DAT 490 Data Science Capstone
Final Report

Aman Chintada – 1216554378
06/21/2023

Prediction and Clustering on Data from the National Basketball Association

1) Predicting the quality of performance in the playoffs based on regular season performance.
2) Analyzing the similarity in quality of performance in the draft combine and the NBA.

Table of Contents

ABSTRACT 3
PROJECT PLAN 4
LITERATURE REVIEW 10
FINAL RESEARCH QUESTIONS 12
EXPLORATORY DATA ANALYSIS 14
METHODOLOGY 22
ANALYSIS 23
DATA VISUALIZATIONS 31
ETHICAL RECOMMENDATIONS 43
CHALLENGES 44
RECOMMENDATIONS AND NEXT STEPS 45
SOURCE CODE 46
REFERENCES 47

ABSTRACT

Being a passionate fan of the game of basketball, the purpose behind this project was to work with data from the National Basketball Association to build a model which predicts a player's performance in the playoffs given their performance in the regular season. Having started this capstone in the midst of the 2023 NBA playoffs, it was fascinating to witness players having all-time great performances in key playoff moments, when they did not have a remarkable performance in the regular season. On the other hand, there are those players who perform consistently well in the regular season and carry on the same level of play to the playoffs. I've always been fascinated by players who 'turn on' in the big moments, where winning matters the most. This fascination fueled the curiosity behind my first research question (**RQ1**), which was to build a regressor to predict playoff performance based on a player's regular season statistics. The first step towards approaching this question was to define 'playoff performance'.

During an extensive literature review, a deep dive of 'all-encompassing statistics' was performed to identify one of them to use for the project. To put it simply, all-encompassing statistics in basketball are metrics which can evaluate a player's performance with a single number. The all-encompassing stat chosen for this project is wins above replacement or WAR, which describes the number of wins a player contributes to their team if they were replaced with an average player. WAR data for this project was sourced from FiveThirtyEight – a website part of abc news, that is known for sports blogging using statistical analytics. The two regressors built as a part of **RQ1** are trained to predict WAR for the playoffs given their regular season statistics as input. All the regular season data was sourced from the NBA statistics webpage using a 'nba_api' API client for 'www.nba.com'. This python package makes it easy to access all regular season data for any given player using the function calls with the necessary parameters. A multiple linear regressor and a random forest regressor were built and trained on select features of regular season data based on the combination of features produced the highest R-Squared score. The random forest regressor performs substantially better on the training data, as it seems to do fit the more complex (and at times, non-linear) relationships between the independent and target variables. However, both models produce similar mean squared error, mean absolute error, and R-Squared scores (about 0.22 for the MLR and 0.27 for the RFR) on a hold-out/test dataset which consists of regular season stats from the latest 2022-23 season.

In the second part of this project (**RQ2**), I wanted to explore similarities in draft combine performance and rookie season performances in the NBA. The NBA draft combine is a showcase of upcoming NBA talent that spans over a few days before the NBA draft. The combine features tests of skill, athleticism, body measures, and level of play. Using the 'nba_api' package as well as RAPTOR data (another all-encompassing statistic from FiveThirtyEight), draft combine drill and anthropometric statistics were collected for all participants from the years 2014-2022. Additionally, their corresponding performance in their first season in the NBA was collected in the form of RAPTOR stats (describe their performance as a combination of box-score and player-tracking data). After collecting and cleaning the data, a K-Means Clustering approach is used to group similar performances in the draft as well as in the NBA. Clustering was performed in two different contexts to produce two different types of labels. The first type of clustering performed was on the draft statistics (similarity in performance amongst several features – wingspan, weight, height, etc. for anthropometrics and vertical jump height, lane agility, bench press, etc. for drills). The second set of labels were assigned based on clustering on offensive and defensive RAPTOR statistics from players' first season in the NBA. Since this aspect of the project was analytical and not predictive, results can be viewed as plots, distributions, and insights, described in great detail in the analysis section of this project. In short, it is noticed that there does not seem to be a definitive relationship between players that perform similarly in the draft and players that have similar RAPTOR scores in their first season in the NBA.

This project certainly opens up pathways to further analysis and predictions by revealing insights into the relationships between different variables, and how their complexity can be captured more efficiently to produce more accurate models. For example, in **RQ1** using a random forest regressor was a step-up to a multiple linear regressor, as it was able to model these relationships more accurately. I hope to continue to improve upon the results of this project, to build a more accurate predictor of playoff performance.

PROJECT PLAN

Profile of the National Basketball Association

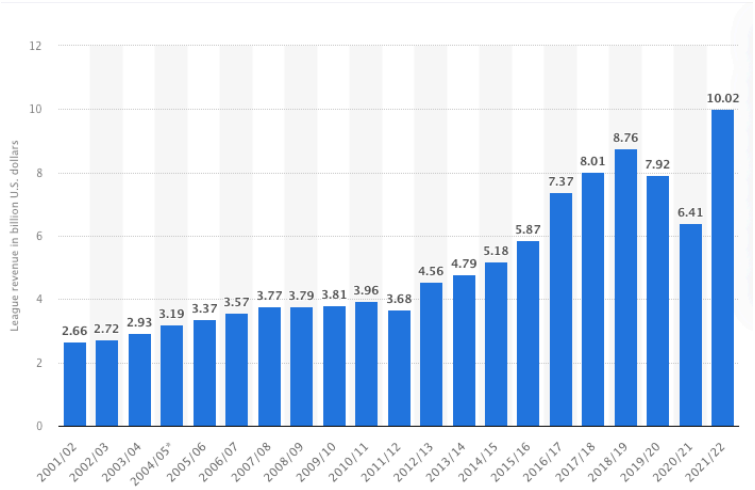
The NBA was founded on the 6th of June 1946, in New York City. The league was born out of a merger between the pre-existing National Basketball League and the Basketball Association of America (started by Walter Brown). It is a private company with its headquarters in New York, 645 Fifth Avenue. The current commissioner of the NBA is Adam Silver

Being one of the biggest sports leagues in the world, the NBA caters to a world-wide population of basketball fans. Consisting of 30 teams, which represent about 21 states in the US (including one from Canada). The association drafts players out of college, and from across the world, making it the most competitive basketball league in the world. 30 teams play through a grueling 82 game regular season, which then leads to 16 teams making the playoffs (or the post-season). Teams then play each other in a seven game series in 4 rounds to get their hands on the championship.

As per an article by Investopedia, the NBA makes most of its income through ticket purchases, TV and media deals, viewer subscriptions, apparel, and fines imposed on various teams and players throughout the season. The association also has other leagues such as the G-League (a preparatory league for the NBA), the WNBA (Women's NBA), the NBA 2K League (e-sports), and the BAL (Basketball Africa League) (Nath, 2022). For this project, only data on the NBA will be used.

Financials

Here is a graphic from Statista displaying the annual revenue in the last 22 years (Gough, 2023):



As seen in the graph above, the NBA crossed a revenue of just over $10 Billion for the first time. Each NBA team has its own set of assets and salary caps. A detailed list of team assets can be found in the Forbes article by Mike Ozanian and Justin Teitelbaum (source cited below).

Leadership

With Adam Silver as the commissioner, the other heads of the NBA are:
Mark A. Tatum – Deputy Commissioner and Chief Operating Officer
Kyle Cavanaugh – President, Administration
Michael Bass – Executive VP and Chief Communications Officer

Other members of NBA leadership can be found at: https://careers.nba.com/leadership/

Competition to the NBA

Being one of the biggest sports organizations in the world, the NBA rivals other major sports organizations in the US such as:

- The National Football League
- Major League Baseball
- National Hockey League

Competition could also include other basketball leagues around the world:

- The Euro league (European Basketball League)
- National Basketball League (Australia)
- National Basketball League (New Zealand)
- Other international leagues

Project Incentive and Intention

Winning an NBA championship is a huge matter of pride for the state represented by the team, as well as the fanbase of the team (which may imply an uptick in future ticket sales, season ticket subscriptions, apparel and more). Besides that, teams that make the playoffs and finals also take advantage of the 'player's pool' which is a sum of money divided amongst teams in the post-season. According to MARCA.com, the winning team in the finals took home about $3.8 million.

Outside of individual and team statistics, there a large number of ingredients that go into the creation of a champion. For this project, the intangibles will be set aside momentarily, and individual player performance will be studied. Understanding a player's performance, specifically in the post-season will be a key piece of insight that can empower teams towards a championship. While the regular season is essential for determining whether a team makes the playoffs, there are certain players and even teams that experience a 'step-up' in the quality of performance. As we are currently in the midst of the 2022-23 NBA playoffs, a player that embodies the motivation behind this project is Jimmy Butler of the Miami Heat. Going into the playoffs, the Heat were the 8th seed, which meant that they would play teams that were ranked above them for the rest of playoffs. Stunningly, inspired by several stellar performances by Jimmy Butler, the Heat managed to make their way to the Eastern Conference Finals (one round before the NBA finals). Sourced from ESPN, here is comparison of Jimmy's regular and post-season performance:

Stats See All

STATS	GP	MIN	FG%	3P%	FT%	REB	AST	BLK	STL	PF	TO	PTS
Regular Season	64	33.4	53.9	35.0	85.0	5.9	5.3	0.3	1.8	1.3	1.6	22.9
Postseason	16	39.1	48.8	34.6	81.0	7.0	5.7	0.7	2.0	1.8	2.1	28.5
Career	754	33.2	46.7	32.3	84.2	5.3	4.2	0.5	1.6	1.4	1.6	18.2

Increments in average points, steals, blocks, assists, rebounds, and minutes played is noticed. In comparison to other players in the 2022-23 regular season, Jimmy was not really considered to be a star player. In fact, he did not even make the All-Star team (24 of the league's top players – voted by fans). On other hand, he is considered to be arguably the best player in the playoffs due to his clutch plays and leadership.

Research Questions

Predicting such qualities in players to step up in big moments can be crucial when drafting or trading for players. Coaches and general management staff of NBA teams could use such data to determine the importance of a player to their team. While it is important to have consistent players who perform well in the regular season, key players who step their performance up in the post-season can be the secret ingredient to winning a championship.

RQ1: Can we predict a player's performance in the playoffs based on their performance in the regular season?

Generally, it is seen that players who have stellar performances in the regular season, continue to perform at that level in the playoffs. As per the motivation of this project, a model would seek to predict the performance of those players who may not have had all-star performances in the playoffs. Predicting playoff performance can be incorporated into coaching strategies, as well as trade decisions.

RQ2: Explore the similarity in performance of NBA rookies in the draft combine (anthropometric and combine drills) and their performance in their first season in the NBA.

The NBA draft combine is an event that takes place before the annual NBA draft, which showcases the skills of upcoming NBA talent who have yet to be drafted. The event consists of various anthropometric measures, athletic tests, drills, and games, presented to an audience of head coaches, team managers, scouts, and other members of the coaching staff. At this event, teams can gain more clarity on a player's abilities before going into the draft. Identifying players that perform similarly on the day of the combine and their first year in the league can possibly be beneficial to teams when drafting players. Additionally, exploring this relationship can reveal more on the quality of these tests themselves, and if they do in fact say anything significant about a player.

Hypotheses

H1: Players who consistently perform well in the regular season will perform well in the playoffs.

The model being developed will predict playoff performance based on regular season performance. Players who perform well in the regular season tend to carry out the same level of play in the playoffs, or see a statistically insignificant decrease in performance (discounting situations such as injuries, suspensions, etc.)

H2: Players with prior playoff experience will perform better than players with no playoff experience.

This is a matter of experience in the playoffs, as performance in the playoffs comes with other factors such as pressure, fewer number of games (less room for error), and other intangible factors, that a player may need experience with.

Data

Most of the data for used for this project was sourced from the official NBA website (https://www.nba.com/stats). Using a python package known as 'nba_api' which is an Application Programming Interface (API) client for the 'stats' page on the NBA website, datasets from several endpoints can be accessed by the user. Here is a link to the list of endpoints: https://github.com/swar/nba_api/tree/master/docs/nba_api/stats/endpoints. In this project, the following endpoints will be used:

- draftboard.md
- draftcombinedrillresults.md
- draftcombineplayeranthro.md
- draftcombinestats.md
- leagueleaders.md
- playercompare.md
- playergamelog.md
- playerprofilev2.md
- playercareerstats.md

From the static datasets, players.md and teams.md will be used.

While each endpoint provides access to a variety of datasets based on parameters, the total number of observations for the relevant datasets is 111975. The variables (column names) may change marginally between datasets, but here are the fields in general: PLAYER_ID, RANK, PLAYER, TEAM_ID, TEAM, GP (Games Played), MIN (number of minutes played), FGM (Field Goals Made), FGA (Field Goals Attempted), FG_PCT (Field Goal % - (FGM/FGA)*100), FG3M (3-pointer made), FG3A (3-pointer attempted), FG3_PCT (3-pointer %: (FG3M/FG3A)*100), FTM (Free Throws Made), FTA (Free Throws Attempted), FT_PCT (Free Throw %: (FTM/FTA)*100), OREB (Offensive Rebounds), DREB (Defensive Rebounds), REB (Total Rebounds), AST (Assists), STL (Steals), BLK (Blocks), TOV (Turnovers), PF (Personal Fouls), PTS (Points Scored), EFF (Efficiency: ((PTS + REB + AST + STL + BLK – Missed FG – Missed FT - TO) / GP)), AST_TOV (Assist to Turnover ratio) and STL_TOV (Steal to Turnover ratio).

Breaking down the number of records per research question, there are 30 NBA regular season teams and 16 teams that make the playoffs. Each team has 15 players and NBA data sourced through the nba_api package dates back to about 71 years (earliest data: 1951-52 season) and 45 seasons since the first playoff tounrament. The top 500 and 150 players in each regular season so far will also be considered. For the first question, we have:

$(30 \times 15 \times 71) + (16 \times 15 \times 45) + (500 \times 71) + (150 \times 71) = 88{,}900$ records

Going through the same process for the second research question, we have about 57 players being drafted every season, 23 seasons of draft data available through the package, 65 observations in the 'draftcombinestats' and 'draftcombinedrills' dataset, and 81 records in the 'draftcombineanthro' dataset. This sums up to:

$(57 \times 23) + (2 \times 65 \times 23) + (81 \times 23) + (25 \times 50 \times 15) = 6{,}164$

All RAPTOR data used in this project comes from FiveThirtyEight's GitHub page (https://github.com/fivethirtyeight/data/tree/master/nba-raptor). Two datasets from this page will be used - modern_RAPTOR_by_player (4,685 records) and latest_RAPTOR_by_player (541 records). The first dataset will be used for RAPTOR and wins above replacement statistic for the regular season and playoffs, and the second will be used to attain RAPTOR stats for the latest 2022-23 NBA season.

The last dataset that used in this project was published on Kaggle by the user Vivo Vinco (https://www.kaggle.com/datasets/vivovinco/20222023-nba-player-stats-regular?resource=download&select=2022-2023+NBA+Player+Stats+-+Regular.csv). The 2022_23_NBA_Player_Stats-Regular (679 records) dataset contains all regular season stats for the 2022-23 season, which will comprise of the test set to evaluate the model.

Measurement

'Quality of Performance' in the NBA, is a key measurement that will be predicted in this project. The first research questions seek to predict the quality of performance in the playoffs based on the quality of performance in the regular season, so identifying and understanding this metric is an essential part of building and utilizing the predictive model.

In order to make predictions on the overall performance of a NBA player, two all-encompassing (a stat that attempts to measure all aspects of a player) advanced statistics will be used to make these predictions:

RAPTOR – Robust Algorithm (using) Player Tracking (and) On/Off Ratings

WAR – Wins Above Replacement

Developed Jay Boice, Nate Silver, and Neil Paine, RAPTOR was introduced to the NBA stat world in 2019, as it was proclaimed to build off of the classic plus-minus statistic which "that measures the number of points a player contributes to his team's offense and defense per 100 possessions, relative to a league-average player. For instance, a player with an offensive RAPTOR rating of +2.1 boosts his team's performance by 2.1 points per 100 offensive possessions" (Silver, 2019). Unlike a few other all-encompassing stats, RAPTOR does not use priors and it describes a player's and their team's performance while the player is on the court. WAR, on the other hand, is a measure of the number of wins a player would contribute to a name if they were replaced by a league-average player. As per the FiveThirtyEight website, "The replacement level estimate is derived from evaluating the historical performance of players on two-way contracts, who are quite literally on the fringes between the major and minor leagues (the NBA and the G League), a status that reflects the traditional definition of replacement-level players." If one wishes to dive into the depth of this stat, a reference included in the list at the end of this plan would be the place to start.

Any stat that claims to all-encompassing in no way implies its perfection. As mentioned before there are several intangible factors such as leadership, impact on the team off the court, chemistry, and many other factors that current statistics do not account for. Having said that, RAPTOR "fails to account for coaching, systems or synergies between teammates." As we are looking at predicting an 'individual' players performance rather than the team's performance, this is not perceived to be a huge hurdle in building and utilizing the model. One of the notable achievements of RAPTOR was the prediction of the Toronto Raptors winning the 2018-19 championship.

Methodology

For the first research question, the methodology involves building and training a multiple linear regression model. The model will predict the RAPTOR measure for a player in the postseason, given their regular season statistics as the input. Using the R-squared statistic in addition to residual analysis, the model can be finetuned by removing certain variables after assessing differences between predicted values and the ground truth.

Incorporating Principal Component Analysis (PCA), the number of independent variables can be altered to pursue an increment in the R-squared measure.

For the second research question, a K-Means Clustering approach seems appropriate, as similarities in performance in the draft combine and similarities in RAPTOR stats during the rookie season are to be observed. Since anthropometric and drill stats from the combine are being used, correlation coefficients between the combine and the athlete's rookie season can be computed to yield further insights in the exploratory data analysis section. We can then move onto visualizing these clusters to observe any distinct patterns.

Computational Outputs and Summaries

RQ1: Can we predict a player's performance in the playoffs based on their performance in the regular season?

As mentioned in the methodology, the multiple linear regression model will look to predict a player's RAPTOR statistic based on their regular season statistics. For their regular season, an average of their points, offensive rebounds, defensive rebounds, assists, steals, blocks, turnovers, free-throw %, 3-point %, and field goal %. Based on these averages as input, the model will predict a player's WAR in the post-season. The model will be trained on approximately 22,500 player's regular season statistics and post-season WAR stats. Using a 80-20 train-test split, the other '20' will be used to test the model. A detailed description of the WAR stat can be found in the 'Measurement' section of the project plan. Visualizations can be used to assess a player's predicted increment or decrement in performance based on their regular season statistics. Scatter plots can be used to study the relationship between the independent variable and the dependent variables. For past years, predicted vs. actual plots can be used as another method to validate the model's accuracy.

RQ2: Explore the similarity in performance of NBA rookies in the draft combine (anthropometric and combine drills) and their performance in their first season in the NBA.

Upon computing clusters based on two specific contexts – clustering based on draft drill and anthropometric data, and clustering based on rookie season offensive and defensive RAPTOR data, we can use a variety of plots, such as cluster visualizations, parallel coordinate plots, and cluster scatterplots to visualize the clusters along with any noticeable patterns or trends.

LITERATURE REVIEW

Predictive Analytics in Sports

A famous example that is consistently cited in the context of analytics in sports is the book "Moneyball", by Michael Lewis. While most teams take on a classical approach of going after star players, with large resources at hand, this book narrates a story of an approach that prioritizes statistics. Based on a specific objective of forming a team with a high on-base percentage, the Oakland A's recruited specific players who they believed would fit their current agenda. Using this unorthodox (at that time) approach, Billy Beane and his staff at the Oakland A's revolutionized the idea of using analytics in sports to aid teams in building their roster based on measures and statistics.

Taking this notion one step forward, predictive analytics seeks to 'predict' outcomes or in games or seasons based on some data about the athlete or team. In an article published by the University of Pennsylvania, the writer Eric Bradlow brings up a crucial point that besides winning, predictive analytics can spot outcomes such as injuries which can be life-threatening. Sports betting is also another well-known avenue where predictive analytics is largely used. Amy Howe (CEO of FanDuel) said that "Our entire business and our entire sports betting platform depends on real-time data, sophisticated analytics, and modeling." In addition to predicting the outcomes in a game, analytical strategies can be employed in improving customer experience at game venues. Bradlow also added that "experts can measure brain data at a granular level and measure an athlete's brain function over time." Which can possibly be used to spot injuries before their occurrence (Bradlow, 2023).

Predictive Analytics in Basketball

As cited in this article from Samford University, analytics within basketball began with the basic collection of simple measures such as points, assists, and rebounds. With the advent of more advanced forms of data collection such as individual player tracking using sensors and cameras, analysts and organizations have been developing what is now commonly known as 'advanced statistics' that reveal deeper insights into a player's and a team's performance (Fitzpatrick, 2022).

An article by media company Bleacher Report lists 13 of the most widely used advanced statistics in basketball. For example, **Player Efficiency Rating** describes a player's productivity measured per-minute, **Win-shares** attempt to measure a player's contribution to their team's victories while they are on the floor, and **True Shooting Percentage** is used to evaluate 'shooting' ability by considering 3-point makes and free-throw statistics (Khan, 2017).

The advanced statistic that currently has the highest root mean squared error value as compared to other is **Daily Adjusted and Regressed Kalman Optimized (DARKO)** created by Kostaya Medvedovsky. This statistic weighs a player performance rates a player with greater weight placed on their recent performance by incorporating plus-minus statistics and their recent box scores. Additionally, DARKO is one of the few advanced stats that incorporates a machine learning model. DARKO has its own website, where Medvedovsky says that "DARKO is built using a combination of classical statistical techniques and modern machine learning methods. DARKO is Bayesian in nature, updating its projections in response to new information, with the amount of the update varying by player and by stat, depending on DARKO's confidence in its prior estimate (Medvedovsky & Patton)."

For this project, the **Robust Algorithm (using) Player Tracking (and) On/Off Rating (RAPTOR) and Wins Above Replacement (WAR)** advanced statistics was used to represent a players overall performance. This stat was authored by Nate Silver et. al. and a breakdown of the statistic with regards to this project can be found in

the project plan. To be brief, this stat works similar to a plus-minus stat which measures a player's contribution to their team's offense and defense per 100 possessions. Like DARKO, RAPTOR incorporates box-scores in addition to ""on-off" component, which evaluates a team's performance when the player and various combinations of his teammates are on or off the floor (Silver, 2019)."

Predictive Analytics in NBA Games

According to an article by Josh Weiner on towardsdatascience.com, there is a model with a prediction of accuracy of 74.1%. The article also makes us aware that analyzing playoffs games in particular, can lead to bias, as "playoff teams are more consistent in a number of stats throughout the regular season, and playoff game expected outcomes likely experience less variance as a result (Weiner, 2022)." In the same article work on predictive analytics with respect to NBA teams has been carried out, but there does not seem to be any specific work on predicting player performance in the playoffs, based on their regular season performance. Since the project cited above was looking to classify games, a logistic regression model and random forest classifier models were built.

FINAL RESEARCH QUESTIONS

Dataset

For this project, I chose to work with all the historical and live data from the National Basketball Association. To access this large database, I am using the 'nba_api' python package. This package is an Application Programming Interface (API) for the 'stats' page on the NBA website (https://www.nba.com/stats). This is an open source package which has several endpoints, giving access to a wide variety of stats from players, teams, and other filters from the year 1951.

Dataset Characteristics

The total number of observations for the relevant datasets is 111975. Here is the split based on the research question:

regseason_teams = 30 (Number of teams in the NBA)
playoff_teams = 16 (Number of teams that make it to the playoffs)
players = 15 (Players per team)
years = 50 (Number of years the data dates back to)
top_500 = 500 (top 500 players in a dataset)
top_150 = 150 (top 150 players)

Question 1: (regseason_teams*players*years + playoff_teams*players*years + top_500*years + top_150*years) – This dataset makes up for the training and validation dataset.

draft_num = 57 (Approximate number of players drafted every year)
years = 23
seasons = 15 (Number of seasons data is being taken from)
draftcombineanthro = 81 (Number of draft combine anthropometric stat observations)
draftcombinestats = 65 (Number of draft combine observations)
draftcombinedrills = 65 (Number of draft combine drill observations)

Question 2: (draft_num * years + years * draftcombinestats + years * draftcombinedrills + draftcombineanthro * years + draft_num * years)

Note: All constants (such as 65 & 81) refer to the number of records in the respective database

Since the nba_api gives access to a variety of datasets, the number of fields in each dataset is marginally different. Here are the types of variables in general: PLAYER_ID, RANK, PLAYER, TEAM_ID, TEAM, GP (Games Played), MIN (number of minutes played), FGM (Field Goals Made), FGA (Field Goals Attempted), FG_PCT (Field Goal % - (FGM/FGA)*100), FG3M (3-pointer made), FG3A (3-pointer attempted), FG3_PCT (3-pointer %: (FG3M/FG3A)*100), FTM (Free Throws Made), FTA (Free Throws Attempted), FT_PCT (Free Throw %: (FTM/FTA)*100), OREB (Offensive Rebounds), DREB (Defensive Rebounds), REB (Total Rebounds), AST (Assists), STL (Steals), BLK (Blocks), TOV (Turnovers), PF (Personal Fouls), PTS (Points Scored), EFF (Efficiency: ((PTS + REB + AST + STL + BLK – Missed FG – Missed FT - TO) / GP)), AST_TOV (Assist to Turnover ratio) and STL_TOV (Steal to Turnover ratio).

Additionally, here is the documentation for the nba_api package:
https://github.com/swar/nba_api/tree/master/docs/

Research Question-1

Can we predict the quality of performance of a player in the playoffs based on their quality of performance in the regular season? Build a predictor (regression model) that predicts the quality of performance of a given player based on their performance in the regular season.

Research Question-2

Explore the similarity of rookies' performance in the NBA draft combine in two specific settings – drills and anthropometrics. Parallelly explore the similarities of these rookies' performances in their first season NBA. Use various visualization techniques to observe these clusters and patterns.

EXPLORATORY DATA ANALYSIS

Data Source and Characteristics

The data used in this project was sourced from locations: NBA.com/stats and the github page for fiverthirtyeight.com. From the data produced by fivethrityeight, all modern (2014-present) and historical (1976-present) RAPTOR statistics were used. They each came in their independent comma separated value datasets. From the NBA website, all regular season data for every NBA player was collected. This resulted in assimilation of regular season data for exactly 4815 NBA players. As per the fiverthirtyeight's website, the historical dataset only consists of approximations of RAPTOR data as live player tracking went online in 2014. Hoping to increase the accuracy of the model by using the most accurate RAPTOR data available, only the modern RAPTOR dataset will be used for this project. In the modern RAPTOR dataset, there are a total of 4685 rows and 21 variables. This dataset consists of all RAPTOR statistics for every player from 2014 to the present day. The variables from this dataset that are relevant for this project are:

player_name: Name of the player

season: season for which RAPTOR data is recorded

poss: Number of possessions played

mp: Number of minutes played

raptor_offense: RAPTOR offensive statistic calculated based on the box score and live player tracking data. This data is for the regular and season and the playoffs combined.

raptor_defense: RAPTOR defensive statistic calculated based on the box score and live player tracking data. This data is for the regular and season and the playoffs combined.

raptor_total: raptor_offense + raptor_defense

war_reg_season: wins above replacement during the regular season. 'war' is an all-encompassing stat computed by the RAPTOR algorithm that depicts an evaluation of a player's worth to their team. This is done by approximating the number of additional wins a player would contribute to the team if they played in-place of a 'replacement' level (average) player.

war_playoffs: wins above replacement during the playoffs.

The regular season statistics dataset sourced from the NBA website has 6780 rows and 28 columns when accounting for all the players from 2014 onwards (considering that we are only working with players in the modern RAPTOR dataset). Here is the list of all the relevant variables from this dataset:

SEASON_ID: season for which regular season data will be collected. This variable will soon be converted into 'season' which has integer data instead of string data
PLAYER_AGE: Player's age in the given season

GP: Games played in the regular season

MIN: Total minutes played in the regular season

FG_PCT: (total field goals made / total field goals attempted) * 100

FG3_PCT: (total 3-point shots made / total 3-point shots attempted) * 100

FT_PCT: Free-Throw make Percentage

OREB: Total offensive rebounds

DREB: Total defensive rebounds

REB: OREB + DREB

AST: Total assists made per regular season

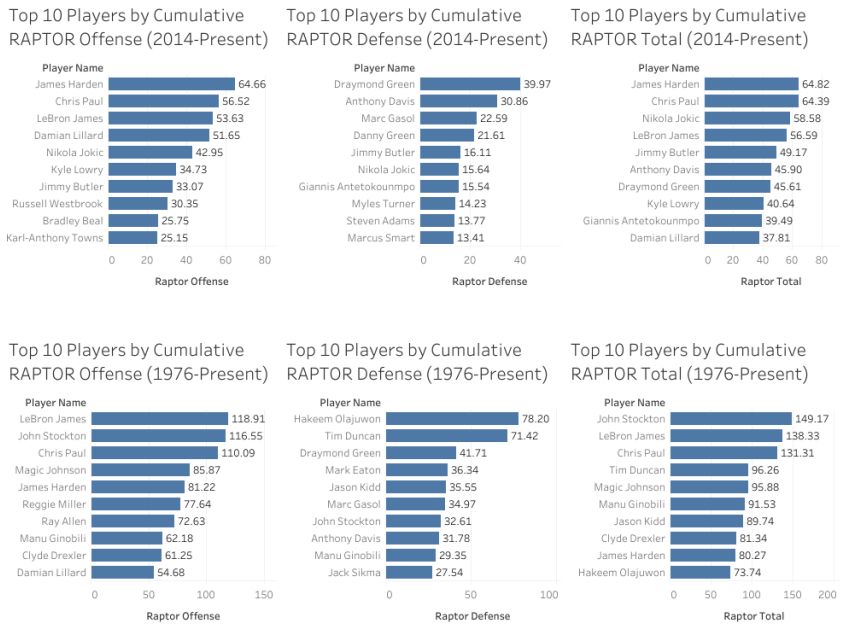
STL: Total successful steals per regular season

BLK: Total blocks per regular season

TOV: Total turnovers committed per regular season

PTS: Total points scored per regular season

Before we dive into the pre-processing aspects of the exploratory data analysis relevant to each research question, here is some EDA with regards to the modern and historical RAPTOR datasets. Presenting the leaders of RAPTOR cumulative stats on offense, defense, and totals in the historic and modern RAPTOR datasets:



Leaders in the offensive RAPTOR statistic include notable players such as LeBron James, James Harden, Damian Lillard, John Stockton, and Chris Paul, who have been known for their offensive contributions. On the defensive side, we see players such as Draymond Green, Hakeem Olajuwon, Anthony Davis, and Tim Duncan. Finally for total RAPTOR, we get an overall view of players who performed well offensively and defensively, except cases where a player had a significantly high offensive or defensive performance. For example, James

Harden is the leading offensive player by cumulative RAPTOR from 2014 even though he is not known for his defense, because his offensive RAPTOR ratings are so high.

When looking at RQ2, we will explore relationships between these RAPTOR categories and draft combine performances for rookies in the NBA.

Data Analysis and Pre-Processing for RQ1

As a brief reminder, RQ1 looks to build a predictive model that predicts a players playoff performance as a 'Wins Above Replacement' value based on their performance in the regular season. The EDA for this question was used to gain insights to set-up the model, based on the strength of individual relationships between the independent variables and the dependent variables. Before we look at these relationships, some pre-processing was performed on the regular season stats dataset to create new fields:

PPG (Points Per Game): PTS / GP

APG (Assists Per Game): Assists / GP

OREBPG (Offensive Rebounds Per Game): OREB / GP

DREBPG (Defensive Rebounds Per Game): DREB / GP

STLPG (Steals Per Game): STL / GP

TOVPG (Turnovers Per Game): TOV / GP

BLKPG (Blocks Per Game): BLK / GP

war_reg_season: wins above replacement for the regular season (sourced from modern RAPTOR dataset)

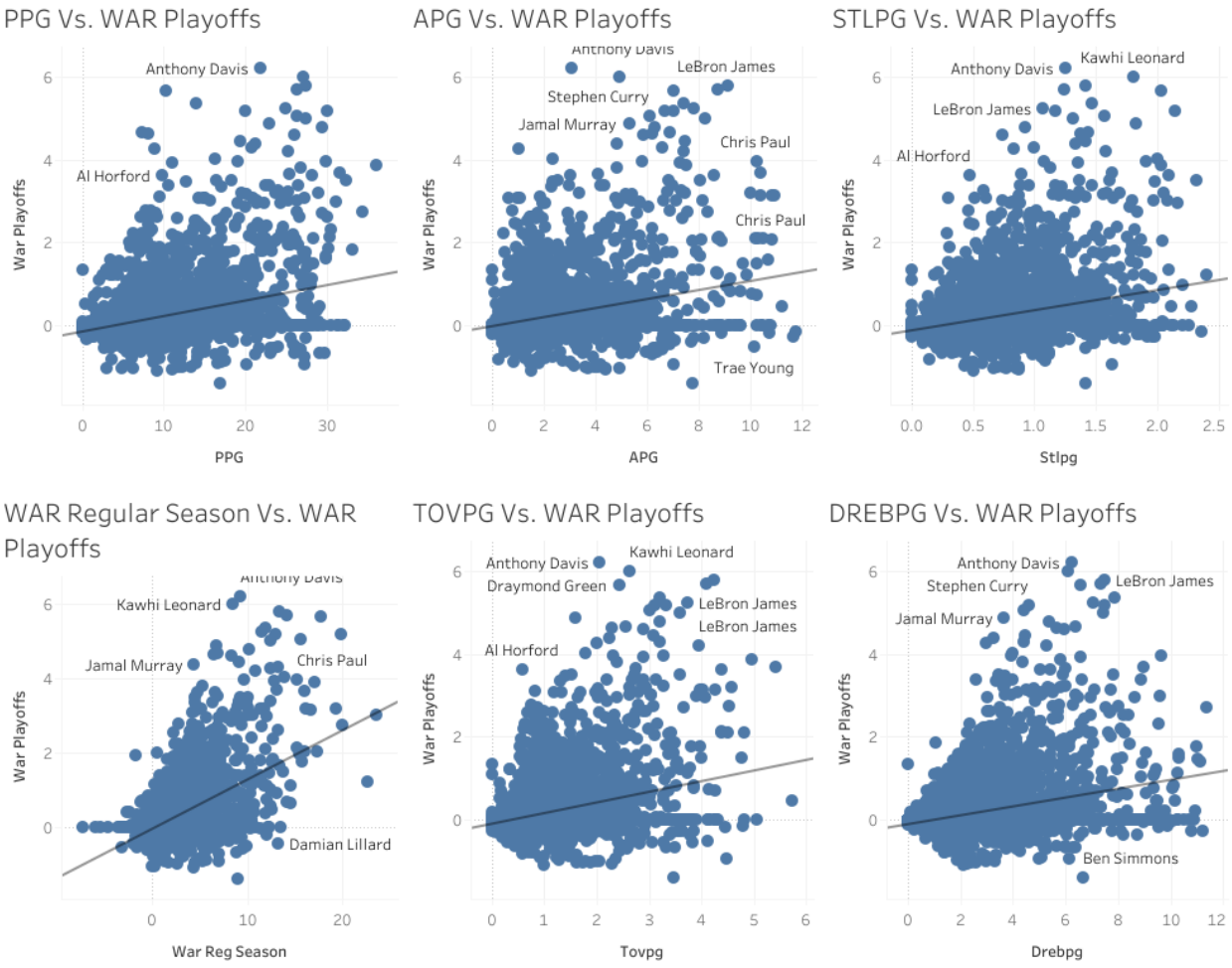
war_playoffs: wins above replacement for the playoffs (sourced from modern RAPTOR dataset)

Instead of using total points, assists, rebounds, steals, blocks, and turnovers per season, creating new variables which the describe the rate of these statistics per game seems to describe a player's average performance across all games played. This incorporation of these variables also takes care of possible drops in overall points due to injuries, suspensions, or other factors that may hinder a player's participation in the regular season. For example, if a player had an incredible points per game average for the first 40 games of the season, and then missed 20 games in-between the season before the playoffs. Credit is given where it is due as his points average is retained as it is calculated over game played. The same goes for assists, steals, rebounds, blocks, and turnovers.

Initially, all the players from the modern RAPTOR database were noted. Using this list of names, all regular season stat records of each player's seasons was assimilated. For each season, the player's war_reg_season and war_playoffs stats were added to the regular season stats dataframe. This makes it easier to plot scatter-plots to study the relationships between the independent variables: FG_PCT, FG3_PCT, FT_PCT, PPG, APG, OREBPG, DREBPG, STLPG, BLKPG, TOVPG, and war_reg_season AND the dependent variable: war_playoffs. After plotting scatter-plots on Tableau, trendlines were added to the plots to get a quick read into the strength of the relationship using the R-Squared statistic. Before the visualizations are presented, here are the list of all the R-Squared stats for the relationship between each independent variable and war_playoffs:

VAR	R^2
FG_PCT	0.0048505
FG3_PCT	0.0021361
FT_PCT	0.0076888
PPG	0.109057
APG	0.0891331
OREBPG	0.0093362
DREBPG	0.0798548
STLPG	0.0828329
BLKPG	0.025358
TOVPG	0.0902862
war_reg_season	0.292346

As seen in the table above, the independent variables that had the strongest relationships with war_playoffs are war_reg_season, PPG, APG, TOVPG, STLPG, and DREBPG. Here are the scatterplots for all 6 relationships:



Exploratory Data Analysis for RQ2

In RQ2, we are looking to explore the strength of the relationship between draft combine drill + anthropometric statistics AND the rookie's performance in their first season in the NBA. Unlike RQ1 where wins above

replacement is used to evaluate a player's overall quality of performance, we will be using the offensive, defensive, and total RAPTOR statistics to determine overall quality of performance.

