Defining NBA Player Offensive Roles through Machine Learning and SportVU Tracking Data

Edgar Jaramillo Rodriguez, University of California Davis

1 Introduction

Beginning in 2012-13 season the NBA began a partnership with STATS to provide SportVU tracking data for their games. SportVU is a six-camera system developed by STATS that captures the real-time positions of players and the ball at a rate of 25 frames per second. Since the 2013-2014 SportVU camera system has been installed and in operation in every game in all 30 NBA arenas. While the complete tracking data is not publicly available, beginning in 2016 the NBA has released player metrics gathered using the new technology. This data, available on `stats.nba.com`, contains the rate at which players attempt different types of plays per game and the outcomes of those attempts. For example, one can find the number of drives a given player attempted per game, how many assists they generated from drives, how many fouls they drew, amongst other things.

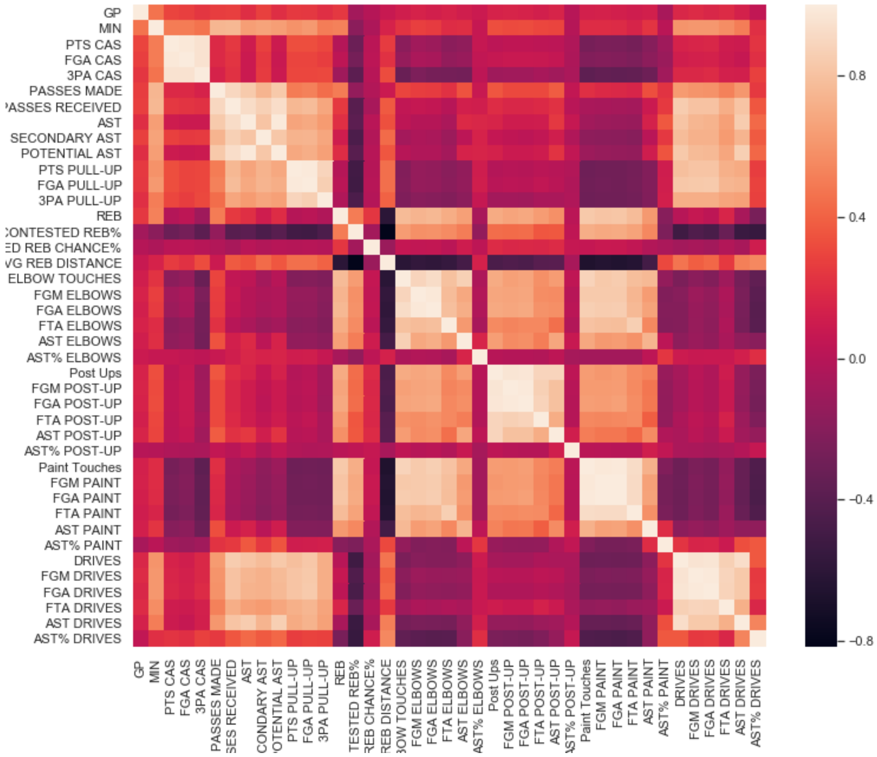
I believe this relatively understudied data set can provide new insights into the different roles players have across the league, going far beyond what we can achieve using only traditional box scores and advanced metrics. Therefore, my goal in this project is to use the data and unsupervised learning techniques to define these new roles and hopefully gain a better understanding of the league and the relative value added by different players. I should note that one limitation of this data is that the bulk of the available metrics are focused on the offensive end of the floor. For that reason, I will focus on only classifying the different *offensive* roles of NBA players, without paying attention to their impact on the defensive end.

