DATA SCIENCE MASTERS
DISSERTATION: SPORTS ANALYTICS

CANDIDATE NUMBER: 277915

WORD COUNT – 14,426

Abstract

This dissertation explores the application of sports analytics in predicting NBA game outcomes through machine learning models. By leveraging data from 2021/2022, 2022/2023 and 2023/2024 NBA seasons, the study employs logistic regression, random forest, and multilayer perceptron models to evaluate player and team performance. The research emphasizes feature engineering, focusing on critical box score metrics that influence game results. The effectiveness of the models is assessed using accuracy, precision, recall, and F1-score metrics, with hyperparameter tuning to enhance performance. The findings demonstrate that advanced data analytics can significantly improve predictive accuracy, providing valuable insights for teams, coaches, fans, and analysts. The study highlights the economic and strategic benefits of integrating data-driven decision-making in sports, contributing to the evolving landscape of sports analytics.

Table of Contents

INTRODUCTION

For many people, participating in sports is a major undertaking in life. One explanation for this is the fact that many people do sports as an exercise regimen and to enhance their quality of life. Another factor is that following and watching professional sports is a popular pastime for both young people and adults. Many people watch sports on television every day, across the world with subscription to different TV channels; some watch with phones or gadgets and others watch sports in different stadium all the over the world. This could be locally or may involve international travel and logistics. Sports fans are also frequently quite interested in analysing coaches' choices, contrasting players' stats, and projecting game results as well as the standings of players and teams competing in tournaments.

The Union of European Football Associations (UEFA) champions league and World Cup in football, World Championships in basketball and swimming, NBA (National Basketball Association) in basketball, the Olympic Games, are a few of the most closely watched sporting events globally. There are other sport competitions in motor racing, boxing, hockey, gymnastics, cricket, golf, baseball, volleyball, table tennis, tennis, badminton, handball etc. Athletes' salaries, broadcasting licence fees, advertising, and game ticket prices are just a few of the billions of dollars that go into the sports sector [1].

In all these, there is need to record players performance and teams' performance and use necessary data science techniques and machine learning algorithms to make predictions. This is a serious business, worth billions of dollars for clubs and bookies all the world. Sports offer rich opportunities for analytics and research beyond just big data, primarily due to its clear rules, expert players, diverse formats, and global reach. The distinctiveness of various sports allows for the examination of different performance aspects, whether individual or team oriented. Furthermore, sports provide a universal platform reflecting broader human behaviour patterns, making them valuable for understanding human behaviour in general. Companies specializing in sports performance measurement and data provision cater to various stakeholders such as clubs, associations, broadcasters, and academic researchers, offering tailored information packages.

This research aims to compile essential sports data to make informed decisions about basketball games, teams, and players. By gathering and analysing comprehensive analytics, the study seeks to develop cutting-edge performance indicators. These indicators will enable coaches, managers, and analysts to assess player performance, optimize team strategies, and enhance overall game

tactics. The integration of advanced data analytics will facilitate more accurate predictions, better injury prevention, and improved player development. Ultimately, this research aspires to transform how decisions are made in basketball, leveraging data-driven insights to achieve competitive advantages and elevate the sport's overall performance standards. The aim of data analytics and mining is to uncover hidden structures [2, 3].

Sports analytics offers substantial benefits for educating future players, technical personnel, and management. Clubs can leverage data analytics to forecast squad composition, enabling strategic recruitment and development plans. By analysing performance metrics, teams can optimize tactics, enhancing gameplay and competitive edge. Furthermore, predictive analytics helps in identifying and mitigating potential risks, ensuring preparedness for unforeseen situations. This data-driven approach supports informed decision-making across all levels, fostering a culture of continuous improvement. Consequently, sports organizations can enhance performance, achieve long-term success, and maintain a sustainable competitive advantage through the effective application of sports analytics [4, 5]. Sports analytics enables governing bodies to generate various odds for different outcomes in sports events. These include odds for team A or B to win a match, the likelihood of a player scoring or assisting, and probabilities for a team to score over or under a certain number of goals or points. By analysing extensive data, organizations can predict game outcomes and publish these predictions on their websites. This information is particularly valuable to bettors who use it to inform their stakes, aiming to profit from accurate predictions. The detailed odds and probabilities help bettors make more informed decisions by understanding potential outcomes better. However, while these analytics-driven predictions can sometimes lead to significant gains for bettors, they are not foolproof. The inherent uncertainty in sports means that predictions can occasionally fail, leading to losses. Despite this, the use of sports analytics continues to grow, providing valuable insights and enhancing the betting experience for many.

Research Objectives

In this research, the main objectives are:

- Analysing individual performance of players in NBA using 2021/2022, 2022/2023 and 2023/2024 seasons as case study.
- To analyse teams' performance in NBA using 2021/2022, 2022/2023 and 2023/2024 seasons.
- To use algorithms in machine learning and data science techniques to build models to predict teams' performance in a competition, that is, win or lose.
- To evaluate which models and features that would be useful in developing accurate predictions.
- To evaluate which models that could reach more than 66% accuracy in NBA.
- To make various decisions and policies from the outcome of the predictions for clubs, fans, and bookmakers.

Scope of Study

The scope of this research is to apply machine learning and data science methods to sports analytics, with an emphasis on NBA basketball. This involves using diverse data science techniques to evaluate and forecast the performance of teams and players, thereby offering significant perspectives for decision-making by organisations, supporters, and bettors. The scope is explained in more depth below:

1. **Data Management and Collection**: With an emphasis on 2021/2022, 2022/2023 and 2023/2024 NBA seasons, the research started with gathering extensive datasets from dependable sources like www.basketball-reference.com. Detailed statistics such as field goal percentages, rebounding, offensive and defensive ratings, and more are included in the datasets. The dataset is suitably cleansed, processed, and organised to make sure it can be analysed.
2. **Exploratory Data Analysis (EDA):** Patterns and trends within the dataset were discovered through the process of exploratory data analysis (EDA). To comprehend key performance metrics and their distributions, pictures and statistical summaries were included. In addition, EDA assists in locating any irregularities or missing data that need attention.
3. **Feature engineering:** From the current data, further metrics and features pertinent to the prediction models were obtained. This involves figuring

out shot choices, defence tactics, player efficiency ratings, and average statistics during a recent stretch of games.

4. **Prediction Modelling:** Using ML techniques to create and assess prediction models is the focus of this research. This was achieved by using models like Logistic Regression, Random Forest, and Multilinear Perceptron. To forecast game results (win or lose) and other performance indicators.
5. **Model Evaluation and Optimization:** The models' precision and capacity to forecast were assessed accordingly. The models were evaluated and refined using methods such as cross-validation, gridsearchCV and other hyperparameter tuning, and performance metrics.
6. **Decision Support and Policy Implications:** Actionable insights are made available to different stakeholders based on the prediction model results. This involves giving fans data-driven forecasts, supporting bookies with odds setting, and assisting teams in developing their strategies.
7. **Documentation, and Dissemination:** Complete documentation of the research findings is provided, along with in-depth explanations of the methods, outcomes, and implications. Recommendations for further study and possible uses in different sports or situations are included in the report.

By covering these areas, the intension of the research is to contribute significantly to sports analytics, leveraging advanced data science techniques to enhance the understanding and prediction of basketball performance.

Justification of the Study

The increasing significance of data-driven decision-making in sports, especially in professional basketball, justifies this study. With its vast data availability and international fan base, the NBA offers an excellent setting for using cutting-edge data science and methods in machine learning. Creating prediction models that can accurately anticipate game outcomes by evaluating player and team performance is one of the goals. Significant ramifications of these models exist for different stakeholders:

1. **Teams and Coaches**: Improved forecasts help maximise player rotations, game plans, and team strategies, which can improve performance and provide an advantage over competitors.

2. **Fans and Analysts**: Data-driven insights can deepen fan engagement and understanding, enhancing the overall viewing experience, and providing robust material for analysts.
3. **Bookmakers and Bettors**: Precise forecasts aid in the establishment of more dependable odds, which benefits bookmakers in managing risk and bettors in making well-informed choices.
4. **Academic Contribution**: This study adds to already existing body of knowledge in sports analytics, showcasing the use of ML (Machine Learning) in the real-world.

The high potential economic and performance benefits, together with the academic value, strongly justify the need for this study, highlighting its relevance and impact in the evolving landscape of sports analytics.

Structure of the Thesis

The thesis is structured in the following ways:

Introduction:

Introduction provides a summary of the thesis, emphasising the role that data analytics plays in tackling major issues in sports with emphasis on basketball. The state of sports analytics in modern day is briefly covered, including business models, technologies, and the effects of data analysis on basketball. Also, an overview of the thesis's rationale, objectives, and scope is presented.

Chapter 1 - Literature Review:

This chapter shows an in-depth review of the relevant literature. It covers several key topics including the assessment of team strength, prediction of game outcomes, analysis of team strength, and a comparative study of the NBA. This literature review is a guide to work previously done in sports analytics and provides backup for the models and why they were selected and utilized.

Chapter 2 – Methodology:

This chapter shows details the research methodology, focusing on evaluating team and player performance through box score and advanced statistics. Emphasis on feature engineering captures game dynamics, leading to robust prediction models. The chapter covers data preparation, exploratory analysis, feature engineering, and model selection, ensuring accurate predictions.

Chapter 3 - Result and Discussion:

This section explains the results and outcomes from all the models used in the study, evaluating their performance in predicting game outcomes. It highlights the input features that provided the best performance, identifying the most influential factors in game result prediction. Through detailed analysis, this chapter offers valuable insights and implications, enhancing the understanding of basketball game predictions.

Chapter 4 - Conclusion and Limitations:

This chapter integrates the descriptive and predictive analyses, drawing conclusions from the research. It compares different models and highlights key findings and limitations of the research.

Chapter 5 - Recommendations:

This chapter displays recommendations from the research, and possible areas for improvement in further research.

CHAPTER ONE

LITERATURE REVIEW

This project focus on sports analytics especially in NBA. Before moving into the problem, it is good to understand the general concept of sports analytics, how data from players and teams are collected, different data science algorithms and machine learning techniques used to arrive at meaningful decision making. This involves looking into previous work (or publications) done in this area in the past years.

All major organizations in sports and professional teams utilize sports analytics to build their rosters, enhance individual player performance, and identify issues that may be challenging for coaching staff to detect. In this rapidly evolving field, continuous innovations have made it increasingly important and indispensable for coaches, staff, and their respective teams [1]. Basing decision-making on sports analytics and predictive analytics gives teams and organizations confidence that their actions are grounded in valid data. Moreover, with machine learning (ML) and data mining (DM) techniques, the development of predictive analytics enables researchers to expand their experiments with sports analytics, propose new methods, and assess their results [29]

Sports analytics started a long time ago and modern research are providing solutions to existing problems or recommending new ways of doing things. As seen in figure 1 below, sports analytics has four main framework which are information gathering, data management, data analysis and decision making. In information gathering, multiple resources like qualitative and quantitative data are used. Data management involves standardization, centralization, and integration of data. Data analysis involves using different metrics to get useful information. While in decision making, the outcome of data analysis is utilized in making quality decisions [1].

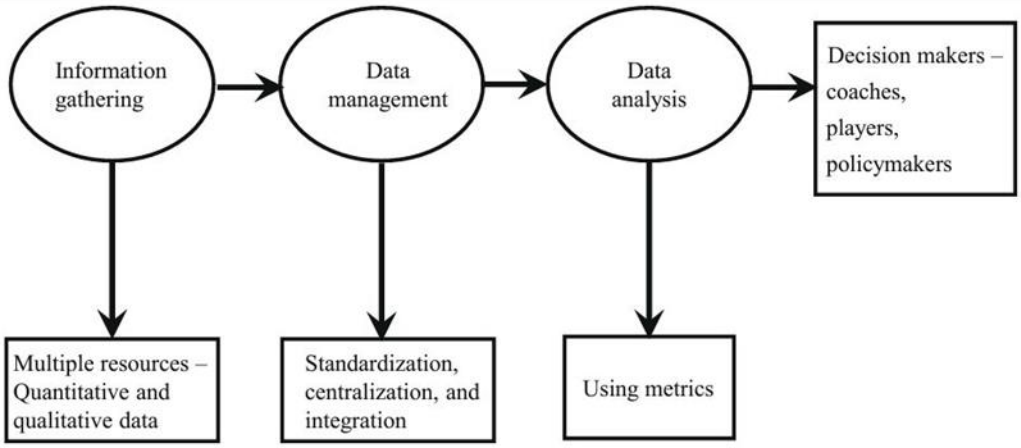


Figure 1: Framework for sports analytics [1]

While machine learning plays important role in various aspects of sports analytics, there is still a need to improve the effectiveness of the models developed using this technology. This section focuses on the machine learning methods used to predict the results of basketball games. Cao [8] developed a predictive model utilizing historical NBA game data with a Naïve Bayes Classifier. The classification accuracy for the test dataset was approximately 65.82%. The framework utilized various characteristics and features associated to NBA games, such as player statistics, opponent statistics, and starting line-up information, all collected from the NBA's official website. These features were stored in a database, pre-processed, and cleaned to prepare them for model derivation using machine learning algorithms. The research was carried out using several techniques for learning, including Support Vector Machine (SVM) [9], logistic regression [10], Naive Bayes [11], and artificial neural networks (ANN) [12], on the collected dataset to develop a sports analytics model for predicting game results. The findings indicated that logistic regression models outperformed the other algorithms considered.

Lorena et al. [25] explored the use of machine learning classifiers. Other researchers utilized various machine learning methods, including k-nearest neighbor, SVM, decision trees, multivariate linear regression, and Naive Bayes [13, 14, 15, 16, 17], to forecast NBA game results. They gathered data with 141 variables related to NBA games from 2009 to 2010 to assess these machine learning methods. The dataset included variables such as fouls, three-pointers, free throws, blocked shots, field goals, home and away status, current streak, number of wins, number of losses, and more. Other experiments were carried out, and the findings indicated that machine learning models could achieve up to 67% accuracy in predicting classifying NBA match outcomes. Also, in the

past, researchers used basic statistics on past game features to rank teams and predict home team winning probabilities [28].

Lieder [18] examined the significant features associated with NBA games to develop a predictive model for game outcomes. The study employed machine learning algorithms such as Logistic Regression, Linear Regression, and Artificial Neural Networks (ANN) to construct predictive models and assess game results. Data from the 2014 NBA season were utilized for training, testing and validating the models. The findings revealed that Logistic Regression achieved an accuracy of nearly 70%.

1.1 PREDICTION MODELS

The decision-making process in data analytics of basketball is achieved using data mining and machine learning algorithms. Previous reports used classification method or clustering, while some used both. In recent time, the most widely used algorithms are:

a) **Linear and Logistic Regression**: Statistically, the aim of linear regression is to predict the value of the dependent variable by modelling the relationship between one or more independent factors and a continuous dependent variable. On the other hand, logistic regression is a statistical method that shows the relationship between one or more independent variables and a categorical variable that is dependent. It is frequently utilised in binary classification jobs to estimate the likelihood of an event happening [4, 5, 6].

A simple mathematical equation or expression for Linear regression is

$Y = a + bX$,

where Y = Dependent variable

X = Independent variable

b = Slope of the line

a = y-intercept

To obtain the linear regression by hand, it is important to find the values of ‘a’ and ‘b’; and substituting them in the slope formula.

Whereas logistic regression uses logistic function in mathematics as the equation between x and y; thereby mapping y as a sigmoid of x.

$$f(x) = 1/1 + e^{-x}.$$

b. **Support Vector Machines (SVM):** These are machine learning algorithms that discovers the best hyperplane with the longest margin or difference between classes to perform regression and classification tasks [7]. It is a sophisticated algorithm capable of delivering high accuracy. It also ensures theoretical safeguards against overfitting. By using a suitable kernel, it can effectively handle data that is not separable in the original feature space. The core idea of the algorithm is to maximize the minimum distance between the hyperplane and the nearest data point [19].

c. **K-Nearest Neighbor (KNN)**

KNN is a technique for classification that places an unlabelled sample point in the class of the point that is closest to it among a group of previously labelled point [20]. This is a less effective, simple lazy learning strategy. The performance of it depends on selecting a suitable value for "k."

However, there is no systematic way to select ‘k’ other than using computationally expensive methods like cross-validation. The algorithm is negatively impacted by noise and is sensitive to irrelevant features. Its performance also changes with the quantity of the dataset because every piece of data needs to be reviewed [21, 22].

d. **Decision Trees**

Decision Trees are not difficult to interpret and explain, capable of handling interactions between features, and can work with various data types (nominal, numeric, textual), missing values, and redundant attributes. They can accommodate missing values and redundant attributes, while also being resilient to noise. Also, they generalize effectively and offer strong performance with relatively minimal computational resources. However, they struggle with high-dimensional data, and building the tree can be time-consuming. They employ a divide-and-conquer strategy, functioning effectively with a small number of important characteristics but badly with numerous intricate relationships. Errors can propagate through the tree, which becomes problematic as the number of classes increases [23, 24].

e. **Random Forests**

Random Forest trains multiple decision trees and predicts the majority class from all trees [25]. Random Forests, often outperforming SVMs, excel in classification problems due to their speed, scalability, noise robustness, lack of overfitting, and ease of interpretation and visualization, with no parameters to manage. However, they slow down with an increasing number of trees during real-time prediction. Improvements include reducing tree correlation and using

various attribute evaluation measures. Another method estimates the average margin of similar instances, discards trees with negative margins, and weights tree votes by the margin [26].

f. **Multilayer Perceptron**

This is an aspect of artificial neural network composed of multiple layers of perceptron. In an MLP, a perceptron normally has several input layers or connections, as seen in figure 2 below; each having a weight assigned to it. It generates a single output by adding up all the weighted inputs that are fed through an activation function. MLPs are effective tools for machine learning tasks like regression and classification because they can learn intricate patterns and correlations inside data by stacking numerous layers of these interconnected perceptrons and modifying the weights during training [27, 8].

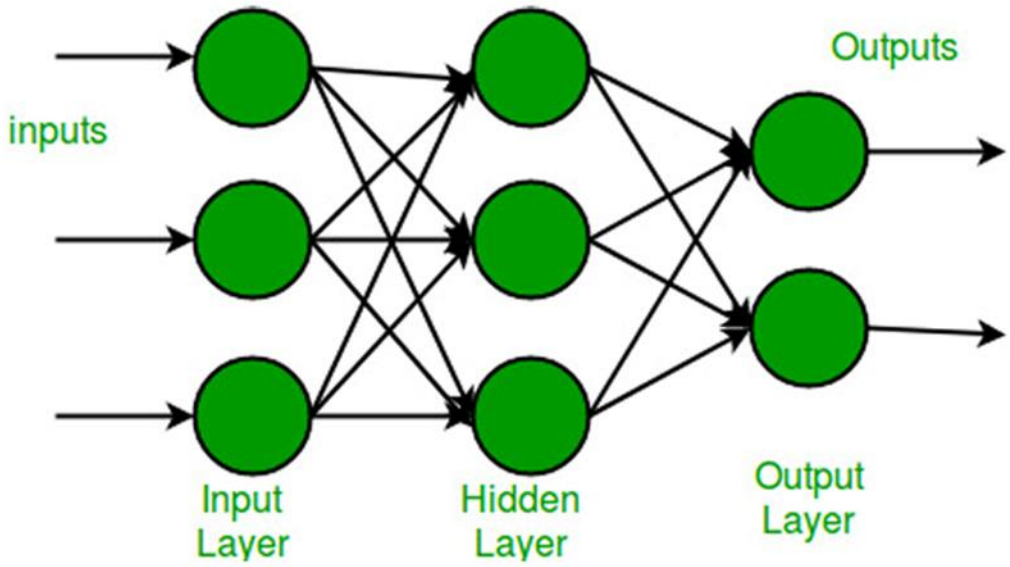


Fig 2: Multilayer Perceptron [30]

CHAPTER TWO

METHODOLOGY

The methodology used in this study involves a comprehensive analysis of existing performance analytics utilized in NBA basketball. This study concentrated on advanced measures linked to the game that include offence, defence, total performance, other aspects, and performance ratings that have been published in the literature.

Furthermore, the methodology aims to enhance understanding of basketball analytics, mitigate uncertainty in current and future events, and quantify player performance attributes to improve forecasting accuracy. Notwithstanding the difficulties presented by the complexity of sports data, this methodology leverages analytics and thorough analyses to extract valuable insights, thus enabling informed decision-making for teams and betting companies alike.