Through the EDA, it was observed that there was no significant correlation between any of the drill or anthropometric variables and any of the three (offense, defense, and total) RAPTOR variables. Starting with draft combine drill data, the data was sourced from the NBA website using the nba_api package. All draft combine data from the year 2014-2023 was sourced, and similar to pre-processing performed for RQ1, RAPTOR offense, defense, and total stats were added to the draft drill data. After dropping all rows with 'None' or 'na' values, a dataframe of 191 rows and 12 variables was produced. The relevant variables for this dataset:

STANDING_VERTICAL_LEAP: Max height that a player can reach by jumping from a stationary standing position

MAX_VERTICAL_LEAP: Max height that a player can reach by using a run-up

LANE_AGILITY_TIME: Time a player takes to do the lane agility test (involves running around the key of the basketball court – in 4 different directions)

MODIFIED_LANE_AGILITY_TIME: Time a player takes to do one stretch of the lane agility test

THREE_QUARTER_SPRINT: Time taken to sprint to the 3/4th mark of the court

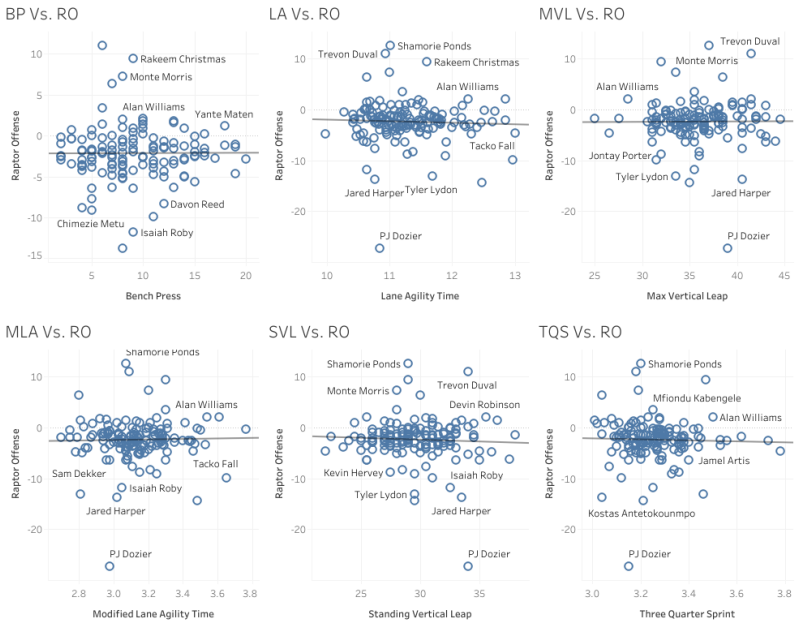
BENCH_PRESS: Maximum number of repetitions performed in the bench press exercise with 185 lbs.

Here is the table with all the R-Squared values for the relationship between draft drill stats and RAPTOR offense, defense, and total:

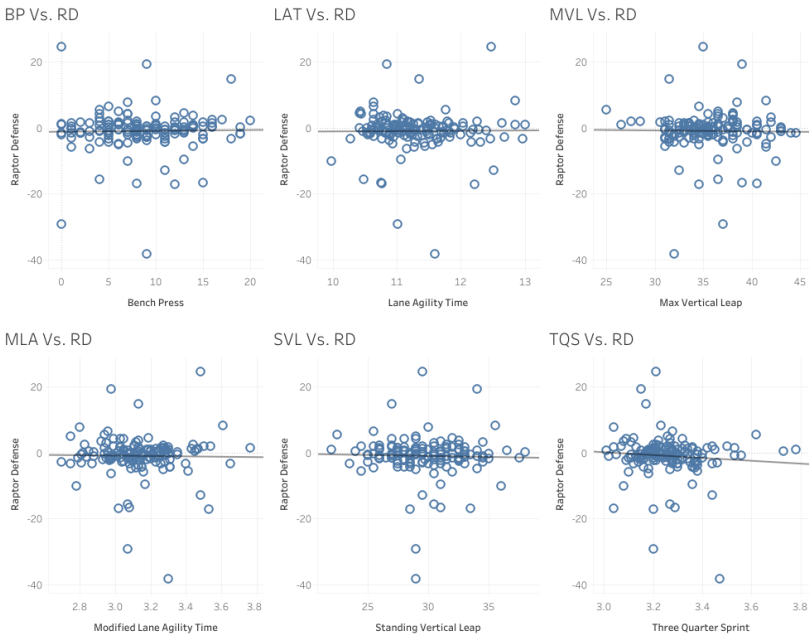
	RAPTOR Offense	RAPTOR Defense	RAPTOR Total
Standing Vertical Leap	0.002738	0.000830	0.0038976
Max Vertical Leap	4.917e-05	0.000247	0.000118
Lane Agility Time	0.001939	9.591e-05	0.000369
Mod. Lane Agility Time	0.000544	0.000286	1.777e-06
3/4th Sprint	0.000896	0.007835	0.0113121
Bench Press	0.001073	0.000368	0.0016192

Scatterplots depicting the relationships described above:

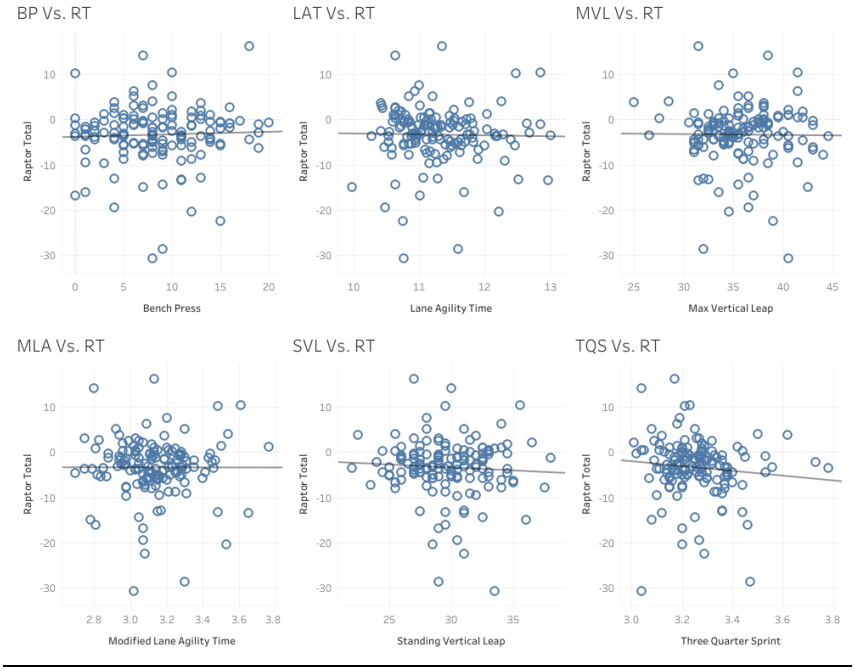
Draft Combine Drill Stats Vs. RAPTOR Offense



Draft Combine Drill Stats Vs. RAPTOR Defense



Draft Combine Drill Stats Vs. RAPTOR Total



Draft anthropometric data was also sourced from the NBA website using the nba_api package. After collecting anthropometric data for all rookies from 2014-2023, the dataset consisted of 539 rows and 18 variables. RAPTOR offense, defense, and total stats were added to the end, which produced a dataset of 369 rows and 22 columns after 'None' and 'na' values were dropped. Here is the list of relevant variables from the draft anthropometric dataset:

HEIGHT_W_SHOES: Player's height without shoes on

STANDING_REACH: Max reach by the player when standing

BODY_FAT_PCT: Body Fat %

HAND_LENGTH: Length of hand from top to bottom

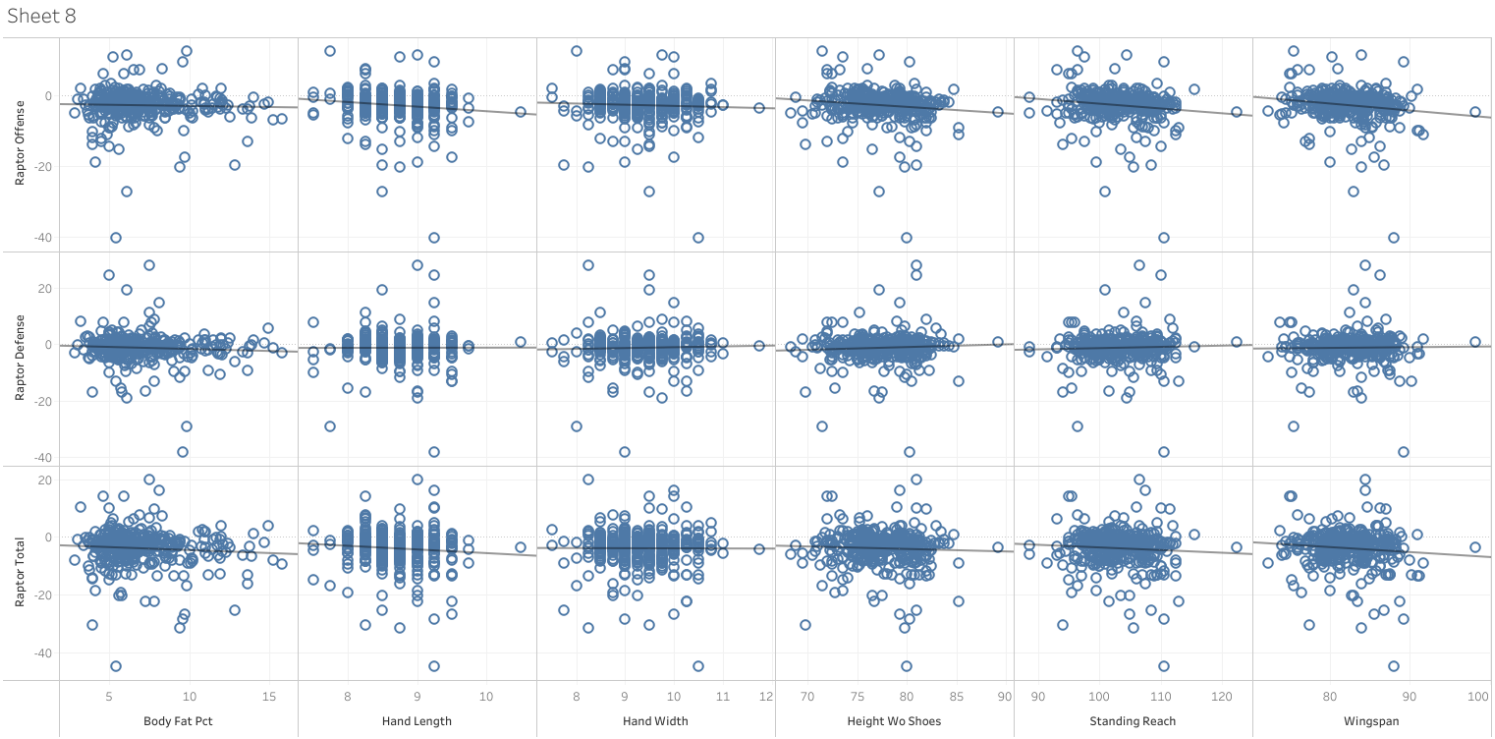
HAND_WIDTH: Width of hand from left to right

WINGSPAN: Distance between tips of left and right fingers when player has their arms fully stretched to their sides

Here is the table with all the R-Squared values for the relationship between draft anthropometric stats and RAPTOR offense, defense, and total:

	RAPTOR Offense	RAPTOR Defense	RAPTOR Total
Height	0.017641	0.003835	0.001945
Standing Reach	0.021182	0.001748	0.004746
Body Fat %	0.001147	0.004684	0.005979
Hand Length	0.017363	9.287e-06	0.008100
Hand Width	0.002595	0.001550	2.123e-05
Wingspan	0.028193	0.000356	0.010521

Draft Anthropometric Vs. RAPTOR (Offense, Defense, and Total)



Based on these low R-Squared values, any sort of regression approach would not prove to be effective. Perhaps a clustering approach can yield further insights into players that perform similarly in the draft and in their first season.

METHODOLOGY

RQ1: Can we predict a player's performance in the playoffs based on their performance in the regular season?

In an attempt to predict the quality of player's performance, a deep literature review of all-encompassing stats was performed. These advanced stats attempt to describe a player's all-round performance with a single number, instead of using multiple statistics for their points, rebounds, assists, steals, etc. The statistic being predicted in this Research Question is WAR or wins above replacement. Further details about these advanced statistics can be found in the exploratory data analysis document, but in brief, it describes a player's contribution to his team when replaced with a 'replaceable' or average player. Several independent variables such as points per game, assists per game, offensive and defensive rebounds per game, steals per game, blocks per game, turnovers per game, and WAR stats for the regular season will be used to predict WAR for the playoffs. This calls for a regression approach since we are modeling the relationship between these variables. In particular, two types of regressors can be built – a multiple linear regression model and a random forest regression model. A multiple linear regression model can aim to predict WAR for the playoffs based on linear relationships between variables. As seen in the EDA document, there does not seem to be a significantly strong relationship between any of the independent and dependent variables. For this reason, an alternative model – random forest regressor can be used to model possibly more complex relationships between independent and dependent variables. Furthermore, using residual analysis, R-Square scores for each combination of independent variables can be tracked to identify the ideal combination of variables that yields the highest R-Squared result.

RQ2: Analyze and Explore the similarities in a rookie's performance in the NBA draft combine and their offensive & defensive RAPTOR in their first season in the NBA.

Using the nba_api python package, one can source draft drill statistics and draft anthropometric statistics from the NBA website in addition to game data. This Research Question aims to study the relationship between a player's performance in the draft drills and anthropometric measurements, which consists of tests of athleticism, speed, agility, strength, and bodily measurements AND their performance in their first season in the NBA. In this research question , RAPTOR data from fivethrityeight will be used as an all-encompassing measure of overall player performance. K-Means Clustering will be performed to assess the similarities in performance based on features such as standing and maximum vertical leap, lane and modified lane agility, three quarter sprint time, bench press, height, hand width and length, body fat %, standing reach, and wingspan AND RAPTOR offense and defense. As a final result, visualizations which depict similarities in players performances will be employed to reveal observations and insights.

ANALYSIS

Models (Pre-processing and Modeling)

In this project, there are two models and one clustering approach that are utilized to answer the two research questions. In **RQ1**, our aim is to build a predictor of an NBA athlete's playoff performance, given their regular season performance as input. To be more specific, playoff performance is to be predicted as Wins Above Replacement (WAR), and regular season performance is measured by the player's age, field goal percentage, 3-point percentage, average points per game, average assists per game, average offensive rebounds per game, average defensive rebounds per game, average steals per game, average blocks per game, average turnovers per game, and WAR for the regular season.

Before going into the details of the models, understanding the pre-processing step of the data science life cycle for **RQ1** is essential. As stated earlier, the 'nba_api' API client was used to source data from the statistics page of the NBA website. Additionally, RAPTOR data from 'FiveThirtyEight' was used to attain WAR statistics for the regular season and the playoffs. RAPTOR statistics are primarily built on on-and-off box scores, as well as player tracking data. Since player tracking data only became available from 2014 onwards, 'FiveThirtyEight' presents its RAPTOR stats for players in two separate csvs: 'modern_RAPTOR_by_player.csv' which consists of all RAPTOR stats for players participating in the NBA from the 2014 season onwards and 'historical_RAPTOR_by_player.csv' which consists of all RAPTOR stats from the 1976 season. On their GitHub page (source of the datasets), FiveThirtyEight states that their 'modern' dataset was more accurate as player tracking in the NBA only began in 2014, while data from the historical dataset were approximations. For this reason, I chose to only use data from 'modern_RAPTOR_by_player.csv'. Now that we have all the RAPTOR stats for every NBA player (and each of their seasons) that participated in the NBA since 2014, we can proceed to source all the regular season statistics for every NBA player since 2014.

Using 'nba_api', a static 'players' endpoint on the NBA stats page was accessed to attain a complete list of all the players that ever played in the NBA. This list sums up to a total of 4815 players. The 'id' and 'full_name' of each player was noted. Using a 'playercareerstats' package from one of the API clients endpoints, we can access career stats of any player, given their 'id' as a parameter. Since we only need regular season stats for players in the 'modern_RAPTOR_by_player.csv' (a total of 665 players) file, we only need to make 665 calls to the 'playercareerstats' endpoint. Through an iterative process, regular season statistics for all 665 players can be compiled into a list of dataframes, which will then be concatenated to form one big dataframe. Once this dataframe is stored as a csv for future use (instead of having to source all the data using the API client again), we can drop all records with player statistics dating before 2014, since we only have RAPTOR data for 2014 onwards. As seen in the EDA section of this project, we can then calculate points, assists, offensive and defensive rebounds, steals, blocks, and turnovers per game, by dividing the respective fields (total points, assists, rebounds, steals, blocks, and turnovers by the total number of games played). Finally, before performing a left merge with the 'modern_RAPTOR' dataset, there were a few differences in player names between the RAPTOR dataset and the regular season stats dataset (e.g. OG Anunoby in the RAPTOR dataset was O.G. Anunoby in the regular season dataset). After adjusting to these changes, and a left merge was performed, the two dataframes were combined to yield a 'combined_stats' dataframe. After deleting duplicate columns, we have a dataframe with the following variables:

Variable	**Definition**
FULL_NAME	Player's full name
PLAYER_ID	Player's ID number from the regular season dataframe
SEASON	Season in the NBA for which regular season and RAPTOR data was populated
TEAM_ID	A unique team ID for each team
TEAM_ABBREVIATION	An abbreviated team name (e.g. Toronto Raptors - TOR)

PLAYER_AGE	Player's age for the given season
GP	Games played per season
MIN	Total minutes played per season
FG_PCT	Field goal percentage (percentage of field goals made over field goals attempted) per season
FG3_PCT	Three-point makes percentage per season (makes over attempts)
FT_PCT	Free-throw percentage per season (makes over attempts)
PPG	Points scored per game
APG	Assists made per game
OREBPG	Offensive rebounds collected per game
DREBPG	Defensive rebounds collected per game
STLPG	Steals per game
BLKPG	Blocks per game
TOVPG	Turnovers committed per game
war_reg_season	WAR for the regular season
war_playoffs	WAR for the playoffs

After completing the data collection, data cleaning, and pre-processing steps, we can begin to build and train our models. The first predictor built using the combined stats dataset is a multiple linear regressor. Using independent variables: PLAYER_AGE, FG_PCT, FG3_PCT, PPG, APG, OREBPG, DREBPG, STLPG, BLKPG, TOVPG, war_reg_season, and the target variable: war_playoffs, a LinearRegression model from the sklearn.linear_model module can be assembled. The model is built with the aim of capturing the combined linear relationships between the dependent and independent variables. Additionally, r2_score and mean_square_error from sklearn.metrics was used to assess the model after training. To create a the training and test set, an 80-20 train-test split performed using train_test_split from sklearn.model, along with a random_state of 42. The linear model was fitted with coefficients in an attempt to minimize the residual sum of squares. When using all the independent variables from above, the coefficients are:

Variable	**Coefficient**
PLAYER_AGE	0.00629
FG_PCT	0.10752
FG3_PCT	-0.02834
PPG	0.00386
APG	0.00953
OREBPG	-0.09861
DREBPG	0.04158
STLPG	0.00362
BLKPG	0.06565
TOVPG	-0.01917
war_reg_season	0.10992

Train and test scores for the model ended up being: 0.27813 and 0.30893 respectively. Instead of considering all independent variables, we can look at combinations of independent variables to build the model upon, such that the R-Squared score is maximized and obtain a model that captures a higher degree of variability in the target variable. Through an iterative process, all 8189 combinations of variables can be used to create models to find the highest possible R-Squared value. This resulting combination of independent variables end up being: PLAYER_AGE, FG_PCT, APG, OREBPG, STLPG, BLKPG, and war_reg_season. Fitting a Linear model to these variables results in a R-Squared score of 0.31225 – a small improvement from the initial score. Here are the summarized results before and after feature selection:

	Before Feature Selection	After Feature Selection
Mean Squared Error	0.32318	0.32163
Train Score	0.27813	0.27361
Test Score	0.30893	0.31225

Saving the model results for the next section, we can proceed to the construction and the training of the next predictor – the random forest regressor. The primary motivation behind using a random forest regressor to predict a player's WAR for the playoffs is to model the complex relationship between the independent and target variables. Through the EDA, it is evident that the linear relationship between the independent variables and WAR for the playoffs is weak. Constructing a multiple linear regressor seems to reveal a low test score, which leads us to the hope of a higher R-Squared metric with random forest regressor. The process that goes into building this regressor is very similar to the one used to build the previous model. Using the same training data set, and the features mentioned in the coefficients table above, a random forest regressor was initialized and fitted to the training set. Similar to the MLR model, an 80-20 train-test split was used with a random state of 42. The regressor was initialized with default values (100 trees, 2 samples to split an internal node, 1 sample to be at a leaf node, and so on). An immediate decrement in the MSE and increment in train and test scores were noticed: MSE = 0.28991, Train Score: 0.90957, and Test Score = 0.38007. Upon running through all possible combinations of the features to gain the highest R-Squared score, the following combination of features were selected: PLAYER_AGE, FG3_PCT, APG, OREBPG, DREBPG, BLKPG, TOVPG, and war_reg_season. Here are the results before and after feature selection:

	Before Feature Selection	After Feature Selection
Mean Squared Error	0.28991	0.27098
Train Score	0.90957	0.90283
Test Score	0.38007	0.42056

Analysis with K-Mean Clustering

RQ2 begins with an attempt to find a relationship between draft combine statistics and a rookie's first year performance in the NBA. Similar to **RQ1**, the nba_api client was used to source draft combine data for all combine participants from 2014 onwards. Draft data was sourced as two separate dataframes – draft drill data and draft anthropometric data. Draft drill data dealt with data from drills conducted at the draft combine – such as lane agility drills, sprints, vertical jumps, bench press reps, and so on. Draft anthropometric data consisted of measurements of weight, height, wingspan, hand height, etc. The pre-processing to set up these datasets, consisted of sourcing draft data as two separate dataframes, and adding on offensive and defensive RAPTOR for all the players in their first season in the NBA. Since we are using the modern RAPTOR dataset with comparatively more accurate RAPTOR data, we only use combine stats from 2014 onwards.

The EDA reveals a significantly low coefficient of determination between independent variables and the target variables (offensive and defensive RAPTOR) for both drill and anthropometric data. R-Squared scores for the independent and target variables along with their visualizations can be found in the EDA section of this project. An understanding of draft combine and rookie season performance can still be attained using a K-Means Clustering approach which could reveal similarities in performances in the draft combine and in the NBA. Going into the clustering approach, here are the features considered in each dataframe:

DRILL

Variable	Definition
STANDING_VERTICAL LEAP	Maximum height a player can reach when jumping from a stationary standing position
MAX_VERTICAL_LEAP	Maximum height a player can reach when jumping with a run-up
LANE_AGILITY_TIME	Time taken to complete a lane agility test
MODIFIED_LANE_AGILITY_TIME	Time taken to complete a modified lane agility test
THREE_QUARTER_SPRINT	Time taken to sprint 3/4ths of the court
BENCH_PRESS	Max number of becnh press reps perfomred with 185lbs
raptor_offense	Offensive RAPTOR for the rookie's first season
raptor_defense	Defensive RAPTOR for the rookie's first season

ANTHROPOMETRIC

Variable	Definition
HEIGHT_WO_SHOES	Player's height without shoes
WEIGHT	Player's weight
WINGSPAN	Player's wingspan
STANDING_REACH	Maximum height accessed by a player from a standing position
BODY_FAT_PCT	Player's body fat %
HAND_LENGTH	Player's hand length
HAND_WIDTH	Player's hand width

For each dataset, there were two instances of clustering performed to generate two sets of labels. One was based on the features mentioned in the tables above, and the second was based on offensive and defensive raptor. This allows us to observe players that performed similarly in the draft and players that performed similarly in the NBA. The KMeans and the StandardScaler packages from sklearn.cluster and sklearn.preprocessing modules (respectively) were used to first transform the dataset and then perform clustering with 5 centers (or labels) and a random state of 42. Similarly, clustering with the same parameters was performed on offensive and defensive RAPTOR.

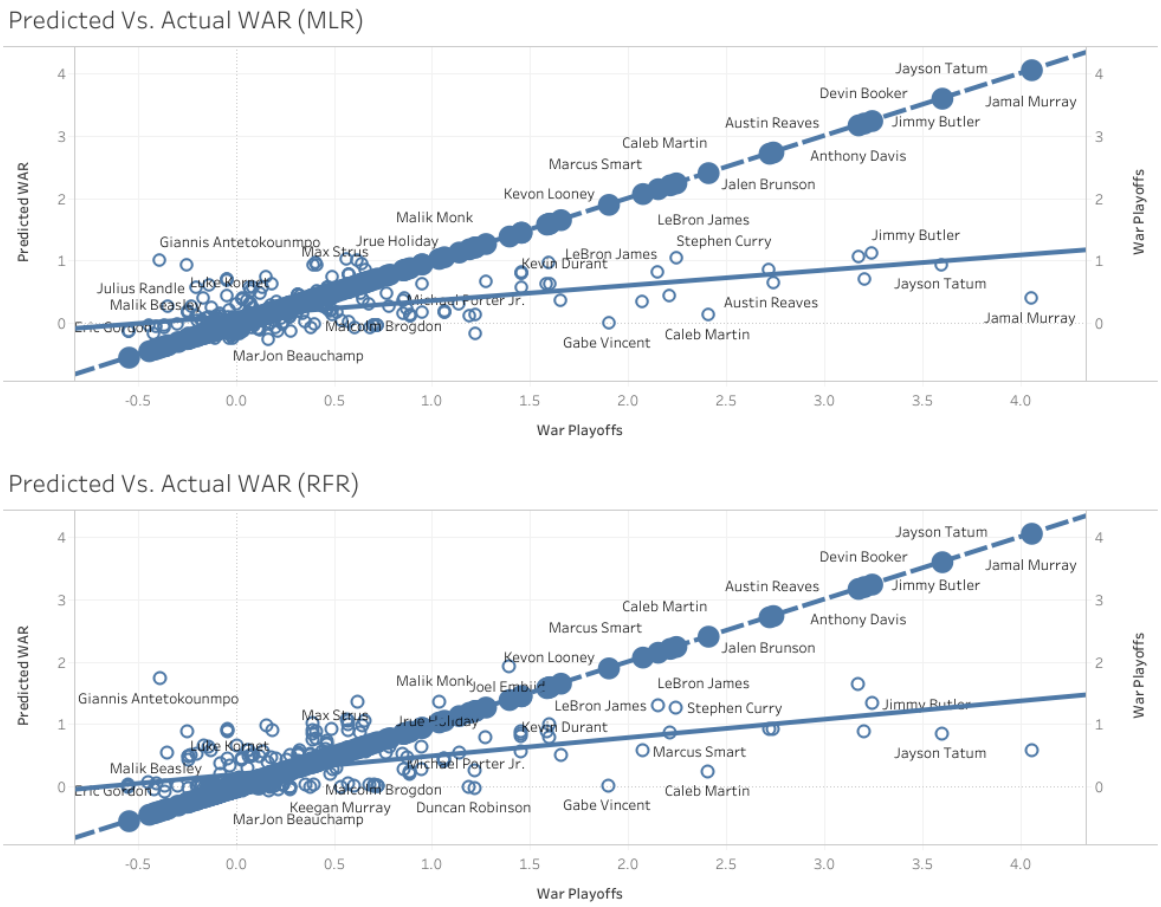
Model Results

In the data visualization section of this project, residual plots and other model results plots can be found – which depict the performance of the model when tested on the training data. The model was used to predict the playoff WAR for each record in the training data, and visualizations were used to depict the results. In this analysis portion, we will be analyzing the model's performance based on a fresh test which consists of the most recent regular season data for the 2022-23 regular season in the NBA. To access regular season data from the most recent season, a dataset published on Kaggle, by Vivo Vinco was used. Pre-processing for this dataset consists of creating features for 'per game' statistics from the season total statistics. This is done by simply dividing the season total by the number of games played. For example, assists per game = assists(season total) / game played. After creating new features and deleting irrelevant ones, a left merge can be performed with the latest_RAPTOR_by_player.csv file which consists of regular season and playoff WAR for the 2022-23 season. The merge will add regular season and playoff WAR for each player. After the merge, all records where the playoff WAR is 0 can be dropped, as that player's team did not make the 2023 playoffs, or they missed the playoffs due to an injury. The multiple linear regressor which was created and trained earlier is then used to

predict playoff WAR. We can now proceed to analyzing the results of the model as we have a MLR predicted playoff WAR as well as the true playoff WAR. The exact same approach is employed when using the random forest regressor to predict playoff WAR. Here are the tabulated results of the both models on this test set:

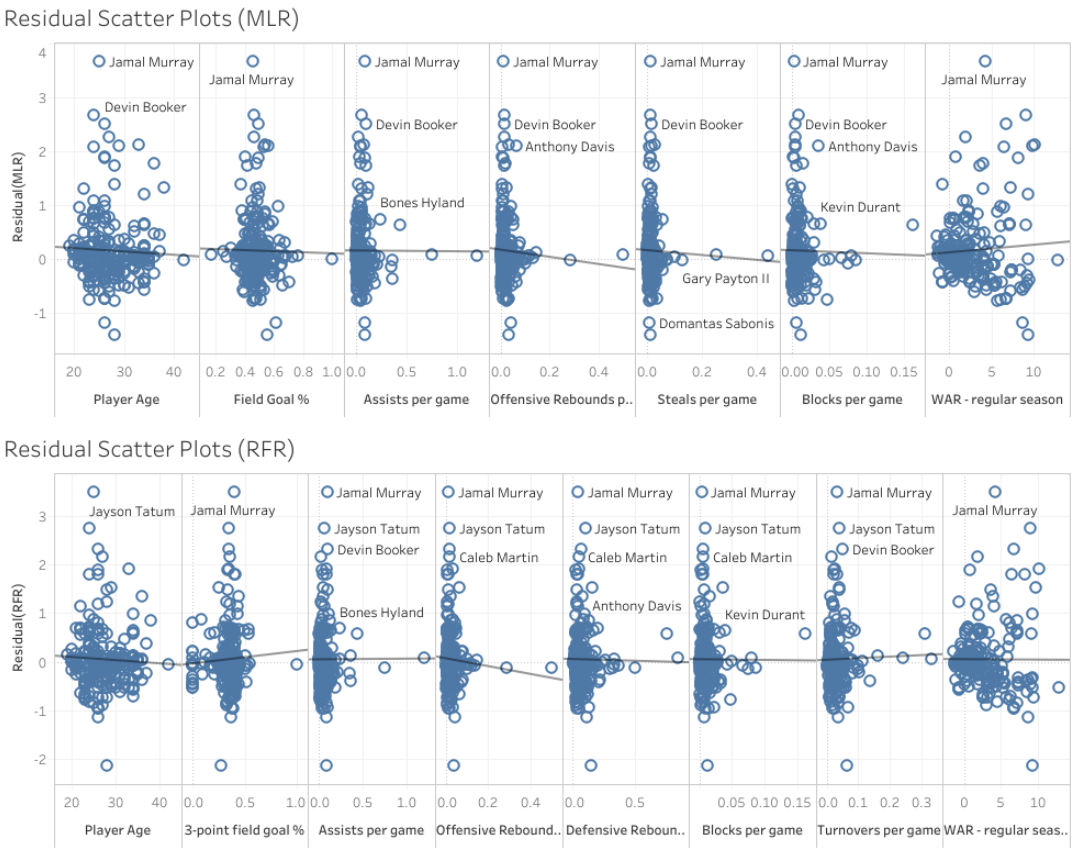
MODEL	Multiple Linear Regressor	Random Forest Regressor
Mean Squared Error	0.38266	0.35865
Mean Absolute Error	0.37041	0.36903
R2 Score	0.21960	0.26858

Additionally, we can use plots to visualize residual and actual vs. predicted plots:



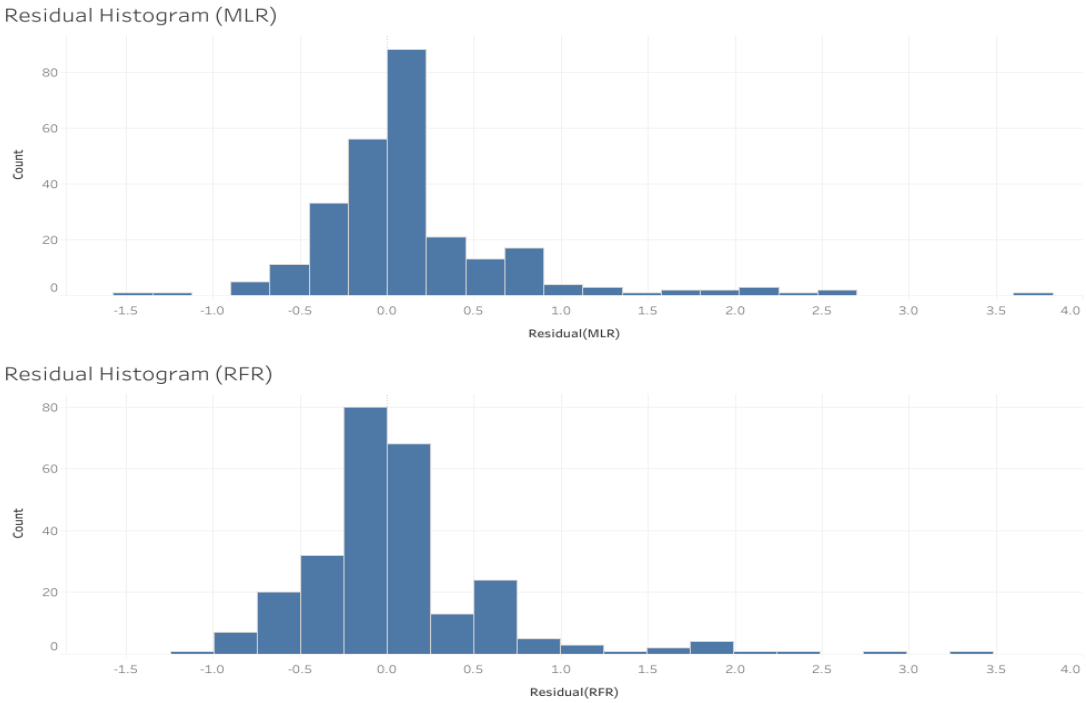
Predicted Vs. Actual WAR – Scatterplots

The thick line represents the diagonal, and the thinner blue line represents the linear trend line between predicted and actual WAR. R-Square scores for the trend lines in the respective plots are: 0.27133 and 0.27582.

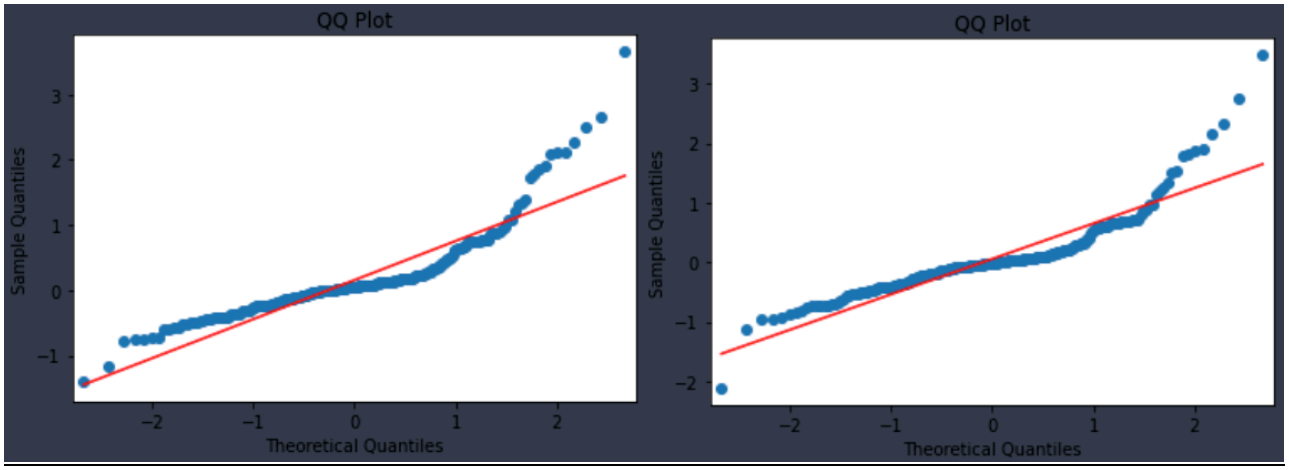


Residual Scatterplots

Through these residuals it is easy to notice the number of outliers in the 2022-23 playoffs. Upon analyzing the relationship between residuals and each relevant feature, the points are not always scattered evenly along the horizon, revealing a need for possible finetuning in the context of these features.



