ADTA - 5940

Advanced Data Analytics Capstone Project

Professor – Dr. Denise R. Philpot

Group 24

Vamshi Krishna Vallam

Kowshik Mannam

Pavankumar Jasti

Lawrence Abhinav Gunkonda

Table of Contents

Vamshi Krishna Vallam, Kowshik Mannam, Pavankumar Jasti, Lawrence Abhinav Gunkonda

NBA Prediction

CHAPTER 1: INTRODUCTION

The National Basketball Association (NBA) was started in June 1946 in New York City. It is a professional basketball tournament in North America and later it became a popular sport in the world. There are audiences worldwide who follow the NBA. In 1946 when the NBA was invented, they named it Basketball Association of America (BAA). Later the name was changed to the National Basketball Association (NBA). NBA is compiled with 30 teams of which 29 teams are from North America and one team is from Canada.

The National Basketball Association (NBA) is one of the best and most followed professional basketball leagues throughout the world. The NBA is known for its exciting games and the presence of elite players attracts audiences around the world. The gameplay consists of spectacular dunks, amazing three-pointers, and magnificent athleticism of players, the composite and varied statistics of the game and the players, strategy, and analysis of the matches. Analyzing an NBA match is a structured assessment of different features of the league, its teams, and players, providing data-driven insights to understand, predict, and enhance the performance of the players and the teams.

The exponential growth of NBA analytics in recent years because of the availability of modern statistics, advanced technology, and growth in the analysis by the game analysts, coaches, teams, and fans have peaked higher into statistics, which is aimed at finding out the hidden patterns and have a piece of prior knowledge about the competitor corners. This analysis will help the teams to make better decisions and allow the sports enthusiasts and viewers to experience the game in a whole new exciting way.

NBA analytics covers a broad range of points which includes players' performance points, team kinetics, wage cap, and much more. NBA analytics is one of the most important reasons for the huge recognition and success of NBA across the world and it also helps the NBA franchises with their popularity. With the availability of big data and analytics across different industries, the use of NBA analytics has immensely changed the perspective of how we understand and experience gameplay, which makes it an exciting convergence of technology, statistics, and sports.

There are a few analytical systems that are used to examine the positives and negatives of the teams and their players. NBA Elo is one of them, which measures the strength of the after every game. NBA Elo uses different statistical analyses to predict the NBA match outcomes by offering different numerical points for the team’s performance. Another analytical rating system is CARMELO (Career- Arc Regression Model Estimator with Local Optimization) which can predict a team’s and a player’s and a team’s probable future performance. This metric provides a calculation of the number of points per 100 possessions that the player contributed to the match for his team. A recent analytical rating system RAPTOR which stands for Robust Algorithm Player Tracking and on/off Ratings is also used to examine the players’ and teams' performance.

Objective

The NBA has a wealthy and legendary history, ranging over several decades of exciting and thrilling games, renowned players, and special memorable moments. In this research, we will be delving deep into the world of advanced statistics and numbers where we will be focusing on how numbers play an important role in producing insights and understanding of a. The importance of numbers and statistics will help the world of sports to make better decisions and choices for the teams. This study of statistics and numbers will be a great revolution in the sporting industry where the analysts, team coaches, and players will hugely benefit from it and excel in their relative areas.

Let us walk through this interesting research of the NBA matches by using the magic of data to open its powers and acknowledge its everlasting impact.

Research Questions

1) Rating Accuracy for Matches:
    - Evaluate the accuracy of Elo ratings, CARMELO ratings, and RAPTOR ratings in predicting the outcomes of matches.
2) Match Quality's Impact on Importance and Total Rating:
    - Analyze whether the quality of a match, as indicated by various factors, significantly influences its importance and the overall rating.
3) Influence of Match Quality on Match Outcomes:
    - Investigate how the quality of a match, as determined by specific metrics, affects the final match outcome for the teams.
4) Neutral Venue vs. Non-Neutral Venue Matches:
    - Compare the differences in match outcomes (score 1 vs. score 2) between neutral venue and non-neutral venue matches for the Teams.
5) Team Performance Variations in Neutral Venues:
    - Examine whether there are substantial variations in the performance of the teams in neutral venues compared to non-neutral venues.
6) Team Ratings During COVID-19:
    - Explore the impact of the COVID-19 pandemic on the Elo, CARMELO, and RAPTOR ratings for the Teams, and understand how this period affected their performance.

CHAPTER 2: LITERATURE REVIEW

Introduction

NBA History

The National Basketball Association (NBA) is the professional basketball league in North America. The league was formed in 1946 and has 30 clubs (29 in the United States and 1 in Canada). The NBA is one of the most popular sports leagues in the world, with 1 billion fans worldwide. The NBA has had a huge business impact. The NBA, which generates billions of dollars annually for the league from ticket sales, merchandise, television and radio broadcasting rights, and sponsorships, has also helped boost the economies of the cities where its teams are located. There are many reasons for the success of the NBA. *(Badenhausen, 2018)*

- Popularity of basketball around the world.
- League's global marketing outreach.
- The best players and coaches in the NBA.
- Exciting and competitive NBA games.
- NBA's commitment to social responsibility

Industries benefitting from NBA:

- Apparel and sporting goods companies: The NBA is one of the most popular sports in the world, and fans love their teams. This has led to a strong demand for apparel and sporting goods emblazoned with the NBA logo. *(NBA, 2008)*
- Media companies: The NBA's television and radio rights are worth billions of dollars a year. This has led to a significant increase in revenue for media companies that broadcast NBA games.
- Food and Beverage Companies: NBA games are often social, and fans enjoy food and

drinks while watching the game. This has led to a significant increase in revenue for food and beverage companies that sell their products on NBA courts and television.

- Tourism: The NBA draws fans from all over the world to its games. This has significantly increased tourism revenue in NBA Tea host cities where NBA Teams are located.

Teams

The NBA's journey began back in 1946, and it's come a long way since. Starting with just11 teams, it now boasts a total of 30 teams, having gone through a series of expansions, reductions,and even a relocation or two. Out of these 30 teams, 29 call the United States home, while one proudly represents Canada.

In its current structure, the league organizes these 30 teams into two conferences, each further divided into three divisions, with five teams in each division. This arrangement, as we knowit today, was established in the 2004-05 season.

If we look at where these teams are based, it's no surprise that the majority are in the easternhalf of the United States. Specifically, 13 teams are in the Eastern Time Zone, while nine operatein the Central Time Zone. The remaining three are situated in the Mountain Time Zone, and five can be found in the Pacific Time Zone. This alignment broadly reflects the population distributionof both the United States and Canada as a whole.

Viewership Demographics

Back in 2013, a survey by Nielsen revealed that the NBA had the youngest audience amongall sports, with a whopping 45 percent of its viewers being under the age of 35. However, fast forward to 2022, and we find that the league still struggles to attract female viewers, as they makeup only 30% of the total audience. (*Piacenza, Joanna (January 25, 2018))*

In 2014, the NBA had a diverse viewership, with 45 percent of its fans identifying as black and 40 percent as white. Notably, it was the only major North American sport without a white majority audience.

As of 2017, there was a significant shift in the NBA's viewership demographics. The proportion of White American viewers dropped to just 34% during the 2016-17 season. In contrast,the number of black viewers increased to 47%, while Hispanic viewers (of any race) accounted for11%, and Asian viewership stood at 8%. Interestingly, the NBA was found to be more favored byDemocrats than Republicans, according to the same poll. (*The Guardian. October 12, 2019. Retrieved December 2, 2020.*)

On a global scale, the NBA found its biggest international market in China, where an astounding 800 million viewers tuned in to watch the 2017-18 season. The NBA's presence in China is massive, with NBA China alone being estimated to be worth a staggering $4 billion. *Spring, Jay (September 10, 2017).*

Impact of COVID 19 on NBA

The 2019-2020 season faced an unprecedented disruption when it was suspended indefinitely on March 11, 2020, due to the COVID-19 pandemic. The NBA Board of Governors decided to resume the season in a unique 22-team format with seeding games and playoffs all heldwithin a "bubble" at Walt Disney World, with no fans in attendance. During this period, the NBA saw a decline in viewership, with a significant drop in the number of viewers, losing 40 to 45 percent of its audience. While some attribute this decline to the trend of "cable-cutting," other major leagues like the NFL and MLB maintained their viewership.

The 2020 NBA Finals' opening game between the Los Angeles Lakersand the Miami Heat had the lowest viewership since 1994, with only 7.41 million viewers, markinga 45 percent decline from the previous year's Finals. Various factors have been cited for this decline, including the league's and players' political stances, player load management, imbalanced talent distribution between conferences, and the shift toward streaming services among younger viewers. *(Cacciola, Scott; Deb, Sopan (March 11, 2020)).*

Rating Systems in NBA

i) The ELO rating system

The Elo rating system was developed in the mid-1900s for the United States Chess Federation to provide accurate methods to track player skills. The Elo rating system is a statistical approach that uses factors in the strength of opponents. Each player is assigned a rating before the match and that rating gets updated after each match. When a lower-ranked player defeats a higher-ranked opponent, there is a significant increase in the rating compared to defeating an opponent of similar rank. While applying the ELO system to basketball, its implementation can vary due to the complexities of the game, particularly the unique impact each player contributed to the team's performance. *(Hatton, E. Predictive Models for NBA Sports Gambling)*

ii) The CARMELO rating system

The statisticians at FiveThirtyEight introduced a career projection tool known as CARMELO. This tool collects players' shooting percentages, wins above replacement (WAR), and/or minus scores to identify historical players with similar attributes to the player being assessed. Analyzingthis data using CARMELO provides a projection of the player's performance range for minutes played.

However, CARMELO ratings are only for players who met certain playing timethresholds in the years 2013-2014 and 2014-2015 seasons, which restricts its scope. CARMELO's WAR projections are important in providing useful comparisons but aren't directly convertible intoactual win figures, which makes them less suitable for assessing individual contributions. CARMELO favors certain players by lacking sufficient data to compare others. Stephen Curry of the Golden State Warriors team has a unique style of play which makes the historical comparisons inadequate and leads to potentially incorrect career projections. *(Silver & Paine, 2015)*

iii) The RAPTOR rating system

Another rating system introduced by FiveThirtyEight's statisticians is RAPTOR (Robust Algorithm using Player Tracking On/Off Ratings). It is a system that the NBA has invested heavily in tracking the player's performance. RAPTOR provides a player's offensive and defensive scores compared to the league's average. Players who perform well in their team's success in the game receive higher ratings. RAPTOR system values skills such as floor spacing, defense, and shot creation, which shows the players who have more possession of the ball will have a higher rating. While predicting the game outcomes RAPTOR combines the player ratings with human inputs such as injury updates, depth charts, and playing time which creates a weighted average of the player ratings for both teams. These ratings are then used to predict the game outcomes. RAPTOR's inaugural season prediction for 2019-20 proved to be very accurate compared to the other prediction systems. RAPTOR's performance was too good and it overperformed sportsbooks for 16 out of the 30 teams in NBA. RAPTOR's performance from the betting perspective didn't see a substantial improvement, as it predicted 18 out of the 30 bets placed. (*Hatton, E. Predictive Models for NBA Sports Gambling*)

A review of machine learning approaches for NBA analytics

When it comes to predicting the outcomes in the world of the NBA, there are essentially two main approaches to play. The first one falls into the category of supervised machine learning,specifically classification methods. In this approach, a human researcher takes the lead, selecting a specific event or action that they want to find in the data. Then, a classification model is put to work, essentially acting as a detective to automatically spot all instances of that chosen event within the dataset.

On the other side, we have unsupervised machine learning, which includes techniques like clustering. In this method, there's no predetermined event chosen by a human. This approach works on finding out the patterns that are already existing within the data. These patterns provide a lot of insights and there is the presence of hidden strategies in this data.

Both the supervised and unsupervised machine learning algorithms execution are different, but both are important in the world of machine learning. Together these machine learning algorithms can do wonders while focusing on the key takeaways and are important techniques to find the patterns and trends in the NBA data.

(*F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V.Dubourg, et al. 2011.)*

Supervised Learning

Supervised learning is like teaching a computer to recognize things. Imagine you have a collection of items, and each item is labeled with a category or name. This labeled data is used to train the computer, so it can learn how to categorize new, unlabeled items.

The process of supervised learning has two important steps:

1. Training: During this stage, the computer goes through the labeled items and figures out the patterns and relationships between the items and their labels. It's like showing the computer various examples and helping it learn the ropes.

2. Testing: After the training, the computer is put to the test. It takes what it learned and triesto correctly label new, unlabeled items. This step helps us understand how accurate the computer's learning process is. (*Y. Reich and S. J. Fenves. 1991*.)

There are different ways to teach the computer – various algorithms – each with its approachto learning and categorizing things. It's like using different teaching methods to help the computer become a better "student."

Unsupervised Learning

Unsupervised learning, also known as clustering, is a more independent and exploratory side of machine learning. Unlike supervised learning where the computer needs a lot of guidance, unsupervised learning does its thing with minimal to no human input. It's like giving the computer a bunch of data and saying, "Figure out how to group these in a way that makes sense."

