

Introduction

The National Basketball Association, or NBA for short, is a professional basketball league situated in North America. Widely acknowledged as the best basketball league in the world, the intersection of competition and financial profit (8.3 billion in the 2019-2020 season) has led to NBA teams increasingly turning to analytics to gain an edge. The "analytics revolution" in the NBA has resulted in changes to how teams play basketball, changes to the perceived value of specific skills, and the rise of data and metrics that the traditional box score (a summary of results on a per-game basis) does not encompass.

Overall the incorporation of data analysis has produced positive results. But there is still dispute on how exactly star players in the league can have their impact quantified and ranked. A variety of metrics aim to encapsulate several different data marks into an all-in-one evaluation of a player's performance. Ranking players by these stats can be said to give a rough estimate of how good a player is, especially when compared to their peers. However, different metrics result in a different consensus of a player's impact, making it hard to reach a definitive conclusion as one metric may emphasize certain factors more than others.

Furthermore, the efficacy of all statistics is a debated topic. For example, a survey of independent and team affiliated NBA analysts that ranked offensive and defensive stats on a scale of 1-10 concluded that publicly available defensive stats grade out at 3.6, offensive stats grade out at 7.6, and private defensive stats (available to teams through their own data analytics branches) grade out at 5.6.

One contributing factor is that many stats (or their components) may not always represent the reality of what occurred in an actual game situation. For example, a statistic that notes player A scored a basket against player B records this information without underlying context. Did player B have to cover for a defensive breakdown from another teammate? Did player A get lucky and score despite good defense from player B? These questions and more require careful study of the game, either in real-time or through recordings. The level of effort needed to track these details and the subjectivity involved (what defines "good defense" for instance?) over an 82 game regular season and four rounds of playoffs with over 500 players is far too great to sustain.

Problem Statement, Goals, and Methodology

The problem this research project is attempting to answer is how to measure the impact of a basketball player as it pertains to winning games. To accomplish this, footage of NBA games and play-by-play data must be analyzed to extract meaningful statistics that the traditional box score does not encapsulate. While we cannot go over every single game and player, we can better focus our efforts by analyzing star players, as they provide the most impact in winning. In addition, we can reduce our sample size of games to the playoffs as the outcome of these games directly leads to winning the NBA championship or not.

Our first step is to isolate star players to analyze and track. Going through NBA All-Star lists (participants of the League's All-Star exhibition games designed to showcase star players), we can select the following players:

- Kevin Durant
- Kawhi Leonard
- Giannis Antetokounpo
- Luka Doncic
- Nikola Jokic
- Damian Lillard

The above players have versatile skill sets and participated in the 2021 NBA playoffs, from which most of our data is to be collected. Using a traditional role definition as per NBA voting, this list has a center, forwards, and guards. But players like Joel Embiid and Nikola Jokic, both centers, have entirely distinct playstyles. Similarly, Luka Doncic and Stephen Curry are both classified as guards, but their teams utilize them differently. To properly assess value, we need to redefine the traditional role listing into one that more accurately represents a player's skill set and what they are asked to do. For this, we will need to use publicly available data rather than our own collected data (as a proper redefinition of roles would require analyzing more than just the 12 chosen star players). We can run a k-means clustering algorithm to group together players that are more alike to get a better understanding of their roles. From there, we can assign role definitions to each cluster based on our own sense of what players in a cluster are tasked to do in a game.

There are two types of plays in basketball. Offensive plays, which occur when a team has possession of the basketball and is trying to score, and defensive plays, which occur when a team does not have possession of the ball and is trying to stop their opponents from scoring. For our defensive statistics, we can measure the following:

- **Help defense/Proper rotation:** The application and rotation of weak side defenders helping out the players on the strong side who are defending the ball.
- **Isolation defense:** The ability of a defender to contain their man in an isolation situation.
- **Shot contests:** The motion of contesting a shot by putting a defender's hand in front of the offensive player's shooting motion to bother their release of the ball.
- **Rim protection:** Denying shot attempts at the rim.
- **Perimeter defense:** The ability of a defender to contain their man on the perimeter (beyond the 3 point line).
- **Interior defense:** The ability of a defender to contain their man inside the perimeter.
- **Offball defense:** The ability of a defender to contain their man when they don't have the ball.
- **Pick and roll defense:** The ability of a defender to contain their man in pick and roll situations.