2 Data Preparation

I scraped the tracking data for the most recent 2018-2019 season from `stats.nba.com` using `Selenium` for Python. For my analysis I chose to look at the tracking data on: drives, paint touches, post-ups, elbow touches, catch and shoot attempts, pull-ups, passing, and rebounding. Within each of these categories I pulled a subset of the available statistics, trying to remove redundant entries. For example, within the "rebounding" category one can find the Field Goal Attempts (FGA), Field Goals Made (FGM), and Field Goal Percentage (FG%) resulting from drives. However, FG% := FGM/FGA, so including all three statistics could result in us over emphasizing this feature when performing our analysis.

To identify other potentially redundant attributes I also computed the data's correlation matrix and plotted it as a heatmap using the `Seaborn` Python library. In the end I was left with 41 attributes, whose correlation matrix is displayed below.

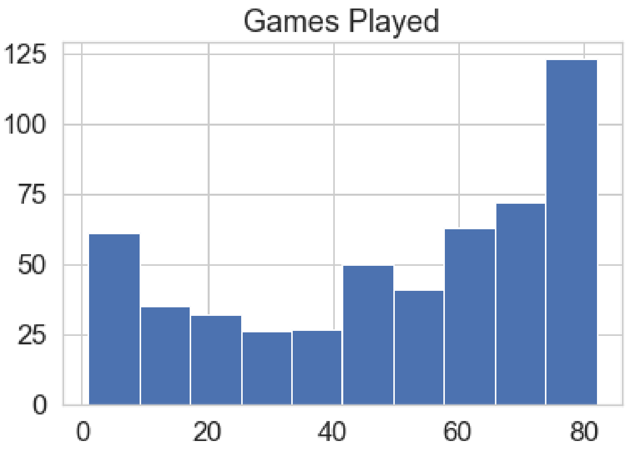
Figure 1: Correlation Heatmap of 41 Final Attributes



As you can see, some variables are still highly correlated, for example 3PA CAS (3 point attempts off catch and shoot shots) and FGA CAS (field goal attempts off catch and shoot shots)[1]. However, because these metrics are capturing sufficiently different phenomena that it is worthwhile to keep them.

The original data set contained the data on all 530 players who played in the NBA during the 2018-2019 regular season. However, many of these players did not play in enough games to provide a sufficiently large sample size for analysis. To determine a cut-off point for the minimum number of games I plotted a histogram of the games played (GP), displayed below. From inspection, there is a natural cutoff point at the 40 game mark, roughly half of the 82 game season.

