UNIVERSITY OF KWAZULU-NATAL

SCHOOL OF MATHS, STATISTICS AND COMPUTER SCIENCE

COMP721: MACHINE LEARNING

MINI-PROJECT

THE NATIONAL BASKETBALL ASSOCIATION (NBA) PREDICTION

Authors:

Thariq Singh

219063421@stu.ukzn.ac.za

Saarisha Govender

221009853@stu.ukzn.ac.za

Callyn Blayne Barath

220010761@stu.ukzn.ac.za

Lerisha Moodley

220036955@stu.ukzn.ac.za

Dhiya Dharampal

221033815@stu.ukzn.ac.za

Table of Contents

1 Introduction

The following link can be used to access the GitHub repository for the project files:

https://github.com/thariqsingh01/Comp721-Project

This project delves into the use of Machine Learning for analysing and predicting outcomes within the National Basketball Association (NBA). Leveraging a dataset from numerous seasons, our study focuses on two primary objectives: identifying outstanding players through outlier detection and predicting the outcomes of NBA games between teams. Using an extensive dataset that includes statistics for both players and coaches, our analysis spans regular season performances, playoff appearances, and coaching records, as well as specific indicators such as points per game, rebounds, assists, and other player metrics.

The project begins with rigorous preprocessing, addressing missing values, data normalization, and dimensionality reduction. We employ techniques like k-nearest neighbors imputation for handling missing data, MinMaxScaler for scaling, and Principal Component Analysis (PCA) to reduce dimensionality while retaining essential information. This preprocessing ensures that our data is in a state suitable for further machine learning analyses, particularly clustering and classification.

To identify standout players, we implement various outlier detection methods, including clustering-based, distance-based, and LOESS (Locally Estimated Scatterplot Smoothing). By analysing performance metrics across the season, we isolate players whose statistics significantly deviate from the norm. This approach allows us to capture a range of player attributes and identify players who contribute uniquely to their teams, whether through scoring, rebounding, or defensive skills.

For game outcome prediction, we use models such as Random Forest, Support Vector Machines (SVMs), Naïve Bayes, and ensemble stacking methods. These models help predict the likelihood of a team winning a game based on a range of team statistics. We also explore different kernel functions in SVM to capture non-linear relationships in the data. Five-fold cross-validation is employed to evaluate model accuracy and reliability, ensuring that the models perform consistently across different data subsets. In particular, ensemble models like Random Forest and stacking algorithms showcase the robustness of machine learning in handling complex, high-dimensional sports data.

By implementing and comparing these diverse techniques, our project provides insights into the unique patterns that define player excellence and team success within the NBA. This project underscores the potential of machine learning for sports analytics, enhancing our ability to analyse complex datasets and gain meaningful insights from the statistical indicators that drive performance in professional basketball.

2 Methods and Techniques

2.1 Preprocessing

2.1.1 Outlier Detection of Players

In the first part of the project, the initial preprocessing involves loading several datasets from different files. Each dataset illustrates various elements of player, coach, team, and draft details in text form. The datasets are loaded into individual pandas DataFrames, each using comma separators. There are factors to consider for various encodings and the management of possibly incorrect lines, which are bypassed to guarantee seamless loading. The assortment of data consists of both numerical and non-numerical attributes, along with absent values in certain datasets. Therefore, preprocessing includes multiple stages to guarantee that the data is tidy, uniform, and in a form appropriate for additional analysis.

An essential aspect of preprocessing is managing absent values in numerical data. To achieve this, the KNNImputer from sklearn is employed to fill in missing values, with each missing value being completed based on the closest five neighbouring data points. This imputation is only completed on 3 files (‘*player_playoffs.txt’*, ‘*player_playoffs_career.txt’* and *‘player_regular_season_career.txt’*) and is carried out solely on the columns containing numeric data since they are appropriate for the KNN algorithm. The non-numeric columns are kept separately to preserve their original condition. Post-imputation, the numerical data is merged with the non-numeric columns, maintaining the organization of each dataset while successfully tackling missing values.

Imputation is a method for handling missing data by substituting missing values with reasonable estimates derived from available data, ensuring models stay accurate without eliminating records. In particular, k-nearest neighbour (k-NN) imputation addresses missing values by utilizing the values from nearby (similar) data points. This approach enhances data integrity and is widely employed in machine learning to reduce the adverse effects of absent data.

For continuous attributes, the missing value for a feature xi is approximated by calculating the average of the values of its k closest neighbours:

$$\hat{x}_i = \frac{1}{k}\sum_{j=1}^{k} x_j \quad (1)$$

where $\hat{x}_i$ is the imputed value, k is the number of neighbors, and x_j represents the values of these neighbours. For categorical features, the imputed value is determined by the most common category among the neighbours:

$$\hat{x}_i = mode(x_1, x_2, \dots, x_k) \quad (2)$$

This k-NN imputation method is adaptable, efficiently managing both continuous and categorical data. Nonetheless, it can be resource-demanding, particularly with extensive datasets, since it necessitates computing distances to identify the closest neighbours for every imputed value.[1]

Additional preprocessing involves combining player data with extra demographic information. This entails merging the firstname and lastname columns to form a Full_Name

column and eliminating the original columns. In preparation for machine learning, a selection of columns (mainly non-numeric identifiers and text fields) are separated, and only the numeric columns are normalized with MinMaxScaler to guarantee that the data is consistently scaled. This stage is crucial for clustering and dimensionality reduction, since variations in data scale can significantly influence clustering results. Ultimately, PCA is utilized to diminish the dataset's dimensions to two main components, facilitating a streamlined two-dimensional depiction that can be employed in subsequent analyses, like clustering.

MinMaxScaler is a normalization method utilized to modify the range of each attribute in a dataset, generally transforming values to a span, such as [0,1]. By normalizing features in this manner, it guarantees that features with larger numerical ranges do not unduly affect machine learning models, especially those based on distance computations. The mathematical formula for MinMax scaling is:

$$\acute{x}_i = \frac{x_i - min(X)}{max(X) - min\,(X)} \tag{3}$$

Where x_i denotes a feature value, min(X) and max(X) represent the minimum and maximum feature values, and $\acute{x}_i$ indicates the scaled feature value. This conversion translates values into a range from 0 to 1, normalizing feature scales while preserving their relative distribution. MinMax scaling can also be generalized to other intervals [a,b] by applying the formula:

$$\acute{x}_i = a + \frac{(x_i - \min\,(x))(b-a)}{\max(X) - \min\,(X)} \tag{4}$$

This adaptable method enables the data to be adjusted to any preferred range [2]. MinMaxScaler maintains the data's structure by ensuring that each feature contributes equally, which is especially beneficial for datasets with different feature ranges.

Principal Component Analysis (PCA) is a statistical method employed to decrease the dimensionality of datasets containing numerous correlated attributes, converting them into a reduced collection of uncorrelated variables known as principal components. Every principal component (PC) is created from a linear combination of the original features, intended to capture the highest variance in the data. This method enables the summarization of high-dimensional data into fewer dimensions, decreasing computational complexity and potentially enhancing model accuracy by addressing concerns such as collinearity and overfitting. Mathematically, if a dataset is represented by a matrix X with N samples and p features, PCA transforms it into a new set of variables T, given by:

$$T = XP \tag{5}$$

where P is a matrix of loadings (eigenvectors) derived from the original features [3]. This transformation ensures that the first few principal components retain most of the dataset’s variance, allowing the remaining components to be discarded with minimal information loss.

2.1.2 Predicting Game Outcomes

(a) Feature Engineering

In this part of the project, we loaded datasets from text files containing team and coach statistics from regular basketball seasons into pandas DataFrames. Several preprocessing steps were then performed to prepare the data for further analysis using various Machine Learning techniques.

To predict the winner of a game between two teams, it is vital to have features that accurately represent the performance of each team. To achieve this, we created a new dataset of features that were engineered based on the features available in the loaded datasets, *team_season* and *coaches_season*.

The new DataFrame describing team season statistics retained key attributes from *team_season,* namely; the team's name and league, the year that a particular season was played in, and the team's estimated pace of play. The pace indicates the number of possessions per 48 minutes played.

In addition to this, several new features were derived from the statistics describing offensive and defensive metrics within the *team_season* dataset. These include metrics such as the offensive points scored per game, the offensive field goal and free throw success rates, the offensive rebound and turnover rates, and the rate of successful offensive three-point attempts. These are computed as follows:

$$Offensive\ FG\ Rate\ =\ \frac{FG\ Made}{FG\ Attempted} \tag{6}$$

$$Offensive\ FT\ Rate\ =\ \frac{FT\ Made}{FT\ Attempted} \tag{7}$$

$$Offensive\ 3P\ Rate\ =\ \frac{3P\ Made}{3P\ Attempted} \tag{8}$$

where FG, FT and 3P are the number of field goals, free throws and 3-point shots respectively. The offensive rebound rate indicates a team's ability to grab offensive rebounds, and the turnover rate is a measure of how often the team loses possession of the ball due to turnovers.

Corresponding features were similarly derived to describe the defensive metrics including the number of defensive points allowed per game, allowed defensive field goals and three-point shots, as well as steal and block rates.

Other calculated metrics that describe a team's overall performance include their win percentage, loss percentage, win-to-loss ratio and points differential per game. The points differential feature is calculated as:

$$Points\ Differential\ =\ Offensive\ Points\ per\ Game\ -\ Defensive\ Points\ per\ Game \tag{9}$$

We calculated a team's possession of the ball using the formula [4]:

$$Possessions = FGA - OREB + TO + 0.4 \times FTA \tag{10}$$

where FGA and FTA are the number of field goal and free throw attempts, respectively. OREB is the number of offensive rebounds, and TO is the number of turnovers. We also derived the offensive and defensive points per 100 possessions to measure the team's efficiency in scoring and preventing points when in possession of the ball. These metrics are expressed as:

$$Offensive\ Points\ per\ 100\ Possessions\ =\ \frac{Offensive\ Points\ Scored}{Possessions} \times 100 \tag{11}$$

$$Defensive\ Points\ per\ 100\ Possessions\ =\ \frac{Defensive\ Points\ Allowed}{Possessions} \times 100 \tag{12}$$

After creating the new dataset with engineered features, we split the data based on the NBA and ABA basketball leagues. This is performed since two teams from different leagues would generally not compete in a game. This splitting allowed us to include coaching statistics for the NBA league only, as no records were available for the ABA league. Using the *coaches_season* dataset, we derived additional features to highlight a coach's influence on a team's overall performance. These features include a coach's win and loss rates, computed as:

$$Win\ Rate\ =\ \frac{Number\ of\ Wins}{Total\ Number\ of\ Games} \tag{13}$$

$$Loss\ Rate\ =\ \frac{Number\ of\ Losses}{Total\ Number\ of\ Games} \tag{14}$$

We also computed the years of coaching experience, playoff participation and playoff win percentage.

A new DataFrame was then created to represent each possible combination of teams in each season for the NBA league. This DataFrame combined the statistics for each team and listed a winner. The team with the higher number of offensive points per game was selected as the winner, and represented by the value 1 for team A and 0 for team B. A similar DataFrame was created for the ABA league.

To merge the NBA and ABA datasets, placeholder columns filled with NaN values were added to the ABA dataset to match the coaching statistics in the NBA dataset. These datasets were then concatenated, and the missing coaching statistics from the ABA dataset were then imputed using the mean from the NBA coach's data. This was achieved by using the *SimpleImputer* from the *scikit-learn* library.

Lastly, we added an additional feature to the dataset which indicated the ABA league's merger with the NBA league in 1976 [5]. This feature took the value 0 for games before the merger and 1 for post-merger games. Merging the NBA and ABA datasets provides historical data for teams from the ABA that joined the NBA league in the merger. This information is significant because it could indicate differences in playing styles and performances resulting from the merger of the two leagues.

(b) Feature Importance Analysis

Once the relevant features were engineered, we performed a feature importance analysis to identify how much each feature contributed to the prediction of the game outcomes.

We first scaled the data using the *StandardScaler* function from the *scikit-learn* library. This process ensures that each feature in the dataset has a mean of 0 and a standard deviation of 1. This process is mathematically expressed as:

$$X_{scaled} = \frac{X-\mu}{\sigma} \tag{15}$$

where X is the original feature vector, μ and σ are the mean and standard deviation of the feature, respectively.

Next, we use the Random Forest classifier to calculate the feature importance scores. This allows us to evaluate the relative impact of each feature on the predictive power of the model. The importance scores are visualised in a horizontal bar graph in Figure 1., where each feature is represented by its index in the DataFrame.

