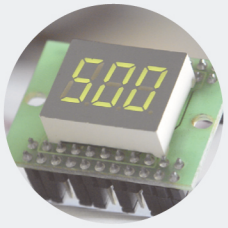
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Modelo de Aprendizagem Automática para Prever Jogos de Basquetebol

JOÃO MIGUEL LIMA ALVES
julho de 2024

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Machine Learning Model for Predicting Basketball Games

João Miguel Lima Alves

Master in Electrical and Computer Engineering
Specialization Area of Automation and Systems

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

July, 2024

This dissertation partially satisfies the requirements of the Thesis/Dissertation course of the program Master in Electrical and Computer Engineering, Specialization Area of Automation and Systems.

Candidate: João Miguel Lima Alves, No. 1171575, 1171575@isep.ipp.pt

Scientific Guidance: Ramiro S. Barbosa, rsb@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

July, 2024

Acknowledgements

I want to thank Dr. Ramiro Barbosa, for the guidance he provided during this dissertation period and for helping me to develop the scientific skills to carry it out.

I would also like to thank Instituto Superior de Engenharia do Porto (ISEP) and all the professors who were part of this journey and helped me acquire the essential tools that I use daily.

Special thanks also to my parents, who have always supported me on this academic journey, giving me all the help I needed to get to where I am today.

To my colleagues who have been a big part of my academic journey. Stephane Monteiro, Bruno Cunha, Filipe Gomes, Francisca Almeida, Tomás Sousa, Bruno Mendes, Carlos Gonçalves, Francisca Almeida Diogo Lopes, Hugo Silva, Diogo Rebelo, Tiago Sousa, Pedro Oliveira and Filipe Campos.

All these people were essential to my academic career and without them, it would have been unthinkable to get to where I am today.

Abstract

Artificial intelligence has emerged as an essential tool in many areas. Sport, due to its widespread appeal, has also embraced this trend and recognized its significance in driving its development. The processing and analytical capacity developed in recent years in artificial intelligence offers a new dimension for understanding and predicting complex events. Predictive analysis, driven by machine learning algorithms, makes it possible to identify patterns and trends that would not be evident using traditional methods.

In the field of sports, machine learning is significantly altering the analysis of athletes' and teams' performance. In the realm of basketball, these techniques afford comprehensive insights into both individual and collective development, facilitating a more precise evaluation of the strategies and tactics employed. The meticulous analysis of every facet of a team's daily routine has the potential to elevate the sport to a heightened level of competition.

This thesis investigates the use of machine learning to predict the outcome of National Basketball Association and Women's National Basketball Association games, using historical data and collective performance metrics. Through the utilization of advanced algorithms, this application seeks to analyze patterns that are crucial for detecting future results. The objective is to demonstrate the capability of these technologies to predict outcomes that may not be attainable solely through human analysis.

The research findings underscore the potential of machine learning to surpass traditional statistical methods in predicting sports outcomes. hrough the integration of comprehensive data and advanced modeling techniques, it is possible to demonstrate the capacity to generate more accurate and pertinent predictions. This approach not only enriches sports analysis but also holds considerable practical value, supporting strategic decision-making in the realm of basketball.

Keywords: Artificial Intelligence, Machine Learning, Basquetball, Sports.

Resumo

A inteligência artificial surge como uma ferramenta essencial em muitos domínios. O desporto, devido ao seu apelo generalizado, também aderiu a esta tendência e reconheceu a sua importância para impulsionar o seu desenvolvimento. A capacidade de processamento e análise desenvolvida nos últimos anos na inteligência artificial oferece uma nova dimensão para a compreensão e previsão de eventos complexos. A análise preditiva, impulsionada por algoritmos de *machine learning*, permite identificar padrões e tendências que não seriam evidentes utilizando métodos tradicionais.

No domínio do desporto, *machine learning* está a alterar significativamente a análise do desempenho dos atletas e das equipas. No domínio do basquetebol, estas técnicas permitem uma visão abrangente do desenvolvimento individual e coletivo, facilitando uma avaliação mais precisa das estratégias e tácticas utilizadas. A análise meticulosa de todas as facetas da rotina diária de uma equipa tem o potencial de elevar o desporto a um nível de competição mais elevado.

Esta tese investiga a utilização de *machine learning* para prever o resultado dos jogos da Associação Nacional de Basquetebol e da Associação Nacional de Basquetebol Feminino, utilizando dados históricos e métricas de desempenho coletivo. Através da utilização de algoritmos avançados, esta investigação procura analisar padrões que são cruciais para detetar resultados futuros. O objetivo é demonstrar a capacidade destas tecnologias para prever resultados que podem não ser alcançados apenas através da análise humana.

Os resultados da dissertação sublinham o potencial de *machine learning* para ultrapassar os métodos estatísticos tradicionais na previsão de resultados desportivos. Através da integração de dados abrangentes e de técnicas de modelação avançadas, pode-se demonstrar a capacidade de gerar previsões mais exactas e pertinentes. Esta abordagem não só enriquece a análise desportiva, como também tem um valor prático considerável, apoiando a tomada de decisões estratégicas no domínio do basquetebol.

Palavras-Chave: Inteligência Artificial, *Machine Learning*, Basquetebol, Desporto.