Residual Histograms



Q-Q Plot for MLR (left) and RFR (right)

As shown above, the distributions of the residuals denote a skew to the right along with several outliers towards the right tail, and the Q-Q plot reflects the deviation from normality seen in the histograms.

K-Means Clustering Results

Since the motivation behind the clustering approach was to analyze clusters based on the performance in the NBA and the combine, it is apt to use some of the results from the data visualization part of this project. For example, one of the key findings from the clustering approach was depicting the means for each RAPTOR cluster label.

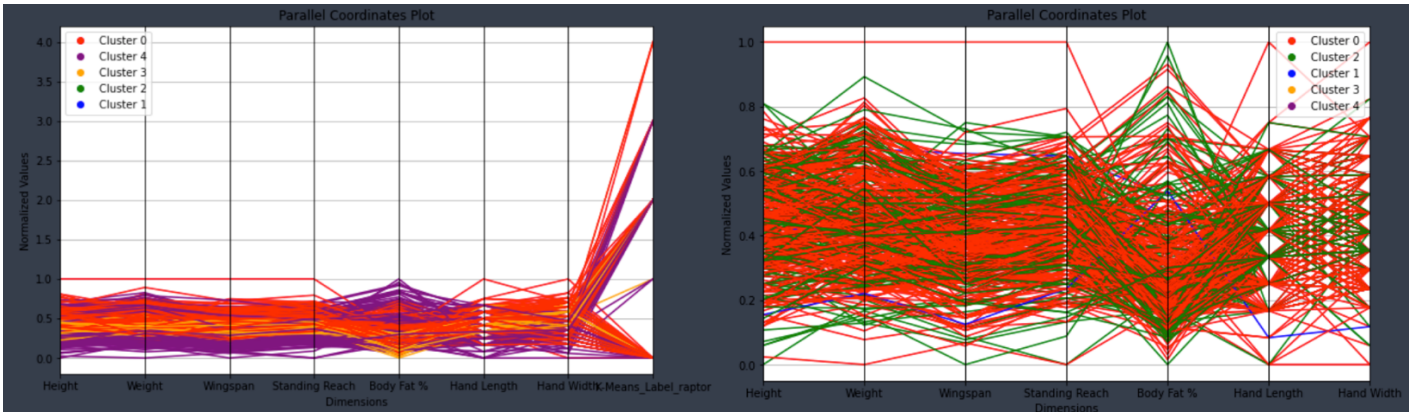
K-Means Label - raptor	Standing Vertical Leap	Max Vertical Leap	Lane Agility Time	Modified Lane Agility Time	Three Quarter Sprint
0	29.340	35.347	11.354	3.140	3.281
1	29.000	34.500	11.305	3.185	3.335
2	29.987	36.288	11.212	3.117	3.230
3	31.750	37.000	11.660	3.230	3.180
4	31.429	37.429	11.110	3.161	3.240

Drill Cluster Means by RAPTOR Label

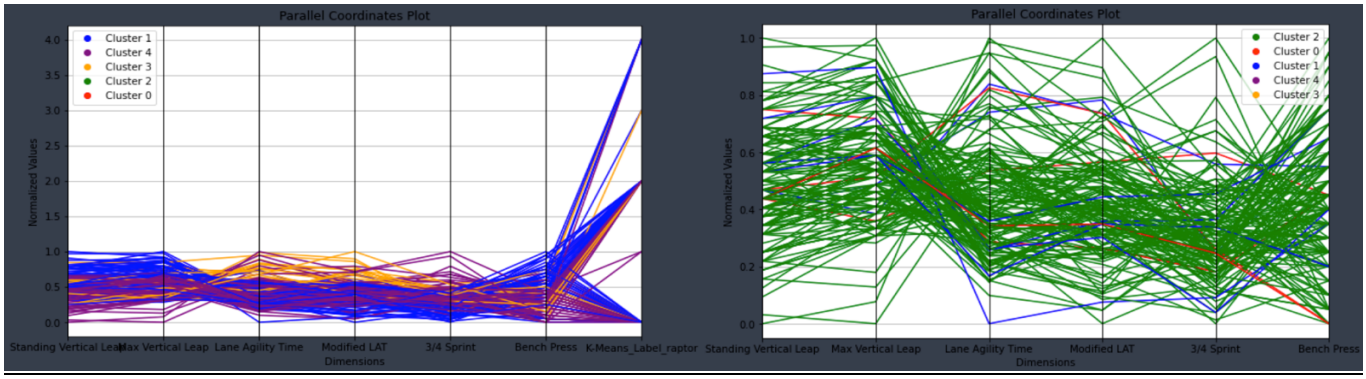
K-Means Label - raptor	Height Without Shoes	Wingspan	Standing Reach	Body Fat %	Hand Length	Hand Width
0	77.358	82.320	103.043	6.763	8.696	9.355
1	76.333	82.917	103.833	8.483	8.667	8.917
2	77.470	82.593	103.101	7.139	8.765	9.374
3	76.795	82.545	103.000	7.041	8.705	9.182
4	79.750	84.583	105.833	6.200	8.917	9.083

Anthropometric Cluster Means by RAPTOR Label

An additional visualization to summarize results of the clustering approach is a parallel coordinates plot, where each colored line depicts a labeled point, and each vertical line depicts a different dimension. Using this plot, cluster patterns can be observed across different dimensions in a single view.



Parallel Coordinate Plots for Anthropometric data (Draft Labels - left & RAPTOR labels - right)



Parallel Coordinate Plots for Draft data (Draft Labels - left & RAPTOR labels - right)

DATA VISUALIZATIONS

Multiple Linear Regression

One of the objectives of **RQ1** was to build a regressor to predict Wins Above Replacement (WAR) for the playoffs based on regular statistics such points per game, assists per game, offensive and defensive rebounds per game, blocks per game, steals per game, turnovers per game, player's age, and WAR for the regular season. One of the regressors built to predict this advanced statistic is a Multiple Linear Regression model. Before going into visualizations depicting results of the model, reiterating key R-Squared values for each independent variable from the exploratory data analysis document:

VAR	**R^2**
PLAYER_AGE	0.004765
FG_PCT	0.0048505
FG3_PCT	0.0021361
FT_PCT	0.0076888
PPG	0.109057
APG	0.0891331
OREBPG	0.0093362
DREBPG	0.0798548
STLPG	0.0828329
BLKPG	0.025358
TOVPG	0.0902862
war_reg_season	0.292346

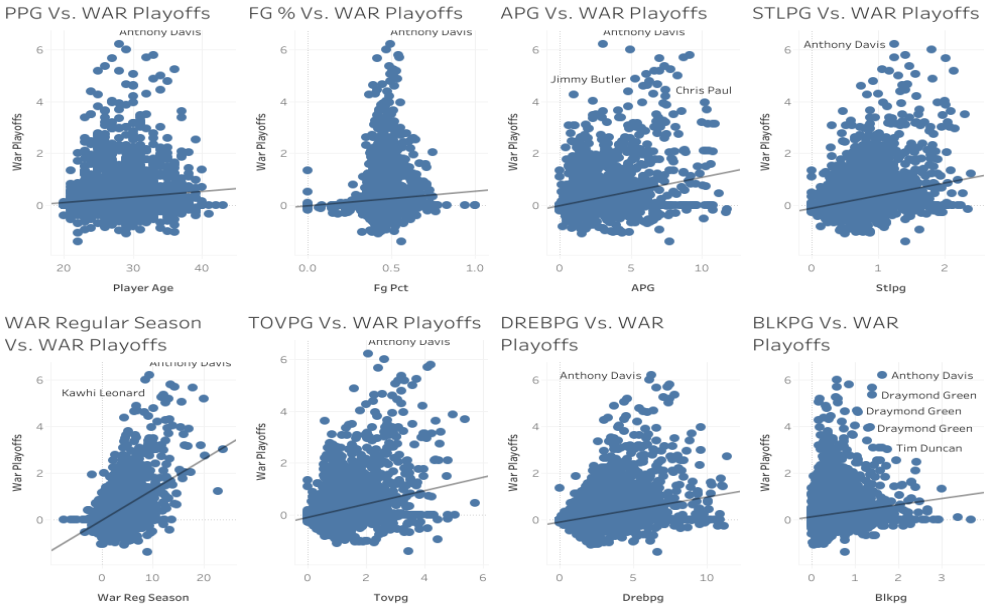


Fig 1.1 – Independent Variables Vs. WAR Playoffs

After visualizing these relationships, all combinations of the variables depicted in the table above were used to build a model, after which the model with the highest r2 score was selected. The combination of independent variables that yielded the highest r2 score: PLAYER_AGE (Player's age), FG_PCT (Field Goal %), APG (Assists per game), OREBPG (Offensive Rebounds per game), STLPG (Steals Per Game), BLKPG (Blocks per game), and war_reg_season (WAR regular season).

R-Square Score for the Multiple Linear Regression Model: **0.31225**
Mean Squared Error for the Multiple Linear Regression Model: **0.32163**

Here is the table displaying the coefficients for each independent variable:

VAR	Coefficient
PLAYER_AGE	0.00596
FG_PCT	0.10490
APG	0.01778
OREBPG	-0.04625
STLPG	0.02112
BLKPG	0.11059
War_reg_season	0.11514

In order to evaluate the performance of the model visually, we can use 5 distinct types of plots to do so:

1) Feature Importance
   Based on the coefficients above, a feature importance bar graph can be plotted to visualize the weightage of each feature in the model:

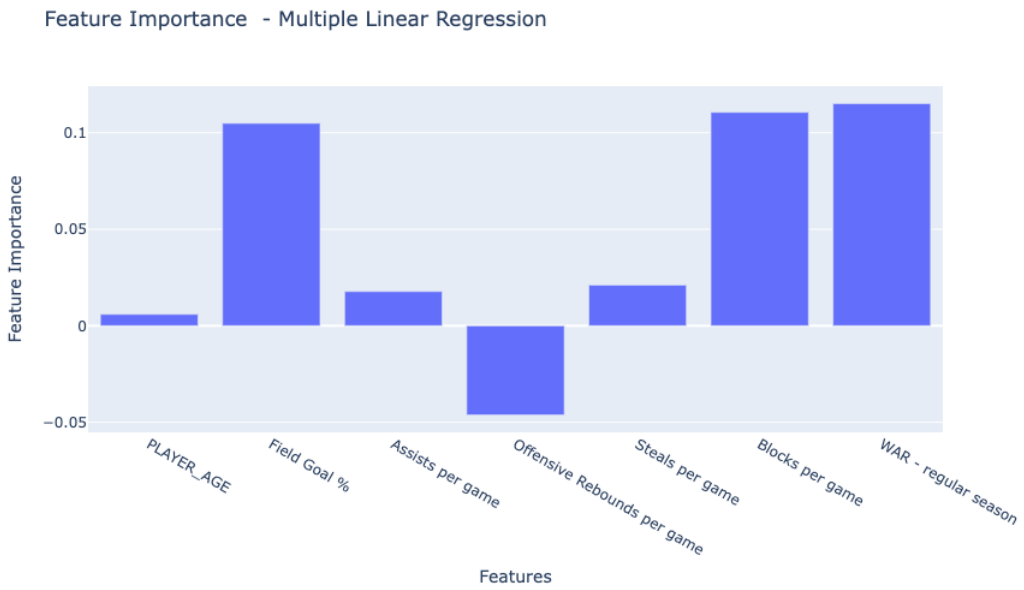


Fig 1.2 – Feature Importance Bar Plot (Multiple Linear Regression)

2) Predicted Vs. Actual WAR Playoff
   Scatterplots can be used to study the relationship between Actual WAR for the playoffs and WAR predicted by the model. Ideally, the points should line up with the diagonal, but here is the plot for this multiple linear regressor:

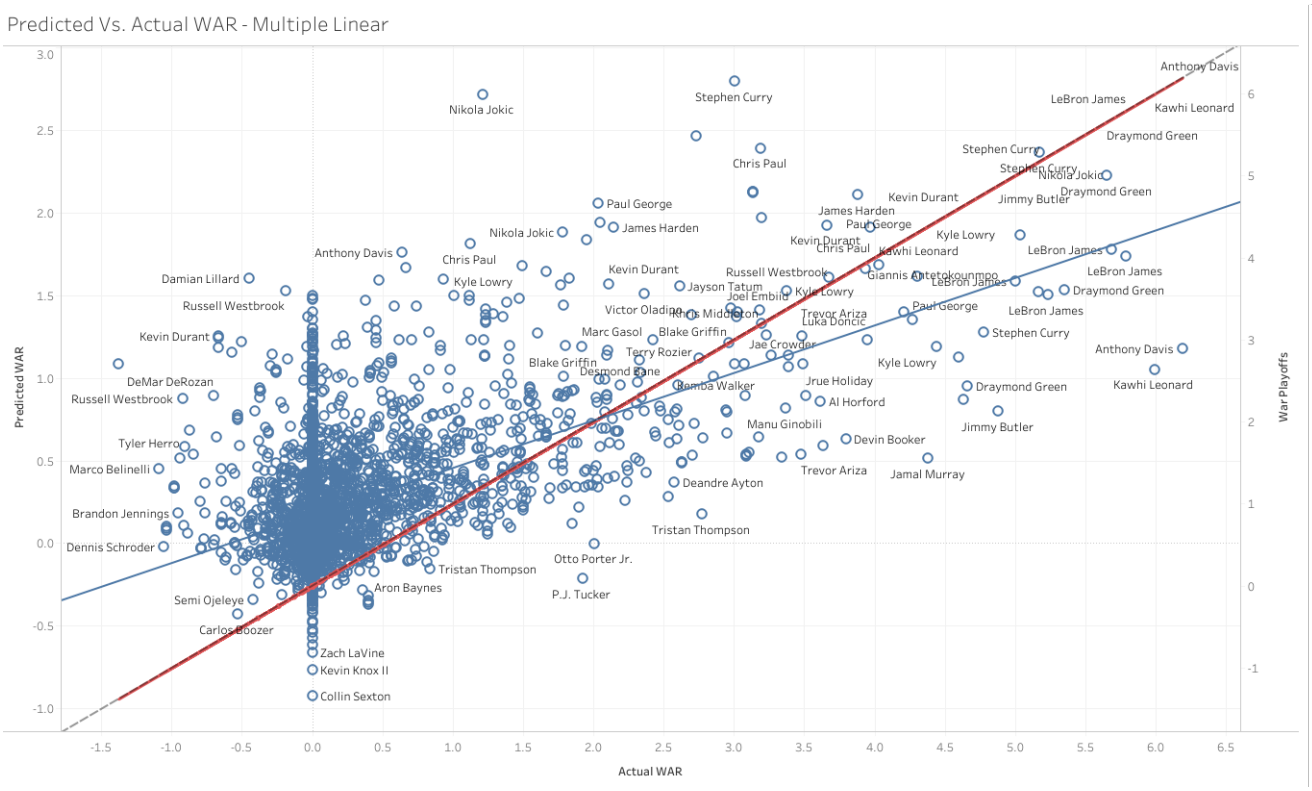


Fig 1.3 – Predicted Vs. Actual WAR (Multiple Linear Regression)

In the figure above, the blue line represents the current linear trend, while the red line represents the constant value of an ideal relationship. An R-Squared score of 0.28607 is obtained between the actual and predicted values.

3) Residual Plots

Branching off of scatterplots for predicted vs. actual WAR, the difference between the predicted and the actual WAR (also known as the residual) can be compared with the independent variables to verify and observe linearity. In addition to study the degree of linearity between the independent variables and the residuals, we can use these plots to look for outliers that may skew the data and affect the model in a certain way. Here are the residual plots for the multiple linear regressor:

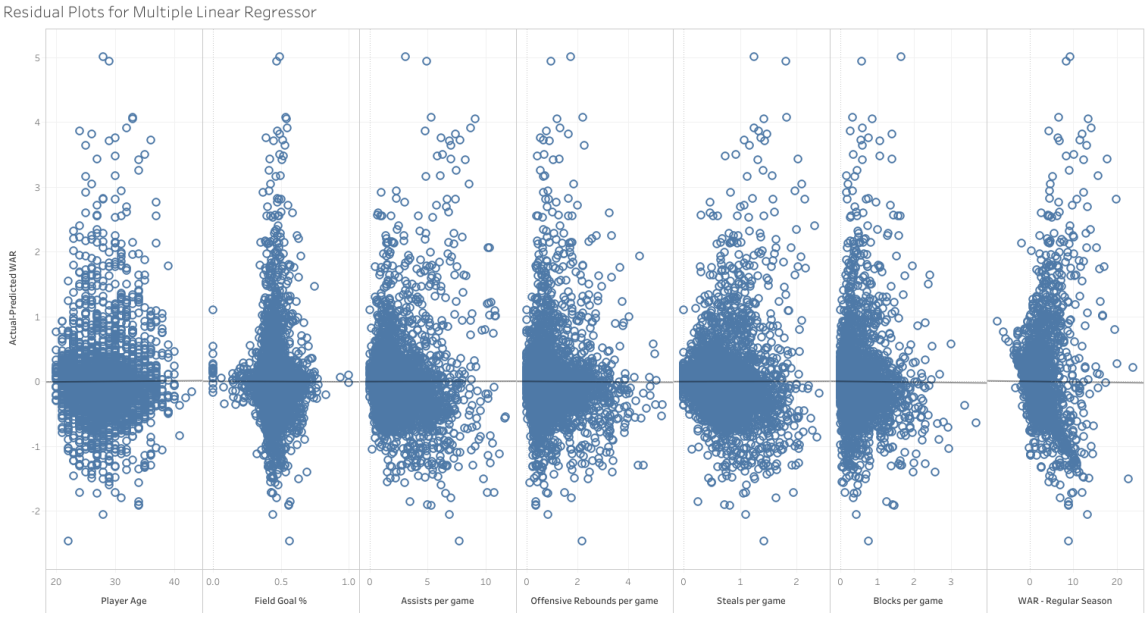


Fig 1.4 – Residual Scatter Plots (Multiple Linear Regression)

As seen in the figure above, residuals around certain values of the independent variables are quite large, indicating the presence of some outliers. While most of these plots have points scattered around the horizon, we saw previously that the R-Squared values for the relationship between the independent and dependent variables was considerably low.

4) Residual Histogram and Q-Q Plots

To study the distribution of the residuals, histograms and q-q plots can be useful. Using these plots the possible departure from a normal distribution can be observed, providing further insights into outliers and skewness.

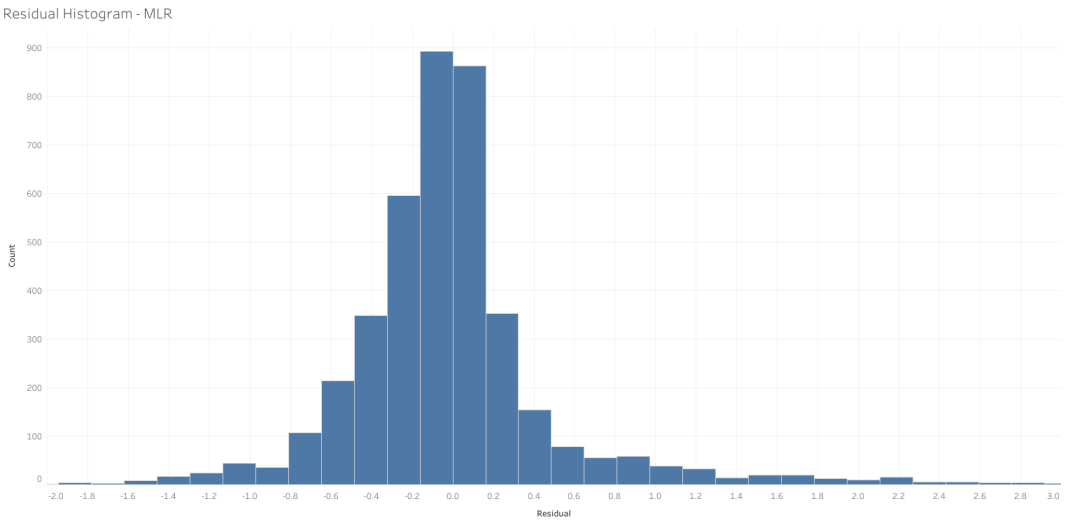


Fig 1.5 – Residual Histogram (Multiple Linear Regression)

The histogram denotes an approximately normal distribution with a slight skew to the right. Most frequent values can be found around -0.3 to 0.3, with the peak in between -0.2 and 0. Now considering a Q-Q plot to take a look at sample and theoretical quantiles to analyze the degree of departure from the normal distribution:

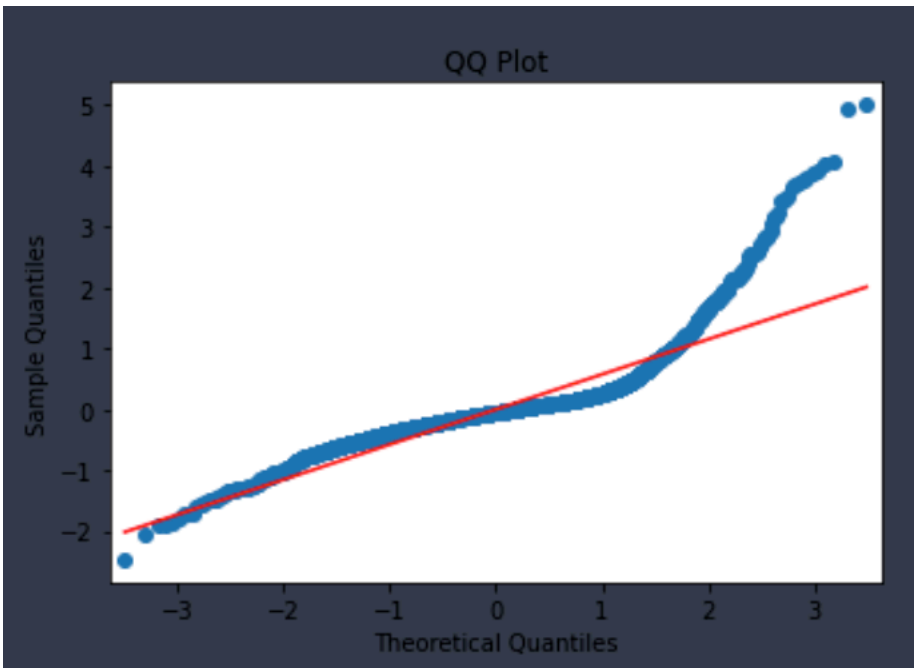


Fig 1.6 – Q-Q Plot (Multiple Linear Regression)

In the Q-Q plot above, a stark diversion from the normal is noticed towards the right tail showing non-normality in this region of the data. On the same side, significant outliers can also be noticed situated far above the red line.

Random Forest Regression

Based on the lack of linearity of correlation between the independent and dependent variables, it was stated in the methodology that a Random Forest Regressor would possibly be more capable of capturing more complex relationships between these two sets of variables. While the random forest regressor lacks interpretability through simple coefficients (as in the case of linear regression), it seems to handle the complexity of interaction quite well and takes better to outliers than a multiple linear regressor. As you will see in these visualizations, the Random Forest Regressor seems to have performed much better than the multiple linear regressor.

Similar to the approach taken to find the highest r2 score for the multiple linear regression model, all combinations of the independent variables depicted in the table above were used to build a model, after which the model with the highest r2 score was selected. The combination of independent variables that yielded the highest r2 score for the forest regressor are PLAYER_AGE (Player's age), FG3_PCT (3-Point Field Goal %), APG (Assists per game), OREBPG (Offensive Rebounds per game), DREBPG (Defensive Rebounds per game), BLKPG (Blocks per game), TOVPG (Turnovers Per Game), and war_reg_season (WAR regular season).

R-Square Score for the Random Forest Regression Model ≈ **0.45816**
Mean Squared Error for the Random Forest Regression Model ≈ **0.26432**

Here is the table displaying the feature importance for each independent variable:

VAR	**Feature Importance**
PLAYER_AGE	0.06276
FG3_PCT	0.06758
APG	0.08518
OREBPG	0.09841
DREBPG	0.08789
BLKPG	0.08555
TOVPG	0.09021
war_reg_season	0.42241

Using the similar types of plots, here are the visualized results of the Random Forest Regressor to predict WAR for the playoffs:

1) Feature Importance
   Based on the coefficients above, a feature importance bar graph can be plotted to visualize the weightage of each feature in the model:

Fig 2.1 – Feature Importance Bar Plot (Random Forest Regression)

Similar to the multiple linear regressor, highest importance is placed on WAR – regular season.

2) Predicted Vs. Actual WAR Playoff

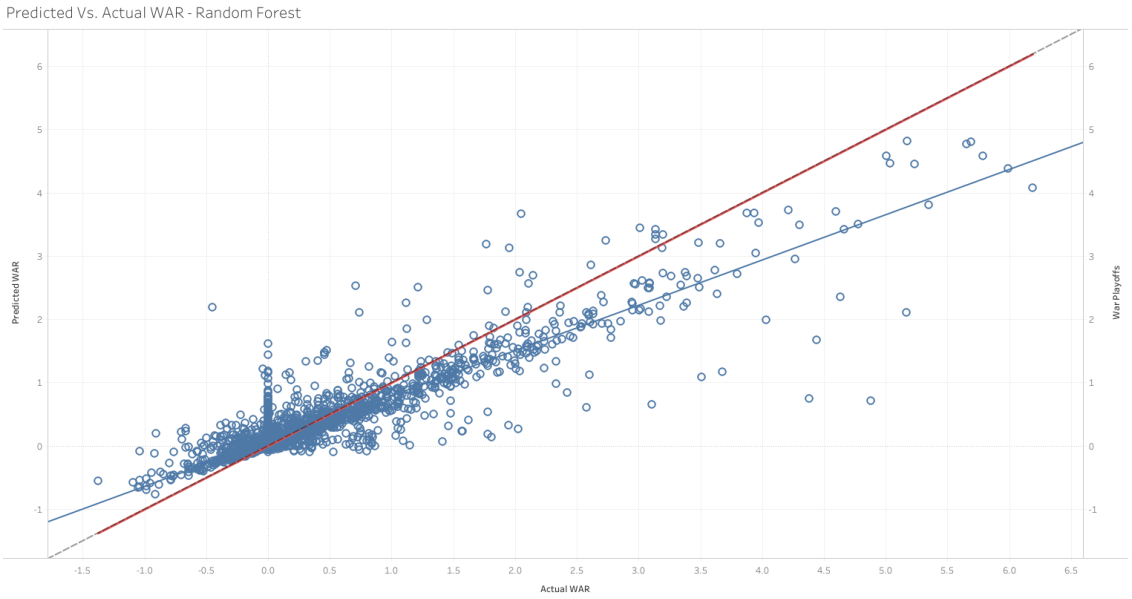


Fig 2.2 – Predicted Vs. Actual WAR (Random Forest Regression)

Compared to the multiple linear regressor, it is evident that points are more closely aligned with the diagonal (red line). An R-Squared score of 0.82878 is obtained between the predicted and actual values.

3) Residual Plots

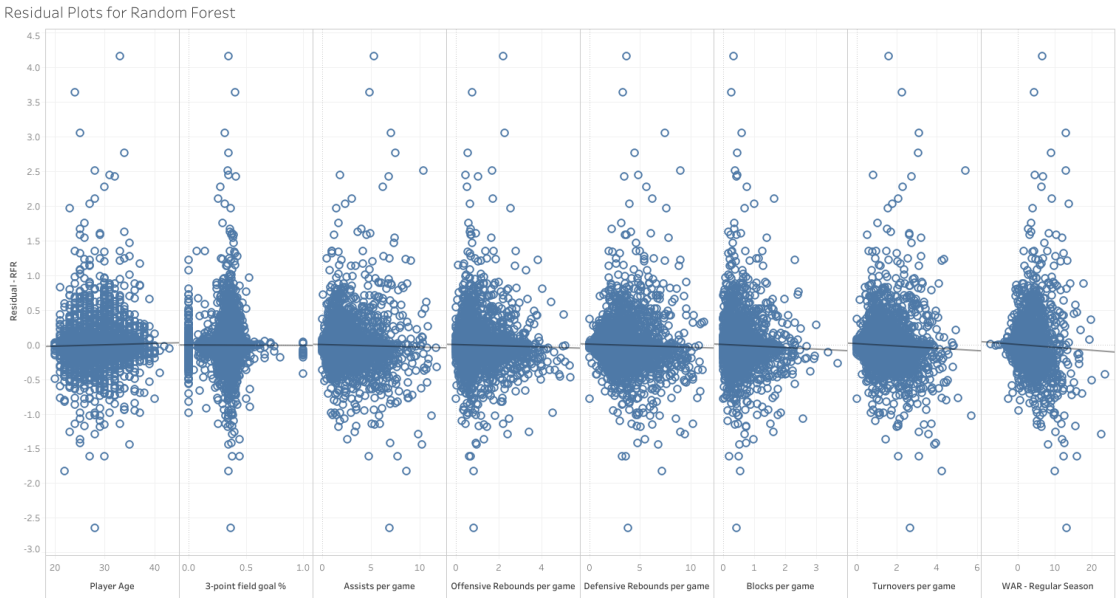


Fig 2.3 – Residual Scatter Plots (Random Forest Regression)

Similar to independent variables used to build the multiple linear regressor, high scatter with several outliers is seen in the residual plots above. However, most of the points continue to be concentrated around the horizon.

4) Residual Histogram and Q-Q Plots

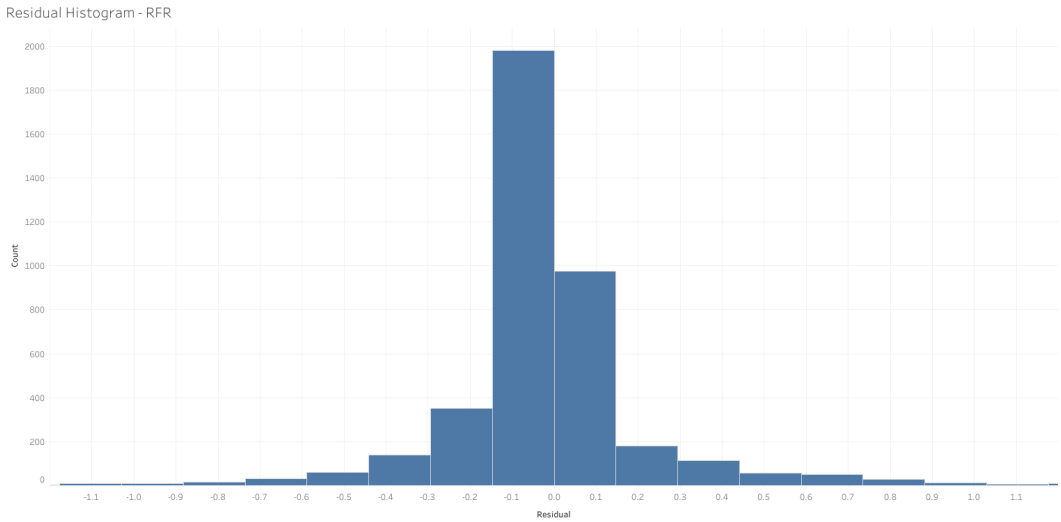


Fig 2.4 – Residual Histogram (Random Forest Regression)

Reflecting similar findings seen on the Predicted Vs. Actual plot, the peak of the histogram lies between -0.1 and 0. There does seem to be a notable skew in either direction (maybe a slight skew to the right).

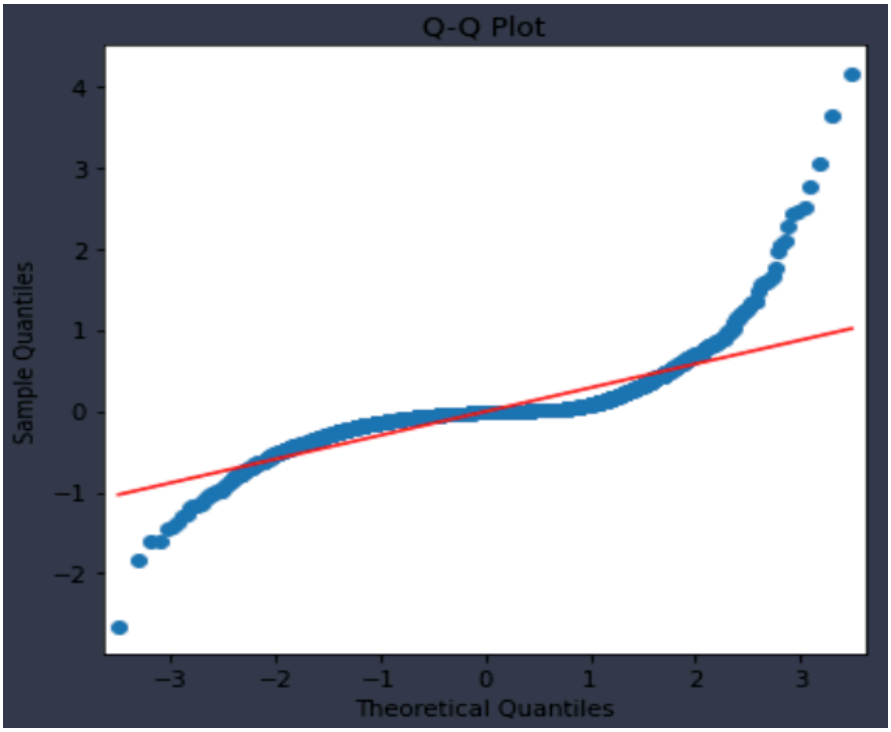


Fig 2.5 – Q-Q Plot (Random Forest Regression)

The Q-Q plot above reflects the central peak seen in the histogram. As opposed to the Q-Q plot for the multiple linear regressor, both the tails deviate from the middle. This also shows a better fit to a normal distribution for this set of residuals.

K-Means Clustering

For the exploratory data analysis segment of **RQ2**, performance in the draft combine was assessed along with the player's RAPTOR score for their first year in the NBA. We worked with two data frames – draft combine drills, which consisted of athletic movements (vertical leap, bench press, 3/4th sprint, etc.), and draft combine anthropometrics, which consisted of bodily measures (height, weight, wingspan, etc.). Through scatterplots and R-Squared analysis, it was seen that most of these variables had a weak linear relationship with RAPTOR scores. As our aim was to explore the relationship between draft scores and RAPTOR measures, a K-Means clustering approach was considered to extract possible trends between 'similar' players in the draft and in the NBA. Before visualizing the results of the cluster analysis, reiterating R-Square values for both datasets:

	RAPTOR Offense	RAPTOR Defense	RAPTOR Total
Standing Vertical Leap	0.002738	0.000830	0.0038976
Max Vertical Leap	4.917e-05	0.000247	0.000118
Lane Agility Time	0.001939	9.591e-05	0.000369
Mod. Lane Agility Time	0.000544	0.000286	1.777e-06
3/4th Sprint	0.000896	0.007835	0.0113121
Bench Press	0.001073	0.000368	0.0016192

Table 3.1 – R-Squared values (Draft Combine Drills)

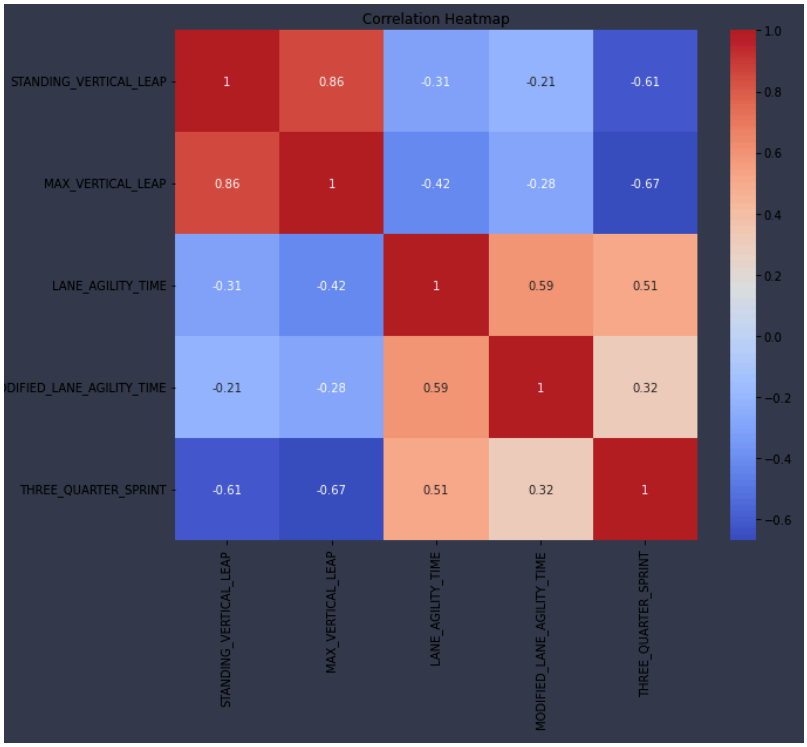


Fig 3.1 – Correlation Heatmap (Drill)

	RAPTOR Offense	RAPTOR Defense	RAPTOR Total
Height	0.017641	0.003835	0.001945
Standing Reach	0.021182	0.001748	0.004746
Body Fat %	0.001147	0.004684	0.005979
Hand Length	0.017363	9.287e-06	0.008100
Hand Width	0.002595	0.001550	2.123e-05
Wingspan	0.028193	0.000356	0.010521

Table 3.2 – R-Squared values (Draft Combine Anthropometrics)

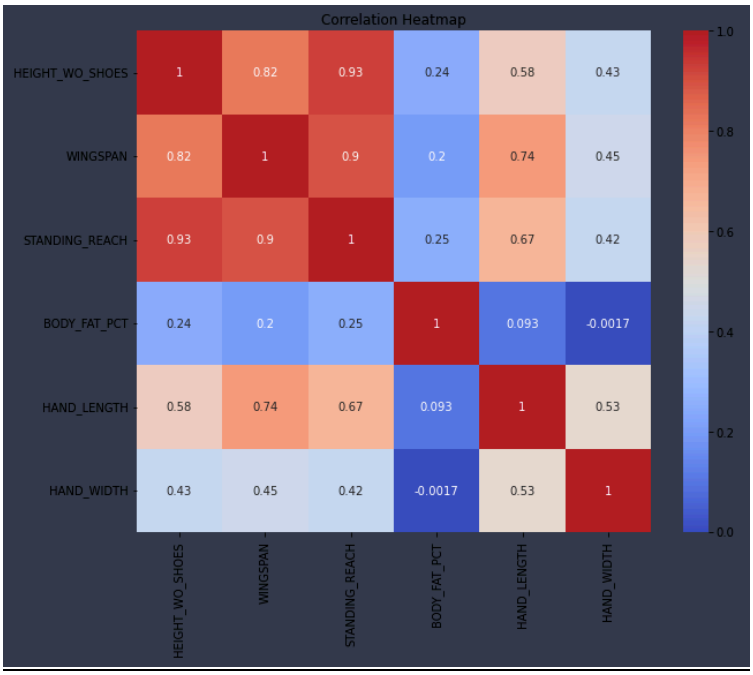


Fig 3.2 – Correlation Heatmap (Anthropometrics)

For each dataset, each record was labelled as 1 out of 5 groups (clusters). This clustering was performed based on these features for the draft combine drill dataset: STANDING_VERTICAL_LEAP, MAX_VERTICAL_LEAP, LANE_AGILITY_TIME, MODIFIED_LANE_AGILITY_TIME, THREE_QUARTER_SPRINT, BENCH_PRESS. For the draft combine anthropometric dataset, clustering was performed on the following features: HEIGHT_W_SHOES, WEIGHT, WINGSPAN, STANDING_REACH, BODY_FAT_PCT, HAND_LENGTH, HAND_WIDTH. Additionally, for each dataset, another round of clustering and a separate set of labels were assigned to each row based on the 'raptor_offense' and 'raptor_defense' features.