While not covering every possible aspect of defensive basketball, these eight categories provide a good insight into whether or not a player played good defense. Each stat should be measured on a per-possession basis, with a grade of -1 (bad defensive contribution), 0 (neutral/good defensive contribution), and 1 (excellent defensive contribution).

Similarly, our offensive statistics can measure the following:

- **Contested shot-making:** The ability of an offensive player to score against shot contests.
- **Onball scoring:** The ability of an offensive player to score in a possession where they primarily handle the ball.
- **Offball scoring:** The ability of an offensive player to score in a possession where they do not primarily handle the ball.
- **Onball creation:** The ability of an offensive player to create an advantage for others when they have the ball.
- **Offball creation:** The ability of an offensive player to create an advantage for others when they do not have the ball.
- **Isolation scoring:** The ability of an offensive player to score in an isolation situation.
- **Midrange scoring:** The ability of an offensive player to score in the midrange.
- **Perimeter scoring:** The ability of an offensive player to score from the perimeter.
- **Rim finishing:** The ability of an offensive player to score at the rim.

These statistics will be gathered by watching games and tallying what the player we collected data for did.

In addition to these metrics, we can aid our model with certain publicly available stats, such as measures of scoring efficiency and rudimentary player impact stats that can help measure teammate value (A logical conclusion to make is that if a player has better teammates, there is less of a burden on them both defensively and offensively, causing an increase in positive statistical measures).

Next, we can also train a classifier to make predictions on the outcome of a game given a set of data modeled with the same features as our collected statistics. We can accomplish this by using a logistic regression model, random forest classifier, and a support vector machine. After training all models, we can compare them to see which one is more accurate in predicting the outcome of a game.

Lastly, we can create an interactive visual representation of our findings using the Altair visualization library. For instance, we can display the data we have charted in games and how players fit into the new clusters we have discovered.

Conclusion

In the classifiers notebook, a variety of models and methodologies were tested. A logistic regression model, random forest model, and a support vector machine were trained on recorded player data. Each model was evaluated with a simple train test split of 75% training data and 25% testing data. In addition, each model was also evaluated using stratified k-folds to create a more robust benchmark.

Ultimately, the support vector machine (using a linear kernel) model had the highest accuracy, at just over 80% accurate. It is possible that the nature of an SVM being suitable for smaller quantities of data could have made it a more precise model due to the size of our dataset. Recording more player data could prove or disprove this theory. The 80% threshold was achieved with cross-validation, which gives us reasonable confidence that it is indicative of the model's true performance given the dataset.

Role redefinition was a slightly trickier subject. Principal component analysis suggested the optimum role classification would be three categories. NBA theory has arrived at this conclusion, with notable figures such as former Boston Celtics coach (and now General Manager for the Celtics) Brad Stevens stating that player positions can be broken down into ball handlers, wings, and bigs. While accurate, we chose to go with the next best clustering number, 6, to try and gain more insight into NBA positions and roles.
















To this end, the clusters identified were traditional big men, 3+D wings, star scorers, role players, defensive specialists, and bench players. Certain clusters such as role players and bench players are not too informative or help us better understand what a player does. However, other clusters, especially defensive specialists, can potentially deliver a deeper understanding of what a player's skill set may enable them to do on the court. We can see through the Golden State Warriors and Brooklyn Nets usage of players such as Gary Payton II, Deandre' Bembry, and Bruce Brown that such a niche in the NBA can be used in unexpected ways to help not only a team's defense but a team's offense as well.

When looking at defensive and offensive ratings (DRTG and ORTG) in correlation with our recorded stats, we can see that the impact of individual performances are hard to isolate effectively from overall team performance, especially on a per-stat basis. For example, A high ORTG or DRTG for a given game does not necessarily result in a win. Combining all of this information in conjunction with a model allows us to better understand whether or not a player's performance was likely to result in a win, though.

Documentation

<https://github.com/viswanath-kasireddy/NBA-Research-Project> All code and data files can be found here. Programming was done in Python, using the Pandas and Scikit-learn libraries for data manipulation and machine learning analysis. The Altair library was used for data visualization.