Contents

List of Figures

List of Tables

List of Acronyms

3PAr	*Three Point Attempt Rate*
AB	*AdaBoost Classifier*
AI	*Artificial Intelligence*
ANN	*Artificial Neural Network*
AST%	*Assist Percentage*
BAG	*Bagging Classifier*
BLK%	*Block Percentage*
CNN	*Convolutional Neural Network*
CSV	*Comma-Separated Values File*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DRB%	*Defensive Rebound Percentage*
DRtg	*Defensive Rating*
eFG%	*Effective Field Goal Percentage*
FTr	*Free Throw Attempt Rate*
HTML	*HyperText Markup Language*
JSON	*JavaScript Object Notation*
KNN	*K Nearest Neighbors*
LR	*Logistic Regression*
LSTM	*Long Short Term Memory*
ML	*Machine Learning*
MLP	*Multi-Layer Perceptron*

MVP	*Most Valuable Player*
NB	*Naives Bayes*
NBA	*National Basketball Association*
NLP	*Natural Language Programming*
ORB%	*Offensive Rebound Percentage*
ORtg	*Offensive Rating*
RF	*Random Forest*
RL	*Reinforcement Learning*
RNN	*Recurrent Neural Network*
RR	*Ridge Regression Classifier*
SARSA	*State-Action-Reward-State-Action*
SSL	*Self-Supervised Learning*
STACK	*Stacking Classifier*
STL%	*Steal Percentage*
SVM	*Support Vector Machines*
TOV%	*Turnover Percentage*
TRB%	*Total Rebound Percentage*
TS%	*True Shooting Percentage*
USG%	*Usage Rating*
WNBA	*Women National Basketball Association*
XGB	*XGBoost Classifier*
XML	*Extensible Markup Language*

Chapter 1

Introduction

Basketball is one of the most famous sports in the world, especially in the United States of America and Canada. Leagues such as the *National Basketball Association* (NBA) act almost like the Champions League model in soccer terms, they are extremely competitive, and they bring together the best teams in the world as well as the best players.

Its popularity in America and Europe makes it one of the main focuses for obtaining data on sports. The information on this sport is very varied, with a wealth of statistics on which to focus to identify patterns. Compared to sports such as soccer, basketball ends up being a much more mathematical sport when it comes to numbers.

Basketball is well-suited for statistical analysis due to its emphasis on individual and collective data, which makes it an attractive option for developing *Machine Learning* (ML) algorithms to identify patterns.

1.1 Context

Basketball is a highly popular sport all over the world, which draws a significant amount of attention from diverse audiences. As depicted in Figure 1.1, basketball was the second most-watched sport globally in 2023, right after soccer. There are several fiercely competitive basketball leagues, but the NBA stands out from the rest due to its immense popularity and the consistent entertainment it provides year after year.

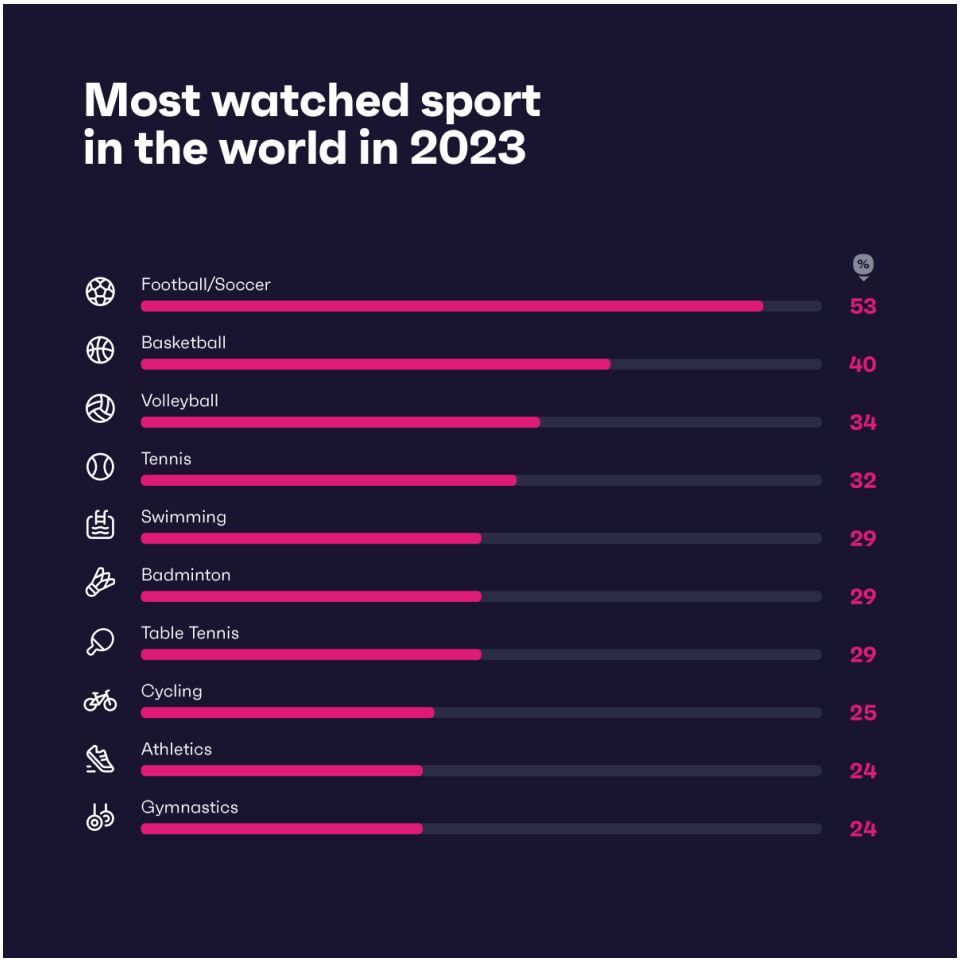


Figure 1.1: Popularity of Basketball [1]

The NBA is the world's foremost men's professional basketball league, established in 1946 with 11 teams. Over the years, the league has grown to include 30 teams through expansion. Since the 2004-2005 season, the teams have been split into two conferences - the East and the West - with three sub-conferences of 5 teams each. A team can have a maximum of 15 players, with only 5 of them allowed to be on the court at the same time.

The professional basketball season consists of two parts: the regular season and the playoffs, with an average length of 8 months. During the regular season, each team plays 82 games, with half of them played at home and the other half on the road. All teams play against each other at least twice during this phase of the season. The main goal of this part of the season is to determine the teams that will qualify for the playoffs (Figure 1.2). The top 8 teams from each conference move on to the playoffs, where their ranking in the regular season plays a crucial role. For instance, in the playoffs, the first-placed team from the Eastern Conference will play against the eighth-placed team from the same conference in a 7-game series, with 4 games played at the higher-ranked team's home and 3 at the lower-ranked team's home.