What this method does is look for natural patterns in the data and then create groups of things that are like each other. It does this by calculating how close or far apart things are, kind of like measuring the distance between them, often using a method called Euclidean distance.

Now, when it comes to forming these groups, there are a few ways to do it. In some cases, items are put into just one group, and that's it. In other cases, an item might belong to more than one group because it has qualities that fit into different categories. Think of it like sorting a big collection of objects into different boxes, and sometimes one item could belong in multiple boxes at the same time.

There's also something called probabilistic clustering, where an item isn't strictly in one group or another; instead, it might belong to a group with a certain probability, kind of like how likely it isto belong there.

In some cases, these groups can get even more detailed. It's like starting with a big category and then breaking it down into smaller and smaller subcategories, forming a hierarchy of clusters.

I. H. Witten and E. Frank. (2002)

Can the NBA game outcome be predicted?

Over the years, there's been quite a bit of research into using machine learning and deep learning to predict outcomes in various sports. One of the early examples was IBM's Advanced Scout in the 1990s, which aimed to uncover hidden patterns in basketball statistics using data mining techniques. This tool used a method called Attribute Focusing to compare attributes across different subsets of data. If a subset had a notably different distribution, it was considered "interesting." However, it didn't offer much explanation for players' stats beyond detecting anomalies. *(Leung and Joseph, 2014)*

More recently, Hidden Markov Models (HMMs) were employed to predict match results by leveraging advanced statistics from NBA games. Accuracy reached a respectable 73%. In the world of football, a Bayesian approach combined with Markovian chains and the Monte-Carlo

method was used to predict game results. However, it relied on neural networks, which made it less interpretable and not suitable for performance analysis or feedback. (*Maher, 1982*)

Another approach involved a Bayesian hierarchical model to predict football results basedon scoring intensity data, considering the attack and defense strengths of the teams involved. In the context of college football games, a data mining technique was proposed to predict game outcomes and unearth valuable insights. This method looked at the historical results of games andpredicted the outcome between two teams by comparing them to similar teams and analyzing theirpast matchups. (*Zimmermann et al., 2013)*

Some researchers explored machine learning techniques specialized in classifier learning to predict the outcomes of the NCAAB matches. Surprisingly, a multi-layer perceptron, which was not widely used at the time, proved to be the most effective. Even more surprising was that there seemed to be a predictive accuracy threshold of around 74% that couldn't be exceeded by either machine learning or statistical techniques. (*McCabe, & Trevathan, 2008*)

Deep learning, specifically Artificial Neural Networks, gained popularity in sports prediction. For instance, a National Football League (NFL) team prediction model achieved an accuracy of 75% (*Kahn, 2003)*, outperforming domain experts by almost 10%. Similar models were built for various sports, such as Rugby League, Australian Rules football, Rugby Union, andEnglish Premier League Football, achieving an average accuracy of 67.5%.

CHAPTER 3: DATA

FiveThirtyEight, a renowned data-driven news, and sports analysis platform has meticulously collected and organized a wealth of data related to the NBA. This all-inclusive dataset consists of different variables such as the team's performance metrics and statistics, match outcomes, and much more. With this vast information in our hands, we will be diving into the numbers and revealing the unseen history of the NBA games that have sculpted the league's dominance in the world.

Detailed description of each type of variable present in the dataset.

Elo ratings

These ratings are a calculation of a team's robustness based on results of head-to-head matches, the points difference between two teams, the quality of the game played, and the level of the opponent team. FiveThirtyEight contains these ratings for every team which will be used for NBA predictions. If a weaker team beats a stronger team, then they will gain more Elo points. If the victory is by a larger margin, then again, the team gets higher Elo ratings.

RAPTOR ratings

These ratings are based on the estimate of a player's performance based on the track record of similar players in the league. FiveThirtyEight uses these ratings to provide its player estimations, which can be used to give rise to the team's talent approximate. RAPTOR ratings use a variety of score calculations in the game such as the points scored, and assists. rebounds, blocks, and steals. RAPTOR ratings make use of tracking the player data to calculate a variety of metrics which include the type of shot taken, distance of the shot, type of the shot, and defensive positioning of the player.

Game metrics

Game metrics: There are a variety of game metrics in FiveThirtyEight, which includes quality, and importance. These metrics help in identifying the importance of each game and help in producing the team ratings for Fair play.

This dataset is provided by the FiveThirtyEight website which focuses on providing data for the sporting industry. This data includes different information about the NBA, which are:

1. Team Statistics: This provides information about different NBA teams present in the league, how these teams performed over time, and what their metrics ratings are.

2. Elo Ratings: These are the metric system that a team has which focuses on the team's strength based on the match outcomes, the display of the game by the opponents, and much more.

3. CARMELO Ratings: It is a more advanced rating system that focuses on a team's performance over the years based on a variety of factors such as player's ball possession, scoring time, points scored, and score differences in the match.

4. RAPTOR Ratings: It is a rating system that aims to analyze players' performance on the court. These ratings can be used for both players and the teams.

5. Match Results: The match results are the outcome between two teams, which consists of the final points, the winning team, and the losing team. There is a specific date associated with each game.

6. Historical Data: This is the data that indicates the presence of all the metrics and the team's performance over different seasons which allows the analysts to keep an eye on the patterns and changes in the NBA league games over time.

7. Match Quality Metrics: These quality indicators are used to figure out how important is the game as they are used to examine different factors such as the team's historical records, and the stake of the current match.

8. Venue Information: This data provides information on whether the match is played at a neutral venue or a non-neutral venue which might impact game results and the audience.

CHAPTER 4: METHODOLOGY

Software and Tools

a) Anaconda: Anaconda is a free, open-source platform in which we can write the code and execute using the Python language and its well-known software which allows us to create different versions of Python code, also to learn and use Python for scientific computing, data science, and machine learning. It can also be used to install and remove many packages from the project environments. It is by continuum.io, a company that specializes in Python development.

b) Jupyter Notebook: Jupyter Notebook (formerly known as IPython Notebook) is a web-based tool for generating and sharing computational papers. It was first named IPython but in 2014, renamed Jupyter. It is a fully open-source product, and users can use every feature for free. It supports more than 40 coding languages including Python, R, and Scala.

c) Python: Python is a high-level programming language. It has code blocks that are used for indentation and readability of code design. Python is dynamically typed, and garbage collected. It supports multiple paradigms, including structured, object-oriented, and functional programming.

d) Numpy: NumPy is a library for the Python language, that allows to addition of large, multi-dimensional arrays and matrices, along with a large collection of high-level functions to operate on these arrays.

e) Pandas: Pandas are open-source packages for Python language which are mostly used in data science/data analysis and machine learning projects. It is built on another library named Numpy, which gives support to multi-dimensional arrays.

f) scikit-learn (sklearn): Scikit-learn is an open-source library for analyzing the data, and the ultimate level for Machine Learning (ML) in the Python environment. Key concepts and features include Algorithmic decision-making methods, including Classification: identifying and categorizing data based on patterns.

g) Matplotlib: It is a visualization library in Python for plotting 2D arrays. It is a multi-platform data visualization library made using NumPy arrays and designed to work with the broader SciPy stack. In the year 2002, It was developed by John Hunter. One of the greatest benefits of visualization is that it allows visual access to big amounts of data in an easier way for visuals. It also involves many plots like bar, scatter, line, histogram, etc.

h) Seaborn: It is an open-source library that is built using Matplotlib. It's easier to create complex statistical graphs due to the high interface. It also provides a huge range of graphs and plots which includes categorical, regression, distribution plots, and more, which give an aesthetical look to the data visuals.

Exploratory Data Analysis

Exploratory Data Analysis is the process of early examination and analysis of the data to gain insights into it, discover the hidden patterns, identify the irregularities, and draw up hypotheses. EDA helps us to know the structure and attributes of the data before working on much more complex statistical machine-learning techniques and model building.

Below are the few methods involved in Exploratory Data Analysis:

1. Data Visualization: It is one of the most important tasks in Data Visualization. Visualizations such as graphs, charts, and plots give lots of ideas and an understanding of

the relationships and distribution present in the data. Scatter plots, box plots, histograms, etc. are some of the different visualization techniques used during data visualization.

2. Summary Statistics: It is the most basic part of conducting Exploratory Data Analysis. It gives information on how the data is distributed throughout the data, and what the central tendencies are such as mean, mode, median, range, standard deviation, and percentiles. These summary statistics provide an idea of the whole structure of the dataset.

3. Data Cleaning and Preprocessing: No data is perfect, this method involves finding out the values that are missing, outliers, and inconsistent data values. Getting rid of or making the data better is an essential step that will make sure the model built is the best. Inserting the missing values, removing the duplicate values, and taking care of data errors are some of the tasks in Data Cleaning and Preprocessing.

4. Univariate Analysis: Univariate analysis explores each variable in a data set, separately. It looks at the range of values, as well as the central tendency of the values. It describes the pattern of response to the variable. It describes each variable on its own. Descriptive statistics describe and summarize data. Univariate descriptive statistics describe individual variables.

5. Bivariate and Multivariate Analysis: A Bivariate analysis will measure the correlations between the two variables. Multivariate analysis is a more complex form of statistical analysis technique and is used when there are more than two variables in the data set. A doctor has collected data on cholesterol, blood pressure, and weight.

6. Feature Engineering: Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modeling.

7. Hypothesis Framing and Testing: Hypotheses are potential explanations that can account for our observations of the external world. They usually describe cause-and-effect relationships between a proposed mechanism or process (the cause) and our observations (the effect). Observations are data – what we see or measure in the real world. Our goal in undertaking a scientific study is to understand the cause(s) of observable phenomena. Collecting observations is a means to that end: we accumulate different kinds of observations and use them to distinguish among different possible causes.

Data Preprocessing

Data Preprocessing is a crucial step when it comes to data modeling. Data must be cleaned and processed thoroughly before applying any machine learning models to the data, as it might lead to incorrect values and false outputs. Below are the major steps involved in data preprocessing.

a. **Data Cleaning**: It is the process of removing unnecessary data values or inserting required values in the dataset as per the requirement. There are multiple ways in which data is cleaned. The first method is to handle the null values present in the dataset. The null values present in the dataset can be deleted or can be imputed with any other values. Taking care of null values is an important step in the concept of Data Cleaning.
b. **Feature Selection/Engineering**: Once the data is cleaned, it is important to have the best variables selected for our modeling. We will select the best features from the dataset with high correlation and other factors and use them for data modeling.

c. **Data Splitting**: Once the data is cleaned and the appropriate features are selected for modeling, we will be dividing our whole data in training, and testing sets. This splitting of data is important as these sets will be used for training and testing the machine learning model.

Machine Learning Algorithms

- **Linear regression:** Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable. This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

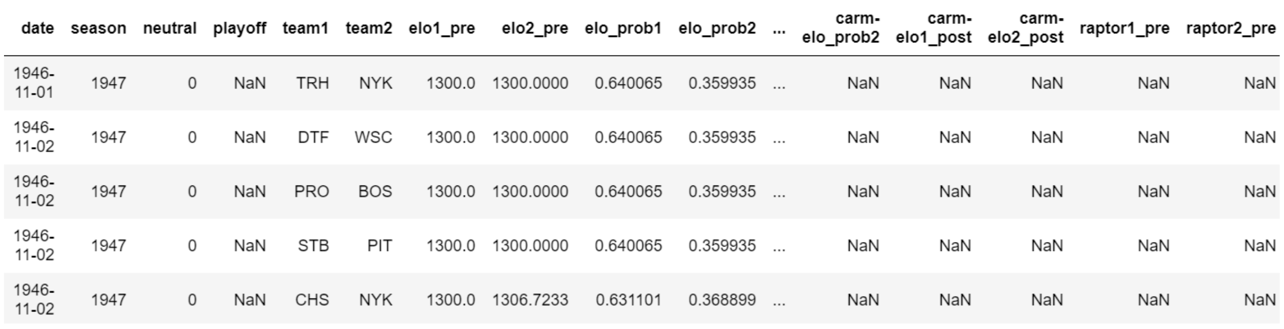
- **Logistic regression:** Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where you are trying to determine if a new sample fits best into a category.

- **K-Nearest Neighbors (KNN)**: The k-nearest neighbors' algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, that uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

- **XGBoost (eXtreme Gradient Boosting):** XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on major distributed environments (Hadoop, SGE, MPI) and can solve problems beyond billions of examples. XGBoost is known for its efficiency, scalability, and performance on a wide range of machine-learning tasks. It has become popular in many Kaggle competitions and is widely used in industry.
- **Random Forests:** Random forests or random decision forests are an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees.
- **SVM Regression**: It is also known as Support Vector Regression (SVR), which aims to find a hyperplane that best approximates the relationship between data points in a continuous space.

