associated with its use - a110 seconds cross iteration avg.(Figure 8). For this reason, we selected the KNN regressor (3.62 second cross iteration avg.) to be implemented for the web application, in order to improve the user experience.



ML Regression Algorithm	Mean Squared Error
Multiple Linear	174.52
K Nearest Neighbors (n=4)	171.75
Decision Tree	171.96
Gradient Boosting	171.71

Figure 7: Performance Table



ML Regresision Algorithm	Average Run Time (seconds)
Multiple Linear	3.248
K Nearest Neighbors (n=4)	3.623
Decision Tree	3.073
Gradient Boosting	110.418

Figure 8: ML Regression Average Run Times

Upon further analysis of each of our algorithms MSE during each iteration, we noticed several interesting details. Firstly, our MSE was growing irrespective of the algorithm (Figure 8). However, after much consideration, numerous events within the NBA could be used to explain this phenomenon, but the increase of player movement was perhaps the biggest reason. With the rise of social media within today's society, players are now held accountable to their success more than ever, and many players will now do whatever it takes to be on a team that can compete for championships. In 2016, for example, Kevin Durant left the Oklahoma City Thunder to join the Golden State Warriors, which was the team that knocked out his own Thunder from the playoffs the season before. His reasoning? To be on a better team with a higher chance of winning it all. The same thing happened in 2018, when Lebron James left the Cleveland Cavaliers to join the Los Angeles Lakers, once again shifting the entire competitive landscape of the NBA. Additionally, other superstars like Jimmy Butler, and Kawhi Leonard also transitioned to other teams further influencing our models negatively.With that being the case, it makes sense why the statistics produced from the 2015 season were not effective in predicting the results for the 2019 season, as the rosters of each team have changed more significantly each consecutive year.

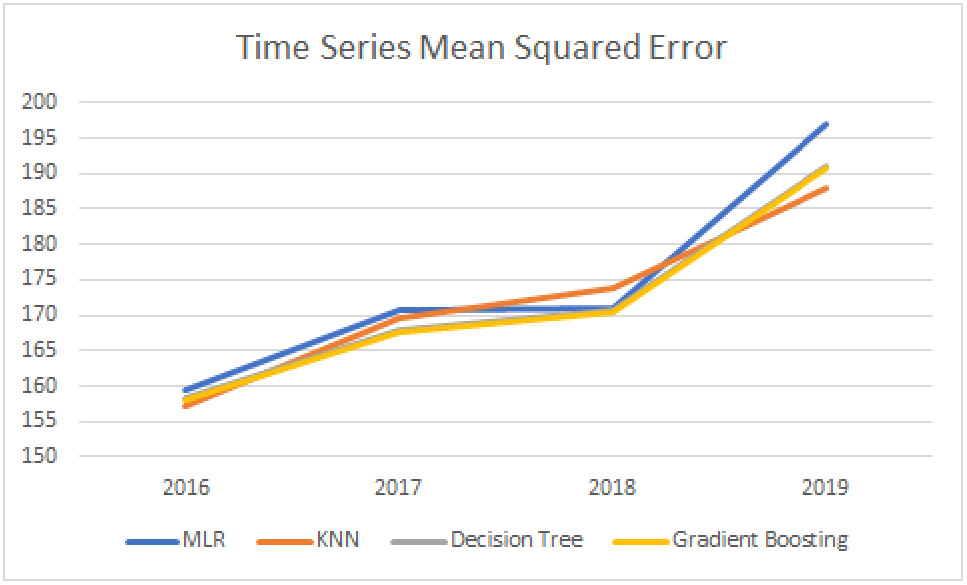


Figure 9: Time Series Mean Squared Graph

6 Conclusion/Future-Work/References

Conclusion:

To surmise, our final project attempted to create a web application that could be used to predict the probability of a victory, based on a projected Margin of Victory. Four machine learning algorithms were used to compare amongst each other (Multiple Linear Regression, KNN Regression, Decision Tree Regression, and Gradient Boosting Regression). After optimizing the KNN regressor (N=4), and the Decision Tree regressor (Max Depth = 4), we implemented a time series cross validation algorithm to determine the best algorithm. All these algorithms had increasing MSEs over each successive iteration, yet Gradient Boosting Regression had the lowest Average Cross Validated MSE (171.71). For the purposes of the web application, however, the KNN regressor was selected, as it had a similar ACVMSE (171.75), yet a significantly better time cost (3.62 seconds iteration average).

Future-Work:

Several additional features could be implemented to add further utility to our potential users. One instance of this would be to predict in-game probabilities of winning between quarters of a match. Another useful feature would be providing access for users to view previously predicted games, and their respective outcomes. Finally, in terms of our experimental design, bolstering our dataset by adding previous decades of statistical team data would be crucial in lowering the MSE over that new, extended period of time. This would provide increased confidence for the user to trust that our models will be able to improve as the years progress, given that, currently, our models are using data which is anomalous in the more grand context of NBA history. Each of these additions to our application would be tremendously valuable and further increase its practicality.

Predicting in-game probabilities is a highly desired feature amongst many sports media, and sports analytics companies. These companies compute these predictions primarily to drive traffic to their websites, and promote viewer interest in matchups through social media. Furthermore, an additional use case would be for Sport Broadcasters. Having in-game probabilities easily accessible would identify critical junctures that sports broadcasters could use to switch between games that were more interesting than the ones currently being televised. Lastly, gamblers could use these probabilities to arbitrage or hedge previously held positions if the outcome is not going in their favor. Conversely, Sportsbooks and Casinos could use these odds to construct betting lines, and spreads for the games they wish to offer to the public. All of these examples are highly valuable commodities to their respective party, and would serve to hold substantial value. Establishing a database of previous predictions is another crucial feature that would have value if implemented. Users would be able to self-judge if the model worked well enough for their particular use case. Furthermore, the probabilities of winning for particular teams could be analyzed to detect trends in season. This could have numerous implications for team managers, as they may be more inclined to perform transactions, given their teams expected probabilities and the trends that could emerge from previous probabilities. Providing a quick reference for past predictions would also allow for further analysis to be conducted on our results, which would allow for further optimization.

Finally, and arguably most importantly, improving our training set to include previous decades of NBA data would allow us to identify which periods of time, within the NBA, had the most turnover, in regards to personnel and coaching movement, and player injuries. Viewing the NBA from a broader time landscape would give us clarity in this matter, and provide our application with further credibility against competitors.

References

[1] Nba all opp stats per possession.

[2] Nba all team stats per possession.

[3] November 2019-20 nba schedule and results.

[4] J. Boice, R. Dottle, E. Koeze, G. Wezerek, N. Silver, N. Paine, A. Waters, and E. Stein. 2019-20 nba predictions, Dec 2019.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 3.2.4.3.5. sklearn.ensemble.gradientboostingclassifier.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. sklearn.neighbors.kneighborsclassifier.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. sklearn.tree.decisiontreeclassifier.