The first team to win 4 games advances to the next round of the playoffs.

Figure 1.2: NBA Playoffs schedule [2]

The NBA gained immense popularity in the 1980s, thanks to the intense rivalry between Larry Bird and Magic Johnson. These two basketball legends not only led their respective teams but also dominated the league during this decade. However, it was Michael Jordan who brought the NBA to its peak of popularity in the 1990s. Jordan is widely regarded as the greatest basketball player of all time, and his star power and charisma transcended boundaries, making basketball the second most-watched sport in the world.

Sports like basketball offer detailed statistical information as can be seen in Figure 1.3, about both individual and collective performance. In the NBA, numerous individual awards are given out throughout the season, which are based on athletic prowess and also translated into statistical data.

These awards are given to outstanding basketball players who have performed exceptionally well in their respective positions throughout the regular season. The *Most Valuable Player* (MVP) award is given to the player who leads in several statistical categories while keeping their team high in standings. Other awards, such as the Defensive Player of the Year award, are given to players who lead in various defensive categories. The All-NBA team recognizes the top five players in each position based on their statistical performance. All of these awards are based on sports performance and are translated into statistical data.

In terms of team stats, the NBA maintains several rankings during the season as the best team offensively (they typically lead many offensive stat metrics like points, assists, or effectiveness). It also has a ranking for teams in terms of defense in which the ranking is more focused on defensive stats (such as steals, blocks, or for example defensive rebounds).

SEASON LEADERS See All Stats

POINTS PER GAME		REBOUNDS PER GAME		ASSISTS PER GAME	
1. **Indiana Pacers**	**122.8**	1. **Boston Celtics**	**46.9**	1. **Indiana Pacers**	**30.5**
2. Boston Celtics	121.0	2. Golden State Warriors	46.6	2. San Antonio Spurs	29.8
3. Milwaukee Bucks	120.7	3. Utah Jazz	46.2	3. Denver Nuggets	29.2
3. Oklahoma City Thunder	120.7	4. Houston Rockets	46.0	4. Toronto Raptors	29.1
5. Atlanta Hawks	119.0	5. New York Knicks	45.9	5. Golden State Warriors	29.0

BLOCKS PER GAME		STEALS PER GAME		FIELD GOAL PERCENTAGE	
1. **Oklahoma City Thunder**	**6.7**	1. **Philadelphia 76ers**	**8.4**	1. **Indiana Pacers**	**50.4**
2. Boston Celtics	6.6	2. New Orleans Pelicans	8.2	2. Oklahoma City Thunder	50.0
3. San Antonio Spurs	6.4	2. Oklahoma City Thunder	8.2	3. Los Angeles Lakers	49.9
4. Memphis Grizzlies	6.2	4. Orlando Magic	8.1	4. Denver Nuggets	49.6
5. Minnesota Timberwolves	6.1	5. Memphis Grizzlies	7.9	5. Phoenix Suns	49.4

THREE POINTERS MADE		THREE POINT PERCENTAGE		FREE THROW PERCENTAGE	
1. **Boston Celtics**	**1129**	1. **Oklahoma City Thunder**	**39.4**	1. **Oklahoma City Thunder**	**83.2**
2. Dallas Mavericks	1006	2. Boston Celtics	39.0	2. Philadelphia 76ers	83.0
3. Golden State Warriors	991	3. LA Clippers	38.7	3. Utah Jazz	82.7
4. Milwaukee Bucks	979	4. Minnesota Timberwolves	38.6	4. LA Clippers	81.9
5. Sacramento Kings	959	5. New Orleans Pelicans	38.0	5. Miami Heat	81.1

Figure 1.3: NBA Team Statistics [3]

All of the statistical metrics in the NBA make it an area of interest for machine learning models. This is due to the ease of finding correlations with these algorithms.

1.2 Problem

The world of data analysis has undergone a remarkable transformation, providing unique solutions to a range of problems. This change has led to a significant increase in the use of data analysis in the field of sports. As a result, new techniques are being frequently developed to evaluate players and teams, drawing meaningful conclusions about their performance.

There are numerous datasets available that could be used for this project. However, if the goal is to conduct an up-to-date study in this field, it is crucial to use datasets that are current and relevant. Obtaining such data is essential to ensure that the most important information about the game is accurate and up to date. Therefore, valid statistical sites about the game will need to be used to feed the machine learning models.

It is crucial to have a good understanding of the various methods that can be used to acquire datasets. The conventional way of obtaining datasets is by searching on websites such as Kaggle, which has a large community that provides this information for free. However, at times, mobility statistics websites can publish this data more

swiftly. Hence, it would be beneficial to utilize Python in order to better comprehend how to access this information.

Identifying patterns that can impact performance at a sporting level can be a daunting task, especially for those involved in sports organizations or clubs. To achieve the best possible results, it becomes crucial to understand the unique dynamics of each sport. Basketball, for instance, presents interesting characteristics that can be associated with a team's success. While it is common to evaluate points scored and conceded, limiting such a complex sport to just two metrics is not ideal. Therefore, analyzing other metrics that can help to better understand the game becomes necessary.

The project involves web scraping to ensure the usage of the most recent dataset. The use of machine learning algorithms has the potential to transform the way sports are comprehended. It enables coaches, managers, and analysts to make informed decisions based on current conditions. By utilizing machine learning models on databases rich in information, valuable data can be extracted, patterns identified, and future trends anticipated. This contributes significantly to the success of teams and athletes.