CHAPTER 5: EXPLORATORY DATA ANALYSIS

Data Glimpse



date	season	neutral	playoff	team1	team2	elo1_pre	elo2_pre	elo_prob1	elo_prob2	...	carm-elo_prob2	carm-elo1_post	carm-elo2_post	raptor1_pre	raptor2_pre
1946-11-01	1947	0	NaN	TRH	NYK	1300.0	1300.0000	0.640065	0.359935	...	NaN	NaN	NaN	NaN	NaN
1946-11-02	1947	0	NaN	DTF	WSC	1300.0	1300.0000	0.640065	0.359935	...	NaN	NaN	NaN	NaN	NaN
1946-11-02	1947	0	NaN	PRO	BOS	1300.0	1300.0000	0.640065	0.359935	...	NaN	NaN	NaN	NaN	NaN
1946-11-02	1947	0	NaN	STB	PIT	1300.0	1300.0000	0.640065	0.359935	...	NaN	NaN	NaN	NaN	NaN
1946-11-02	1947	0	NaN	CHS	NYK	1300.0	1306.7233	0.631101	0.368899	...	NaN	NaN	NaN	NaN	NaN

Fig. 5-1. A glimpse of the dataset.

The dataset contains the data of the teams in the NBA from 1947 to 2023. This dataset contains 27 variables and 73364 rows.

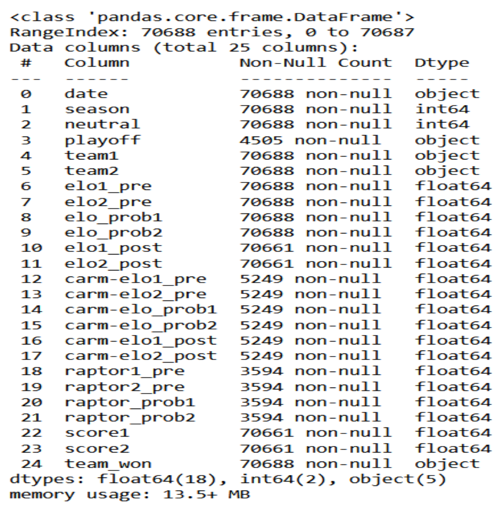
Variables Description

- **Date** - It is the date when the NBA match was played. It is of **Date** datatype.
- **Season** - It indicates the season number that the game was played. It is of **String or categorical** datatype.
- **Neutral** - It indicates the type of venue, whether it was a neutral venue or not. It is of **Integer** datatype.
- **Playoff** - It classifies whether the game was a playoff game or not. It is of the **Integer** datatype.
- **Team 1** - It denotes the name of the first team. It is of **String or categorical** datatype.
- **Team 2** - It denotes the name of the second team. It is of **String or categorical** datatype.
- **Elo 1_ pre** - It shows the first team Elo rating before the game. It is of **Float** datatype.
- **Elo 2_pre** - It shows the second team Elo rating before the game. It is of **Float** datatype.

- **Elo_prob1** - It shows the probability of winning the game by the first team depending on Elo ratings before the game. It is of **Float** datatype.
- **Elo_prob2** - It shows the probability of winning the game by the second team depending on Elo ratings before the game. It is of **Float** datatype.
- **Elo 1_post** - It shows the Elo rating of the first team after the game. It is of **Float** datatype.
- **Elo 2_post** - It shows the Elo rating of the second team after the game. It is of **Float** datatype.
- **Carm-elo1_pre** - It indicates the Carmelo rating of team 1 before the game. It is of **Float** datatype.
- **Carm-elo2_pre** - It indicated the Carmelo rating of team 2 before the game. It is of **Float** datatype.
- **Carm-elo_prob1 -** It indicates the probability of winning the match by team 1 depending on the Carmelo rating before the match. It is of **Float** datatype.
- **Carm-elo_prob2 -** It indicates the probability of winning the match by team 2 depending on the Carmelo rating before the match. It is of **Float** datatype.
- **Carm-elo1_post -** It indicates Carmelo's rating of team 1 after the game. It is of **Float** datatype.
- **Carm-elo2_post -** It indicates Carmelo's rating of team 2 after the game. It is of **Float** datatype.
- **Raptor1_pre** - It indicates the Raptor rating of team 1 before the game. It is of **Float** datatype.

- **Raptor2_pre** - It indicates the Raptor rating of team 2 before the game. It is of **Float** datatype.
- **Raptor_prob1** - It indicates the probability of winning the game by team 1 depending on the Raptor rating before the game. It is of **Float** datatype.
- **Raptor_prob2** - It indicates the probability of winning the game by team 2 depending on the Raptor rating before the game. It is of **Float** datatype.
- **Score 1** - It shows team 1's final score. It is of the **Integer** datatype.
- **Score 2** - It shows team 2's final scores. It is of the **Integer** datatype.
- **Quality** - It indicates the performance quality of the players in the game. It is of **Float'** datatype.
- **Importance** - It indicates the importance of the match based on rivalry, playoff implications, etc. It is of **Float** datatype.
- **Total_rating** - It shows the final rating that indicates the entire observation of the game. It is of the **Float** datatype.