The red dashed line indicates a cutoff threshold, set at 0.001. Features with importance scores below this threshold were removed from the dataset as they were considered to have minimal influence on the model's predictions. A low threshold was chosen to ensure that sufficient features remained for dimensionality reduction using Principal Component Analysis (PCA). This will further refine the feature set and enhance model performance.

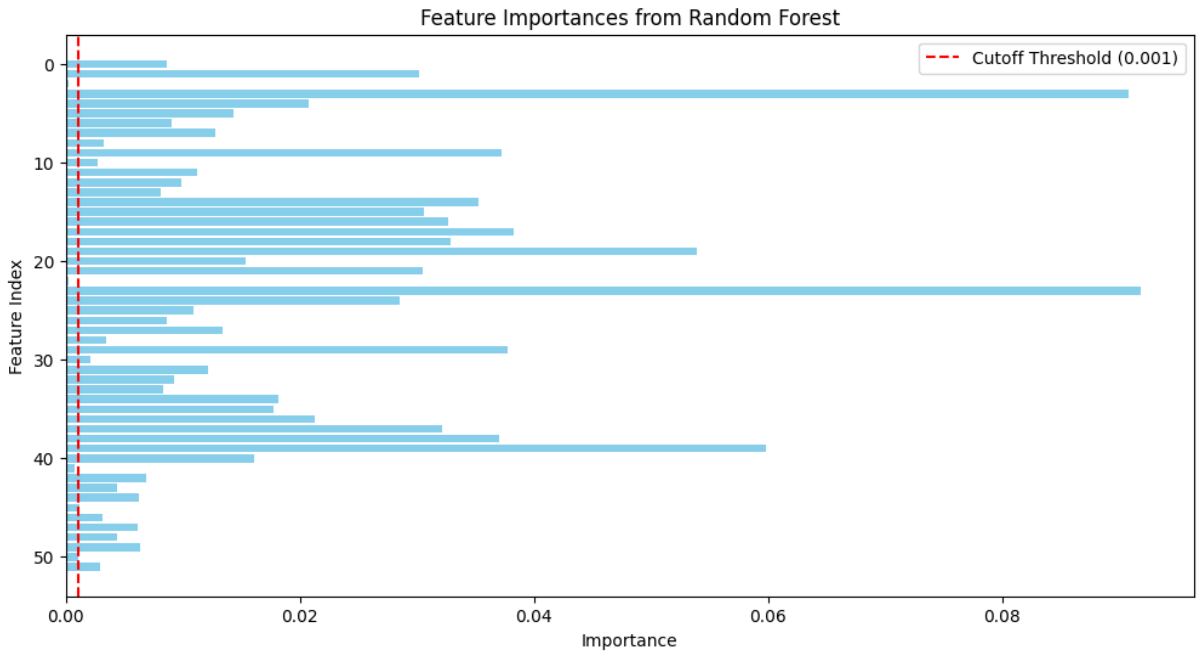

Figure 1. Horizontal bar graph depicting the importance scores for each engineered feature (represented by its index in the DataFrame) obtained using the Random Forest classifier. The red dashed line marks the cutoff threshold value. All feature values with importance scores below this value will be removed before model training.

2.2 Data Visualisation

2.2.1 Outlier Detection of Players

In the realm of professional basketball, understanding player performance is crucial for determining outstanding players within the NBA. Outliers tend to display performance metrics that significantly deviate from the average. This analysis delves into the performance metrics of NBA players across an entire season, aiming to identify outliers and gain insights into exceptional performances. We first aim to identify the performance metrics that contribute the most to outstanding performance. The bar chart below (Figure 2.) provides a visual representation of the average performance metrics across an entire season per game to help us identify which metric to focus on in quantifying outstanding player performance [8]. Performance metrics analysed included points, assists, rebounds, steals, and blocks. Points per game emerges as the most significant metric, highlighting the emphasis on scoring in basketball. Assists and rebounds follow closely, emphasizing their importance in facilitating scoring opportunities and controlling the flow of the game. While steals and blocks are valuable defensive contributions, their average values are lower, suggesting a general prioritization of offensive output.

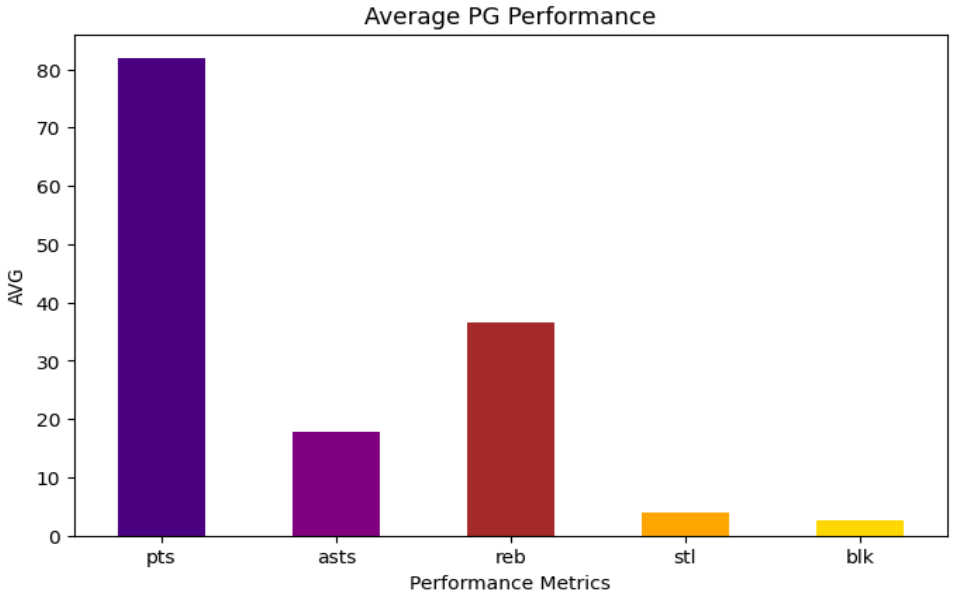

Figure 2. Bar Graph of Average Performance Metrics Across All NBA Seasons

To identify exceptional players, we seek those who significantly outperform these averages. However, it's crucial to recognize that outstanding performance isn't solely defined by points. Players who excel in multiple areas, including assists, rebounds, steals, and blocks, often contribute more holistically to their team's success. To understand how the distribution of performance metrics have changed across various NBA seasons we compute a time series of the average performance metrics and how they’ve changed and varied across different seasons demonstrated in the plot (Figure 3.) below. Points per year show the highest values, peaking in the 1970s after a steep rise from the 1940s, followed by a gradual decline in the ensuing decades. This trend might reflect changes in scoring efficiency, game pace, or even shifts in offensive strategies.

Rebounds also exhibit a sharp increase during the early years, stabilizing after reaching their peak around the 1960s and displaying mild fluctuations thereafter. This consistency could indicate the enduring importance of rebounding across different basketball eras, despite variations in playing styles. Assists, meanwhile, show a rapid initial rise, peaking around the same time as rebounds, and then remaining steady, suggesting an increasing emphasis on passing and team coordination in the sport's mid-century years [8 , 9].

Steals, blocks, and turnovers, which start with relatively low values, show a modest upward trend, especially from the 1970s onward. This increase may reflect a growing focus on defensive skills in basketball, as metrics like steals and blocks became more integral to player and team performance assessments [9]. Turnovers, by contrast, remain low and stable, possibly due to better ball-handling skills or limited early tracking of this metric.

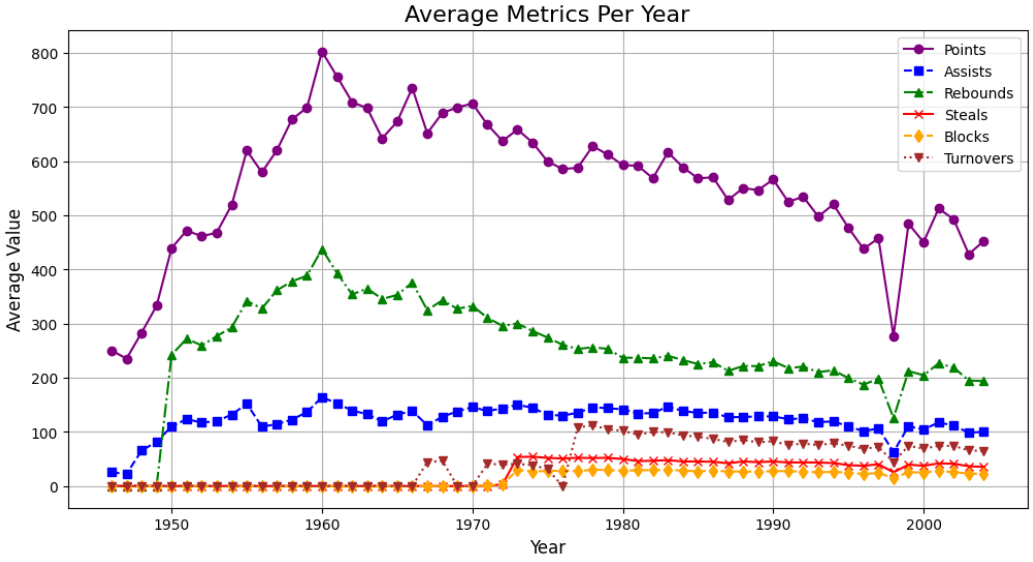

Figure 3. Average Performance Metrics Across Time

Now that we better understand the performance metrics of the NBA and how they change across seasons, we take a more in-depth look into the distribution and leaders of key NBA scoring metrics across all seasons, focusing on field goals, free throws, and points, displayed in Figure 4. below. The top row contains histograms showing the distribution of values for each metric, while the bottom row highlights the top four players in each category. This breakdown allows us to analyse both the general scoring trends among NBA players and identify the standout performers in each area. The "Field Goals Distribution" histogram reveals that most players score fewer than 50 field goals, with a steep drop-off as the number increases, indicating that only a select few players consistently reach higher field goal counts. Correspondingly, the bar chart for the "Top 4 Field Goal Scorers" identifies Hakeem Olajuwon, Michael Jordan, and Shaquille O'Neal as among the highest achievers, each recording a similar peak in field goals made, showcasing their scoring efficiency close to the basket [10].

The "Field Throws Distribution" graph shows a similar trend, with most players achieving low free throw counts, suggesting that only elite players frequently reach the free-throw line. Michael Jordan appears twice in the "Top 4 Field Throwers" chart, highlighting his dominance in drawing fouls and converting free throws [9,10]. Other notable players include

Larry Bird and Jerry West, known for their precision shooting and consistency. Lastly, the "Point Distribution" chart has an even steeper drop-off, with a majority of players scoring under ten points, emphasizing the exceptional scoring abilities required to rank among the top. The "Top 4 Scorers" chart showcases prolific three-point shooters like Reggie Miller and Ray Allen, whose ability to score from beyond the arc contributed significantly to their high point totals, alongside Dennis Scott and Dan Majerle. Together, these distributions and leaderboards underscore the rarity of high-scoring achievements in the NBA and highlight the impact of legendary players who consistently excelled across all scoring metrics, setting benchmarks that define elite performance in professional basketball.