Figure 2: Games Played Histogram



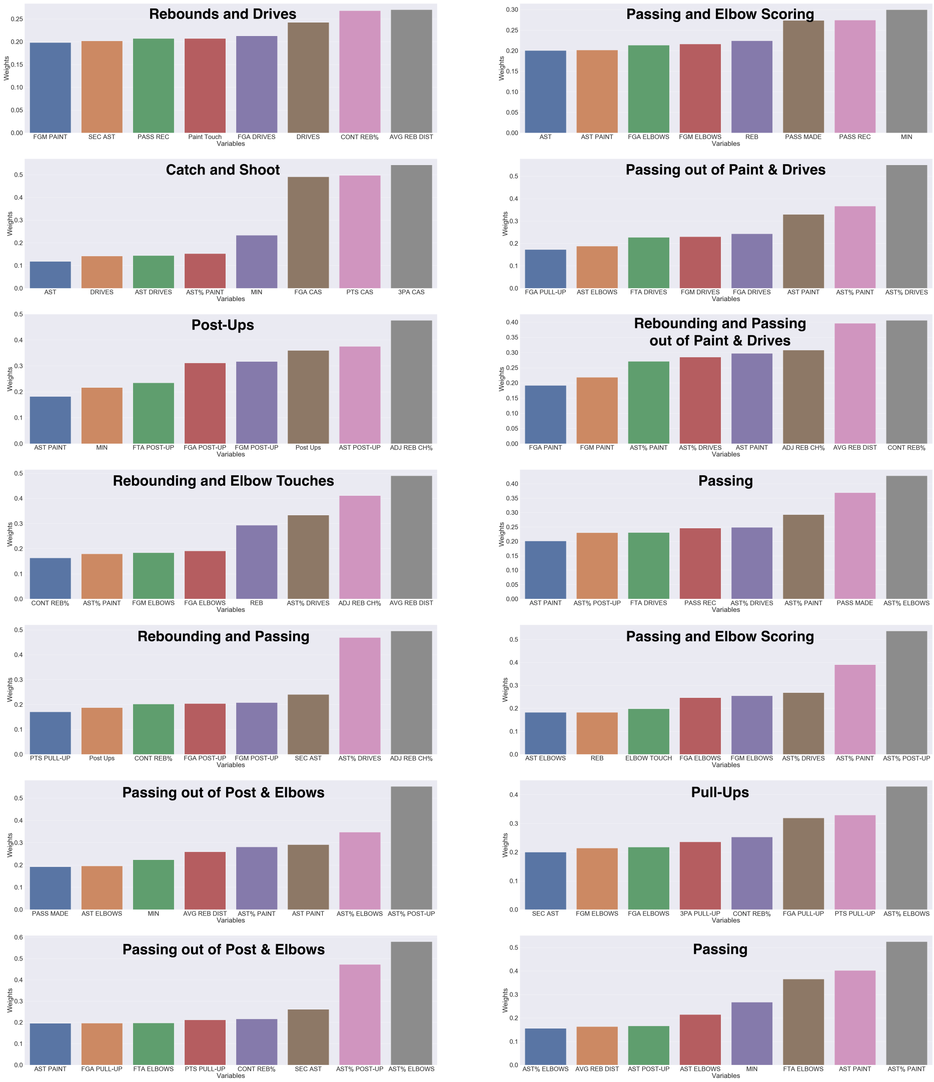
[1] For a complete glossary of the variable names refer to https://stats.nba.com/help/glossary/

After performing this cutoff I was left with 353 eligible players. At this point I also decided not to use Games Played as a variable in my analysis, since beyond the 40 games threshold that statistic would mostly just reflect injuries which we do not want to consider when looking at the offensive role of a player. This left me with a data set of 353 players and 40 attributes.

3 Principle Component Analysis

My first goal was to try to understand the data by analyzing its principal components. I used the `sklearn` Python library to normalize the data and compute the principle components. I found that the leading 14 components (ordered by magnitude of the associated singular value) are able to explain 95.2% of the variance in the data. I then looked at the 8 largest coefficients of each these components to see what attributes each component was capturing. Somewhat surprisingly, the largest coefficients of each component often came from one or two of the initial categories (drives, post-ups, etc) our data came from. Below is a bar plot displaying the weights of the 8 largest coefficients of the 14 components, along with labels to highlight the category (or categories) the component is most influenced by.

Figure 3: Top 8 Coefficients of 14 Leading Components from PCA



As we can see, the most prevalent categories are passing and rebounding. My intuition for this is that passing, in particular passing rates from different areas of the court, is a big indicator of player's role on the offensive end. Meanwhile, rebounding statistics inform where a player is on the court even when they don't have the ball and can be a good proxy for height in many cases.

4 Hierarchical Clustering

My main objective with this project was to use unsupervised learning techniques to classify NBA players into different offensive roles. Initially I only planned to do so with the k-means clustering algorithm we covered in class. While this produced good results (more on this in Section 5) in my reading I came across hierarchical clustering techniques and thought they might also work well for this task. Because we did not cover hierarchical clustering in class, in this section I present a brief overview of how a few standard hierarchical clustering algorithms work.

4.1 Overview

Recall k-means looks to solve the following problem: how do we partition data into k clusters, where k is given, so that points in the same cluster are more similar to each other than to points in the other clusters, with respect to some given similarity metric. Hierarchical clustering goes a step further and imposes an ordering on the clusters themselves, so that each cluster is contained in a larger parent cluster and clusters in the same parent cluster are more similar to each other than to clusters in different parent clusters.