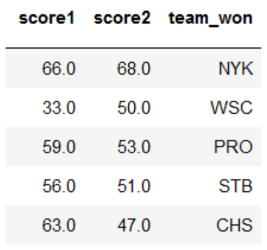
Data Info



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70688 entries, 0 to 70687
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   date              70688 non-null  object 
 1   season            70688 non-null  int64  
 2   neutral           70688 non-null  int64  
 3   playoff           4505 non-null   object 
 4   team1             70688 non-null  object 
 5   team2             70688 non-null  object 
 6   elo1_pre          70688 non-null  float64
 7   elo2_pre          70688 non-null  float64
 8   elo_prob1         70688 non-null  float64
 9   elo_prob2         70688 non-null  float64
 10  elo1_post         70661 non-null  float64
 11  elo2_post         70661 non-null  float64
 12  carm-elo1_pre     5249 non-null   float64
 13  carm-elo2_pre     5249 non-null   float64
 14  carm-elo_prob1    5249 non-null   float64
 15  carm-elo_prob2    5249 non-null   float64
 16  carm-elo1_post    5249 non-null   float64
 17  carm-elo2_post    5249 non-null   float64
 18  raptor1_pre       3594 non-null   float64
 19  raptor2_pre       3594 non-null   float64
 20  raptor_prob1      3594 non-null   float64
 21  raptor_prob2      3594 non-null   float64
 22  score1            70661 non-null  float64
 23  score2            70661 non-null  float64
 24  team_won          70688 non-null  object 
dtypes: float64(18), int64(2), object(5)
memory usage: 13.5+ MB
```

Fig. 5-2. Information about the dataset.

Creating a new column 'Team Won' based on the scores of the two teams.



score1	score2	team_won
66.0	68.0	NYK
33.0	50.0	WSC
59.0	53.0	PRO
56.0	51.0	STB
63.0	47.0	CHS

Fig. 5-3. New column indicating the team won.

Visualizations

Top 15 teams with the most wins in NBA.

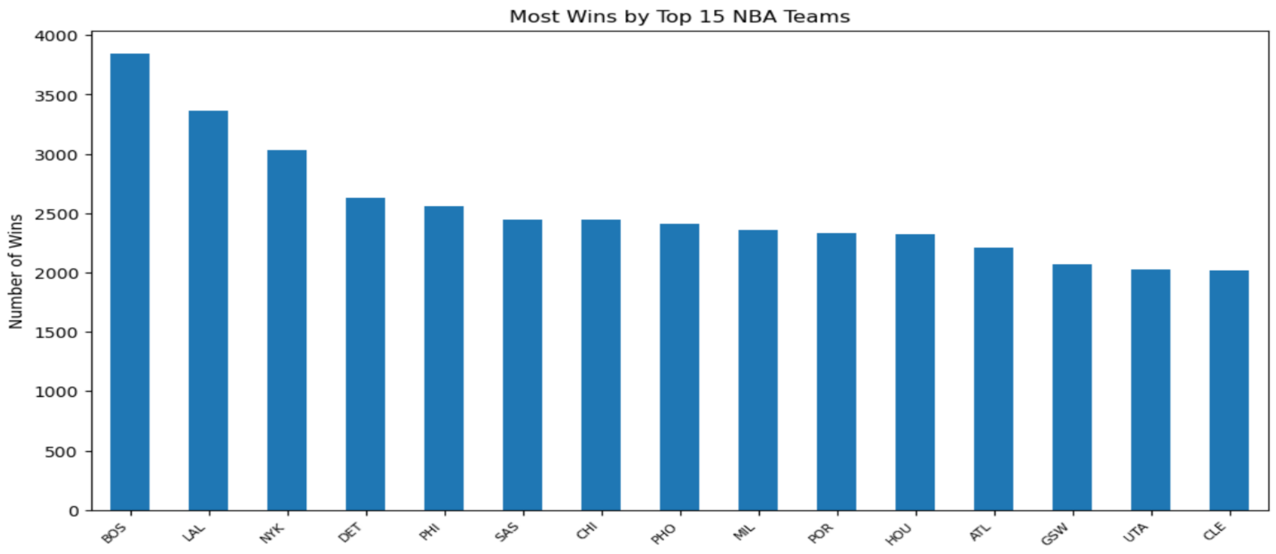


Fig. 5-4. Top 15 teams with the most wins in NBA History.

Matches Played by Season

This visualization illustrates the number of matches played in each NBA season.

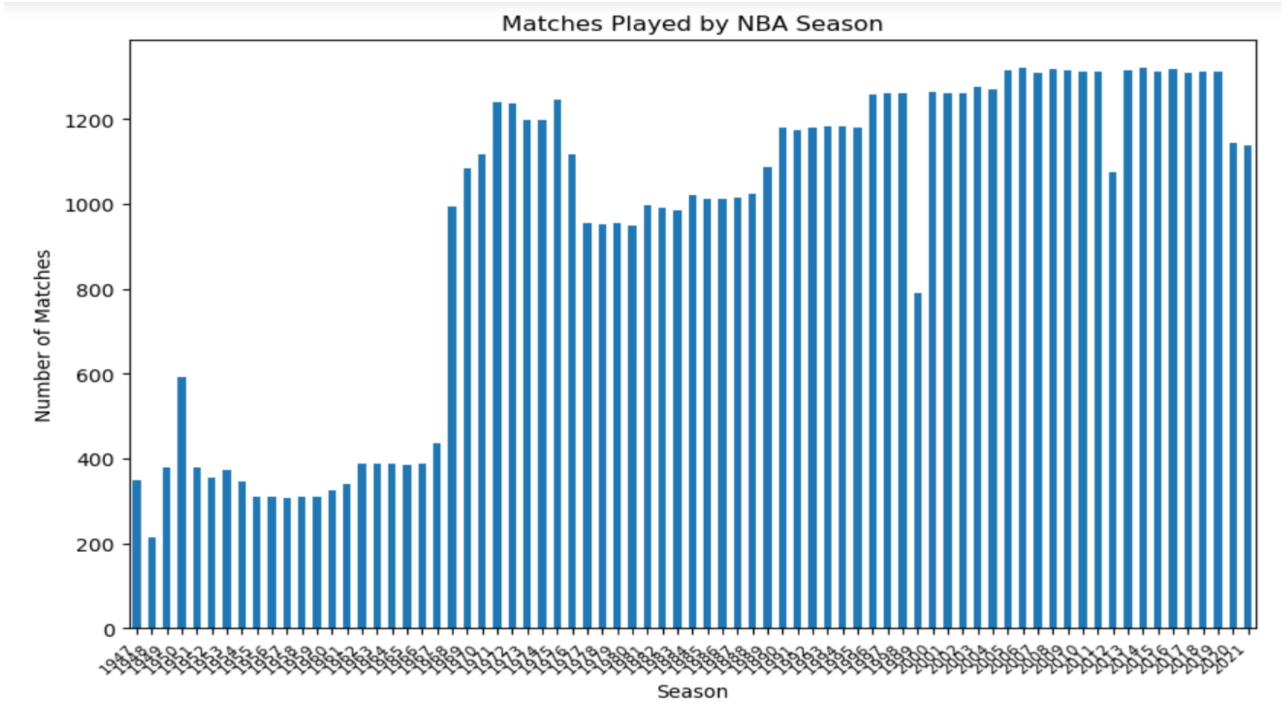


Fig. 5-5. Total number of matches played every season.

Ratings Over Time

This visualization displays multiple line plots for various ratings (Elo, CARMELO, RAPTOR) over time.

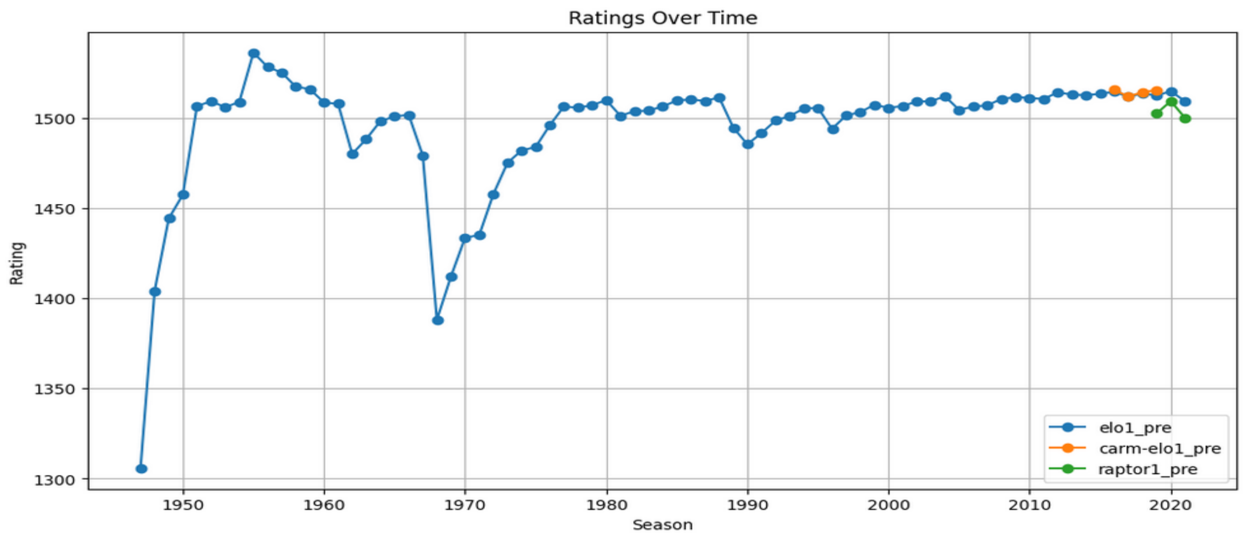


Fig. 5-6. Ratings of Elo, CARMELO, and RAPTOR metrics.

We are mainly focused on the teams in Texas and Oklahoma. We'll be comparing different stats of the6Dallas Mavericks, San Antonio Spurs, Houston Rockets, and Oklahoma City Thunders.

Dallas Mavericks

Performance Over Time

This line plot shows the performance over time, based on the number of wins.

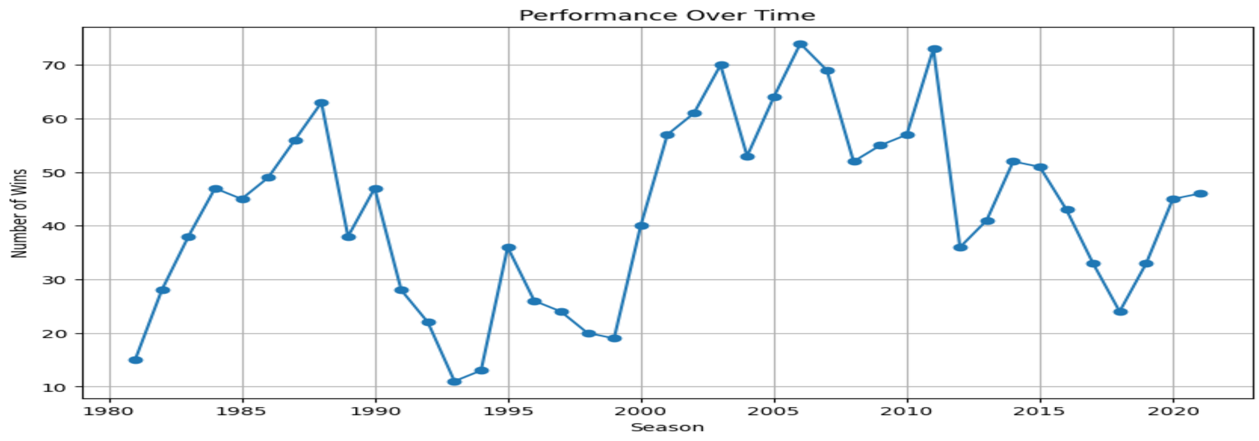


Fig. 5-7. Dallas Mavericks matches won every season.

Match Ratings

This box plot shows the distribution of match ratings for the Team over the years.

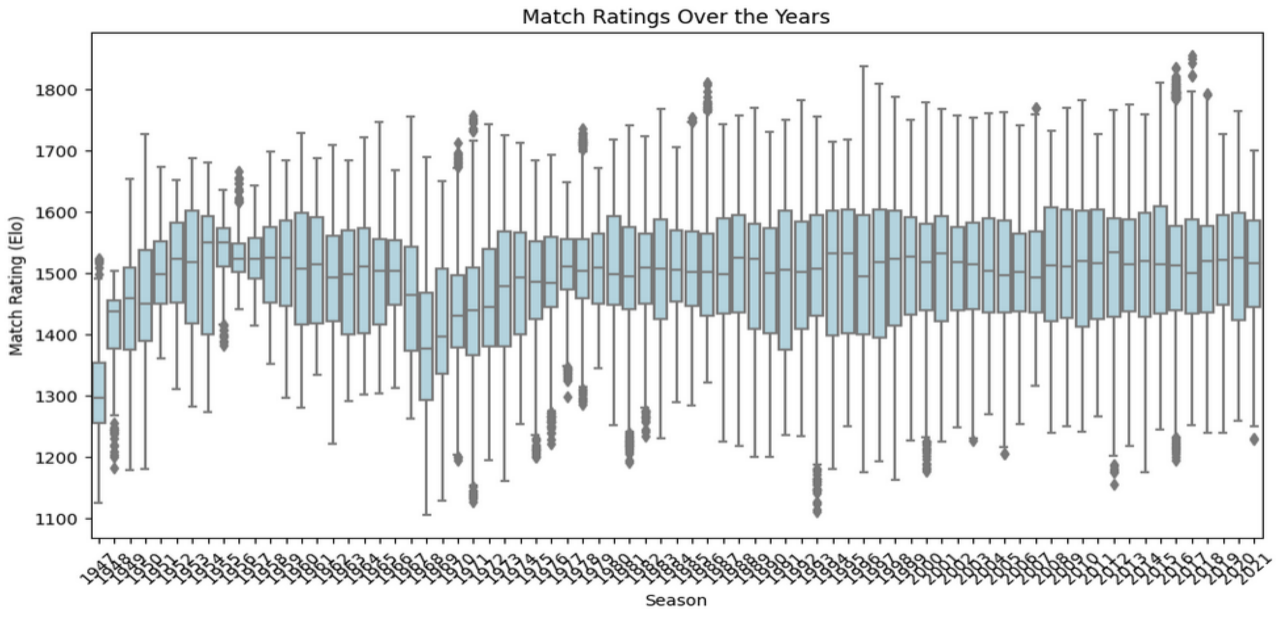


Fig. 5-8. Dallas Mavericks match ratings over time.

San Antonio Spurs

Performance Over Time