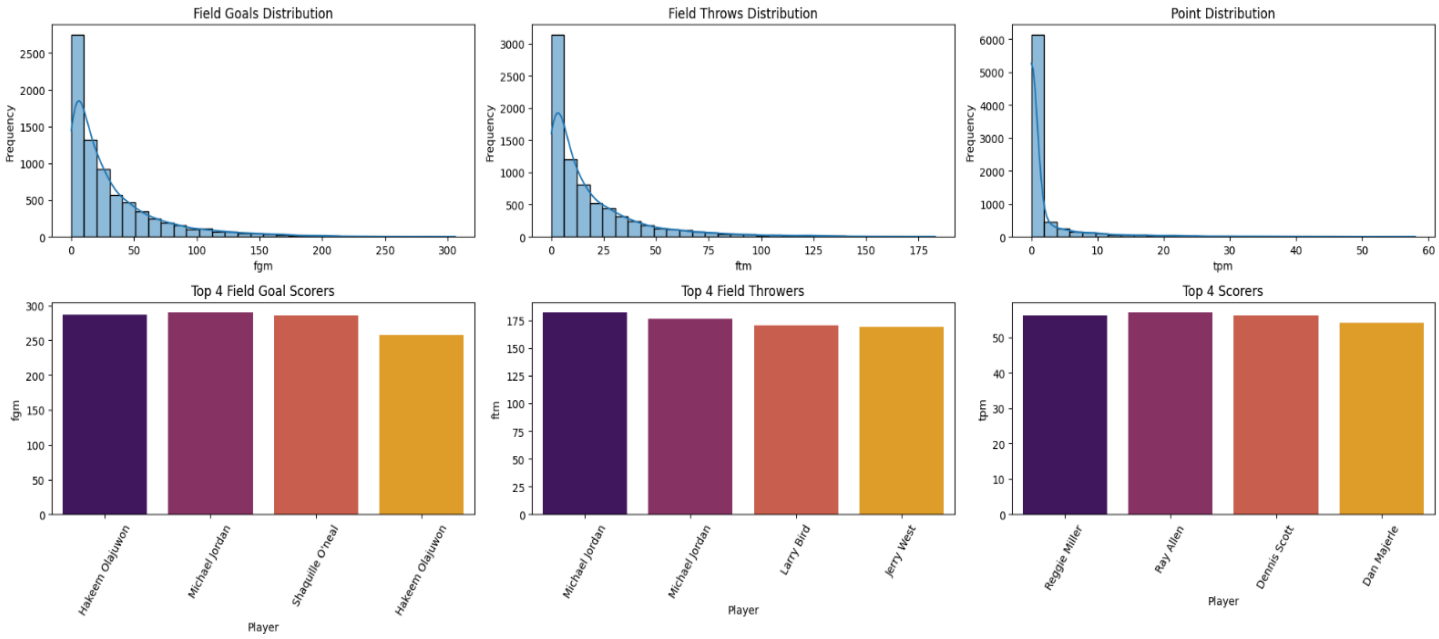

Figure 4. Field Goals, Fields Throws and Point Distribution, Along with Top 4 Performers for Each Metric

By further examining the distribution of field goal and free throw attempts, we can gain insights into the factors that contribute to individual success. In particular, the scatter plot of free throw attempts versus field goal attempts provides a visual representation of these relationships, allowing us to identify trends, outliers displayed in Figure 5. The "Free Throw Attempts vs. Field Goal Attempts (Playoffs)" scatter plot reveals a positive correlation between the two variables, indicating that players who attempt more field goals generally also attempt more free throws. The trendline further solidifies this relationship, suggesting that as field goal attempts increase, so do free throw attempts. However, the scatter is quite dispersed, implying that while there is a general trend, individual player variations exist. This could be attributed to differences in playing style, position, and team strategies.

Moving on to the specifics of the plot, we can observe that most players cluster towards the lower end of both axes, indicating that many players have relatively low attempts in both categories. This suggests that only a select group of elite players consistently reach the free-throw line and attempt a high number of field goals. The presence of notable players like Michael Jordan, Larry Bird, and Jerry West in the Figure 3. chart reinforces this idea. These players were known for their ability to draw fouls and convert free throws, contributing significantly to their overall scoring. Similarly, the Figure 3. chart highlights the rarity of

high-scoring performances. The majority of players score under ten points, emphasizing the exceptional skill and consistency required to reach the top echelons of scoring. Figure 3. chart showcases players like Reggie Miller and Ray Allen, who excelled in three-point shooting, a skill that became increasingly important in recent decades.

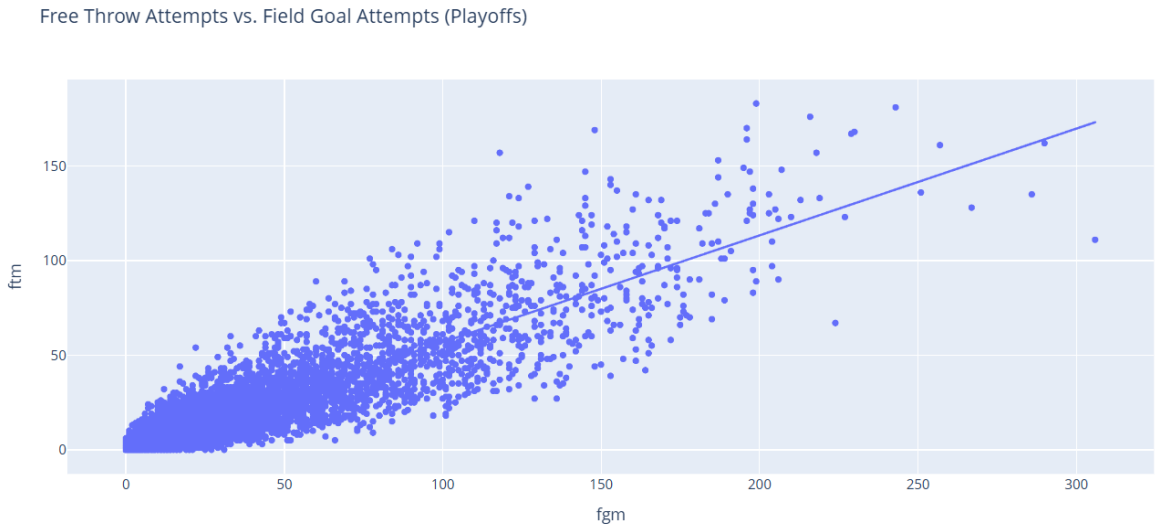

Figure 5. Free Throw Attempts versus Field Goal Attempts

Finally, wish to examine the score of metrics that characterise the outstanding players in the NBA. By having a grasp of the characteristics of outstanding NBA players, we can develop player profiles, better understanding the similarities and features that make these players elites of the game. We visualise characteristics of the 5 players with the highest points tally across all NBA seasons and compute plot box plots (Figure 6.) to showcase the different characteristics separating exceptional players from standard NBA players.

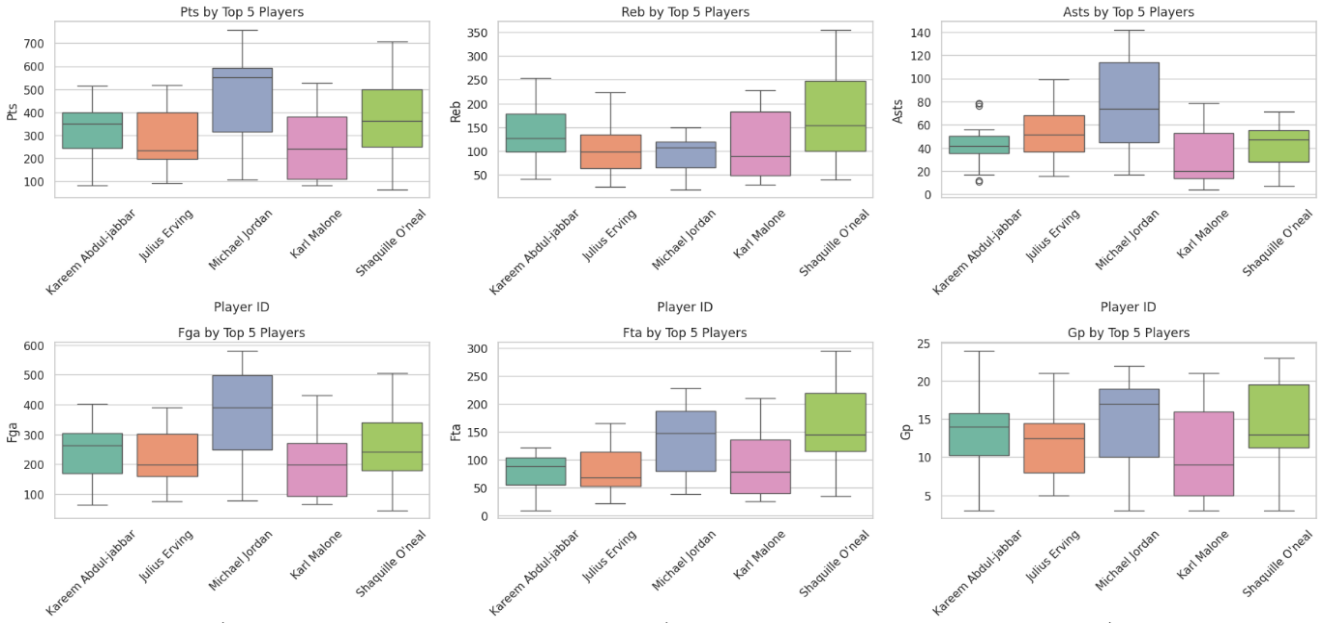

Figure 6. Box-Plots Of the Top 5 NBA Basketball Players Performance Metrics

The box plot analysis of the top 5 NBA players with the highest points reveals distinct characteristics that set these players apart and define their greatness on the court. Michael Jordan stands out with the highest range in points scored, alongside a high frequency of field goal attempts and significant assist contributions. His ability to contribute not only as a primary scorer but also as a playmaker highlights his offensive versatility and aggressive approach, making him a dominant figure in both scoring and facilitating team plays.

Shaquille O'Neal showcases a different form of dominance, excelling in rebounds and free throw attempts. His high rebound count and frequent trips to the free-throw line reflect his physical presence in the paint, where he draws contact and impacts possession. O'Neal's playing style emphasizes his strength and ability to influence the game through defensive stops and by consistently drawing fouls [8,10].

Kareem Abdul-Jabbar and Karl Malone exhibit more efficient scoring and rebounding styles, characterized by fewer field goal attempts and moderate assist numbers. These players focus on high-percentage shots and play within structured team offenses, making them efficient scorers who contribute reliably without needing excessive ball control. Abdul-Jabbar, in particular, has a significant presence in rebounds, showing his role as both a scorer and a defensive asset.

Julius Erving presents a balanced performance across scoring, rebounding, and efficiency. While he does not reach the extremes of Jordan or O'Neal in terms of specific metrics, his consistent contributions in points and rebounds reflect a well-rounded skill set. Erving plays a key role in scoring while maintaining a selective approach to shooting, showcasing his balanced and efficient style.

In summary, each of these high-performing players brings unique strengths to the court. Michael Jordan's all-around offensive versatility, Shaquille O'Neal's physical dominance and foul-drawing ability, Abdul-Jabbar and Malone's efficient and structured scoring, and Erving's balanced contribution all highlight the different ways that great players impact the game. Together, these characteristics illustrate the diverse skill sets that contribute to high levels of performance in the NBA.

2.2.2 Predicting Game Outcomes

It is essential to understand the performance of teams according to key metrics and how these metrics relate to the outcome of a game. We visualise the teams' performance based on our engineered features such as the point differentials per game and offensive field goal rates (Figures 7. and 8.). Next, we examine the distribution of game outcomes between each possible team to better understand the overall team performances (Figure 9.). We then make comparisons on performance based on the offensive points per game and point differentials of teams in both the NBA and ABA basketball leagues (Figures 10. and 11.). Finally, we analyse the trend of the average offensive points scored per game over almost six decades (Figure 12.).

The scatter plot in Figure 7. depicts the relationship between a team's point differential and its win percentage. The point differential describes the number of points by which teams win or lose in their games. There is an upward trend evident in the plot, indicating a clear positive

correlation between the point differential and win percentage of a team. The point differential serves as a key indicator of a team's overall offensive and defensive performance. Teams with higher positive point differentials tend to have higher win percentages over a season, as larger point differentials indicate consistent offensive scoring and effective defensive capabilities, leading to more positive game outcomes. In contrast to this, teams with negative point differentials potentially resulting from poor offensive or defensive performances, tend to show lower win percentages. This plot highlights that teams with a balanced approach in offensive and defensive strategies are more likely to secure consistent wins throughout a season.

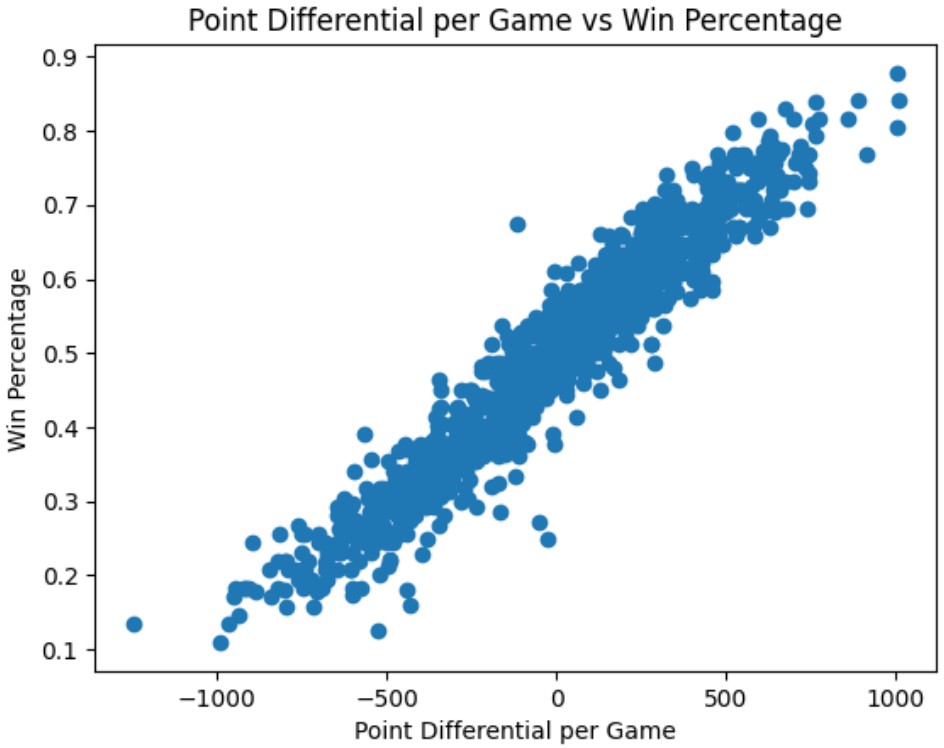

Figure 7. Scatter plot depicting the positive relationship between a team's point differential and win percentage.

