Final Project Report
STATS 170B, Spring 2020

Project Title: NBA Team Victor Prediction

Authors:
Caleb Pitts, 24763389, clpitts@uci.edu
Ray Yi, 43695282, chirayy@uci.edu

Github:

Introduction / Problem Statement

The National Basketball Association (NBA) is the biggest basketball league in the world. It had approximately 38.05 million fans in 2019. With 30 teams in the league to cheer for, the biggest debate every year boils down to: “Who will win the championship this year?”. As NBA fans ourselves, we decided to utilize the NBA api that contains box scores from nba.com and advanced statistics from basketball-reference.com to build a model that predicts NBA teams’ records based on individual players’ past statistics. Our objective here is to figure out whether our approach, given a team’s current season roster and taking that roster’s previous season performance is a viable method to predict current season team performance. If so, how accurate would that prediction be? Our second objective is to assess that given past specific matchup data, are we able to reliably predict the victor of a matchup for upcoming games, if so, how accurate is this victor prediction?

After running different machine learning models tackling the two objectives, we gathered some key insights and results. We are able to predict team season records with 65% accuracy and specific matchup victors with 74% accuracy. Advanced statistics, features derived from formulas (Ex: PER - Player Efficiency), are particularly useful, which makes sense as they are essentially composed of standard box scores. Random forest worked the best for both problems, which can be justified by our numerous features. Given the results, the above mentioned approaches are valid to determine the respective outcomes,

Related Work

Past approaches, including Torres’ ‘Prediction of NBA games based on Machine Learning Methods’, used metrics such as point differentials and win-loss percentage to predict specific matchup outcomes given past matchup history. We drew inspiration in how they utilize models such as Linear Regression and Random Forest to forecast results and applied it to our metrics. Furthermore, Torres mentioned that there are far more detailed statistics to be considered, which prompted our inspiration to find and include advanced metrics in our model. As regular participants in NBA Fantasy leagues, we realized that NBA Fantasy Points that are assigned to players post-game (exact formula detailed later) can be useful to determine player impact and sports analysts at ESPN developed this metric.

Data Sets

The first dataset we used was from nba_api, a well organized python library that pulls data from nba.com. Using this library, we were able to obtain 3 DataFrames that serve different purposes to tackle both our objectives. The first DataFrame (**Figure 1**) consists of all NBA players' season performance, specifically their season box score, from 2012-2020. While nba_api supports data all the way from 1989, we chose to analyze data starting from 2012 because it best represents modern basketball; numerous rule and playstyle changes have taken place since 1989, and 2012 marks the beginning of the modern NBA era. The biggest struggle to collect this data was to append season and team name to each player, we were able to do this accurately by manipulating the DataFrame within the for loop. Otherwise, since this library is updated consistently by the creator, we didn't have to do much cleaning. In terms of the features in this dataset, basic box scores such as field goal made (FGM) and assists (AST) are included, which are simple indicators to assess player performance. The second DataFrame (**Figure 2**) consists of teams' records (W_PCT) throughout the seasons. The significance of this DataFrame is that it includes our dependent variable, the win percentage of each team. Our third DataFrame (**Figure 3**) captures specific matchup results, important features are the matchup (Team A vs Team B) and the corresponding box score of Team A.

PLAYER_NAME	GP	W	L	W_PCT	MIN	FGM	FGA	FG_PCT
Al Horford	74	42	32	0.568	2756.445000	576	1060	0.543
Anthony Morrow	24	17	7	0.708	300.893333	47	111	0.423
Anthony Tolliver	62	28	34	0.452	963.160000	82	216	0.380
Dahntay Jones	28	13	15	0.464	381.176667	32	82	0.390
DeShawn Stevenson	56	30	26	0.536	1157.818333	98	262	0.374