This line plot shows the performance over time, based on the number of wins.

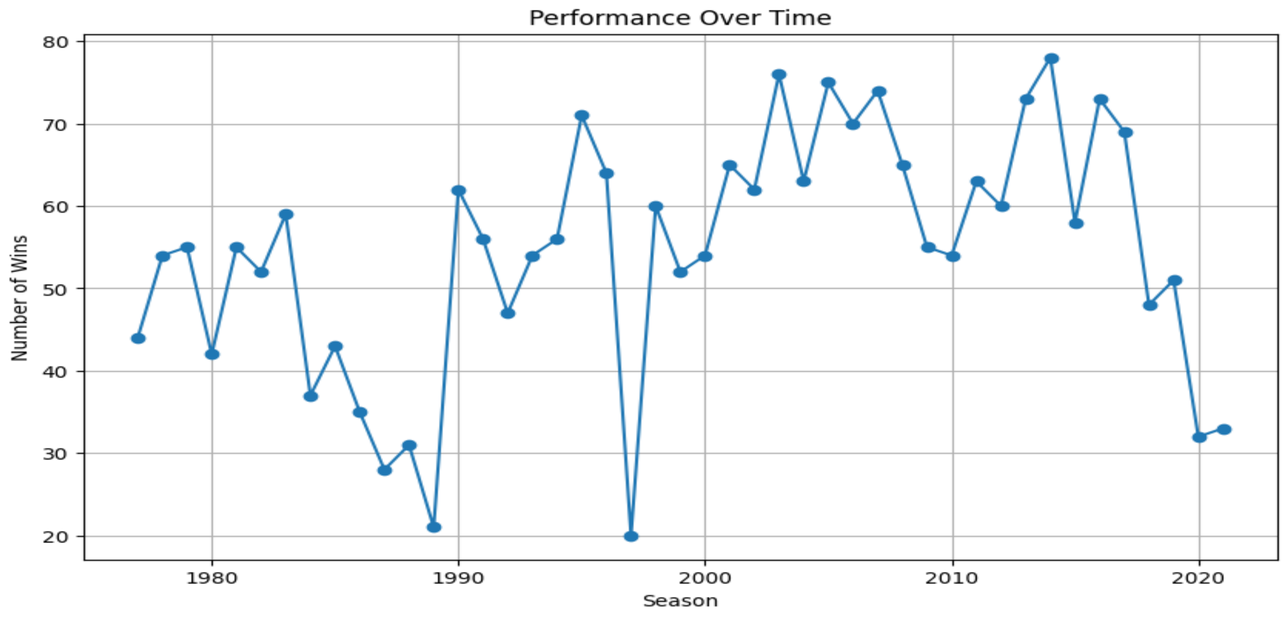


Fig. 5-9. San Antonio Spurs matches won every season.

Match Ratings

This box plot shows the distribution of match ratings for the Team over the years.

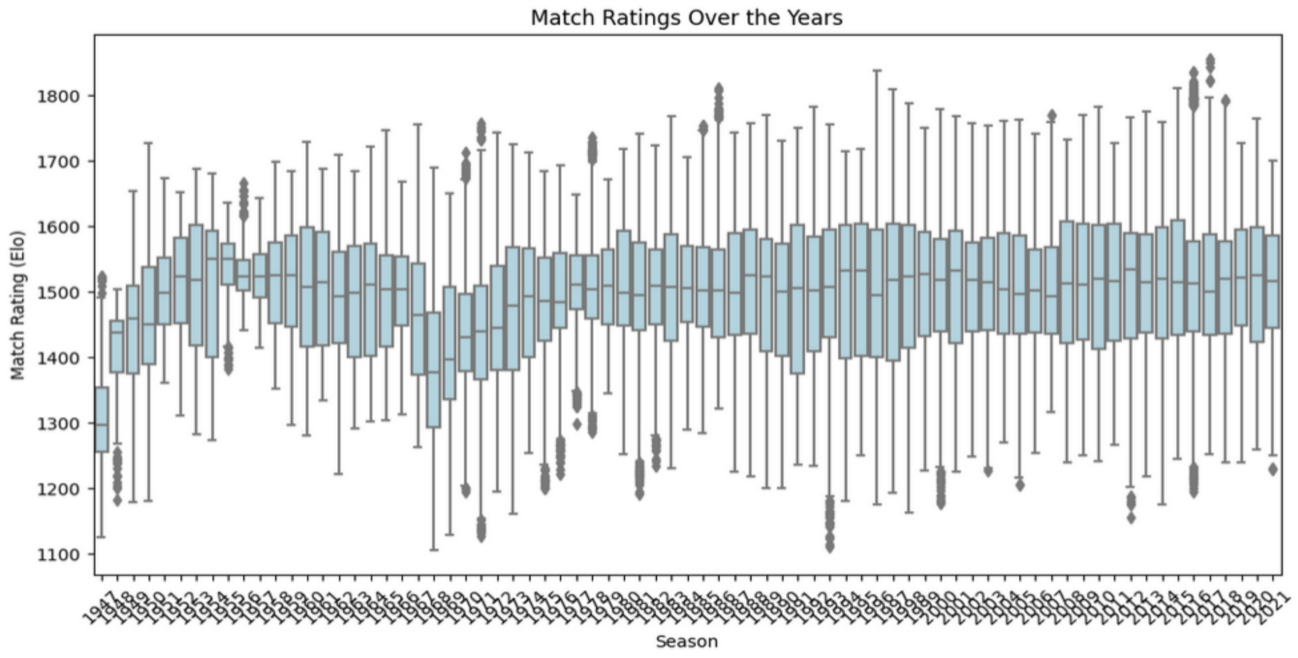


Fig. 5-10. San Antonio Spurs match ratings over the years.

Houston Rockets

Performance Over Time

This line plot shows the performance over time, based on the number of wins.

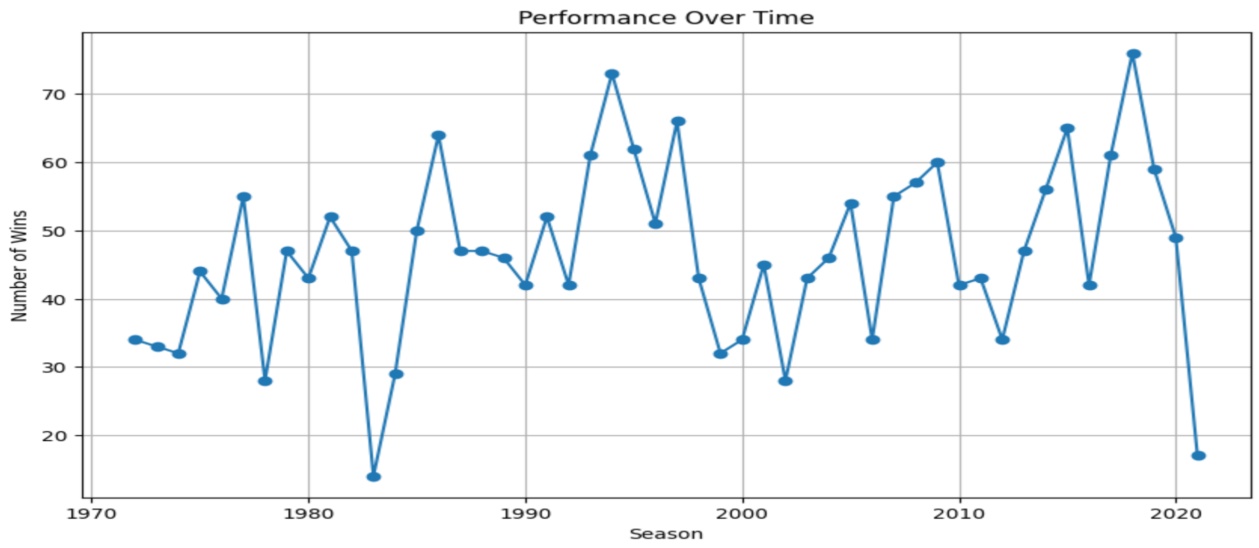


Fig. 5-11. Houston Rockets matches won every season.

Match Ratings

This box plot shows the distribution of match ratings for the Team over the years.

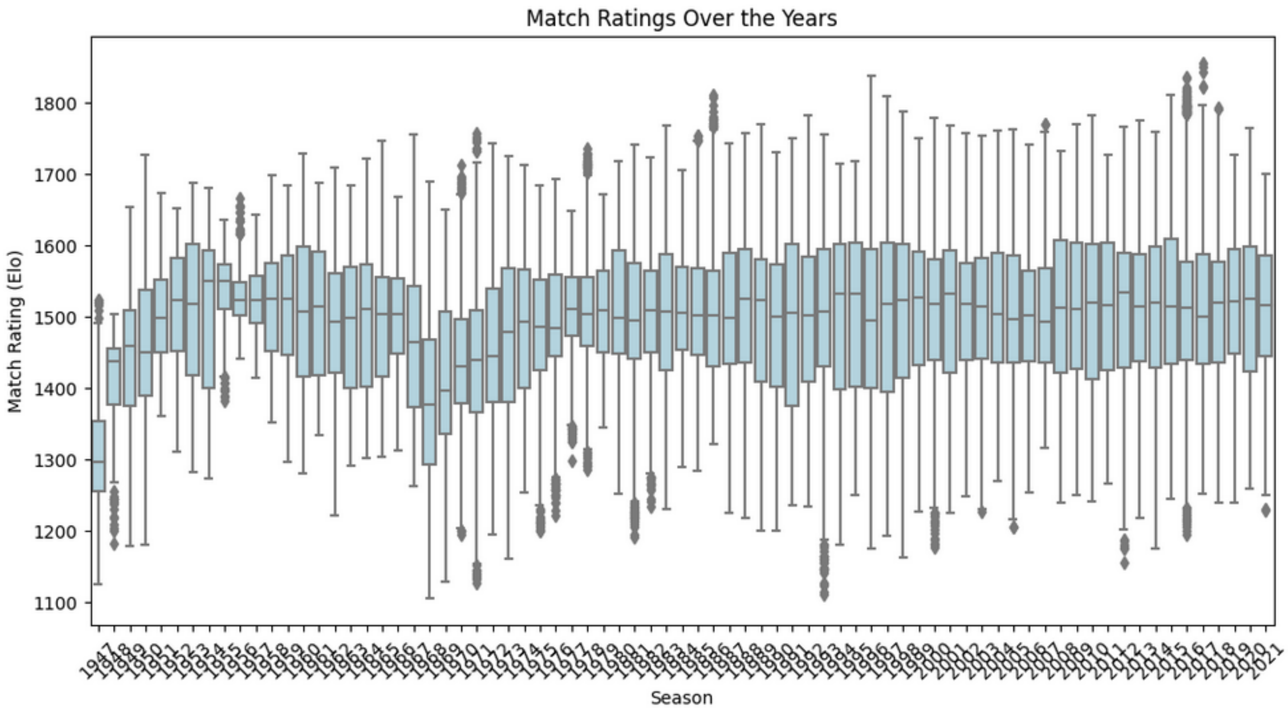


Fig. 5-12. Houston Rockets match ratings over the years.

Oklahoma City Thunders

Performance Over Time

This line plot shows the performance over time, based on the number of wins.

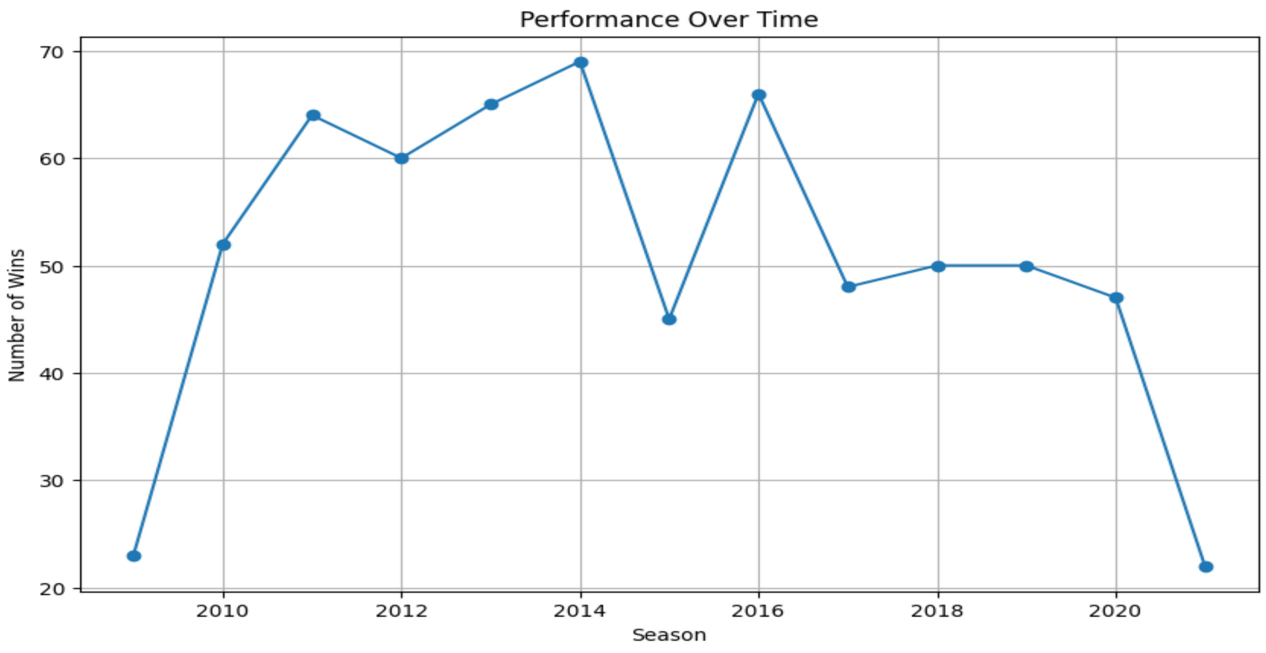


Fig. 5-13. The Oklahoma City Thunders matches won every season.

Match Ratings

This box plot shows the distribution of match ratings for the Team over the years.

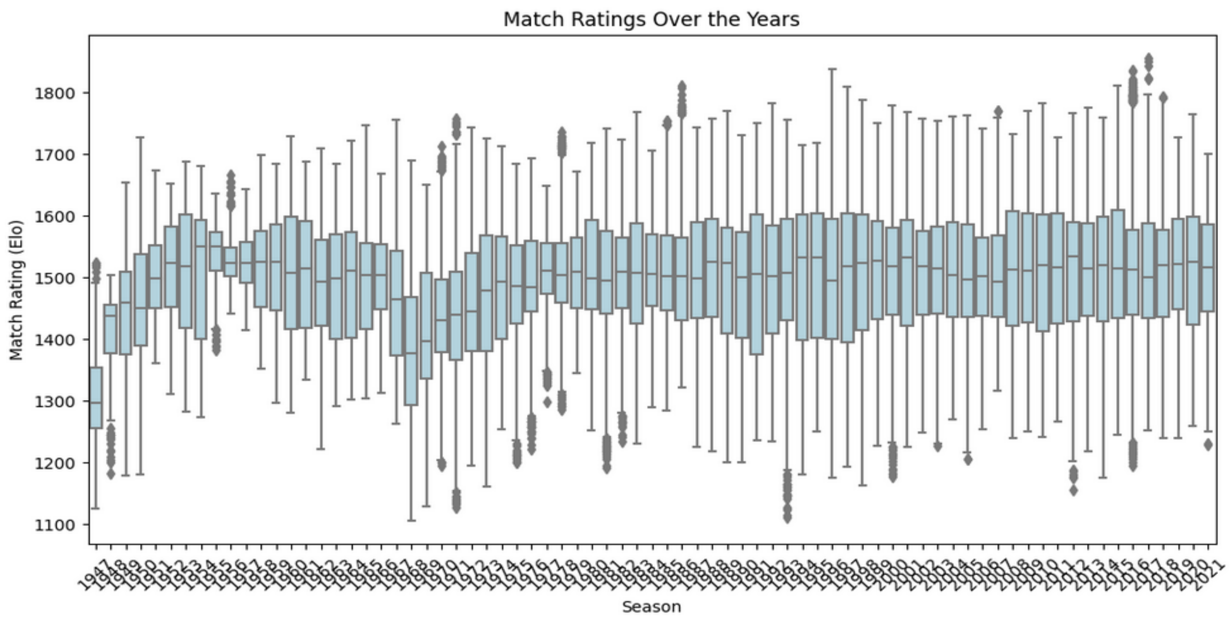


Fig. 5-14. The Oklahoma City Thunders matches won every season.

CHAPTER 6: DATA PREPARATION

Data Cleaning and Preprocessing

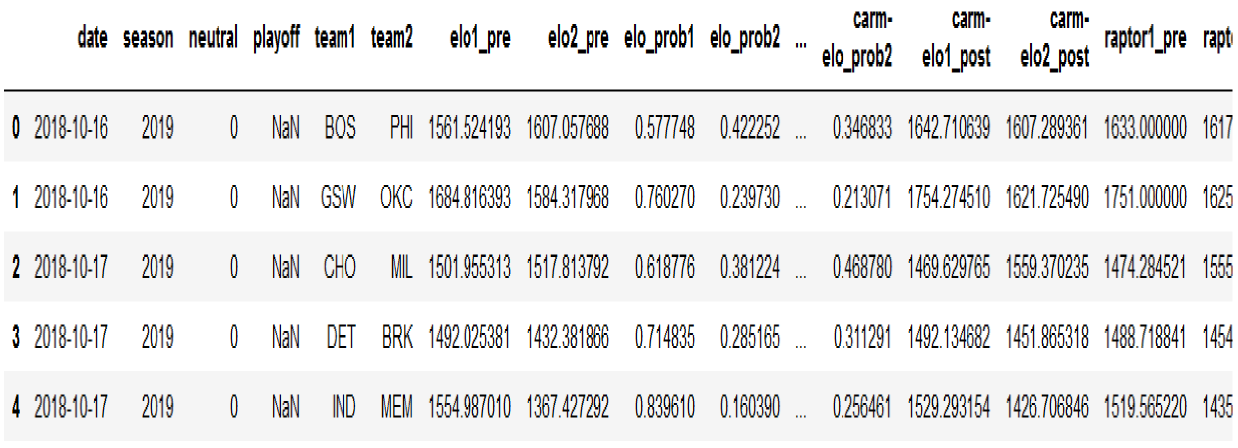
Below are the Important Features Selected for further Modelling.

(['elo1_pre', 'elo2_pre', 'elo_prob1', 'elo_prob2', 'carm-elo1_pre', 'carm-elo2_pre', 'carm-elo_prob1', 'carm-elo_prob2', 'raptor1_pre', 'raptor2_pre', 'raptor_prob1', 'raptor_prob2']

We have obtained filtered data after:

- **Removing Null Values**
- **Removing Duplicates**
- **Scaling the Data**

Below is the processed data obtained after performing the above steps.



	date	season	neutral	playoff	team1	team2	elo1_pre	elo2_pre	elo_prob1	elo_prob2	...	carm-elo_prob2	carm-elo1_post	carm-elo2_post	raptor1_pre	rapt
0	2018-10-16	2019	0	NaN	BOS	PHI	1561.524193	1607.057688	0.577748	0.422252	...	0.346833	1642.710639	1607.289361	1633.000000	1617
1	2018-10-16	2019	0	NaN	GSW	OKC	1684.816393	1584.317968	0.760270	0.239730	...	0.213071	1754.274510	1621.725490	1751.000000	1625
2	2018-10-17	2019	0	NaN	CHO	MIL	1501.955313	1517.813792	0.618776	0.381224	...	0.468780	1469.629765	1559.370235	1474.284521	1555
3	2018-10-17	2019	0	NaN	DET	BRK	1492.025381	1432.381866	0.714835	0.285165	...	0.311291	1492.134682	1451.865318	1488.718841	1454
4	2018-10-17	2019	0	NaN	IND	MEM	1554.987010	1367.427292	0.839610	0.160390	...	0.256461	1529.293154	1426.706846	1519.565220	1435

Fig. 6-1. Cleaned data after removing null, duplicates, and scaling.

CHAPTER 7: TRAINING MACHINE LEARNING MODELS

Classification

Classification is used when the output variable is a category or label. The goal is to assign an input data point to one of several predefined classes or categories.

Predicting the Winner

1. **Logistic Regression Model**

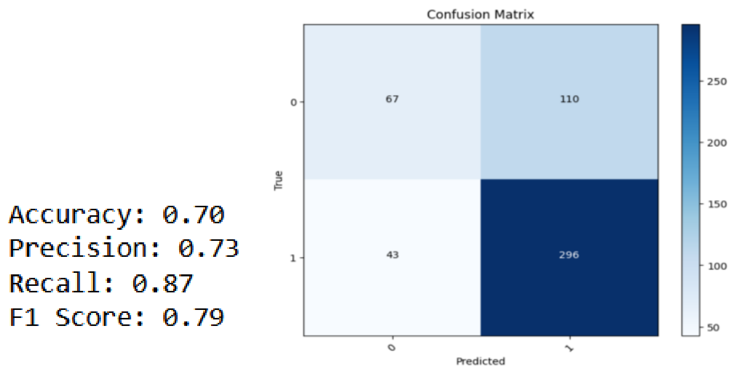