There are two basic strategies for hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). In the agglomerative approach, every data point begins as its own cluster, and clusters are iteratively merged to form parent clusters. The pair chosen for merging is the pair with the *smallest* inter-group dissimilarity, calculated by some given measure of dissimilarity known as a linkage measure (more on this in Section 4.2). We repeat this process until all the data has been grouped into a single parent cluster. The divisive approach works in the opposite way. All the data begins in single cluster and is split into two using a standard clustering algorithm such as k-means. At the next step we split one of the new clusters into two, and repeat this process until the data has been split into singleton clusters. The split is chosen so as to produce two new groups with the *largest* inter-group dissimilarity. With both methods there are $N-1$ levels in the final hierarchy, where N is the size of the data set X. Each level represents a particular partition of the data into disjoint clusters. The user must determine which level of the hierarchy is the ideal cut-off point to get a useful or "natural" partition of the data. This is analogous to the dilemma of k-means, which requires the user to specify the number of clusters to seek beforehand.

For this application I will be using the agglomerative approach because there is more literature available on it and because the `scipy` library contains an efficient implementation of this approach.

4.2 Choices of Linkage Measure

In order to perform agglomerative clustering on some finite set X, we must define a measure of dissimilarity, called a linkage, between subsets G and H of X. To do so we work under the assumption that our data points in X live in some normed vector space, so we already have a notion of distances between individual points $x, y \in X$ given by their difference in norm $\|x-y\|$. For our purposes we can assume that our points live in Euclidean space so that the dissimilarity, $D(G, H)$, between G and H will be function of the squared euclidean distances $\|x - y\|^2$ of pairs of points $x, y \in G \cup H$.

The simplest choice of linkage measure, know as Single-Linkage Distance, is defined so

$$D_{SL}(G, H) = \min_{x \in G, y \in H} \|x - y\|^2.$$

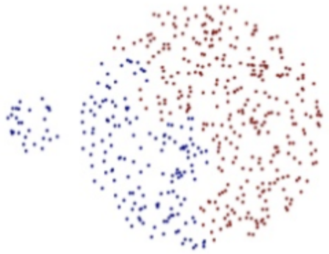
One advantage of Single-Linkage is that is computationally cheap (at least compared to the following). However, because Single Linkage only requires that a single pair of points , $x \in G$ and $y \in H$, be small for two groups G and H to be considered similar, this method has a tendency to combine clusters linked by a series of close intermediate points. This phenomenon, known to as chaining, is often considered a flaw of the method, since we often want to produce convex elliptical clusters.

Another option, known as Complete Linkage, takes the opposite extreme, defining the dissimilarity between G and H to be

$$D_{CL}(G, H) = \max_{x \in G, y \in H} \|x - y\|^2.$$

It is just as computationally cheap as Single Linkage and produces more balanced clusters. However, Complete Linkage does have a tendency to break up large "natural" clusters in an effort to produce clusters with small diameter (the largest distance between pairs of points in a cluster), as in the figure below.

Figure 4: Complete Linkage splitting large natural clusters



Two common compromises are Group Average Linkage and Ward Linkage. Group Average Linkage measures the average distance between pairs of points in G and H, defined by:

$$D_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{x \in G} \sum_{y \in H} \|x - y\|^2,$$

where N_G and N_H denote the number of points in G and H respectively. Meanwhile Ward Linkage measures the difference between the intragroup sum of squares resulting from merging the two clusters minus the sum of the intragroup sum of squares for each the two clusters separately, explicitly defined by:

$$D_W(G, H) = \sum_{x \in G \cup H} \|x - r_{GH}\|^2 - \left(\sum_{x \in G} \|x - r_G\|^2 + \sum_{x \in H} \|x - r_H\|^2 \right),$$

where r_G, r_H, r_{GH} denote the centroids of G, H, and $G \cup H$ respectively.