Figure 1: (4513, 64) Player season performance, reflected by box scores

TEAM_ID	TEAM_NAME	GROUP_VALUE	GP	W	L	W_PCT
1610612737	Atlanta Hawks	2012-13	82	44	38	0.537
1610612737	Atlanta Hawks	2013-14	82	38	44	0.463
1610612737	Atlanta Hawks	2014-15	82	60	22	0.732
1610612737	Atlanta Hawks	2015-16	82	48	34	0.585
1610612737	Atlanta Hawks	2016-17	82	43	39	0.524

Figure 2: (240, 57) Teams' season records

Team_ID	Game_ID	GAME_DATE	MATCHUP	WL	W	L	W_PCT
1610612737	21900969	MAR 11, 2020	ATL vs. NYK	L	20.0	47.0	0.299
1610612737	21900957	MAR 09, 2020	ATL vs. CHA	W	20.0	46.0	0.303
1610612737	21900943	MAR 07, 2020	ATL @ MEM	L	19.0	46.0	0.292
1610612737	21900930	MAR 06, 2020	ATL @ WAS	L	19.0	45.0	0.297
1610612737	21900905	MAR 02, 2020	ATL vs. MEM	L	19.0	44.0	0.302

Figure 3: (120238, 28) Matchup box scores and results

The second dataset was from basketball_reference.com, which features all players' advanced statistics to reflect their season performance. Advanced statistics are metrics that measure a player's performance based on formulas. For example, the advanced statistic per-minute performance (PER) adds positive box statistics such as field goals (FGM) and free throws (FTM), and subtracts negative box statistics such as turnovers (TOV) and personal fouls (PF) for a score that indicates a player's overall season performance. **Figure 4** shows this DataFrame, as you can see, the player names have slashes appended at the back and some player names have accented characters, to join this data with nba_api DataFrame, we used difflib, a library that matches strings based on character similarity, to match the player names precisely.

Player	Pos	Age	Tm	G	MP	PER
Quincy Acy\acyqu01	PF	22	Toronto Raptors	29	342	15.9
Jeff Adrien\adrieje01	PF	26	Charlotte Hornets	52	713	13.4
Arron Afflalo\afflaar01	SF	27	Orlando Magic	64	2307	13.0
Josh Akognon\akognjo01	PG	26	Dallas Mavericks	3	9	15.3
Cole Aldrich\aldrico01	C	24	TOT	45	388	11.1

Figure 4: (5004, 32) Player season performance, reflected by advanced statistics

Ultimately, to join datasets from nba_api and basketball_reference to get player season performance of box scores and advanced statistics, we joined by player name and season. Then, we passed in this joined DataFrame and **Figure 2** to the expand_projected_team_stats() function that takes a given season roster's season-1 performance, weighted by minutes played (MP), to match with the season team record, shown in **Figure 5**. If you observe the first row, you can see the 2013-14 Hawks win percentage, as well as that roster's 2012-13 player win

percentage (PLAYER_W_PCT_2012-13), one of the many features to predict win percentage. This is the final dataset passed into our machine learning algorithms.

TEAM_NAME	GROUP_VALUE	W	L	W_PCT	PLAYER_W_PCT_2012-13
Atlanta Hawks	2013-14	38	44	0.463	0.507648
Atlanta Hawks	2014-15	60	22	0.732	0.556594
Atlanta Hawks	2015-16	48	34	0.585	0.590055
Atlanta Hawks	2016-17	43	39	0.524	0.531541
Atlanta Hawks	2017-18	24	58	0.293	0.494432

Figure 5: Final joined DataFrame (210, 322)

By plotting scatter plots and heatmaps with **Figure 5**, we were able to visualize the effectiveness of features such as VORP (points per possession) and PER on determining season win percentage. Furthermore, we recognized features that are negatively correlated to win percentage, such as offensive rebounds (ORB) which will also be useful to include during the modeling stage. **Figure 6** shows the positive correlation between PER and W_PCT for the 2018-19 rosters and tells the story that teams with more high performing players tend to get a better record, with some exceptions such as the Toronto Raptors (TOR), a case of overperformance. Perhaps the most telling visualization is a heatmap of all the variables with win percentages throughout the seasons **(Figure 7)**, the impact of each variable on season win percentage can be interpreted by seeing how consistently red or green a particular column is. The heatmap reveals that VORP and WS (number of wins a player produces for his team) contribute heavily to win percentage while ORB brings down win percentage, which makes intuitive sense as offensive rebounds are results of missed shots, revealing low field goal percentage and questionable shot selection.