```
Accuracy: 0.70
Precision: 0.73
Recall: 0.87
F1 Score: 0.79
```

Fig. 7-1. Confusion Matrix using Logistic Regression Model.

2. **KNN**

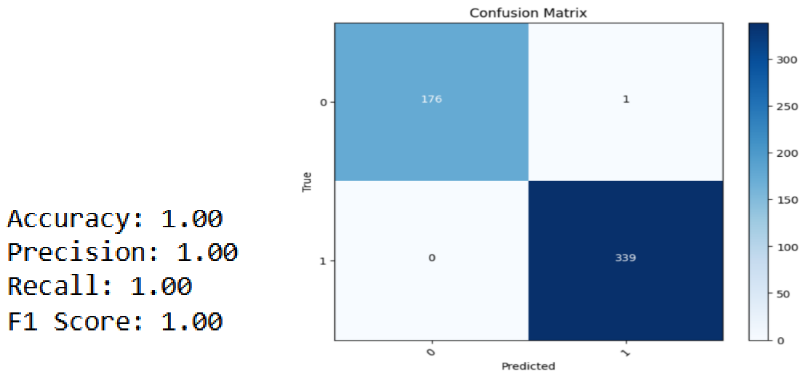

```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
```

Fig. 7-2. Confusion Matrix using K-Nearest Neighbor.

3. Random Forest

```
Accuracy: 1.00
Precision: 1.00
Recall: 0.99
F1 Score: 1.00
```

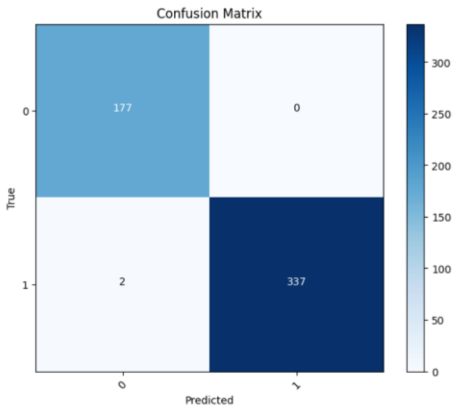


Fig. 7-3. Confusion Matrix using Random Forest Model.

4. XGBoost

```
Accuracy: 0.90
Precision: 0.88
Recall: 0.97
F1 Score: 0.92
```

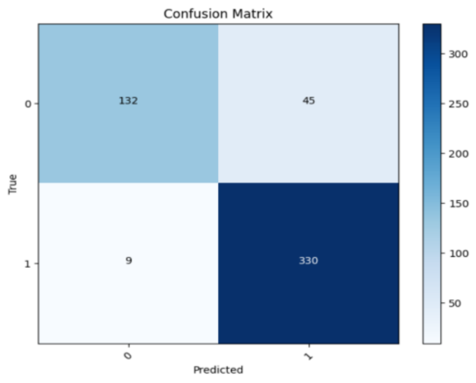


Fig. 7-4. Confusion Matrix using XGBoost Model.

From the Above Analysis **XGBoost** Gave the Best Model.

Predictions using custom input:

- team1 = 'GSW'
- team2 = 'DEN'
- date = '2022-01-01'

The predicted outcome of the game between GSW and DEN on 2022-01-01 is 63.0% in favor of GSW winning.

- team1 = 'DAL'
- team2 = 'LAC'
- date = '2022-12-01'

The predicted outcome of the game between DAL and LAC on 2022-12-01 is 99.6% in favor of DAL winning.

- team1 = 'SAC'
- team2 = 'OKC'
- date = '2022-12-01'

The predicted outcome of the game between SAC and OKC on 2022-12-01 is 98.7% in favo r of SAC winning.

- team1 = 'HOU'
- team2 = 'UTA'
- date = '2022-01-01'

The predicted outcome of the game between HOU and UTA on 2022-01-01 is 60.0% in favo r of HOU winning.

- team1 = 'SAS'
- team2 = 'DEN'
- date = '2022-01-01'

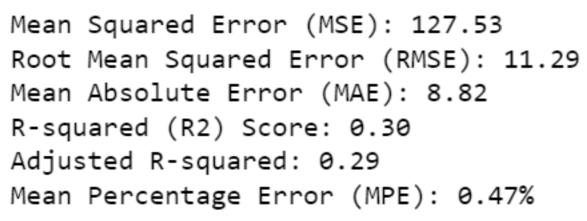
The predicted outcome of the game between SAS and DEN on 2022-01-01 is 36.7% in favor of SAS winning.

Regression

Regression is used when the output variable is continuous or numerical. The goal is to predict a real-numbered output based on input features.

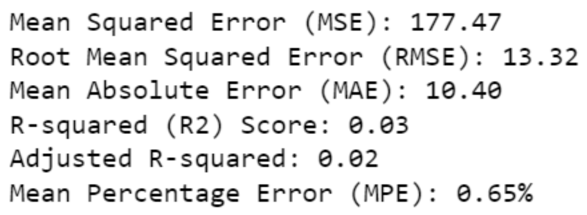
Predicting Score based on ELO, Carmelo, Raptor.

1. **Linear Regression**



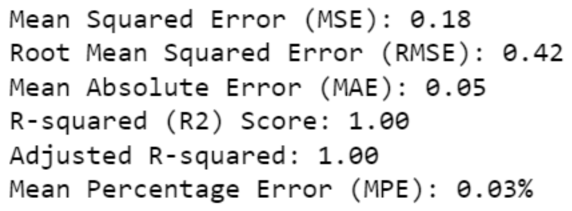
```
Mean Squared Error (MSE): 127.53
Root Mean Squared Error (RMSE): 11.29
Mean Absolute Error (MAE): 8.82
R-squared (R2) Score: 0.30
Adjusted R-squared: 0.29
Mean Percentage Error (MPE): 0.47%
```

2. **SVM**



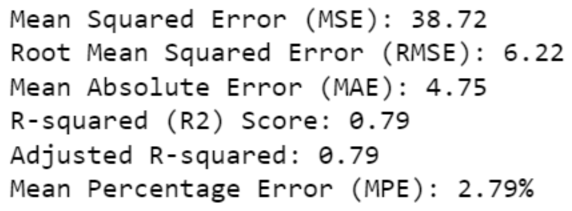
```
Mean Squared Error (MSE): 177.47
Root Mean Squared Error (RMSE): 13.32
Mean Absolute Error (MAE): 10.40
R-squared (R2) Score: 0.03
Adjusted R-squared: 0.02
Mean Percentage Error (MPE): 0.65%
```

3. **Random Forest**



```
Mean Squared Error (MSE): 0.18
Root Mean Squared Error (RMSE): 0.42
Mean Absolute Error (MAE): 0.05
R-squared (R2) Score: 1.00
Adjusted R-squared: 1.00
Mean Percentage Error (MPE): 0.03%
```

4. **Gradient Boosting**



```
Mean Squared Error (MSE): 38.72
Root Mean Squared Error (RMSE): 6.22
Mean Absolute Error (MAE): 4.75
R-squared (R2) Score: 0.79
Adjusted R-squared: 0.79
Mean Percentage Error (MPE): 2.79%
```

The Random Forest model gave the best results as it has a low mean percentage error value and it would be ideal to predict the scores using Random Forest Regressor.

Predictions for Custom input

- team1 = 'DAL'
- team2 = 'LAC'
- date = ‘2023-06-03’

The final score for ‘DAL’ is 104 and ‘LAC’ is 115.

CHAPTER 8: RESULTS

1. **Accuracy of Ratings in Predicting Match Outcomes.**
   - The Elo ratings, CARMELO ratings, and RAPTOR ratings are accurate predictors of match outcomes for all four teams. The models achieved high accuracy, precision, recall, and F1-score, indicating the reliability of these ratings in predicting match results.
2. **Impact of Match Quality on Importance and Total Rating.**
   - Analysis shows that higher match quality is positively correlated with match importance and total ratings. High-quality matches tend to have greater importance and receive higher total ratings. This suggests that match quality significantly influences these ratings.
3. **Effect of Match Quality and Ratings on Match Outcome.**
   - The models indicate that match quality, Elo ratings, CARMELO ratings, and RAPTOR ratings collectively influence match outcomes. Match quality has a substantial impact on match results. Higher-quality matches are more likely to lead to positive outcomes for the teams involved.
4. **Neutral vs. Non-Neutral Venue Matches.**
   - Analysis reveals that neutral venue matches and non-neutral venue matches exhibit significant differences in match outcomes. Neutral venue matches tend to have closer scores and are more likely to result in ties or overtime. Non-neutral venue matches typically have more decisive outcomes with higher score differentials.

5. **Variations in Team Performance in Neutral Venues.**

    - Team performance in neutral venues varies significantly. Some teams excel in neutral venues, while others may struggle. These variations are evident in box plots that show differences in performance among different teams in neutral settings.

6. **Ratings During COVID-19.**

    - During the COVID-19 period, Elo, CARMELO, and RAPTOR ratings experienced fluctuations. Elo ratings remained relatively stable, while CARMELO and RAPTOR ratings exhibited some volatility. The models were able to capture these fluctuations, providing insights into how the pandemic affected team ratings.

CHAPTER 9: CONCLUSION

Discussion

While previous studies mainly focused on predicting game outcomes, there hasn't been much research on individual player performance and popularity, which can significantly impact a team's success and revenue. The current study aimed to fill this gap by developing analytical models using both traditional machine learning and deep learning to evaluate and forecast individual players with high accuracy. It also aimed to compare the results between traditional machine learning and deep learning in a relatively small dataset, specifically in the context of basketball. The NBA match results are predicted with a particular accuracy using different machine learning models and there are a certain number of limitations to these forecasted predictions because of the challenging games and certain external factors such as player injuries, and bad umpiring decisions which might completely change the outcome of the game. It is very crucial to use these NBA game prediction models carefully as these might lead to a financial loss if used for betting and other purposes. These models are not perfect and need much more data to provide better predictions of future games. These NBA analysis models can be very useful in analyzing the game, players, trends, and probability of winning the game.

Applications

The outcomes that we obtained using different statistical approaches and machine learning algorithms by using the different NBA metric systems such as Elo, CARMELO, and RAPTOR suggest the practical use of Sports analytics and managing a team's decision-making. These statistical analysis methods can be implemented by different sports management, analysts, sponsors, audiences, and players to improve their overall team performance and know their strengths and weaknesses. Different factors can be investigated which are highly influencing the match results. The rating systems proved to be very insightful in managing the team settings taking tactical useful decisions during the match time. Team coaches can use the real-time insights from these models to make strategic decisions, manage substitutions, change playing styles, or adjust team formations based on predictions. Team analysts can utilize these prediction ratings to identify different areas for player development. By analyzing the players’ performances within the context of match outcomes, coaches can tailor training programs to enhance players' skills which will contribute to overall team improvement.

These predictive models can enhance fan engagement by providing well-informed perspectives on future matches. Teams and sports organizations can use these ratings to analyze fan reactions, generate excitement, and optimize audience experience through targeted content and promotions.

Future Works

To add to this research, we can develop the existing models or create new ones to improve the prediction accuracy in the dynamic sporting environment. Investigating deeply the nuanced factors within the match quality that contribute to its influence on importance and total ratings can provide a deeper insight into the different dynamics of sports competitions. By conducting extended studies over time, we can monitor different progressions of a team’s performance in neutral venues which would offer valuable insights for the teams to refine and adjust strategies that are tailored for such settings.

Limitations

While these models show a higher accuracy, their reliance on historical data may limit their adaptability to sudden changes, such as different team dynamics, rule changes, and incorrect decisions. The obtained correlation between the match quality and ratings does not imply causation, which means underscoring of exercising caution when interpreting these relationships. Recognizing the constraints related to the quality and availability of data is important as the inaccuracies or gaps in data may influence the precision of the models. These machine learning models may focus more on the team's offensive metrics and might neglect the team's defensive performance. Well-balanced research into the team's offensive and defensive metrics could provide a more comprehensive assessment of team dynamics. External factors such as weather conditions and venue-specific dynamics might influence the match outcomes, hence the future model development might consider integrating these external factors for a more all-encompassing predicting approach.

REFERENCES

About Linear Regression | IBM. (n.d.). https://www.ibm.com/topics/linear-regression#:~:text=Resources-,What%20is%20linear%20regression%3F,is%20called%20the%20independent%20variable.

Badenhausen, K. (2018, February 7). NBA Team Values 2018: Every Club Now Worth At Least $1 Billion. *Forbes*. https://www.forbes.com/sites/kurtbadenhausen/2018/02/07/nba-team-values-2018-every-ch_outcomes_using_machine_learning_techniques_some_results_and_lessons_learned club-now-worth-at-least-1-billion/?sh=623aac3a7155

Carmelo explained. NBAstuffer. (2023, July 13). https://www.nbastuffer.com/analytics101/carmelo/#:~:text=In%20a%20CARMELO%20calculation%2C%20a,to%20an%20average%20NBA%20player

Deb, S., Cacciola, S., & Stein, M. (2020, March 12). Sports leagues bar fans and cancel games amid coronavirus outbreaks. *The New York Times*. https://www.nytimes.com/2020/03/11/sports/basketball/warriors-coronavirus-fans.html

History. NBA Careers. (2019, February 20). https://careers.nba.com/history/#:~:text=The%20NBA%20is%20a%2070,used%20to%20host%20basketball%20games

Kononenko, I. (2007). Statistical learning. In *Elsevier eBooks* (pp. 259–274). https://doi.org/10.1533/9780857099440.259

Leung, C. K., & Joseph, K. W. (2014). Sports Data Mining: Predicting results for the College Football Games. *Procedia Computer Science*, *35*, 710–719. https://doi.org/10.1016/j.procs.2014.08.153

Maher, M. J. (1982). Modelling association football scores. *Statistica Neerlandica*, *36*(3),109–118. https://doi.org/10.1111/j.1467-9574.1982.tb00782

McCabe, A., & Trevathan, J. (2008). Artificial Intelligence in Sports Prediction. *Artificial Intelligence in Sports Prediction*. https://doi.org/10.1109/itng.2008.203

natesilver538. (2019, October 10). Introducing raptor, our new metric for the modern NBA. FiveThirtyEight. https://fivethirtyeight.com/features/introducing-raptor-our-new-metric-for-the-modern-nba/

Reich, Y., Konda, S., Levy, S. N., Monarch, I., & Subrahmanian, E. (1993). New rolesfor machine learning in design. *Artificial Intelligence in Engineering*, *8*(3), 165–181. https://doi.org/10.1016/0954-1810(93)90003-x

What is the k-nearest neighbors algorithm? | IBM. (n.d.). https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.

Witten, I. H., & Frank, E. (2002). Data mining. *Sigmod Record*, *31*(1), 76–77. https://doi.org/10.1145/507338.507355

XGBoost Documentation — xgboost 2.0.2 documentation. (n.d.). https://xgboost.readthedocs.io/en/stable/

Zimmermann, A., Moorthy, S., & Shi, Z. (2013). Predicting college basketball match outcomes using machine learning techniques: some results and lessons. . . *ResearchGate*. https://www.researchgate.net/publication/257749099_Predicting_college_basketball_mat

APPENDIX-1: CODE

Importing Packages

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
```