1.3 Objectives

The main objective of this thesis is to develop a model for predicting basketball games, focusing specifically on the NBA, which is the most popular basketball league in the world. Although the topic of predicting NBA games is not exclusively academic, it has significant value for a wide variety of audiences, including sports enthusiasts, bettors who wish to base their decisions on prediction algorithms, and analysts who are increasingly relying on this type of algorithm.

This thesis aims to investigate the importance of machine learning algorithms in predicting results in sports contexts, with a specific focus on analyzing basketball games. To this end, a vast dataset will be used that covers a variety of game-related parameters, both at an individual and collective level. The analysis will focus on historical data from different game metrics, to develop a model capable of predicting results as accurately as possible.

The analysis of datasets is a crucial metric that is often disregarded. This work focuses on comprehending the influence of datasets, to understand the impact of two distinct datasets on machine learning. A study will be conducted using both NBA and *Women National Basketball Association* (WNBA) datasets.

In addition, this research will address the analysis of the various techniques employed in predicting sports results through machine learning, to understand the impact that these techniques have on the performance of the prediction model. A comprehensive analysis of the most common techniques will be conducted, including

regression models, neural networks, and decision trees, among others, to assess their effectiveness and suitability for the task of predicting results in basketball games.

In summary, these are the main themes of this work:

- Development of a machine learning model for predicting basketball results
- Processing the information obtained so that it can be used for machine learning algorithms
- Web scrapping website data to obtain the dataset
- Study of the impact of different techniques on the prediction of machine learning models
- Study of the impact of different datasets
- Realization of the project report

The purpose of this work is to make a significant contribution to the fields of machine learning and sports analytics. It aims to underscore the importance of these constantly evolving technologies, which have the potential to provide a significant competitive advantage to those who adopt and apply them effectively.

1.4 Scheduling the Work

Figure 1.4 shows how the time for the dissertation was organized.

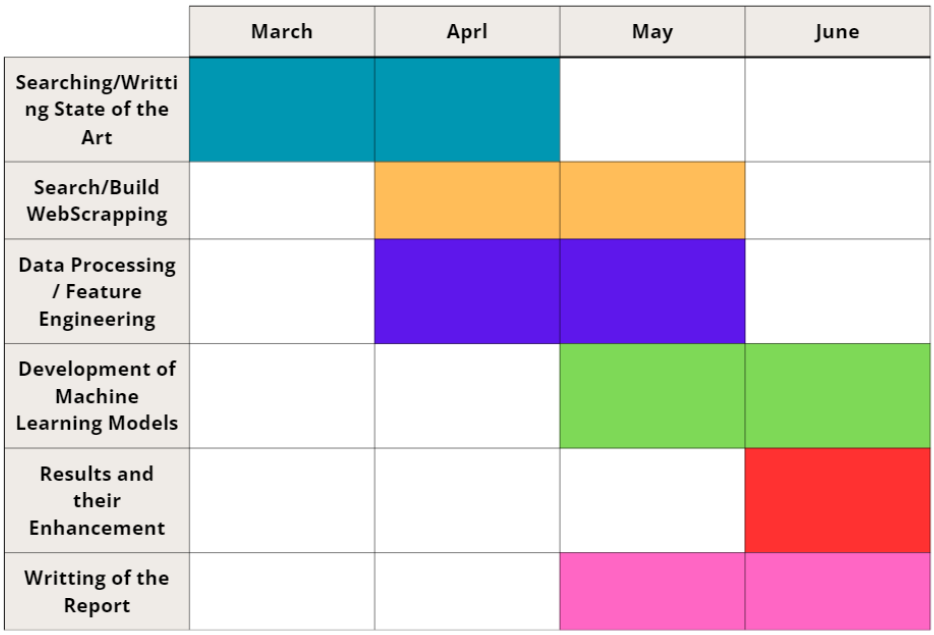


Figure 1.4: Plan of the Work

1.5 Organization of the Dissertation

This dissertation is made up of an introduction in Chapter 1, which provides a theoretical introduction to the topic presented, and describes the problem and how this work will be developed.

Chapter 2 contains a theoretical study of the various topics present in artificial intelligence, the most popular algorithms, the concepts of web scraping and the works studied for the development of the dissertation.

Chapter 3 deals with the architecture of the work and explains which statistics will be applied to the machine learning model.

Following that, Chapter 4 explains how web scraping was carried out and how the data was processed so that it could be used for the dataset.

Chapter 5 discusses the machine learning model from its construction to the selection of features and the resulting feature engineer.

Chapter 6 discusses the results produced by the machine learning model.

Finally, Chapter 7 concludes and discusses improvements that could be made in the future to obtain better predictions.

Chapter 2

State of the Art

This chapter discusses Artificial Intelligence and its subfields such as machine learning and deep learning. Web scrapping will play an important role in the work as it will be used to obtain the dataset and then train the model. Certain articles will also be covered from the thematic point of view of this work.

2.1 Artificial Intelligence

Artificial Intelligence (AI) has been transforming sports and taking them to new heights of success that were once thought impossible. This trend can be observed in various sports, such as baseball, tennis, basketball, and soccer. AI technology provides advanced real-time statistics that enable analysts and spectators to obtain detailed insights about the games. The continued evolution of AI plays a significant role in the development of teams and the fan experience worldwide. It helps players and teams to reach their maximum potential [4].

Artificial Intelligence refers to the creation of machines that can perform functions similar to human brains, such as learning and problem-solving [5]. AI can manifest in various areas or materials, such as sensors, robotics, and geolocation, and can perform tasks that typically require human intelligence or intervention. Examples of AI-powered devices we encounter daily include digital assistants like chat GPT, GPS guidance systems, and autonomous vehicles.

AI has gone through many cycles of increased popularity, but it was when ChatGPT was launched that it attracted a great deal of attention. The last time it

reached this level of popularity was with advances in computer vision. Today, AI can learn to synthesize not only human language but also other types of data, including images, videos, software codes, and even molecular structures [6].