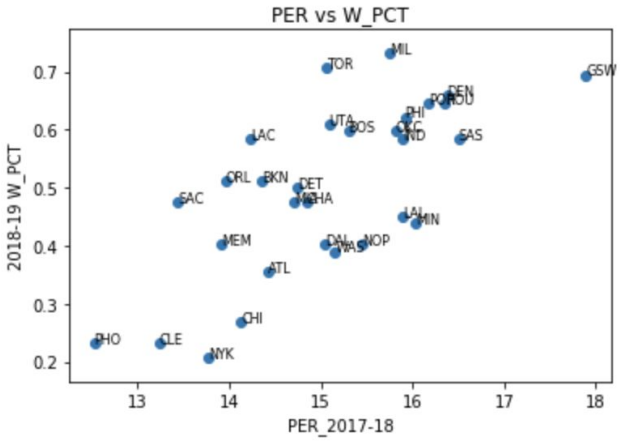

Figure 6: PER 2017-18 vs 2018-19 W_PCT scatterplot

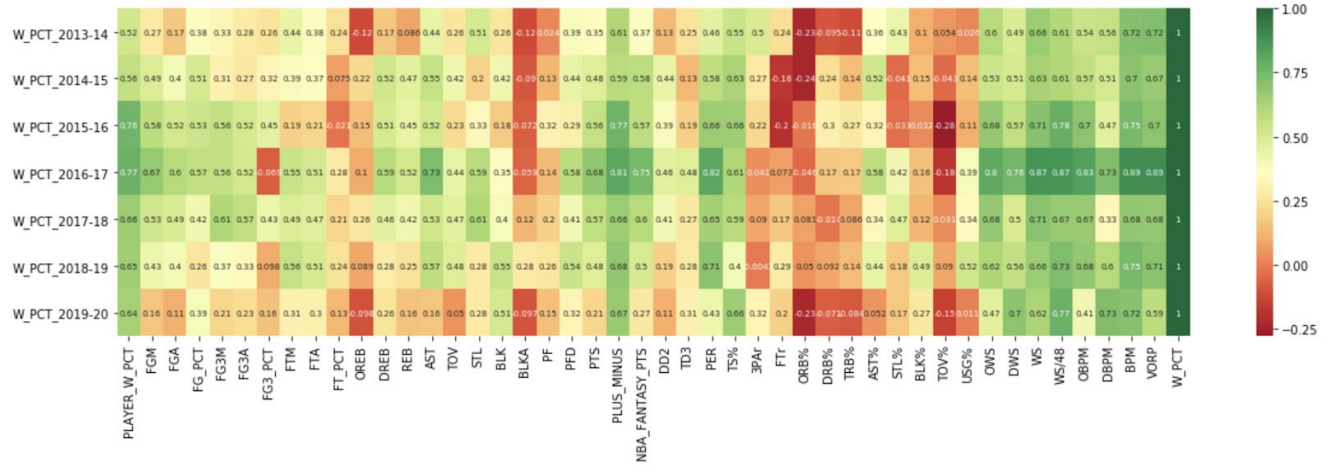

Figure 7: Heatmap of all variables and W_PCT throughout the seasons

Overall Technical Approach

Our overall technical approach was split into two camps. On one hand, we tried to predict what percent of games a single team would win in any given season. On the other hand, we attempted to predict the winning team given any matchup of two teams. However, both processes had the same underlying data preparation and model construction, with differences occurring when feeding in specific variables to each model, shaping the output, and validating our results.

Our general data pipeline is shown in **Figure 8**.