Beginning with the draft combine drill dataset, we can use color scatter plots to visualize the clusters assigned based on the independent variables and raptor offense and defense. Positions such as point guard, shooting guard, center, small forward, and power forward were also represented to check if cluster contained the same positions.

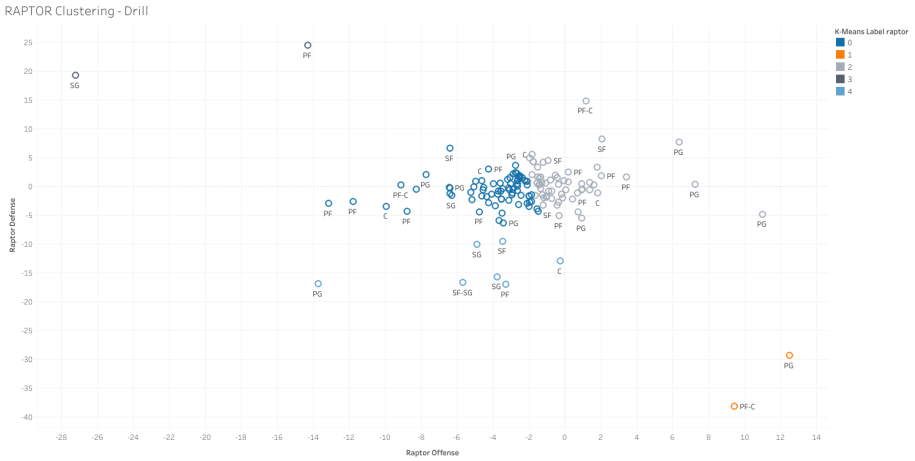


Fig 3.3 – RAPTOR Scatter Plot with Clusters (Drill)

Clusters can also be visualized for each pair of features:

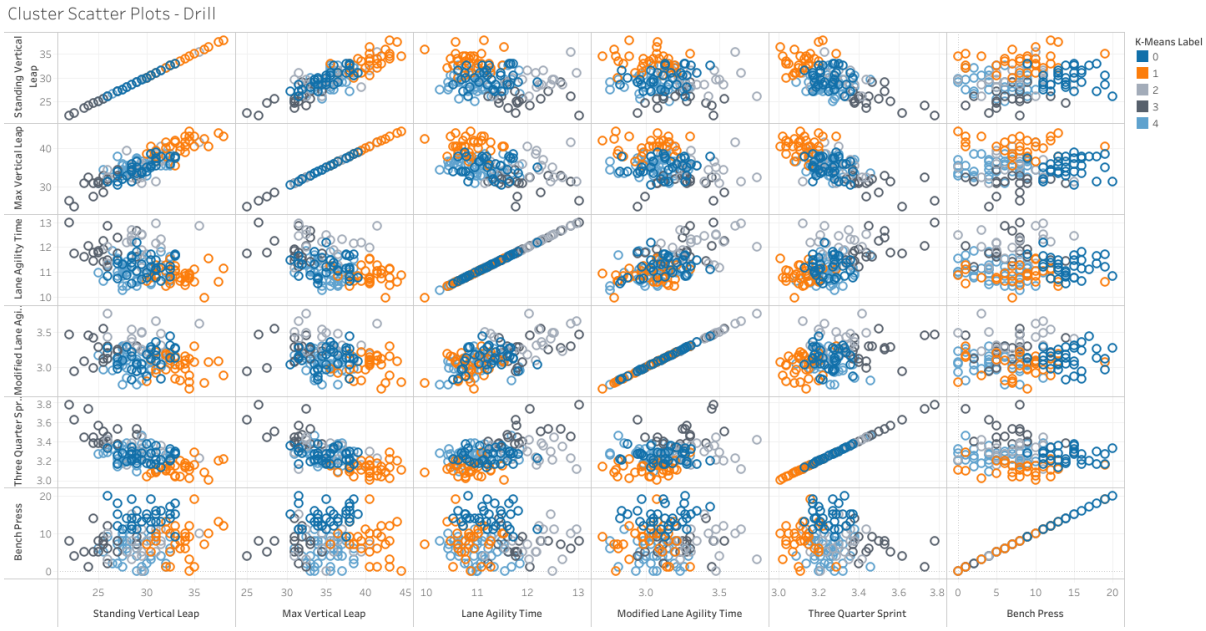


Fig 3.4 – Scatter Plots with Clusters (Drill)

Additionally, we can also look at tables which display the means of each field based on cluster labels for the independent variables, as well as clusters for offensive and defensive RAPTOR.

K-Means Label	Standing Vertical Leap	Max Vertical Leap	Lane Agility Time	Modified Lane Agility Time	Three Quarter Sprint
0	29.879	35.258	11.224	3.123	3.249
1	33.484	40.645	10.853	3.017	3.122
2	29.632	35.342	12.106	3.396	3.291
3	25.417	30.889	11.932	3.238	3.483
4	28.790	35.136	11.027	3.067	3.252

Table 3.3 – Cluster Means by Features (Drill)

K-Means Label - raptor	Standing Vertical Leap	Max Vertical Leap	Lane Agility Time	Modified Lane Agility Time	Three Quarter Sprint
0	29.340	35.347	11.354	3.140	3.281
1	29.000	34.500	11.305	3.185	3.335
2	29.987	36.288	11.212	3.117	3.230
3	31.750	37.000	11.660	3.230	3.180
4	31.429	37.429	11.110	3.161	3.240

Table 3.4 – Cluster Means by RAPTOR (Drill)

Using the same set of visualizations to look at draft anthropometric clustering:

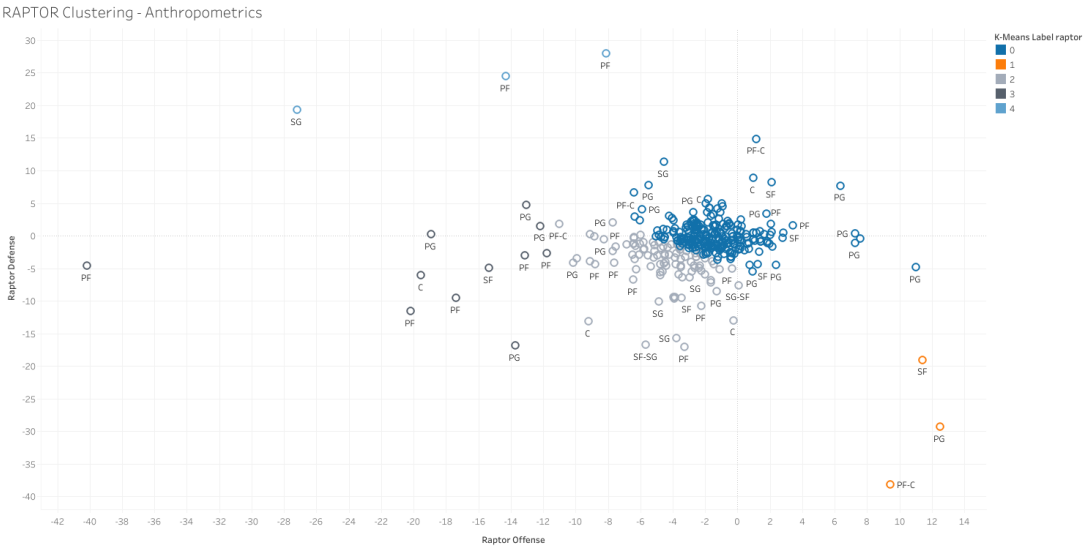


Fig 3.5 - RAPTOR Scatter Plot with Clusters (Anthropometric)

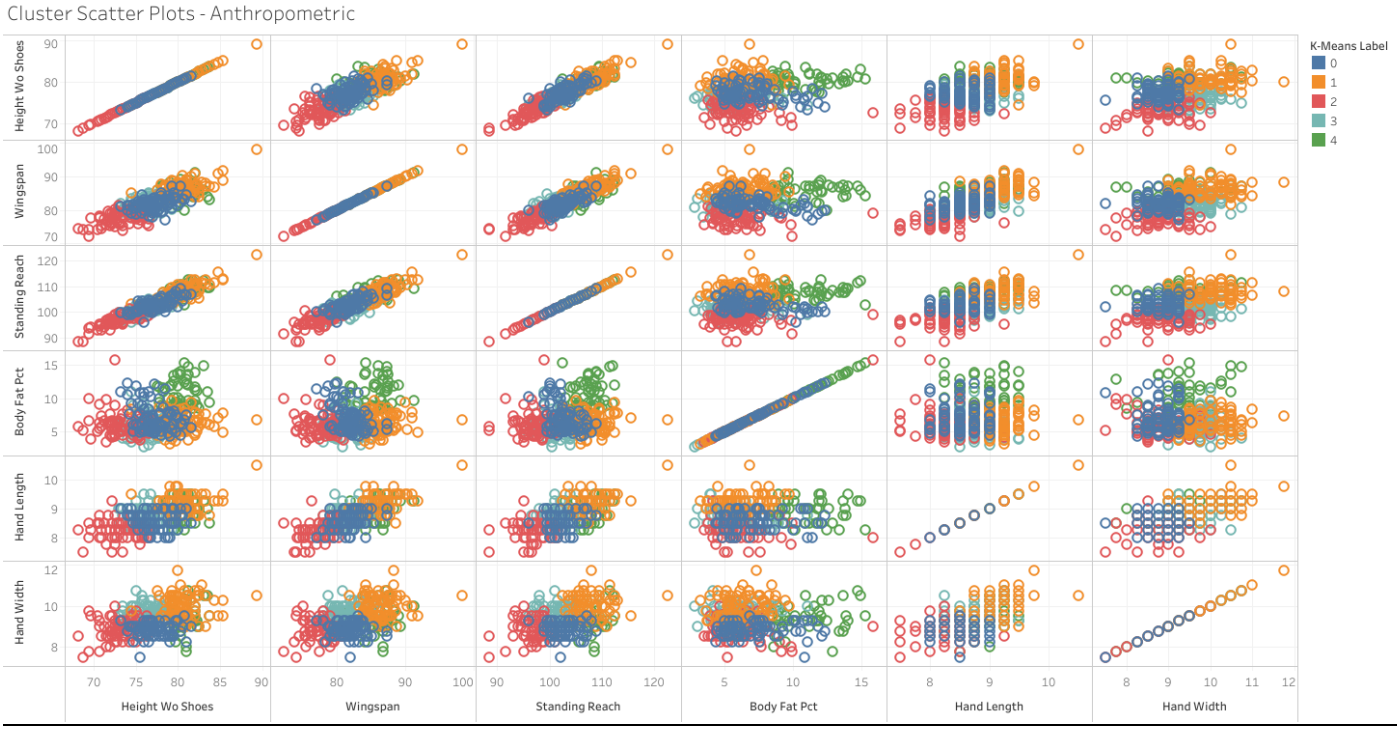


Fig 3.6 – Scatter Plots with Clusters (Anthropometrics)

K-Means Label	Height Without Shoes	Wingspan	Standing Reach	Body Fat %	Hand Length	Hand Width
0	77.199	81.795	102.614	7.034	8.560	8.840
1	80.583	86.494	108.065	6.256	9.176	9.985
2	73.283	77.617	97.080	5.991	8.233	8.797
3	76.216	81.534	101.263	5.786	8.734	9.691
4	80.511	85.441	107.713	11.014	8.910	9.410

Table 3.5 - Cluster Means by Features (Anthropometrics)

K-Means Label - raptor	Height Without Shoes	Wingspan	Standing Reach	Body Fat %	Hand Length	Hand Width
0	77.358	82.320	103.043	6.763	8.696	9.355
1	76.333	82.917	103.833	8.483	8.667	8.917
2	77.470	82.593	103.101	7.139	8.765	9.374
3	76.795	82.545	103.000	7.041	8.705	9.182
4	79.750	84.583	105.833	6.200	8.917	9.083

Table 3.4 – Cluster Means by RAPTOR (Anthropometric)

ETHICAL RECOMMENDATIONS

Basketball, like many other sports, is a game determined by an unfathomable number of variables. Besides tangibles such as the quality of players, coaches, staff, and each player's recent performances, a game in the National Basketball Association is always determined to swing either way. Analytics within sports began with the simple practice of collecting data and noticing trends in player's and team's performances. These trends can then be incorporated into business decision made by major sports organizations to not only develop strategies, improve players, and diversify teams, but to build successful and winning organizations through the draft and trading for players. Besides being used within athletics itself, analytics also serves as fuel to sports betting, where individuals bet on the performance team/player or the outcome of a game with the hopes of making a profit. Considering these multiple avenues which use predictive analytics in sports, there are a few ethical recommendations to consider.

Predictive analytics is already a huge part of decisions that are being made by teams today in the NBA. Based on predictions of player's performances or team's strategies, organizations can prepare themselves for an upcoming season or playoff series. Well renowned organizations such as FiveThirtyEight (source of data for this capstone project), are at work to building more accurate predictors within basketball and other sports. While these endeavors strive for a theoretical ceiling, it is important to note that no model in sports will ever be accurate, simply because there are still values that cannot be measured. For example, mindset, mental preparation, team and player chemistry, pressure from other extenuating circumstances, and health of players (possible even a stroke of luck) still seem to be intangible variables for the most part. While these intangibles exist, no model will ever truly predict the outcome of a game. Additionally, a highly accurate predictor can potentially deplete the value that goes into and being produced by a game, which is ideally played while keeping the ideas of fairness and sportsmanship alive. Models that start to predict deeper aspects of the game can have an effect on the 'joy' of the game itself.

Coming to the way player's use stats, predictive analytics can end up backfiring. Take for example a player sees a prediction from a highly accurate model, saying that they will not have a great playoff series. How does the player incorporate this 'feedback'? Do they simply give up based on this prediction or do they continue to play with a free mind as they should have in the first place? Conversely, a player predicted to have a brilliant playoff run, may possibly end up under-performing due to the pressure of having to produce at the predicted level. Similarly, do teams decide to not play a player due to a bad prediction, instead of giving them a chance, because they are more than just a number? These are ethical questions to keep in mind as entities strive towards building more reliable predictors.

Sports betting involves placing sums of money on specific outcomes of a game, a series, a player's performance, or other aspects of a game. Analytics are currently used to measure 'odds' that define the likelihood of a particular outcome. Once models gain a certain level of accuracy, ethical guidelines on betting will have to monitored and finetuned very carefully, as bettors are bound to place a higher or lower amount of money based on the model's predictions. Since this aspect of sports involves individual's own resources, ethical recommendations will have to be evaluated more closely.

From a technical perspective with regards to building this particular model, only data from the 2014 NBA season was used. This is due to the availability of accurate RAPTOR statistics form this season onwards. Considering that the NBA extended far before 2014, there are bound to be certain biases and skews within the training data of the model. This is certainly one Data to Algorithm Bias to be aware of, when using this regressor. More specifically, we may see cases of representation bias (as the data is only representative of stats from a specific timeframe), aggregation bias (outcomes for a specific sub-group cannot necessarily be extrapolated), and possible longitudinal data fallacy, with player's performance changing over time. These biases will also have to be checked for as Algorithm to User Biases as well. There may be instances of Popularity bias, with some star players playing several more minutes and games, enunciating their results. Most of the ethical recommendations specified in prior paragraphs cover certain aspects of User Interaction Bias, or how stakeholders, organizations and athletes will take to such models.

CHALLENGES

Beginning with the process of identifying a data source, the 'nba_api' package was a great tool to use, but it required calls that needed to be made to the NBA website in order to pull data. The process of pulling data for around 4000 players became quite cumbersome due to several halts and errors that were faced during these calls/requests to the NBA website. Iterative approaches in the project, that would assimilate data for several players had to be done in segments, instead of running one big loop over the entire list of players. While we are on the topic of data, there are many other advanced statistics that are there to choose from. In fact, a few of these stats such as Daily Plus Minus and Estimated Plus Minus are said to be quite accurate as well. While these stats did not have much of a database, FiveThirtyEight provided great resources on their GitHub page with all the relevant stats presented as csvs.

Another major challenge faced was in building a regressor that produced an R-Squared score of higher than 0.5. Due to the high degree of variability in the game of basketball, as well as ambiguity around the construction of the WAR statistic, further finetuning beyond feature selection was not very feasible. Identifying a suitable regressor was a challenge with regards to the research questions. With regards to the clustering approach, the challenge lay in interpreting trends and patterns that lay within large groups of clusters. Some of the visualizations, while useful, can prove to be noisy.

Coming back to the data, only players from the 2014 season onwards was used in order to simplify the pre-processing step and to use the most accurate data available. While data from the 1970s was available, player tracking only began in 2014 (an important ingredient in calculating RAPTOR). Perhaps this sliced portion of the data missed some key observations, which lead to biases in the model.

RECOMMENDATIONS AND NEXT STEPS

This project can be carried forward in several different ways. Personally, I would recommend attempting to use different types of regression algorithms such as logistic regression, KNN models, support vector machines, decision trees, and perhaps neural network regression to predict playoff performance. With the random forest regressor itself, hyperparameter finetuning can also be carried out to produce a model with a higher R-Squared score.

A clustering approach is useful to find similarities, but maybe the features upon which clustering is performed can be selected more carefully. Separate clustering can be carried out based on offensive and defensive statistics. Hypothesis can be formulated before moving into clustering to test relationships between data. For example, one can hypothesize that a player with a good lane agility score in the combine may attain a good defensive rating in the NBA.

With regards to the data, other all-encompassing stats such as daily plus-minus and estimated plus-minus can be used instead of RAPTOR. These stats may provide more insights into predicting playoff performance as well as the feasibility of doing so.

SOURCE CODE

RAPTOR Analysis (Data from FiveThirtyE

In [1]:
```
import pandas as pd
import plotly.express as px
```

Historical RAPTOR Analysis

In [2]:
```
rap_hist = pd.read_csv('historical_RAPTOR_by_player.csv')
```

In [3]:
```
rap_hist
```

	player_name	player_id	season	poss	mp	raptor_offense	raptor_defense	raptor_tot
0	Alaa Abdelnaby	abdelal01	1991	640	303	-3.938450	-0.510076	-4.448526
1	Alaa Abdelnaby	abdelal01	1992	1998	959	-2.553849	-0.197943	-2.751792
2	Alaa Abdelnaby	abdelal01	1993	2754	1379	-2.373736	-2.069808	-4.443544
3	Alaa Abdelnaby	abdelal01	1994	320	159	-6.140056	-2.748312	-8.888368
4	Alaa Abdelnaby	abdelal01	1995	984	506	-3.846543	-1.268012	-5.114556
...	...	...	...	...	...	...	...	...
19154	Ivica Zubac	zubaciv01	2018	871	410	-2.903709	2.688832	-0.214877
19155	Ivica Zubac	zubaciv01	2019	2345	1079	-2.362444	1.813768	-0.548676
19156	Ivica Zubac	zubaciv01	2020	3447	1646	1.099849	3.549458	4.649308
19157	Ivica Zubac	zubaciv01	2021	3908	1910	-0.909039	2.525735	1.616696
19158	Ivica Zubac	zubaciv01	2022	3786	1852	-0.693765	0.723358	0.029594

19159 rows × 15 columns

Modern RAPTOR Analysis

In [4]:
```
rap_modern = pd.read_csv('modern_RAPTOR_by_player.csv')
```

In [5]:
```
rap_modern_ord = rap_modern.sort_values(by = 'raptor_total', ascending = F
rap_modern_ord
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	ra
3023	Naz Mitrou-Long	mitrona01	2018	3	1	47.557637	7.313599	54
1834	Udonis Haslem	hasleud01	2021	7	3	17.508430	33.030435	50
4275	Tyler Ulis	ulisty01	2019	2	1	-4.461871	57.305705	52
493	Marques Bolden	boldema01	2020	7	3	-2.460983	36.336087	33
4054	Max Strus	strusma01	2020	15	6	9.913626	6.490655	16
...	...	...	...	...	...	...	...	...
2718	Will Magnay	magnawi01	2021	7	3	-28.675785	-13.528044	-4
1976	John Holland	hollajo02	2016	2	1	-40.303496	-21.371025	-6
3412	Anzejs Pasecniks	pasecan01	2021	13	6	-44.186275	-15.089827	-5
2967	CJ Miles	milescj01	2022	5	2	-3.351891	-48.188577	-5
1096	Sam Dekker	dekkesa01	2022	2	1	-20.286107	-20.124121	-4

4685 rows × 21 columns

In [6]:
```
rap_modern
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	r
0	Alex Abrines	abrinal01	2017	2387	1135	0.745505	-0.372938	0
1	Alex Abrines	abrinal01	2018	2546	1244	0.317549	-1.725325	-
2	Alex Abrines	abrinal01	2019	1279	588	-3.215683	1.078399	-
3	Precious Achiuwa	achiupr01	2021	1581	749	-4.122966	1.359278	-
4	Precious Achiuwa	achiupr01	2022	3802	1892	-2.521510	1.763502	-
...	...	...	...	...	...	...	...	..
4680	Ivica Zubac	zubaciv01	2018	871	410	-2.581325	1.643240	-
4681	Ivica Zubac	zubaciv01	2019	2345	1079	-2.336006	1.400274	-
4682	Ivica Zubac	zubaciv01	2020	3447	1646	0.516422	3.529428	4
4683	Ivica Zubac	zubaciv01	2021	3908	1910	-0.371753	2.960613	2
4684	Ivica Zubac	zubaciv01	2022	3786	1852	-0.920033	1.986362	1

4685 rows × 21 columns

In [7]:
```
fig = px.scatter(x=rap_modern['mp'], y=rap_modern['raptor_total'])
fig.update_layout(
    title='RAPTOR Total Vs. Minutes Played',
    xaxis_title='Minutes Played',
    yaxis_title='RAPTOR Total'
)
fig.show()
```

In [8]:
```
fig = px.scatter(x=rap_modern['poss'], y=rap_modern['raptor_total'])
fig.update_layout(
    title='RAPTOR Total Vs. Possessions Played',
    xaxis_title='Possessions Played',
    yaxis_title='RAPTOR Total'
)
fig.show()
```

In [9]:
```
fig = px.bar(rap_modern, x='player_name', y='raptor_box_total')
fig.show()
```

In [10]:
```
steph_data_2016 = rap_modern[(rap_modern['player_name'] == 'Stephen Curry'
steph_data_2016
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	r
1027	Stephen Curry	curryst01	2016	7053	3314	10.532883	1.688492	1

1 rows × 21 columns

Analysis of Leaders in the Modern RAPTOR Dataset (2014

In [11]:
```
rap_modern['mp'].mean()
```

```
1186.7069370330844
```

The average number of minutes played in the modern RAPTOR database is 1186. T
benchmark of 1000 minutes played, and assess RAPTOR stats for all modern playe
minutes per season. Filtering by this benchmark:

In [18]:
```
rap_modern_tot = rap_modern[rap_modern['mp'] >= 1000].sort_values(by = 'ra
rap_modern_tot
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	r
2304	Nikola Jokic	jokicni01	2022	5481	2647	8.443342	6.293564	1
1027	Stephen Curry	curryst01	2016	7053	3314	10.532883	1.688492	1
3428	Chris Paul	paulch01	2014	5387	2643	7.845314	3.246182	1
1026	Stephen Curry	curryst01	2015	7192	3439	8.504968	1.677165	1
1744	James Harden	hardeja01	2019	6728	3291	9.959259	1.792524	1
3429	Chris Paul	paulch01	2015	6633	3302	7.508889	2.398886	9
1745	James Harden	hardeja01	2020	6253	2931	9.543226	1.219780	1
1743	James Harden	hardeja01	2018	6466	3172	9.390432	1.451237	1
2578	Kawhi Leonard	leonaka01	2016	5365	2719	5.487882	4.530604	1
2582	Kawhi Leonard	leonaka01	2020	4919	2359	6.273804	3.200929	9

10 rows × 21 columns

Above, we see the top 10 players ranked by overall RAPTOR. Jokic (two-time back
while James Harden has the most number of RAPTOR leading seasons in the top 1

In [20]:
```
rap_modern_off = rap_modern[rap_modern['mp'] >= 1000].sort_values(by = 'ra
rap_modern_off
```

_defense	raptor_total	war_total	war_reg_season	war_playoffs	predator_offense	predator_defe
47	12.487858	26.666873	23.659318	3.007555	10.635103	2.712822
34	10.715281	22.791368	20.057946	2.733422	10.893131	1.676179
77	9.212807	20.653304	15.619612	5.033692	9.791555	0.593993
04	10.067769	20.926163	17.048015	3.878148	9.434385	1.615119
50	6.799651	10.760575	10.760575	0.000000	8.087152	-1.656422
71	11.038262	25.089873	19.914785	5.175088	8.787715	2.657047
21	9.249489	17.533917	15.756031	1.777885	8.685700	1.018531
03	10.669395	22.609142	19.418015	3.191128	8.772725	2.137896
06	14.572411	23.905554	22.693872	1.211682	8.961989	5.602515
37	10.466843	19.842384	16.705593	3.136792	8.042809	2.728677

Here are the top 10 players ranked by offensive RAPTOR from the 2014 season

In [22]:
```
rap_modern_def = rap_modern[rap_modern['mp'] >= 1000].sort_values(by = 'ra
rap_modern_def
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	r
1563	Rudy Gobert	goberru01	2021	5381	2563	-0.898293	7.147038	6
1564	Rudy Gobert	goberru01	2022	4762	2317	0.158153	6.970416	7
1645	Draymond Green	greendr01	2017	6520	3064	1.167765	6.094466	7
2304	Nikola Jokic	jokicni01	2022	5481	2647	8.443342	6.293564	1
1559	Rudy Gobert	goberru01	2017	5735	2990	0.675235	5.217186	5
1562	Rudy Gobert	goberru01	2020	5355	2603	-0.666003	5.932678	5
1643	Draymond Green	greendr01	2015	6795	3274	1.100895	5.141915	6
1644	Draymond Green	greendr01	2016	7778	3687	2.645310	4.760084	7
1229	Tim Duncan	duncati01	2016	3443	1754	-0.800494	5.014145	4
1146	Gorgui Dieng	dienggo01	2020	2355	1096	-1.914852	4.992603	3

10 rows × 21 columns

Top 10 players ranked by defensive RAPTOR from the 2014 season

Analysis of Leaders in the Historical RAPTOR Dataset (197

In [23]:
```
rap_hist['mp'].mean()
```

```
1363.0184769559999
```

Here, we notice that the average number of minutes played is slightly higher. Maint
used for the modern RAPTOR dataset should be appropriate.

In [24]:
```
rap_hist_tot = rap_hist[rap_hist['mp'] >= 1000].sort_values(by = 'raptor_t
rap_hist_tot
```

	player_name	player_id	season	poss	mp	raptor_offense	raptor_defense	raptor_tot
9057	Nikola Jokic	jokicni01	2022	5481	2647	8.520715	6.051696	14.572411
8517	LeBron James	jamesle01	2009	6760	3634	9.334994	3.245821	12.580815
3950	Stephen Curry	curryst01	2016	7053	3314	10.379411	2.108447	12.487858
9313	Michael Jordan	jordami01	1991	7391	3723	9.129036	3.162250	12.291286
13404	Chris Paul	paulch01	2009	5934	3203	8.195175	3.653223	11.848398
8518	LeBron James	jamesle01	2010	6611	3426	9.196731	2.250084	11.446815
13409	Chris Paul	paulch01	2014	5387	2643	7.723907	3.668439	11.392346
9310	Michael Jordan	jordami01	1988	7498	3738	7.511207	3.677306	11.188513
3949	Stephen Curry	curryst01	2015	7192	3439	8.600890	2.437371	11.038262
9311	Michael Jordan	jordami01	1989	8031	3973	8.297340	2.711836	11.009176

Top 10 players ranked by Total RAPTOR since 1972 (for an individual season)

In [25]:
```
rap_hist_off = rap_hist[rap_hist['mp'] >= 1000].sort_values(by = 'raptor_o
rap_hist_off
```

	player_name	player_id	season	poss	mp	raptor_offense	raptor_defense	raptor_tot
3950	Stephen Curry	curryst01	2016	7053	3314	10.379411	2.108447	12.487858
7041	James Harden	hardeja01	2019	6728	3291	9.566047	1.149234	10.715281
8517	LeBron James	jamesle01	2009	6760	3634	9.334994	3.245821	12.580815
3951	Stephen Curry	curryst01	2017	6963	3239	9.308684	-0.095877	9.212807
8518	LeBron James	jamesle01	2010	6611	3426	9.196731	2.250084	11.446815
9313	Michael Jordan	jordami01	1991	7391	3723	9.129036	3.162250	12.291286
13403	Chris Paul	paulch01	2008	6582	3492	9.040852	0.968721	10.009573
8971	Magic Johnson	johnsma02	1990	6706	3313	9.010918	0.289609	9.300527
7040	James Harden	hardeja01	2018	6466	3172	8.750965	1.316804	10.067769
3955	Stephen Curry	curryst01	2021	4715	2152	8.738701	-1.939050	6.799651

Top 10 players ranked by Offensive RAPTOR since 1972 (for an individual season)

In [26]:
```
rap_hist_def = rap_hist[rap_hist['mp'] >= 1000].sort_values(by = 'raptor_d
rap_hist_def
```

	player_name	player_id	season	poss	mp	raptor_offense	raptor_defense	raptor_tot
6322	Rudy Gobert	goberru01	2021	5381	2563	-0.120712	7.907439	7.786727
17662	Ben Wallace	wallabe01	2004	7325	3974	-1.467349	7.340098	5.872749
14684	David Robinson	robinda01	1992	5198	2564	2.535955	7.179032	9.714987
17661	Ben Wallace	wallabe01	2003	6571	3595	-1.362756	6.944065	5.581309
12957	Hakeem Olajuwon	olajuha01	1991	4527	2191	-0.012196	6.847959	6.835763
6323	Rudy Gobert	goberru01	2022	4762	2317	0.063774	6.822549	6.886323
12956	Hakeem Olajuwon	olajuha01	1990	6976	3285	-1.441270	6.736072	5.294802
14691	David Robinson	robinda01	1999	3981	2154	1.105714	6.691488	7.797202
17664	Ben Wallace	wallabe01	2006	6435	3532	-1.491701	6.479343	4.987642
12955	Hakeem Olajuwon	olajuha01	1989	6822	3186	0.196380	6.440926	6.637306

Top 10 players ranked by Defensive RAPTOR since 1972 (for an individual season)

In [27]:
```
rap_hist
```

	player_name	player_id	season	poss	mp	raptor_offense	raptor_defense	raptor_tot
0	Alaa Abdelnaby	abdelal01	1991	640	303	-3.938450	-0.510076	-4.448526
1	Alaa Abdelnaby	abdelal01	1992	1998	959	-2.553849	-0.197943	-2.751792
2	Alaa Abdelnaby	abdelal01	1993	2754	1379	-2.373736	-2.069808	-4.443544
3	Alaa Abdelnaby	abdelal01	1994	320	159	-6.140056	-2.748312	-8.888368
4	Alaa Abdelnaby	abdelal01	1995	984	506	-3.846543	-1.268012	-5.114556
...	...	...	...	...	...	...	...	...
19154	Ivica Zubac	zubaciv01	2018	871	410	-2.903709	2.688832	-0.214877
19155	Ivica Zubac	zubaciv01	2019	2345	1079	-2.362444	1.813768	-0.548676
19156	Ivica Zubac	zubaciv01	2020	3447	1646	1.099849	3.549458	4.649308
19157	Ivica Zubac	zubaciv01	2021	3908	1910	-0.909039	2.525735	1.616696
19158	Ivica Zubac	zubaciv01	2022	3786	1852	-0.693765	0.723358	0.029594

19159 rows × 15 columns

In [28]:
```
smallest = rap_hist['season'].idxmin()
```

In [29]:
```
smallest
```

```
5
```

In [30]:
```
rap_hist.loc[smallest]
```

```
player_name         Kareem Abdul-Jabbar
player_id                     abdulka01
season                             1977
poss                               7674
mp                                 3483
raptor_offense                 4.544044
raptor_defense                 3.103855
raptor_total                   7.647899
war_total                     18.488255
war_reg_season                15.456216
war_playoffs                   3.032039
predator_offense               4.762807
predator_defense               2.624501
predator_total                 7.387308
pace_impact                   -0.502962
Name: 5, dtype: object
```

In [39]:
```
rap_modern[rap_modern['mp']>=1000].drop_duplicates(subset=['player_name'])
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	r
0	Alex Abrines	abrinal01	2017	2387	1135	0.745505	-0.372938	0
4	Precious Achiuwa	achiupr01	2022	3802	1892	-2.521510	1.763502	-(
6	Quincy Acy	acyqu01	2015	2517	1287	-2.014956	-1.268440	-(
16	Steven Adams	adamsst01	2014	3078	1528	-1.270250	0.056605	-(
25	Bam Adebayo	adebaba01	2018	2921	1445	-2.096991	0.888476	-(
...	...	...	...	...	...	...	...	..
4652	Trae Young	youngtr01	2019	5516	2503	2.097204	-3.469197	-(
4657	Cody Zeller	zelleco01	2014	2970	1469	-0.899884	-0.370387	-(
4666	Tyler Zeller	zellety01	2014	2088	1049	-1.256580	2.095302	0
4677	Ante Zizic	zizican01	2019	2240	1082	-2.614802	-1.767888	-(
4681	Ivica Zubac	zubaciv01	2019	2345	1079	-2.336006	1.400274	-(

665 rows × 21 columns

In [40]:
```
of_names = rap_modern[rap_modern['mp']>=1000].drop_duplicates(subset=['play
```

In [41]:
```
list_of_names
```

```
['Alex Abrines',
 'Precious Achiuwa',
 'Quincy Acy',
 'Steven Adams',
 'Bam Adebayo',
 'Arron Afflalo',
 'LaMarcus Aldridge',
 'Nickeil Alexander-Walker',
 'Grayson Allen',
 'Jarrett Allen',
 'Lavoy Allen',
 'Ray Allen',
 'Tony Allen',
 'Al-Farouq Aminu',
 'Alan Anderson',
 'Chris Andersen',
 'James Anderson',
 'Justin Anderson',
 'Kyle Anderson',
 'Ryan Anderson',
 'Giannis Antetokounmpo',
 'Carmelo Anthony',
 'Cole Anthony',
```

In [ ]:

Exploratory Data Analysis

Import necessary packages

In [88]:
```
import pandas as pd
import plotly.express as px
from nba_api.stats.endpoints import playercareerstats
from nba_api.stats.static import players
from nba_api.stats.static import teams
from nba_api.stats.endpoints import leagueleaders
```

To begin this section of the EDA, we want to combine RAPTOR data with NBA player data sourced from the NBA site using the nba_api package. First we need to tabulate all the players in the NBA fro the 1976 season, as well as their seasons. Then we can proceed to assign raptor stats to each playe for each season.