All four linkages possess the property that the dissimilarity between merged clusters is monotone increasing with the level of the merger, i.e. if C_1, C_2 are the clusters merged at step i an C_3, C_4 are the clusters merged at step $i + 1$ then

$$D(C_1, C_2) \leq D(C_3, C_4).$$

I will prove this fact for the Ward Linkage (the one I use in my application) and leave the rest as an exercise for the reader.

Theorem 1. *Let C_1, C_2 be the clusters merged at stage i of an agglomerative clustering with the Ward linkage, and let C_3, C_4 be the clusters merged at stage $i+1$. Then $D_W(C_1, C_2) \leq D_W(C_3, C_4)$.*

Proof. If neither C_3 nor C_4 equal $C_1 \cup C_2$ then the result follows immediately because we assumed C_1 and C_2 were chosen for merging so as to minimize $D_W(C_i, C_j)$ amongst all available clusters C_i, C_j. Hence WLOG say $C_4 = C_1 \cup C_2$ and suppose by way of contradiction that $D_W(C_1, C_2) > D_W(C_3, C_1 \cup C_2)$. Recall that, by construction, $D_W(C_1, C_2) \leq D_W(C_1, C_3)$, so $D_W(C_1, C_3) > D_W(C_3, C_1 \cup C_2)$. Thus, letting r_* denote the centroid of the union of the corresponding C_i's, we have:

$$\sum_{x \in C_1 \cup C_3} \|x - r_{13}\|^2 - \sum_{x \in C_1} \|x - r_1\|^2 - \sum_{x \in C_3} \|x - r_3\|^2 > \sum_{x \in C_1 \cup C_2 \cup C_3} \|x - r_{132}\|^2 - \sum_{x \in C_1 \cup C_2} \|x - r_{12}\|^2 - \sum_{x \in C_3} \|x - r_3\|^2$$

$$\begin{aligned}
&\implies \sum_{x \in C_1 \cup C_3} \|x - r_{13}\|^2 + \sum_{x \in C_1 \cup C_2} \|x - r_{12}\|^2 - \sum_{x \in C_1} \|x - r_1\|^2 > \sum_{x \in C_1 \cup C_2 \cup C_3} \|x - r_{132}\|^2 \\
&\implies \sum_{x \in C_1 \cup C_3} \|x - r_1\|^2 + \sum_{x \in C_1 \cup C_2} \|x - r_1\|^2 - \sum_{x \in C_1} \|x - r_1\|^2 > \sum_{x \in C_1 \cup C_2 \cup C_3} \|x - r_{132}\|^2 \\
&\implies \sum_{x \in C_1 \cup C_2 \cup C_3} \|x - r_1\|^2 > \sum_{x \in C_1 \cup C_2 \cup C_3} \|x - r_{132}\|^2,
\end{aligned}$$

a contradiction. Hence it must be the case that $D_W(C_1, C_2) \leq D_W(C_3, C_4)$, as desired. □

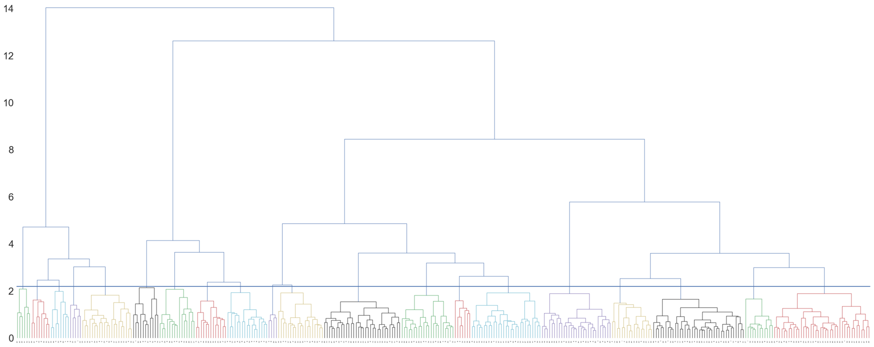
This property allows us to plot the resulting hierarchy as a binary tree so that the height of each node is proportional to the value of the intergroup dissimilarity between its two daughters. The terminal nodes, representing starting singleton clusters, are plotted at height zero. This type of diagram is called a *dendrogram* and provides a easily interpretable description of the hierarchical clustering. In my analysis I used a dendrogram to identify a good cutoff point for the number of clusters.