The histogram in Figure 8. illustrates the frequency distribution of offensive field goal rates. This plot shows that most teams have offensive field goal rates between 0.40 and 0.50. This suggests that most teams have a 40% - 50% success rate when attempting a field goal, which offers insight into typical team performances. Notably, the distribution is skewed with a tail on the left (negatively skewed), indicating that a smaller subset of teams has a lower offensive field goal rate. This suggests that these teams struggle with consistently making successful field goal attempts. This provides valuable insight into understanding the variability in team performance which assists in predicting game outcomes.

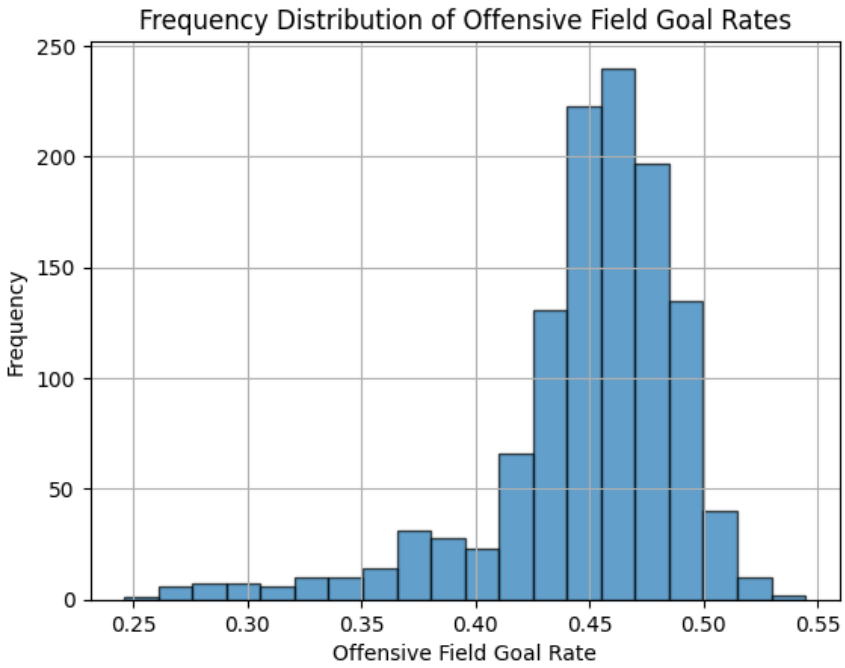

Figure 8. Frequency distribution of offensive field goal rates

The pie chart in Figure 9. visually represents the proportion of positive game outcomes between two teams, with the larger blue segment of the chart showing that Team B won 55.2 % of the games, and the orange segment indicating that Team A won 44.8 % of the games. Since Team A and Team B represent the first and second teams in all possible team matchups in a season, this chart provides an overview of the overall competitiveness between the teams in each matchup. The higher percentage of games won by Team B indicates that the second team in the matchup exhibited a slightly better performance throughout the season. This visual comparison enables more accurate predictions of future match results by highlighting the frequency of positive game outcomes in a season for each team.

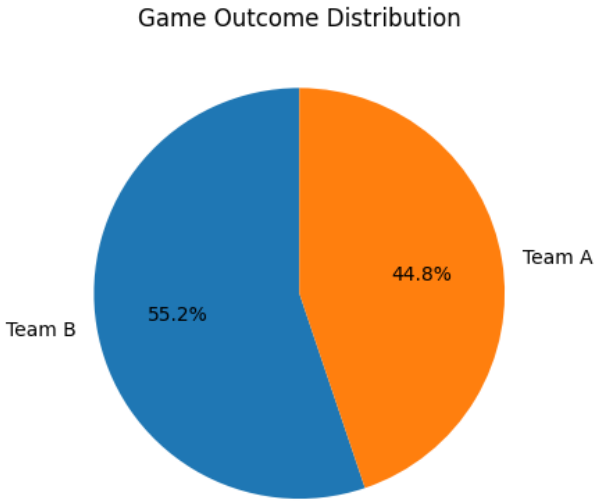

Figure 9. Pie chart displaying the proportion of games won by Team A (depicted in orange) and Team B (depicted in blue).

The box plots in Figure 10. illustrate a comparison of the offensive points scored per game between the NBA and ABA leagues, labelled “N” and “A”, respectively. The plot describing the NBA league shows that the median is approximately 105 points per game. The NBA league also shows a wider interquartile range (IQR) spanning approximately 95 to 110 points, suggesting moderate score variability. Notably, there is a significant number of outliers below 80 offensive points per game, indicating that some NBA teams achieved low offensive scores per game.

In contrast, the ABA league shows a higher median of approximately 112 offensive points per game. The IQR for the ABA league is narrower, with values ranging from approximately 109 to 115 points. This reflects more consistent scoring with less variability than the NBA league. There are fewer extreme outliers than the NBA league present with scores above 120 points and below 100 points per game. This implies that while consistency is present in the point scoring of the ABA league, there are deviations from typical offensive performances, possibly resulting from exceptional performances by star players or other factors, such as strong defensive performances. These insights are essential in understanding team offensive strategies across both leagues to better predict the winner of a game between two teams.

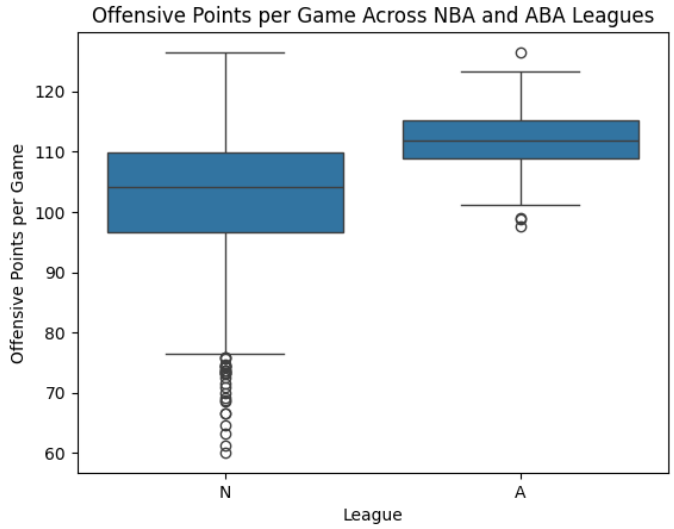

Figure 10. Box plots comparing the distribution of the offensive points scored per game by teams in the NBA and ABA basketball leagues, labelled “N” (left) and “A” (right), respectively.

The box plots in Figure 11. provide a visual comparison of the point differential per game for games played in the NBA and ABA leagues, labelled ‘N’ and ‘A’, respectively. The y-axis represents the number of points by which teams win or lose in their games, ranging from -1000 to 1000. A negative value indicates that a team allowed more defensive points than it scored, while a positive value implies more offensive points were scored than defensive points allowed.

League “N” shows a higher number of extreme outliers, indicating some fluctuations in team performance. However, the median value for both leagues is close to zero, suggesting that most games are balanced regarding scoring. The narrower IQR in league “N” indicates more consistent team performances which allow for more predictable game outcomes. In contrast, League “A” has a wider IQR with no extreme outliers, suggesting slightly greater variability

in team performance, leading to larger point differentials. This visualisation underscores the differences in point-scoring variability between both leagues and provides valuable insights into the dynamics of team performance.

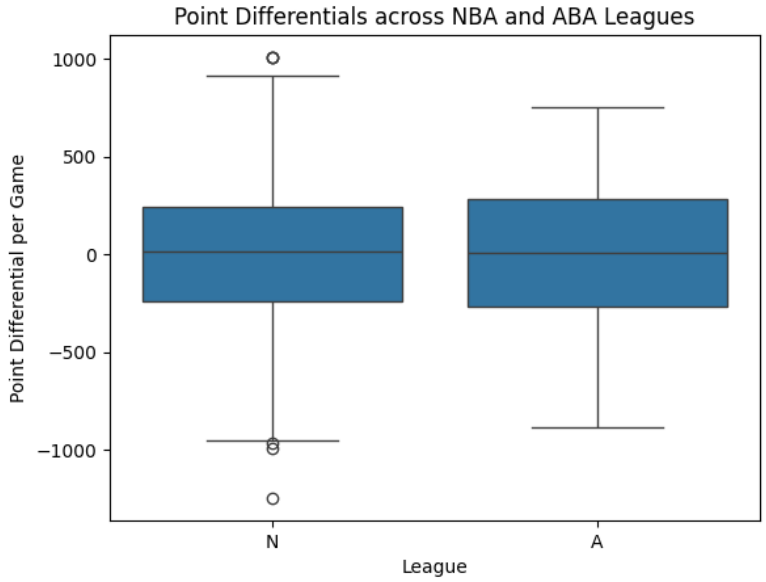

Figure 11. Box plots comparing the distribution of the point differentials achieved by teams in the NBA and ABA leagues, labelled "N" (left) and "A" (right) respectively.

The line graph in Figure 12. depicts the trend in the average offensive points scored per game over approximately six decades. This graph shows a notable rise in the average offensive points per game from the late 1940s to the early 1960s with a peak in the early 1960s at approximately 118 points per game. This is followed by a noticeable decline after the mid-1970s.

Historical context, particularly the merger of the ABA and NBA basketball leagues in 1976 [5], is essential in interpreting this trend. This merger integrated many ABA teams into the NBA league, blending different strategies and playing styles from both leagues. Following the merger, there was a slight increase in the average points scored, which then stabilised around 110 points in the mid-1980s. From the late 1980s onward, the average offensive points per game declined steadily, reaching 95 points by 2003. This plot reflects the long-term impact of the merger, illustrating how the integration of different playing styles influenced the offensive performance of the teams.

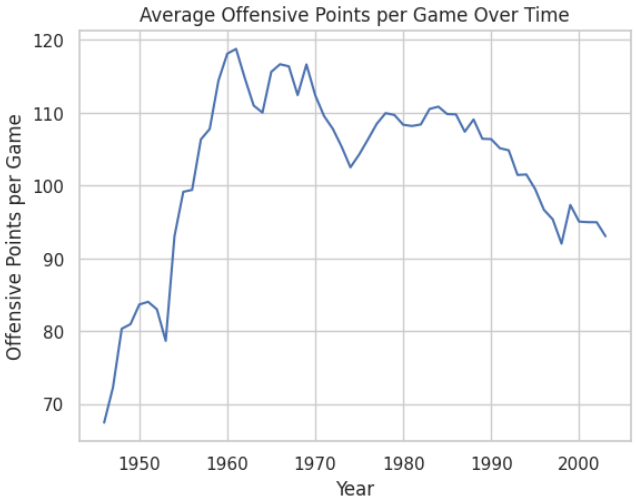

Figure 12. Line graph showing the trend of the average offensive points scored per game over time.

2.3 Model Selection and Training

2.3.1 Outlier Detection of Players

(a) Anomaly Detection Using K-Means Clustering and Principal Component Analysis

Outlier detection is a critical process in data mining, especially when analysing datasets that contain high-dimensional data, such as NBA player statistics. In this analysis, we utilize Principal Component Analysis (PCA) and K-Means clustering to detect outliers among NBA players, identifying those whose performance metrics are statistically different from the rest. This approach is especially relevant for the NBA dataset, where numerous player attributes can lead to high dimensionality, making it challenging to isolate unique performance patterns without reducing complexity.

The process begins by preprocessing the NBA data, where non-numeric columns, such as identifiers and player names, are removed. The remaining numeric data undergoes normalization using Min-Max scaling to ensure that all variables are within the same range [0,1]. This scaling is essential for K-Means clustering, which is sensitive to the magnitude of data and can produce skewed results if variables are on different scales.

Following normalization, we apply PCA, which reduces the dimensionality of the dataset to two principal components (PC1 and PC2). PCA is chosen here to distil the data into its most significant patterns, capturing the variance that represents the core structure of player performance metrics [11]. The two-dimensional transformation of the data allows us to visualize clusters and identify outliers in a simpler, more interpretable space. In the context of NBA performance analysis, these principal components capture combined metrics that reflect comprehensive aspects of player contributions, such as scoring, efficiency, and physical dominance, which would otherwise be challenging to visualize in high-dimensional space.