2.1 Dataset

There are lots of datasets about NBA in different websites like Kaggle. For this project, the dataset was obtained from www.basketballreference.com. The website has a lot of data sets for all NBA teams (both home and away performances) for the recent years. The dataset has important features like field goal percentage, 3-point field goals and percentage, 2-point field goals and percentage, total rebounds, offensive and defensive ratings etc. The website is a comprehensive resource for basketball statistics and history. Launched in 2004 by Justin Kubatko, it offers extensive data on NBA (National Basketball Association), WNBA (Women National Basketball Association), and international basketball leagues. The site includes traditional stats like points and rebounds, as well as advanced metrics such as Player Efficiency Rating (PER) and Win Shares. The easiest way to use this dataset is to download as Microsoft excel file.

- The dataset in Microsoft excel format was converted into CSV file.
- The CSV file was read into python notebook.
- The dataset was cleaned, wrangled, and NaNs (Not a Number) removed.
- Exploratory Data Analysis (EDA) was carried out on the dataset.
- Additional metrics were added to the dataset that are helpful to the machine learning model to understand and predict outcomes and trends from the given data.

- Models and features were carefully examined to determine the ones that would be useful for accurate predictions.
- Prediction models (using machine learning) were used to predict outcome of NBA matches, that is, wins or loses.
- Prediction models were evaluated to see the ones that could reach more than 66% accuracy.
- Finally, from the predictions, necessary decisions that will be useful to clubs, fans, pundits, and bookmakers were made.

2.1.1 Player Statistics

Player statistics in the NBA are detailed metrics that track individual player performance in various aspects of the game. These statistics show a comprehensive view of a player's contributions, and/or effectiveness. Here are the primary categories typically included in player statistics:

-Age: The player's age at the time of the season.

-Tm: The player’s team during the season.

-G: The number of games the player participated in.

-MP: Minutes played per game on average.

-PER (Player Efficiency Rating): A rating of a player's per-minute performance.

-TS% (True Shooting Percentage): A measure of shooting efficiency that measures three-point field goals, field goals and free throws.

-3PAr (Three-Point Attempt Rate): The proportion of field goal attempts that are three-pointers.

-FTr (Free Throw Rate): Attempted free throw per field goal attempt.

-ORB% (Offensive Rebound Percentage): an estimation of the proportion of offensive rebounds that a player can grab while they are on the court.

-DRB% (Defensive Rebound Percentage): an estimation of the proportion of defensive rebounds that a player can grab while they are on the court.

-TRB% (Total Rebound Percentage): a calculation used to determine the proportion of total rebounds that a player can grab while on the court.

-AST% (Assist Percentage): an estimation of the proportion of teammates' field goals that a player helped while they were on the court.

-STL% (Steal Percentage): an estimate of the proportion of opponent possessions that the player steals while they are on the floor.

-BLK% (Block Percentage): a rough estimate of the proportion of opponent two-point field goals that the player blocks while on the court.

-TOV% (Turnover Percentage): Calculated as the player's turnover rate per 100 plays.

-USG% (Usage Percentage): a calculation that estimates the proportion of team plays a player uses while on the court.

-OWS (Offensive Win Shares): an estimation of how many victories a player has helped bring about because of their offensive play.

-DWS (Defensive Win Shares): An approximation of how many victories a player has helped with because of their defence.

-WS (Win Shares): an estimation of the quantity of victories a player has contributed.

-WS/48 (Win Shares Per 48 Minutes): Win Shares every forty-eight minutes, with a league average of approximately 100.

-OBPM (Offensive Box Plus/Minus): An average team can be represented by a box score approximate of the offensive points per 100 possessions that a player is worth more than the league average.

-DBPM (Defensive Box Plus/Minus): An average team's worth of defensive points per 100 possessions, calculated from a player's box score estimate above the league average.

-BPM (Box Plus/Minus): An average team's box score represents the points per 100 possessions that a player is worth more than the league average player.

-VORP (Value Over Replacement Player): a per-minute estimation of a player's box score based on comparison to a replacement-level player.

The website has a rule stopping people from downloading more than five (5) hundred player statistics at a time. So, few rows and columns were downloaded, opened, and the remaining dataset were copied and pasted into the file. After that, the file was converted into CSV to enable reading and performing other data processing on Jupiter notebook. The three (3) files are namely:

-Advanced player stats for 2021-2022 season

-Advanced player stats for 2022-2022 season, and

-Advanced player stats for 2023-2024 season, respectively.

On the Jupiter notebook, a code was written to merge these three (3) files into one data frame called merged_player_stats_2021_2024.

2.1.2 Team Box Score Statistics

A team box score includes aggregated statistics for the entire team and provides an overall picture of how the team performed. The primary categories typically included in a team box score are:

-Points (PTS): Total points scored by the team.

-Field Goals (FG): Overall number of field goals made by the team.

-Field Goals Attempted (FGA): Total number of field goals attempted by the team.

-Field Goal Percentage (FG%): The percentage of field goals made from those attempted by the team.

-Three-Point Field Goals (3P): Overall number of three-point field goals made by the team.

-Three-Point Field Goals Attempt (3PA): Total number of three-point field goals attempted by the team.

-Three-Point Field Goal Percentage (3P%): The percentage of three-point field goals made from those attempted by the team.

-Free Throws Made (FT): Team's total free throws.

-Free Throws Attempted (FTA): Team's attempted free throws

-Free Throw Percentage (FT%): The percentage of free throws made from those attempted by the team.

-Rebounds (REB): Team's rebound.

-Offensive Rebounds (OREB): Number of team's offensive rebounds.

-Defensive Rebounds (DREB): Team's total rebound

-Assists (AST): Team's assists.

-Steals (STL): Team's steal.

-Blocks (BLK): Total number of blocked shots by the team.

-Turnovers (TO): Team's turnover.

-Personal Fouls (PF): Team's personal fouls.

On the website containing these data, there are data for home teams and for visitor teams. The data set was gotten by downloading home team statistics files in excel sheet for 2021-2022, 2022-2023, 2023-2024 and visitor team statistics for 2021-2022, 2022-2023, and 2023-2024 seasons. The excel files were opened and converted directly to csv files to enable python libraries to read them accurately.

The code for merging these files were written on Jupiter notebook; thereby having combined team statistics for further data processing.

2.1.3. Impact of Box score features in making predictions

Box score features are among the most critical components in making predictions in basketball, especially in the NBA. These features provide a detailed snapshot of player and team performance through various statistical categories such as points, assists, rebounds, blocks, steals and shooting percentages. The comprehensive nature of box score data lays a solid foundation for effective feature engineering, which is the process of making new features from the existing data to improve model performance.

By selecting the right box score features, analysts can identify the most relevant statistics that significantly influence game outcomes. For example, factors such as a team's field goal percentage, defensive rebounds, and turnovers can be strong predictors of a game's result. Proper feature selection helps in isolating these impactful variables, leading to more accurate predictions.

Moreover, box score features facilitate the identification of trends and patterns within the data. For instance, examining the correlation between assists and winning percentages can reveal the importance of teamwork and ball movement in successful gameplay. These insights are invaluable for building robust predictive models.

When integrated into predictive modelling, well-chosen box score features enhance the model's accuracy and reliability. They enable analysts to make informed decisions, such as optimizing player rotations, strategizing game plans, and making informed betting choices. In essence, box score features are the backbone of effective NBA predictions, driving the analytical process toward achieving better outcomes and supporting data-driven decision-making.

2.1.4 Ethical Considerations and Limitations

In the data collection process, strict ethical guidelines were adhered to, ensuring compliance with the terms of use and copyright restrictions set by the data

sources. Special consideration and compliance were given to the General Data Protection Regulation (GDPR). Despite efforts to obtain comprehensive data, certain limitations were acknowledged that may affect the overall quality and completeness of dataset.

Firstly, specific game-related variables may be absent, and this may potentially impact the depth and accuracy of the analysis.

Secondly, the dynamic nature of online data sources poses a challenge to the consistency of available data. Online data sources frequently update, leading to discrepancies in data collected over different periods, which can impact the reliability of the analysis and predictive modelling.

Additionally, robust data validation techniques were employed to identify and rectify inconsistencies or errors in the collected data, enhancing the overall quality and integrity of the dataset.

2.2. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) for NBA prediction involves investigating datasets to summarize their main characteristics, often with visual methods. EDA is crucial as it helps to spot anomalies, uncover patterns, and check assumptions.

For NBA prediction, EDA began with data cleaning, ensuring that the dataset is free from inconsistencies and missing values. Next, descriptive statistics such as mean, median, and standard deviation are calculated for key variables like points, rebounds, assists, and player efficiency ratings. This step helped in understanding the central tendency and variability within the data.

Visualization techniques, including histograms, box plots, and scatter plots, are then employed. For instance, plots can reveal relationships between variables, such as the correlation between a team's three-point shooting percentage and their win rate. Box plots can show the distribution of player performance metrics, highlighting outliers.

Heatmaps are useful for visualizing correlation matrices, indicating how different variables interact. This can help identify which statistics are most predictive of game outcomes. Time-series analysis may also be conducted to observe performance trends over seasons.

Ultimately, EDA helped in feature selection and engineering, identifying the most relevant variables for predictive modelling. It provided insights that guide the development of robust, data-driven NBA prediction models.

2.2.1 Data Overview and Summary Statistics

	count	mean	std	min	25%	50%	75%	max
Rk	180.0	15.500000	8.679585	1.000	8.00000	15.500	23.000	30.000
G	180.0	82.000000	0.000000	82.000	82.00000	82.000	82.000	82.000
MP	180.0	241.491111	0.786001	240.300	240.90000	241.500	242.100	243.700
FG	180.0	41.591111	1.715695	37.500	40.50000	41.600	42.825	47.000
FGA	180.0	88.435000	2.364814	82.300	86.77500	88.400	90.200	94.400
FG%	180.0	0.470322	0.015029	0.430	0.46200	0.470	0.480	0.507
3P	180.0	12.538333	1.143027	10.400	11.70000	12.500	13.100	16.600
3PA	180.0	34.830556	2.735104	28.800	32.70000	34.450	36.700	43.200
3P%	180.0	0.359894	0.013801	0.323	0.35075	0.360	0.369	0.395
2P	180.0	29.055556	1.884462	24.500	27.80000	29.000	30.400	33.900
2PA	180.0	53.602222	3.290191	43.300	51.47500	53.700	55.800	61.800
2P%	180.0	0.542189	0.019281	0.494	0.52875	0.541	0.557	0.589
FT	180.0	17.450000	1.453584	13.300	16.37500	17.450	18.500	21.000
FTA	180.0	22.370000	1.769351	17.300	21.10000	22.300	23.625	26.600
FT%	180.0	0.780239	0.020330	0.713	0.76800	0.780	0.792	0.835
ORB	180.0	10.438333	1.032612	7.600	9.70000	10.400	11.000	14.100
DRB	180.0	33.368889	1.418406	30.100	32.40000	33.400	34.200	37.500
TRB	180.0	43.808333	1.781465	38.800	42.67500	43.800	44.900	49.200
AST	180.0	25.548333	1.748726	21.600	24.35000	25.500	26.600	30.800
STL	180.0	7.465556	0.696305	6.000	7.00000	7.400	7.900	9.800
BLK	180.0	4.837222	0.720493	3.000	4.40000	4.800	5.200	6.600
TOV	180.0	13.819444	1.060099	11.700	13.07500	13.750	14.500	16.800
PF	180.0	19.450000	1.149083	15.600	18.60000	19.600	20.300	22.100
PTS	180.0	113.169444	4.083337	103.700	110.40000	113.100	115.825	123.300
Year	180.0	2022.000000	0.818774	2021.000	2021.00000	2022.000	2023.000	2023.000

Table 1. Summary Statistics of combined_team_stats_cleaned Dataframe

The summary statistics displayed in table1 above provided a comprehensive overview of the dataset combined_stats_cleaned. Here's a detailed breakdown of what each statistic represents and how it was obtained:

-Count: The total count of non-missing values for each column. In this dataset, each column has 180 observations.

-Mean: The average value of the data in each column. It is determined by adding all the values together and dividing by the number of values.

-Standard Deviation (std): Measures the spread of the data using mean as reference. A higher standard deviation indicates more variability.

-Minimum (min): The smallest value in each column.

Quartiles

25% (Q1): The first quartile, also known as the lower quartile, represents the 25th percentile of the data. It indicates that 25% of the data points fall below this value.

50% (Median): The median value, which is the 50th percentile of the data. It divides the dataset into two equal halves.

75% (Q3): The third quartile, also known as the upper quartile, represents the 75th percentile of the data. It indicates that 75% of the data points fall below this value.

Maximum (max)

Maximum (max): The largest value in each column.

Interpretation of Summary Statistics

Consistency: For some columns, such as G (Games), all values are consistent (82 games per season), leading to a standard deviation of 0.

Variability: Other columns, such as PTS (Points) and AST (Assists), show significant variability, as indicated by their standard deviations. This variability provides insights into the performance differences between teams.

Central Tendency: The mean, median, and quartiles give us an insight of the central tendency of the data. For instance, the mean and median PTS are close (113.17 and 113.1), suggesting a relatively symmetrical distribution of points scored.

Skewness: If the mean is greater than the median, the data might be right-skewed (positively skewed), and if the mean is less than the median, the data

might be left-skewed (negatively skewed). For instance, PTS mean (113.17) and median (113.1) are very close, indicating a symmetrical distribution.

2.2.2 Correlation Analysis

2.2.2.1 Correlation of Team Statistics

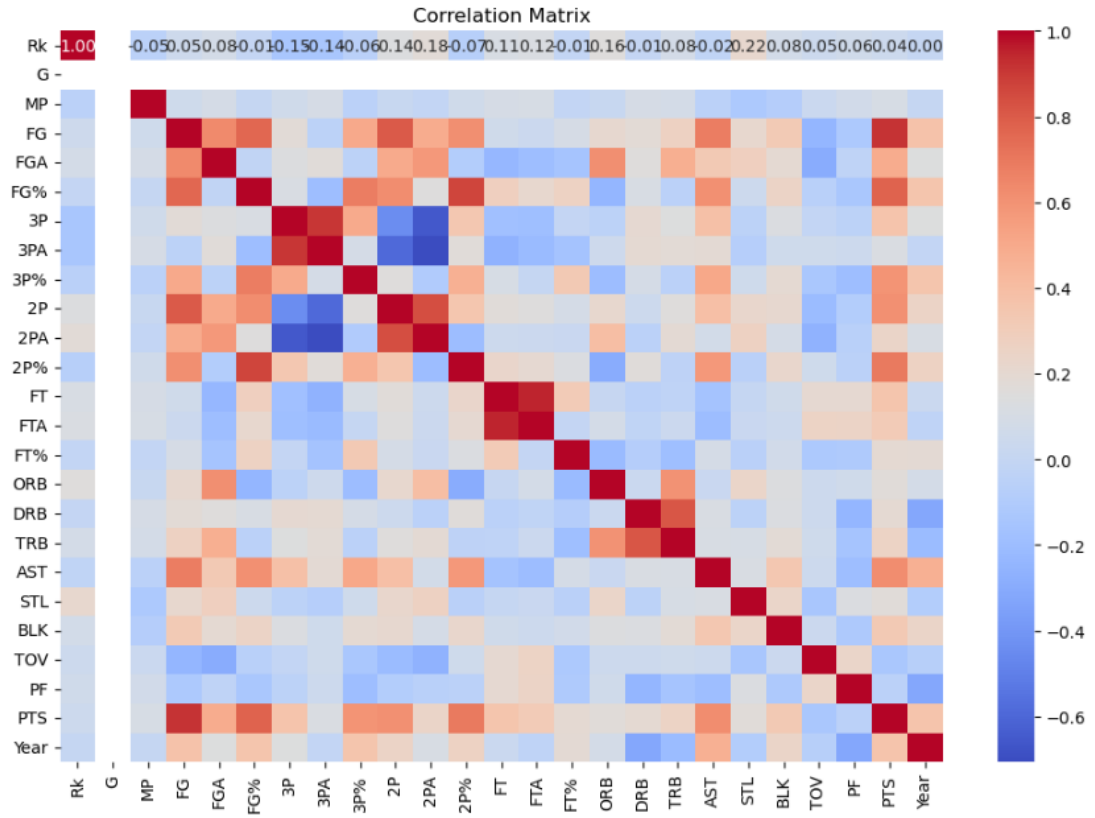


Figure 3: Correlation of Team Statistics

As seen in fig 3 above, correlation of team statistics is used to displays features that relate to each other:

Color Scale:

- Red Shades: Indicate a positive correlation.
- Blue Shades: Indicate a negative correlation.
- Intensity: The deeper the color, the stronger the correlation.

Correlation Values:

+1: Perfect positive correlation. It shows variable directly proportional to the other.

0: No correlation. The variables do not have a linear relationship.
-1: Perfect negative correlation. It shows inverse proportionality of variables.

Interpretation:

Diagonal Elements:

These are all equal to 1 (dark red) because each variable is perfectly correlated with itself.

Strong Positive Correlations:

- FG (Field Goals) and PTS (Points): As the number of field goals increases, the points scored also increase.
- FGA (Field Goals Attempted) and FG (Field Goals): More attempts generally lead to more field goals.
- 3PA (3-Point Field Goals Attempted) and 3P (3-Point Field Goals): More attempts generally lead to more 3-pointers.
- ORB (Offensive Rebounds) and TRB (Total Rebounds): Offensive rebounds are a component of total rebounds.
- DRB (Defensive Rebounds) and TRB (Total Rebounds): Defensive rebounds are also a component of total rebounds.

Strong Negative Correlations:

- FG% (Field Goal Percentage) and FGA (Field Goals Attempted): Often, more attempts can lower the shooting percentage if accuracy decreases with volume.
- TOV (Turnovers) and AST (Assists): High turnovers can negatively impact assists as it reflects poor ball handling.

Notable Observations:

- FG% (Field Goal Percentage) and PTS (Points): A positive correlation means better shooting accuracy generally leads to more points.
- FT% (Free Throw Percentage) and PTS (Points): Although free throw percentage is important, its correlation with total points might not be as strong as field goals.

Practical Implications:
Understanding these correlations helps in identifying which factors most significantly affect performance metrics. For example, coaches might focus on improving FG% (Field Goal Percentage) and reducing TOV (Turnovers) to enhance overall team performance.

2.2.2.2 Correlation of Player Statistics

Correlation of player statistics is shown in fig 4 below.

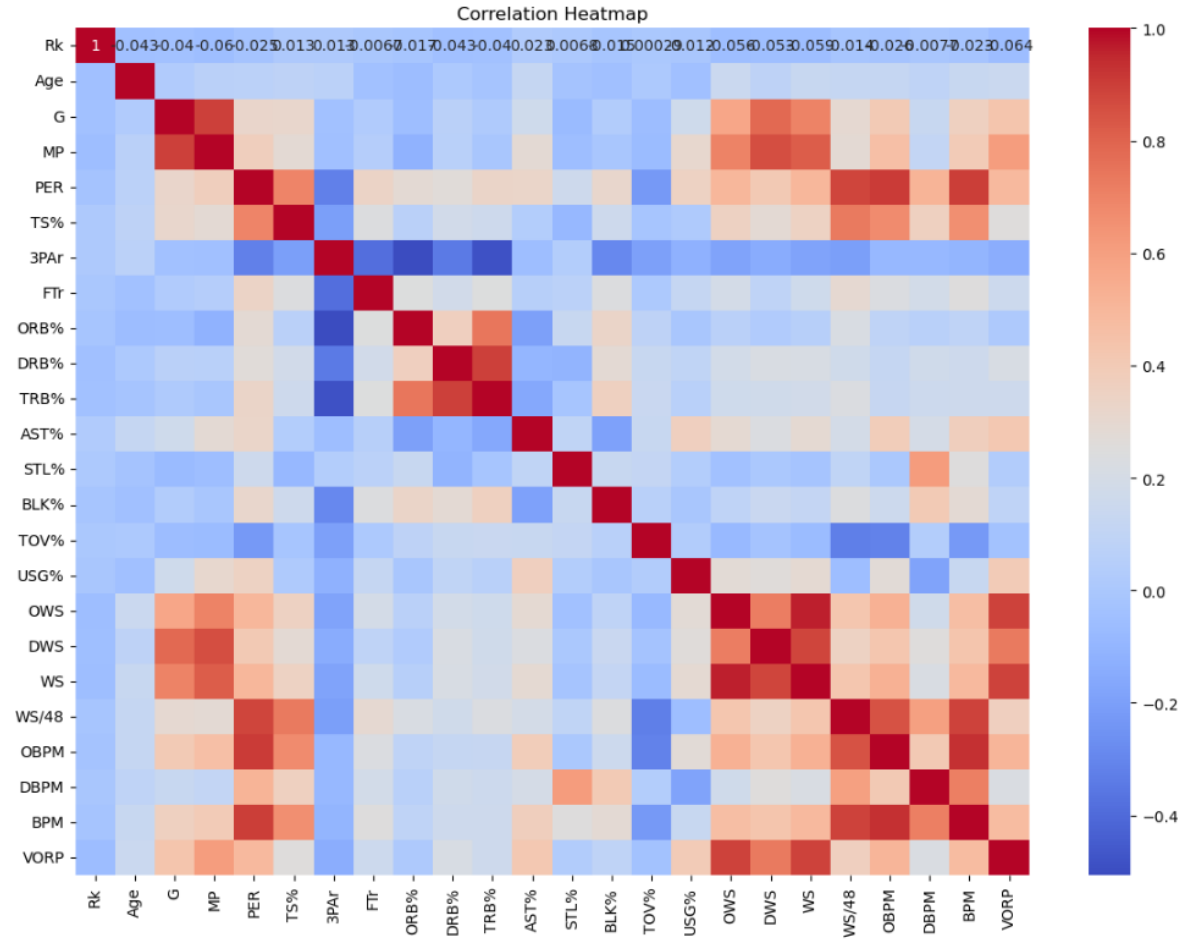


Fig 4: Correlation of Player Statistics

Correlation Coefficient

The degree and direction of a linear relationship between two variables are determined by the correlation coefficient (r).