Figure 8: Data Pipeline

Each cell above represents one crucial piece of our data and training pipeline. For the first two cells to the left, we collected data. We had to extract data using API calls set by the NBA API standards. Basketball reference had a raw csv file that we could acquire and later join the NBA API data.

Our second step was joining the player data from both datasets. We joined based on the player's name and season stats. We had to preprocess much of the basketball reference data in order to get the players names to line up with the format of the NBA API. We utilized the difflib

package to cleanly format the player names and then join most player stats for every season a player participated in.

With the merged player dataset, we conducted a huge amount of preprocessing in order to “flatten” the dataset so that we could cleanly feed it into our model. The problem was that we had way too many rows of data on players and not enough data on the teams themselves to predict off of. So, we chose to “simulate” teams by having a function that takes a list of active NBA players and looks at those players prior performances, then aggregates each player metric into one metric representing the weighted average of the players’ performances. For example, if we had a list of 6 players and we wanted an aggregate free throw success percentage for these players, then we would look at the prior season that each of these players played in, regardless of what team they were on, then record how they played that season. Then, we would take the average free throw success percentage and weight that percentage by the amount of time that the player played on the cour in minutes during the prior season. We did this so that players who did not play very much on the court did not have the same impact to an overall metric as another player who plays almost all the time on the court. In **Figure 9** below, we illustrate how we took any player regardless of what team they were on and got the weighted average of each measured player metric for each prior season.

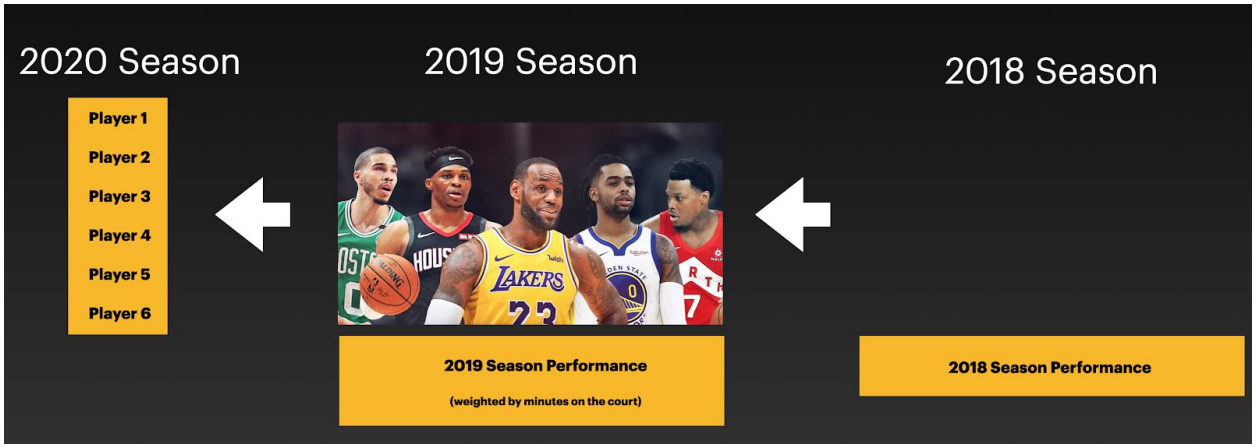

Figure 9: Player Weighted Simulations / Metric Aggregations

Once we had our aggregated data for each season, we trained the data on all prior seasons to predict the current season of matchup winners and team season win percentages. We will delve more into the details of how we did this and measured our success in the Experiments and Evaluation Section.

To train our model, we used a random forest classifier and regressor for our win/loss classification target and win percentage target respectively. We chose to use random forest since it excels at considering many features to predict a target. Our training dataset had hundreds of variables since we replicated each type of player stat for each and every season aggregated against all players. This proved to be much better than our other attempts with logistic regression, multiple linear regression, and the Bradley-Terry model.

Once our model was trained, we recorded our metrics, which we cover in the evaluation section, then saved the model as a pickle file. We then use this pickle file to quickly boot up and predict the season win percentage of a team given the team and season, and predict which of two teams will win a match.