Once the data is transformed, we proceed to determine the optimal number of clusters (k) for K-Means clustering. The Elbow method is applied, which involves calculating the sum of squared errors (SSE) for different values of k. A plot of SSE versus the number of clusters reveals an "elbow point" (Figure 13.), indicating the value of k where adding more clusters

yields diminishing returns in reducing error. This criterion helps ensure that the clustering captures natural groupings within the data without overfitting.

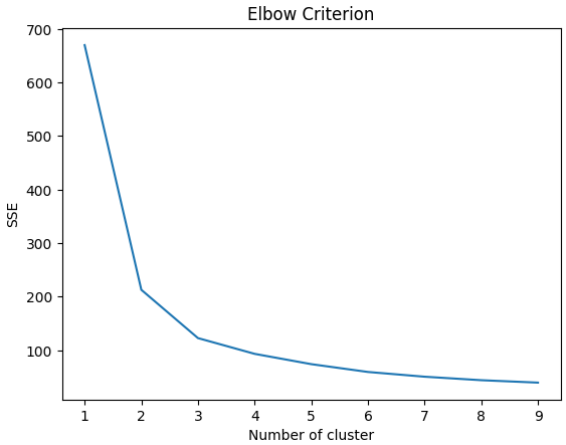

Figure 13. Elbow Criterion for Number of Clusters

We also compute the silhouette score for each cluster count, which quantifies how similar players are within clusters compared to other clusters. The silhouette score helps validate the choice of k, ensuring well-separated and cohesive clusters.

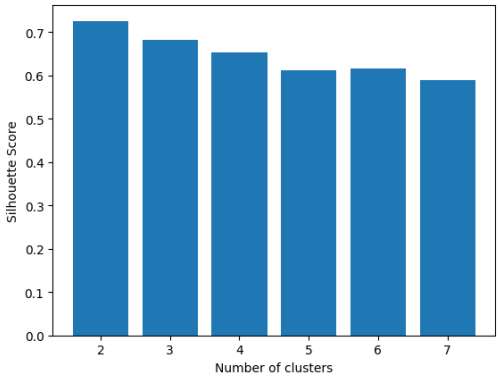

Figure 14. Silhouette Scores Against Number of Clusters

With the optimal number of clusters determined, we assign each player to a cluster using K-Means clustering based on their coordinates in the principal component space (PC1, PC2). Each cluster represents a group of players with similar performance profiles. The cluster centroids are also calculated, representing the average position of players within each cluster. Formally we present the K-Means clustering algorithm as follows adapted from A. Barai, and L. Dey [11] :

Algorithm 1 Cluster-Based Outlier Detection in K-Means Clustering

Input: Dataset $X = \{x_1, x_2, \ldots, x_m\} \subset R^n$, Number of clusters k
Output: Updated dataset X' with smallest cluster removed

Phase 1: Initial Clustering and Smallest Cluster Identification
Run K-Means clustering on X to obtain k clusters, $C_1, C_2, \ldots, C_k$
Let $|C_j|$ represent the number of points in cluster C_j for $j = 1, 2, \ldots, k$
Identify the cluster $C_{\min} = \arg\min_j |C_j|$ with the smallest number of points

Phase 2: Outlier Removal and Re-Clustering
Let $X' = X \setminus C_{\min}$ (remove all points in the smallest cluster)
Run K-Means clustering on X' to obtain new clusters $C'_1, C'_2, \ldots, C'_{k-1}$
Compute updated clustering accuracy $\text{Accuracy}_{\text{final}}$ and silhouette index $\text{Silhouette}_{\text{final}}$

To identify outliers, we employ a Gaussian-based approach, which assesses the likelihood of a player belonging to the underlying distribution of each cluster. First, we estimate the mean (μ) and covariance (σ) of the data in the two-dimensional principal component space, assuming a multivariate Gaussian distribution. We then calculate the probability density function (PDF) for each player based on these parameters. Players with probability scores below a defined threshold (epsilon = 0.05) are considered outliers, as their performance metrics deviate significantly from the typical profiles within their clusters.

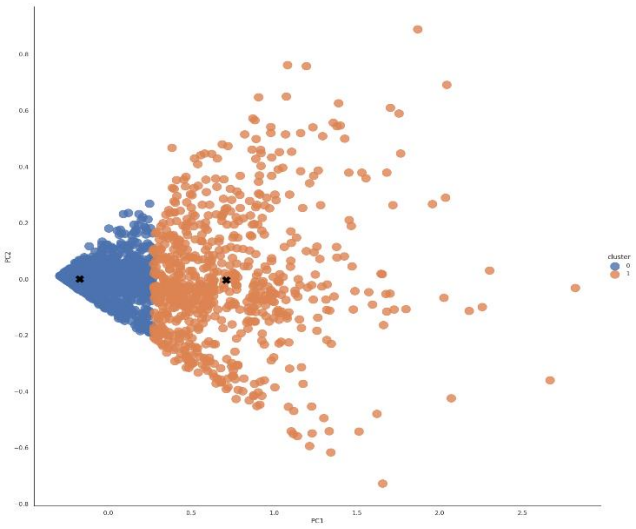

Figure 15. Gaussian Distribution of Players on Scatter Plot

In the context of NBA performance, these outliers represent players with exceptionally unique attributes. For example, a player who scores very high in points while having unusually low rebounds or assists could be flagged as an outlier. This method contrasts with simpler clustering approaches by incorporating multivariate relationships that capture the complexity of player performance, providing a nuanced analysis of anomalies. By isolating these outliers, we identify NBA players who have distinctive styles or specialized skills, setting them apart from the general population of players. Outliers are highlighted in yellow in Figure 16. below.

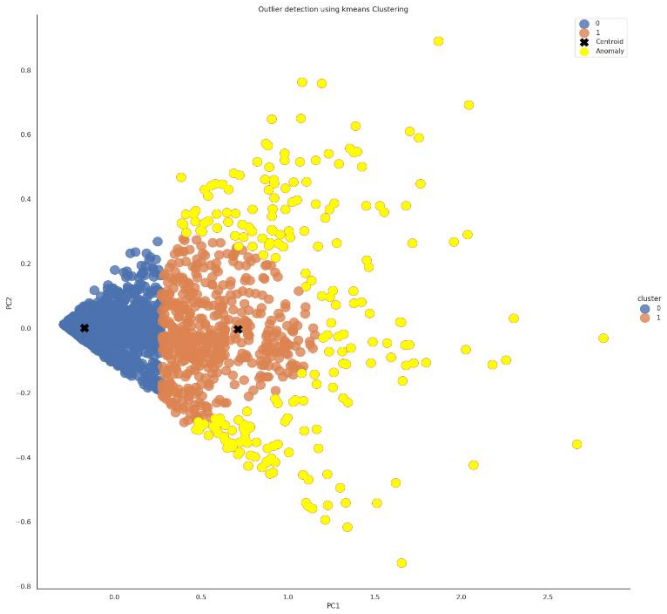

Figure 16. Outliers Displayed on Gaussian Distribution

(b) Distance Based Outlier Detection

The distance-based probabilistic outlier detection algorithm applied to NBA player data involves four main steps, each carefully designed to identify outliers based on player performance metrics.

First, the player data undergoes data normalization. The selected performance metrics, such as games played, points, rebounds, and assists, are normalized to a [0,1] range using Min-Max scaling. This normalization ensures that metrics with different scales are comparable, which is essential for accurate distance calculations in a multivariate space . We then perform the distance-based outlier detection utilizing the following algorithm displayed below adapted from D. Muhr, M. Affenzeller, and J. Küng [12]:

Algorithm 2 Distance-Based Outlier Detection with Probabilistic Scoring

1. **Input:** Dataset X of N players, where each player i has a performance vector $x_i \subset R^d$ (with d features).
2. **Step 1: Calculate Mean Vector**
   - Compute the mean vector $\mu \subset R^d$ across all players:
   $$\mu = \frac{1}{N} \sum_{i-1}^{N} x_i$$
3. **Step 2: Calculate Euclidean Distances**
   - For each player i, calculate the Euclidean distance d_i from the mean vector μ:
   $$d_i = \|x_i - \mu\|_2 = \sqrt{\sum_{j-1}^{d} (x_{ij} - \mu_j)^2}$$
   where x_{ij} is the j-th feature of the player vector x_i and μ_j is the j-th feature of the mean vector μ.
4. **Step 3: Determine Threshold for Outlier Detection**
   - Compute the threshold T as the 97th percentile of the distances $\{d_i\}_{i-1}^{N}$:
   $$T = \text{percentile}(d, 97)$$
5. **Step 4: Calculate Probabilistic Outlier Scores**
   - Let $d_{\max} = \max_{i-1,\ldots,N} d_i$.
   - For each player i, calculate the probabilistic outlier score P_i as follows:
   $$P_i = \begin{cases} 0 & \text{if } d_i \leq T \\ \frac{d_i - T}{d_{\max} - T} & \text{if } d_i > T \end{cases}$$
6. **Output:** Probabilistic outlier scores $\{P_i\}_{i-1}^{N}$ for each player, indicating the likelihood of each player being an outlier.

Finally, in result visualization, players are ranked by their probabilistic scores to identify those with the highest likelihood of being outliers. For clarity, we generate a scatter plot of the distances, with the outliers highlighted. This visual representation allows us to see the spread of distances and better understand how the probabilistic scoring helps differentiate outliers from non-outliers.

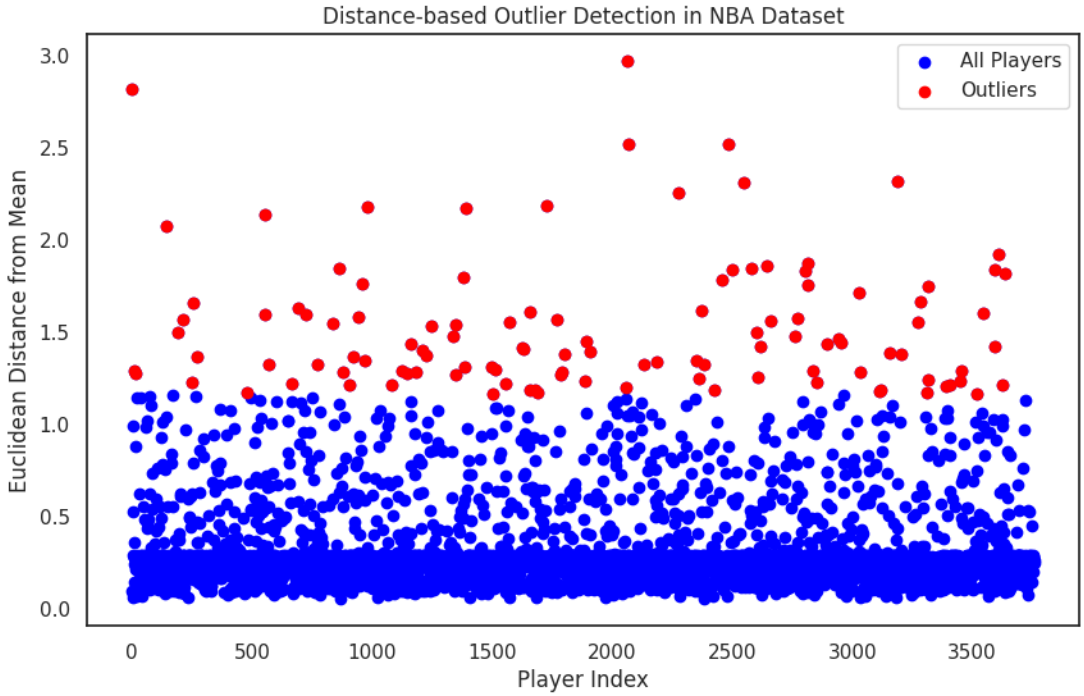

Figure 17. Player Distribution by Distance

(c) Local Linear Regression (LOESS) Based Classification

This process applies LOESS regression to detect outliers in an NBA dataset by leveraging both numerical and categorical attributes, aiming to capture nuanced relationships in player performance data.

First, the data is pre-processed by normalizing numerical attributes, such as points scored, rebounds, assists, steals, and blocks, scaling them between the range [0,1]. This step ensures uniformity, reducing the effect of varying scales on regression calculations. The algorithm for Local Linear Regression can be described below adapted from a paper by L.-C. Chen, T.-T. Kuo, W.-C. Lai, S.-D. Lin, and C.-H. Tsai [13] :

1. Data Preparation: One-Hot Encoding Categorical Features

Given:

- $\mathbf{X}_{\text{num}}$ is the matrix of numerical attributes (e.g., points, rebounds, assists, steals, blocks).
- $\mathbf{X}_{\text{cat}}$ is the matrix of categorical attributes (e.g., player position, league).

