**kNN Regression Predicting Win Totals of NBA Teams**

*Jamie Luna, Jeremy Kozlowski, Liam Quach, Yaniv Sagy*

## Overview

Our team employed a k-nearest neighbors Regression algorithm on NBA player data to predict team performance. We utilized a dataset spanning the 1972 to 2022 seasons, with each season comprising 82 games. To identify the seven most influential players from each team, we selected the seven players with the greatest average minutes played per game. The starting players, who play the most minutes, are the most influential members of the team, along with two bench players. To evaluate a player's effectiveness both offensively and defensively, we averaged the following attributes over the best seven players per team: field goal percentage, 3-point percentage, free throw percentage, rebounds per game, assists per game, steals per game, blocks per game, turnovers per game, and points per game.

To identify teams with similar multidimensional characteristics, we plotted these data points and computed the win totals of teams based on the hyperparameter k. The k value determines the number of nearest points to consider. By averaging the win totals of the nearest teams, we could predict a team's performance.

## The Process

To start with, we had to clean the data to make it usable. We did this with R and Python to easily average the statistics and extract the statistics we felt were most valuable to a player. Our main algorithm, k-nearest neighbors, was implemented in Java. We first read the input file to populate multiple hashmaps to assign data to teams and represent the data as a whole. This data was then normalized based on the season the data point is from and then standardized. We then create our training and testing data randomly by selecting 80% of the data to be our training data and 20% of our data to be testing data. This was done randomly to not let the evolution of basketball potentially impact our results. The main problem we are trying to solve is

determining k. We did this using 10-fold cross validation on our training set in an attempt to narrow down k. This method breaks the training data into ten nearly equally sized chunks, with one acting as the testing set. This allows us to find the average error for a chunk, across ten chunks, for a single k value. After comparing the average error for numerous k values, we can find the smallest error and use that value as k on our testing set to get the results.

**Results**

After performing our algorithm from k = 1 to k = 30 on our training set with cross-validation, we determined that k = 1 has the lowest average error of 5.92. After running our algorithm on the testing set, we received an average error of 11.84. As seen in our error, our model was not very proficient at predicting the win totals of NBA teams. This could be for a variety of reasons. One reason could be that our assumptions of the data were incorrect. The best seven players with the most minutes per game might not be the top seven players that impact the game. The statistics that we chose to focus on may not be an accurate representation on how proficient a player is for their team. These are only a few of the many assumptions we made that could have affected our results. Another reason could be that k-nearest neighbors was not the most appropriate algorithm to choose for our data. While these results were not perfect, we received useful insight into the k-nearest neighbors algorithm and how to implement many machine learning concepts such as cross-validation, training and testing set separation, and result interpretation.