Software

For our project we interacted heavily with the NBA API, which is built upon Python. The API was very well documented and very consistent across all endpoints so we don't have any complaints with using this software besides the api call limits. We had to take breaks between multiple calls in order to not get kicked from requesting data from the API. We utilized Jupyter Notebook to test small snippets of code and alter the training of our models without having to rerun everything. We used the difflib library to reformat the format of names in the basketball reference player data, which was extremely useful in joining our external dataset. We also used Scikit-learn for all of our model training and validation. We utilized Scikit-learn's random forest classifier and regressor as well as a few other models when testing to see which type of model would best fit our data that had many features. We used the pickle package to save our models and metrics output for easier reading and production level models in the future.

Experiments and Evaluation

To evaluate our models, we chose to feed in data rather than a traditional train/test split. We could not do a traditional split since random data from different seasons would confuse the model and possibly leak present or future information into the training dataset that we would not know if we were making a prediction during a season. We measured the accuracy and error of our models by looking at how well it performed by season. So for each season, we would use all prior seasons as training data and the current season as the testing dataset. Each season had 30 teams, so there would always be about 30 entries in the testing dataset. An illustration of how we "fed" previous seasons worth of training data is shown in **Figure 10** below. Each bar represents an aggregate variable for that team in a given season. There were about 30 entries per bar representing each team during a single season.

Our results showed that we were learning off prior data since we achieved an accuracy above 50% in both classification models. We knew that our baseline accuracy would be 50% since a model that just guesses would achieve this. When predicting whether a team will have a win percent that is above or below 50%, we achieved 60-73% accuracy, depending on the season being validated. When predicting the raw win percent which falls between 0 and 1, we achieved a mean absolute error of about 0.1, meaning our model can predict a team's win percentage within 10% of their actual win percentage, on average. A full breakdown of these accuracies are shown in the bar plots in **Figure 11** below.

For our most lofty goal, which was predicting which team would win given two teams and a season, we could only measure accuracy since we either predicted the correct winner right or wrong. We achieved modest results with an average accuracy of 68%, meaning regardless of the season being predicted, we can predict which team will beat the other given two teams. For a full breakdown of the accuracies by season for tackling this prediction problem, refer to **Figure 12** below.