Steps:

1. One-hot encode $\mathbf{X}_{\text{cat}}$:

$$\mathbf{X}_{\text{binary}} = \text{OneHotEncode}(\mathbf{X}_{\text{cat}})$$

2. Concatenate $\mathbf{X}_{\text{num}}$ and $\mathbf{X}_{\text{binary}}$ to form the complete feature matrix:

$$\mathbf{X} = [\mathbf{X}_{\text{num}} \mid \mathbf{X}_{\text{binary}}]$$

2. LOESS Regression Setup

For each player data point $\mathbf{x}_i$ in $\mathbf{X}$:

a. Distance Calculation

- Compute Euclidean distance for numerical attributes:

$$d_{\text{num}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_{i,\text{num}} - \mathbf{x}_{j,\text{num}}\|_2$$

- Compute Hamming distance for categorical attributes:

$$d_{\text{cat}}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{p} \mathbf{1}(\mathbf{x}_{i,\text{binary}}^k \neq \mathbf{x}_{j,\text{binary}}^k)$$

- Combine distances into a hybrid metric:

$$D(\mathbf{x}_i, \mathbf{x}_j) = d_{\text{num}}(\mathbf{x}_i, \mathbf{x}_j) + d_{\text{cat}}(\mathbf{x}_i, \mathbf{x}_j)$$

b. Neighbor Selection

- Define the fraction of neighbors to consider, $f \in (0, 1]$.
- Sort distances $D(\mathbf{x}_i, \mathbf{x}_j)$ in ascending order and select the nearest $k = \max(3, f \cdot n)$ neighbors of $\mathbf{x}_i$.

c. Weight Assignment

- Compute weights based on proximity:

$$w_{ij} = \left(1 - \left(\frac{d_{\text{num}}(\mathbf{x}_i, \mathbf{x}_j)}{\max_j d_{\text{num}}(\mathbf{x}_i, \mathbf{x}_j)}\right)^3\right)^3$$

d. Local Regression Fitting

- Fit a local regression model using the weighted neighbors to estimate the predicted value $\hat{y}_i$ for $\mathbf{x}_i$.

3. Outlier Detection: Residual Calculation and Thresholding

For each player $\mathbf{x}_i$:

- Compute the residual r_i:

$$r_i = |y_i - \hat{y}_i|$$

- Define the outlier threshold:

$$T = 2 \cdot \sigma_r$$

  where σ_r is the standard deviation of residuals across all players.

- Mark $\mathbf{x}_i$ as an outlier if:

$$r_i > T$$

4. Output

Return a list of players where $r_i > T$, indicating players whose performance deviates significantly from expectations based on peers with similar attributes.

Finally, the results are visualized by plotting both actual and smoothed (predicted) data points, highlighting the outliers in Figure 18. This graphical analysis helps validate the model's ability to detect players with unique performance profiles, providing a refined view of outlier detection that considers interdependencies within a heterogeneous dataset [13]. This LOESS-based approach, therefore, offers a robust and adaptable method for identifying atypical players in complex data environments.

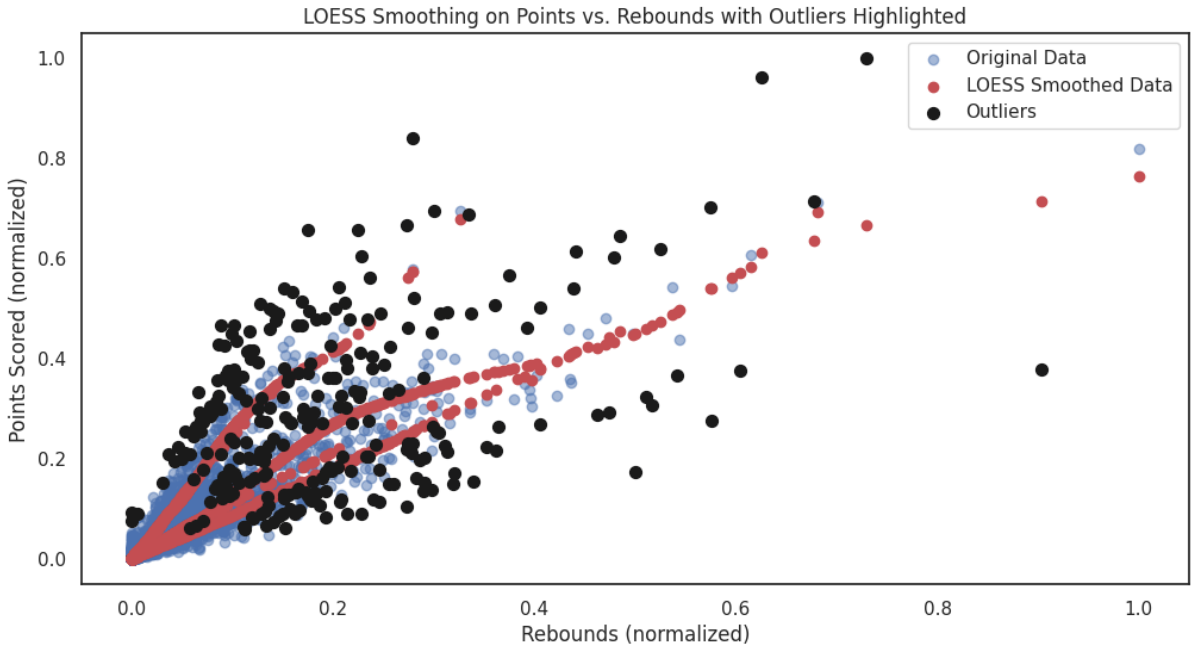

Figure 18. LOESS Smoothing against Rebounds

2.3.2 Model Selection and Training for Game Prediction

(a) Principle Component Analysis

Using a feature reduction technique can help reduce the computational overhead of training and prevent the model from overfitting. Principle Component Analysis (PCA) is a popular feature reduction technique in machine learning [6]. It aims to reduce the dimension of the training and testing data while also retaining the variance of the data. PCA works by computing the mean vector μ and the covariance matrix of the original datapoints. The diagonal of the covariance matrix describes the variance (how much a variable varies). The eigenvalues and eigenvectors of the covariance matrix are then computed and the top n vectors are chosen, where n is the dimension that the features are being reduced to. The points are then projected onto the subspace spanned by these eigenvectors so that the new point is given by

$$y = A(x - \mu) \tag{16}$$

where y is the new point, x is the old point, μ is the corresponding value in the mean vector and A is a matrix where the rows are made up of the n eigenvectors. PCA was used to reduce the dimension of the data to 12 features. This is what the following machine learning models were trained with to output some game prediction.

(b) Five-fold Cross Validation

All machine learning models used in game prediction were done using K-fold cross validation with K=5. In this case, it's called five-fold cross validation. The reason for this was to ensure that each machine learning model performed consistently across the entire dataset

and not just one specific split. The training and test data were concatenated, and divided into five folds, where four of the folds were used for training, and one for testing. This was done five times, changing which one of the five folds was used for testing each time. The average of these five accuracies and their standard deviation are used to evaluate the performance of each machine learning model [7].

(c) Model 1 – Random Forest Classifier and Decision Tree Classifier

A Random Forest Classifier is an ensemble machine learning method that is primarily made up of Decision Trees. The outputs of each Decision Tree are merged to improve the prediction accuracy of the machine learning model, rather than just using one Decision Tree machine learning algorithm for a classification task. The Decision Tree algorithm starts off with all of the training data at a singular node that undergoes sequential splitting until we arrive at two pure child nodes. There are cases where there aren't two 100% pure nodes. The machine learning model learns the best condition for splitting at each node by maximising the Information Gain at that particular split. This is done by choosing the split that minimises the entropy of the child nodes for each split. Entropy is the measure of information contained in a state. High entropy (1) corresponds to least information and low entropy (0) corresponds to most information, entropy=0 is found at a pure node. Entropy is calculated using:

$$Entropy = \sum -p_i \log_2(p_i) \tag{17}$$

Where, p_i is the probability of class i. Information Gain is then calculated by subtracting the combined entropy of the child nodes from the entropy of the parent node.

$$IG = E(parent) - \sum w_i E(child_i) \tag{18}$$

Where w_i is the weight with respect to a child node and is equal to the relative size of the child node compared to the parent node. When the machine learning model is being trained with a Decision Tree Classifier, it compares every possible split and chooses the split that has the highest Information Gain and it continues this process at each split until it reaches a state of two pure nodes.

Random Forest Classifiers uses a technique called Bootstrapping. Bootstrapping consists of creating multiple subsets of training data by sampling with replacement and learning one model, a Decision Tree in this case, on each replicate. Given certain features, each trained Decision Tree will output a predicted class, but the model will output a final prediction based on the majority voting from all individual Decision Trees.

(d) Model 2 – Naïve Bayes Classifier

The Gaussian Naïve Bayes classifier is a probabilistic machine learning model that assumes the features in the data are independent of each other. Bayes theorem is the foundation of the Gaussian Naïve Bayes classifier.

$$Bayes\ Theorem:\ P(h \mid D) = \frac{P(D|h)P(h)}{P(D)} \tag{19}$$

Bayes' theorem is used to calculate the probability of a class *C*, also known as the *posterior probability*, from past (training) data. For Gaussian Naïve Bayes, each feature is assumed to follow a Gaussian distribution within each class. Due to this, the *likelihood* for each feature is modelled as a Gaussian distribution and used in this formula

$$posterior\ probability\ =\ \frac{likelihood \times prior}{evidence} \tag{20}$$

Where *prior* is the prior probability of the class in general and *evidence* is the probability of getting certain values for certain features. Given a new observation with new feature values the Gaussian Naïve Bayes classifier computes the posterior probability for each class using Bayes' theorem and these new features. Whichever class has the highest probability is the class that is assigned to this new observation.

(e) Model 3 - Support Vector Machines

Support Vector Machines (SVMs) are machine learning models that find an optimal hyperplane that can separate the data points of different classes in a high-dimensional space. For non-linearly separable data, SVMs use a kernel function to map the data into a higher-dimensional space where it can be linearly separated. SVMs are trained to find the decision boundary that maximizes the margin between two classes. The support vectors that correlate to the margin are vectors that point from the decision boundary to the points from each class that are closest to the decision boundary. The margin is the distance between the hyperplane and the support vectors. SVMs aim to maximize this margin while also minimising the classification errors, as a larger margin implies a better generalisation to new test data.

It aims to maximise the margin by finding optimal weights, $\overline{w}$, through the weight optimisation problem expressed as,

$$\min_{\overline{w}} \frac{1}{2} \| \overline{w} \|^{\ 2}\ subject\ to\ y_i(\overline{w_i} \cdot \overline{x_i}) \geq 1,\ \forall i \tag{21}$$

where $\overline{w}$ is the vector of weights that defines the decision boundary, $\overline{x_i}$ is the feature vector for training sample *i* and $y_i \in \{-1,1\}$ is the class label for training sample *i*.

The class prediction $f(\overline{x_q})$, for a new example x_q can be given by

$$f(\overline{x_q}) = sign\left(\sum_{i=0}^{N} \overline{w}_i y_i K(\overline{x_q}, \overline{x_i})\ \right) \tag{22}$$

where the kernel function

$$K(\overline{x_q}, \overline{x_i}) \tag{23}$$

is a similarity measure between the two datapoints $\overline{x_i}$ and $\overline{x_q}$. There are many kernel functions that can be used when utilising SVMs. In this project, we use the three kernel functions shown in Table 1.

Table 1: Table of formulas for the three kernel functions used to train Support Vector Machines in this project.

Kernel Function Name	Kernel Function
Linear	$K(\overline{x_q}, \overline{x_i}) = \overline{x_q} \cdot \overline{x_i}$ (24)
Polynomial	$K(\overline{x_q}, \overline{x_i}) = (\overline{x_q} \cdot \overline{x_i} + c)^d$ (25)
Radial Basis Function (RBF)	$K(\overline{x_q}, \overline{x_i}) = \exp\left(\frac{-\|\overline{x_q} - \overline{x_i}\|^2}{2\sigma^2}\right)$ (26)

The Linear kernel function is typically used for linearly separable data, the Polynomial kernel function is usually used when there is some type of polynomial relation between the datapoints and the RBF function is commonly used with non-linearly separable data, yielding complex, non-linear decision boundaries.

(f) Model 4 – Stacking

The last model that will be used to predict the game outcome, is a stacking model that applies two base learner machine learning models, whose outputs are the input for a meta-learner machine learning algorithm. The meta-learner then outputs a final predicted class. The two base learners chosen for this project are the k-Nearest Neighbour Classification (KNN) algorithm and the Decision Tree algorithm and the meta-learner is the Gaussian Naïve Bayes algorithm. Descriptions for the Decision Tree algorithm and the Gaussian Naïve Bayes algorithm can be found in Section 2.3.4(c) and Section 2.3.4(d), respectively.