The Altair.ipynb file contains the data visualizations created in this project. Simply click on it in the github repository.

 viswanath-kasireddy Update README.md	ed7ca66 34 minutes ago	 19 commits
 AdvancedStats.csv	Add files via upload	4 months ago
 Altair.ipynb	Created using Colaboratory	13 days ago
 Classifiers.ipynb	Created using Colaboratory	13 days ago
 PerGameStats.csv	Add files via upload	4 months ago
 Project Summary-2.pdf	Add files via upload	38 minutes ago
 README.md	Update README.md	34 minutes ago
 Role Redefinition_ Preliminary Thou...	Add files via upload	5 months ago
 RoleRedefinition.ipynb	Created using Colaboratory	4 months ago
 ShootingStats.csv	Add files via upload	4 months ago
 Stat Tracking - Sheet1-2.csv	Add files via upload	2 months ago
 Stat Tracking - Sheet1.csv	Add files via upload	2 months ago
 Visualization Mock Up.pdf	Add files via upload	4 months ago
 total.html	Add files via upload	13 days ago

From there, click on the (blue) google colab button.

2370 lines (2370 sloc) | 711 KB

<> | Raw | Blame

Open in Colab

Standard Imports

```
In [ ]: import altair as alt
import pandas as pd
import numpy as np
import panel as pn
pn.extension('vega')
import matplotlib.pyplot as plot
%matplotlib inline
from math import log2
from sklearn import tree
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
import graphviz
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

Read player tracking data, drop qualitative features

```
In [ ]: df = pd.read_csv('https://raw.githubusercontent.com/viswanath-kasireddy/CS370/main/Stat%20Tracking%20-%20Sheet1-2.csv')
df = df.drop('Player', 1)
df = df.drop('Game', 1)
df = df[:-1]
X = df.drop('Win_Loss', axis = 1)
y = df.Win_Loss
```

Format player tracking data for visualizations

```
In [ ]: from vega_datasets import data
url = "https://raw.githubusercontent.com/viswanath-kasireddy/CS370/main/Stat%20Tracking%20-%20Sheet1-2.csv"
vf = pd.read_csv(url)
vf.dropna()
#vf = vf.drop('Player', 1)
#vf = vf.drop('Game', 1)
vf = vf[...]
```

Once in the colab file, click on runtime, then run all.

Altair.ipynb

File Edit View Insert Runtime Tools Help

RAM 100% Disk 100%

Editing

+ Code + Text

Run all ⌘/Ctrl+F9

Run before ⌘/Ctrl+F8

Run the focused cell ⌘/Ctrl+Enter

Run selection ⌘/Ctrl+Shift+Enter

Run after ⌘/Ctrl+F10

Interrupt execution ⌘/Ctrl+M |

Restart runtime ⌘/Ctrl+M .

Restart and run all

Factory reset runtime

Change runtime type

Manage sessions

View runtime logs

```
[21] import altair as alt
import pandas as pd
import numpy as np
import panel as pn
pn.extension('vega')
import matplotlib.pyplot as plot
%matplotlib inline
from math import log2
from sklearn import tree
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
import graphviz
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

Read player tracking data, drop qualitative features

```
[22] df = pd.read_csv('https://raw.githubusercontent.com/viswanath-kasireddy/CS370/main/Stat%20Tracking%20-%20Sheet1-2.csv')
df = df.drop('Player', 1)
df = df.drop('Game', 1)
df = df[:-1]
X = df.drop('Win_Loss', axis = 1)
y = df.Win_Loss
```

IF University 1:32 PM

Green check marks by the sides of each cell indicate that it has finished running. Scroll to the bottom of the file to view the interactive graph. The first tab contains the visualization of our player clustering. The graph can be scrolled and zoomed in, while hovering over any plotted points gives you more information about a corresponding player.

Cluster definitions are as follows:

Cluster 0: Traditional Big Men.

Cluster 1: 3+D Wings.

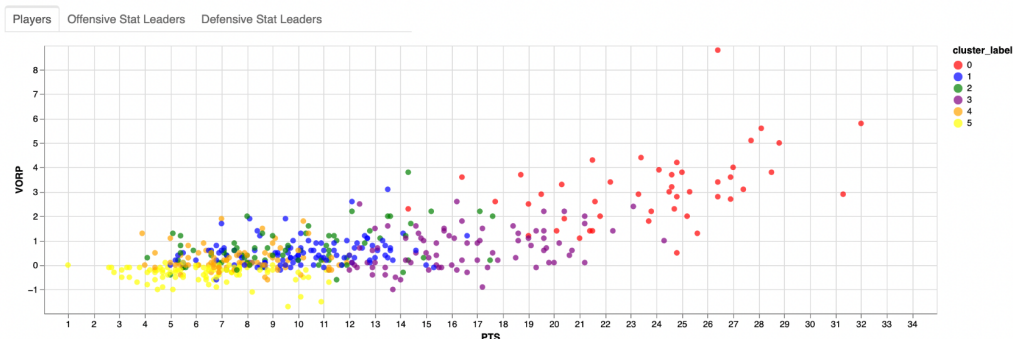
Cluster 2: Star Scorers.