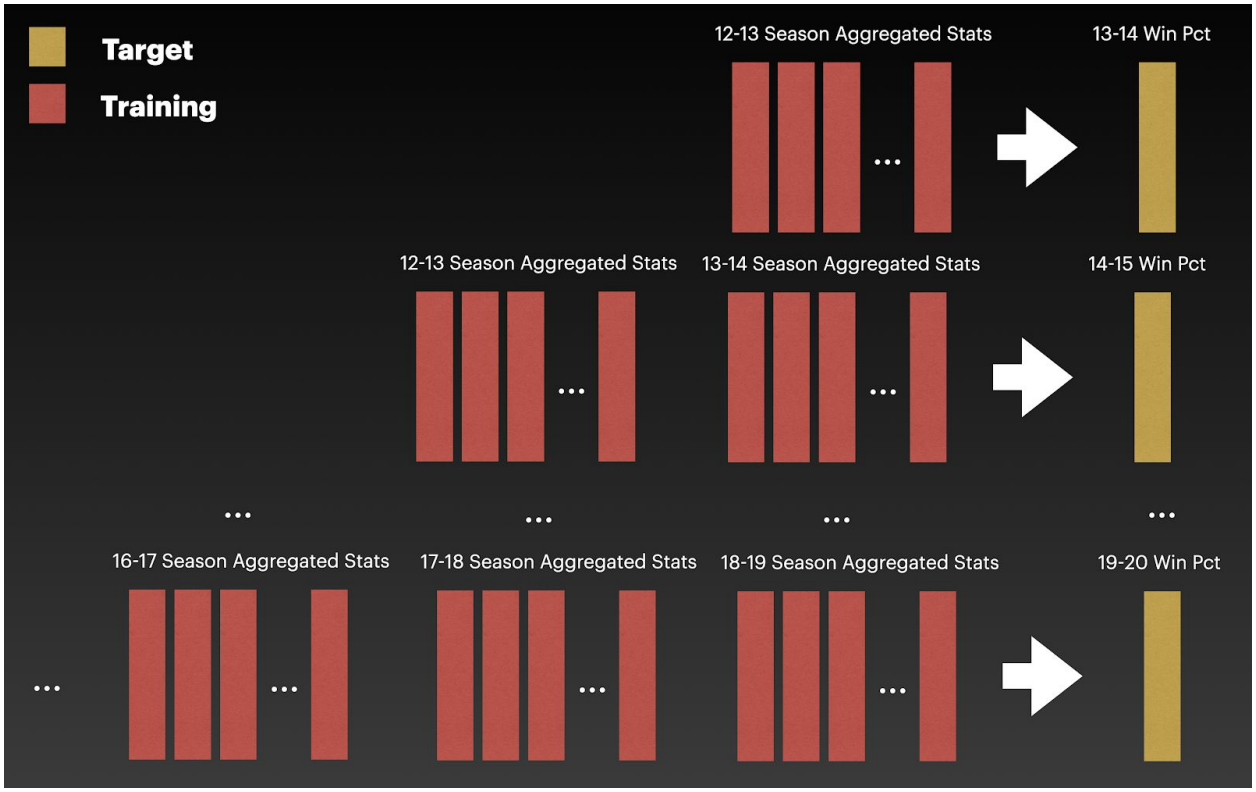

Figure 10: Training and Testing Methodology

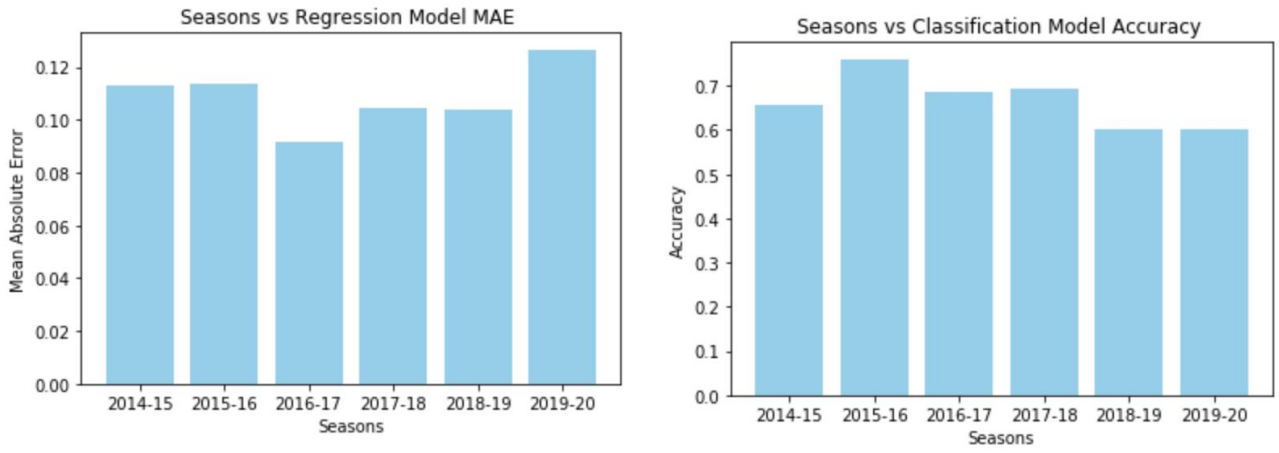

Figure 11: MAE and Accuracy for predicting raw win percent and win percent group by season