The k-Nearest Neighbour algorithm works by taking the instance $\overline{x}$ to be classified and find the k nearest neighbours of $\overline{x}$ in the training data. For the two points that are closest to $\overline{x}$, the algorithm then determines the class of the majority of the instances among the k nearest neighbours. The algorithm will return the class with the majority of the instances as the classification of x.

3 Results and Discussion

3.1 Evaluation Metrics

a) *Accuracy:* measures the overall correctness of the model. Higher accuracy indicates better performance.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (27)$$

b) Precision: indicates how often the model correctly predicts positives, thus reducing false positives. Higher precision means fewer incorrect positive predictions.

$$Precision = \frac{TP}{TP+FP} \quad (28)$$

c) *Recall:* measures the model's ability to correctly classify true positives. Higher recall signifies fewer false negatives.

$$Recall = \frac{TP}{TP+FN} \tag{29}$$

d) *F1-Score:* balances the precision and recall metrics. A higher F1-Score indicates a better balance, especially in cases of uneven class distribution.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{30}$$

3.2 Outlier Detection of Players

We implemented three outlier detection methods: Clustering-Based, Distance-Based, and LOESS Smoothing. To determine the optimal number of clusters for the Clustering-Based method, we analysed the sum of squared distances and used the elbow method (figure 13). The elbow point, where the curve significantly bends, indicated that two clusters were ideal. We further confirmed this using the silhouette score (figure 14), which was highest for two clusters.

For Clustering-Based Outlier Detection, we assigned data points to two clusters and identified outliers as those falling below a specific distance threshold from the cluster centre. For Distance-Based Outlier Detection, we calculated the Euclidean distance of each data point from the mean and labelled points exceeding the 97th percentile of distances as outliers. LOESS Smoothing, which can handle categorical attributes, was used to calculate residual errors between original and smoothed values. Points with residuals exceeding two standard deviations were classified as outliers.

To provide insights into the reasons for outlier classification, we employed natural language explanations for each method. We evaluated the performance of each method using a confusion matrix and calculated accuracy, precision, recall, and F1-score.

3.2.1 Results for Outlier Detection of Players

Table 2: Results of the three models implemented for Outlier Detection of Players.

Model Name	Accuracy	Precision	Recall	F1-Score
LOESS Smoothing	90.43	58.17	36.17	44.61
Distance-Based Outlier Detection	91.48	**87.61**	23.40	36.90
Clustering-Based Outlier Detection	**91.91**	72.97	**38.30**	**50.23**

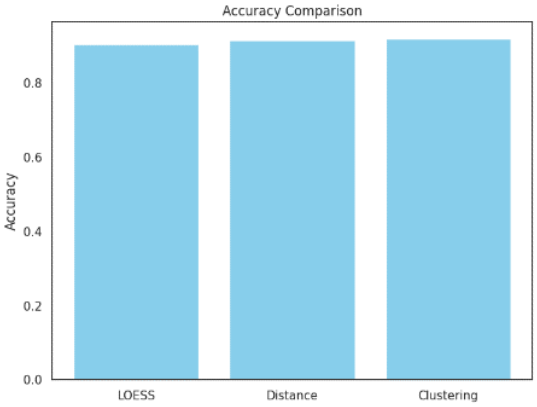

Figure 19. Accuracy Bar Graph.

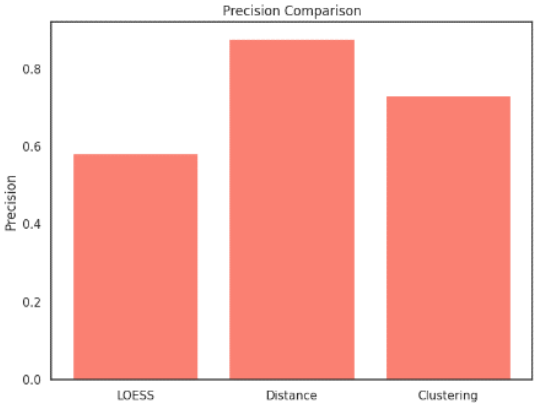

Figure 20. Precision Bar Graph.

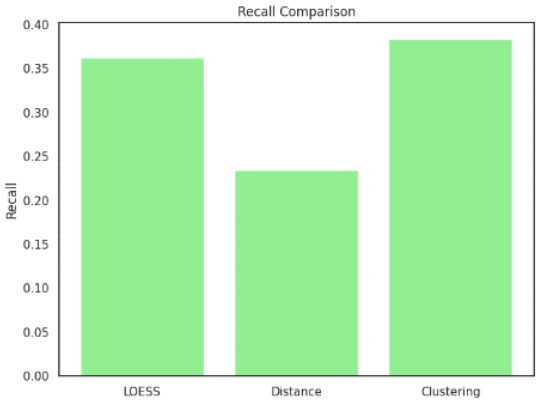

Figure 21. Recall Bar Graph.

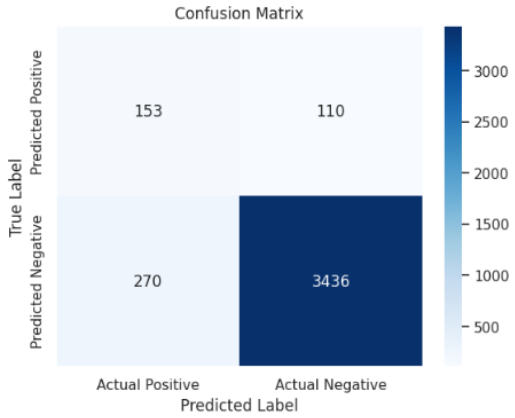

Figure 22. LOESS Smoothing Confusion Matrix.

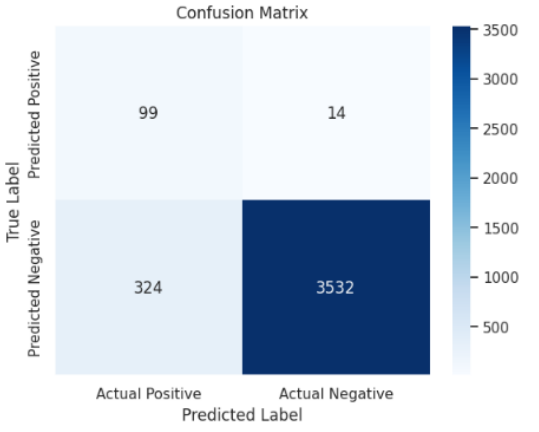

Figure 23. Distance Based Outlier Detection Confusion Matrix.

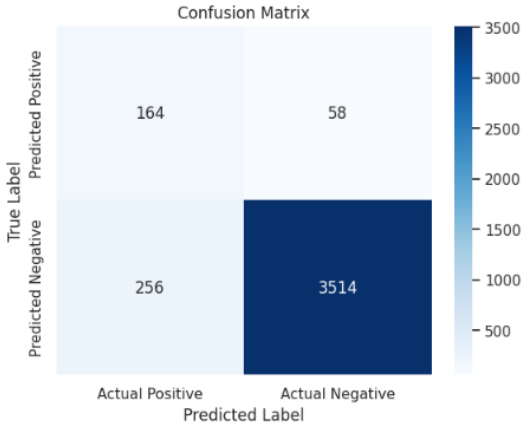

Figure 24. Clustering Based Outlier Detection Confusion Matrix.

```
['Kareem Abdul-jabbar' 'Mahmo Abdul-rauf' 'Alvan Adams' 'Mark Aguirre'
 'Cadillac Anderson' 'Nick Anderson' 'Shandon Anderson' 'Nate Archibald'
 'Paul Arizin' 'Alvin Attles' 'Stacey Augmon' 'Charles Barkley'
 'Dick Barnett' 'Rick Barry' 'Elgin Baylor' 'Butch Beard' 'Dave Bing'
 'Larry Bird' 'Otis Birdsong' 'Rolando Blackman' 'Corie Blount'
 'Arlen Bockhorn' 'Tom Boerwinkle' 'Muggsy Bogues' 'Bill Bradley'
 'Jim Brewer' 'Bill Bridges' 'Junior Bridgeman' 'Fred Brown' 'P.j. Brown'
 'Rogera Brown' 'Michael Cage' 'Mack Calvin' 'Tony Campbell' 'Kenny Carr'
 'Joe Barry Carroll' 'George Carter' 'Bill Cartwright' 'Harvey Catchings'
 'Tom Chambers' 'Don Chaney' 'Rex Chapman' 'Calbert Cheaney'
 'Doug Christie' 'Jim Cleamons' 'Jack Coleman' 'Lester Conner'
 'Michael Cooper' 'Terry Cummings' 'Earl Cureton' 'Lou Dampier'
 'Bob Dandridge' 'Adrian Dantley' 'Brad Daugherty' 'Antonio Davis'
 'Dale Davis' 'Johnny Davis' 'Terry Davis' 'Walter Davis'
```

Figure 25. Sample of the names of the predicted outliers.

3.2.2 Analysis of Results for Outlier Detection of Players

a) LOESS Smoothing:

The *LOESS smoothing* technique shows an accuracy of 90.43%, indicating that a significant proportion of the predictions were correct. However, its *precision* of 58.17% suggests that among all the predictions it made for outstanding players, only 58.17% were correct. This means that it had a relatively high false positive rate, suggesting that many non-outstanding players were mistakenly classified as outstanding.

The *recall* of 36.17% indicates that only 36.17% of the actual outstanding players were correctly identified. This relatively low recall indicates that LOESS smoothing missed many of the truly outstanding players, classifying them as non-outstanding. As a result, the *F1-score* of 44.61% reflects a modest balance between precision and recall, showing room for improvement in both detecting outstanding players and reducing false positives.

The high number of true negatives (3436) indicates that the model was generally good at identifying non-outstanding players, but the number of false positives (270) indicates that many non-outstanding players were misclassified as outstanding. Additionally, a substantial number of false negatives (110) shows that many actual outstanding players were missed.

b) Distance-Based Outlier Detection:

The *distance-based outlier detection* method performs slightly better than LOESS smoothing in terms of accuracy (91.48%), but it has a substantially higher *precision* of 87.61%. This means it is much better at correctly identifying outstanding players compared to LOESS. However, the *recall* of 23.40% shows that it misses a significant number of actual outstanding players, only identifying 23.40% of them.

This contrast between *precision* and *recall* results in a lower *F1-score* of 36.90%, indicating that while this method is good at avoiding false positives, it fails to capture many of the true positives. This suggests that the model is highly conservative, identifying only the most obvious outstanding players, but potentially missing a large portion of them.

This method had a high number of true negatives (3532), indicating a good ability to identify non-outstanding players. However, it also had a high false positive count (324), which

impacted its overall precision. Additionally, the small number of false negatives (14) shows that it missed very few actual outstanding players.

c) Clustering-Based Outlier Detection:

The *clustering-based outlier detection* method performs similarly to the distance-based method in terms of accuracy (91.91%), but it strikes a better balance between *precision* (72.97%) and *recall* (38.30%). The higher *precision* indicates that it has fewer false positives than LOESS smoothing, and the *recall* of 38.30% is better than both LOESS and distance-based methods, meaning it identifies more true positives.

The *F1-score* of 50.23% reflects a more balanced performance, showing that clustering-based outlier detection is better at capturing a wider range of true positives while still maintaining a reasonably low false positive rate.

Clustering-based outlier detection strikes a better balance between false positives (256) and false negatives (58), capturing more true positives (164) and maintaining a reasonable false positive rate (256). The method correctly identified 164 outstanding players and missed 58 (FN), showing it is better at detecting outliers than the other methods.

3.2.3 Discussion for Outlier Detection of Players:

From the results, it's clear that the clustering-based outlier detection method provides the most balanced performance across all metrics, achieving a reasonable trade-off between precision and recall. With an accuracy of 91.91%, a precision of 72.97%, and a recall of 38.30%, it identifies a larger portion of true positives compared to both LOESS smoothing and distance-based methods.

While distance-based outlier detection shows a high precision of 87.61%, its recall is low (23.40%), meaning it is too conservative and misses a large number of true positives. This approach is effective at reducing false positives but at the cost of detecting fewer actual outstanding players.

LOESS smoothing, on the other hand, is less effective than both the clustering-based and distance-based methods, with lower precision and recall, as well as a relatively lower F1 score. This method seems to misclassify a considerable number of non-outstanding players as outstanding and misses many true positives.