The history of artificial intelligence dates back to 1949 when there was a big problem: computers could follow instructions but could not remember what they had done. In addition, owning a computer at that time was extremely expensive, the price of renting one was around 200 000 USD a month, which meant that only large technology companies or very prestigious universities had access [7].

Between 1957 and 1979 was a period of great progress for AI, but also great difficulty. Several of the programming languages still used today were created during this period, and the fact that the film industry was exploring the idea of robots a lot meant that artificial intelligence gained a peak in popularity around this time. Interesting projects also emerged, such as the first anthropomorphic robot (Figure 2.1) that was built in Japan, or the prototype of an autonomous vehicle. However, the American government did not show much interest in continuing to finance the development of this area of artificial intelligence [8].

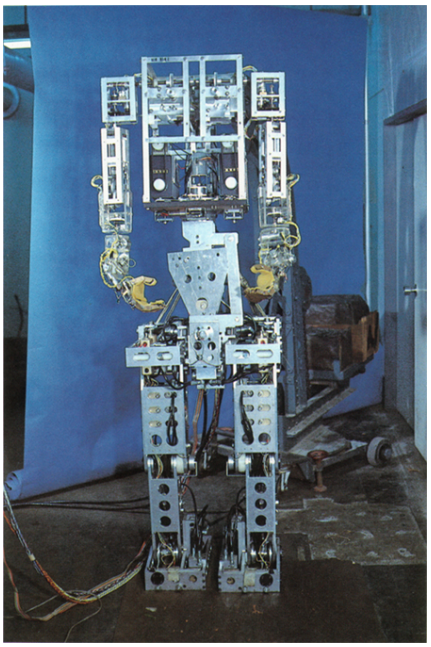


Figure 2.1: First anthropomorphic robot [9]

The period between 1980 and 1987 was characterized as the AI boom, due to the rapid growth and interest that existed during this period. Deep learning became very popular, and this type of technique allowed computers to learn from their mistakes and make independent, conscious decisions. In 1987 Alacrity was launched by the company Alactrious Inc, which was the first software strategy managerial advisory system that used a complex expert system with 3000+ rules [10].

Around 1987, the long-feared AI Winter took place. Various organizations in the field of Artificial Intelligence have already drawn attention to a possible period of low consumption and low investment by private investors in the field. This was because both private and public investors realized that what they had to invest was too high compared to what was the return on investment in that period. This lasted until 1993 and slowed down the AI area considerably [8].

Between 1993 and 2011, there was significant growth in the field of artificial intelligence. During this time, an AI system defeated the world chess champion which was a revolutionary milestone. It showed the promising potential of AI to revolutionize the world as it was seen at the time. Additionally, AI was integrated into everyday life, such as the voice recognition software in Windows computers. This period also witnessed the emergence of innovative projects such as the first automated rover to land on Mars without human intervention. Furthermore, large companies like Twitter and Facebook used AI in the advertising algorithms and product recommendations on their platforms [8]. Lastly, Apple introduced Siri, which was the first virtual assistant that promised to make human daily life much easier.

In summary, Figure 2.2 illustrates the most important events in artificial intelligence. It is crucial to comprehend the developments that have occurred over the years to fully grasp our current level of advancement.

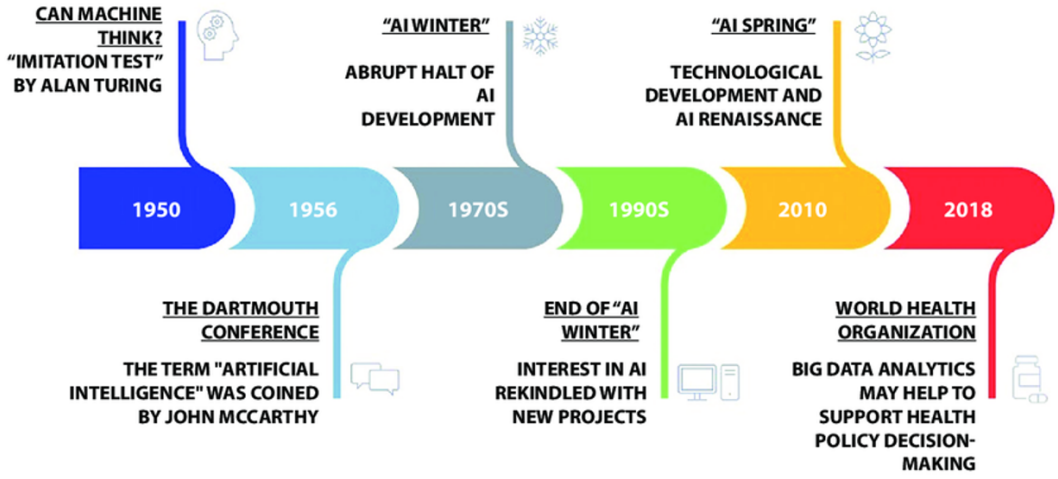


Figure 2.2: Timeline Artificial Intelligence [11]

Today Artificial Intelligence plays a fundamental role in our society, applications such as OpenAI's ChatGPT practically became a personal assistant that contains all the information on the Internet at the touch of a button. Many companies are adopting these models to be more efficient and achieve better results and, as it was expected, sports are following suit and are increasingly relying on this area to get

the best out of each athlete and obtain the best possible results in the competitive context.

2.2 Machine Learning

Figure 2.3 shows that artificial intelligence is divided into sub-fields such as Machine Learning, *Deep Learning* (DL), and Neural Networks. Neural networks are a subset of machine learning, and deep learning is a subset of neural networks.