The value of r ranges from -1 to 1:

- r = 1: Perfect correlation that is positive

- r = -1: Perfect correlation that is negative.

- r = 0: No correlation.

Heatmap Interpretation

Color Scale:

- Red indicates a positive correlation.
- Blue indicates a negative correlation.
- White/Neutral colors indicate little to no correlation.
- Color Intensity: The more intense the color, the stronger the correlation.

Key Observations

-Diagonal Line: The diagonal from the top-left to the bottom-right is dark red, indicating a perfect correlation (r = 1) of each statistic with itself.

- Positive Correlations:
  - PER (Player Efficiency Rating) and WS (Win Shares) have a strong positive correlation with VORP (Value Over Replacement Player).
  - OBPM (Offensive Box Plus/Minus) and BPM (Box Plus/Minus) are highly correlated with VORP.

-Negative Correlations:
  - TOV% (Turnover Percentage) tends to have a negative correlation with other performance metrics like TS% (True Shooting Percentage) and OWS (Offensive Win Shares).

-Notable Pairs:
  - MP (Minutes Played) and G (Games Played) are highly positively correlated, indicating that players who play more games also tend to accumulate more minutes.
  - DRB% (Defensive Rebound Percentage) and TRB% (Total Rebound Percentage) have a strong positive correlation, as defensive rebounds contribute significantly to total rebounds.

2.2.3 Average Advanced Player Statistics by Team

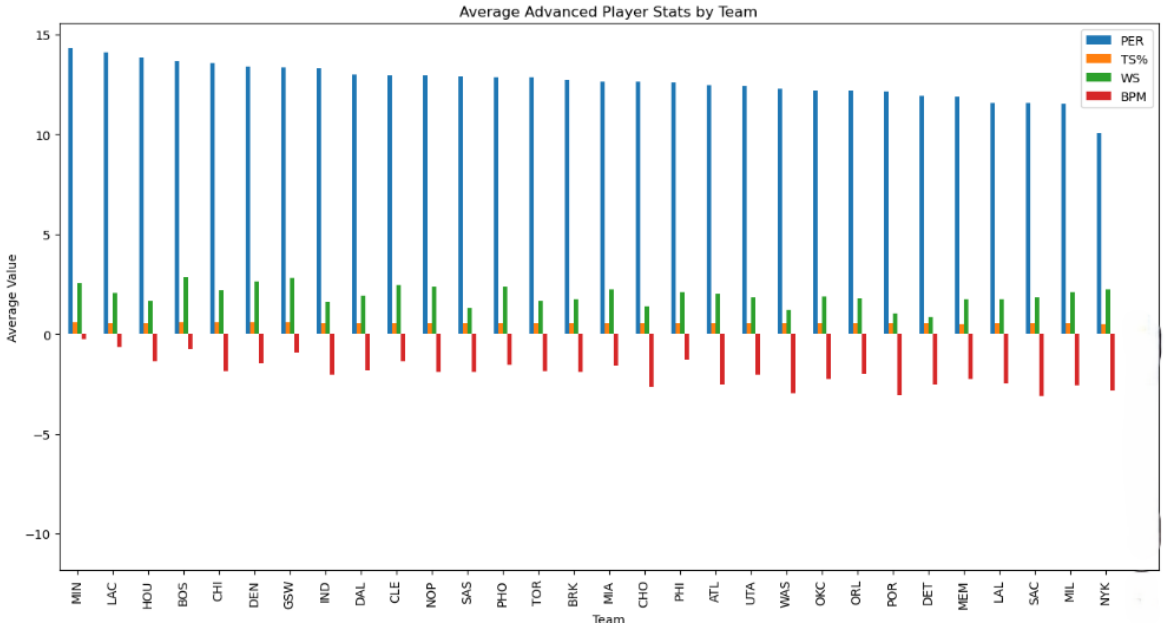


Fig 5: Average Advanced Player Statistics by Team

The Average Advanced Player Stats by Team" in fig 5 above shows the average values for four advanced basketball statistics for each NBA team, represented by different colored bars. The statistics shown are:

-PER (Player Efficiency Rating)- represented by blue bars.
-TS% (True Shooting Percentage) - represented by orange bars.
-WS (Win Shares) - represented by green bars.
-BPM (Box Plus/Minus) - represented by red bars.

Explanation:

Teams:
- The x-axis lists NBA teams using their abbreviations (e.g., MIN for Minnesota Timberwolves, LAC for Los Angeles Clippers, etc.).

Advanced Stats:
PER (Player Efficiency Rating):
- Used to determine per-minute performance of a player. A PER of 15 is considered average.
- In the chart, most teams have average PER values around 15, indicating that players across the league have similar per-minute productivity. The blue bars are consistent across all teams.

TS% (True Shooting Percentage):
- Accounts for a player's shooting efficiency, considering field goals, three-pointers, and free throws.
- The orange bars show slight variation among teams, but generally, they hover around the same range, indicating similar shooting efficiencies across teams.

WS (Win Shares):
- Approximate how many wins a player adds to their team.
- The green bars are positive for all teams, indicating that players generally contribute positively to their team's success. However, the heights of these bars are relatively small compared to PER, suggesting that while players contribute to wins, the impact is not as pronounced as their individual efficiency (PER).

BPM (Box Plus/Minus):
- Measures a player's total impact on their team. Positive BPM indicates a positive impact, while negative BPM indicates a negative impact.
- The red bars are negative for most teams, suggesting that on average, players have a negative impact when considering the league's overall performance. The extent of the negative BPM varies among teams, with some teams having deeper negative impacts.

Specific Observations:
- High PER Teams: Minnesota (MIN), Los Angeles Clippers (LAC), and Houston (HOU) have high average PER values, indicating efficient player performances.
- Negative BPM: Almost all teams have a negative average BPM, implying a league-wide trend where the average player's impact, as measured by BPM, is negative.

- Consistent TS% and WS: True Shooting Percentage (TS%) and Win Shares (WS) show less variation across teams. Most teams have positive and small values for WS, indicating that players contribute to wins but not dramatically.

Interpretation:
- Efficiency (PER): Players are consistently efficient across the league, with most teams showing similar values.
- Shooting (TS%): Shooting efficiency is fairly consistent among teams, suggesting similar shooting performance.
- Win Contribution (WS): Players contribute positively to wins, but the impact

is modest.
- Impact (BPM): Despite individual efficiency (PER), the overall impact (BPM) is negative for most teams, indicating potential areas for improvement in player performance or team strategies.

2.2.4 Top Players by Influence Score

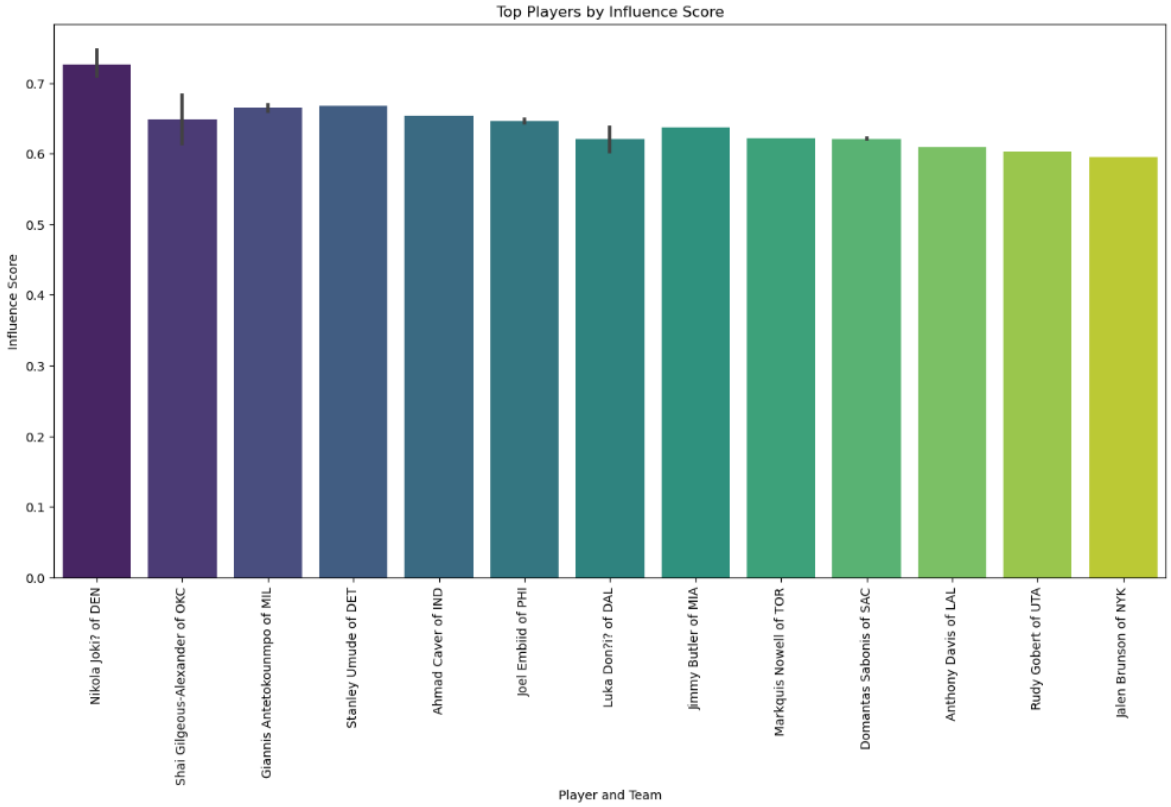


Fig 6: Top Players by Influence

The bar chart in fig 6 above represents the top NBA players ranked by their influence score. The y-axis shows the 'Influence Score', ranging from 0.0 to 0.7. The influence score quantifies each player's impact or influence in the context being measured. The X-Axis lists the players along with their respective teams. The players included are:

1. Nikola Jokić of DEN (Denver Nuggets)
2. Shai Gilgeous-Alexander of OKC (Oklahoma City Thunder)
3. Giannis Antetokounmpo of MIL (Milwaukee Bucks)
4. Stanley Umude of DET (Detroit Pistons)
5. Ahmad Caver of IND (Indiana Pacers)
6. Joel Embiid of PHI (Philadelphia 76ers)
7. Luka Dončić of DAL (Dallas Mavericks)

8. Jimmy Butler of MIA (Miami Heat)
9. Markquis Nowell of TOR (Toronto Raptors)
10. Domantas Sabonis of SAC (Sacramento Kings)
11. Anthony Davis of LAL (Los Angeles Lakers)
12. Rudy Gobert of UTA (Utah Jazz)
13. Jalen Brunson of NYK (New York Knicks)

Each bar represents a player's influence score, with the height of the bar corresponding to the player's influence score. Nikola Jokić of the Denver Nuggets has the highest influence score, close to 0.7. Other players like Shai Gilgeous-Alexander, Giannis Antetokounmpo, and others also have significant influence with scores around 0.6.

2.3 Player Efficiency Rating

Player Efficiency Rating (PER) summarizes a basketball player's performance into a single per-minute rating. It accounts for various statistics, adjusting for pace and weighting different contributions. With a league average of 15, PER allows for comparison across players, aiding in player evaluation, scouting, development, and fan engagement. There was no need of calculating PER separately since it is contained in Advanced player statistics dataset.

2.4 Feature Selection

Feature selection is an important step in the machine learning process as it involves choosing the most relevant features (independent variables) that are useful in predicting the target variable (dependent variable). In the context of predicting whether a team will win ('Win'), the following features have been selected:

-MP (Minutes Played)

-FG% (Field Goal Percentage)

-3P% (Three-Point Percentage)

-2PA (Two-Point Attempts)

-FT (Free Throws Made)

-FT% (Free Throw Percentage)

-DRB (Defensive Rebounds)

-TOV (Turnovers)

-PF (Personal Fouls)

Explanation of Feature Selection

- Relevance to the Target Variable: Each selected feature has a direct or indirect impact on the outcome of a game. For instance, shooting percentages (FG%, 3P%, FT%) directly affect the points scored, while turnovers (TOV) and personal fouls (PF) can hinder performance and provide scoring opportunities for the opponent.
- Predictive Power: These features collectively cover various aspects of the game, such as shooting efficiency, defensive capability, ball handling, and overall game strategy.
- Data Availability and Quality: The selected features are commonly recorded in basketball statistics, ensuring that there is sufficient data available for training and testing the predictive model. Additionally, these metrics are standardized and well-understood in the context of basketball analytics.

For all the models, the features above were used except Multilayer Layer Perceptron that all the features were used.

2.5 Data Preprocessing

Loading Data

Multiple CSV files containing team statistics for different seasons (e.g., Home Teams 2021-2022, Home Teams 2022-2023) are loaded into separate DataFrames. Each DataFrame was assigned columns to indicate the type of data (Home/Visitor) and the year of the data collection.

Combining Data

All the individual DataFrames are concatenated into a single DataFrame, `combined_stats`. This unified DataFrame aggregates data across different years and types (home and visitor games).

Handling Missing Values

The DataFrame is checked for missing values using `isnull().sum()`.
Missing values are handled by dropping rows with any missing values to make sure the dataset is complete and ready for analysis.

Checking for Duplicates

Duplicate rows in the DataFrame are identified using the `duplicated().sum()` function.
Any duplicate rows are removed using the `drop_duplicates()` function to ensure each data point is unique.

Checking Data Types

The data types of each column are verified using `dtypes`.
Ensuring that each column has the appropriate data type is crucial for accurate analysis and modeling.

Standardizing and Normalizing Data

Before loading the data into the models, it was standardized using `StandardScaler` from scikit-learn to ensure all features have a mean of 0 and a standard deviation of 1. This step is crucial for models that are sensitive to the scale of the data, such as logistic regression.

Feature Selection

Key features for the analysis and modeling are identified. For instance, features like 'MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', and 'PF' are selected for modeling.

Splitting Data

The combined dataset was split into training and validation sets using the `train_test_split` function. Typically, 70% of the dataset was used for training, while 15% was used for testing and 15% for validation.

Saving Cleaned Data

The cleaned and processed DataFrame is saved to a new CSV file Combined_Team_Stats_Cleaned for future use or further analysis. This ensures that the preprocessing steps do not need to be repeated.

2.6 Recursive Feature Elimination Especially in Logistic Regression

Recursive Feature Elimination (RFE) is utilized in selecting important features by recursively considering smaller sets of features. For the logistic regression model described, RFE was applied as follows:

-Model Initialization: The logistic regression model was trained initially with features like 'MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', and 'PF'.

-Feature Ranking: The model ranks feature in order of importance.

-Elimination Process: The feature that is least important is removed, and the model is retrained. This process is repeated iteratively, removing one feature at a time.

-Cross-Validation: The model is evaluated using cross-validation at each step to ensure performance remains optimal as features are eliminated.

-Optimal Features Selection: The process continues until important features are detected, balancing model complexity and performance.

RFE involves fitting the logistic regression model to all features and minimizing the logistic loss function. Features are ranked and recursively removed based on their coefficients until the optimal subset is identified. This approach enhances model performance and interpretability by focusing on the most influential features.

2.7 Model Selection

-Logistic Regression

Choice: Selected for its simplicity and effectiveness in binary classification.

-Random Forest

Choice: Utilized for its robustness and accuracy by averaging multiple decision trees.

-Multilayer Perceptron (MLP)

Choice: Selected for its ability to model complex non-linear relationships.

2.8 Hyperparameter Tuning for The Models

For Logistic Regression (LR):

For LR, hyperparameter tuning focuses on optimizing:

-C: Inverse of regularization strength; typical values are 0.01, 0.1, 1, 10, 100.

-Solver: Optimization algorithm; popular choices are 'lbfgs' and 'liblinear'.

-Penalty: Regularization type, with 'l2' being the most common.

-Max_iter: Maximum iterations for convergence, usually set to 100, 200, or 300.

For Random Forest (RF):

RF tuning involves:

-n_estimators: Number of trees, typically between 50 and 200.

-max_depth: Maximum tree depth; common values are None, 5, 10, 15.

-min_samples_split: Minimum samples to split a node, ranging from 2 to 40.

-min_samples_leaf: Minimum samples per leaf, also 2 to 40.

-max_features: Features considered per split; options include 'sqrt', 'log2', None.

-bootstrap: Whether to use bootstrap samples; True or False.

For Multilayer Perceptron (MLP):

MLP tuning includes:

-hidden_layer_sizes: Number and size of layers; configurations like (50,), (100,), (50, 50), (100, 50).

-activation: Activation function, 'tanh' or 'relu'.

-solver: Optimizer for weight updates, 'sgd' or 'adam'.

-alpha: L2 regularization term; values like 0.0001, 0.001, 0.01.

-learning_rate: used for weight updates; 'constant' or 'adaptive'.

GridSearchCV, and RandomizedSearchCV systematically explore these hyperparameters. GridSearchCV exhaustively searches a predefined grid, while RandomizedSearchCV samples parameter settings from distributions. Both utilize cross-validation to assess and choose the optimal model by considering performance metrics like accuracy, runtime balance, and optimization efficiency.

2.9 Evaluation of Model Performance

This was carried out using:

-Accuracy

Accuracy is the ratio of observations correctly predicted to the total observations.

Mathematical expression:

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

where:

TP = True Positives,

TN =True Negatives,

FP =False Positives,

FN =False Negatives.

-Confusion Matrix

Comparison actual versus predicted classifications.

Example of a confusion matrix:

[[TP, FP], [FN, TN]]

-Classification Report

This includes precision, recall, and F1-score for each class.

Precision: The ratio of correctly predicted positives to the total predicted positives.

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

Recall: The ratio of correctly predicted positives to all the observations in the actual class.