Cluster 3: Role Players.

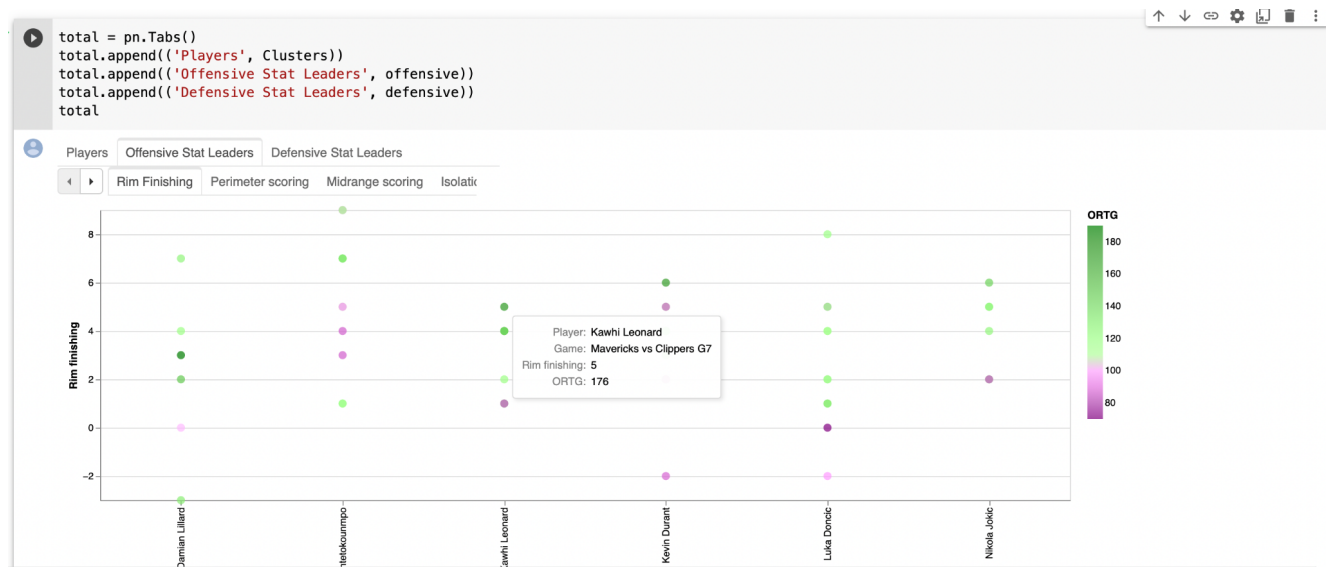
Cluster 4: Defensive Specialists.

Cluster 5: Bench Players.

```
[81] total = pn.Tabs()  
total.append(('Players', Clusters))  
total.append(('Offensive Stat Leaders', offensive))  
total.append(('Defensive Stat Leaders', defensive))  
total
```



Other tabs display offensive and defensive stat visualizations. Click on the arrows to move through graphs within a given tab. As before, hovering over any plotted points will reveal more information about the corresponding player.



Similarly, this process can be repeated for Classifiers.ipynb file. Click on the file in the Github repository, then click on the Google Colab button. Once in Google Colab, run the file to view the accuracy for each model and its configurations. The best performing support vector machine model is located at the bottom of the file, and on average reaches around 80% accuracy.

Support Vector Machine with Stratified K-Folds and linear kernel. Most sound model with around ~80 percent accuracy.

```
[ ] clf2 = RepeatedStratifiedKFold(n_splits=10, n_repeats=40)
n_scores2 = cross_val_score(clf, X, y, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
print('Accuracy: %.3f (%.3f)' % (mean(n_scores2), std(n_scores2)))
```

Accuracy: 0.812 (0.166)