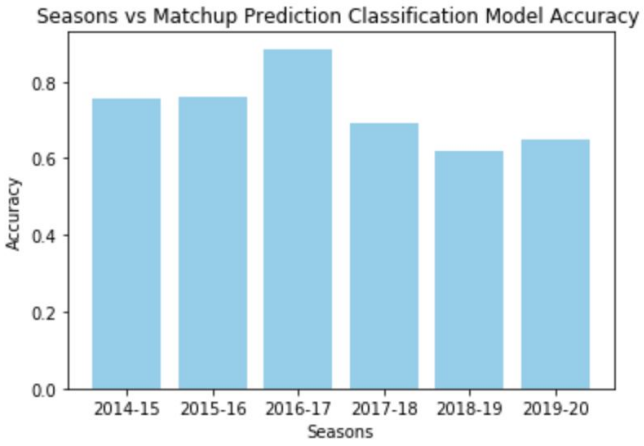

Figure 12: Classification accuracy of team matchup prediction by season

Notebook Description

We split our notebooks into two sections. **nba_collection_eda.ipynb** is for pulling the data, saving it to csv, and doing cleaning on the merged dataset. It also has EDA on the player and team dataset as well as plots showing the final metrics from our models trained in the other notebook. **model_collection.ipynb** reads in all the data generated from the other notebook and fits the training data. The model metrics and the models themselves are saved to a pickle file for later use. We also have a data folder that houses all data and pickled model metrics and model objects.

Authors Participation

Ray focused on streamlining the data collection from the NBA API and basketball reference, then merging these two datasets into a player and team dataset. He then focused on EDA an initial data cleaning methods. Caleb focused on machine learning preprocessing of the data by weighting individual player performance by season. Caleb then validated the models and saved the models and output to pickle files for future use. We both spent an equal amount of time on the project proposal, progress reports, and final report.

Collecting Team Stats	Ray Yi, Caleb Pitts
Collecting Player Stats (primary and secondary sources)	Ray Yi
Feature Engineering, Cleaning	Caleb Pitts
Descriptive Statistics	Ray Yi (50%), Caleb Pitts (50%)
Variable Exploration, EDA	Ray Yi
Random Forest Model, Validation	Caleb Pitts

Powerpoint Presentation	Ray Yi (50%), Caleb Pitts (50%)
ML Model Visualizations	Caleb Pitts
First Draft Report	Ray Yi (50%), Caleb Pitts (50%)

Conclusion and Final Discussion

We ended up with 65% accuracy to predict team record given the team roster's previous season performance and 74% accuracy to predict specific matchup outcomes, indicating that our approach to tackle these problems works. With NBA teams placing an emphasis on in-game analytics, game statistics have been documented with great detail and organization, allowing us to pull from metrics that are previously unavailable, such as Player Points per Possession (PER) and Win Shares (WS). These advanced statistics are more impactful indicators of W_PCT than standard box scores, which makes sense as they are derived from formulas composed of box score statistics anyway. Random Forest consistently outperformed Multiple Linear Regression for both of our objectives, which reveals its strength in handling datasets with numerous features. While Multiple Linear Regression is fast to implement, it struggles when features that are positively correlated to one another incorporated together.

The biggest hurdle is definitely the unconventional nature of our machine learning problem. To predict the current season team record, we had to use the current roster's individual performance last season from their respective teams, which led to extra functions to test and debug, whereas a traditional problem would have independent variables ready to predict dependent variables. Our biggest takeaway is that the data wrangling process to shape the data to its final form takes up the most time; considerations to weigh the features and ensuring that the data is accurate are time-consuming. If I were in charge of the lab, predicting player injury would be a helpful insight to have, and there would be an emphasis on studying players' health history. Each player's health report would need to be made public, and once we're able to predict player injury potential, we can then assess the player's impact on a team with a new perspective. Considering the effectiveness of advanced statistics, it's also worthwhile to look into deriving new formulas to gauge player performance.

Appendix/References

Torres, R. *Prediction of NBA games based on Machine Learning Methods*, December, 2013.

"Basketball Statistics and History." *Basketball*, www.basketball-reference.com/.

Jee, Ken, et al. "How to Get NBA Data Using the nba_api Python Module (Beginner)." *Playing Numbers*, 23 Dec. 2019, www.playingnumbers.com/2019/12/how-to-get-nba-data-using-the-nba_api-python-module-beginner/.