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

F1-Score: Average of Precision and Recall.

$$\text{F1-Score} = \frac{2*P*R}{(P+R)}$$

2.10 Data Instances

In building the models, the dataset was split into training set, testing set, and validation set as shown below:

For Logistic Regression Model:

```
# Splitting the data into training (70%) and remaining (30%) sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)

# Further splitting the remaining data into validation (15%) and testing (15%) sets
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

```
#checking for data instances for train set

X_train.shape
```

```
(126, 9)
```

```
#checking the data instances for test set
X_test.shape
```

```
(27, 9)
```

```
#checking for data instances for validation set

X_val.shape
```

```
(27, 9)
```

Fig 7: Data Instances for Logistic Regression

As seen in fig 7 above; initially, the data set was split into 70% for training and 30% for testing. The 30% for testing was further split to 15% for testing and 15% for validation.

This ensured proper splitting, giving the following:

Training set = 126

Testing set = 27

Validation = 27

Features = 9 respectively.

For Random Forest Model:

```
# Splitting the dataset: 70% train, 15% validation, 15% test
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50, random_state=42, stratify=y_temp)
```

```
#Checking data instances for the train set
X_train.shape
```

```
(126, 9)
```

```
#Checking data instances for test set

X_test.shape
```

```
(27, 9)
```

```
#Checking data instances for validation set
X_val.shape
```

```
(27, 9)
```

Fig 8: Data instances for Random Forest

In fig 8 above, the dataset was split into 70% for training, 15% for testing and 15% for validation. With proper splitting, the following were used:

Training set = 126

Testing set = 27

Validation set = 27

Features = 9 respectively.

For Multilayer Perceptron Model:

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

```
#Checking data instances for test set

X_test.shape
```

```
(27, 22)
```

```
#Checking data instances for train set

X_train.shape
```

```
(126, 22)
```

```
#Checking data instances for validation set
X_val.shape
```

```
(27, 22)
```

Fig 9: Data instances for Multilayer Perceptron Model

In fig 9 above, the dataset was split training (70%), testing (15%) and validation (15%).

With these, we have:

Training set = 126

Testing set = 27

Validation set = 27

Features = 22 respectively.

2.11 Receiver Operating Characteristics (ROC)

This is utilized in machine learning and statistics to evaluate binary classifier's performance. Typically, it signifies the following:

-True Positive Rate (TPR) vs. False Positive Rate (FPR): The ROC curve plots the TPR against FPR.

-Area Under the Curve (AUC): The AUC (Area Under the ROC Curve) indicates how effectively the classifier can differentiate between the two classes.

-**Comparison of Classifiers**: The ROC curve helps in comparing different classifiers. The curve closer to the top left corner indicates a better-performing model.

Here, validation ROC curve and Test ROC curve were displayed for all the models before and after hyperparameter tunning.

2.12 Learning Curve

Learning Curve is used to analyse and comprehend the model's performance as the training dataset size grows. The significance is:

-Training score: This is the blue line that shows how the performance of the model on the training dataset changes as more training examples are added.

- Cross-Validation Score: This is the red line displaying how the performance of the model on a validation set changes as more training examples are added.

Gap Between Lines: The difference in gap between the two curves indicate the degree of overfitting or underfitting.

-Convergence: Ideally, as the number of training examples increases, the training and cross-validation scores should converge to a high level, indicating that the model is performing well on both the training and validation sets.

CHAPTER THREE

RESULT AND DISCUSSION

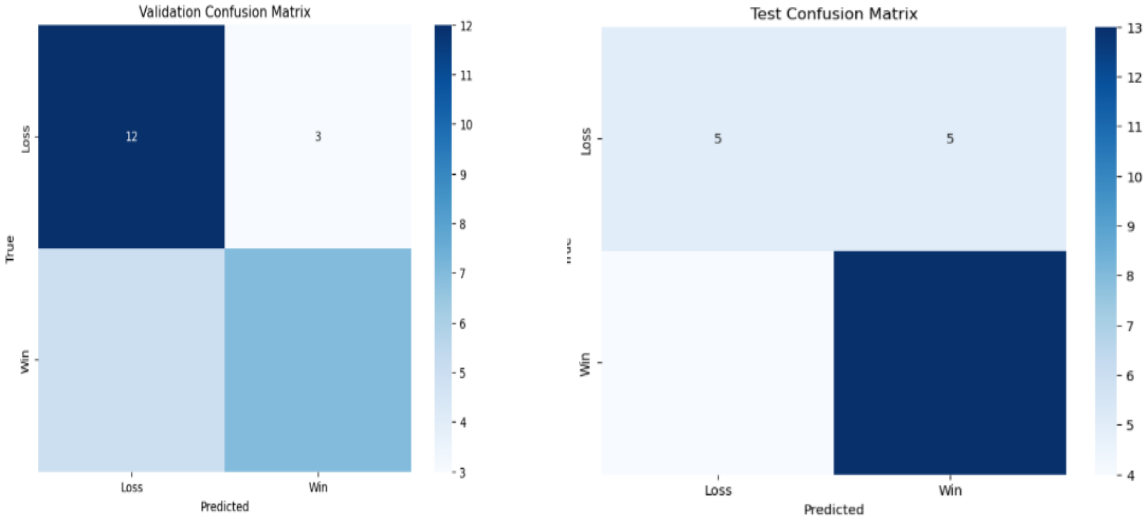
This chapter displays the results of everything caried out to arrive at useful results and outcome, and the analysis of the results with further discussion.

In this study, three (3) models were built namely Logistic Regression, Random Forest (RF) and Multilayer Perceptron (MLP). The features that were fed in the models were carefully selected to give the best performance of the models. After initial running of any of the models, hyperparameter tunning was carried out to improve the model's performance.

The models were evaluated in terms of Accuracy (Training Accuracy, Validation Accuracy, Test Accuracy), classification reports like Precision, Recall, F1 score, and plots were made for confusion matrix, Receiver Operating Characteristics (ROC) and Learning Curve. After analysing the models, an in-depth comparison of the models was done to showcase the model with the best performance.

3.1 LOGISTIC REGRESSION MODEL

The following results in fig 10 below were obtained:




```
Training Accuracy: 0.70
Validation Accuracy: 0.70
Test Accuracy: 0.67
Cross-validation Scores: [0.73076923 0.44       0.68       0.68       0.8        ]
Mean Cross-validation Score: 0.67
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.80      0.75        15
           1       0.70      0.58      0.64        12

    accuracy                           0.70        27
   macro avg       0.70      0.69      0.69        27
weighted avg       0.70      0.70      0.70        27

Test Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.50      0.53        10
           1       0.72      0.76      0.74        17

    accuracy                           0.67        27
   macro avg       0.64      0.63      0.63        27
weighted avg       0.66      0.67      0.66        27

Validation Confusion Matrix:
[[12  3]
 [ 5  7]]
Test Confusion Matrix:
[[ 5  5]
 [ 4 13]]
```

Fig 10: Logistic Regression Model Results

Validation Set Confusion Matrix:

	Predicted Loss	Predicted Win
True Loss	12	3
True Win	5	7

Class 0 (Loss):

True Positives (TP): 12

False Positives (FP): 5

True Negatives (TN): 7

False Negatives (FN): 3

-Metrics Calculation for Class 0 (Loss):

Precision: $P = \frac{TP}{(TP+FP)} = \frac{12}{(12+5)} = \frac{12}{17} = 0.71$Recall: $R = \frac{TP}{(TP+FN)} = \frac{12}{(12+3)} = \frac{12}{15} = 0.80$

F1-Score: $F1 = \frac{2*P*R}{(P+R)} = \frac{2*0.71*0.80}{(0.71+0.80)} = \frac{1.136}{1.51} = 0.75$

Class 1 (Win):

Precision P = $\frac{TP}{(TP+FP)} = \frac{7}{(7+3)} = \frac{7}{10} = 0.70$

Recall R = $\frac{TP}{(TP+FN)} = \frac{7}{(7+5)} = \frac{7}{12} = 0.58$

F1-Score: $\frac{2*P*R}{(P+R)}$ = $\frac{2*0.70*0.58}{(0.70+0.58)} = \frac{0.812}{1.28} = 0.64$

Overall Metrics for Validation Set:

Accuracy = $\frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{(12+7)}{(12+7+5+3)} = \frac{19}{27} = 0.70$

Macro Average:

Precision: P_macro = $\frac{(0.71+0.70)}{2} = 0.70$

Recall: R_macro = $\frac{(0.80+0.58)}{2} = 0.69$

F1-Score: F1_macro = (0.75 + 0.64) / 2 = 0.695 ≈ 0.69

Weighted Average:

Precision: P_weighted = $\frac{(0.71 * 15 + 0.70 * 12)}{(15 + 12)} = \frac{(10.65 + 8.4)}{27} = \frac{19.05}{27} = 0.70$

Recall: R_weighted = $\frac{(0.80 * 15 + 0.58 * 12)}{(15 + 12)} = \frac{(12 + 6.96)}{27} = 0.70$

F1-Score: F1_weighted = $\frac{(0.75 * 15 + 0.64 * 12)}{(15 + 12)} = \frac{11.25 + 7.68)}{27} = 0.70$

Test Set Confusion Matrix:

	Predicted Loss	Predicted Win
True Loss	5	5
True Win	4	13

Class 0 (Loss):

True Positives (TP): 5

False Positives (FP): 4

True Negatives (TN): 13

False Negatives (FN): 5

-Metrics Calculation for Class 0 (Loss):

Precision P = $\frac{TP}{(TP + FP)} = \frac{5}{(5 + 4)} = \frac{5}{9} = 0.56$

Recall R = $\frac{TP}{(TP + FN)} = \frac{5}{5(5+5)} = \frac{5}{10} = 0.50$

F1-Score: F1 = $\frac{2 * P * R}{(P + R)} = \frac{2 * 0.56 * 0.50}{(0.56 + 0.50)} = \frac{0.56}{1.06} = 0.53$

Class 1 (Win):

Precision: P = $\frac{TP}{(TP + FP)} = \frac{13}{(13 + 5)} = \frac{13}{18} = 0.72$

Recall: R = $\frac{TP}{(TP + FN)} = \frac{13}{(13 + 4)} = \frac{13}{17} = 0.76$

F1-Score: F1 = $\frac{2 * P * R}{(P + R)} = \frac{2 * 0.72 * 0.76}{(0.72 + 0.76)} = \frac{1.094}{1,48} = 0.74$

Overall Metrics for Test Set:

Accuracy: Accuracy = $\frac{(TP + TN)}{(TP + TN + FP + FN)} = \frac{(5 + 13)}{(5 + 13 + 4 + 5)} = \frac{18}{27} = 0.67$

Macro Average:

Precision: P_macro = $\frac{(0.56 + 0.72)}{2} = 0.64$

Recall: R_macro = $\frac{(0.50 + 0.76)}{2} = 0.63$

F1-Score: F1_macro = $\frac{(0.53 + 0.74)}{2} = 0.63$

Weighted Average:

Precision: P_weighted = $\frac{(0.56 * 10 + 0.72 * 17)}{(10 + 17)} = \frac{(5.6 + 12.24)}{27} = 0.66$

Recall: R_weighted = $\frac{(0.50 * 10 + 0.76 * 17)}{(10 + 17)} = \frac{(5 + 12.92)}{27} = 0.67$

F1-Score: F1_weighted = $\frac{(0.53 * 10 + 0.74 * 17)}{(10 + 17)} = \frac{(5.3 + 12.58)}{27} = 0.66$

The results from the validation and test sets highlight several key implications about the performance of the predictive model for classifying 'Loss' and 'Win' outcomes.

Firstly, the overall accuracy for both sets is approximately 70% for the validation set and 67% for the test set, indicating the model's reliability in making correct predictions most of the time. However, these figures also imply room for improvement, as roughly one-third of predictions are incorrect.

For the 'Loss' class, precision is approximately 0.71 in the validation set and 0.56 in the test set, suggesting the model is less effective at avoiding false positives in the test set. The recall for 'Loss' is higher in the validation set (0.80) than in the test set (0.50), indicating a decline in identifying true losses.

For the 'Win' class, both precision and recall are more consistent between sets, with precision around 0.70-0.72 and recall around 0.58-0.76. This consistency shows better performance in predicting wins, although the lower recall in the validation set indicates some wins are missed.

The macro and weighted averages reflect similar trends, reinforcing the need for improving precision and recall, particularly for the 'Loss' class in the test set. Overall, these metrics suggest that while the model has a solid foundation,

enhancing its capability to correctly predict losses will improve its effectiveness.

Performance of Logistic Regression Model

-ROC (Receiver Operating Characteristic) Curve:

The ROC curve in fig 11 below shows the performance of a classification model with the true positive rate (TPR) against the false positive rate (FPR). The blue line represents the validation set with an AUC (area under the curve) of 0.82, indicating good performance. The orange line represents the test set with an AUC of 0.76, showing slightly worse performance. The diagonal dashed line represents a random classifier with AUC of 0.5. The higher the curve above this line, the better the model's performance, demonstrating that the model performs reasonably well on both sets.

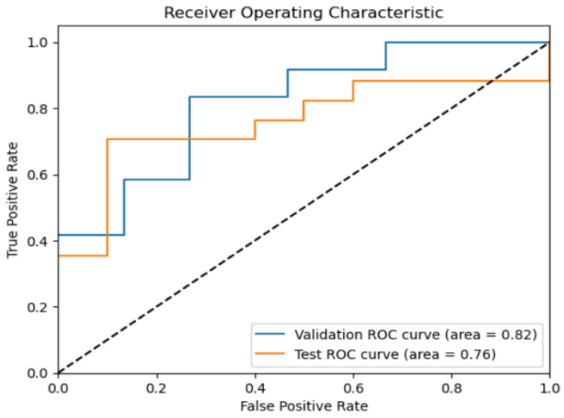


Fig 11: ROC curve

-Learning Curve

The learning curve in fig 12 below displays the training and cross-validation scores of a logistic regression model as the training examples increases. The training score is represented by red line, initially high but drops as more examples are added, indicating overfitting at the beginning. The cross-validation score is represented by the green line, initially low and fluctuating, but gradually increasing as more data is used, indicating improved generalization. The meeting of the training and cross-validation scores suggests that the model's performance stabilizes with more training data, reducing overfitting and stabilizing its ability on generalizing unseen data.

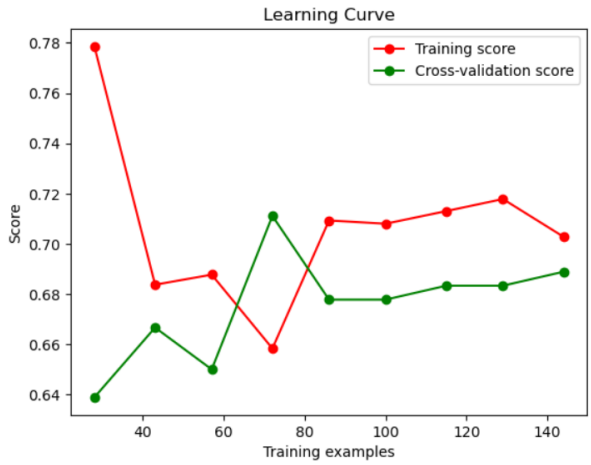


Fig 12: Learning Curve

3.1.2 LOGISTIC REGRESSION MODEL WITH HYPERPARAMETER TUNNING

As seen in fig 13 below, the following results were obtained.

```
Best Parameters: {'C': 100, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
Training Accuracy with Best Model: 0.78
Validation Accuracy with Best Model: 0.78
Validation Classification Report with Best Model:
              precision    recall  f1-score   support

           0       0.76      0.87      0.81        15
           1       0.80      0.67      0.73        12

    accuracy                           0.78        27
   macro avg       0.78      0.77      0.77        27
weighted avg       0.78      0.78      0.77        27

Test Accuracy with Best Model: 0.74
Test Classification Report with Best Model:
              precision    recall  f1-score   support

           0       0.64      0.70      0.67        10
           1       0.81      0.76      0.79        17

    accuracy                           0.74        27
   macro avg       0.72      0.73      0.73        27
weighted avg       0.75      0.74      0.74        27

Validation Confusion Matrix:
[[13  2]
 [ 4  8]]
Test Confusion Matrix:
[[ 7  3]
 [ 4 13]]
```

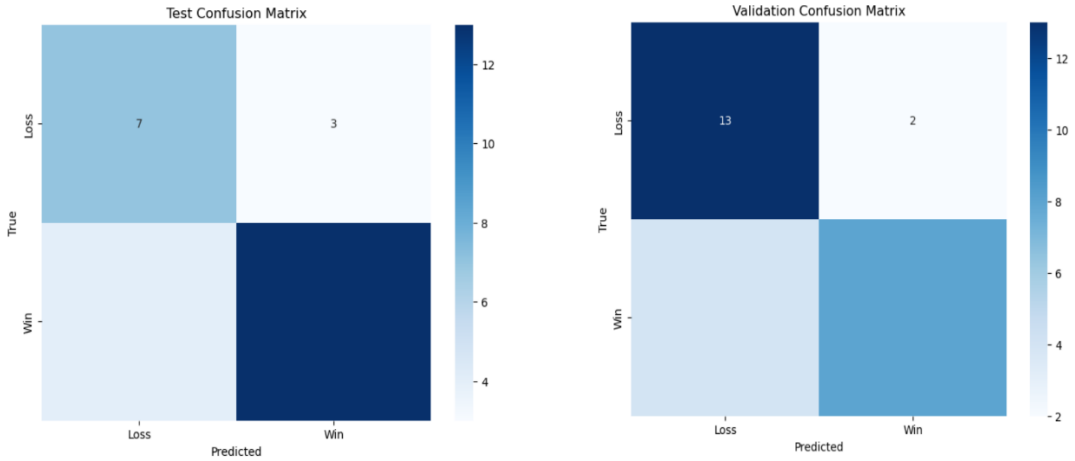


Fig 13: Logistic Regression Model with Hyperparameter Tunning

Validation Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.76	0.87	0.81	15
1	0.80	0.67	0.73	12
Accuracy			0.78	27
Macro Avg	0.78	0.77	0.77	27
Weighted Avg	0.78	0.78	0.77	27

Validation Confusion Matrix

[[13 2]
[4 8]]

Test Metrics

Test Accuracy with Best Model: 0.74

Test Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.64	0.70	0.67	10
1	0.81	0.76	0.79	17
Accuracy			0.74	27
Macro Avg	0.72	0.73	0.73	27
Weighted Avg	0.75	0.74	0.74	27

Test Confusion Matrix

[[7 3]
 [4 13]]

Calculations

Accuracy

Accuracy is the ratio of correctly predicted instances to the total instances.

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

Validation Accuracy Calculation:

Accuracy = $\frac{(13 + 8)}{(13 + 8 + 2 + 4)} = \frac{21}{27} = 0.78$

Test Accuracy Calculation:

Accuracy = $\frac{(7 + 13)}{(7 + 13 + 3 + 4)} = \frac{20}{27} = 0.74$

PrecisionPrecision= $\frac{TP}{(TP + FP)}$

Validation Precision for Class 0:

Precision_0 = $\frac{13}{(13 + 2)} = \frac{13}{15} = 0.87$

Validation Precision for Class 1:

Precision_1 = $\frac{8}{(8 + 4)} = \frac{8}{12} = 0.67$

Test Precision for Class 0:

Precision_0 = $\frac{7}{(7 + 3)} = \frac{7}{10} = 0.70$

Test Precision for Class 1:

Precision_1 = $\frac{13}{(13 + 4)} = \frac{13}{17} = 0.76$

Recall

Recall = $\frac{TP}{(TP+FN)}$

Validation Recall for Class 0:

$$\text{Recall\_0} = \frac{13}{(13+4)} = \frac{13}{17} = 0.76$$

Validation Recall for Class 1:

$$\text{Recall\_1} = \frac{8}{(8+2)} = \frac{8}{10} = 0.80$$

Test Recall for Class 0:

$$\text{Recall\_0} = \frac{7}{(7+4)} = \frac{7}{11} = 0.64$$

Test Recall for Class 1:

$$\text{Recall\_1} = \frac{13}{(13+3)} = \frac{13}{16} = 0.81$$

F1-Score

$$\text{F1-Score} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Validation F1-Score for Class 0:

$$\text{F1-Score\_0} = \frac{2 * (0.87 * 0.76)}{(0.87 + 0.76)} = \frac{2 * (0.6612)}{1.63} = 0.81$$

Validation F1-Score for Class 1:

$$\text{F1-Score\_1} = \frac{2 * (0.67 * 0.80)}{(0.67 + 0.80)} = \frac{2 * (0.536)}{1.47} = 0.73$$

Test F1-Score for Class 0:

$$\text{F1-Score\_0} = \frac{2 * (0.70 * 0.64)}{(0.70 + 0.64)} = \frac{2 * (0.448)}{1.34} = 0.67$$

Test F1-Score for Class 1:

$$\text{F1-Score\_1} = \frac{2 * (0.76 * 0.81)}{(0.76 + 0.81)} = \frac{2 * (0.6156)}{1.57} = 0.79$$

This logistic regression analysis with hyperparameter tuning presents the best parameters, test, training, and validation results. The model actualised training accuracy of 0.78, validation accuracy of 0.78, and test accuracy of 0.74. Detailed classification reports for validation and test data show precision, recall, and F1-scores for both classes. Confusion matrices display the true positives,

false positives, true negatives, and false negatives for validation and test sets, highlighting the model's performance in correctly predicting wins and losses. The analysis emphasizes areas for potential improvement in model predictions.

Performance of Logistic Regression Model with Hyperparameter Tunning

-ROC curve

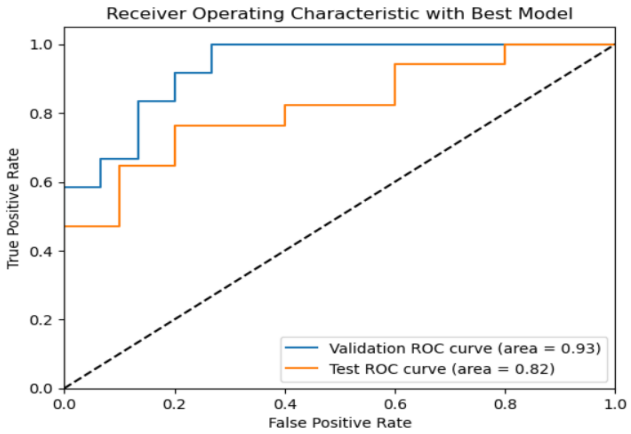


Fig 14: ROC curve for logistic Regression with Hyperparameter Tunning

The ROC (Receiver Operating Characteristic) curve plot in fig 14 above compares the performance of the model on validation and test datasets. The plot displays two curves: the blue line for validation data with an AUC (Area Under Curve) of 0.93, and the orange line for test data with an AUC of 0.82. These curves illustrate the model's ability to differentiate classes. Additionally, cross-validation scores are presented, indicating variability in model performance across different folds, with scores ranging from 0.48 to 0.96 and a mean cross-validation score of 0.71, suggesting moderate model performance.

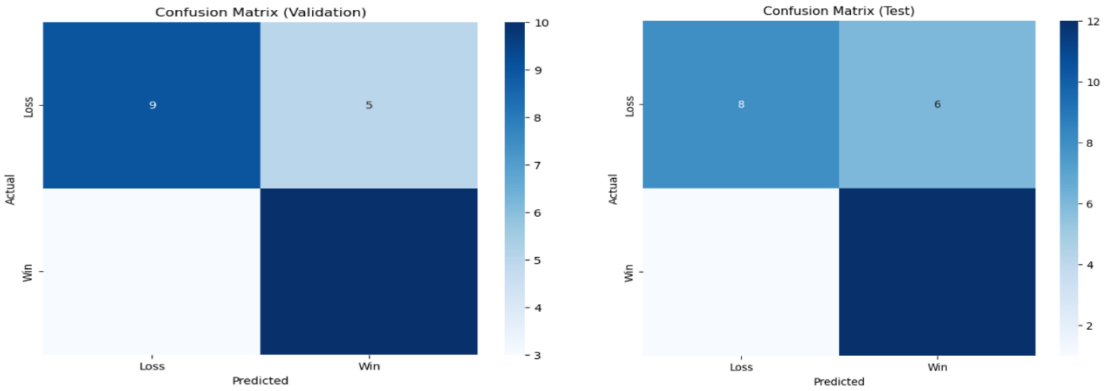
-Learning Curve

Fig 15: Learning curve for Logistic Regression with Hyperparameter tunning

Learning curve plot in fig 15 above illustrates the model's performance with varying amounts of training data. The red line represents the training score, while the green line represents the cross-validation score. Initially, the training score is high but decreases as more data is added, stabilizing around 0.76. The cross-validation score starts lower, around 0.64, and improves with more training examples, eventually converging near the training score around 0.74. This indicates that as more data is used, the model generalizes better, reducing overfitting and improving validation performance, suggesting an effective learning process.

3.2 **RANDOM FOREST MODEL**

For Random Forest model, the following results in fig 16 below were obtained