5 Results

When I was first attempting to cluster the data using only the standard k-means algorithm, my biggest challenge was trying to find the most suitable choice of k as we have no way of knowing how many clusters would lead to a "natural" partition of the data. My solution to this was to first try to cluster the data using an agglomerative clustering technique, and to use those results to inform my choice of k. I used `scipy`'s clustering library to do this, using the Ward Linkage measure described in the previous section. Below is the resulting dendrogram, which provides an accessible summary of the results.

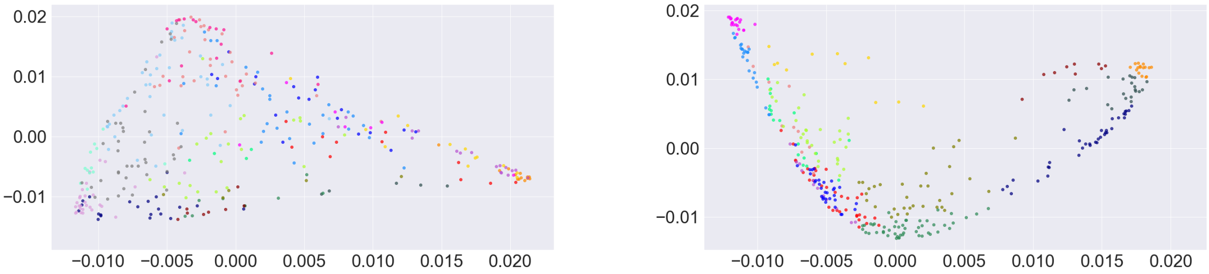
I've added a horizontal line at $y = 2.2$ and set the diagram to color each sub-tree below the line the same color. These colored groups at the bottom represent the 20 clusters we would get using this cut-off. As you can see, the clusters are relatively uniform in size, so I believe this cut-off will provide good results.

Figure 5: Dendrogram from Performing Agglomerative Clustering with Ward Linkage



To further visualize the results I projected the normalized initial data down onto 2 dimensions using a diffusion map and colored each cluster with a unique color. Encouraged by these results I also performed a k-means clustering with $k = 20$. The plots of both the diffusion maps are displayed below.

Figure 6: Cluster Plots after Perfoming Diffusion Map, Hierarchical Clustering left & k-Means right



After inspecting the clusters myself I found that the hierarchical clustering yielded subjectively better results. Their main strength being the nested structure of the clusters which allows us to see explicitly which clusters were more similar using the dendrogram above.

Below is a table with a few representative players from each cluster and a subjective title a I gave to each to help readers who may not be familiar with the NBA. To get a sense of the nesting you can match up each cluster to its position on the dendrogram above (1 appears on the far left, 20 on the far right). I've also highlighted clusters according to their parent cluster when we only look only at the final four levels, to get a sense of the 4 largest categories the players are grouped in.

The clustering algorithm actually does an incredible job of sorting players into different roles in a sensible hierarchy. The blue clusters are compromised of the best "bigs" in the league. The green clusters are compromised of wings and guards who spend a large portion of their time on the court with the ball in their hands. The yellow clusters are compromised of players who mostly play off ball and space the floor through shooting. The red cluster contains all the back-ups and bench level players. The only category that is hard to make sense of is 20, which contains players from multiple positions and of highly variant abilities. I wonder if this somehow became the "leftovers" category that attracted players that were hard to sort.