In conclusion, while all methods have their merits, clustering-based outlier detection appears to be the most effective at balancing precision and recall in the context of identifying outstanding NBA players. Further improvements in precision and recall can be made by exploring hybrid models or tuning the existing methods to achieve a better trade-off between false positives and false negatives.

3.3. Game Prediction

3.3.1. Results for Game Prediction

The resulting mean classification accuracies for each machine learning model and their corresponding standard deviation can be found in Table 3, noting once again that the mean and standard deviation were calculated from five-fold cross validation.

Table 3: Table showing the different machine learning models used for game prediction, their resulting mean accuracies and corresponding standard deviations calculated over five folds during five-fold cross validation.

Machine Learning Model	Accuracy (%)
Random Forest Classifier	91.23 ± 0.57
Gaussian Naïve Bayes	70.97 ± 0.44
Support Vector Machine: linear kernel function	71.82 ± 0.35
Support Vector Machine: polynomial kernel function	74.55 ± 0.57
Support Vector Machine: Radial basis kernel function	76.82 ± 0.36
Stacking	89.05 ± 0.66

Because the standard deviation is <1% for each accuracy, we will not be showing the evaluation metrics and confusion matrices for all five folds when implementing the various machine learning models for game prediction. Instead, displayed in Table 4 are the classification *accuracy*, *precision*, *recall* and *F1-score* for each class on the fifth fold of five-fold cross validation when implementing each machine learning model in game prediction.

Table 4: Table showing the Accuracy, Precision, Recall and F1-Score for each class on the fifth fold of five-fold cross validation when implementing each machine learning model in game prediction.

Machine Learning Model	Accuracy (%)	Precision (%)		Recall (%)		F1-Score (%)	
		Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
Random Forest Classifier	**92**	**92**	**92**	**93**	**91**	**93**	**90**
Gaussian Naïve Bayes	71	71	73	82	58	76	65
Support Vector Machine: linear kernel function	71	72	71	79	62	75	66
Support Vector Machine: polynomial kernel function	75	74	78	85	63	79	69
Support Vector Machine: Radial basis kernel function	77	76	78	85	68	80	73
Stacking	89	90	88	90	88	90	88

The corresponding confusion matrices for the fifth fold of the five-fold cross validation when implementing the various machine learning models can be found in Figures 26-31.

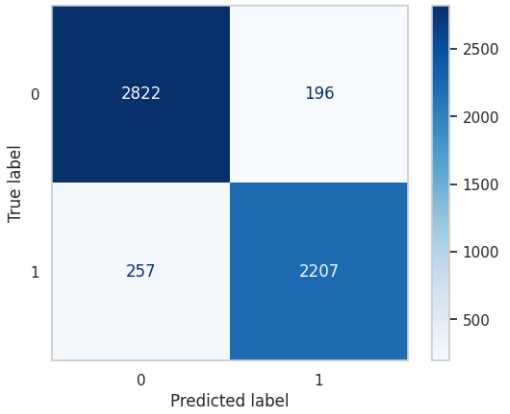

Figure 26. Confusion Matrix using Random Forest Classifier

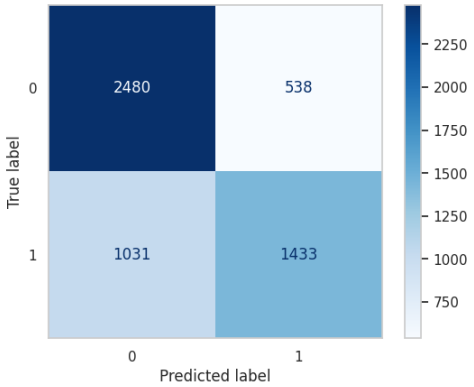

Figure 27. Confusion Matrix using Gaussian Naïve Bayes Classifier

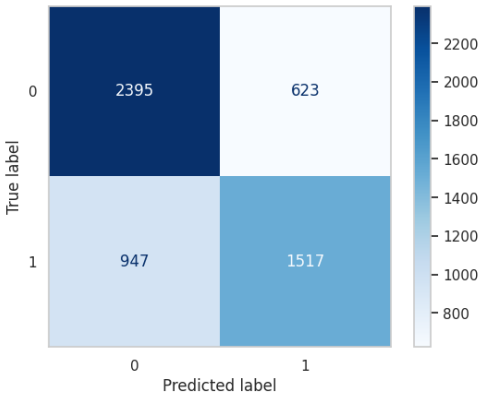

Figure 28. Confusion Matrix when using SVM with linear kernel function Classifier

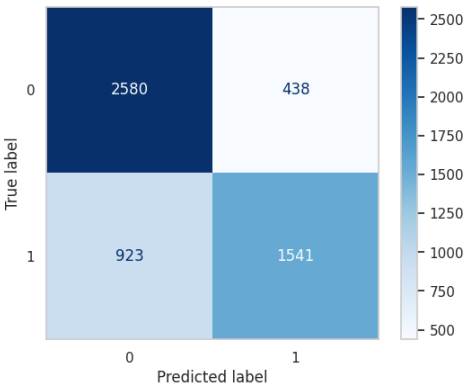

Figure 29. Confusion matrix when using SVM with polynomial kernel function

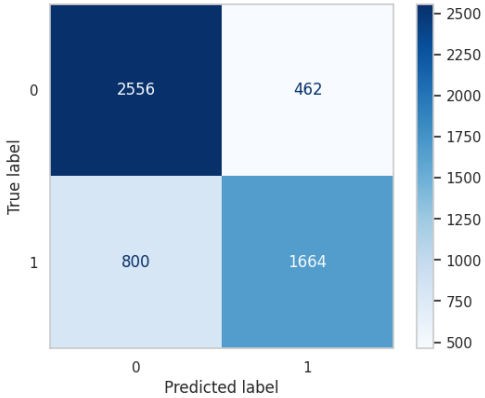

Figure 30. Confusion matrix when using SVM with radial basis function as kernel function

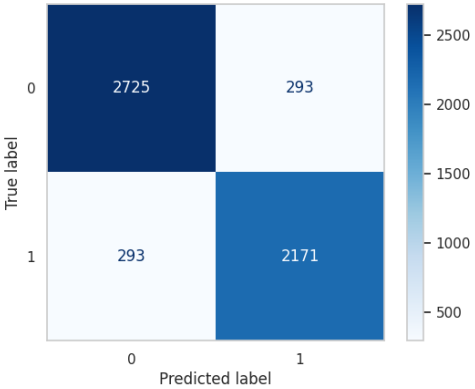

Figure 31. Confusion matrix for stacking model

3.3.2. Discussion for Game Prediction

The mean classification accuracies for each machine learning model had a low, <1%, standard deviation. This indicates that these machine learning models performed consistently across all five folds of the five-fold cross validation and that the PCA feature reduction technique employed has indeed worked.

Despite all the machine learning models showing consistency in their performance across the five-fold cross validation, some of the models did, most definitely, outperform the others. This can be noted by observing the mean accuracies for the three versions of the SVM. For an SVM where the kernel function is linear, the accuracy is 71.82 ± 0.35 %, where the kernel function is polynomial, it is 74.55 ± 0.57 % and where the kernel function is the Radial Basis function, the accuracy is 76.82 ± 0.36 %. From this we can observe that as the kernel function becomes more complex, the classification accuracy increases.

We also note the higher performing models by the high mean accuracy of 91.23 ± 0.57% for the Random Forest Classifier and the accuracy of 89.05 ± 0.66% for the Stacking algorithm. When comparing these results to the mean accuracies for the remaining machine learning models, one can observe the robustness of these two ensemble models in comparison to just using one machine learning model. As one can expect, the classification accuracies for the two ensemble models are substantially high, with the highest performing model being the Random Forest Classifier.

4 Conclusion

This project aimed to apply machine learning techniques to analyze NBA data from the 2004-2005 season, focusing on two primary objectives: identifying outstanding players through outlier detection and predicting game outcomes between teams. The results underscore the potential of machine learning in sports analytics, while also illustrating the unique strengths and limitations of various algorithms within this domain.

In the outlier detection task, three models—LOESS Smoothing, Distance-Based Outlier Detection, and Clustering-Based Outlier Detection—were evaluated on their ability to identify standout players. The analysis revealed that the Clustering-Based Outlier Detection method offered the best balance between precision and recall, achieving an accuracy of 91.91%, a precision of 72.97%, and a recall of 38.30%. This method excelled at identifying more true positives than both LOESS and Distance-Based methods, making it the most effective at distinguishing outstanding players with minimal misclassifications. While Distance-Based Outlier Detection achieved a high precision of 87.61%, its recall was significantly lower, resulting in a conservative approach that missed a substantial number of true outliers. Overall, Clustering-Based Outlier Detection demonstrated the strongest balance, suggesting that clustering approaches may be more suitable for identifying top-performing players in complex datasets with overlapping features.

For the game prediction task, various machine learning models, including Random Forest, Gaussian Naïve Bayes, Support Vector Machines (SVMs) with different kernels, and a Stacking model, were implemented and evaluated using five-fold cross-validation. We note that when using the SVM machine learning model for game prediction, as the kernel function

becomes more complex, the classification accuracy also increases. In implementing various machine learning models for game prediction, we also note that the classification accuracies for the two ensemble models are substantially high, with the highest performing model being the Random Forest Classifier. The Random Forest Classifier achieved a mean accuracy of 91.23% with a low standard deviation, followed closely by the Stacking algorithm with an accuracy of 89.05%. These ensemble models consistently outperformed single algorithms like SVM and Naïve Bayes, demonstrating the robustness of ensemble methods in complex prediction tasks. The superior performance of the Random Forest model indicates its effectiveness in handling a large variety of features within the dataset, while the consistent performance across folds suggests that feature selection techniques, including PCA, effectively enhanced the predictive stability of each model.

In summary, this project successfully applied machine learning techniques to achieve two distinct but complementary goals in sports analytics. The clustering-based approach for outlier detection and the Random Forest model for game prediction emerged as the most effective techniques, offering strong performance across multiple evaluation metrics. Future work could focus on integrating more advanced feature engineering techniques or exploring hybrid models to further improve precision and recall for both tasks. Additionally, expanding the analysis to incorporate recent seasons or broader datasets could provide further insights, potentially enhancing the applicability and generalizability of these machine learning models in NBA analytics.

5 References

[1] Gustavo and M. C. Monard, "A Study of K-Nearest Neighbour as an Imputation Method.," *Soft Computing Systems - Design, Management and Applications, HIS 2002, December 1-4, 2002, Santiago, Chile*, vol. 30, pp. 251–260, Jan. 2002, Available: https://www.researchgate.net/publication/220981745_A_Study_of_K-Nearest_Neighbour_as_an_Imputation_Method

[2] L. B. V. de Amorim, G. D. C. Cavalcanti, and R. M. O. Cruz, "The choice of scaling technique matters for classification performance," Applied Soft Computing, vol. 133, p. 109924, Jan. 2023, doi: https://doi.org/10.1016/j.asoc.2022.109924.

[3] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder, "The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data," Applications and Innovations in Intelligent Systems XIII, pp. 209–222, doi: https://doi.org/10.1007/1-84628-224-1_16.

[4] Oliver, Dean. *Basketball on paper: rules and tools for performance analysis*. U of Nebraska Press, 2011.

[5] Harris, Curtis Matthew. *Hardwood Revolution: The NBA's Growth and Player Revolt, 1950–1976*. American University, 2021.

[6] I. T. Jolliffe, *Principal Component Analysis*, https://link.springer.com/book/10.1007/b98835, Springer New York, NY, 2002.

[7] P. N. Stuart Russel, Artificial Intelligence: A Modern Approach, 2010.

[8] S. Fragala, and J. R. Stout, "Performance changes in NBA basketball players vary in starters vs. nonstarters over a competitive season," *The Journal of Strength & Conditioning Research*, vol. 27, no. 3, pp. 611-615, 2013.

[9] A. Franks, A. Miller, L. Bornn, and K. Goldsberry, "Counterpoints: Advanced defensive metrics for nba basketball."
[10] J. Courel-Ibáñez, A. P. McRobert, E. Ortega Toro, and D. Cárdenas Vélez, “Inside game effectiveness in NBA basketball: Analysis of collective interactions,” *Kinesiology,* vol. 50, no. 2., pp. 218-227, 2018.
[11] A. Barai, and L. Dey, “Outlier detection and removal algorithm in k-means and hierarchical clustering,” 2017.
[12] D. Muhr, M. Affenzeller, and J. Küng, “A probabilistic transformation of distance-based outliers,” *Machine Learning and Knowledge Extraction,* vol. 5, no. 3, pp. 782-802, 2023.
[13] L.-C. Chen, T.-T. Kuo, W.-C. Lai, S.-D. Lin, and C.-H. Tsai, "Prediction-based outlier detection with explanations." pp. 44-49.