```
Training Accuracy: 1.00
Validation Accuracy: 0.70
Test Accuracy: 0.74
Cross-validation scores: [0.80769231 0.84       0.88       0.76       0.76       ]
Average cross-validation score: 0.81
Confusion Matrix (Validation):
[[ 9  5]
 [ 3 10]]
Confusion Matrix (Test):
[[ 8  6]
 [ 1 12]]
Classification Report (Validation):
              precision    recall  f1-score   support

           0       0.75      0.64      0.69        14
           1       0.67      0.77      0.71        13

    accuracy                           0.70        27
   macro avg       0.71      0.71      0.70        27
weighted avg       0.71      0.70      0.70        27

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.89      0.57      0.70        14
           1       0.67      0.92      0.77        13

    accuracy                           0.74        27
   macro avg       0.78      0.75      0.73        27
weighted avg       0.78      0.74      0.73        27
```

Fig 16: Random Forest Model

Training and Validation Metrics:

Training Accuracy: 1.00 (100%)
Validation Accuracy: 0.70 (70%)
Test Accuracy: 0.74 (74%)
Cross-validation scores: [0.80769231, 0.84, 0.88, 0.76, 0.76]
Average cross-validation score: 0.81 (81%)

Confusion Matrices:

Validation Set Confusion Matrix:

	Predicted	
Actual	Loss	Win
Loss	9	5
Win	3	10

Test Set Confusion Matrix:

	Predicted	
Actual	Loss	Win
Loss	8	6
Win	1	12

Classification Reports:

Validation Set Classification Report:

	precision	recall	f1-score	support
0	0.75	0.64	0.69	14
1	0.67	0.77	0.71	13
accuracy			0.70	27
macro avg	0.71	0.71	0.70	27
weighted avg	0.71	0.70	0.70	27

Test Set Classification Report:

	precision	recall	f1-score	support
0	0.89	0.57	0.70	14
1	0.67	0.92	0.77	13
accuracy			0.74	27
macro avg	0.78	0.75	0.73	27
weighted avg	0.78	0.74	0.73	27

Calculations:

Accuracy:

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

Validation Set Accuracy:

Accuracy = $\frac{(9 + 10)}{27} = \frac{19}{27} = 0.70$

Test Set Accuracy:

Accuracy = $\frac{(8 + 12)}{27} = \frac{20}{27} = 0.74$

Precision:

Precision = $\frac{TP}{TP+FP}$

Validation Set Precision:

- Class 0: Precision_0 = $\frac{9}{(9 + 3)} = \frac{9}{12} = 0.75$
- Class 1: Precision_1 = $\frac{10}{(10 + 5)} = \frac{10}{15} = 0.67$

Test Set Precision:

- Class 0: Precision_0 = $\frac{8}{(8 + 1)} = \frac{8}{9} = 0.89$
- Class 1: Precision_1 = $\frac{12}{(12 + 6)} = \frac{12}{18} = 0.67$

Recall:

Recall = $\frac{TP}{(TP+FN)}$

Validation Set Recall:

- Class 0: Recall_0 = $\frac{9}{(9 + 5)} = \frac{9}{14} = 0.64$
- Class 1: Recall_1 = $\frac{10}{(10 + 3)} = \frac{10}{13} = 0.77$

Test Set Recall:

- Class 0: Recall_0 = $\frac{8}{(8+6)} = \frac{8}{14} = 0.57$
- Class 1: Recall_1 = $\frac{12}{(12+1)} = \frac{12}{13} = 0.92$

F1-Score:

F1-Score = $\frac{2 * (Precision * Recall}{(Precision + Recall)}$

Validation Set F1-Score:

- Class 0: F1-Score_0 = $\frac{2 * (0.75 * 0.64)}{(0.75 + 0.64)} = \frac{2 * 0.48}{1.39} = 0.69$
- Class 1: F1-Score_1 = $\frac{2 * (0.67 * 0.77)}{(0.67 + 0.77)} = \frac{2 * 0.52}{1.44} = 0.71$

Test Set F1-Score:

- Class 0: F1-Score_0 = $\frac{2 * (0.89 * 0.57)}{(0.89 + 0.57)} = \frac{2 * 0.51}{1.46} = 0.70$
- Class 1: F1-Score_1 = $\frac{2 * (0.67 * 0.92)}{(0.67 + 0.92)} = \frac{2 * 0.62}{1.59} = 0.77$

In the results from the Random Forest classifier, the first confusion matrix - the validation confusion matrix shows 9 true negatives, 5 false positives, 3 false negatives, and 10 true positives and the test confusion matrix indicates 8 true negatives, 6 false positives, 1 false negative, and 12 true positives. Metrics include training accuracy (1.00), validation accuracy (0.70), and test accuracy (0.74). Cross-validation scores range from 0.76 to 0.88, averaging 0.81. The detailed classification reports highlight precision, recall, and F1-scores for both validation and test sets, reflecting balanced performance. These metrics illustrate the model's overall effectiveness and robustness.

Performance of Random Forest Model

-ROC curve

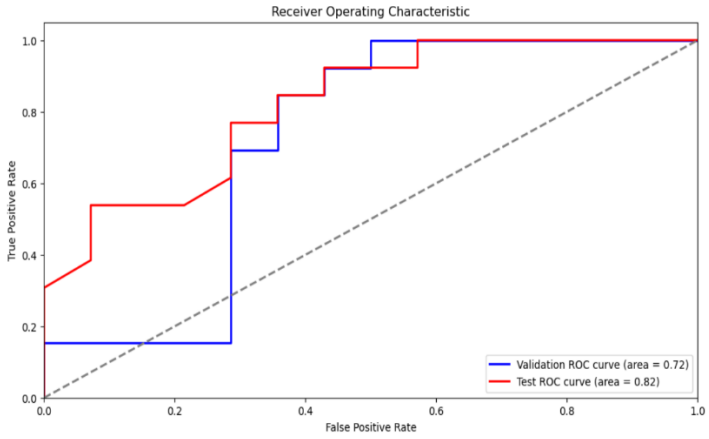


Fig 17: ROC curve for Random Forest Model

The Receiver Operating Characteristic (ROC) curve in fig 17 above is for classification of the Random Forest model, comparing its performance on validation and test datasets. It shows the plots of the True Positive Rate (TPR) against the False Positive Rate (FPR). The diagonal dashed line represents a random classifier's performance. The blue line shows the ROC curve for the validation set with an Area Under the Curve (AUC) of 0.72, while the red line represents the test set with an AUC of 0.82. A higher AUC indicates better model performance, with the test set showing stronger performance in this case.

-Learning Curve

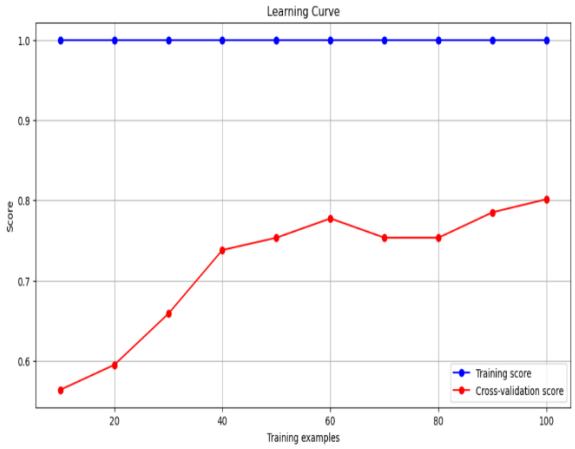


Fig 18: Learning Curve for Random Forest Model

The learning curve in fig 18 above shows the performance of the Random Forest model as the number of training examples increases. The blue line represents the training score, consistently at 1.0, indicating the model perfectly

fits the training data. The red line shows the cross-validation score, which starts low but improves as more training examples are used, eventually stabilizing around 0.78. The difference between the training and cross-validation scores indicates overfitting, where the model excels on the training data but performs less effectively on unseen data.

3.2.1 RANDOM FOREST WITH HYPERPARAMETER TUNNING

The Random Forest model achieved a training accuracy of 0.78, validation accuracy of 0.63, and test accuracy of 0.67. Cross-validation yielded an average score of 0.79. The confusion matrices for both validation and test data show a balanced mix of true positives and false negatives, indicating room for improvement. The validation classification report indicates precision and recall values of 0.70 and 0.50 for class 0, and 0.59 and 0.77 for class 1, resulting in an overall accuracy of 0.63. The test report shows precision and recall values of 0.78 and 0.50 for class 0, and 0.61 and 0.85 for class 1, with an accuracy of 0.67. This suggests the model performs moderately well but could benefit from further tuning. The results are shown in fig 19 below.

```
Training Accuracy: 0.78
Validation Accuracy: 0.63
Test Accuracy: 0.67
Cross-validation scores: [0.84615385 0.88       0.8        0.72       0.68       ]
Average cross-validation score: 0.79
Confusion Matrix (Validation):
[[ 7  7]
 [ 3 10]]
Confusion Matrix (Test):
[[ 7  7]
 [ 2 11]]
Classification Report (Validation):
              precision    recall  f1-score   support

           0       0.70      0.50      0.58        14
           1       0.59      0.77      0.67        13

    accuracy                           0.63        27
   macro avg       0.64      0.63      0.62        27
weighted avg       0.65      0.63      0.62        27

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.78      0.50      0.61        14
           1       0.61      0.85      0.71        13

    accuracy                           0.67        27
   macro avg       0.69      0.67      0.66        27
weighted avg       0.70      0.67      0.66        27
```

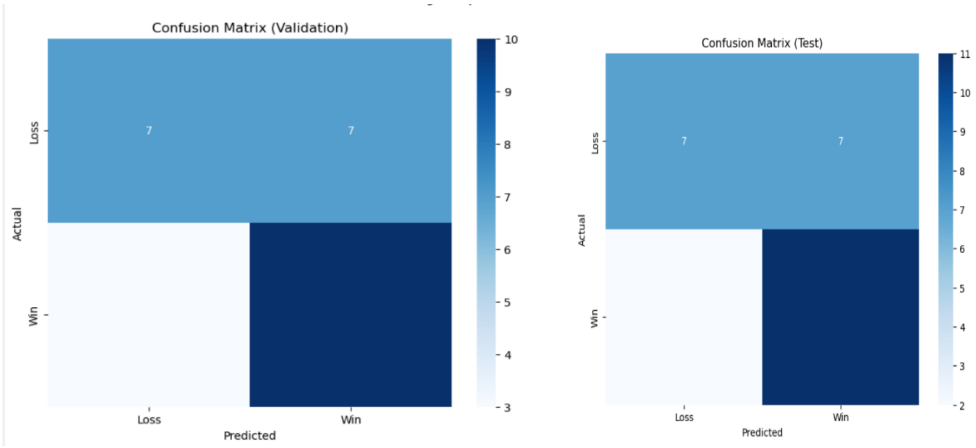


Fig 19: Random Forest Model with Hyperparameter Tunning

Training, Validation, and Test Accuracy

- Training Accuracy: 0.78
- Validation Accuracy: 0.63
- Test Accuracy: 0.67

Cross-validation Scores

- Scores: [0.84615385, 0.88, 0.8, 0.72, 0.68]
- Average Cross-validation Score: 0.79

Confusion Matrices

Validation Confusion Matrix

	Predicted Loss	Predicted Win
Actual Loss	7	7
Actual Win	3	10

Test Confusion Matrix

	Predicted Loss	Predicted Win
Actual Loss	7	7
Actual Win	2	11

Classification Reports

Validation Classification Report

Class	Precision	Recall	F1-score	Support
0	0.70	0.50	0.58	14
1	0.59	0.77	0.67	13
Accuracy			0.63	27
Macro avg	0.64	0.63	0.62	27
Weighted avg	0.65	0.63	0.62	27

Test Classification Report

Class	Precision	Recall	F1-score	Support
0	0.78	0.50	0.61	14
1	0.61	0.85	0.71	13
Accuracy			0.67	27
Macro avg	0.69	0.67	0.66	27
Weighted avg	0.70	0.67	0.66	27

Mathematical Calculations

1. Accuracy: $\frac{(TP + TN)}{(TP + TN + FP + FN)}$

   - Validation: $\frac{(7 + 10)}{27} = \frac{17}{27} = 0.63$
   - Test: $\frac{(7 + 11)}{27} = \frac{18}{27} = 0.67$

2. Precision: $\frac{TP}{(TP+FP)}$

   - Validation (Class 0): $\frac{7}{(7 + 7)} = \frac{7}{14} = 0.50$
   - Validation (Class 1): $\frac{10}{(10 + 3)} = \frac{10}{13} = 0.77$
   - Test (Class 0): $\frac{7}{(7 + 7)} = \frac{7}{14} = 0.50$
   - Test (Class 1): $\frac{11}{(11 + 2)} = \frac{11}{13} = 0.85$

3. Recall: $\frac{TP}{(TP+FN)}$

   - Validation (Class 0): $\frac{7}{(7 + 7)} = \frac{7}{14} = 0.50$
   - Validation (Class 1): $\frac{10}{(10 + 3)} = \frac{10}{13} = 0.77$

- Test (Class 0): $\frac{7}{(7+7)} = \frac{7}{14} = 0.50$
- Test (Class 1): $\frac{11}{(11+2)} = \frac{11}{13} = 0.85$

4. F1-Score: $\frac{2 * (Precision * Recall)}{(Precision + Recall)}$

- Validation (Class 0): $\frac{2 * (0.70 * 0.50)}{(0.70 + 0.50)} = 0.58$
- Validation (Class 1): $\frac{2 * (0.59 * 0.77)}{(0.59 + 0.77)} = 0.67$
- Test (Class 0): $\frac{2 * (0.78 * 0.50)}{(0.78 + 0.50)} = 0.61$
- Test (Class 1): $\frac{2 * (0.61 * 0.85)}{(0.61 + 0.85)} = 0.71$

Performance of Random Forest Model with Hyperparameter

-ROC Curve

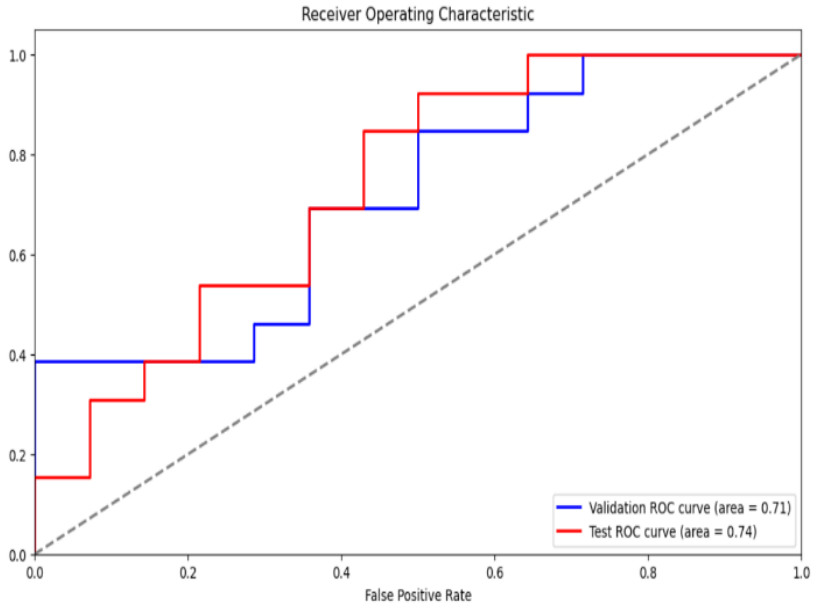


Fig 20: ROC for Random Forest Model with Hyperparameter Tunning

The ROC curve for the Random Forest model in fig 20 above shows the model's performance in distinguishing between the two classes. The blue line represents the validation set with an area under the curve (AUC) of 0.71, while the red line represents the test set with an AUC of 0.74. An AUC closer to 1 indicates better

performance. These AUC values imply the model has a fair level of discrimination ability, performing slightly better on the test set than on the validation set. However, there is still room for improvement to enhance the model's predictive accuracy.

-Learning Curve

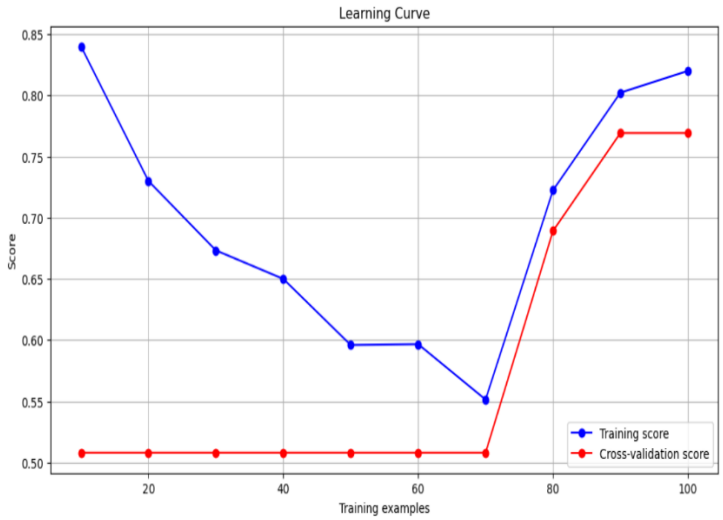


Fig 21: Learning curve of Random Forest with Hyperparameter Tunning

The learning curve in fig 21 above illustrates the model's performance as the number of training examples increases. The training score is represented by the blue line, which starts high but decreases as more data is used, indicating that the model fits the training data well initially but overfits as complexity increases. The red line represents the cross-validation score, which starts low but increases, showing the model's generalization ability improves with more training data. The convergence of both lines towards the end suggests that the model benefits from more data, achieving a balance between bias and variance, and indicating that additional training data could further enhance model performance.

3.3 **MULTILAYER PERCEPTRON (MLP) MODEL**

The following results were obtained as seen in fig 22 below.