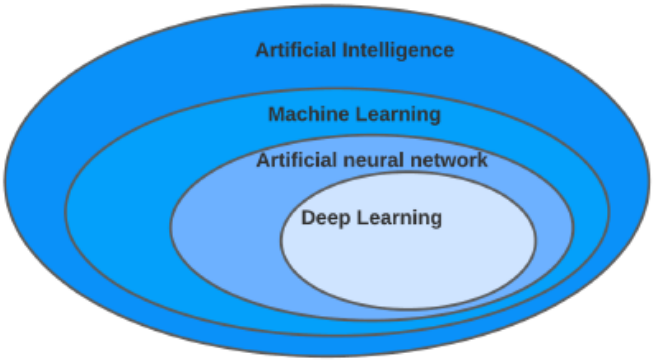


Figure 2.3: Scheme for Artificial Intelligence Concepts

Machine learning is a technique that imitates human intelligence. There are five types of machine learning models: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and self-supervised machine learning. Figure 2.4 explains how the three of the most popular machine learning terms are distinguished.

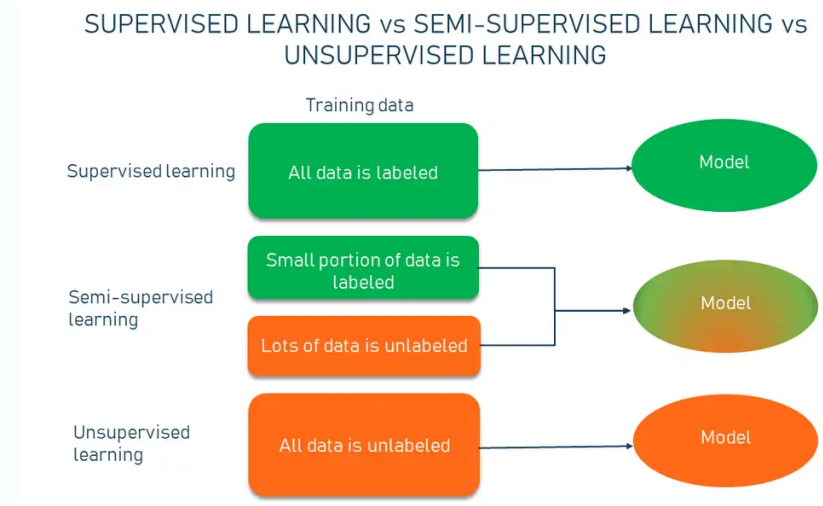


Figure 2.4: Comparison of Machine Learning Types [12]

2.2.1 Supervised Learning

Supervised machine learning involves the development of algorithms capable of discerning patterns and formulating hypotheses based on provided instances to forecast outcomes for future instances. Its primary objective is data categorization utilizing historical information [13]. This learning paradigm is commonly applied to two types of problems: regression and classification.

Regression problems entail fitting data where the output variable assumes values within an interval, on the real axis, or a region on the complex plane. Regression is a fundamental task in machine learning with numerous real-world applications, such as forecasting pocket money indices or predicting inflation.

On the other hand, classification problems are problems in which an object is to be classified in one of the n classes based on the similarity index of its features with that of each class. Classes are groups of identical objects, for example, color, shape, or size [14]. These types of problems are widely used in medical diagnostics and image identification.

Figure 2.5 shows the typical architecture of a supervised model. Supervised learning is a process that involves several stages. The first step is to collect labeled data, which is then cleaned and normalized to obtain accurate results, after this, the appropriate model is selected and trained. During the training process, the model adjusts its parameters to minimize the difference between the predicted and actual labels. Once the model is trained, it should be evaluated using a separate set of data, known as the test data. It is important to adjust the model's hyperparameters while the model is trained and validated to improve its performance. After these steps, the model is ready to be used on data it has never visualized.

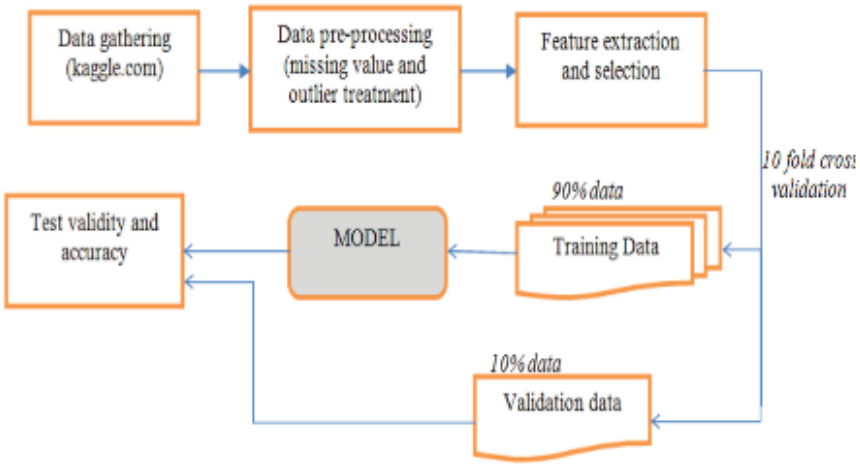


Figure 2.5: Supervised Learning [13]

Linear Regression

Linear regression is a statistical algorithm used in data science and machine learning to establish a linear relationship between an independent variable and a dependent variable, making it easier to predict future results. This model estimates the values of the coefficients used in the representation with the data that are available [15]. Figure 2.6 illustrates how the two variables mentioned earlier react.

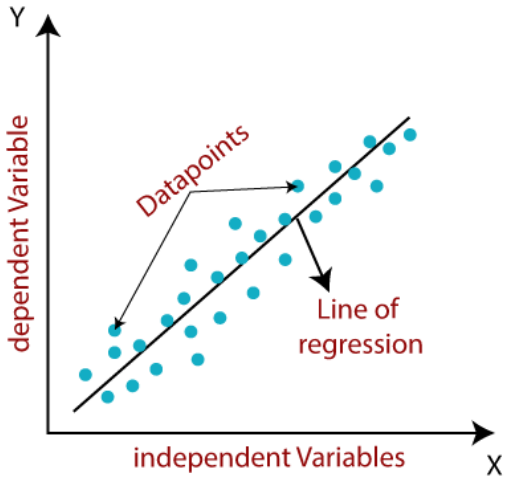


Figure 2.6: Linear Regression [16]

The independent variable, also known as the explanatory variable, remains constant among fluctuations in other variables. On the other hand, the dependent variable varies in response to changes in the independent variable. Through this relationship, the regression model predicts the value of the dependent variable, which serves as the response variable [17].