In [97]:
```
list_of_players = players.get_players()
```

In [98]:
```
len(list_of_players)
```

```
4815
```

In [99]:
```
list_of_players[0]
```

```
{'id': 76001,
 'full_name': 'Alaa Abdelnaby',
 'first_name': 'Alaa',
 'last_name': 'Abdelnaby',
 'is_active': False}
```

In [111]:
```
player_ids = []
player_names = []
player_id_dict = {}
for p in list_of_players:
    player_ids.append(p['id'])
    player_names.append(p['full_name'])
    player_id_dict[p['full_name']] = p['id']
```

Proceeding to combine regular season statistics for every player(for each of their seasons)

In [116]:
```
rap_modern = pd.read_csv('modern_RAPTOR_by_player.csv')
```

In [117]:
```
= rap_modern[rap_modern['mp']>=1000].drop_duplicates(subset=['player_name'])['player_name'].tolist(
```

In [118]:
```
len(list_of_names)
```

```
665
```

In [121]:
```
len(player_ids)
reg_season_list = []
```

In [318]:
```
for i in range(660,665):
    temp_career = playercareerstats.PlayerCareerStats(player_id = player_id_dict[list_of_names[i]]
    temp_df = temp_career.get_data_frames()[0]
    reg_season_list.append(temp_df)
```

In [319]:
```
len(reg_season_list)
```

665

In [302]:
```
#list_of_names[331] = 'Enes Freedom' from Enes Kanter
# Had to change OG Anunoby to O.G. Anunoby
#list_of_names[340] = 'Kevin Knox II' from Kevin Knox
#list_of_names[430] = 'Marcus Morris Sr.' from Marcus Morris
#list_of_names[600] = 'P.J. Tucker' from PJ Tucker
```

In [320]:
```
merged_df_600 = pd.DataFrame()
merged_df_665 = pd.concat(reg_season_list)
```

In [321]:
```
merged_df_665
```

TEAM_ABBREVIATION	PLAYER_AGE	GP	GS	MIN	FGM	...	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TOV	PF	PT
OKC	23.0	68	6	1055.0	134	...	0.898	18	68	86	40	37	8	33.0	114	40
OKC	24.0	75	8	1134.0	115	...	0.848	26	88	114	28	38	8	25.0	124	35
OKC	25.0	31	2	588.0	56	...	0.923	5	43	48	20	17	6	14.0	53	16
MIA	21.0	61	4	737.0	124	...	0.509	73	135	208	29	20	28	43.0	91	30
TOR	22.0	73	28	1725.0	265	...	0.595	146	327	473	82	37	41	84.0	151	66
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TOT	22.0	59	37	1039.0	212	...	0.802	115	247	362	63	14	51	70.0	137	52
LAC	23.0	72	70	1326.0	236	...	0.747	197	346	543	82	16	66	61.0	168	59
LAC	24.0	72	33	1609.0	257	...	0.789	189	330	519	90	24	62	81.0	187	65
LAC	25.0	76	76	1852.0	310	...	0.727	217	427	644	120	36	77	114.0	203	78
LAC	26.0	76	76	2169.0	326	...	0.697	236	520	756	77	29	98	117.0	219	81

In [323]:
```
merged_df_665.to_csv('modern_regular_stats.csv', index=False)
```

We now have a csv of all regular season stats for all players in the modern RAPTOR dataset. Importing the csv, we can begin to add 'wins above replacement' RAPTOR stats for the regular seas and the playoffs to each row of this dataset.

In [30]:
```
regular_season_stats = pd.read_csv("modern_regular_stats.csv")
```

In [31]:
```
regular_season_stats
```

TEAM_ABBREVIATION	PLAYER_AGE	GP	GS	MIN	FGM	...	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TOV	PF	P
OKC	23.0	68	6	1055.0	134	...	0.898	18	68	86	40	37	8	33.0	114	40
OKC	24.0	75	8	1134.0	115	...	0.848	26	88	114	28	38	8	25.0	124	35
OKC	25.0	31	2	588.0	56	...	0.923	5	43	48	20	17	6	14.0	53	16
MIA	21.0	61	4	737.0	124	...	0.509	73	135	208	29	20	28	43.0	91	30
TOR	22.0	73	28	1725.0	265	...	0.595	146	327	473	82	37	41	84.0	151	66
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
TOT	22.0	59	37	1039.0	212	...	0.802	115	247	362	63	14	51	70.0	137	52
LAC	23.0	72	70	1326.0	236	...	0.747	197	346	543	82	16	66	61.0	168	59
LAC	24.0	72	33	1609.0	257	...	0.789	189	330	519	90	24	62	81.0	187	65
LAC	25.0	76	76	1852.0	310	...	0.727	217	427	644	120	36	77	114.0	203	78
LAC	26.0	76	76	2169.0	326	...	0.697	236	520	756	77	29	98	117.0	219	81

Creating a dictionary of ids to full names

In [32]:
```
list_of_players = players.get_players()
player_id_name_dict = {}
for p in list_of_players:
    player_id_name_dict[p['id']] = p['full_name']
```

In [33]:
```
name_list = []
for ident in regular_season_stats['PLAYER_ID']:
    name_list.append(player_id_name_dict[ident])
```

In [34]:
```
regular_season_stats['FULL_NAME'] = name_list
```

In [35]:
```
regular_season_stats.columns
```

```
Index(['PLAYER_ID', 'SEASON_ID', 'LEAGUE_ID', 'TEAM_ID', 'TEAM_ABBREVIATION',
       'PLAYER_AGE', 'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A',
       'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL',
       'BLK', 'TOV', 'PF', 'PTS', 'FULL_NAME'],
      dtype='object')
```

In [36]:
```
regular_season_stats
```

AM_ABBREVIATION	PLAYER_AGE	GP	GS	MIN	FGM	...	OREB	DREB	REB	AST	STL	BLK	TOV	PF	PTS	FULL_NAM
C	23.0	68	6	1055.0	134	...	18	68	86	40	37	8	33.0	114	406	Alex Abrines
C	24.0	75	8	1134.0	115	...	26	88	114	28	38	8	25.0	124	353	Alex Abrines
C	25.0	31	2	588.0	56	...	5	43	48	20	17	6	14.0	53	165	Alex Abrines
	21.0	61	4	737.0	124	...	73	135	208	29	20	28	43.0	91	304	Precious Achiuwa
F	22.0	73	28	1725.0	265	...	146	327	473	82	37	41	84.0	151	664	Precious Achiuwa
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	22.0	59	37	1039.0	212	...	115	247	362	63	14	51	70.0	137	525	Ivica Zubac
	23.0	72	70	1326.0	236	...	197	346	543	82	16	66	61.0	168	596	Ivica Zubac
	24.0	72	33	1609.0	257	...	189	330	519	90	24	62	81.0	187	650	Ivica Zubac
	25.0	76	76	1852.0	310	...	217	427	644	120	36	77	114.0	203	785	Ivica Zubac
	26.0	76	76	2169.0	326	...	236	520	756	77	29	98	117.0	219	818	Ivica Zubac

We now want create new columns for Points per game (PPG), Assists per game (APG), Offensive Rebounds pe game (OREBPG), Defensive Rebounds per game (DREBPG), Steals per game (STLPG), Blocks per game (BLKPG), and Turnovers per game (TOVPG).

In [37]:
```
regular_season_stats['PPG'] = regular_season_stats['PTS']/regular_season_stats['GP']
```

In [38]:
```
regular_season_stats['APG'] = regular_season_stats['AST']/regular_season_stats['GP']
regular_season_stats['OREBPG'] = regular_season_stats['OREB']/regular_season_stats['GP']
regular_season_stats['DREBPG'] = regular_season_stats['DREB']/regular_season_stats['GP']
regular_season_stats['STLPG'] = regular_season_stats['STL']/regular_season_stats['GP']
regular_season_stats['BLKPG'] = regular_season_stats['BLK']/regular_season_stats['GP']
regular_season_stats['TOVPG'] = regular_season_stats['TOV']/regular_season_stats['GP']
```

In [39]:
```
regular_season_stats
```

	PLAYER_ID	SEASON_ID	LEAGUE_ID	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	GS	MIN	FGM	...	PF
0	203518	2016-17	0	1610612760	OKC	23.0	68	6	1055.0	134	...	114
1	203518	2017-18	0	1610612760	OKC	24.0	75	8	1134.0	115	...	124
2	203518	2018-19	0	1610612760	OKC	25.0	31	2	588.0	56	...	53
3	1630173	2020-21	0	1610612748	MIA	21.0	61	4	737.0	124	...	91
4	1630173	2021-22	0	1610612761	TOR	22.0	73	28	1725.0	265	...	151
...	...	...	...	...	...	...	...	...	...	...	...	...
6775	1627826	2018-19	0	0	TOT	22.0	59	37	1039.0	212	...	137
6776	1627826	2019-20	0	1610612746	LAC	23.0	72	70	1326.0	236	...	168
6777	1627826	2020-21	0	1610612746	LAC	24.0	72	33	1609.0	257	...	187
6778	1627826	2021-22	0	1610612746	LAC	25.0	76	76	1852.0	310	...	203
6779	1627826	2022-23	0	1610612746	LAC	26.0	76	76	2169.0	326	...	219

6780 rows × 35 columns

In [40]:
```
del regular_season_stats['LEAGUE_ID']
```

In [41]:
```
regular_season_stats.columns
```

```
Index(['PLAYER_ID', 'SEASON_ID', 'TEAM_ID', 'TEAM_ABBREVIATION', 'PLAYER_AGE',
       'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT',
       'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK',
       'TOV', 'PF', 'PTS', 'FULL_NAME', 'PPG', 'APG', 'OREBPG', 'DREBPG',
       'STLPG', 'BLKPG', 'TOVPG'],
      dtype='object')
```

Now deleting all the columns that we would not need

In [42]:
```
regular_season_stats = regular_season_stats.drop(['GS', 'FGM', 'FGA', 'FG3M',
                                                  'FG3A', 'FTM', 'FTA', 'OREB',
                                                  'DREB', 'REB', 'AST', 'STL',
                                                  'BLK', 'TOV', 'PF', 'PTS'], axis=1)
```

In [43]:
```
regular_season_stats
```

ID	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	FT_PCT	FULL_NAME	PPG	APG
	1610612760	OKC	23.0	68	1055.0	0.393	0.381	0.898	Alex Abrines	5.970588	0.588235
	1610612760	OKC	24.0	75	1134.0	0.395	0.380	0.848	Alex Abrines	4.706667	0.373333
	1610612760	OKC	25.0	31	588.0	0.357	0.323	0.923	Alex Abrines	5.322581	0.645161
	1610612748	MIA	21.0	61	737.0	0.544	0.000	0.509	Precious Achiuwa	4.983607	0.475410
	1610612761	TOR	22.0	73	1725.0	0.439	0.359	0.595	Precious Achiuwa	9.095890	1.123288
	...	...	...	...	...	...	...	...	...	...	...
	0	TOT	22.0	59	1039.0	0.559	0.000	0.802	Ivica Zubac	8.898305	1.067797
	1610612746	LAC	23.0	72	1326.0	0.613	0.000	0.747	Ivica Zubac	8.277778	1.138889
	1610612746	LAC	24.0	72	1609.0	0.652	0.250	0.789	Ivica Zubac	9.027778	1.250000
	1610612746	LAC	25.0	76	1852.0	0.626	0.000	0.727	Ivica Zubac	10.328947	1.578947
	1610612746	LAC	26.0	76	2169.0	0.634	0.000	0.697	Ivica Zubac	10.763158	1.013158

Reordering the columns to make it easier to read

In [45]:
```
regular_season_stats.insert(0, 'FULL_NAME', regular_season_stats.pop('FULL_NAME'))
```

In [46]:
```
regular_season_stats
```

TION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	FT_PCT	PPG	APG	OREBPG	DREBPG	STLPG	BLKPG	TOVP
	23.0	68	1055.0	0.393	0.381	0.898	5.970588	0.588235	0.264706	1.000000	0.544118	0.117647	0.4852
	24.0	75	1134.0	0.395	0.380	0.848	4.706667	0.373333	0.346667	1.173333	0.506667	0.106667	0.3333
	25.0	31	588.0	0.357	0.323	0.923	5.322581	0.645161	0.161290	1.387097	0.548387	0.193548	0.4516
	21.0	61	737.0	0.544	0.000	0.509	4.983607	0.475410	1.196721	2.213115	0.327869	0.459016	0.7049
	22.0	73	1725.0	0.439	0.359	0.595	9.095890	1.123288	2.000000	4.479452	0.506849	0.561644	1.1506
	...	...	...	...	...	...	...	...	...	...	...	...	...
	22.0	59	1039.0	0.559	0.000	0.802	8.898305	1.067797	1.949153	4.186441	0.237288	0.864407	1.1864
	23.0	72	1326.0	0.613	0.000	0.747	8.277778	1.138889	2.736111	4.805556	0.222222	0.916667	0.8472
	24.0	72	1609.0	0.652	0.250	0.789	9.027778	1.250000	2.625000	4.583333	0.333333	0.861111	1.1250
	25.0	76	1852.0	0.626	0.000	0.727	10.328947	1.578947	2.855263	5.618421	0.473684	1.013158	1.5000
	26.0	76	2169.0	0.634	0.000	0.697	10.763158	1.013158	3.105263	6.842105	0.381579	1.289474	1.5394

Saving the new dataframe as a csv for later use

In [79]:
```
regular_season_stats.to_csv('regular_season_stats_mod.csv', index = False)
```

In [48]:
```
rap_modern = pd.read_csv('modern_RAPTOR_by_player.csv')
rap_modern
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	raptor_box_total	raptor_onoff_o
0	Alex Abrines	abrinal01	2017	2387	1135	0.745505	-0.372938	0.372567	-0.418553
1	Alex Abrines	abrinal01	2018	2546	1244	0.317549	-1.725325	-1.407776	-1.291727
2	Alex Abrines	abrinal01	2019	1279	588	-3.215683	1.078399	-2.137285	-6.158856
3	Precious Achiuwa	achiupr01	2021	1581	749	-4.122966	1.359278	-2.763688	-4.050779
4	Precious Achiuwa	achiupr01	2022	3802	1892	-2.521510	1.763502	-0.758008	-1.687893
...	...	...	...	...	...	...	...	...	...
4680	Ivica Zubac	zubaciv01	2018	871	410	-2.581325	1.643240	-0.938084	-3.444739
4681	Ivica Zubac	zubaciv01	2019	2345	1079	-2.336006	1.400274	-0.935733	-1.757992
4682	Ivica Zubac	zubaciv01	2020	3447	1646	0.516422	3.529428	4.045850	3.314971
4683	Ivica Zubac	zubaciv01	2021	3908	1910	-0.371753	2.960613	2.588860	-2.541995
4684	Ivica Zubac	zubaciv01	2022	3786	1852	-0.920033	1.986362	1.066330	0.343447

4685 rows × 21 columns

Now let's look at how we can add the 'war' RAPTOR stat for the regular and season and the playoffs for each player. For each player in the regular_season_stats dataset, we want to add their respective 'war_reg_season' and 'war_playoffs'. We can search for the players respective RAPTOR stat by searching the 'rap_modern' datas using 'FULL_NAME' and 'SEASON_ID'. First, we need to convert 'SEASON_ID' into an int with the single year format (e.g. from '2016-17' to 2016)

In [50]:
```
exp_str = '2016-17'
year = int(exp_str[:4])
year
```

2016

In [52]:
```
years = []
for y in regular_season_stats['SEASON_ID']:
    years.append(int(y[:4]))
```

In [54]:
```
regular_season_stats.insert(2, 'SEASON', years)
```

In [56]:
```
del regular_season_stats['SEASON_ID']
```

In [57]:
```
regular_season_stats
```

EAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	FT_PCT	PPG	APG	OREBPG	DREBPG	STLPG
KC	23.0	68	1055.0	0.393	0.381	0.898	5.970588	0.588235	0.264706	1.000000	0.544118
KC	24.0	75	1134.0	0.395	0.380	0.848	4.706667	0.373333	0.346667	1.173333	0.506667
KC	25.0	31	588.0	0.357	0.323	0.923	5.322581	0.645161	0.161290	1.387097	0.548387
A	21.0	61	737.0	0.544	0.000	0.509	4.983607	0.475410	1.196721	2.213115	0.327869
OR	22.0	73	1725.0	0.439	0.359	0.595	9.095890	1.123288	2.000000	4.479452	0.506849
...	...	...	...	...	...	...	...	...	...	...	...
OT	22.0	59	1039.0	0.559	0.000	0.802	8.898305	1.067797	1.949153	4.186441	0.237288
\C	23.0	72	1326.0	0.613	0.000	0.747	8.277778	1.138889	2.736111	4.805556	0.222222
\C	24.0	72	1609.0	0.652	0.250	0.789	9.027778	1.250000	2.625000	4.583333	0.333333
\C	25.0	76	1852.0	0.626	0.000	0.727	10.328947	1.578947	2.855263	5.618421	0.473684
\C	26.0	76	2169.0	0.634	0.000	0.697	10.763158	1.013158	3.105263	6.842105	0.381579

Upon scrolling up to the section where the regular_season_stats dataframe was being assimilated, it was notice that a few full names in the rap_modern dataframe were slightly incorrect. Populating the regular_season_stat dataframe with RAPTOR stats would be interrupted by slight name changes. As the name differences have bee kept track of above, the necessary changes can be made in the rap_modern dataframe.

In [58]:
```
#list_of_names[331] = 'Enes Freedom' from Enes Kanter
# Had to change OG Anunoby to O.G. Anunoby
#list_of_names[340] = 'Kevin Knox II' from Kevin Knox
#list_of_names[430] = 'Marcus Morris Sr.' from Marcus Morris
#list_of_names[600] = 'P.J. Tucker' from PJ Tucker
```

In [61]:
```
rap_modern['player_name'] = rap_modern['player_name'].replace('Enes Kanter', 'Enes Freedom')
rap_modern['player_name'] = rap_modern['player_name'].replace('OG Anunoby', 'O.G. Anunoby')
rap_modern['player_name'] = rap_modern['player_name'].replace('Kevin Knox', 'Kevin Knox II')
rap_modern['player_name'] = rap_modern['player_name'].replace('Marcus Morris', 'Marcus Morris Sr.'
rap_modern['player_name'] = rap_modern['player_name'].replace('PJ Tucker', 'P.J. Tucker')
```

In [64]:
```
rap_modern[rap_modern['player_name']=='Enes Freedom']
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	raptor_box_total	raptor_onoff_o
2382	Enes Freedom	kanteen01	2014	4136	2138	-3.045494	-1.796736	-4.842230	-2.917159
2383	Enes Freedom	kanteen01	2015	4210	2135	-0.057597	-0.749248	-0.806845	2.515448
2384	Enes Freedom	kanteen01	2016	4149	2044	3.129517	0.497602	3.627119	-0.218940
2385	Enes Freedom	kanteen01	2017	3240	1578	1.539305	-0.174973	1.364332	1.929721
2386	Enes Freedom	kanteen01	2018	3735	1830	2.082297	-1.371132	0.711164	1.544990
2387	Enes Freedom	kanteen01	2019	4362	2101	0.807871	-0.543977	0.263894	0.026699
2388	Enes Freedom	kanteen01	2020	2266	1085	1.136162	0.543030	1.679192	3.904585
2389	Enes Freedom	kanteen01	2021	3727	1814	1.981250	-2.275327	-0.294077	4.120736
2390	Enes Freedom	kanteen01	2022	844	411	-1.297013	2.829146	1.532133	0.907752

9 rows × 21 columns

In [65]:
```
regular_season_stats[regular_season_stats['FULL_NAME']=='Enes Freedom']
```

	FULL_NAME	PLAYER_ID	SEASON	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT
3564	Enes Freedom	202683	2011	1610612762	UTA	20.0	66	874.0	0.496	0.000
3565	Enes Freedom	202683	2012	1610612762	UTA	21.0	70	1078.0	0.544	1.000
3566	Enes Freedom	202683	2013	1610612762	UTA	22.0	80	2138.0	0.491	0.000
3567	Enes Freedom	202683	2014	1610612762	UTA	23.0	49	1326.0	0.491	0.317
3568	Enes Freedom	202683	2014	1610612760	OKC	23.0	26	809.0	0.566	0.750
3569	Enes Freedom	202683	2014	0	TOT	23.0	75	2136.0	0.519	0.356
3570	Enes Freedom	202683	2015	1610612760	OKC	24.0	82	1721.0	0.576	0.476
3571	Enes Freedom	202683	2016	1610612760	OKC	25.0	72	1533.0	0.545	0.132
3572	Enes Freedom	202683	2017	1610612752	NYK	26.0	71	1830.0	0.592	0.000
3573	Enes Freedom	202683	2018	1610612752	NYK	27.0	44	1128.0	0.536	0.318
3574	Enes Freedom	202683	2018	1610612757	POR	27.0	23	512.0	0.577	0.250
3575	Enes Freedom	202683	2018	0	TOT	27.0	67	1639.0	0.549	0.294
3576	Enes Freedom	202683	2019	1610612738	BOS	28.0	58	983.0	0.572	0.143
3577	Enes Freedom	202683	2020	1610612757	POR	29.0	72	1757.0	0.604	0.250
3578	Enes Freedom	202683	2021	1610612738	BOS	30.0	35	411.0	0.526	0.400

It is noticed that rap_modern and regular_season_stats have unequal number of rows because rap_modern ha player seasons from 2014 onwards whereas regular_season_stats has player seasons from when they first started (possible before 2014), as seen in the above example with 'Enes Freedom'. Dropping all rows in regular_season_stats that have records prior to 2014

In [66]:
```
regular_season_stats = regular_season_stats.drop(
    regular_season_stats[regular_season_stats['SEASON']<2014].index)
```

In [67]:
```
regular_season_stats
```

	FULL_NAME	PLAYER_ID	SEASON	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT
0	Alex Abrines	203518	2016	1610612760	OKC	23.0	68	1055.0	0.393	0.381
1	Alex Abrines	203518	2017	1610612760	OKC	24.0	75	1134.0	0.395	0.380
2	Alex Abrines	203518	2018	1610612760	OKC	25.0	31	588.0	0.357	0.323
3	Precious Achiuwa	1630173	2020	1610612748	MIA	21.0	61	737.0	0.544	0.000
4	Precious Achiuwa	1630173	2021	1610612761	TOR	22.0	73	1725.0	0.439	0.359
...	...	...	...	...	...	...	...	...	...	...
6775	Ivica Zubac	1627826	2018	0	TOT	22.0	59	1039.0	0.559	0.000
6776	Ivica Zubac	1627826	2019	1610612746	LAC	23.0	72	1326.0	0.613	0.000
6777	Ivica Zubac	1627826	2020	1610612746	LAC	24.0	72	1609.0	0.652	0.250
6778	Ivica Zubac	1627826	2021	1610612746	LAC	25.0	76	1852.0	0.626	0.000
6779	Ivica Zubac	1627826	2022	1610612746	LAC	26.0	76	2169.0	0.634	0.000

4493 rows × 18 columns

In [75]:
```
regular_season_stats.merge(rap_modern[['player_name', 'season', 'war_reg_season', 'war_playoffs']],
                           left_on = ['FULL_NAME', 'SEASON'],
                           right_on = ['player_name', 'season'], how = 'left')
```

In [76]:
```
stat_merge
```

...CT	FG3_PCT	...	APG	OREBPG	DREBPG	STLPG	BLKPG	TOVPG	player_name	season	war_reg_season	war_playo...
	0.381	...	0.588235	0.264706	1.000000	0.544118	0.117647	0.485294	NaN	NaN	NaN	NaN
	0.380	...	0.373333	0.346667	1.173333	0.506667	0.106667	0.333333	Alex Abrines	2017.0	1.447708	-0.198700
	0.323	...	0.645161	0.161290	1.387097	0.548387	0.193548	0.451613	Alex Abrines	2018.0	0.465912	0.311392
	0.000	...	0.475410	1.196721	2.213115	0.327869	0.459016	0.704918	NaN	NaN	NaN	NaN
	0.359	...	1.123288	2.000000	4.479452	0.506849	0.561644	1.150685	Precious Achiuwa	2021.0	-0.246776	0.000721
	...	...	...	...	...	...	...	...	...	...	...	...
	0.000	...	1.067797	1.949153	4.186441	0.237288	0.864407	1.186441	Ivica Zubac	2018.0	0.531968	0.000000
	0.000	...	1.138889	2.736111	4.805556	0.222222	0.916667	0.847222	Ivica Zubac	2019.0	1.114995	0.102138
	0.250	...	1.250000	2.625000	4.583333	0.333333	0.861111	1.125000	Ivica Zubac	2020.0	5.020903	1.203304
	0.000	...	1.578947	2.855263	5.618421	0.473684	1.013158	1.500000	Ivica Zubac	2021.0	3.704630	0.556697
	0.000	...	1.013158	3.105263	6.842105	0.381579	1.289474	1.539474	Ivica Zubac	2022.0	2.626958	0.000000

In [80]:
```
stat_merge = stat_merge.dropna()
```

In [81]:
```
stat_merge
```

	FULL_NAME	PLAYER_ID	SEASON	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT
1	Alex Abrines	203518	2017	1610612760	OKC	24.0	75	1134.0	0.395	0.380
2	Alex Abrines	203518	2018	1610612760	OKC	25.0	31	588.0	0.357	0.323
4	Precious Achiuwa	1630173	2021	1610612761	TOR	22.0	73	1725.0	0.439	0.359
5	Precious Achiuwa	1630173	2022	1610612761	TOR	23.0	55	1141.0	0.485	0.269
6	Quincy Acy	203112	2014	1610612752	NYK	24.0	68	1287.0	0.459	0.300
...	...	...	...	...	...	...	...	...	...	...
4488	Ivica Zubac	1627826	2018	0	TOT	22.0	59	1039.0	0.559	0.000
4489	Ivica Zubac	1627826	2019	1610612746	LAC	23.0	72	1326.0	0.613	0.000
4490	Ivica Zubac	1627826	2020	1610612746	LAC	24.0	72	1609.0	0.652	0.250
4491	Ivica Zubac	1627826	2021	1610612746	LAC	25.0	76	1852.0	0.626	0.000
4492	Ivica Zubac	1627826	2022	1610612746	LAC	26.0	76	2169.0	0.634	0.000

4059 rows × 22 columns

In [82]:
```
del stat_merge['season']
del stat_merge['player_name']
```

In [83]:
```
stat_merge
```

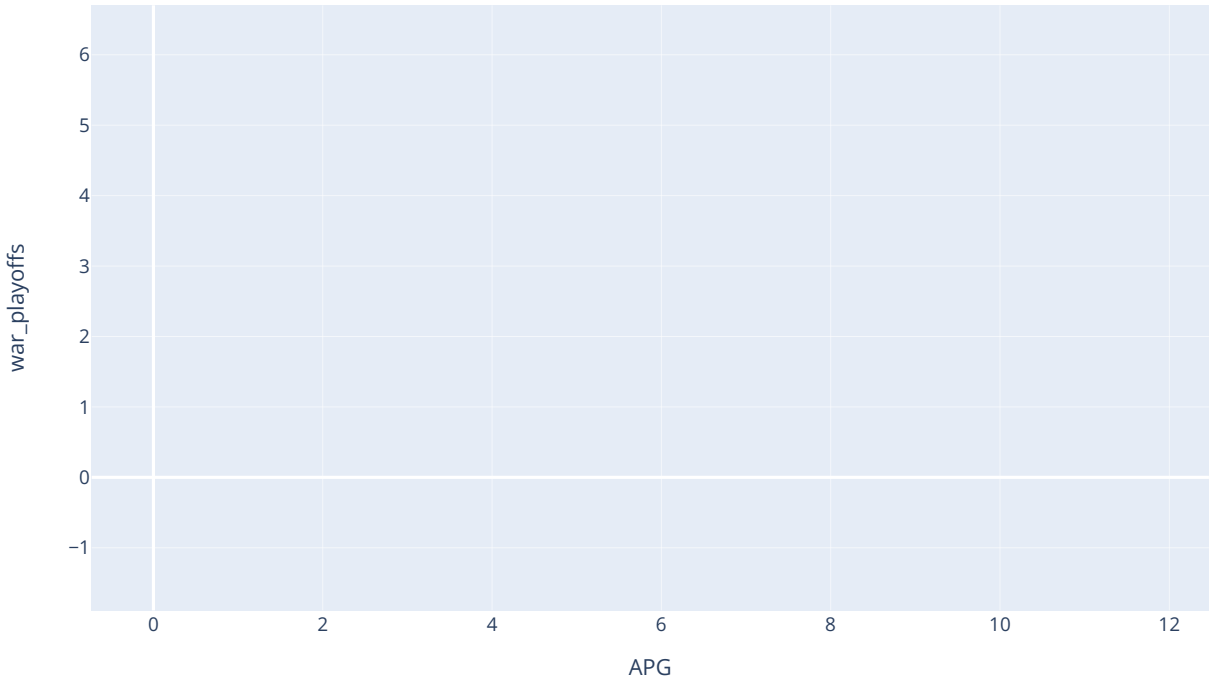
PLAYER_ID	SEASON	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	FT_PCT	PPG	
203518	2017	1610612760	OKC	24.0	75	1134.0	0.395	0.380	0.848	4.706667	0.
203518	2018	1610612760	OKC	25.0	31	588.0	0.357	0.323	0.923	5.322581	0.
1630173	2021	1610612761	TOR	22.0	73	1725.0	0.439	0.359	0.595	9.095890	1.
1630173	2022	1610612761	TOR	23.0	55	1141.0	0.485	0.269	0.702	9.236364	0.
203112	2014	1610612752	NYK	24.0	68	1287.0	0.459	0.300	0.784	5.852941	1.
...	...	...	...	...	...	...	...	...	...	...	...
1627826	2018	0	TOT	22.0	59	1039.0	0.559	0.000	0.802	8.898305	1.
1627826	2019	1610612746	LAC	23.0	72	1326.0	0.613	0.000	0.747	8.277778	1.
1627826	2020	1610612746	LAC	24.0	72	1609.0	0.652	0.250	0.789	9.027778	1.
1627826	2021	1610612746	LAC	25.0	76	1852.0	0.626	0.000	0.727	10.328947	1.
1627826	2022	1610612746	LAC	26.0	76	2169.0	0.634	0.000	0.697	10.763158	1.

ns

Now the final dataset, upon which the data analysis and modeling can be performed, is ready. As seen above, t dataset has a total of 4059 rows and 20 columns. Now saving the dataframe as a csv, and moving onto analysis and visualizations in Tableau.

In [84]:
```
stat_merge.to_csv('combined_stats.csv', index = False)
```

In [91]:
```
fig = px.scatter(stat_merge, x = 'APG', y = 'war_playoffs', trendline = 'ols')
fig.show()
```



Methodology

Upon performing Exploratory Data Analysis, it seems that Points Per Game (PPG), Assists Per Game (APG) , Defenseive Rebounds Per Game (DREBPG), Steals Per Game (STLPG), Turnovers Per Game (TOVPG), and Wins Above Replacement for Regular Season (WAR_regular_season), seemed to have the highest R^2 values for correlation with Wins Above Replacement for the playoffs (WAR_playoffs), which is the value to be predicted

Now, a multiple linear regression model will be constructed on the combined_stats dataset, to predict WAR_playoffs given Age, Field Goal% (FG_PCT), 3-point % (FG3_PCT), Free-Throw% (FT_PCT), PPG, APG, Offensive Rebounds Per Game (OREBPG), DREBPG, STLPG, Blocks Per Game (BLKPG), TOVPG, and war_reg_season as input.