```
Training Accuracy: 0.75
Validation Accuracy: 0.74
Test Accuracy: 0.78
Validation Confusion Matrix:
[[ 9  6]
 [ 1 11]]
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.60      0.72        15
           1       0.65      0.92      0.76        12

    accuracy                           0.74        27
   macro avg       0.77      0.76      0.74        27
weighted avg       0.79      0.74      0.74        27

Test Confusion Matrix:
[[ 4  6]
 [ 0 17]]
Test Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.40      0.57        10
           1       0.74      1.00      0.85        17

    accuracy                           0.78        27
   macro avg       0.87      0.70      0.71        27
weighted avg       0.84      0.78      0.75        27
```

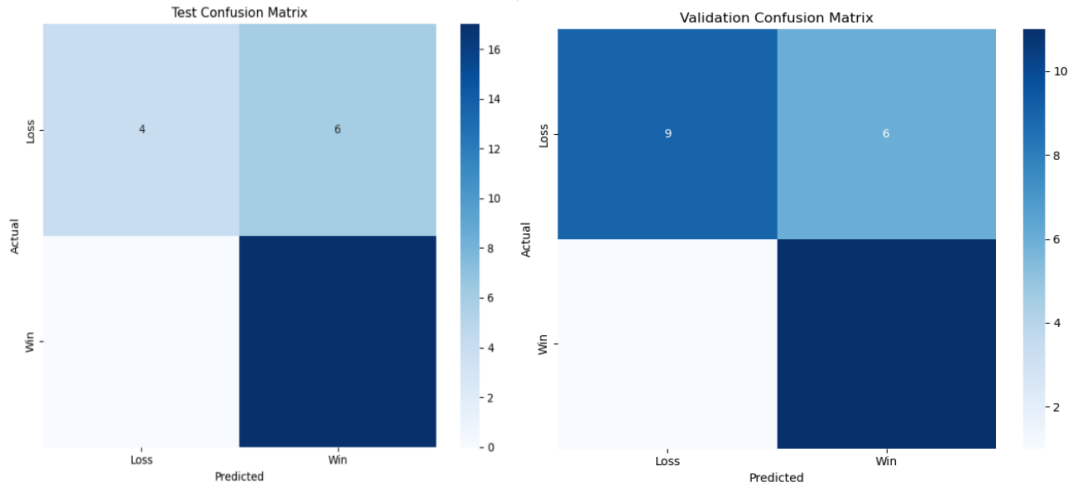


Fig 22: Multilayer Perceptron

Validation Dataset

Confusion Matrix:
[[9, 6], [1, 11]]

Classification Report:

Class	Precision	Recall	F1-Score	Support
0	0.9	0.6	0.72	15
1	0.65	0.92	0.76	12
Accuracy		0.74	0.74	27
Macro Avg	0.77	0.76	0.74	27
Weighted Avg	0.79	0.74	0.74	27

Metrics Calculations:

Accuracy:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} = \frac{(9 + 11)}{(9 + 11 + 6 + 1)} = \frac{20}{27} = 0.74$$

Precision (Class 0):

$$\text{Precision} = \frac{TP}{(TP+FP)} = \frac{9}{(9+6)} = \frac{9}{15} = 0.60$$

Recall (Class 0):

$$\text{Recall} = \frac{TP}{(TP + FN)} = \frac{9}{(9 + 1)} = \frac{9}{10} = 0.90$$

F1-Score (Class 0):

$$\text{F1-Score} = \frac{2 * (Precision * Recall)}{(Precision + Recall)} = \frac{2 * (0.60 * 0.90)}{(0.60 + 0.90)} = \frac{2 * 0.54}{1.50} = 0.72$$

Test Dataset

Confusion Matrix:
[[4, 6], [0, 17]]

Classification Report:

Class	Precision	Recall	F1-Score	Support
0	1.0	0.4	0.57	10
1	0.74	1.0	0.85	17
Accuracy		0.78	0.78	27
Macro Avg	0.87	0.7	0.71	27
Weighted Avg	0.84	0.78	0.77	27

Metrics Calculations:

Accuracy:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} = \frac{(4 + 17)}{(4 + 17 + 6 + 0)} = \frac{21}{27} = 0.78$$

Precision (Class 0):

$$\text{Precision} = \frac{TP}{(TP + FP)} = \frac{4}{(4+0)} = \frac{4}{4} = 1.00$$

Recall (Class 0):

$$\text{Recall} = \frac{TP}{(TP + FN)} = \frac{4}{(4 + 6)} = \frac{4}{10} = 0.40$$

F1-Score (Class 0):

$$\text{F1-Score} = \frac{2 * (Precision * Recall)}{(Precision + Recall)} = \frac{2 * (1.00 * 0.40)}{(1.00 + 0.40)} = \frac{2 * 0.40}{1.40} = 0.57$$

Cross-Validation Scores

Cross-validation scores: [0.80769231, 0.64, 0.8, 0.68, 0.8]
Mean CV score: 0.75

The Multilayer Perceptron model was evaluated using both validation and test datasets. The training, validation, and test accuracies are 0.75, 0.74, and 0.78, respectively. The validation confusion matrix shows 9 correct predictions for 'Loss' and 11 for 'Win', while the test confusion matrix shows 4 correct predictions for 'Loss' and 17 for 'Win'. The precision, recall, and F1-scores indicate that the model performs better in predicting 'Win' than 'Loss'. The cross-validation scores averaged to 0.75, indicating consistent model performance. Improvements can be made by balancing the dataset, hyperparameter tuning, and feature engineering.

Performance of Multilayer Perceptron Model

-ROC Curve

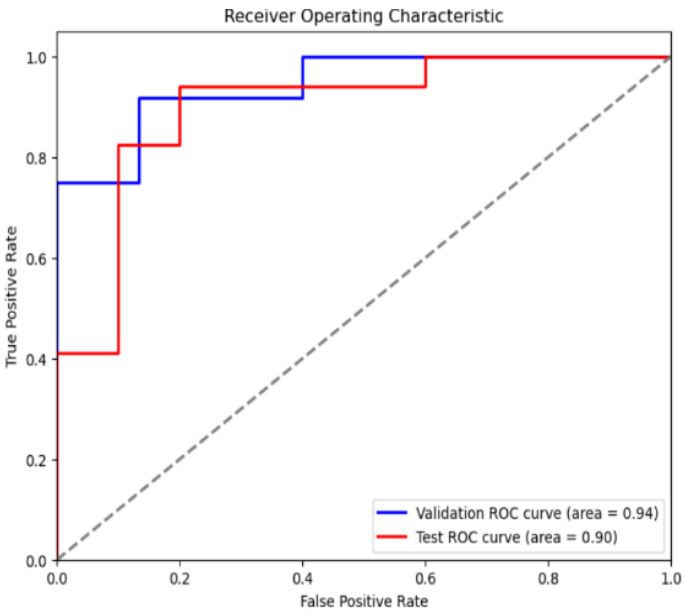


Fig 23. ROC for Multilayer Perceptron Model

Receiver Operating Characteristic (ROC) curve in fig 23 above shows the performance of the MLP (Multilayer Perceptron) model. The blue curve represents the validation ROC with an area under the curve (AUC) of 0.94, indicating excellent model performance on the validation set. The red curve represents the test ROC with an AUC of 0.90, also indicating strong performance, but slightly lower than the validation set. The dashed line represents a random classifier with an AUC of 0.5.

-Learning Curve

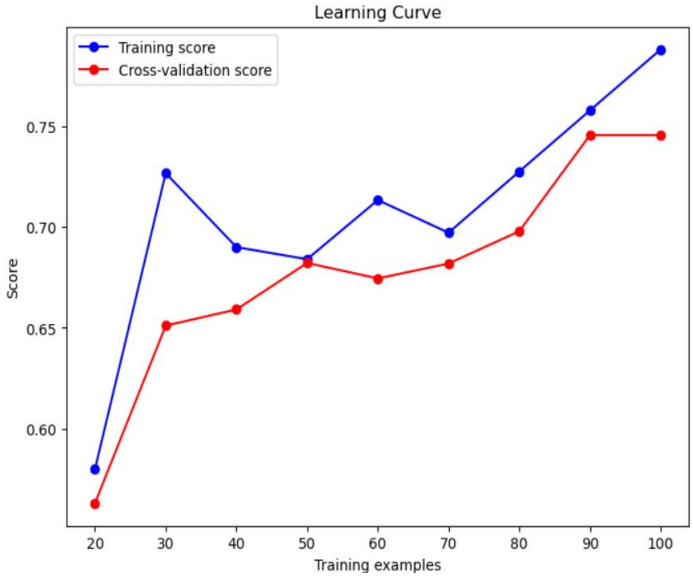
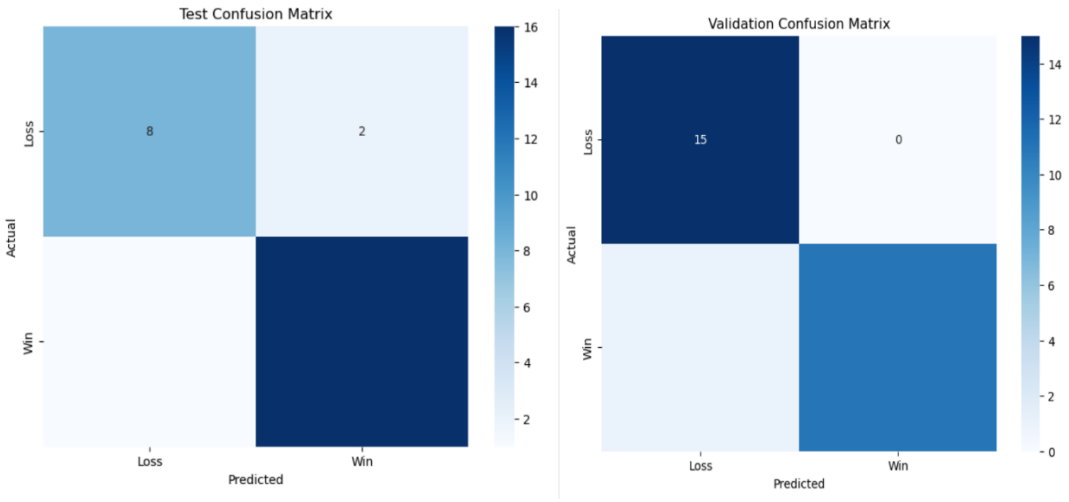


Fig 24. Learning Curve for Multilayer Perceptron

The Learning curve for multilayer perceptron (MLP) model in fig 24 above shows the relationship between the training score (blue) and the cross-validation score (red) as the number of training examples increases. Initially, both scores rise sharply, indicating the model is learning effectively from more data. The training score consistently remains higher than the cross-validation score, highlighting potential overfitting. As training examples increase beyond 80, the difference between the training and cross-validation scores narrows, suggesting improved model generalization. The plateauing of the cross-validation score indicates the model's performance stabilizes with more data, reaching around 0.75 accuracy.

3.3.1 MULTILAYER PERCEPTRON WITH HYPERPARAMETER TUNNING

The following results in fig 25 below were obtained.




```
Training Accuracy: 0.91
Validation Accuracy: 0.96
Test Accuracy: 0.89
Validation Confusion Matrix:
[[15  0]
 [ 1 11]]
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        15
           1       1.00      0.92      0.96        12

    accuracy                           0.96        27
   macro avg       0.97      0.96      0.96        27
weighted avg       0.97      0.96      0.96        27

Test Confusion Matrix:
[[ 8  2]
 [ 1 16]]
Test Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.80      0.84        10
           1       0.89      0.94      0.91        17

    accuracy                           0.89        27
   macro avg       0.89      0.87      0.88        27
weighted avg       0.89      0.89      0.89        27
```

Fig 25: MLP with Hyperparameter Tunning

Training, Validation, and Test Accuracies

Training Accuracy: 0.91

Validation Accuracy: 0.96

Test Accuracy: 0.89

Validation Confusion Matrix

	Predicted	
Actual	Loss	Win
Loss	15	0
Win	1	11

Validation Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.94	1.00	0.97	15
1	1.00	0.92	0.96	12
Accuracy			0.96	27
Macro Avg	0.97	0.96	0.96	27
Weighted Avg	0.97	0.96	0.96	27

Test Confusion Matrix

	Predicted	
Actual	Loss	Win
Loss	8	2
Win	1	16

Test Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.89	0.80	0.84	10
1	0.89	0.94	0.91	17
Accuracy			0.89	27
Macro Avg	0.89	0.87	0.88	27
Weighted Avg	0.89	0.89	0.89	27

Mathematical Calculations

Accuracy:

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

- For Validation:

$$\text{Accuracy} = \frac{(15 + 11)}{27} = \frac{26}{27} = 0.96$$

- For Test:

Accuracy = $\frac{(8+16)}{27} = \frac{24}{27} = 0.89$

Precision:

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

- For Class 0 in Validation:

Precision = $\frac{15}{(15+0)} = \frac{15}{15} = 1.00$

- For Class 1 in Validation:

Precision = $\frac{11}{(11+1)} = 0.92$

- For Class 0 in Test:

Precision = $\frac{8}{(8+2)} = 0.80$

- For Class 1 in Test:

Precision = $\frac{16}{(16+1)} = \frac{16}{17} = 0.94$

Recall:

Recall = $\frac{TP}{(TP+FN)}$

- For Class 0 in Validation:

Recall = $\frac{15}{(15+0)} = 1.00$

- For Class 1 in Validation:

Recall = $\frac{11}{(11+1)} = 0.92$

- For Class 0 in Test:

Recall = $\frac{8}{(8+2)} = 0.80$

- For Class 1 in Test:

Recall = $\frac{16}{(16+1)} = 0.94$

F1-Score:

F1-Score = $\frac{2 * (Precision * Recall)}{(Precision + Recall)}$

- For Class 0 in Validation:

F1-Score = $\frac{2 * (0.94 * 1.00)}{(0.94 + 1.00)} = 0.97$

- For Class 1 in Validation:

F1-Score = $\frac{2 * (1.00 * 0.92)}{(1.00 + 0.92)} = 0.96$

- For Class 0 in Test:

F1-Score = $\frac{2 * (0.89 * 0.80)}{(0.89 + 0.80)} = 0.84$

- For Class 1 in Test:

F1-Score = $\frac{2 * (0.89 * 0.94)}{(0.89 + 0.94)} = 0.91$

The Multilayer Perceptron (MLP) model exhibits strong performance with a training accuracy of 0.91, a validation accuracy of 0.96, and a test accuracy of 0.89. The validation confusion matrix shows perfect prediction for class 0 (loss) with 15 true positives and no false positives, and high prediction accuracy for class 1 (win) with 11 true positives and 1 false negative. The test confusion matrix indicates 8 true positives and 2 false negatives for class 0, and 16 true positives and 1 false positive for class 1. Precision, recall, and F1-scores are consistently high, reflecting reliable model performance across all classes.

Performance of Multilayer Perceptron Model with Hyperparameter Tunning

-ROC Curve

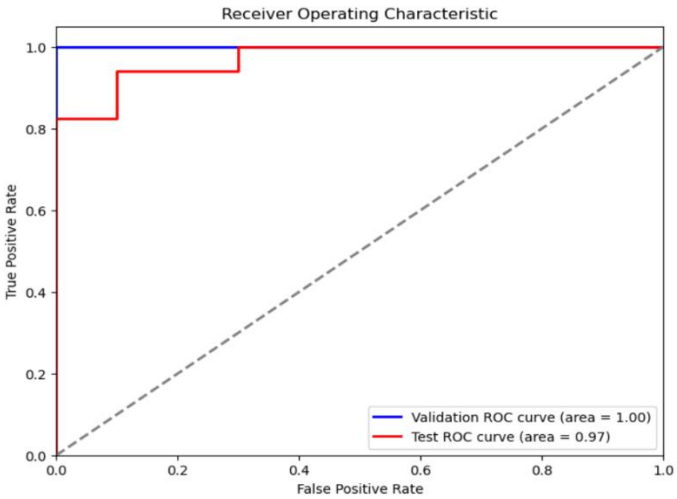


Fig 26: ROC for Multilayer Perceptron with Hyperparameter Tunning

The ROC (Receiver Operating Characteristic) curve for the Multilayer Perceptron (MLP) model in fig 26 above illustrates its performance in distinguishing between classes. The validation ROC curve (in blue) has an area under the curve (AUC) of 1.00, indicating perfect classification with no false positives. The test ROC curve (in red) has an AUC of 0.97, demonstrating excellent model performance with a high TPR (true positive rate) and low FPR (false positive rate). These high AUC values confirm the model's robust ability to discriminate between the classes, with near-perfect results on the validation set and strong results on the test set.

-Learning Curve

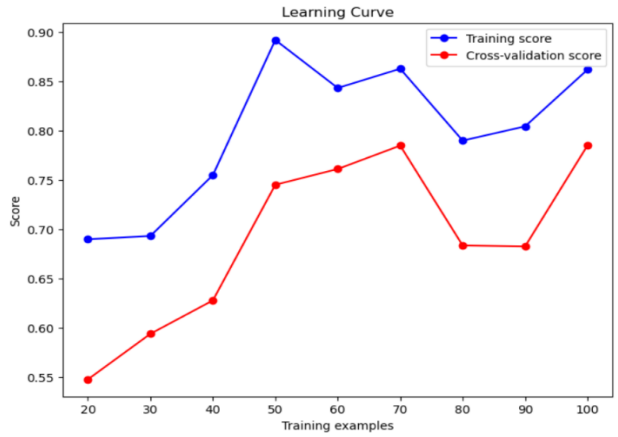


Fig 27: Learning Curve for Multilayer Perceptron with Hyperparameter Tunning

The learning curve for the Multilayer Perceptron (MLP) model in fig 27 above shows the training and cross-validation scores as the number of training examples increases. The training score (blue) generally improves and remains high, indicating how well the model reads the training data. The cross-validation score gets better with more training examples but fluctuates a bit. The gap between the two curves suggests room for improvement in model generalization to unseen data.

3.4 COMPARISON OF MODELS

After building the models and performing hyperparameter tuning, the Multilayer Perceptron (MLP) model emerges as the best among the three models (Logistic Regression, Random Forest, and Multilayer Perceptron).

Model	Class	Testing Accuracy	Precision	Recall	F1-Score	Support
Logistic Regression	0	74	64	70	67	10
	1		81	76	79	17
Random Forest	0	67	78	50	61	14
	1		61	85	71	13
Multilayer Perceptron	0	89	89	80	84	10
	1		89	94	91	17

Table 2: Comparison of Models

As seen in table 2 above, the three machine learning models were evaluated on two classes (0 and 1) using metrics like Accuracy, Precision, Recall, F1- score, and Support.

-Testing Accuracy: The Multilayer Perceptron (MLP) model outperforms the others with an accuracy of 89%, indicating it is the most effective overall.

Logistic Regression follows with 74%, while Random Forest has the lowest accuracy at 67%.

-Precision: Precision for class 0 is highest in the MLP model at 89%, indicating it is the most reliable in predicting class 0 correctly. The Random Forest model has a lower precision for class 1 (61%), suggesting it has a higher rate of false positives for this class compared to the other models.

-Recall: The MLP model also excels in recall, especially for class 1 (94%), meaning it is highly effective at correctly identifying true positives in this class. The Random Forest model has the lowest recall for class 0 at 50%, showing a tendency to miss actual positives.

-F1-Score: The MLP model again leads with higher F1-scores, particularly in class 1 (91%), reflecting a smooth balance between P and R (precision and recall). The Logistic Regression and Random Forest models show lower F1-scores, particularly for class 0.

Overall, the Multilayer Perceptron model is the most robust across all metrics, making it the preferred choice among the three.

CHAPTER FOUR

CONCLUSION AND LIMITATIONS

4.1 CONCLUSION

This dissertation has demonstrated the significant potential of sports analytics in predicting NBA game outcomes using machine learning models. The comprehensive analysis incorporated logistic regression, random forest, and multilayer perceptron models, each showing varying degrees of success in predicting game results based on historical data from the 2021/2022 to 2023/2024 NBA seasons. By employing rigorous feature engineering and hyperparameter tuning processes, the study was able to optimize model performance, ensuring that each algorithm was finely tuned to obtain the most useful insights from the data.

The feature engineering process involved identifying and selecting critical box score metrics that significantly influence game outcomes. These metrics included field goal percentage, three-point percentage, free throw percentage, rebounds, assists, steals, blocks, turnovers, and personal fouls. By focusing on these key performance indicators, the models were able to capture the nuanced dynamics of basketball games, leading to more accurate predictions.

Hyperparameter tuning was another crucial aspect of the study, involving systematic adjustments to the models' parameters to enhance their predictive accuracy. Techniques such as grid search and cross-validation were employed to determine the optimal parameter settings, which improved the models' ability to perform between the training data to unseen game data. This process was essential in refining the models and ensuring that they were robust and reliable.

The results underscored the importance of key box score metrics in influencing game outcomes. For instance, field goal percentage was found to be a critical determinant of a team's offensive efficiency, while rebounds, both offensive and defensive, played a significant role in controlling possession and limiting opponents' scoring opportunities. Turnovers, on the other hand, were identified as a critical factor negatively impacting a team's performance, as they represent lost scoring opportunities and increased chances for the opponent to score.

Overall, the research confirmed that integrating advanced data analytics into sports can offer substantial advantages for teams, coaches, analysts, and other stakeholders. The insights gained from this study can be applied to optimize player rotations, develop game strategies, and enhance overall team performance. For fans and analysts, the data-driven insights deepen their understanding of the game and enrich the viewing experience.

Furthermore, the dissertation provides a robust foundation for future work, encouraging further exploration and application of machine learning techniques in sports analytics. Future research could move beyond this study by adding more advanced algorithms, such as deep learning, and exploring additional performance metrics. The ultimate goal is to enhance predictive accuracy and strategic decision-making in sports, leveraging data-driven insights to achieve competitive advantages and elevate the overall standards of performance in the NBA and other sports leagues.

4.2 LIMITATIONS

The limitations of this dissertation are multifaceted, reflecting challenges in data availability, model selection, and generalization.

Firstly, the study relies on publicly available datasets, which may not capture all relevant variables or offer the granularity needed for nuanced analysis, potentially leading to biases or gaps in predictive accuracy.

Secondly, the selected models, while robust, may not fully account for the complexities and dynamics inherent in NBA games, such as player injuries, coaching strategies, and psychological factors, which are difficult to quantify and incorporate into a predictive framework.

Thirdly, the study also focuses on a relatively short timeframe, which might limit the generalizability of the findings across different seasons or in response to evolving team dynamics and player performance trends.

Furthermore, the dissertation assumes that historical data patterns will continue in the future, which may not hold true due to the unpredictable nature of sports.

Lastly, computational limitations, particularly in hyperparameter tuning and model training, may have constrained the exploration of more complex models that could potentially offer better performance. These factors collectively highlight the challenges in achieving high predictive accuracy and the need for ongoing refinement and validation of the models used.

CHAPTER FIVE

RECOMMENDATIONS

Building on the insights gained from this research, several recommendations are proposed for future work and practical applications.

Firstly, enlarging the dataset to include more seasons and additional performance metrics could improve model accuracy and robustness.

Secondly, integrating real-time data streams and in-game statistics could enhance predictive capabilities, offering dynamic and timely insights.

Thirdly, exploring other advanced ML (machine learning) algorithms, like deep learning and ensemble methods, may yield better performance.

Additionally, collaboration with NBA teams and analysts can provide practical validation and refinement of the models.

Lastly, extending the application of these models to other sports could validate their versatility and effectiveness. Emphasizing ethical considerations, especially data privacy and security, remains crucial as the use of analytics in sports continues to grow. Implementing these recommendations will further the impact of sports analytics, driving innovation and strategic advancements in the industry.

BIBLIOGRAPHY

1. Morgulev, E., Azar, O.H. and Lidor, R., 2018. Sports analytics and the big-data era. *International Journal of Data Science and Analytics*, *5*, pp.213-222. https://doi.org/10.1007/s41060-017-0093-7
2. Dunham, M.H., 2006. *Data mining: Introductory and advanced topics*. Pearson Education India.
3. Andrienko, G., Gunopulos, D., Ioannidis, Y., Kalogeraki, V., Katakis, I., Morik, K. and Verscheure, O., 2016. Mining urban data (Part B). *Information Systems*, *57*, pp.75-76. https://doi.org/10.1016/j.is.2016.01.001
4. Vinué Visús, G. and Epifanio, I., 2019. Forecasting basketball players' performance using sparse functional data, pp 1–26.
5. Cervone, D., Bornn, L. and Goldsberry, K., 2016, March. NBA court realty. In *10th MIT Sloan Sports Analytics Conference*.
6. Franks, A., Miller, A., Bornn, L. and Goldsberry, K., 2015, February. Counterpoints: Advanced defensive metrics for nba basketball. In *9th annual MIT sloan sports analytics conference, Boston, MA* (Vol. 10), pp. 1–8
7. Nagarajan, R. and Li, L., 2017, November. Optimizing NBA player selection strategies based on salary and statistics analysis. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)* (pp. 1076-1083). IEEE.
8. Cao, C., 2012. Sports data mining technology used in basketball outcome prediction.
9. Burges, C.J., 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), pp.121-167.
10. David, W. H., Lemeshow, S. and Rodney, X, S., 2000. Applied logistic regression. Wiley, New York, pp 236–269
11. Langley, P., Iba, W., Thompson, K,. 1992. An analysis of Bayesian classifiers. In: The tenth national conference on artificial intelligence, vol. 24. AAAI Press, San Jose, pp 399–406
12. Schalkoff, R.J., 1997. Artificial neural networks. International ed. McGraw-Hill, New York

13. Keller, J.M., Gray, M.R. and Givens, J.A., 1985. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4), pp.580-585.
14. Langley, P., Iba, W. and Thompson, K., 1992, July. An analysis of Bayesian classifiers. In *Aaai* (Vol. 90, pp. 223-228).
15. Lewis, D.D., 1998, April. Naive (Bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning* (pp. 4-15). Berlin, Heidelberg: Springer Berlin Heidelberg.
16. Meyer, D., Leisch, F. and Hornik, K., 2003. The support vector machine under test. *Neurocomputing*, *55*(1-2), pp.169-186.
17. Quinlan, J.R., 1986. Induction of decision trees. *Machine learning*, *1*, pp.81-106. https://doi.org/10.1007/bf00116251
18. Lieder, N., 2018. Can machine-learning methods predict the outcome of an nba game?. *Available at SSRN 3208101*. http://dx.doi.org/10.2139/ssrn.3208101
19. Aly, M., 2005. Survey on multiclass classification methods. *Neural Netw*, *19*(1-9), p.2.
20. Lorena, A.C., Jacintho, L.F., Siqueira, M.F., De Giovanni, R., Lohmann, L.G., De Carvalho, A.C. and Yamamoto, M., 2011. Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications*, *38*(5), pp.5268-5275.
21. Pernkopf, F., 2005. Bayesian network classifiers versus selective k-NN classifier. *Pattern recognition*, *38*(1), pp.1-10.
22. Islam, M.J., Wu, Q.J., Ahmadi, M. and Sid-Ahmed, M.A., 2007, November. Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers. In *2007 international conference on convergence information technology (ICCIT 2007)* (pp. 1541-1546). IEEE.
23. Xhemali, D., Hinde, C.J. and Stone, R., 2009. Naïve bayes vs. decision trees vs. neural networks in the classification of training web pages.
24. Amor, N.B., Benferhat, S. and Elouedi, Z., 2004, March. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 420-424).
25. Lorena, A.C., Jacintho, L.F., Siqueira, M.F., De Giovanni, R., Lohmann, L.G., De Carvalho, A.C. and Yamamoto, M., 2011. Comparing machine learning classifiers in potential distribution modelling. *Expert Systems with Applications*, *38*(5), pp.5268-5275.
26. Robnik-Šikonja, M., 2004, September. Improving random forests. In *European conference on machine learning* (pp. 359-370). Berlin, Heidelberg: Springer Berlin Heidelberg.

27. Han, J., Pei, J. and Tong, H., 2022. Data mining: concepts and techniques. Morgan kaufmann.
28. Zak, T.A., Huang, C.J. and Siegfried, J.J., 1979. Production efficiency: the case of professional basketball. *Journal of Business*, pp.379-392.
29. Terner, Z. and Franks, A., 2021. Modeling player and team performance in basketball. *Annual Review of Statistics and Its Application*, *8*(1), pp.1-23.https://doi.org/10.1146/annurev-statistics-040720-015536
30. https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/

CODE APPENDICES

Appendix A – file path for loading the dataset.