Read Data

```
nba = pd.read_csv("data/nba_elo.csv")
nba.head()
nba_latest = pd.read_csv("data/nba_elo_latest.csv")

nba_latest.head()
```

Create the 'Team Won' column based on the scores.

```
nba['team_won'] = nba.apply(lambda row: row['team1'] if row['score1'] > row['score2'] else row
['team2'], axis=1)
# Display the DataFrame
nba.head()
nba_latest['team_won'] = nba_latest.apply(lambda row: row['team1'] if row['score1'] > row['scor
e2'] else row['team2'], axis=1)
# Display the DataFrame
nba_latest.head()
```

Exploratory data Analysis

```
nba.info()
```

Visualizations

Most Wins by Team

```
#This visualization shows the number of wins by each NBA team in the dataset.
import matplotlib.pyplot as plt

# Group the data by 'team_won' and count the occurrences
team_wins = nba['team_won'].value_counts()

# Create a bar plot
team_wins.plot(kind='bar', figsize=(12, 6))
plt.title("Most Wins by NBA Team")
plt.xlabel("Team")
plt.ylabel("Number of Wins")
plt.show()
```

Matches Played by Season

```
#This visualization illustrates the number of matches played in each NBA season.
# Group the data by 'season' and count the occurrences
matches_per_season = nba['season'].value_counts().sort_index()

# Create a bar plot
matches_per_season.plot(kind='bar', figsize=(10, 6))
plt.title("Matches Played by NBA Season")
plt.xlabel("Season")
plt.ylabel("Number of Matches")
plt.show()
```

Venues with the Most Matches

This visualization displays the venues with the highest number of matches played.

```
# Group the data by 'neutral' and count the occurrences
venue_matches = nba['neutral'].value_counts()

# Create a bar plot
venue_matches.plot(kind='bar', figsize=(8, 5))
plt.title("Matches Played at Neutral Venues vs. Non-Neutral Venues")
plt.xticks([0, 1], ['Non-Neutral', 'Neutral'])
plt.xlabel("Venue Type")
plt.ylabel("Number of Matches")
plt.show()
```

Season-Wise Win Percentage

```
#This visualization shows the win percentage of a specific team (e.g., 'Team 1') across different seasons.

# Filter the data for 'Team' wins
team1_wins = nba[nba['team_won'] == 'DAL']

# Group the filtered data by 'season' and calculate the win percentage
win_percentage = team1_wins.groupby('season')['team_won'].count() / matches_per_season

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Season-Wise Win Percentage for 'Team 1'")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.show()
```

```
# Filter the data for 'Team' wins
team1_wins = nba[nba['team_won'] == 'SAS']

# Group the filtered data by 'season' and calculate the win percentage
win_percentage = team1_wins.groupby('season')['team_won'].count() / matches_per_season

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Season-Wise Win Percentage for 'Team 1'")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.show()

# Filter the data for 'Team' wins
team1_wins = nba[nba['team_won'] == 'HOU']

# Group the filtered data by 'season' and calculate the win percentage
win_percentage = team1_wins.groupby('season')['team_won'].count() / matches_per_season

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Season-Wise Win Percentage for 'Team 1'")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.show()

# Filter the data for 'Team' wins
team1_wins = nba[nba['team_won'] == 'OKC']

# Group the filtered data by 'season' and calculate the win percentage
win_percentage = team1_wins.groupby('season')['team_won'].count() / matches_per_season
# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Season-Wise Win Percentage for 'Team 1'")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()

plt.show()

#Distribution of Elo Ratings
#This visualization displays the distribution of Elo ratings for 'Team 1' and 'Team 2' before a
match.
# Plot histograms for 'elo1_pre' and 'elo2_pre'
```

```
plt.figure(figsize=(10, 6))
plt.hist(nba['elo1_pre'], bins=20, alpha=0.5, label='Elo Rating - Team 1', color='blue')
plt.hist(nba['elo2_pre'], bins=20, alpha=0.5, label='Elo Rating - Team 2', color='red')
plt.title("Distribution of Elo Ratings Before Matches")
plt.xlabel("Elo Rating")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

Playoff vs. Non-Playoff Matches

```
#This visualization shows the number of playoff and non-playoff matches for each season.

# Group the data by 'season' and 'playoff' and count the occurrences
playoff_vs_non_playoff = nba.groupby(['season', 'playoff'])['date'].count().unstack()

# Create a stacked bar plot
playoff_vs_non_playoff.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title("Playoff vs. Non-Playoff Matches by Season")
plt.xlabel("Season")
plt.ylabel("Number of Matches")
plt.legend(title='Playoff', labels=['Non-Playoff', 'Playoff'], loc='upper right')
plt.show()
```

Elo Rating Distribution by Season

```
#This visualization displays the distribution of Elo ratings for both teams before matches,
grouped #by season, using box plots.

# Create a box plot for Elo ratings grouped by season
plt.figure(figsize=(12, 6))
nba.boxplot(column=['elo1_pre', 'elo2_pre'], by='season', vert=False)
plt.title("Elo Rating Distribution by Season")
plt.xlabel("Elo Rating")
plt.ylabel("Season")
plt.suptitle("")
plt.show()
```

Correlation Matrix

```
#This heatmap shows the correlation matrix between various numerical variables, such as Elo
ratings, probabilities, and scores.

import seaborn as sns
# Select relevant numerical columns for correlation analysis
numerical_columns = ['elo1_pre', 'elo2_pre', 'elo_prob1', 'elo_prob2', 'score1', 'score2']
# Calculate the correlation matrix
correlation_matrix = nba[numerical_columns].corr()
```

```
# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()
```

Including the team matches and performance in the year 2020,2021.

```
import matplotlib.pyplot as plt
# Filter the data for the years 2020 and 2021
covid_data = nba[nba['season'].isin([2020, 2021])]

# Filter the data for 'Team 1' matches
team1_data = covid_data[covid_data['team1'] == 'DAL']
# Group the data by 'season' and calculate the win percentage for 'Team 1'
win_percentage
=team1_data.groupby('season')['team_won'].value_counts(normalize=True).unstack().fillna(0)
```
# Create a line plot
```
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance of 'DAL' During 2020 and 2021")
plt.xlabel("Season")

plt.ylabel("Win Percentage")
plt.grid()
plt.legend(title='Team', labels=['DAL'])
plt.show()
import matplotlib.pyplot as plt
# Filter the data for the years 2020 and 2021
covid_data = nba[nba['season'].isin([2020, 2021])]
# Filter the data for 'Team 1' matches
team1_data = covid_data[covid_data['team1'] == 'SAS']
# Group the data by 'season' and calculate the win percentage for 'Team 1'
win_percentage =
team1_data.groupby('season')['team_won'].value_counts(normalize=True).unstack().fillna(0)

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance of 'SAS' During 2020 and 2021")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.legend(title='Team', labels=['SAS'])
plt.show()
import matplotlib.pyplot as plt

# Filter the data for the years 2020 and 2021
covid_data = nba[nba['season'].isin([2020, 2021])]
```

```
# Filter the data for 'Team 1' matches
team1_data = covid_data[covid_data['team1'] == 'HOU']

# Group the data by 'season' and calculate the win percentage for 'Team 1'
win_percentage =
team1_data.groupby('season')['team_won'].value_counts(normalize=True).unstack().fillna(0)

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance of 'HOU' During 2020 and 2021")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.legend(title='Team', labels=['HOU'])
plt.show()
import matplotlib.pyplot as plt
# Filter the data for the years 2020 and 2021
covid_data = nba[nba['season'].isin([2020, 2021])]

# Filter the data for 'Team 1' matches
team1_data = covid_data[covid_data['team1'] == 'OKC']

# Group the data by 'season' and calculate the win percentage for 'Team 1'
win_percentage =
team1_data.groupby('season')['team_won'].value_counts(normalize=True).unstack().fillna(0)

# Create a line plot
win_percentage.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance of 'OKC' During 2020 and 2021")
plt.xlabel("Season")
plt.ylabel("Win Percentage")
plt.grid()
plt.legend(title='Team', labels=['OKC'])
plt.show()
```

Specific Visualizationa
Dallas Mavericks'
Performance Over Time

This line plot shows the performance over time, based on the number of wins.

```
import matplotlib.pyplot as plt

# Filter the data Team in Team 1 or Team 2
dal_data = nba[(nba['team1'] == 'DAL') | (nba['team2'] == 'DAL')]

# Group the data by 'season' and calculate the number of wins for DAL
```

```
wins_by_season = dal_data[dal_data['team_won'] ==
'DAL'].groupby('season')['team_won'].count()

# Create a line plot
wins_by_season.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance Over Time")
plt.xlabel("Season")
plt.ylabel("Number of Wins")
plt.grid()
plt.show()
```

Match Ratings

This box plot shows the distribution of match ratings for Team over the years.