In [93]:
```
combined_stats = pd.read_csv('combined_stats.csv')
```

In [94]:
```
combined_stats
```

FG_PCT	FG3_PCT	FT_PCT	PPG	APG	OREBPG	DREBPG	STLPG	BLKPG	TOVPG	war_reg_season	war_playoffs
0.395	0.380	0.848	4.706667	0.373333	0.346667	1.173333	0.506667	0.106667	0.333333	1.447708	-0.198700
0.357	0.323	0.923	5.322581	0.645161	0.161290	1.387097	0.548387	0.193548	0.451613	0.465912	0.311392
0.439	0.359	0.595	9.095890	1.123288	2.000000	4.479452	0.506849	0.561644	1.150685	-0.246776	0.000721
0.485	0.269	0.702	9.236364	0.909091	1.818182	4.145455	0.563636	0.545455	1.072727	2.309611	-0.046953
0.459	0.300	0.784	5.852941	1.000000	1.161765	3.264706	0.397059	0.323529	0.882353	0.415918	0.000000
...	...	...	...	...	...	...	...	...	...	...	...
0.559	0.000	0.802	8.898305	1.067797	1.949153	4.186441	0.237288	0.864407	1.186441	0.531968	0.000000
0.613	0.000	0.747	8.277778	1.138889	2.736111	4.805556	0.222222	0.916667	0.847222	1.114995	0.102138
0.652	0.250	0.789	9.027778	1.250000	2.625000	4.583333	0.333333	0.861111	1.125000	5.020903	1.203304
0.626	0.000	0.727	10.328947	1.578947	2.855263	5.618421	0.473684	1.013158	1.500000	3.704630	0.556697
0.634	0.000	0.697	10.763158	1.013158	3.105263	6.842105	0.381579	1.289474	1.539474	2.626958	0.000000

Building a Multiple Linear Regression Model

In [366]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

In [367]:
```
X = combined_stats[['PLAYER_AGE', 'FG_PCT', 'FG3_PCT', 'PPG', 'APG', 'OREBPG', 'DREBPG',
                    'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season']]
y = combined_stats['war_playoffs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
mlr_playoff_predictor = LinearRegression()
mlr_playoff_predictor.fit(X_train, y_train)
y_pred = mlr_playoff_predictor.predict(X_test)
mlr_playoff_predictor.coef_
```

```
array([ 0.00629254,  0.10752286, -0.02834451,  0.00385555,  0.0095318 ,
       -0.09860856,  0.04158363,  0.00361977,  0.0656474 , -0.01916523,
        0.1099223 ])
```

In [370]:
```
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
print(f"Train Score: {mlr_playoff_predictor.score(X_train,y_train)}")
print(f"Test Score: {mlr_playoff_predictor.score(X_test, y_test)}")
```

```
Mean Squared Error: 0.3231751925967567
Train Score: 0.2781334698252439
Test Score: 0.3089340597031007
```

Not being satisfied with the current r2_score, rebuilding the model to aim for a better score.
Testing all combinations of the independent variables to get the highest r2_score

In [361]:
```
for i in range(1,12):
    for comb in combinations(inputs, i):
        combination.append(comb)
```

In [362]:
```
combination_list = [list(t) for t in combination]
```

In [363]:
```
len(combination_list)
```

```
8189
```

In [156]:
```
r2_list = []
for comb in combination_list:
    X = combined_stats[comb]
    y = combined_stats['war_playoffs']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    mlr_playoff_predictor = LinearRegression()
    mlr_playoff_predictor.fit(X_train, y_train)
    y_pred = mlr_playoff_predictor.predict(X_test)
    r2_score_list.append(r2_score(y_test, y_pred))
```

In [164]:
```
r2_score_list.index(max(r2_score_list))
```

```
1598
```

In [161]:
```
combination_list[1598]
```

```
['PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season']
```

In [381]:
```
combined_stats
```

	FULL_NAME	PLAYER_ID	SEASON	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT
0	Alex Abrines	203518	2017	1610612760	OKC	24.0	75	1134.0	0.395	0.380
1	Alex Abrines	203518	2018	1610612760	OKC	25.0	31	588.0	0.357	0.323
2	Precious Achiuwa	1630173	2021	1610612761	TOR	22.0	73	1725.0	0.439	0.359
3	Precious Achiuwa	1630173	2022	1610612761	TOR	23.0	55	1141.0	0.485	0.269
4	Quincy Acy	203112	2014	1610612752	NYK	24.0	68	1287.0	0.459	0.300
...	...	...	...	...	...	...	...	...	...	...
4054	Ivica Zubac	1627826	2018	0	TOT	22.0	59	1039.0	0.559	0.000
4055	Ivica Zubac	1627826	2019	1610612746	LAC	23.0	72	1326.0	0.613	0.000
4056	Ivica Zubac	1627826	2020	1610612746	LAC	24.0	72	1609.0	0.652	0.250
4057	Ivica Zubac	1627826	2021	1610612746	LAC	25.0	76	1852.0	0.626	0.000
4058	Ivica Zubac	1627826	2022	1610612746	LAC	26.0	76	2169.0	0.634	0.000

4059 rows × 24 columns

In [410]:
```
X = combined_stats[['PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season']]
y = combined_stats['war_playoffs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
mlr_playoff_predictor = LinearRegression()
mlr_playoff_predictor.fit(X_train, y_train)
y_pred = mlr_playoff_predictor.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
print(f"Train Score: {mlr_playoff_predictor.score(X_train,y_train)}")
print(f"Test Score: {mlr_playoff_predictor.score(X_test, y_test)}")
```

```
Mean Squared Error: 0.3216255569311499
Train Score: 0.2736088240600827
Test Score: 0.3122477435900517
```

In [384]:
```
X_test
```

	PLAYER_AGE	FG_PCT	APG	OREBPG	STLPG	BLKPG	war_reg_season
3284	28.0	0.444	4.465753	0.767123	1.273973	0.342466	5.221274
2417	23.0	0.482	0.716667	0.900000	0.416667	0.650000	0.286765
4056	24.0	0.652	1.250000	2.625000	0.333333	0.861111	5.020903
746	24.0	0.437	0.500000	0.227273	0.227273	0.000000	-0.110400
1978	31.0	0.503	3.835616	0.835616	0.958904	0.698630	5.930054
...	...	...	...	...	...	...	...
3407	25.0	0.549	3.100000	1.550000	0.912500	0.650000	3.279707
3033	31.0	0.614	3.607143	2.589286	0.767857	0.892857	1.035639
2868	26.0	0.403	0.660714	0.553571	0.303571	0.000000	0.719233
2644	25.0	0.448	3.870130	0.415584	0.636364	0.259740	3.612770
2221	27.0	0.555	0.909091	1.818182	0.418182	0.872727	1.245723

812 rows × 7 columns

Using this combination of independent variables: ['PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season'], the highest possible r2_score of 0.3122 is attained

In [383]:
```
mlr_playoff_predictor.coef_
```

```
array([ 0.00596087,  0.10490216,  0.01777683, -0.04624937,  0.02111547,
        0.11058918,  0.11514209])
```

Testing

In [392]:
```
test_set = pd.read_csv('2022_23_NBA_Player_Stats-Regular.csv', delimiter = ';')
```

In [401]:
```
test_set
```

	Rk	Player	Pos	PLAYER_AGE	Tm	G	GS	MP	FG	FGA	...	AST	STL	BLK	TOV	PF	PTS	APG	OREBPG
0	1	Precious Achiuwa	C	23	TOR	55	12	20.7	3.6	7.3	...	0.9	0.6	0.5	1.1	1.9	9.2	0.016364	0.032727
1	2	Steven Adams	C	29	MEM	42	42	27.0	3.7	6.3	...	2.3	0.9	1.1	1.9	2.3	8.6	0.054762	0.121429
2	3	Bam Adebayo	C	25	MIA	75	75	34.6	8.0	14.9	...	3.2	1.2	0.8	2.5	2.8	20.4	0.042667	0.033333
3	4	Ochai Agbaji	SG	22	UTA	59	22	20.5	2.8	6.5	...	1.1	0.3	0.3	0.7	1.7	7.9	0.018644	0.011864
4	5	Santi Aldama	PF	22	MEM	77	20	21.8	3.2	6.8	...	1.3	0.6	0.6	0.8	1.9	9.0	0.016883	0.014286
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
674	535	Thaddeus Young	PF	34	TOR	54	9	14.7	2.0	3.7	...	1.4	1.0	0.1	0.8	1.6	4.4	0.025926	0.024074
675	536	Trae Young	PG	24	ATL	73	73	34.8	8.2	19.0	...	10.2	1.1	0.1	4.1	1.4	26.2	0.139726	0.010959
676	537	Omer Yurtseven	C	24	MIA	9	0	9.2	1.8	3.0	...	0.2	0.2	0.2	0.4	1.8	4.4	0.022222	0.100000
677	538	Cody Zeller	C	30	MIA	15	2	14.5	2.5	3.9	...	0.7	0.2	0.3	0.9	2.2	6.5	0.046667	0.113333
678	539	Ivica Zubac	C	25	LAC	76	76	28.6	4.3	6.8	...	1.0	0.4	1.3	1.5	2.9	10.8	0.013158	0.040789

679 rows × 34 columns

In [400]:
```
test_set.columns
```

```
Index(['Rk', 'Player', 'Pos', 'PLAYER_AGE', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA',
       'FG_PCT', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'APG', 'OREBPG', 'STLPG', 'BLKPG'],
      dtype='object')
```

In [395]:
```
test_set = test_set.rename(columns = {'Age':'PLAYER_AGE', 'FG%':'FG_PCT'})
```

In [398]:
```
test_set['APG'] = test_set['AST'] / test_set['G']
test_set['OREBPG'] = test_set['ORB'] / test_set['G']
test_set['STLPG'] = test_set['STL'] / test_set['G']
test_set['BLKPG'] = test_set['BLK'] / test_set['G']
```

In [402]:
```
latest_raptor = pd.read_csv('latest_RAPTOR_by_player.csv')
```

In [403]:
```
latest_raptor
```

ffense	raptor_onoff_defense	...	raptor_offense	raptor_defense	raptor_total	war_total	war_reg_season	war_playoffs	predatc
	-3.754012	...	-1.764165	-0.281497	-2.045662	0.405779	0.405779	0.000000	-1.85202
	0.182394	...	0.418196	3.548488	3.966684	3.891773	3.891773	0.000000	0.02748
	2.504072	...	-1.090541	2.609266	1.518725	7.147990	5.693736	1.454254	-0.65283
	-0.567843	...	-1.013782	-1.963597	-2.977379	-0.140554	-0.140554	0.000000	-1.15318
	-1.337711	...	-0.907369	-0.676021	-1.583390	1.057917	1.159442	-0.101525	-1.24496
	...	...	...	...	...	...	...	...	...
	-2.874806	...	-2.045551	0.153818	-1.891733	0.349452	0.349452	0.000000	-1.64461
	-0.288591	...	4.877143	-0.927539	3.949604	9.771244	9.128405	0.642839	5.38073
	-4.981171	...	0.772167	-7.362626	-6.590459	-0.191031	-0.095585	-0.095446	0.116561
	-1.798122	...	-2.278901	-3.226784	-5.505685	-0.525003	-0.488672	-0.036331	-2.17281
	-2.292199	...	-2.063214	1.586266	-0.476948	2.661869	2.148938	0.512932	-2.52503

In [404]:
```
test = test_set.merge(latest_raptor[['player_name', 'war_reg_season', 'war_playoffs']],
                      left_on = ['Player'],
                      right_on = ['player_name'], how = 'left')
```

In [405]:
```
test
```

m	G	GS	MP	FG	FGA	...	TOV	PF	PTS	APG	OREBPG	STLPG	BLKPG	player_name	war_reg_season	war_playof
R	55	12	20.7	3.6	7.3	...	1.1	1.9	9.2	0.016364	0.032727	0.010909	0.009091	Precious Achiuwa	0.405779	0.000000
M	42	42	27.0	3.7	6.3	...	1.9	2.3	8.6	0.054762	0.121429	0.021429	0.026190	Steven Adams	3.891773	0.000000
A	75	75	34.6	8.0	14.9	...	2.5	2.8	20.4	0.042667	0.033333	0.016000	0.010667	Bam Adebayo	5.693736	1.454254
A	59	22	20.5	2.8	6.5	...	0.7	1.7	7.9	0.018644	0.011864	0.005085	0.005085	Ochai Agbaji	-0.140554	0.000000
M	77	20	21.8	3.2	6.8	...	0.8	1.9	9.0	0.016883	0.014286	0.007792	0.007792	Santi Aldama	1.159442	-0.101525
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
R	54	9	14.7	2.0	3.7	...	0.8	1.6	4.4	0.025926	0.024074	0.018519	0.001852	Thaddeus Young	0.349452	0.000000
L	73	73	34.8	8.2	19.0	...	4.1	1.4	26.2	0.139726	0.010959	0.015068	0.001370	Trae Young	9.128405	0.642839
A	9	0	9.2	1.8	3.0	...	0.4	1.8	4.4	0.022222	0.100000	0.022222	0.022222	Omer Yurtseven	-0.095585	-0.095446
A	15	2	14.5	2.5	3.9	...	0.9	2.2	6.5	0.046667	0.113333	0.013333	0.020000	Cody Zeller	-0.488672	-0.036331
C	76	76	28.6	4.3	6.8	...	1.5	2.9	10.8	0.013158	0.040789	0.005263	0.017105	Ivica Zubac	2.148938	0.512932

In [406]:
```
test = test.dropna()
```

In [408]:
```
test = test[test['war_playoffs']!=0]
```

Now our test set is ready. We can run MLR on this set and observe the results:

In [411]:
```
test['war_playoffs_mlrpredicted'] = mlr_playoff_predictor.predict(
test[['PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season']])
```

```
/var/folders/j5/jb5rf2r16tq0vv5rg8_pkqf40000gn/T/ipykernel_18477/1327556940.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [412]:
```
test
```

FGA	...	PF	PTS	APG	OREBPG	STLPG	BLKPG	player_name	war_reg_season	war_playoffs	war_playoffs_mlrpredicte
14.9	...	2.8	20.4	0.042667	0.033333	0.016000	0.010667	Bam Adebayo	5.693736	1.454254	0.570544
6.8	...	1.9	9.0	0.016883	0.014286	0.007792	0.007792	Santi Aldama	1.159442	-0.101525	0.023161
5.0	...	1.5	6.2	0.030508	0.005085	0.008475	0.006780	Nickeil Alexander-Walker	2.413743	1.060769	0.177348
4.7	...	1.6	6.3	0.058333	0.005556	0.019444	0.011111	Nickeil Alexander-Walker	2.413743	1.060769	0.183148
5.4	...	1.3	5.9	0.060870	0.013043	0.013043	0.013043	Nickeil Alexander-Walker	2.413743	1.060769	0.172015
...	...	...	...	...	...	...	...	...	...	...	...
5.3	...	1.6	5.7	0.024324	0.010811	0.010811	0.005405	Ziaire Williams	0.357142	-0.161608	-0.079387
19.0	...	1.4	26.2	0.139726	0.010959	0.015068	0.001370	Trae Young	9.128405	0.642839	0.950126
3.0	...	1.8	4.4	0.022222	0.100000	0.022222	0.022222	Omer Yurtseven	-0.095585	-0.095446	-0.098490
3.9	...	2.2	6.5	0.046667	0.113333	0.013333	0.020000	Cody Zeller	-0.488672	-0.036331	-0.105034
6.8	...	2.9	10.8	0.013158	0.040789	0.005263	0.017105	Ivica Zubac	2.148938	0.512932	0.171865

In [415]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
mse = mean_squared_error(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("Mean Squared Error:", mse)
mae = mean_absolute_error(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("Mean Absolute Error:", mae)
r2 = r2_score(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("R2 Score:", r2)
```

```
Mean Squared Error: 0.38266416057783903
Mean Absolute Error: 0.3704110835354391
R2 Score: 0.2195987252287983
```

Predictions for the multiple linear regression model

In [255]:
```
combined_stats
```

M_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	FT_PCT	PPG	APG	OREBPG	DREBPG	
2760	OKC	24.0	75	1134.0	0.395	0.380	0.848	4.706667	0.373333	0.346667	1.173333	(
2760	OKC	25.0	31	588.0	0.357	0.323	0.923	5.322581	0.645161	0.161290	1.387097	(
2761	TOR	22.0	73	1725.0	0.439	0.359	0.595	9.095890	1.123288	2.000000	4.479452	(
2761	TOR	23.0	55	1141.0	0.485	0.269	0.702	9.236364	0.909091	1.818182	4.145455	(
2752	NYK	24.0	68	1287.0	0.459	0.300	0.784	5.852941	1.000000	1.161765	3.264706	(
...	...	...	...	...	...	...	...	...	...	...	...	.
	TOT	22.0	59	1039.0	0.559	0.000	0.802	8.898305	1.067797	1.949153	4.186441	(
2746	LAC	23.0	72	1326.0	0.613	0.000	0.747	8.277778	1.138889	2.736111	4.805556	(
2746	LAC	24.0	72	1609.0	0.652	0.250	0.789	9.027778	1.250000	2.625000	4.583333	(
2746	LAC	25.0	76	1852.0	0.626	0.000	0.727	10.328947	1.578947	2.855263	5.618421	(
2746	LAC	26.0	76	2169.0	0.634	0.000	0.697	10.763158	1.013158	3.105263	6.842105	(

To the combined_stats dataframe, predicted wins above replacement scores for the playoffs can be added as a seperate column. Once that is complete, Actual Vs. Predicted Plots can be created.

In [258]:
```
combined_stats['war_playoffs_mlrpredicted'] = mlr_playoff_predictor.predict(
combined_stats[['PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season']])
```

In [259]:
```
combined_stats
```

...	PPG	APG	OREBPG	DREBPG	STLPG	BLKPG	TOVPG	war_reg_season	war_playoffs	war_playoffs_mlrpredict
...	4.706667	0.373333	0.346667	1.173333	0.506667	0.106667	0.333333	1.447708	-0.198700	0.072839
...	5.322581	0.645161	0.161290	1.387097	0.548387	0.193548	0.451613	0.465912	0.311392	-0.014337
...	9.095890	1.123288	2.000000	4.479452	0.506849	0.561644	1.150685	-0.246776	0.000721	-0.142388
...	9.236364	0.909091	1.818182	4.145455	0.563636	0.545455	1.072727	2.309611	-0.046953	0.166756
...	5.852941	1.000000	1.161765	3.264706	0.397059	0.323529	0.882353	0.415918	0.000000	-0.044139
...	...	...	...	...	...	...	...	...	...	...
...	8.898305	1.067797	1.949153	4.186441	0.237288	0.864407	1.186441	0.531968	0.000000	-0.010978
...	8.277778	1.138889	2.736111	4.805556	0.222222	0.916667	0.847222	1.114995	0.102138	0.038108
...	9.027778	1.250000	2.625000	4.583333	0.333333	0.861111	1.125000	5.020903	1.203304	0.501210
...	10.328947	1.578947	2.855263	5.618421	0.473684	1.013158	1.500000	3.704630	0.556697	0.367862
...	10.763158	1.013158	3.105263	6.842105	0.381579	1.289474	1.539474	2.626958	0.000000	0.257569

Building a Random Forrest Regression Model

We saw above, that there do not seem to be strong linear relationships between the independent variables and the war_playoff statistic which we are looking to predict. The intention of building a random forrest regressor stems from the need to capture possible non-linear relationships between the independent and dependent variables. Relationships between features themselves can be assessed using this model. We saw above, that there do not seem to be strong linear relationships between the independent variables

In [298]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.inspection import plot_partial_dependence
```

In [433]:
```
X = combined_stats[['PLAYER_AGE', 'FG_PCT', 'FG3_PCT', 'PPG', 'APG', 'OREBPG', 'DREBPG',
                    'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season']]
y = combined_stats['war_playoffs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_playoff_predictor = RandomForestRegressor()
rf_playoff_predictor.fit(X_train, y_train)
y_pred = rf_playoff_predictor.predict(X_test)

print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
print(f"Train Score: {rf_playoff_predictor.score(X_train,y_train)}")
print(f"Test Score: {rf_playoff_predictor.score(X_test, y_test)}")
```

```
Mean Squared Error: 0.2899093649535073
Train Score: 0.9095691872788324
Test Score: 0.38006848148627637
```

It is immediatley noticed that a higher r2_score is attained. Now attempting to try using different combinations o independent variables to obtain a possibly higher r2_square.

In [174]:
```
combintation = []
for i in range(1,12):
    for comb in combinations(inputs, i):
        combination.append(comb)
combination_list = [list(t) for t in combination]
```

In [181]:
```
len(combination_list)
```

```
6142
```

In [203]:
```
r2_list = []
```

In [224]:
```
for comb in range(5000,6143):
    X = combined_stats[combination_list[comb]]
    y = combined_stats['war_playoffs']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    rf_playoff_predictor = RandomForestRegressor()
    rf_playoff_predictor.fit(X_train, y_train)
    y_pred = rf_playoff_predictor.predict(X_test)
    r2_list.append(r2_score(y_test, y_pred))
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [224], in <cell line: 1>()
      1 for comb in range(5000,6143):
----> 2     X = combined_stats[combination_list[comb]]
      3     y = combined_stats['war_playoffs']
      4     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

IndexError: list index out of range
```

In [222]:
```
len(combination_list)
```

```
6142
```

In [215]:
```
combination_list[2000]
```

```
['PLAYER_AGE',
 'FG_PCT',
 'FG3_PCT',
 'PPG',
 'OREBPG',
 'STLPG',
 'BLKPG',
 'TOVPG',
 'war_reg_season']
```

In [227]:
```
r2_list.index(max(r2_list))
```

```
1926
```

In [229]:
```
max(r2_list)
```

```
0.45816599440144246
```

In [233]:
```
combination_list[1924]
```

```
['PLAYER_AGE',
 'FG3_PCT',
 'APG',
 'OREBPG',
 'STLPG',
 'BLKPG',
 'TOVPG',
 'war_reg_season']
```

It looks like the maximum possible r2_score amongst all possible combinations of the independent variables is approximately 0.45816599440144246. The index (to find the exact varaible set) is around 1926.

In [238]:
```
for i in range(1930,1940):
    X = combined_stats[combination_list[i]]
    y = combined_stats['war_playoffs']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    rf_playoff_predictor = RandomForestRegressor()
    rf_playoff_predictor.fit(X_train, y_train)
    y_pred = rf_playoff_predictor.predict(X_test)
    print(r2_score(y_test, y_pred))
```

```
0.3898236710721418
0.3904329745816165
0.35438738557251515
0.3722064518194289
0.39105447545267846
0.42929749126169015
0.15643720429018326
0.14078621184044726
0.3618096495010171
0.12327458425894966
```

In [254]:
```
for i in range(1920,1930):
    X = combined_stats[combination_list[i]]
    y = combined_stats['war_playoffs']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    rf_playoff_predictor = RandomForestRegressor()
    rf_playoff_predictor.fit(X_train, y_train)
    y_pred = rf_playoff_predictor.predict(X_test)
    print(f'Combination: {combination_list[i]}.Score:{r2_score(y_test, y_pred)}')
```

```
Combination: ['PLAYER_AGE', 'FG3_PCT', 'PPG', 'DREBPG', 'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season'].Score:0.3793141809851724
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'OREBPG', 'DREBPG', 'STLPG', 'BLKPG', 'TOVPG'].Score:0.21704509793904148
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'OREBPG', 'DREBPG', 'STLPG', 'BLKPG', 'war_reg_season'].Score:0.4141975100556064
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'OREBPG', 'DREBPG', 'STLPG', 'TOVPG', 'war_reg_season'].Score:0.407251585063226
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'OREBPG', 'DREBPG', 'BLKPG', 'TOVPG', 'war_reg_season'].Score:0.422408609375385
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season'].Score:0.4210197100613979
Combination: ['PLAYER_AGE', 'FG3_PCT', 'APG', 'DREBPG', 'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season'].Score:0.4172825766061111
Combination: ['PLAYER_AGE', 'FG3_PCT', 'OREBPG', 'DREBPG', 'STLPG', 'BLKPG', 'TOVPG', 'war_reg_season'].Score:0.4034938542189
2
Combination: ['PLAYER_AGE', 'PPG', 'APG', 'OREBPG', 'DREBPG', 'STLPG', 'BLKPG', 'TOVPG'].Score:0.1708134426102218
Combination: ['PLAYER_AGE', 'PPG', 'APG', 'OREBPG', 'DREBPG', 'STLPG', 'BLKPG', 'war_reg_season'].Score:0.3762479879144919
```

In [435]:
```
X = combined_stats[['PLAYER_AGE',
     'FG3_PCT',
     'APG',
     'OREBPG',
     'DREBPG',
     'BLKPG',
     'TOVPG',
     'war_reg_season']]
y = combined_stats['war_playoffs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_playoff_predictor = RandomForestRegressor()
rf_playoff_predictor.fit(X_train, y_train)
y_pred = rf_playoff_predictor.predict(X_test)
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred)}")
print(f"Train Score: {rf_playoff_predictor.score(X_train,y_train)}")
print(f"Test Score: {rf_playoff_predictor.score(X_test, y_test)}")
```

```
Mean Squared Error: 0.26489367131690505
Train Score: 0.9078867994619302
Test Score: 0.43356112028150795
```

In [262]:
```
combined_stats
```

	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	MIN	FG_PCT	FG3_PCT	...	APG	OREBPG	DREBPG	STLPG
	1610612760	OKC	24.0	75	1134.0	0.395	0.380	...	0.373333	0.346667	1.173333	0.506667
	1610612760	OKC	25.0	31	588.0	0.357	0.323	...	0.645161	0.161290	1.387097	0.548387
	1610612761	TOR	22.0	73	1725.0	0.439	0.359	...	1.123288	2.000000	4.479452	0.506849
	1610612761	TOR	23.0	55	1141.0	0.485	0.269	...	0.909091	1.818182	4.145455	0.563636
	1610612752	NYK	24.0	68	1287.0	0.459	0.300	...	1.000000	1.161765	3.264706	0.397059
	...	...	...	...	...	...	...	...	...	...	...	...
	0	TOT	22.0	59	1039.0	0.559	0.000	...	1.067797	1.949153	4.186441	0.237288
	1610612746	LAC	23.0	72	1326.0	0.613	0.000	...	1.138889	2.736111	4.805556	0.222222
	1610612746	LAC	24.0	72	1609.0	0.652	0.250	...	1.250000	2.625000	4.583333	0.333333
	1610612746	LAC	25.0	76	1852.0	0.626	0.000	...	1.578947	2.855263	5.618421	0.473684
	1610612746	LAC	26.0	76	2169.0	0.634	0.000	...	1.013158	3.105263	6.842105	0.381579

We now have the combined dataset with predicted values from both models - multiple linear regression and random forrest models. Saving as a csv for further visualization in Tableau.

In [263]:
```
combined_stats.to_csv('combined_stats_withpred.csv', index = False)
```

Testing

In [418]:
```
test
```

	Rk	Player	Pos	PLAYER_AGE	Tm	G	GS	MP	FG	FGA	...	PF	PTS	APG	OREBPG	STLPG	BLKPG	pl
2	3	Bam Adebayo	C	25	MIA	75	75	34.6	8.0	14.9	...	2.8	20.4	0.042667	0.033333	0.016000	0.010667	Ba
4	5	Santi Aldama	PF	22	MEM	77	20	21.8	3.2	6.8	...	1.9	9.0	0.016883	0.014286	0.007792	0.007792	Sa
5	6	Nickeil Alexander-Walker	SG	24	TOT	59	3	15.0	2.2	5.0	...	1.5	6.2	0.030508	0.005085	0.008475	0.006780	Nic Ale Wa
6	6	Nickeil Alexander-Walker	SG	24	UTA	36	3	14.7	2.3	4.7	...	1.6	6.3	0.058333	0.005556	0.019444	0.011111	Nic Ale Wa
7	6	Nickeil Alexander-Walker	SG	24	MIN	23	0	15.5	2.1	5.4	...	1.3	5.9	0.060870	0.013043	0.013043	0.013043	Nic Ale Wa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
663	526	Ziaire Williams	SF	21	MEM	37	4	15.2	2.3	5.3	...	1.6	5.7	0.024324	0.010811	0.010811	0.005405	Zia
675	536	Trae Young	PG	24	ATL	73	73	34.8	8.2	19.0	...	1.4	26.2	0.139726	0.010959	0.015068	0.001370	Tra
676	537	Omer Yurtseven	C	24	MIA	9	0	9.2	1.8	3.0	...	1.8	4.4	0.022222	0.100000	0.022222	0.022222	On Yu
677	538	Cody Zeller	C	30	MIA	15	2	14.5	2.5	3.9	...	2.2	6.5	0.046667	0.113333	0.013333	0.020000	Co
678	539	Ivica Zubac	C	25	LAC	76	76	28.6	4.3	6.8	...	2.9	10.8	0.013158	0.040789	0.005263	0.017105	Ivi

265 rows × 38 columns

In [419]:
```
test.columns
```

```
Index(['Rk', 'Player', 'Pos', 'PLAYER_AGE', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA',
       'FG_PCT', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'APG', 'OREBPG', 'STLPG', 'BLKPG', 'player_name', 'war_reg_season',
       'war_playoffs', 'war_playoffs_mlrpredicted'],
      dtype='object')
```

In [420]:
```
test = test.rename(columns = {'3P%':'FG3_PCT'})
test['DREBPG'] = test['DRB'] / test['G']
test['TOVPG'] = test['TOV'] / test['G']
```

In [422]:
```
test = test.drop(['Rk','Pos','Tm','FG','FGA','3P','3PA','2P','2PA','2P%','eFG%','FT',
                  'FTA','ORB','DRB','TRB','AST','STL','BLK','TOV','PF','PTS'],axis = 1)
```

Now we can run our Random Forest predictor on this set

In [427]:
```
test['war_playoffs_rfpredicted'] = rf_playoff_predictor.predict(
test[['PLAYER_AGE','FG3_PCT','APG','OREBPG','DREBPG','BLKPG','TOVPG','war_reg_season']])
```

RESULTS

In [430]:
```
mse = mean_squared_error(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("Mean Squared Error:", mse)
mae = mean_absolute_error(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("Mean Absolute Error:", mae)
r2 = r2_score(test['war_playoffs'], test['war_playoffs_mlrpredicted'])
print("R2 Score:", r2)

mse = mean_squared_error(test['war_playoffs'], test['war_playoffs_rfpredicted'])
print("Mean Squared Error:", mse)
mae = mean_absolute_error(test['war_playoffs'], test['war_playoffs_rfpredicted'])
print("Mean Absolute Error:", mae)
r2 = r2_score(test['war_playoffs'], test['war_playoffs_rfpredicted'])
print("R2 Score:", r2)
```

```
Mean Squared Error: 0.38266416057783903
Mean Absolute Error: 0.3704110835354391
R2 Score: 0.2195987252287983
Mean Squared Error: 0.35864558223443554
Mean Absolute Error: 0.369026935256167
R2 Score: 0.26858196193714234
```

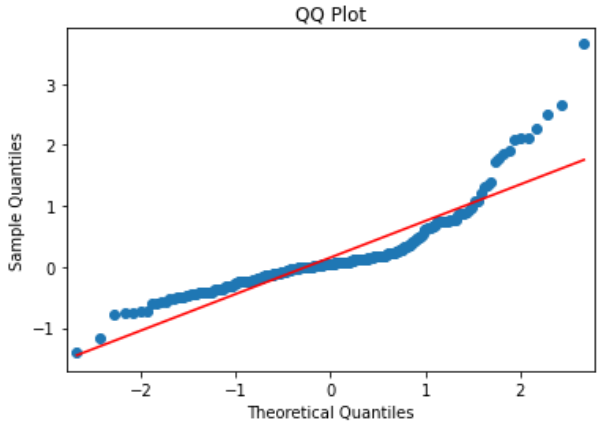
In [431]:
```
test.to_csv('test_results.csv', index = False)
```

In [436]:
```
test[]
```

REBPG	STLPG	BLKPG	war_reg_season	war_playoffs	war_playoffs_mlrpredicted	DREBPG	TOVPG	war_playoffs_rfpredict
033333	0.016000	0.010667	5.693736	1.454254	0.570544	0.089333	0.033333	0.568552
014286	0.007792	0.007792	1.159442	-0.101525	0.023161	0.048052	0.010390	-0.003005
005085	0.008475	0.006780	2.413743	1.060769	0.177348	0.025424	0.015254	0.449987
005556	0.019444	0.011111	2.413743	1.060769	0.183148	0.038889	0.036111	0.414509
013043	0.013043	0.013043	2.413743	1.060769	0.172015	0.065217	0.017391	0.452921
...	...	...	...	...	...	...	...	...
010811	0.010811	0.005405	0.357142	-0.161608	-0.079387	0.045946	0.027027	-0.019577
010959	0.015068	0.001370	9.128405	0.642839	0.950126	0.030137	0.056164	0.977216
100000	0.022222	0.022222	-0.095585	-0.095446	-0.098490	0.188889	0.044444	-0.043132
113333	0.013333	0.020000	-0.488672	-0.036331	-0.105034	0.173333	0.060000	-0.015430
040789	0.005263	0.017105	2.148938	0.512932	0.171865	0.089474	0.019737	0.295122

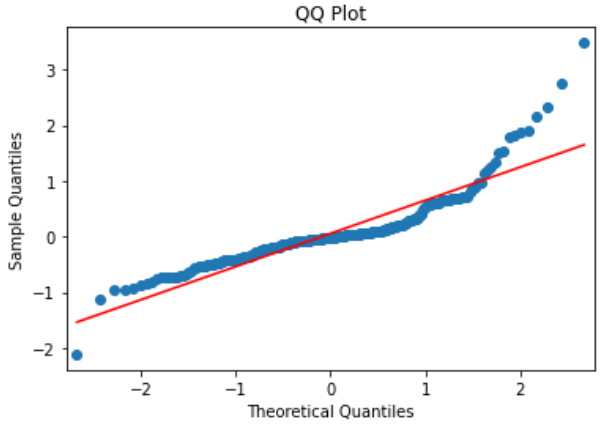
In [437]:
```
test['Residuals_ML'] = test['war_playoffs'] - test['war_playoffs_mlrpredicted']
```

In [438]:
```
sm.qqplot(test['Residuals_ML'], line='s')
plt.title("QQ Plot")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.show()
```



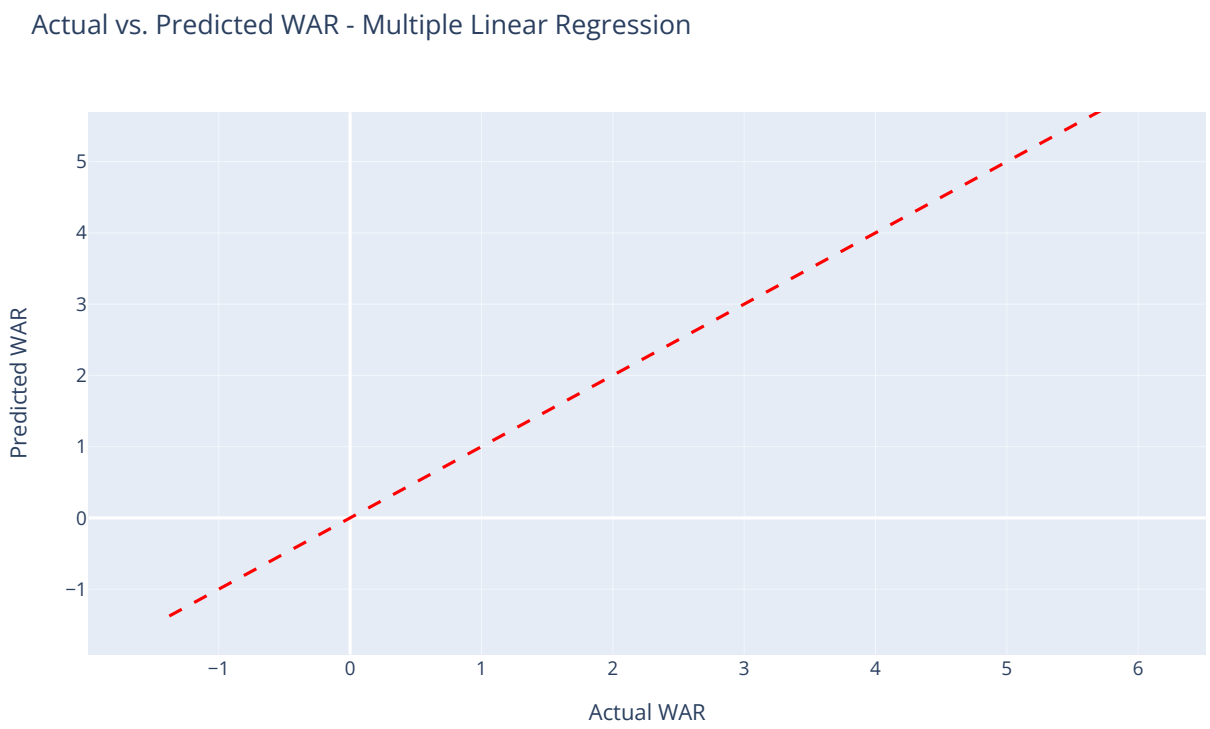
In [439]:
```
test['Residuals_RF'] = test['war_playoffs'] - test['war_playoffs_rfpredicted']
```

In [440]:
```
sm.qqplot(test['Residuals_RF'], line='s')
plt.title("QQ Plot")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.show()
```

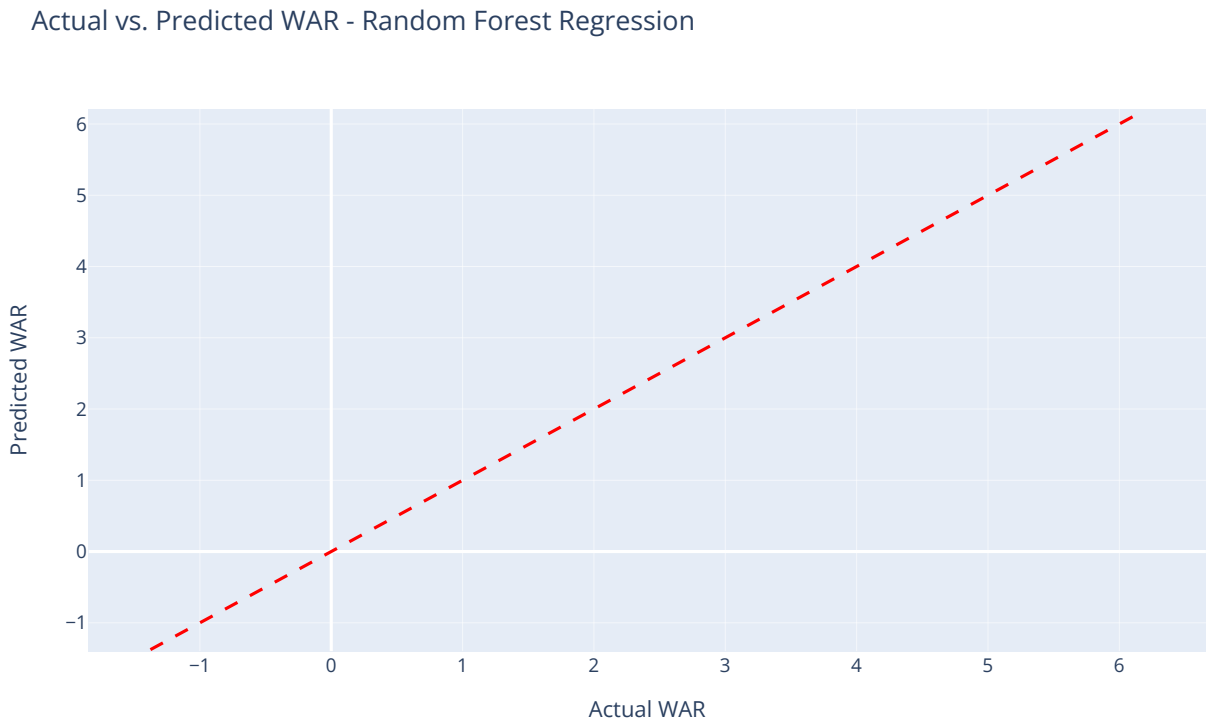


Actual Vs. Predicted WAR Scatter Plots

In [267]:
```
x='war_playoffs', y='war_playoffs_mlrpredicted', title='Actual vs. Predicted WAR - Multiple Linear
ombined_stats['war_playoffs'].min(), y0=combined_stats['war_playoffs'].min(),
'war_playoffs'].max(), y1=combined_stats['war_playoffs'].max(),
ed', dash='dash'))
ctual WAR')
redicted WAR')
```



In [268]:
```
px.scatter(combined_stats, x='war_playoffs', y='war_playoffs_rfpredicted', title='Actual vs. Predi
.add_shape(type='line', x0=combined_stats['war_playoffs'].min(), y0=combined_stats['war_playoffs'].mi
           x1=combined_stats['war_playoffs'].max(), y1=combined_stats['war_playoffs'].max(),
           line=dict(color='red', dash='dash'))
.update_xaxes(title_text = 'Actual WAR')
.update_yaxes(title_text = 'Predicted WAR')
.show()
```

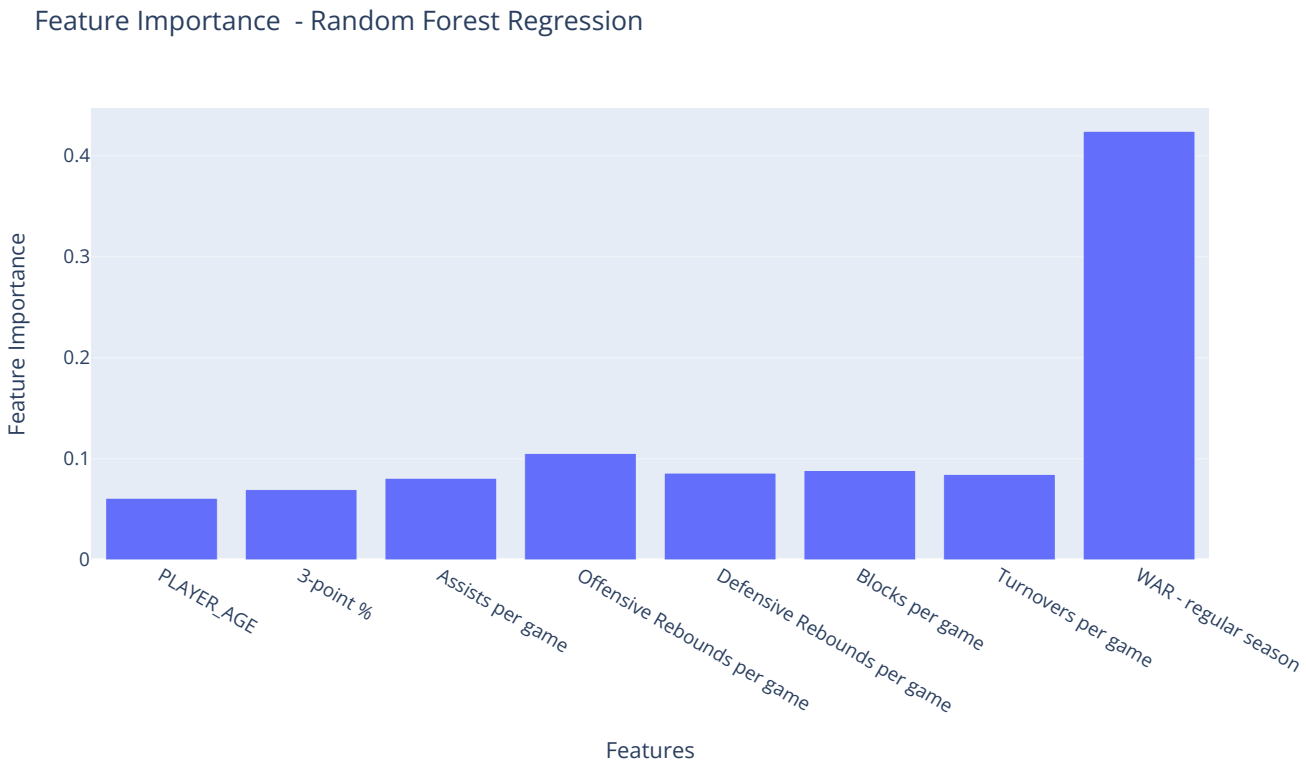


Feature Importance Plots

In [356]:
```
features = ['PLAYER_AGE','3-point %','Assists per game','Offensive Rebounds per game','Defensive Re
feature_imp = rf_playoff_predictor.feature_importances_.tolist()
feature_imp
```