```
# Defining file paths
file_paths = {
    'home_2021': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Home Teams 2021-2022 Season.xls.csv',
    'home_2022': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Home Teams 2022-2023 Season.xls.csv',
    'home_2023': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Home Teams 2023-2024 Season.xls.csv',
    'visitor_2021': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Visitor Teams 2021-2022 Season.xls.csv',
    'visitor_2022': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Visitor Teams 2022-2023 Season.xls.csv',
    'visitor_2023': 'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Visitor Teams 2023-2024 Season.xls.csv'
}

# Loading data and add 'Type' and 'Year' columns
home_2021 = pd.read_csv(file_paths['home_2021'])
home_2021['Type'] = 'Home'
home_2021['Year'] = 2021

home_2022 = pd.read_csv(file_paths['home_2022'])
home_2022['Type'] = 'Home'
home_2022['Year'] = 2022

home_2023 = pd.read_csv(file_paths['home_2023'])
home_2023['Type'] = 'Home'
home_2023['Year'] = 2023

visitor_2021 = pd.read_csv(file_paths['visitor_2021'])
visitor_2021['Type'] = 'Visitor'
visitor_2021['Year'] = 2021

visitor_2022 = pd.read_csv(file_paths['visitor_2022'])
visitor_2022['Type'] = 'Visitor'
visitor_2022['Year'] = 2022

visitor_2023 = pd.read_csv(file_paths['visitor_2023'])
visitor_2023['Type'] = 'Visitor'
visitor_2023['Year'] = 2023

# Displaying the first few rows of each dataframe to verify
(home_2021.head(), home_2022.head(), home_2023.head(), visitor_2021.head(), visitor_2022.head(), visitor_2023.head())
```

Appendix B – Data Cleaning and wrangling.

Cleaning and Wrangling

[20]:
```
# Combining all the data into a single DataFrame
combined_stats = pd.concat([home_2021, home_2022, home_2023, visitor_2021, visitor_2022, visitor_2023], ignore_index=True)

# Displaying the first few rows of the combined dataframe
print(combined_stats.head())
```

```
    Rk                  Team     G    MP    FG    FGA    FG%    3P   3PA  \
0  1.0  Minnesota Timberwolves*  82.0  241.2  41.6  91.0  0.457  14.8  41.3
1  2.0       Memphis Grizzlies*  82.0  241.2  43.5  94.4  0.461  11.5  32.7
2  3.0         Milwaukee Bucks*  82.0  240.9  41.8  89.4  0.468  14.1  38.4
3  4.0       Charlotte Hornets*  82.0  242.4  42.8  91.4  0.468  13.9  38.2
4  5.0           Phoenix Suns*  82.0  240.6  43.7  90.1  0.485  11.6  31.9

    3P%  ...   DRB   TRB   AST  STL  BLK   TOV    PF    PTS  Type  Year
0  0.358  ...  32.9  44.2  25.7  8.8  5.6  14.3  21.8  115.9  Home  2021
1  0.353  ...  35.0  49.2  26.0  9.8  6.5  13.2  19.8  115.6  Home  2021
2  0.366  ...  36.5  46.7  23.9  7.6  4.0  13.4  18.2  115.5  Home  2021
3  0.365  ...  33.7  44.6  28.1  8.6  4.9  13.3  19.9  115.3  Home  2021
4  0.364  ...  35.5  45.3  27.4  8.6  4.4  12.9  19.9  114.8  Home  2021

[5 rows x 27 columns]
```

[21]:
```
#Viewing first 20 components of the dataset
combined_stats.head(20)
```

[21]:

	Rk	Team	G	MP	FG	FGA	FG%	3P	3PA	3P%	...	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Type	Year
0	1.0	Minnesota Timberwolves*	82.0	241.2	41.6	91.0	0.457	14.8	41.3	0.358	...	32.9	44.2	25.7	8.8	5.6	14.3	21.8	115.9	Home	2021
1	2.0	Memphis Grizzlies*	82.0	241.2	43.5	94.4	0.461	11.5	32.7	0.353	...	35.0	49.2	26.0	9.8	6.5	13.2	19.8	115.6	Home	2021
2	3.0	Milwaukee Bucks*	82.0	240.9	41.8	89.4	0.468	14.1	38.4	0.366	...	36.5	46.7	23.9	7.6	4.0	13.4	18.2	115.5	Home	2021
3	4.0	Charlotte Hornets	82.0	242.4	42.8	91.4	0.468	13.9	38.2	0.365	...	33.7	44.6	28.1	8.6	4.9	13.3	19.9	115.3	Home	2021
4	5.0	Phoenix Suns*	82.0	240.6	43.7	90.1	0.485	11.6	31.9	0.364	...	35.5	45.3	27.4	8.6	4.4	12.9	19.9	114.8	Home	2021
5	6.0	Atlanta Hawks*	82.0	240.3	41.5	88.3	0.470	12.9	34.4	0.374	...	33.9	44.0	24.6	7.2	4.2	11.9	18.7	113.9	Home	2021
6	7.0	Utah Jazz*	82.0	240.6	40.6	86.2	0.471	14.5	40.3	0.360	...	35.6	46.3	22.4	7.2	4.9	14.0	18.9	113.6	Home	2021
7	8.0	San Antonio Spurs	82.0	241.5	43.2	92.7	0.467	11.3	32.0	0.352	...	34.3	45.3	27.9	7.6	4.9	12.7	18.1	113.2	Home	2021
8	9.0	Brooklyn Nets*	82.0	240.9	42.0	88.4	0.475	11.5	31.7	0.361	...	34.1	44.4	25.3	7.1	5.5	14.1	20.4	112.9	Home	2021
9	10.0	Denver Nuggets*	82.0	241.5	41.7	86.3	0.483	12.7	35.9	0.353	...	34.9	44.1	27.8	7.2	3.7	14.5	20.0	112.7	Home	2021
10	11.0	Los Angeles Lakers	82.0	243.7	41.6	88.8	0.469	12.0	34.5	0.347	...	34.5	44.0	24.0	7.6	5.2	14.5	20.2	112.1	Home	2021
11	12.0	Boston Celtics*	82.0	242.7	40.7	87.4	0.466	13.2	37.1	0.356	...	35.5	46.1	24.8	7.2	5.8	13.6	18.5	111.8	Home	2021
12	13.0	Chicago Bulls*	82.0	240.6	41.7	86.9	0.480	10.6	28.8	0.369	...	33.7	42.3	23.9	7.1	4.1	12.8	18.8	111.6	Home	2021
13	14.0	Indiana Pacers	82.0	242.4	41.4	89.5	0.463	12.2	35.4	0.344	...	32.6	43.9	25.4	7.1	5.6	14.4	20.4	111.5	Home	2021
14	15.0	Golden State Warriors*	82.0	240.6	40.5	86.4	0.469	14.3	39.4	0.364	...	35.7	45.5	27.1	8.8	4.5	14.9	21.0	111.0	Home	2021
15	16.0	Sacramento Kings	82.0	241.5	40.5	88.1	0.460	11.4	33.2	0.344	...	33.4	42.9	23.7	7.2	4.5	14.1	18.9	110.3	Home	2021
16	17.0	Miami Heat*	82.0	242.1	39.6	84.8	0.467	13.6	35.8	0.379	...	33.9	43.7	25.5	7.4	3.2	14.6	20.5	110.0	Home	2021
17	18.0	Philadelphia 76ers*	82.0	241.5	39.4	84.5	0.466	11.6	31.8	0.364	...	33.8	42.3	23.7	7.7	5.3	12.5	19.4	109.9	Home	2021
18	19.0	Houston Rockets	82.0	240.9	39.4	86.4	0.456	13.5	38.7	0.349	...	32.4	42.0	23.6	7.3	4.7	16.5	20.6	109.7	Home	2021
19	20.0	Toronto Raptors*	82.0	242.1	40.6	91.3	0.445	11.9	34.2	0.349	...	32.0	45.3	22.1	9.0	4.6	12.5	19.6	109.4	Home	2021

20 rows × 27 columns

[22]:
```
#Viewing last 20 components of thr dataset
combined_stats.tail(20)
```

[22]:

	Rk	Team	G	MP	FG	FGA	FG%	3P	3PA	3P%	...	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Type	Year
166	12.0	Memphis Grizzlies	82.0	241.2	41.2	86.9	0.474	13.4	35.4	0.378	...	35.0	45.6	26.6	8.1	6.5	15.1	18.8	112.8	Visitor	2023
167	13.0	Houston Rockets	82.0	242.1	40.8	88.1	0.463	12.3	35.4	0.348	...	34.2	44.9	24.4	7.3	5.9	13.8	19.6	113.2	Visitor	2023
168	14.0	Phoenix Suns*	82.0	241.2	42.0	90.6	0.464	13.1	36.0	0.364	...	30.4	41.3	26.3	8.4	4.5	12.7	19.7	113.2	Visitor	2023
169	15.0	Brooklyn Nets	82.0	241.5	41.6	88.5	0.470	13.0	34.9	0.372	...	34.0	44.3	25.5	6.9	5.0	12.6	18.2	113.3	Visitor	2023
170	16.0	Chicago Bulls	82.0	243.7	41.1	87.0	0.473	14.6	39.5	0.370	...	33.3	43.4	27.9	6.8	4.9	14.0	18.8	113.7	Visitor	2023
171	17.0	Sacramento Kings	82.0	242.1	41.6	86.8	0.480	13.1	33.8	0.387	...	33.4	42.5	26.9	7.4	4.5	13.9	18.3	114.8	Visitor	2023
172	18.0	Golden State Warriors	82.0	241.8	42.1	90.4	0.466	13.3	37.0	0.359	...	32.0	42.9	26.7	7.7	5.0	13.0	17.9	115.2	Visitor	2023
173	19.0	Portland Trail Blazers	82.0	242.4	42.5	86.6	0.491	11.7	33.3	0.351	...	33.2	43.8	27.1	8.9	6.4	14.3	17.9	115.4	Visitor	2023
174	20.0	Dallas Mavericks*	82.0	240.3	43.0	90.4	0.475	13.1	35.6	0.368	...	34.1	45.1	27.5	7.4	4.0	13.7	20.3	115.6	Visitor	2023
175	21.0	Milwaukee Bucks*	82.0	241.5	43.2	91.9	0.470	12.6	35.3	0.356	...	33.7	44.0	26.5	7.1	4.2	12.0	19.2	116.4	Visitor	2023
176	22.0	Charlotte Hornets	82.0	240.6	43.4	87.7	0.494	13.6	36.2	0.377	...	34.8	45.4	28.7	7.1	4.8	13.6	17.5	116.8	Visitor	2023
177	23.0	Los Angeles Lakers*	82.0	242.1	44.4	93.7	0.474	14.3	37.9	0.376	...	33.2	44.0	28.2	8.2	4.8	13.4	19.7	117.4	Visitor	2023
178	24.0	San Antonio Spurs	82.0	241.8	44.9	92.3	0.487	12.7	34.0	0.373	...	34.8	45.3	28.0	8.9	4.6	13.4	17.9	118.6	Visitor	2023
179	25.0	Toronto Raptors	82.0	241.5	44.7	91.2	0.491	13.7	36.3	0.376	...	34.0	45.2	28.6	7.3	5.8	13.9	18.1	118.8	Visitor	2023
180	26.0	Detroit Pistons	82.0	240.9	43.6	89.0	0.490	12.1	32.7	0.370	...	33.5	43.1	27.0	8.9	6.0	12.4	17.8	119.0	Visitor	2023
181	27.0	Indiana Pacers*	82.0	240.3	44.5	89.8	0.496	10.7	29.3	0.365	...	32.4	43.4	24.6	6.6	5.4	13.9	18.3	120.2	Visitor	2023
182	28.0	Atlanta Hawks	82.0	242.1	44.6	90.2	0.495	14.0	36.4	0.384	...	33.6	44.2	28.2	7.8	5.6	14.1	19.4	120.5	Visitor	2023
183	29.0	Utah Jazz	82.0	241.5	44.6	91.6	0.487	14.8	37.4	0.395	...	31.3	42.2	29.8	8.6	6.4	12.3	19.2	120.5	Visitor	2023
184	30.0	Washington Wizards	82.0	240.6	46.1	92.9	0.496	12.3	33.9	0.362	...	36.8	48.9	29.0	8.0	6.0	14.0	18.0	123.0	Visitor	2023
185	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Visitor	2023

20 rows × 27 columns

[23]:
```
#Viewing number of columns
len(combined_stats.columns)
```

[23]:
```
27
```

Appendix C – Handling Missing Values and Checking for Duplicates

Handling Missing Values

:
```
# Checking for missing values in each column
missing_values = combined_stats.isnull().sum()
print("Missing values in each column:")
print(missing_values)

# Filling missing values or dropping rows/columns with missing values
combined_stats_cleaned = combined_stats.dropna()

# Verifying that there are no more missing values
print("Missing values after cleaning:")
print(combined_stats_cleaned.isnull().sum())
```

Checking for duplicates

```
# Checking for duplicates
duplicates = combined_stats_cleaned.duplicated().sum()
print(f'Number of duplicate rows: {duplicates}')

# Dropping duplicate rows
combined_stats_cleaned = combined_stats_cleaned.drop_duplicates()

# Verifying that there are no more duplicates
print(f'Number of duplicate rows after cleaning: {combined_stats_cleaned.duplicated().sum()}')
```

Appendix D – Correlation Heatmap for Team Statistics

```
numeric_df = combined_stats_cleaned.select_dtypes(include=[float, int])
```

```
# Computing the correlation matrix
corr_matrix = numeric_df.corr()

# Plotting the heatmap
plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Appendix E – Loading Player Statistics

```
# File paths
file_paths = [
    'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Advanced player stats 2021-2022 season.csv',
    'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Advanced player stats 2022-2023 season.csv',
    'C:/Users/co363/OneDrive - University of Sussex/Dissertation - sports analytics/Advanced player stats 2023-2024 season.csv'
]

# Loading the dataframes with latin1 encoding
dataframes = [pd.read_csv(file, encoding='latin1') for file in file_paths]

# Merging the dataframes
merged_df = pd.concat(dataframes, ignore_index=True)
```

```
# Displaying the first 20 rows of the merged dataframe
merged_df.head(20)
```