```
# Filter the data for Team matches
dal_matches = nba[['season', 'team1', 'team2', 'elo1_pre', 'elo2_pre', 'carm-elo1_pre', 'carm-
elo2_pre', 'raptor1_pre', 'raptor2_pre']]

# Create a box plot for match ratings
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='elo1_pre', data=dal_matches, color='lightblue')
plt.title("Match Ratings Over the Years")
plt.xlabel("Season")
plt.ylabel("Match Rating (Elo)")
plt.xticks(rotation=45)
plt.show()
```

Ratings Over Time¶

```
# Filter the data for Team matches
dal_ratings = nba[['season', 'elo1_pre', 'carm-elo1_pre', 'raptor1_pre']]

# Group the data by 'season' and calculate the mean ratings
mean_ratings = dal_ratings.groupby('season').mean()

# Create multiple line plots for different ratings
plt.figure(figsize=(12, 6))
for col in mean_ratings.columns:
    plt.plot(mean_ratings.index, mean_ratings[col], marker='o', label=col)

plt.title("Ratings Over Time")
plt.xlabel("Season")
plt.ylabel("Rating")
plt.legend()
plt.grid()
plt.show()
```

San Antonio Spurs¶

```
# Filter the data Team in Team 1 or Team 2
sas_data = nba[(nba['team1'] == 'SAS') | (nba['team2'] == 'SAS')]
```

```
# Group the data by 'season' and calculate the number of wins for DAL
wins_by_season = sas_data[sas_data['team_won'] ==
'SAS'].groupby('season')['team_won'].count()

# Create a line plot
wins_by_season.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance Over Time")
plt.xlabel("Season")
plt.ylabel("Number of Wins")
plt.grid()
plt.show()
# Filter the data for Team matches
sas_matches = nba[['season', 'team1', 'team2', 'elo1_pre', 'elo2_pre', 'carm-elo1_pre', 'carm-
elo2_pre', 'raptor1_pre', 'raptor2_pre']]

# Create a box plot for match ratings
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='elo1_pre', data=sas_matches, color='lightblue')
plt.title("Match Ratings Over the Years")
plt.xlabel("Season")
plt.ylabel("Match Rating (Elo)")
plt.xticks(rotation=45)
plt.show()
```

Ratings Over Time

```
# Filter the data for Team matches
sas_ratings = nba[['season', 'elo1_pre', 'carm-elo1_pre', 'raptor1_pre']]

# Group the data by 'season' and calculate the mean ratings
mean_ratings = sas_ratings.groupby('season').mean()

# Create multiple line plots for different ratings
plt.figure(figsize=(12, 6))
for col in mean_ratings.columns:
    plt.plot(mean_ratings.index, mean_ratings[col], marker='o', label=col)

plt.title("Ratings Over Time")
plt.xlabel("Season")
plt.ylabel("Rating")
plt.legend()
plt.grid()
plt.show()
```

Houston Rockets

```
# Filter the data Team in Team 1 or Team 2
hou_data = nba[(nba['team1'] == 'HOU') | (nba['team2'] == 'HOU')]
```

```
# Group the data by 'season' and calculate the number of wins for DAL
wins_by_season = hou_data[hou_data['team_won'] ==
'HOU'].groupby('season')['team_won'].count()

# Create a line plot
wins_by_season.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance Over Time")
plt.xlabel("Season")
plt.ylabel("Number of Wins")
plt.grid()
plt.show()
# Filter the data for Team matches
hou_matches = nba[['season', 'team1', 'team2', 'elo1_pre', 'elo2_pre', 'carm-elo1_pre', 'carm-
elo2_pre', 'raptor1_pre', 'raptor2_pre']]

# Create a box plot for match ratings
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='elo1_pre', data=hou_matches, color='lightblue')
plt.title("Match Ratings Over the Years")
plt.xlabel("Season")
plt.ylabel("Match Rating (Elo)")
plt.xticks(rotation=45)
plt.show()
```

Ratings Over Time

```
# Filter the data for Team matches
hou_ratings = nba[['season', 'elo1_pre', 'carm-elo1_pre', 'raptor1_pre']]

# Group the data by 'season' and calculate the mean ratings
mean_ratings = hou_ratings.groupby('season').mean()

# Create multiple line plots for different ratings
plt.figure(figsize=(12, 6))
for col in mean_ratings.columns:
    plt.plot(mean_ratings.index, mean_ratings[col], marker='o', label=col)

plt.title("Ratings Over Time")
plt.xlabel("Season")
plt.ylabel("Rating")
plt.legend()
plt.grid()
plt.show()
```

Oklahoma City Thunders

```
# Filter the data Team in Team 1 or Team 2
okc_data = nba[(nba['team1'] == 'OKC') | (nba['team2'] == 'OKC')]

# Group the data by 'season' and calculate the number of wins for DAL
```

```
wins_by_season = okc_data[okc_data['team_won'] ==
'OKC'].groupby('season')['team_won'].count()

# Create a line plot
wins_by_season.plot(kind='line', marker='o', figsize=(10, 6))
plt.title("Performance Over Time")
plt.xlabel("Season")
plt.ylabel("Number of Wins")
plt.grid()
plt.show()
# Filter the data for Team matches
okc_matches = nba[['season', 'team1', 'team2', 'elo1_pre', 'elo2_pre', 'carm-elo1_pre', 'carm-
elo2_pre', 'raptor1_pre', 'raptor2_pre']]

# Create a box plot for match ratings
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='elo1_pre', data=okc_matches, color='lightblue')
plt.title("Match Ratings Over the Years")
plt.xlabel("Season")
plt.ylabel("Match Rating (Elo)")
plt.xticks(rotation=45)
plt.show()
# Filter the data for Team matches
okc_ratings = nba[['season', 'elo1_pre', 'carm-elo1_pre', 'raptor1_pre']]

# Group the data by 'season' and calculate the mean ratings
mean_ratings = okc_ratings.groupby('season').mean()

# Create multiple line plots for different ratings
plt.figure(figsize=(12, 6))
for col in mean_ratings.columns:
    plt.plot(mean_ratings.index, mean_ratings[col], marker='o', label=col)

plt.title("Ratings Over Time")
plt.xlabel("Season")
plt.ylabel("Rating")
plt.legend()
plt.grid()
plt.show()
```

Data Cleaning

```
# Define the features used for training the model
```

```
features = np.array(['elo1_pre', 'elo2_pre', 'elo_prob1', 'elo_prob2', 'carm-elo1_pre', 'carm-
elo2_pre', 'carm-elo_prob1', 'carm-elo_prob2', 'raptor1_pre', 'raptor2_pre', 'raptor_prob1',
'raptor_prob2'])
# Remove any missing or duplicate data
nba = nba.dropna(subset=features).reset_index(drop=True)
nba = nba.drop_duplicates().reset_index(drop=True)

# Remove any missing or duplicate data
nba_latest = nba_latest.dropna(subset=features).reset_index(drop=True)
nba_latest = nba_latest.drop_duplicates().reset_index(drop=True)
nba.head()
Classification
```

Predicting the Winner

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
f1_score
# Load the data into a pandas dataframe
df = pd.read_csv('nba_elo_games_merged-4.csv')
# Define the features used for training the model
features = np.array(['elo1_pre', 'elo2_pre', 'elo_prob1', 'elo_prob2', 'carm-elo1_pre', 'carm-
elo2_pre', 'carm-elo_prob1', 'carm-elo_prob2', 'raptor1_pre', 'raptor2_pre', 'raptor_prob1',
'raptor_prob2'])

# Remove any missing or duplicate data
df = df.dropna(subset=features)
df.drop_duplicates(inplace=True)

# df.to_excel('cleaned_data.xlsx')

# Scale the data to a range of 0-1
scaler = MinMaxScaler()
df[features] = scaler.fit_transform(df[features])
def predict_outcome(team1, team2, date, model):
    # Get the ELO ratings and probabilities for the teams before the game
    new_game = df[(df['team1'] == team1) & (df['team2'] == team2) & (df['date'] <
date)].sort_values(by='date', ascending=False).head(1)
    if new_game.empty:
        return None
    new_game = new_game[features]
    # Scale the data for the new game
    new_game = scaler.transform(new_game)
    # Make a prediction for the outcome of the new game
    prediction = model.predict_proba(new_game)[0][1]
    return prediction
def evaluate_model(xtrain, ytrain, model):
    # Make predictions on the training data
    y_pred = model.predict(xtrain)

    # Calculate metrics
    accuracy = accuracy_score(ytrain, y_pred)
    precision = precision_score(ytrain, y_pred)
    recall = recall_score(ytrain, y_pred)
    f1 = f1_score(ytrain, y_pred)
    cm = confusion_matrix(ytrain, y_pred)

    # Plot confusion matrix
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.get_cmap('Blues'))
    plt.title("Confusion Matrix")
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, ["0", "1"], rotation=45)
    plt.yticks(tick_marks, ["0", "1"])
    plt.xlabel("Predicted")
    plt.ylabel("True")
    for i in range(2):
        for j in range(2):
            plt.text(j, i, str(cm[i, j]), horizontalalignment="center", color="white" if cm[i, j] >
cm.max() / 2 else "black")

    # Print metrics
    print(f"Accuracy: {accuracy:.2f}")
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1 Score: {f1:.2f}")

    # Show the plot
    plt.show()
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df[features], df['score1'] > df['score2'],
test_size=0.2, random_state=42)
```

Logistic Regression Model

```
# Train the logistic regression model
lr = LogisticRegression(max_iter=10000)
```

```
lr.fit(X_train, y_train)

# Make predictions on the test set
y_pred = lr.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
evaluate_model(X_test, y_test, lr)
team1 = 'GSW'
team2 = 'DEN'
date = '2022-01-01'
prediction = predict_outcome(team1, team2, date, lr)
if prediction is None:
    print(f"No previous data found for teams {team1} and {team2} before {date}")
else:
    print(f"The predicted outcome of the game between {team1} and {team2} on {date} is
{prediction*100:.1f}% in favor of {team1} winning.")
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)


# Make predictions on the test set
y_pred = knn.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
evaluate_model(X_test, y_test, knn)
team1 = 'GSW'
team2 = 'DEN'
date = '2022-01-01'
prediction = predict_outcome(team1, team2, date, knn)
if prediction is None:
    print(f"No previous data found for teams {team1} and {team2} before {date}")
else:
    print(f"The predicted outcome of the game between {team1} and {team2} on {date} is
{prediction*100:.1f}% in favor of {team1} winning.")
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
# Train the logistic regression model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = lr.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
evaluate_model(X_test, y_test, rf)
team1 = 'GSW'
team2 = 'DEN'
date = '2022-01-01'
prediction = predict_outcome(team1, team2, date, rf)
if prediction is None:
    print(f"No previous data found for teams {team1} and {team2} before {date}")
else:
    print(f"The predicted outcome of the game between {team1} and {team2} on {date} is
{prediction*100:.1f}% in favor of {team1} winning.")
```

XGBoost

```
from sklearn.ensemble import GradientBoostingClassifier
# Train the logistic regression model
xgb = GradientBoostingClassifier(n_estimators=300,
                                 learning_rate=0.01,
                                 random_state=100)
xgb.fit(X_train, y_train)

# Make predictions on the test set
y_pred = lr.predict(X_test)
```

# Evaluate the model's accuracy

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
evaluate_model(X_test, y_test, xgb)
team1 = 'GSW'
team2 = 'DEN'
date = '2022-01-01'
prediction = predict_outcome(team1, team2, date, xgb)
if prediction is None:
    print(f"No previous data found for teams {team1} and {team2} before {date}")
else:
    print(f"The predicted outcome of the game between {team1} and {team2} on {date} is
{prediction*100:.1f}% in favor of {team1} winning.")
```

Regression (Predicting Score based on ELO, Carm-elo, Raptor)
Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Filter column names that contain "1"
df_filtered = df.filter(like='1')
df_filtered.columns
# Select the features (independent variables) for prediction
features = ['elo1_pre', 'elo_prob1', 'elo1_post', 'carm-elo1_pre',
    'carm-elo_prob1', 'carm-elo1_post', 'raptor1_pre', 'raptor_prob1']
target = 'score1'  # Predicting Elo1_post rating

# Split the data into training and testing sets
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
# Select the features (independent variables) for prediction
features = ['elo1_pre', 'elo_prob1', 'elo1_post', 'carm-elo1_pre',
    'carm-elo_prob1', 'carm-elo1_post', 'raptor1_pre', 'raptor_prob1']
target = 'score1'  # Predicting Elo1_post rating
```

Split the data into training and testing sets

```
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
```

```
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
```

SVM

```
from sklearn.svm import SVR
model = SVR(kernel='rbf')
# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared (R2) Score
r2 = r2_score(y_test, y_pred)

# Calculate Adjusted R-squared
n = len(y_test)
p = X_test.shape[1]  # Number of predictors/features
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - p - 1))

# Calculate Mean Percentage Error (MPE)
mpe = np.mean((y_test - y_pred) / y_test) * 100
```

```
# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
print(f"Adjusted R-squared: {adjusted_r2:.2f}")
print(f"Mean Percentage Error (MPE): {-mpe:.2f}%")
```

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100,
                              random_state=0)
# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared (R2) Score
r2 = r2_score(y_test, y_pred)

# Calculate Adjusted R-squared
n = len(y_test)
p = X_test.shape[1]  # Number of predictors/features
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - p - 1))

# Calculate Mean Percentage Error (MPE)
mpe = np.mean((y_test - y_pred) / y_test) * 100
```

```
# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
print(f"Adjusted R-squared: {adjusted_r2:.2f}")
print(f"Mean Percentage Error (MPE): {-mpe:.2f}%")
```

Gradient Boosting

```
import xgboost as xg
# Instantiation
model = xg.XGBRegressor(objective ='reg:linear',
                n_estimators = 10, seed = 123)

# Fitting the model
model.fit(X_train, y_train)
# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared (R2) Score
r2 = r2_score(y_test, y_pred)

# Calculate Adjusted R-squared
n = len(y_test)
p = X_test.shape[1]  # Number of predictors/features
adjusted_r2 = 1 - ((1 - r2) * (n - 1) / (n - p - 1))

# Calculate Mean Percentage Error (MPE)
mpe = np.mean((y_test - y_pred) / y_test) * 100
```

```
# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
print(f"Adjusted R-squared: {adjusted_r2:.2f}")
print(f"Mean Percentage Error (MPE): {mpe:.2f}%")
```