Table 1: Hierarchical Clustering Results

Cluster	**Cluster Title**	**Size**	**Representative Players**
1	Elite Scoring Bigs with Range	6	Joel Embiid, Nikola Jokic, Karl-Anthony Towns
2	Elite Scoring Bigs without Range	8	Anthony Davis, Jusuf Nurkic, Julius Randle
3	Elite “Roll” Bigs	8	Andre Drummond, Clint Capela, Steven Adams
4	Playmaking Bigs	4	Draymond Green, Al Horford, Thaddeus Young
5	Tier 1 Back-Up Bigs	21	Robin Lopez, Enes Kanter, Ivica Zubac
6	Elite Lead Forwards	11	Kawhi Leonard, Giannis Antetokounmpo
7	Elite Lead Ball Handlers	15	James Harden, Lebron James, Damian Lillard
8	Secondary Ball Handlers	13	Kyle Lowry, Ricky Rubio, Fred VanVleet
9	Lead Ball Handlers	17	Lou Williams, DeAaron Fox, Chris Paul
10	Unique Well Rounded Wings	4	Andre Iguodala, Pat Connaughton
11	Tier 1 Back-Up Ball Handlers	19	Tony Parker, Jeremy Lin, Frank Ntilikina
12	Floor Spacing Wings/ Guards	32	Seth Curry, Avery Bradley, Kyle Korver
13	Floor Spacing Bigs/Wings	22	Brook Lopez, Kyle Kuzma, Nikola Mirotic
14	Elite 3 Point Shooters	7	Klay Thompson, Steph Curry, JJ Redick
15	Tier 1 Jack of All Trades	29	Lonzo Ball, Eric Gordon, Khris Middleton
16	Tier 2 Back Up Bigs	29	Jordan Bell, Nerlens Noel, Tyson Chandler
17	Tier 2 Back Up Ball Handlers	17	Yogi Ferrell, Cameron Payne, Quinn Cook
18	Tier 2 Jack of All Trades	38	Shaun Livingston, George Hill, Tyreke Evans
19	End of the Bench Bigs/Wings	12	Alfonzo McKinnie, TJ Leaf
20	Hard to Fit Bigs/ Wings	40	PJ Tucker, Dirk Nowitzki, Al-Farouq,Aminu, Thon Maker, OG Anunoby

The k-means algorithm produced pretty similar final clusters, but I will omit them because they do not convey the extra information that the hierarchy benefits from.

6 Conclusion and Further Research

Overall I am very happy with the results we got. I think the clusters above are informative and can help both fans and executives better understand the game. In terms of future research, it would be interesting to now analyze the salaries and contracts both as averages of each cluster and within each cluster. This could help us see how executives value different types of players and perhaps help us identify inefficiencies in the market. It would also be interesting, now that we have these labels for each player, to try to use machine learning to predict a player’s role in the NBA based off of their collegiate statistics. This would be of tremendous value to teams prior to the draft. Finally, I would also like to see this same analysis performed but with access to more of SportVU tracking, which might provide better insight into player’s defensive roles and give us a more wholistic classifier.

References

[1] ZACH MCCANN. *Player tracking transforming NBA analytics*. May 19, 2012.
`https://www.espn.com/blog/playbook/tech/post/_/id/492/492`

[2] National Basketball Association Release. *Stats LLC and NBA to make STATS SportVU Player Tracking data available to more fans than ever before*. January 19, 2016.
`https://pr.nba.com/stats-llc-nba-sportvu-player-tracking-data/`

[3] TREVOR HASTIE, ROBERT TIBSHIRANI, JEROMRE FRIEDMAN. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, NY, Second ed. 2017.

[4] VINEET PAULSON. *Hierarchical Clustering Using Python And Scipy*
`https://stepupanalytics.com/hierarchical-clustering-using-python-scipy/`
Steup-Up Analytics. Aug 3, 2018