Appendix F – Logistic Regression

```
# Defining features and target variable
features = ['MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', 'PF']
target = 'Win'

# Separating features and target from the dataset
X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

# Splitting the data into training (70%) and remaining (30%) sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)

# Further splitting the remaining data into validation (15%) and testing (15%) sets
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Initializing the logistic regression model
logistic_regression = LogisticRegression()

# Training the model
logistic_regression.fit(X_train, y_train)

# Making predictions on the training set
y_train_pred = logistic_regression.predict(X_train)

# Calculating the accuracy of the model on the training set
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f'Training Accuracy: {train_accuracy:.2f}')

# Making predictions on the validation set
y_val_pred = logistic_regression.predict(X_val)

# Calculating the accuracy of the model on the validation set
val_accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {val_accuracy:.2f}')

# Making predictions on the test set
y_test_pred = logistic_regression.predict(X_test)

# Calculating the accuracy of the model on the test set
test_accuracy = accuracy_score(y_test, y_test_pred)
print(f'Test Accuracy: {test_accuracy:.2f}')

# Performing cross-validation and calculating scores
cross_val_scores = cross_val_score(logistic_regression, X_train, y_train, cv=5, scoring='accuracy')
print(f'Cross-validation Scores: {cross_val_scores}')
print(f'Mean Cross-validation Score: {cross_val_scores.mean():.2f}')

# Generating the classification report for the validation set
val_classification_report = classification_report(y_val, y_val_pred)
print(f'Validation Classification Report:\n{val_classification_report}')

# Generating the classification report for the test set
test_classification_report = classification_report(y_test, y_test_pred)
print(f'Test Classification Report:\n{test_classification_report}')

# Confusion matrix for validation set
val_confusion_matrix = confusion_matrix(y_val, y_val_pred)
print(f'Validation Confusion Matrix:\n{val_confusion_matrix}')

# Confusion matrix for test set
test_confusion_matrix = confusion_matrix(y_test, y_test_pred)
print(f'Test Confusion Matrix:\n{test_confusion_matrix}')

# Plotting the confusion matrix for validation set
plt.figure(figsize=(8, 6))
sns.heatmap(val_confusion_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.title('Validation Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Plotting the confusion matrix for test set
plt.figure(figsize=(8, 6))
sns.heatmap(test_confusion_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Calculating the ROC AUC score and plotting the ROC curve for the validation set
val_roc_auc = roc_auc_score(y_val, logistic_regression.predict_proba(X_val)[:, 1])
fpr_val, tpr_val, _ = roc_curve(y_val, logistic_regression.predict_proba(X_val)[:, 1])

# Calculating the ROC AUC score and plotting the ROC curve for the test set
test_roc_auc = roc_auc_score(y_test, logistic_regression.predict_proba(X_test)[:, 1])
fpr_test, tpr_test, _ = roc_curve(y_test, logistic_regression.predict_proba(X_test)[:, 1])

plt.figure()
plt.plot(fpr_val, tpr_val, label=f'Validation ROC curve (area = {val_roc_auc:.2f})')
plt.plot(fpr_test, tpr_test, label=f'Test ROC curve (area = {test_roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()

# Plotting the learning curve
train_sizes, train_scores, val_scores = learning_curve(logistic_regression, X, y, cv=5, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
val_scores_mean = np.mean(val_scores, axis=1)

plt.figure()
plt.plot(train_sizes, train_scores_mean, 'o-', color='r', label='Training score')
plt.plot(train_sizes, val_scores_mean, 'o-', color='g', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.show()
```

Appendix G – Logistic Regression with Hyperparameter Tunning

```
# Defining features and target variable
features = ['MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', 'PF']
target = 'Win'

# Separating features and target from the dataset
X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

# Splitting the data into training (70%) and remaining (30%) sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)

# Further splitting the remaining data into validation (15%) and testing (15%) sets
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Defining the parameter grid
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['lbfgs', 'liblinear'],
    'penalty': ['l2'],  # 'l2' is the default penalty, 'l1' is not supported by 'lbfgs'
    'max_iter': [100, 200, 300]
}

# Initializing the logistic regression model
logistic_regression = LogisticRegression()

# Initializing GridSearchCV with the logistic regression model and parameter grid
grid_search = GridSearchCV(estimator=logistic_regression, param_grid=param_grid, cv=5, scoring='accuracy')

# Fitting GridSearchCV to the training data
grid_search.fit(X_train, y_train)

# Extracting the best model
best_model = grid_search.best_estimator_

# Printing the best parameters
print(f'Best Parameters: {grid_search.best_params_}')

# Calculating the training accuracy with the best model
y_train_pred_best = best_model.predict(X_train)
train_accuracy_best = accuracy_score(y_train, y_train_pred_best)
print(f'Training Accuracy with Best Model: {train_accuracy_best:.2f}')

# Evaluating the best model on the validation set
y_val_pred_best = best_model.predict(X_val)
val_accuracy_best = accuracy_score(y_val, y_val_pred_best)
print(f'Validation Accuracy with Best Model: {val_accuracy_best:.2f}')

# Generating the classification report for the validation set with the best model
val_classification_report_best = classification_report(y_val, y_val_pred_best)
print(f'Validation Classification Report with Best Model:\n{val_classification_report_best}')

# Making predictions on the test set with the best model
y_test_pred_best = best_model.predict(X_test)

# Calculating the accuracy of the best model on the test set
test_accuracy_best = accuracy_score(y_test, y_test_pred_best)
print(f'Test Accuracy with Best Model: {test_accuracy_best:.2f}')

# Generating the classification report for the test set with the best model
test_classification_report_best = classification_report(y_test, y_test_pred_best)
print(f'Test Classification Report with Best Model:\n{test_classification_report_best}')

# Confusion matrix for validation set
val_confusion_matrix = confusion_matrix(y_val, y_val_pred_best)
print(f'Validation Confusion Matrix:\n{val_confusion_matrix}')

# Confusion matrix for test set
test_confusion_matrix = confusion_matrix(y_test, y_test_pred_best)
print(f'Test Confusion Matrix:\n{test_confusion_matrix}')

# Plotting the confusion matrix for validation set
plt.figure(figsize=(8, 6))
sns.heatmap(val_confusion_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.title('Validation Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Plotting the confusion matrix for test set
plt.figure(figsize=(8, 6))
sns.heatmap(test_confusion_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Calculating the ROC AUC score and plotting the ROC curve for the validation set with the best model
val_roc_auc_best = roc_auc_score(y_val, best_model.predict_proba(X_val)[:, 1])
fpr_val_best, tpr_val_best, _ = roc_curve(y_val, best_model.predict_proba(X_val)[:, 1])

# Calculating the ROC AUC score and plotting the ROC curve for the test set with the best model
test_roc_auc_best = roc_auc_score(y_test, best_model.predict_proba(X_test)[:, 1])
fpr_test_best, tpr_test_best, _ = roc_curve(y_test, best_model.predict_proba(X_test)[:, 1])

plt.figure()
plt.plot(fpr_val_best, tpr_val_best, label=f'Validation ROC curve (area = {val_roc_auc_best:.2f})')
plt.plot(fpr_test_best, tpr_test_best, label=f'Test ROC curve (area = {test_roc_auc_best:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic with Best Model')
plt.legend(loc='lower right')
plt.show()

# Performing cross-validation and calculating scores
cross_val_scores = cross_val_score(best_model, X_train, y_train, cv=5, scoring='accuracy')
print(f'Cross-validation Scores: {cross_val_scores}')
print(f'Mean Cross-validation Score: {cross_val_scores.mean():.2f}')

# Plotting the learning curve for the best model
train_sizes_best, train_scores_best, val_scores_best = learning_curve(best_model, X, y, cv=5, scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean_best = np.mean(train_scores_best, axis=1)
val_scores_mean_best = np.mean(val_scores_best, axis=1)

plt.figure()
plt.plot(train_sizes_best, train_scores_mean_best, 'o-', color='r', label='Training score')
plt.plot(train_sizes_best, val_scores_mean_best, 'o-', color='g', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve with Best Model')
plt.legend(loc='best')
plt.show()
```

Appendix H – Random Forest

```
features = ['MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', 'PF']
target = 'Win'

X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

# Splitting the dataset: 70% train, 15% validation, 15% test
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50, random_state=42, stratify=y_temp)

# Standardizing the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Training the Random Forest model
random_forest = RandomForestClassifier(random_state=42)
random_forest.fit(X_train_scaled, y_train)

# Predicting and evaluating on validation and test sets
y_val_pred = random_forest.predict(X_val_scaled)
y_test_pred = random_forest.predict(X_test_scaled)

# Accuracy scores
train_accuracy = accuracy_score(y_train, random_forest.predict(X_train_scaled))
val_accuracy = accuracy_score(y_val, y_val_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f'Training Accuracy: {train_accuracy:.2f}')
print(f'Validation Accuracy: {val_accuracy:.2f}')
print(f'Test Accuracy: {test_accuracy:.2f}')

# Cross-validation scores
cross_val_scores = cross_val_score(random_forest, X_train_scaled, y_train, cv=5)
print(f'Cross-validation scores: {cross_val_scores}')
print(f'Average cross-validation score: {np.mean(cross_val_scores):.2f}')

# Confusion matrix
conf_matrix_val = confusion_matrix(y_val, y_val_pred)
conf_matrix_test = confusion_matrix(y_test, y_test_pred)
print('Confusion Matrix (Validation):')
print(conf_matrix_val)
print('Confusion Matrix (Test):')
print(conf_matrix_test)

# Classification report
class_report_val = classification_report(y_val, y_val_pred)
class_report_test = classification_report(y_test, y_test_pred)
print('Classification Report (Validation):')
print(class_report_val)
print('Classification Report (Test):')
print(class_report_test)

# ROC AUC curve
fpr_val, tpr_val, _ = roc_curve(y_val, random_forest.predict_proba(X_val_scaled)[:,1])
roc_auc_val = auc(fpr_val, tpr_val)

fpr_test, tpr_test, _ = roc_curve(y_test, random_forest.predict_proba(X_test_scaled)[:,1])
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(12, 6))
plt.plot(fpr_val, tpr_val, color='blue', lw=2, label=f'Validation ROC curve (area = {roc_auc_val:.2f})')
plt.plot(fpr_test, tpr_test, color='red', lw=2, label=f'Test ROC curve (area = {roc_auc_test:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()

# Learning curve
train_sizes, train_scores, val_scores = learning_curve(random_forest, X_train_scaled, y_train, cv=5, n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
val_scores_mean = np.mean(val_scores, axis=1)

plt.figure(figsize=(12, 6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='blue', label='Training score')
plt.plot(train_sizes, val_scores_mean, 'o-', color='red', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.grid()
plt.show()

# Plotting the confusion matrix for validation set
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_val, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Validation)')
plt.show()

# Plotting the confusion matrix for test set
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Test)')
plt.show()
```

Appendix I – Random Forest with Hyperparameter Tunning

```
features = ['MP', 'FG%', '3P%', '2PA', 'FT', 'FT%', 'DRB', 'TOV', 'PF']
target = 'Win'

X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

# Splitting the dataset: 70% train, 15% validation, 15% test
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, random_state=42, stratify=y)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50, random_state=42, stratify=y_temp)

# Standardizing the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Hyperparameter tuning with RandomizedSearchCV
param_distributions = {
    'n_estimators': randint(50, 200),  # Reduced number of trees
    'max_depth': [None, 5, 10, 15],  # Limited depth
    'min_samples_split': randint(10, 40),  # Further increased minimum samples to split
    'min_samples_leaf': randint(10, 40),  # Further increased minimum samples per leaf
    'max_features': ['sqrt', 'log2', None],  # Restricting features considered at each split
    'bootstrap': [True, False]
}

random_search = RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
                                   param_distributions=param_distributions,
                                   n_iter=100,
                                   cv=5,
                                   verbose=2,
                                   n_jobs=-1,
                                   random_state=42)

random_search.fit(X_train_scaled, y_train)

# Best estimator from random search
best_rf = random_search.best_estimator_

# Predicting and evaluating on validation and test sets
y_val_pred = best_rf.predict(X_val_scaled)
y_test_pred = best_rf.predict(X_test_scaled)

# Accuracy scores
train_accuracy = accuracy_score(y_train, best_rf.predict(X_train_scaled))
val_accuracy = accuracy_score(y_val, y_val_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f'Training Accuracy: {train_accuracy:.2f}')
print(f'Validation Accuracy: {val_accuracy:.2f}')
print(f'Test Accuracy: {test_accuracy:.2f}')

# Cross-validation scores
cross_val_scores = cross_val_score(best_rf, X_train_scaled, y_train, cv=5)
print(f'Cross-validation scores: {cross_val_scores}')
print(f'Average cross-validation score: {np.mean(cross_val_scores):.2f}')

# Confusion matrix
conf_matrix_val = confusion_matrix(y_val, y_val_pred)
conf_matrix_test = confusion_matrix(y_test, y_test_pred)
print('Confusion Matrix (Validation):')
print(conf_matrix_val)
print('Confusion Matrix (Test):')
print(conf_matrix_test)

# Classification report
class_report_val = classification_report(y_val, y_val_pred)
class_report_test = classification_report(y_test, y_test_pred)
print('Classification Report (Validation):')
print(class_report_val)
print('Classification Report (Test):')
print(class_report_test)

# ROC AUC curve
fpr_val, tpr_val, _ = roc_curve(y_val, best_rf.predict_proba(X_val_scaled)[:, 1])
roc_auc_val = auc(fpr_val, tpr_val)

fpr_test, tpr_test, _ = roc_curve(y_test, best_rf.predict_proba(X_test_scaled)[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(12, 6))
plt.plot(fpr_val, tpr_val, color='blue', lw=2, label=f'Validation ROC curve (area = {roc_auc_val:.2f})')
plt.plot(fpr_test, tpr_test, color='red', lw=2, label=f'Test ROC curve (area = {roc_auc_test:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()

# Learning curve
train_sizes, train_scores, val_scores = learning_curve(best_rf, X_train_scaled, y_train, cv=5, n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
val_scores_mean = np.mean(val_scores, axis=1)

plt.figure(figsize=(12, 6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='blue', label='Training score')
plt.plot(train_sizes, val_scores_mean, 'o-', color='red', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.grid()
plt.show()

# Plotting the confusion matrix for validation set
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_val, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Validation)')
plt.show()

# Plotting the confusion matrix for test set
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (Test)')
plt.show()
```

Appendix J – Multilayer Perceptron

```
# Define features and target
features = ['MP', 'FG', 'FGA', 'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF',
target = 'Win'

# Splitting the dataset into training, testing, and validation sets
X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Training the MLPClassifier with increased regularization and early stopping
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=500, solver='adam', random_state=42, alpha=0.01, early_stopping=True, validation_fraction=0.1, n_i
mlp.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = mlp.predict(X_train_scaled)
y_val_pred = mlp.predict(X_val_scaled)
y_test_pred = mlp.predict(X_test_scaled)

# Accuracy Scores
train_accuracy = accuracy_score(y_train, y_train_pred)
val_accuracy = accuracy_score(y_val, y_val_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

# Printing Accuracy Scores
print(f'Training Accuracy: {train_accuracy:.2f}')
print(f'Validation Accuracy: {val_accuracy:.2f}')
print(f'Test Accuracy: {test_accuracy:.2f}')

# Confusion Matrices and Classification Reports
conf_matrix_val = confusion_matrix(y_val, y_val_pred)
conf_matrix_test = confusion_matrix(y_test, y_test_pred)
class_report_val = classification_report(y_val, y_val_pred)
class_report_test = classification_report(y_test, y_test_pred)

print('Validation Confusion Matrix:')
print(conf_matrix_val)
print('Validation Classification Report:')
print(class_report_val)

print('Test Confusion Matrix:')
print(conf_matrix_test)
print('Test Classification Report:')
print(class_report_test)

# Plotting Confusion Matrices
plt.figure(figsize=(14,6))
plt.subplot(1, 2, 1)
sns.heatmap(conf_matrix_val, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Validation Confusion Matrix')

plt.subplot(1, 2, 2)
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Test Confusion Matrix')

plt.tight_layout()
plt.show()

# Cross-validation scores
cv_scores = cross_val_score(mlp, X_train_scaled, y_train, cv=5)
print(f'Cross-validation scores: {cv_scores}')
print(f'Mean CV score: {cv_scores.mean():.2f}')

# ROC Curve
y_val_proba = mlp.predict_proba(X_val_scaled)[:, 1]
y_test_proba = mlp.predict_proba(X_test_scaled)[:, 1]

fpr_val, tpr_val, _ = roc_curve(y_val, y_val_proba)
roc_auc_val = auc(fpr_val, tpr_val)

fpr_test, tpr_test, _ = roc_curve(y_test, y_test_proba)
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(8,6))
plt.plot(fpr_val, tpr_val, color='blue', lw=2, label=f'Validation ROC curve (area = {roc_auc_val:.2f})')
plt.plot(fpr_test, tpr_test, color='red', lw=2, label=f'Test ROC curve (area = {roc_auc_test:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# Learning Curve
train_sizes, train_scores, val_scores = learning_curve(mlp, X_train_scaled, y_train, cv=5, n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
val_scores_mean = np.mean(val_scores, axis=1)

plt.figure(figsize=(8,6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='blue', label='Training score')
plt.plot(train_sizes, val_scores_mean, 'o-', color='red', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.show()
```

Appendix K - Multilayer Perceptron with Hyperparameter Tunning

```
# Defining features and target
features = ['MP', 'FG', 'FGA', 'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF',
target = 'Win'


X = combined_stats_cleaned[features]
y = combined_stats_cleaned[target]

# Splitting the dataset into training, testing, and validation sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Scaling the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Defining the parameter grid for hyperparameter tuning
param_grid = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['constant', 'adaptive'],
}

# Initializing the MLPClassifier
mlp = MLPClassifier(max_iter=500, random_state=42, early_stopping=True, validation_fraction=0.1, n_iter_no_change=10)

# Performing GridSearchCV
grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train_scaled, y_train)

# Getting the best parameters and estimator
best_params = grid_search.best_params_
best_mlp = grid_search.best_estimator_

print(f"Best parameters found: {best_params}")

# Predictions with the best estimator
y_train_pred = best_mlp.predict(X_train_scaled)
y_val_pred = best_mlp.predict(X_val_scaled)
y_test_pred = best_mlp.predict(X_test_scaled)

# Accuracy Scores
train_accuracy = accuracy_score(y_train, y_train_pred)
val_accuracy = accuracy_score(y_val, y_val_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

# Printing Accuracy Scores
print(f'Training Accuracy: {train_accuracy:.2f}')
print(f'Validation Accuracy: {val_accuracy:.2f}')
print(f'Test Accuracy: {test_accuracy:.2f}')

# Confusion Matrix and Classification Report for validation set
conf_matrix_val = confusion_matrix(y_val, y_val_pred)
class_report_val = classification_report(y_val, y_val_pred)

print('Validation Confusion Matrix:')
print(conf_matrix_val)
print('Validation Classification Report:')
print(class_report_val)

# Confusion Matrix and Classification Report for test set
conf_matrix_test = confusion_matrix(y_test, y_test_pred)
class_report_test = classification_report(y_test, y_test_pred)

print('Test Confusion Matrix:')
print(conf_matrix_test)
print('Test Classification Report:')
print(class_report_test)

# Plotting Confusion Matrix for validation set
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_val, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Validation Confusion Matrix')
plt.show()

# Plotting Confusion Matrix for test set
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix_test, annot=True, fmt='d', cmap='Blues', xticklabels=['Loss', 'Win'], yticklabels=['Loss', 'Win'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Test Confusion Matrix')
plt.show()

# ROC Curve
y_val_proba = best_mlp.predict_proba(X_val_scaled)[:, 1]
y_test_proba = best_mlp.predict_proba(X_test_scaled)[:, 1]

fpr_val, tpr_val, _ = roc_curve(y_val, y_val_proba)
roc_auc_val = auc(fpr_val, tpr_val)

fpr_test, tpr_test, _ = roc_curve(y_test, y_test_proba)
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(8,6))
plt.plot(fpr_val, tpr_val, color='blue', lw=2, label=f'Validation ROC curve (area = {roc_auc_val:.2f})')
plt.plot(fpr_test, tpr_test, color='red', lw=2, label=f'Test ROC curve (area = {roc_auc_test:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# Learning Curve
train_sizes, train_scores, val_scores = learning_curve(best_mlp, X_train_scaled, y_train, cv=5, n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
val_scores_mean = np.mean(val_scores, axis=1)

plt.figure(figsize=(8,6))
plt.plot(train_sizes, train_scores_mean, 'o-', color='blue', label='Training score')
plt.plot(train_sizes, val_scores_mean, 'o-', color='red', label='Cross-validation score')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.show()
```