```
[0.06276382927257879,
 0.06758196248109885,
 0.08518031314926752,
 0.09841201987045844,
 0.08789212963812326,
 0.08555375262857162,
 0.09020578229681121,
 0.42241021066309037]
```

In [288]:
```
fig = px.bar(x=features, y=feature_imp, title='Feature Importance  - Random Forest Regression')
fig.update_xaxes(title_text = 'Features')
fig.update_yaxes(title_text = 'Feature Importance')
fig.show()
```



In [ ]:
```
#'PLAYER_AGE', 'FG_PCT', 'APG', 'OREBPG', 'STLPG', 'BLKPG', 'war_reg_season'
```

In [354]:
```
features = ['PLAYER_AGE','Field Goal %','Assists per game','Offensive Rebounds per game','Steals p
feature_imp = mlr_playoff_predictor.coef_.tolist()
```

In [355]:
```
fig = px.bar(x=features, y=feature_imp, title='Feature Importance  - Multiple Linear Regression')
fig.update_xaxes(title_text = 'Features')
fig.update_yaxes(title_text = 'Feature Importance')
fig.show()
```

Partial Dependence Plots

In [302]:
```
rf_playoff_predictor.resid
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [302], in <cell line: 1>()
----> 1 rf_playoff_predictor.resid

AttributeError: 'RandomForestRegressor' object has no attribute 'resid'
```

In [304]:
```
ined_stats['Residuals_RF'] =  combined_stats['war_playoffs'] - combined_stats['war_playoffs_rfpredic
```

In [305]:
```
combined_stats
```

PG	STLPG	BLKPG	TOVPG	war_reg_season	war_playoffs	war_playoffs_mlrpredicted	war_playoffs_rfpredicted	Residuals_l
33	0.506667	0.106667	0.333333	1.447708	-0.198700	0.072839	-0.127594	-0.071106
97	0.548387	0.193548	0.451613	0.465912	0.311392	-0.014337	0.192320	0.119072
52	0.506849	0.561644	1.150685	-0.246776	0.000721	-0.142388	0.008770	-0.008048
55	0.563636	0.545455	1.072727	2.309611	-0.046953	0.166756	-0.001794	-0.045159
06	0.397059	0.323529	0.882353	0.415918	0.000000	-0.044139	0.012497	-0.012497
...	...	...	...	...	...	...	...	...
41	0.237288	0.864407	1.186441	0.531968	0.000000	-0.010978	0.002181	-0.002181
56	0.222222	0.916667	0.847222	1.114995	0.102138	0.038108	0.095505	0.006633
33	0.333333	0.861111	1.125000	5.020903	1.203304	0.501210	0.599630	0.603674
21	0.473684	1.013158	1.500000	3.704630	0.556697	0.367862	0.285210	0.271487
05	0.381579	1.289474	1.539474	2.626958	0.000000	0.257569	0.032597	-0.032597

In [306]:
```
from statsmodels.stats.outliers_influence import OLSInfluence
```

In [319]:
```
oob_scores = rf_playoff_predictor.oob_prediction_
mdi_values = rf_playoff_predictor.feature_importances_
cooks_distance_proxy = (oob_scores**2) / mdi_values
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [319], in <cell line: 1>()
----> 1 oob_scores = rf_playoff_predictor.oob_prediction_
      2 mdi_values = rf_playoff_predictor.feature_importances_
      3 cooks_distance_proxy = (oob_scores**2) / mdi_values

AttributeError: 'RandomForestRegressor' object has no attribute 'oob_prediction_'
```

In [320]:
```
rf_playoff_predictor
```

```
RandomForestRegressor()
```

In [321]:
```
independent_variables = ['age', 'build']
enumerate(independent_variables)
```

```
<enumerate at 0x7fe1c31636c0>
```

In [323]:
```
X = combined_stats[['PLAYER_AGE',
 'FG3_PCT',
 'APG',
 'OREBPG',
 'DREBPG',
 'BLKPG',
 'TOVPG',
 'war_reg_season']]
y = combined_stats['war_playoffs']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_playoff_predictor = RandomForestRegressor()
rf_playoff_predictor.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

In [330]:
```
independent_variables = ['PLAYER_AGE',
 'FG3_PCT',
 'APG',
 'OREBPG',
 'DREBPG',
 'BLKPG',
 'TOVPG',
 'war_reg_season']
```

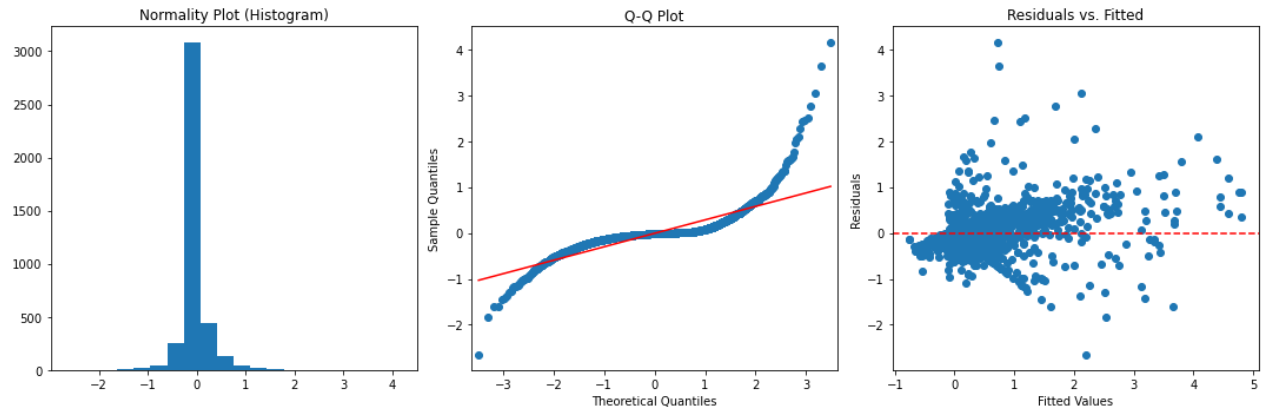
In [334]:
```
import statsmodels.api as sm
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Normality plot (histogram)
axes[0].hist(combined_stats['Residuals_RF'], bins=20)
axes[0].set_title('Normality Plot (Histogram)')

# Q-Q plot
sm.qqplot(combined_stats['Residuals_RF'], line='s', ax=axes[1])
axes[1].set_title('Q-Q Plot')

# Residuals vs. Fitted plot
axes[2].scatter(combined_stats['war_playoffs_rfpredicted'], combined_stats['Residuals_RF'])
axes[2].axhline(0, color='red', linestyle='--')
axes[2].set_xlabel('Fitted Values')
axes[2].set_ylabel('Residuals')
axes[2].set_title('Residuals vs. Fitted')

plt.tight_layout()
plt.show()
```

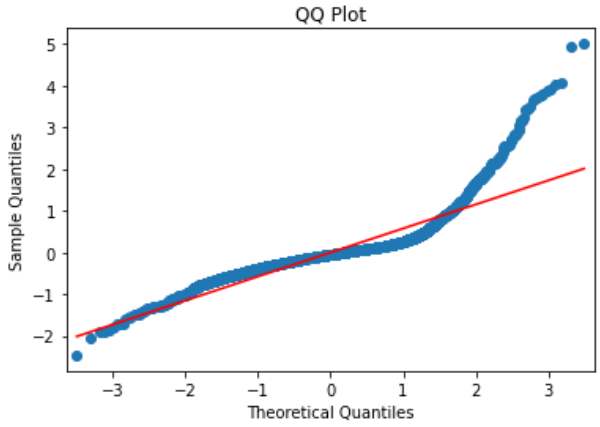


In [343]:
```
d_stats['Residuals_ML'] = combined_stats['war_playoffs'] - combined_stats['war_playoffs_mlrpredicted
```

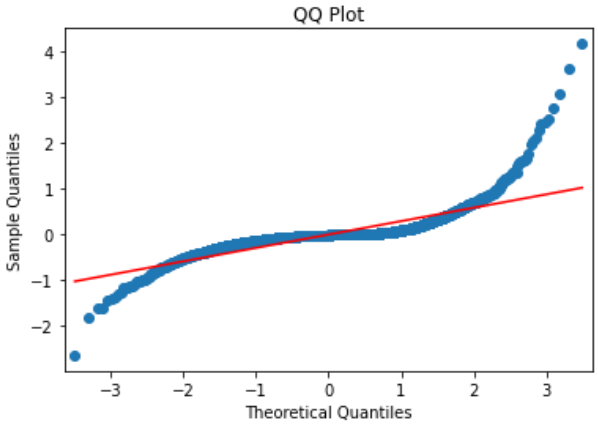
In [346]:
```
combined_stats
```

LKPG	TOVPG	war_reg_season	war_playoffs	war_playoffs_mlrpredicted	war_playoffs_rfpredicted	Residuals_RF	Residuals_M
106667	0.333333	1.447708	-0.198700	0.072839	-0.127594	-0.071106	-0.271540
193548	0.451613	0.465912	0.311392	-0.014337	0.192320	0.119072	0.325730
561644	1.150685	-0.246776	0.000721	-0.142388	0.008770	-0.008048	0.143109
545455	1.072727	2.309611	-0.046953	0.166756	-0.001794	-0.045159	-0.213709
323529	0.882353	0.415918	0.000000	-0.044139	0.012497	-0.012497	0.044139
	...	...	...	...	...	...	...
864407	1.186441	0.531968	0.000000	-0.010978	0.002181	-0.002181	0.010978
916667	0.847222	1.114995	0.102138	0.038108	0.095505	0.006633	0.064031
861111	1.125000	5.020903	1.203304	0.501210	0.599630	0.603674	0.702094
013158	1.500000	3.704630	0.556697	0.367862	0.285210	0.271487	0.188835
289474	1.539474	2.626958	0.000000	0.257569	0.032597	-0.032597	-0.257569

In [350]:
```
sm.qqplot(combined_stats['Residuals_ML'], line='s')
plt.title("QQ Plot")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.show()
```



In [351]:
```
sm.qqplot(combined_stats['Residuals_RF'], line='s')
plt.title("QQ Plot")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.show()
```



In [ ]:

In [8]:
```
from nba_api.stats.endpoints import playercareerstats
```

Top 100 leaders by points in the 2022-23 playoffs

In [277]:
```
top_500 = leagueleaders.LeagueLeaders(
    season='1951-52',
    season_type_all_star='Regular Season',
    stat_category_abbreviation='PTS'
).get_data_frames()[0][:400]
top_500
```

	PLAYER_ID	RANK	PLAYER	TEAM_ID	TEAM	GP	MIN	FGM	FGA	FG_PCT	...	REB	AST	STL	BLK	TOV	PF	PTS
0	76056	1	Paul Arizin	1610612744	PHW	66	2939	548	1222	0.448	...	745	170	None	None	None	250	167
1	600012	2	George Mikan	1610612747	MNL	64	2572	545	1414	0.385	...	866	194	None	None	None	286	152
2	600003	3	Bob Cousy	1610612738	BOS	66	2681	512	1388	0.369	...	421	441	None	None	None	190	143
3	77429	4	Ed Macauley	1610612738	BOS	66	2631	384	888	0.432	...	529	232	None	None	None	174	126
4	76514	5	Bob Davies	1610612758	ROC	65	2394	379	990	0.383	...	189	390	None	None	None	269	105
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
110	77941	111	John Rennicke	1610612737	MIH	6	54	4	18	0.222	...	9	1	None	None	None	7	11
111	76745	112	Jerry Fowler	1610612737	MIH	6	41	4	13	0.308	...	10	2	None	None	None	9	9
112	77723	113	Paul Noel	1610612758	ROC	8	32	2	9	0.222	...	4	3	None	None	None	6	6
113	77859	114	John Pilch	1610612747	MNL	9	41	1	10	0.100	...	9	2	None	None	None	10	5
114	78163	115	Zeke Sinicola	1610612765	FTW	3	15	1	4	0.250	...	1	0	None	None	None	2	2

115 rows × 28 columns

Earliest year is 1951

In [152]:
```
# DataSet Size Calculation
regseason_teams = 30
playoff_teams = 16
players = 15
years = 50
top_500 = 500
top_150 = 150
```

In [259]:
```
regseason_teams*players*years + playoff_teams*players*years + top_500*years + top_150*years  + (57
```

```
111975
```

DRAFT STATS

Draft Combine Drill Results

In [171]:
```
from nba_api.stats.endpoints import draftcombinedrillresults
```

In [211]:
```
test = draftcombinedrillresults.DraftCombineDrillResults(season_year = 2000).get_data_frames()[0]
test
```

LEAP	MAX_VERTICAL_LEAP	LANE_AGILITY_TIME	MODIFIED_LANE_AGILITY_TIME	THREE_QUARTER_SPRINT	BENCH_PRESS
29.0	11.83	None	3.38	13.0
29.0	13.80	None	NaN	0.0
29.5	12.30	None	3.40	10.0
31.0	13.04	None	3.47	15.0
29.5	11.53	None	3.55	NaN
...	...	...	...	...
30.0	12.45	None	3.43	15.0
29.0	11.98	None	3.46	13.0
31.0	12.08	None	3.33	7.0
35.5	10.75	None	3.23	0.0
30.5	12.38	None	3.45	11.0

Draft History

In [190]:
```
from nba_api.stats.endpoints import drafthistory
```

In [220]:
```
test = drafthistory.DraftHistory(season_year_nullable = 2000).get_data_frames()[0]
test
```

	PERSON_ID	PLAYER_NAME	SEASON	ROUND_NUMBER	ROUND_PICK	OVERALL_PICK	DRAFT_TYPE	TEAM_ID	TEAM
0	2030	Kenyon Martin	2000	1	1	1	Draft	1610612751	New Je
1	2031	Stromile Swift	2000	1	2	2	Draft	1610612763	Vancou
2	2032	Darius Miles	2000	1	3	3	Draft	1610612746	Los An
3	2033	Marcus Fizer	2000	1	4	4	Draft	1610612741	Chicag
4	2034	Mike Miller	2000	1	5	5	Draft	1610612753	Orland
5	2035	DerMarr Johnson	2000	1	6	6	Draft	1610612737	Atlanta
6	2036	Chris Mihm	2000	1	7	7	Draft	1610612741	Chicag
7	2037	Jamal Crawford	2000	1	8	8	Draft	1610612739	Clevela
8	2038	Joel Przybilla	2000	1	9	9	Draft	1610612745	Housto
9	2039	Keyon Dooling	2000	1	10	10	Draft	1610612753	Orland
10	2040	Jerome Moiso	2000	1	11	11	Draft	1610612738	Boston
11	2041	Etan Thomas	2000	1	12	12	Draft	1610612742	Dallas
12	2042	Courtney Alexander	2000	1	13	13	Draft	1610612753	Orland
13	2043	Mateen Cleaves	2000	1	14	14	Draft	1610612765	Detroit
14	2044	Jason Collier	2000	1	15	15	Draft	1610612749	Milwau
15	2045	Hedo Turkoglu	2000	1	16	16	Draft	1610612758	Sacran
16	2046	Desmond Mason	2000	1	17	17	Draft	1610612760	Seattle
17	2047	Quentin Richardson	2000	1	18	18	Draft	1610612746	Los An
18	2048	Jamaal Magloire	2000	1	19	19	Draft	1610612766	Charlot
19	2049	Speedy Claxton	2000	1	20	20	Draft	1610612755	Philade
20	2050	Morris Peterson	2000	1	21	21	Draft	1610612761	Toronto
21	2051	Donnell Harvey	2000	1	22	22	Draft	1610612752	New Yo
22	2052	DeShawn Stevenson	2000	1	23	23	Draft	1610612762	Utah
23	2053	Dalibor Bagaric	2000	1	24	24	Draft	1610612741	Chicag
24	2054	Jake Tsakalidis	2000	1	25	25	Draft	1610612756	Phoeni
25	2055	Mamadou N'diaye	2000	1	26	26	Draft	1610612743	Denver
26	2056	Primoz Brezec	2000	1	27	27	Draft	1610612754	Indiana
27	2057	Erick Barkley	2000	1	28	28	Draft	1610612757	Portlan
28	2058	Mark Madsen	2000	1	29	29	Draft	1610612747	Los An
29	2060	Marko Jaric	2000	2	1	30	Draft	1610612746	Los An
30	2061	Dan Langhi	2000	2	2	31	Draft	1610612742	Dallas
31	2062	A.J. Guyton	2000	2	3	32	Draft	1610612741	Chicag
32	2063	Jake Voskuhl	2000	2	4	33	Draft	1610612741	Chicag
33	2064	Khalid El-Amin	2000	2	5	34	Draft	1610612741	Chicag
34	2065	Mike Smith	2000	2	6	35	Draft	1610612764	Washir
35	2066	Soumaila Samake	2000	2	7	36	Draft	1610612751	New Je
36	2067	Eddie House	2000	2	8	37	Draft	1610612748	Miami
37	2059	Eduardo Najera	2000	2	9	38	Draft	1610612745	Housto
38	2068	Lavor Postell	2000	2	10	39	Draft	1610612752	New Yo
39	2069	Hanno Mottola	2000	2	11	40	Draft	1610612737	Atlanta
40	2070	Chris Carrawell	2000	2	12	41	Draft	1610612759	San An
41	2071	Olumide Oyedeji	2000	2	13	42	Draft	1610612760	Seattle
42	2072	Michael Redd	2000	2	14	43	Draft	1610612749	Milwau
43	2073	Brian Cardinal	2000	2	15	44	Draft	1610612765	Detroit
44	2074	Jabari Smith	2000	2	16	45	Draft	1610612758	Sacran
45	2075	DeeAndre Hulett	2000	2	17	46	Draft	1610612761	Toronto
46	2076	Josip Sesar	2000	2	18	47	Draft	1610612760	Seattle
47	2077	Mark Karcher	2000	2	19	48	Draft	1610612755	Philade

	PERSON_ID	PLAYER_NAME	SEASON	ROUND_NUMBER	ROUND_PICK	OVERALL_PICK	DRAFT_TYPE	TEAM_ID	TEAM
48	2078	Jason Hart	2000	2	20	49	Draft	1610612749	Milwau
49	2079	Kaniel Dickens	2000	2	21	50	Draft	1610612762	Utah
50	2080	Igor Rakocevic	2000	2	22	51	Draft	1610612750	Minnes
51	2081	Ernest Brown	2000	2	23	52	Draft	1610612748	Miami
52	2082	Dan McClintock	2000	2	24	53	Draft	1610612743	Denver
53	2083	Cory Hightower	2000	2	25	54	Draft	1610612759	San An
54	2084	Chris Porter	2000	2	26	55	Draft	1610612744	Golden
55	2085	Jaquay Walls	2000	2	27	56	Draft	1610612754	Indiana
56	2086	Scoonie Penn	2000	2	28	57	Draft	1610612737	Atlanta
57	2087	Pete Mickeal	2000	2	29	58	Draft	1610612742	Dallas

Draft Combine Stats

In [213]:
```
from nba_api.stats.endpoints import draftcombinestats
```

In [217]:
```
test = draftcombinestats.DraftCombineStats(season_all_time = 2000).get_data_frames()[0]
test
```

	SEASON	PLAYER_ID	FIRST_NAME	LAST_NAME	PLAYER_NAME	POSITION	HEIGHT_WO_SHOES	HEIGHT_WO_SHOES_F
0	2000	2124	Malik	Allen	Malik Allen	PF-C	80.25	6' 8.25"
1	2000	12019	Harold	Arceneaux	Harold Arceneaux	SG-SF	76.50	6' 4.5"
2	2000	12020	Lamont	Barnes	Lamont Barnes	PF-C	80.50	6' 8.5"
3	2000	12131	Mario	Bland	Mario Bland	PF	77.50	6' 5.5"
4	2000	2056	Primoz	Brezec	Primoz Brezec	C	84.75	7' 0.75"
...	...	...	...	...	...	...	...	...
60	2000	2074	Jabari	Smith	Jabari Smith	PF-C	81.75	6' 9.75"
61	2000	12027	Jarrett	Stephens	Jarrett Stephens	PF	77.25	6' 5.25"
62	2000	12028	Bootsy	Thornton	Bootsy Thornton	SG	75.50	6' 3.5"
63	2000	12029	Jaquay	Walls	Jaquay Walls	PG	73.25	6' 1.25"
64	2000	12030	Jameel	Watkins	Jameel Watkins	PF-C	81.50	6' 9.5"

65 rows × 24 columns

Draft Combine Anthro

In [224]:
```
from nba_api.stats.endpoints import draftcombineplayeranthro
```

In [235]:
```
test = draftcombineplayeranthro.DraftCombinePlayerAnthro(season_year = 2023).get_data_frames()[0]
test
```

	TEMP_PLAYER_ID	PLAYER_ID	FIRST_NAME	LAST_NAME	PLAYER_NAME	POSITION	HEIGHT_WO_SHOES	HEIGHT_WO_
0	1641725	1641725	Trey	Alexander	Trey Alexander	SG	75.25	6' 3.25"
1	1641735	1641735	Amari	Bailey	Amari Bailey	SG	75.25	6' 3.25"
2	1641734	1641734	Emoni	Bates	Emoni Bates	SF	80.25	6' 8.25"
3	1641736	1641736	Reece	Beekman	Reece Beekman	PG	73.50	6' 1.50"
4	1641710	1641710	Anthony	Black	Anthony Black	PG	77.75	6' 5.75"
...	...	...	...	...	...	...	...	...
76	1641705	1641705	Victor	Wembanyama	Victor Wembanyama	C	NaN	None
77	1641727	1641727	Dariq	Whitehead	Dariq Whitehead	SG	77.75	6' 5.75"
78	1641715	1641715	Cam	Whitmore	Cam Whitmore	SF	77.75	6' 5.75"
79	1630592	1630592	Jalen	Wilson	Jalen Wilson	SF	77.50	6' 5.50"
80	1631209	1631209	Isaiah	Wong	Isaiah Wong	SG	74.50	6' 2.50"

81 rows × 18 columns

Research Question 2

In this question, we are looking to explore the relationship between a rookie's performance in the draft combine and their performance in their first season in the NBA.

Importing the necessary data from using the nba_api package

In [2]:
```
from nba_api.stats.endpoints import draftcombinedrillresults
from nba_api.stats.endpoints import draftcombineplayeranthro
import pandas as pd
```

In [3]:
```
rap_modern = pd.read_csv('modern_RAPTOR_by_player.csv')
```

Begenning with assimilating draft combine drills data

In [17]:
```
list_of_years = [2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
draft_drill_dfs = []
for y in list_of_years:
    draft_drill_dfs.append(draftcombinedrillresults.DraftCombineDrillResults(season_year = y)
                           .get_data_frames()[0])
```

In [21]:
```
draft_drill = pd.concat(draft_drill_dfs)
```

In [25]:
```
draft_drill = draft_drill.dropna()
```

In [26]:
```
draft_drill
```

LEAP	MAX_VERTICAL_LEAP	LANE_AGILITY_TIME	MODIFIED_LANE_AGILITY_TIME	THREE_QUARTER_SPRINT	BENCH_PRESS
	43.0	11.13	2.88	3.22	12.0
	35.5	11.21	3.35	3.58	9.0
	41.0	10.71	3.05	3.26	10.0
	34.5	10.27	2.75	3.28	8.0
	34.5	10.94	3.29	3.18	4.0
	...	...	...	...	...
	35.5	11.85	3.24	3.30	11.0
	40.5	10.76	3.11	3.07	5.0
	35.0	10.39	3.09	3.34	4.0
	31.5	10.83	3.27	3.33	20.0
	37.5	10.70	2.94	3.30	8.0

Now raptor data from the rap_modern dataframe can be added to the df above. Since we only want rookie season stats from the rap_modern dataframe, we can filter:

In [33]:
```
rap_modern = rap_modern.drop_duplicates(subset = ['player_name'])
```

In [34]:
```
rap_modern
```

	player_name	player_id	season	poss	mp	raptor_box_offense	raptor_box_defense	raptor_box_total	raptor_onoff_offer
0	Alex Abrines	abrinal01	2017	2387	1135	0.745505	-0.372938	0.372567	-0.418553
3	Precious Achiuwa	achiupr01	2021	1581	749	-4.122966	1.359278	-2.763688	-4.050779
5	Quincy Acy	acyqu01	2014	1716	847	-1.716079	0.133115	-1.582964	-0.324811
11	Jaylen Adams	adamsja01	2019	952	428	-2.679920	-4.592002	-7.271921	-0.535827
14	Jordan Adams	adamsjo01	2015	518	258	2.311410	-0.266178	2.045233	0.583715
...	...	...	...	...	...	...	...	...	...
4666	Tyler Zeller	zellety01	2014	2088	1049	-1.256580	2.095302	0.838722	0.321454
4673	Stephen Zimmerman	zimmest01	2017	219	108	-4.290836	-10.321977	-14.612813	-1.332745
4674	Paul Zipser	zipsepa01	2017	1970	979	-2.300085	-1.499653	-3.799738	-0.941495
4676	Ante Zizic	zizican01	2018	498	237	1.436917	-0.686749	0.750168	-2.438343
4679	Ivica Zubac	zubaciv01	2017	1305	609	-3.320225	-2.901762	-6.221987	-2.791027

1322 rows × 21 columns

We want 'raptor_offense', 'raptor_defense', and 'raptor_total' for each player in the draft_drill dataset

In [35]:
```
tat_merge = draft_drill.merge(rap_modern[['player_name', 'raptor_offense', 'raptor_defense', 'raptor_
                                          left_on = ['PLAYER_NAME'],
                                          right_on = ['player_name'], how = 'left')
```

In [40]:
```
stat_merge = stat_merge.dropna()
del stat_merge['PLAYER_ID']
```

In [43]:
```
del stat_merge['FIRST_NAME']
del stat_merge['LAST_NAME']
del stat_merge['player_name']
```

In [44]:
```
stat_merge
```

/_TIME	MODIFIED_LANE_AGILITY_TIME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total
2.88		3.22	12.0	-1.446198	0.200054	-1.246143
2.75		3.28	8.0	-0.428323	-3.226064	-3.654387
3.29		3.18	4.0	-3.576653	0.576872	-2.999781
3.30		3.47	9.0	9.431762	-38.107211	-28.675450
3.08		3.20	13.0	-6.295377	-1.567892	-7.863270
...		...	...	...	...	...
2.87		3.37	15.0	-1.996431	-3.440502	-5.436933
3.04		3.37	9.0	-4.240979	-2.872757	-7.113736
3.11		3.07	5.0	-7.725549	2.024594	-5.700956
3.27		3.33	20.0	-2.890988	2.193436	-0.697552
2.94		3.30	8.0	-1.345147	-0.425551	-1.770698

Saving this dataframe as a csv for further analysis in Tableau

In [45]:
```
stat_merge.to_csv('draft_combine_drill.csv', index = False)
```

In [54]:
```
stat_merge
```

P	LANE_AGILITY_TIME	MODIFIED_LANE_AGILITY_TIME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defen
	11.13	2.88	3.22	12.0	-1.446198	0.200054
	10.27	2.75	3.28	8.0	-0.428323	-3.226064
	10.94	3.29	3.18	4.0	-3.576653	0.576872
	11.60	3.30	3.47	9.0	9.431762	-38.107211
	10.58	3.08	3.20	13.0	-6.295377	-1.567892
	...	...	...	...	...	...
	10.77	2.87	3.37	15.0	-1.996431	-3.440502
	11.16	3.04	3.37	9.0	-4.240979	-2.872757
	10.76	3.11	3.07	5.0	-7.725549	2.024594
	10.83	3.27	3.33	20.0	-2.890988	2.193436
	10.70	2.94	3.30	8.0	-1.345147	-0.425551

In [ ]:
```
```

Moving onto draft anthropometric data

In [27]:
```
draft_anthro_dfs = []
for y in list_of_years:
    draft_anthro_dfs.append(draftcombineplayeranthro.DraftCombinePlayerAnthro(season_year = y).
                            get_data_frames()[0])
```

In [28]:
```
draft_anthro = pd.concat(draft_anthro_dfs).dropna()
```

In [29]:
```
draft_anthro
```

WINGSPAN	WINGSPAN_FT_IN	STANDING_REACH	STANDING_REACH_FT_IN	BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH
82.00	6' 10"	102.0	8' 6"	10.80	8.50	7.50
86.75	7' 2.75"	107.5	8' 11.5"	13.35	8.25	8.75
84.00	7' 0"	104.5	8' 8.5"	6.05	9.00	9.75
88.00	7' 4"	108.5	9' 0.5"	7.45	9.25	10.00
84.75	7' 0.75"	107.0	8' 11"	8.70	8.75	9.00
...	...	...	...	...	...	...
79.00	6' 7.00"	102.0	8' 6.00"	7.20	8.50	9.25
86.25	7' 2.25"	105.5	8' 9.50"	8.90	8.50	9.50
85.00	7' 1.00"	108.5	9' 0.50"	8.00	8.50	10.50
86.75	7' 2.75"	108.5	9' 0.50"	10.80	9.00	10.25
90.50	7' 6.50"	117.0	9' 9.00"	5.40	9.00	9.75

In [47]:
```
draft_anthro.merge(rap_modern[['player_name', 'raptor_offense', 'raptor_defense', 'raptor_total']],
                   left_on = ['PLAYER_NAME'],
                  right_on = ['player_name'], how = 'left')
```

In [49]:
```
stat_merge_anthro = stat_merge_anthro.dropna()
```

In [50]:
```
stat_merge_anthro
```

V_SHOES	HEIGHT_W_SHOES_FT_IN	...	WINGSPAN_FT_IN	STANDING_REACH	STANDING_REACH_FT_IN	BODY_FAT_PCT	HANI
	6' 4.75"	...	6' 10"	102.0	8' 6"	10.80	8.50
	6' 8.5"	...	7' 2.75"	107.5	8' 11.5"	13.35	8.25
	6' 6.25"	...	7' 0"	104.5	8' 8.5"	6.05	9.00
	6' 9.75"	...	7' 0.75"	107.0	8' 11"	8.70	8.75
	6' 9.25"	...	7' 1"	107.0	8' 11"	6.00	9.00
	...	...	...	...	...	...	...
	6'7.25"	...	6'11.0"	103.0	8'7.0"	4.10	8.50
	6'5.0"	...	6'9.75"	103.0	8'7.0"	5.00	8.75
	6'9.75"	...	6'10.25"	106.5	8'10.5"	4.40	9.00
	6'9.0"	...	7'0.75"	107.0	8'11.0"	5.60	9.00
	6'0.25"	...	6'5.25"	94.0	7'10.0"	6.70	8.50

In [51]:
```
del stat_merge_anthro['player_name']
del stat_merge_anthro['FIRST_NAME']
del stat_merge_anthro['LAST_NAME']
```

In [52]:
```
stat_merge_anthro
```

_REACH	STANDING_REACH_FT_IN	BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total
	8' 6"	10.80	8.50	7.50	2.095890	0.518038	2.613928
	8' 11.5"	13.35	8.25	8.75	-3.784998	-2.691065	-6.476062
	8' 8.5"	6.05	9.00	9.75	11.435454	-19.022201	-7.586747
	8' 11"	8.70	8.75	9.00	-4.496620	-3.048730	-7.545350
	8' 11"	6.00	9.00	8.50	-0.656204	1.464091	0.807887
	...	...	...	...	...	...	...
	8'7.0"	4.10	8.50	9.25	-2.581993	-2.677651	-5.259644
	8'7.0"	5.00	8.75	8.50	-1.846780	-1.700197	-3.546978
	8'10.5"	4.40	9.00	8.75	-0.993926	-0.900954	-1.894880
	8'11.0"	5.60	9.00	9.50	-0.209772	-2.803946	-3.013718
	7'10.0"	6.70	8.50	9.00	-1.290293	-8.510070	-9.800363

Saving this dataframe as a csv for further analysis in Tableau

In [53]:
```
stat_merge_anthro.to_csv('draft_combine_anthro.csv', index = False)
```

We can use a K-Means clustering approach to group players that performed similarly in the draft combine drills. Using K-Means clustering with 5 clusters, we can study the commonalities in each group.