Equation (2.1) represents Linear Regression.

$$y = a_0 + a_1 x + \epsilon \tag{2.1}$$

Where:

y = Dependent variable (Target variable)

x = Independent variable (Predictor variable)

a_0 = Intercept of the line (Gives an additional degree of freedom)

a_1 = Linear regression coefficient (scale factor to each input value)

ϵ = Random error

It is possible to have more than one independent variable; this type of regression is called Multiple Linear Regression.

Linear Regression is a very advantageous analytical approach when the data set has at least two variables. It is applied in various fields such as stock market forecasting and scientific analysis.

Logistic Regression

Logistic Regression is a machine learning algorithm that is used to predict the probability of certain classes based on some dependent variables. The output of this type of regression is always between 0 and 1 as can be seen in Figure 2.7, which makes it suitable for binary classification tasks.

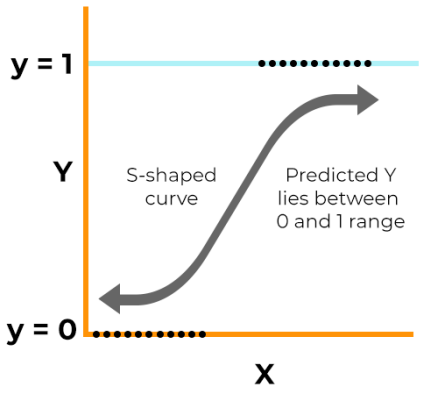


Figure 2.7: Logistic Regression Diagram [18]

This regression, unlike Linear Regression, can accept both continuous and discrete variables in its input and always has the same result in its output 1 or 0 (either it is or it is not).

The main advantages of using this type of regression are:

- Better Performance with Linearly Separable Data: Logistic regression tends to perform well when the data is linearly separable, making it suitable for classification tasks where the classes can be effectively separated by a linear boundary.
- Low Computational Requirements and Interpretability: Logistic regression does not demand extensive computational resources, making it efficient for large datasets. Additionally, its simplicity and linearity make it highly interpretable, allowing for an easy understanding of the model's predictions.
- No Feature Scaling or Tuning Required: Logistic regression does not necessitate feature scaling or extensive hyperparameter tuning, reducing the preprocessing overhead. It can handle input features of varying scales without impacting performance significantly.

- Relevance Measure and Direction of Association: Logistic regression provides insights into the relevance of predictors through the coefficient sizes. Additionally, the direction of association (positive or negative) between the predictors and the outcome variable can be easily inferred from the sign of the coefficients.

This algorithm works by modeling the relationship between independent variables and the probability of a given class. Logistic regression uses the sigmoid function as shown in equation (2.2). The sigmoid is essential because it transforms the linear part of the equation into a probability between 0 and 1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Where:

- x: Input variable, typically the linear combination of predictor variables and coefficients in logistic regression.
- e: Euler's number, approximately equal to 2.71828.

Equation (2.3) represents the general equation for logistic regression :

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}} \tag{2.3}$$

Where:

- $P(Y = 1|X)$: Probability of the dependent variable Y being equal to 1 given the independent variables X.
- β_0: Intercept term.
- $\beta_1, \beta_2, \ldots, \beta_n$: Coefficients of the independent variables.
- $X_1, X_2, \ldots, X_n$: Independent variables.
- e: Euler's number, approximately equal to 2.71828.

The logistic regression algorithm is widely used in predicting weather conditions and sports results.

Naive Bayesian Model

The Naives Bayes Model is a machine learning algorithm that relies on Bayes Theorem to perform classification. It is considered naive because it assumes conditional independence between the features used for classification. It assumes that the presence of a feature in a class is not related to the presence of any other feature. Equation (2.4) represents the Naives Bayes model.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}. \tag{2.4}$$

Where:

- P(A|B) - Probability of A occurring given B;
- P(B|A) - Probability of B occurring given A;
- P(A) - Probability of A;
- P(B) - Probability of B;

Data sets are classified by determining the class of a new data instance based on its features. During training, the model calculates class probabilities and conditional attribute probabilities using the training data. When a new set of data is fed into the model for classification, the model calculates the probability of the data belonging to each class based on its unique characteristics [19]. The class that is assigned to the data is the one with the highest conditional probability.

K Neighbors Classifier

The *K Nearest Neighbors* (KNN) algorithm is extensively employed for classification tasks. It belongs to the family of distance-based machine learning algorithms, meaning it does not explicitly construct a model [20]. Instead, KNN stores training instances and classifies new instances by comparing them to previously observed instances.

In KNN, the parameter K represents the number of nearest neighbors to consider when classifying new data points. This parameter profoundly influences the model's behavior: a lower value of K makes the model more susceptible to noise and outliers, potentially leading to a more irregular decision boundary [21]. For instance, if K equals 3, the three closest values to a given data point would determine its class. Conversely, a higher K value may result in incorrect classification, so selecting an appropriate value of K is critical, often achieved through cross-validation.

KNN shines in scenarios where the decision boundary is complex and irregular, making it well-suited for small datasets. However, its performance may degrade with larger datasets due to increased computational and memory requirements.

Random Forest Model

Random Forest is a very popular and effective machine learning method. This method works by creating several decision trees. Decision trees establish rules for making decisions, and the algorithm creates a structure very similar to a flowchart, with nodes where a condition is checked and if it is met, the flow continues along

a branch, otherwise it goes along another branch to meet the next one, and so on until the end of the tree [22]. Figure 2.8 shows the Random Forest structure.