In [148]:
```
stat_merge
```

	TEMP_PLAYER_ID	PLAYER_NAME	POSITION	STANDING_VERTICAL_LEAP	MAX_VERTICAL_LEAP	LANE_AGILITY_TIME
0	15001	Justin Anderson	SF-SG	38.0	43.0	11.13
3	15004	Devin Booker	SG	27.5	34.5	10.27
4	15005	Anthony Brown	SF-SG	27.5	34.5	10.94
5	15007	Rakeem Christmas	PF-C	29.0	32.0	11.60
6	15008	Pat Connaughton	SG	37.5	44.0	10.58
...	...	...	...	...	...	...
184	1629678	Admiral Schofield	SF	30.0	34.0	10.77
185	1629621	Marial Shayok	SF-PF	29.5	36.5	11.16
187	1629682	Tremont Waters	PG	30.5	40.5	10.76
189	1629684	Grant Williams	PF	26.0	31.5	10.83
190	1629685	Dylan Windler	SF	29.0	37.5	10.70

145 rows × 14 columns

In [60]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
# Feature selection
X = stat_merge[['STANDING_VERTICAL_LEAP', 'MAX_VERTICAL_LEAP',
            'LANE_AGILITY_TIME', 'MODIFIED_LANE_AGILITY_TIME',
            'THREE_QUARTER_SPRINT', 'BENCH_PRESS']]
s = StandardScaler()
X_s = s.fit_transform(X)
```

In [61]:
```
kmc = KMeans(n_clusters=5, random_state=42)
kmc.fit(X_s)
```

```
KMeans(n_clusters=5, random_state=42)
```

Now we can assign labels to each record

In [63]:
```
stat_merge['K-Means_Label'] = kmc.labels_
```

We also want to create seperate labels based on raptor scores for the player's first year in the NBA

In [113]:
```
del stat_merge['K-Means_Label_raptor']
```

In [114]:
```
X_r = stat_merge[['raptor_offense', 'raptor_defense']]
X_rs = s.fit_transform(X_r)
kmc_r = KMeans(n_clusters=5, random_state=42)
kmc_r.fit(X_rs)
stat_merge['K-Means_Label_raptor'] = kmc_r.labels_
```

In [115]:
```
stat_merge
```

IME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
	3.22	12.0	-1.446198	0.200054	-1.246143	1	2
	3.28	8.0	-0.428323	-3.226064	-3.654387	4	2
	3.18	4.0	-3.576653	0.576872	-2.999781	4	0
	3.47	9.0	9.431762	-38.107211	-28.675450	3	1
	3.20	13.0	-6.295377	-1.567892	-7.863270	1	0
	...	...	...	...	...	...	...
	3.37	15.0	-1.996431	-3.440502	-5.436933	0	0
	3.37	9.0	-4.240979	-2.872757	-7.113736	4	0
	3.07	5.0	-7.725549	2.024594	-5.700956	1	0
	3.33	20.0	-2.890988	2.193436	-0.697552	0	0
	3.30	8.0	-1.345147	-0.425551	-1.770698	4	2

In [ ]:

In [116]:
```
stat_merge
```

IME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
	3.22	12.0	-1.446198	0.200054	-1.246143	1	2
	3.28	8.0	-0.428323	-3.226064	-3.654387	4	2
	3.18	4.0	-3.576653	0.576872	-2.999781	4	0
	3.47	9.0	9.431762	-38.107211	-28.675450	3	1
	3.20	13.0	-6.295377	-1.567892	-7.863270	1	0
	...	...	...	...	...	...	...
	3.37	15.0	-1.996431	-3.440502	-5.436933	0	0
	3.37	9.0	-4.240979	-2.872757	-7.113736	4	0
	3.07	5.0	-7.725549	2.024594	-5.700956	1	0
	3.33	20.0	-2.890988	2.193436	-0.697552	0	0
	3.30	8.0	-1.345147	-0.425551	-1.770698	4	2

In [149]:
```
ped_stat_merge = stat_merge.groupby('K-Means_Label')[['STANDING_VERTICAL_LEAP', 'MAX_VERTICAL_LEAP',
       'LANE_AGILITY_TIME', 'MODIFIED_LANE_AGILITY_TIME',
       'THREE_QUARTER_SPRINT', 'BENCH_PRESS']].mean()
```

In [150]:
```
grouped_stat_merge
```

K-Means_Label	STANDING_VERTICAL_LEAP	MAX_VERTICAL_LEAP	LANE_AGILITY_TIME	MODIFIED_LANE_AGILITY_TIME	THRI
0	29.878788	35.257576	11.223939	3.123030	3.248
1	33.483871	40.645161	10.852581	3.017097	3.121
2	29.631579	35.342105	12.105789	3.396316	3.290
3	25.416667	30.888889	11.932222	3.238333	3.482
4	28.789773	35.136364	11.026591	3.067045	3.252

We can also obtain cluster centers

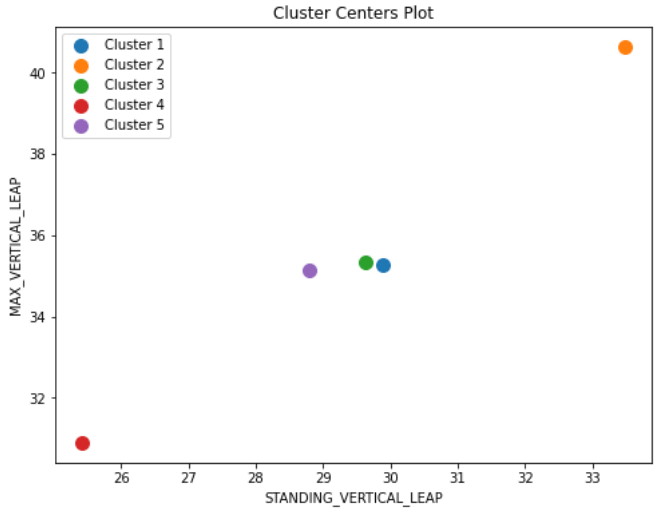
In [119]:
```
cc = s.inverse_transform(kmc.cluster_centers_)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [119], in <cell line: 1>()
----> 1 cc = s.inverse_transform(kmc.cluster_centers_)

File ~/opt/anaconda3/lib/python3.9/site-packages/sklearn/preprocessing/_data.py:1035, in StandardScaler.inverse_transform(self,
X, copy)
   1033 else:
   1034     if self.with_std:
-> 1035         X *= self.scale_
   1036     if self.with_mean:
   1037         X += self.mean_

ValueError: operands could not be broadcast together with shapes (5,6) (2,) (5,6)
```

In [120]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
for i in range(5):
    plt.scatter(cc[i, 0], cc[i, 1], s=100, label=f'Cluster {i+1}')
plt.xlabel('STANDING_VERTICAL_LEAP')
plt.ylabel('MAX_VERTICAL_LEAP')
plt.title('Cluster Centers Plot')
plt.legend()
plt.show()
```



Convert to a csv to move to tableau for visualization

In [121]:
```
stat_merge
```

:AL_LEAP	MAX_VERTICAL_LEAP	LANE_AGILITY_TIME	MODIFIED_LANE_AGILITY_TIME	THREE_QUARTER_SPRINT	BENCH_PRE
43.0	11.13	2.88	3.22	12.0	
34.5	10.27	2.75	3.28	8.0	
34.5	10.94	3.29	3.18	4.0	
32.0	11.60	3.30	3.47	9.0	
44.0	10.58	3.08	3.20	13.0	
...	...	...	...	...	
34.0	10.77	2.87	3.37	15.0	
36.5	11.16	3.04	3.37	9.0	
40.5	10.76	3.11	3.07	5.0	
31.5	10.83	3.27	3.33	20.0	
37.5	10.70	2.94	3.30	8.0	

In [122]:
```
stat_merge.to_csv('draft_combine_drill_labels.csv', index = False)
```

Performing the same clustering approach on the draft anthropometric dataframe

In [91]:
```
stat_merge_anthro
```

_REACH	STANDING_REACH_FT_IN	BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total
8' 6"	10.80	8.50	7.50	2.095890	0.518038	2.613928	
8' 11.5"	13.35	8.25	8.75	-3.784998	-2.691065	-6.476062	
8' 8.5"	6.05	9.00	9.75	11.435454	-19.022201	-7.586747	
8' 11"	8.70	8.75	9.00	-4.496620	-3.048730	-7.545350	
8' 11"	6.00	9.00	8.50	-0.656204	1.464091	0.807887	
...	...	...	...	...	...	...	
8'7.0"	4.10	8.50	9.25	-2.581993	-2.677651	-5.259644	
8'7.0"	5.00	8.75	8.50	-1.846780	-1.700197	-3.546978	
8'10.5"	4.40	9.00	8.75	-0.993926	-0.900954	-1.894880	
8'11.0"	5.60	9.00	9.50	-0.209772	-2.803946	-3.013718	
7'10.0"	6.70	8.50	9.00	-1.290293	-8.510070	-9.800363	

In [94]:
```
stat_merge_anthro
```

_REACH	STANDING_REACH_FT_IN	BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total
	8' 6"	10.80	8.50	7.50	2.095890	0.518038	2.613928
	8' 11.5"	13.35	8.25	8.75	-3.784998	-2.691065	-6.476062
	8' 8.5"	6.05	9.00	9.75	11.435454	-19.022201	-7.586747
	8' 11"	8.70	8.75	9.00	-4.496620	-3.048730	-7.545350
	8' 11"	6.00	9.00	8.50	-0.656204	1.464091	0.807887
	...	...	...	...	...	...	...
	8'7.0"	4.10	8.50	9.25	-2.581993	-2.677651	-5.259644
	8'7.0"	5.00	8.75	8.50	-1.846780	-1.700197	-3.546978
	8'10.5"	4.40	9.00	8.75	-0.993926	-0.900954	-1.894880
	8'11.0"	5.60	9.00	9.50	-0.209772	-2.803946	-3.013718
	7'10.0"	6.70	8.50	9.00	-1.290293	-8.510070	-9.800363

In [96]:
```
X_a = stat_merge_anthro[['HEIGHT_WO_SHOES', 'WEIGHT', 'WINGSPAN', 'STANDING_REACH',
                         'BODY_FAT_PCT', 'HAND_LENGTH', 'HAND_WIDTH']]
X_as = s.fit_transform(X_a)
kmc_a = KMeans(n_clusters=5, random_state=42)
kmc_a.fit(X_as)
stat_merge_anthro['K-Means_Label'] = kmc_a.labels_
```

```
/var/folders/j5/jb5rf2r16tq0vv5rg8_pkqf40000gn/T/ipykernel_29948/4220578284.py:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [97]:
```
stat_merge_anthro
```

DING_REACH_FT_IN	BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total	K-Means_Label
	10.80	8.50	7.50	2.095890	0.518038	2.613928	0
5"	13.35	8.25	8.75	-3.784998	-2.691065	-6.476062	4
'	6.05	9.00	9.75	11.435454	-19.022201	-7.586747	3
	8.70	8.75	9.00	-4.496620	-3.048730	-7.545350	4
	6.00	9.00	8.50	-0.656204	1.464091	0.807887	0
	...	...	...	...	...	...	...
	4.10	8.50	9.25	-2.581993	-2.677651	-5.259644	3
	5.00	8.75	8.50	-1.846780	-1.700197	-3.546978	0
"	4.40	9.00	8.75	-0.993926	-0.900954	-1.894880	0
"	5.60	9.00	9.50	-0.209772	-2.803946	-3.013718	1
"	6.70	8.50	9.00	-1.290293	-8.510070	-9.800363	2

In [108]:
```
del stat_merge_anthro['K-Means_Label_raptor']
```

In [109]:
```
X_ar = stat_merge_anthro[['raptor_offense', 'raptor_defense']]
X_ars = s.fit_transform(X_ar)
kmc_ar = KMeans(n_clusters=5, random_state=42)
kmc_ar.fit(X_ars)
stat_merge_anthro['K-Means_Label_raptor'] = kmc_ar.labels_
```

```
/var/folders/j5/jb5rf2r16tq0vv5rg8_pkqf40000gn/T/ipykernel_29948/2134130367.py:5: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [110]:
```
stat_merge_anthro
```

R_ID	PLAYER_NAME	POSITION	HEIGHT_WO_SHOES	HEIGHT_WO_SHOES_FT_IN	HEIGHT_W_SHOES	HEIGHT_W_SHOES_FT_IN
	Jordan Adams	SG	75.50	6' 3.5"	76.75	6' 4.75"
	Kyle Anderson	SF	79.50	6' 7.5"	80.50	6' 8.5"
	Thanasis Antetokounmpo	SF	77.25	6' 5.25"	78.25	6' 6.25"
	Cameron Bairstow	PF	80.75	6' 8.75"	81.75	6' 9.75"
	Khem Birch	PF	79.50	6' 7.5"	81.25	6' 9.25"
	...	...	...	...	...	...
	Joe Wieskamp	SF	77.75	6'5.75"	79.25	6'7.25"
	Aaron Wiggins	SG-SF	76.50	6'4.5"	77.00	6'5.0"
	Ziaire Williams	SF	80.25	6'8.25"	81.75	6'9.75"
	Moses Wright	PF	79.75	6'7.75"	81.00	6'9.0"
	McKinley Wright IV	PG	71.25	5'11.25"	72.25	6'0.25"

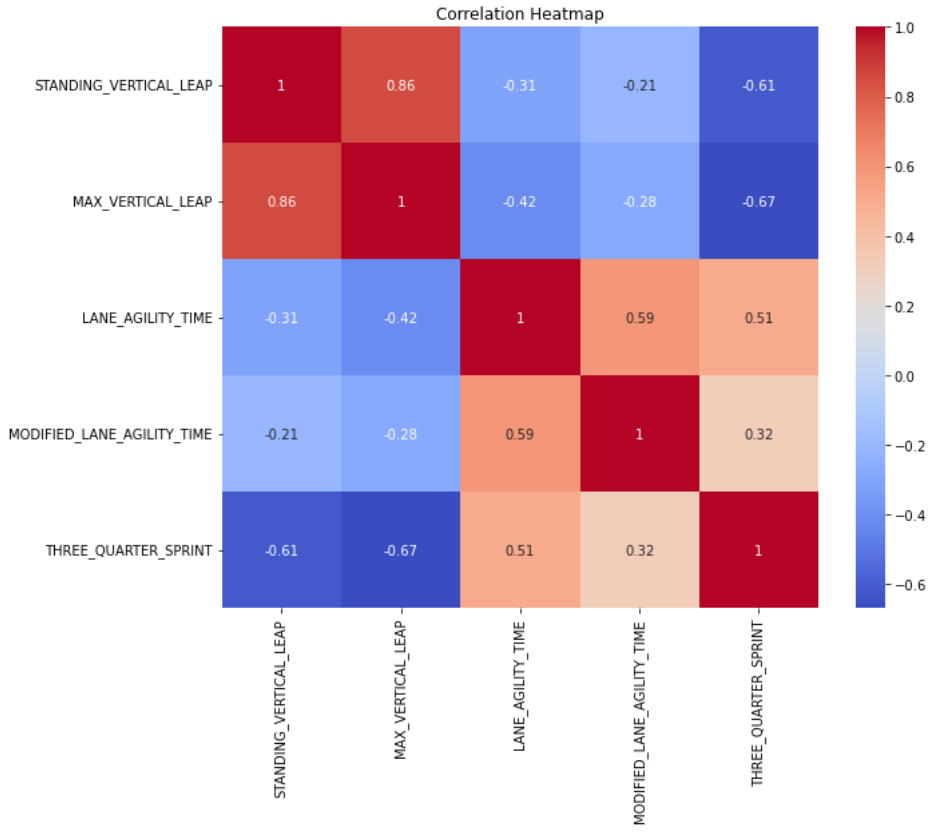
In [111]:
```
stat_merge_anthro.to_csv('draft_combine_anthro_labels.csv', index = False)
```

Correlation Heatmap

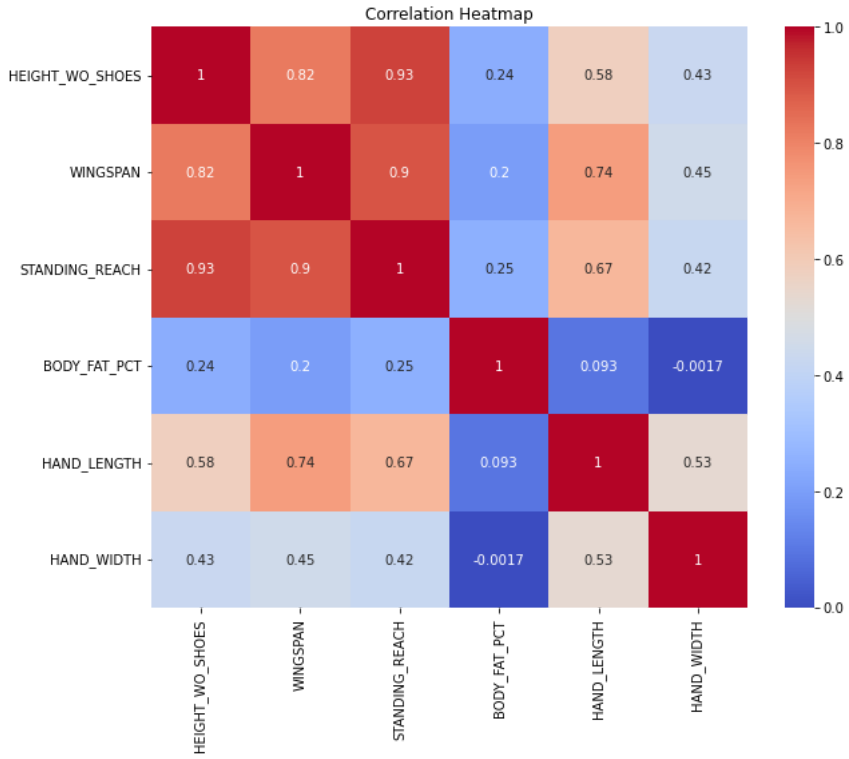
In [123]:

IME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
	3.22	12.0	-1.446198	0.200054	-1.246143	1	2
	3.28	8.0	-0.428323	-3.226064	-3.654387	4	2
	3.18	4.0	-3.576653	0.576872	-2.999781	4	0
	3.47	9.0	9.431762	-38.107211	-28.675450	3	1
	3.20	13.0	-6.295377	-1.567892	-7.863270	1	0
	...	...	...	...	...	...	...
	3.37	15.0	-1.996431	-3.440502	-5.436933	0	0
	3.37	9.0	-4.240979	-2.872757	-7.113736	4	0
	3.07	5.0	-7.725549	2.024594	-5.700956	1	0
	3.33	20.0	-2.890988	2.193436	-0.697552	0	0
	3.30	8.0	-1.345147	-0.425551	-1.770698	4	2

In [128]:
```
import seaborn as sns
plot_data = stat_merge[['STANDING_VERTICAL_LEAP', 'MAX_VERTICAL_LEAP', 'LANE_AGILITY_TIME',
            'MODIFIED_LANE_AGILITY_TIME', 'THREE_QUARTER_SPRINT', 'BENCH_PRESS']]
correlation_matrix = plot_data.corr()
labels = stat_merge['K-Means_Label']
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



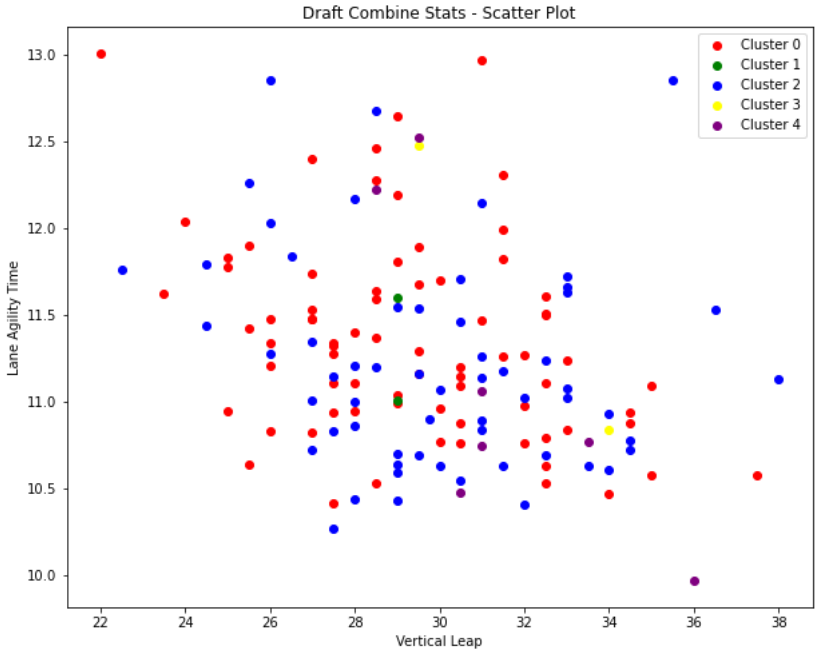
In [130]:
```
plot_data = stat_merge_anthro[['HEIGHT_WO_SHOES', 'WEIGHT', 'WINGSPAN', 'STANDING_REACH',
                               'BODY_FAT_PCT', 'HAND_LENGTH', 'HAND_WIDTH']]
correlation_matrix = plot_data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [151]:
```
stat_merge
```

TIME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
3.22		12.0	-1.446198	0.200054	-1.246143	1	2
3.28		8.0	-0.428323	-3.226064	-3.654387	4	2
3.18		4.0	-3.576653	0.576872	-2.999781	4	0
3.47		9.0	9.431762	-38.107211	-28.675450	3	1
3.20		13.0	-6.295377	-1.567892	-7.863270	1	0
...		...	...	...	...	...	...
3.37		15.0	-1.996431	-3.440502	-5.436933	0	0
3.37		9.0	-4.240979	-2.872757	-7.113736	4	0
3.07		5.0	-7.725549	2.024594	-5.700956	1	0
3.33		20.0	-2.890988	2.193436	-0.697552	0	0
3.30		8.0	-1.345147	-0.425551	-1.770698	4	2

In [152]:
```
plot_data = stat_merge[['STANDING_VERTICAL_LEAP', 'MAX_VERTICAL_LEAP', 'LANE_AGILITY_TIME',
            'MODIFIED_LANE_AGILITY_TIME', 'THREE_QUARTER_SPRINT', 'BENCH_PRESS']]
c_labels = stat_merge['K-Means_Label_raptor']
c_mapping = {
    0: 'red',
    1: 'green',
    2: 'blue',
    3: 'yellow',
    4: 'purple'
}
plt.figure(figsize=(10, 8))
for label, color in c_mapping.items():
    data = plot_data[c_labels == label]
    plt.scatter(data['STANDING_VERTICAL_LEAP'], data['LANE_AGILITY_TIME'], color = color,
                label = f'Cluster {label}')
plt.xlabel('Vertical Leap')
plt.ylabel('Lane Agility Time')
plt.title('Draft Combine Stats - Scatter Plot')
plt.legend()
plt.show()
```



In [143]:
```
stat_merge_anthro
```

BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
10.80	8.50	7.50	2.095890	0.518038	2.613928	0	0
13.35	8.25	8.75	-3.784998	-2.691065	-6.476062	4	2
6.05	9.00	9.75	11.435454	-19.022201	-7.586747	3	1
8.70	8.75	9.00	-4.496620	-3.048730	-7.545350	4	2
6.00	9.00	8.50	-0.656204	1.464091	0.807887	0	0
...	...	...	...	...	...	...	...
4.10	8.50	9.25	-2.581993	-2.677651	-5.259644	3	2
5.00	8.75	8.50	-1.846780	-1.700197	-3.546978	0	0
4.40	9.00	8.75	-0.993926	-0.900954	-1.894880	0	0
5.60	9.00	9.50	-0.209772	-2.803946	-3.013718	1	0
6.70	8.50	9.00	-1.290293	-8.510070	-9.800363	2	2

In [144]:
```
df = stat_merge_anthro.groupby('K-Means_Label')[['HEIGHT_WO_SHOES',
                'WINGSPAN', 'STANDING_REACH',
                'BODY_FAT_PCT', 'HAND_LENGTH', 'HAND_WIDTH']].mean()
```

In [145]:
```
df.to_csv('avg_anthro_kcluster.csv')
```

In [146]:
```
df = stat_merge_anthro.groupby('K-Means_Label_raptor')[['HEIGHT_WO_SHOES',
                'WINGSPAN', 'STANDING_REACH',
                'BODY_FAT_PCT', 'HAND_LENGTH', 'HAND_WIDTH']].mean()
```

In [147]:
```
df.to_csv('avg_anthro_kclusterraptor.csv')
```

In [154]:
```
stat_merge
```

IME	THREE_QUARTER_SPRINT	BENCH_PRESS	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
3.22		12.0	-1.446198	0.200054	-1.246143	1	2
3.28		8.0	-0.428323	-3.226064	-3.654387	4	2
3.18		4.0	-3.576653	0.576872	-2.999781	4	0
3.47		9.0	9.431762	-38.107211	-28.675450	3	1
3.20		13.0	-6.295377	-1.567892	-7.863270	1	0
...		...	...	...	...	...	...
3.37		15.0	-1.996431	-3.440502	-5.436933	0	0
3.37		9.0	-4.240979	-2.872757	-7.113736	4	0
3.07		5.0	-7.725549	2.024594	-5.700956	1	0
3.33		20.0	-2.890988	2.193436	-0.697552	0	0
3.30		8.0	-1.345147	-0.425551	-1.770698	4	2

In [205]:
```
tat_merge[['STANDING_VERTICAL_LEAP','MAX_VERTICAL_LEAP','LANE_AGILITY_TIME',
           'MODIFIED_LANE_AGILITY_TIME', 'THREE_QUARTER_SPRINT', 'BENCH_PRESS', 'K-Means_Label_raptor
            'K-Means_Label']]
 the parallel coordinates plot
```

In [206]:
```
new_names = {
    'STANDING_VERTICAL_LEAP':'Standing Vertical Leap',
    'MAX_VERTICAL_LEAP':'Max Vertical Leap',
    'LANE_AGILITY_TIME':'Lane Agility Time',
    'MODIFIED_LANE_AGILITY_TIME':'Modified LAT',
    'THREE_QUARTER_SPRINT':'3/4 Sprint',
    'BENCH_PRESS':'Bench Press'
}
temp = temp.rename(columns = new_names)
```

In [207]:
```
temp
```

	Standing Vertical Leap	Max Vertical Leap	Lane Agility Time	Modified LAT	3/4 Sprint	Bench Press	K-Means_Label_raptor	K-Means_Label
0	38.0	43.0	11.13	2.88	3.22	12.0	2	1
3	27.5	34.5	10.27	2.75	3.28	8.0	2	4
4	27.5	34.5	10.94	3.29	3.18	4.0	0	4
5	29.0	32.0	11.60	3.30	3.47	9.0	1	3
6	37.5	44.0	10.58	3.08	3.20	13.0	0	1
...	...	...	...	...	...	...	...	...
184	30.0	34.0	10.77	2.87	3.37	15.0	0	0
185	29.5	36.5	11.16	3.04	3.37	9.0	0	4
187	30.5	40.5	10.76	3.11	3.07	5.0	0	1
189	26.0	31.5	10.83	3.27	3.33	20.0	0	0
190	29.0	37.5	10.70	2.94	3.30	8.0	2	4

145 rows × 8 columns

In [208]:
```
dimensions = ['Standing Vertical Leap', 'Max Vertical Leap', 'Lane Agility Time',
              'Modified LAT', '3/4 Sprint', 'Bench Press']
temp_norm = (temp[dimensions] - temp[dimensions].min()) / (temp[dimensions].max() - temp[dimensions]
```

In [209]:
```
temp_norm['K-Means_Label_raptor'] = temp['K-Means_Label_raptor']
temp_norm['K-Means_Label'] = temp['K-Means_Label']
```

In [210]:
```
temp_norm
```

	Standing Vertical Leap	Max Vertical Leap	Lane Agility Time	Modified LAT	3/4 Sprint	Bench Press	K-Means_Label_raptor	K-Means_Label
0	1.0	0.923077	0.381579	0.169811	0.272727	0.6	2	1
3	0.34375	0.487179	0.098684	0.04717	0.350649	0.4	2	4
4	0.34375	0.487179	0.319079	0.556604	0.220779	0.2	0	4
5	0.4375	0.358974	0.536184	0.566038	0.597403	0.45	1	3
6	0.96875	0.974359	0.200658	0.358491	0.246753	0.65	0	1
...	...	...	...	...	...	...	...	...
184	0.5	0.461538	0.263158	0.160377	0.467532	0.75	0	0
185	0.46875	0.589744	0.391447	0.320755	0.467532	0.45	0	4
187	0.53125	0.794872	0.259868	0.386792	0.077922	0.25	0	1
189	0.25	0.333333	0.282895	0.537736	0.415584	1.0	0	0
190	0.4375	0.641026	0.240132	0.226415	0.376623	0.4	2	4

145 rows × 8 columns

In [ ]:
```
del temp_norm
```

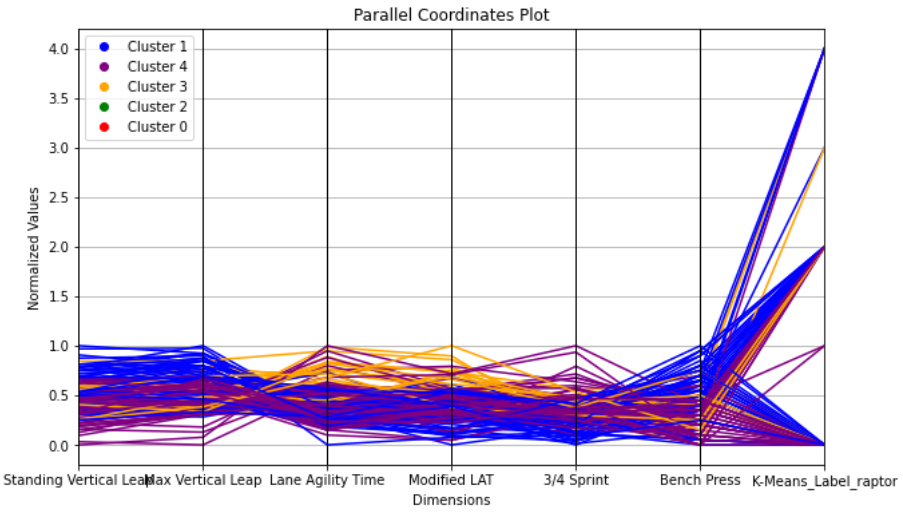
In [211]:
```
ort pandas as pd
ort matplotlib.pyplot as plt

or_map = {0: 'red', 1: 'blue', 2: 'green', 3: 'orange', 4: 'purple'}
figure(figsize=(10, 6))
lotting.parallel_coordinates(temp_norm, 'K-Means_Label', color=[color_map[i] for i in temp_norm['K-M

id a legend
end_labels = [plt.Line2D([], [], color=color_map[i], marker='o', linestyle='', label='Cluster {}'.for
legend(handles=legend_labels)

istomize the plot appearance
xlabel('Dimensions')
ylabel('Normalized Values')
title('Parallel Coordinates Plot')

ow the plot
show()
```



In [167]:
```
stat_merge_anthro
```

BODY_FAT_PCT	HAND_LENGTH	HAND_WIDTH	raptor_offense	raptor_defense	raptor_total	K-Means_Label	K-Means_Label_raptor
10.80	8.50	7.50	2.095890	0.518038	2.613928	0	0
13.35	8.25	8.75	-3.784998	-2.691065	-6.476062	4	2
6.05	9.00	9.75	11.435454	-19.022201	-7.586747	3	1
8.70	8.75	9.00	-4.496620	-3.048730	-7.545350	4	2
6.00	9.00	8.50	-0.656204	1.464091	0.807887	0	0
...	...	...	...	...	...	...	...
4.10	8.50	9.25	-2.581993	-2.677651	-5.259644	3	2
5.00	8.75	8.50	-1.846780	-1.700197	-3.546978	0	0
4.40	9.00	8.75	-0.993926	-0.900954	-1.894880	0	0
5.60	9.00	9.50	-0.209772	-2.803946	-3.013718	1	0
6.70	8.50	9.00	-1.290293	-8.510070	-9.800363	2	2

In [194]:
```
stat_merge_anthro[['HEIGHT_WO_SHOES','WEIGHT','WINGSPAN',
           'STANDING_REACH', 'BODY_FAT_PCT', 'HAND_LENGTH', 'HAND_WIDTH', 'K-Means_Label_raptor', 'K-
es = {
IGHT_WO_SHOES':'Height',
IGHT':'Weight',
NGSPAN':'Wingspan',
ANDING_REACH':'Standing Reach',
DY_FAT_PCT':'Body Fat %',
ND_LENGTH':'Hand Length',
ND_WIDTH':'Hand Width'

temp.rename(columns = new_names)
```

In [195]:
```
dimensions = ['Height','Weight','Wingspan',
                    'Standing Reach', 'Body Fat %', 'Hand Length', 'Hand Width']
```

In [196]:
```
temp_norm = pd.DataFrame()
```

In [197]:
```
temp[dimensions]
```

	Height	Weight	Wingspan	Standing Reach	Body Fat %	Hand Length	Hand Width
0	75.50	208.8	82.00	102.0	10.80	8.50	7.50
1	79.50	230.1	86.75	107.5	13.35	8.25	8.75
2	77.25	205.2	84.00	104.5	6.05	9.00	9.75
4	80.75	252.4	84.75	107.0	8.70	8.75	9.00
5	79.50	208.8	85.00	107.0	6.00	9.00	8.50
...	...	...	...	...	...	...	...
472	77.75	204.80	83.00	103.0	4.10	8.50	9.25
473	76.50	190.00	81.75	103.0	5.00	8.75	8.50
474	80.25	188.40	82.25	106.5	4.40	9.00	8.75
475	79.75	225.80	84.75	107.0	5.60	9.00	9.50
476	71.25	192.20	77.25	94.0	6.70	8.50	9.00

369 rows × 7 columns

In [198]:
```
temp['Weight'] = temp['Weight'].astype(float)
```

In [199]:
```
temp_norm = (temp[dimensions] - temp[dimensions].min()) / (temp[dimensions].max() - temp[dimensions
```

In [202]:
```
temp_norm['K-Means_Label'] = temp['K-Means_Label']
temp_norm['K-Means_Label_raptor'] = temp['K-Means_Label_raptor']
```

In [203]:
```
temp_norm
```

	Height	Weight	Wingspan	Standing Reach	Body Fat %	Hand Length	Hand Width	K-Means_Label	K-Means_Label_raptor
0	0.345238	0.426934	0.375000	0.397059	0.615385	0.333333	0.000000	0	0
1	0.535714	0.579513	0.557692	0.558824	0.811538	0.250000	0.294118	4	2
2	0.428571	0.401146	0.451923	0.470588	0.250000	0.500000	0.529412	3	1
4	0.595238	0.739255	0.480769	0.544118	0.453846	0.416667	0.352941	4	2
5	0.535714	0.426934	0.490385	0.544118	0.246154	0.500000	0.235294	0	0
...	...	...	...	...	...	...	...	...	...
472	0.452381	0.398281	0.413462	0.426471	0.100000	0.333333	0.411765	3	2
473	0.392857	0.292264	0.365385	0.426471	0.169231	0.416667	0.235294	0	0
474	0.571429	0.280802	0.384615	0.529412	0.123077	0.500000	0.294118	0	0
475	0.547619	0.548711	0.480769	0.544118	0.215385	0.500000	0.470588	1	0
476	0.142857	0.308023	0.192308	0.161765	0.300000	0.333333	0.352941	2	2

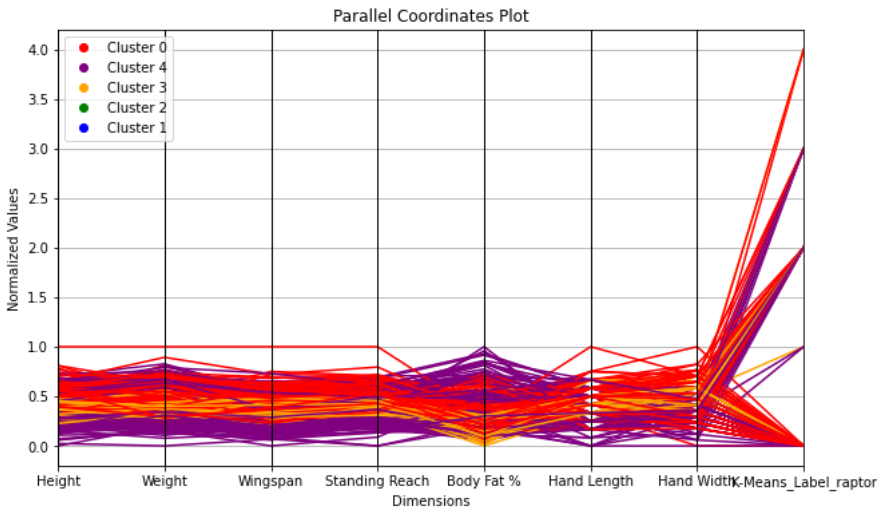
369 rows × 9 columns

In [204]:
```
{0: 'red', 1: 'blue', 2: 'green', 3: 'orange', 4: 'purple'}
igsize=(10, 6))
parallel_coordinates(temp_norm, 'K-Means_Label', color=[color_map[i] for i in temp_norm['K-Means_Lab

nd
s = [plt.Line2D([], [], color=color_map[i], marker='o', linestyle='', label='Cluster {}'.format(i)) 
andles=legend_labels)

the plot appearance
Dimensions')
Normalized Values')
arallel Coordinates Plot')

lot
```



In [ ]:

REFERENCES

FDEZ, E. M. (2022, October 17). *How much money will the NBA champion earn?*. MARCA. https://www.marca.com/en/basketball/nba/2022/10/17/634d10f122601d33288b45ae.html

Gough, C. (2023, January 9). *Total NBA League Revenue 2022*. Statista. https://www.statista.com/statistics/193467/total-league-revenue-of-the-nba-since-2005/#:~:text=Total%20revenue%20of%20the%20National%20Basketball%20Association%202001%2D2022&text=The%20total%20league%20revenue%20of,over%2010%20billion%20U.S.%20dollars.

Leadership. NBA Careers. (n.d.). https://careers.nba.com/leadership/

Nath, T. I. (2022, July 13). *The NBA's business model*. Investopedia. https://www.investopedia.com/articles/investing/070715/nbas-business-model.asp

Ozanian, M. (2022, October 29). *NBA Team values 2022: For the first time in two decades, the top spot goes to a franchise that's not the Knicks or Lakers*. Forbes. https://www.forbes.com/sites/mikeozanian/2022/10/27/nba-team-values-2022-for-the-first-time-in-two-decades-the-top-spot-goes-to-a-franchise-thats-not-the-knicks-or-lakers/?sh=28d745e51cce

Silver, N. (2019, October 10). *How our raptor metric works*. FiveThirtyEight. https://fivethirtyeight.com/features/how-our-raptor-metric-works/

Bradlow, E. (2023). *How analytics can boost competitiveness in sports*. Knowledge at Wharton. https://knowledge.wharton.upenn.edu/article/how-analytics-can-boost-competitiveness-in-sports/

Fitzpatrick, W. (2022, December 2). *A crash course in basketball analytics*. Samford University. https://www.samford.edu/sports-analytics/fans/2020/A-Crash-Course-in-Basketball-Analytics

Khan, E. (2017, October 3). *Advanced NBA stats for dummies: How to understand the new hoops math*. Bleacher Report. https://bleacherreport.com/articles/1813902-advanced-nba-stats-for-dummies-how-to-understand-the-new-hoops-math

Medvedovsky, K., & Patton, A. (n.d.). *What is Darko?*. DARKO Exploration. https://apanalytics.shinyapps.io/DARKO/

Silver, N. (2019, October 10). *How our raptor metric works*. FiveThirtyEight. https://fivethirtyeight.com/features/how-our-raptor-metric-works/

Weiner, J. (2022, July 15). *Predicting the outcome of NBA games with Machine Learning*. Medium. https://towardsdatascience.com/predicting-the-outcome-of-nba-games-with-machine-learning-a810bb768f20