Data sets are labeled and the goal is to determine the class of a new data instance based on its features. During the training phase, this model calculates the probabilities of each class and the conditional probabilities of each attribute given to each class using the training data. When a new data snapshot is presented to the model for classification, it then calculates the probability of belonging to each class based on the characteristics of the snapshot. The assigned class is the one with the highest conditional probability.

This type of algorithm has 4 stages:

- Sample selection
- Selection of the variables
- Construction of the next trees
- Result

In the first stage, the algorithm randomly selects some samples from the training data, but not all of it. This stage uses a resampling method called bootstrap, where the selected samples can be repeated in the selection. In the variable selection stage for each node, the first node of the tree is selected where the first condition is verified, giving rise to the first two branches, after which the algorithm randomly selects the variables to make new trees and new branches and this is already part of the third stage. In the fourth and final stage, it is possible to visualize the results and understand the path chosen by Random Forest [23].

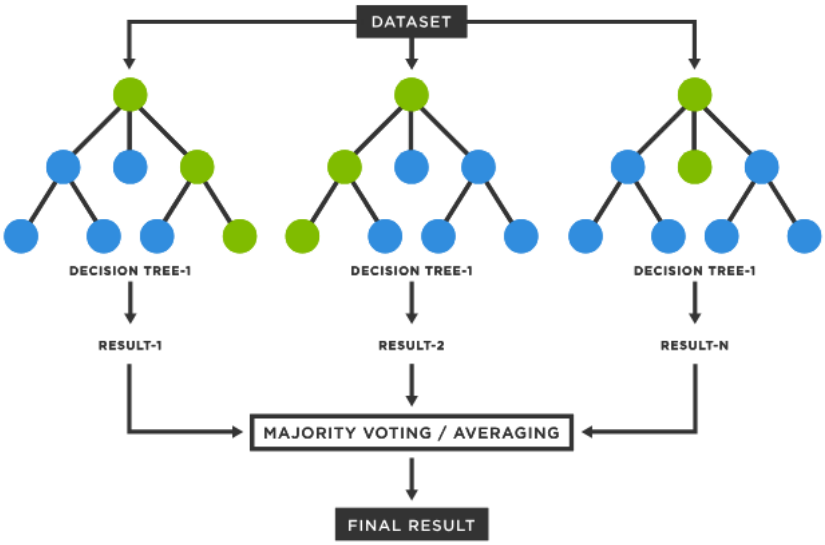


Figure 2.8: Random Forest Model [24]

Support Vector Machines

Support Vector Machines (SVM) are a very effective machine learning algorithm for classification problems in which the data is linearly separable, that is they can be divided into two or more classes by a straight line or hyperplane in higher dimensions [25]. This type of algorithm is chosen to maximize the margin, which is the distance between the hyperplane and the closest data points (known as support vectors) of each class. Maximizing the margin helps to improve the accuracy of the model. Even if the data are non-linear, the SVM solves the problem by mapping the input data onto a higher dimensional space where it can be linearly separable [26].

This method aims to minimize the classification error while maximizing the margin. When the hyperplane is found, the SVM can classify the new data points by determining which side of the hyperplane they lie on. The main advantages of using this technique are:

- Effective in large spaces;
- Memory efficient;
- Versatile, as different kernel functions can be used to handle various types of data distributions;
- Resistant to overfitting (when the model becomes too complex for the amount of data available);

SVM can be quite computationally intensive, especially for large data sets, and may not perform as well as expected if the number of resources is much greater than the number of samples.

XGBoost Classifier

XGBoost is a gradient-boosting machine learning algorithm. This algorithm combines several predictions from various models to produce a final result. Generally, the model use several decision trees, each trained sequentially, and where each tree corrects the residual errors made by the previous trees [27].

This algorithm is specifically designed to handle large amounts of data. It can execute computations quickly and efficiently. As it is being trained, it provides insights into which characteristics contribute the most towards the model's accuracy. Several parameters can be adjusted to achieve optimal results with this model.

2.2.2 Unsupervised Learning

Unsupervised learning is a machine learning paradigm in which the algorithm is trained on unlabeled data, i.e. data in which no expected answers are provided.

The main objective of this type of learning is to explore the intrinsic structure of the data, identify patterns or structures, and learn useful representations [28].

Unlike supervised learning, the algorithm does not receive explicit information about what each instance of data represents. Some of the unsupervised learning algorithms are:

- Clustering
- Association
- Dimensionality reduction

Unsupervised learning is particularly useful for tasks that involve analyzing large volumes of unlabeled data. This method enables businesses to extract valuable insights from data even when no labels are available. It helps them to comprehend the fundamental structure of datasets and discover patterns and relationships between datasets without requiring human intervention [29].

2.2.3 Semi-Supervised Learning

Semi-supervised learning is a hybrid approach between supervised learning and unsupervised learning. In this paradigm, the training data set is composed of a small percentage of labeled data, along with a large amount of unlabeled data.

There are two approaches to this type of learning: transduction and induction. In transduction, the model is trained with both labeled and unlabeled data and makes predictions for the unlabeled data in the training set itself. In induction, the model is trained using only the labeled data and then the unlabeled data is used to adjust or improve the model [12].

One of the major benefits of using semi-supervised learning is that it allows to utilize both labeled and unlabeled data, which can result in the development of more robust and versatile models. This type of learning is commonly employed in systems that involve speech recognition and image pattern recognition [30].

2.2.4 Reinforcement Learning

Figure 2.9 shows the working structure of *Reinforcement Learning* (RL). RL is a type of machine learning based on behavioral psychology, where an agent learns to perform actions in an environment to maximize a cumulative reward over time [31]. The agent is rewarded or penalized for its actions based on an established metric (usually points), encouraging the agent to continue with good practices and discard bad ones. The Q function represented by Equation (2.5), also known as the action value function, is a crucial concept used to estimate the quality of a given action in a given state. This function is defined as the expected cumulative reward obtained

by taking an action a in a state s, and then following a policy π to select subsequent actions.

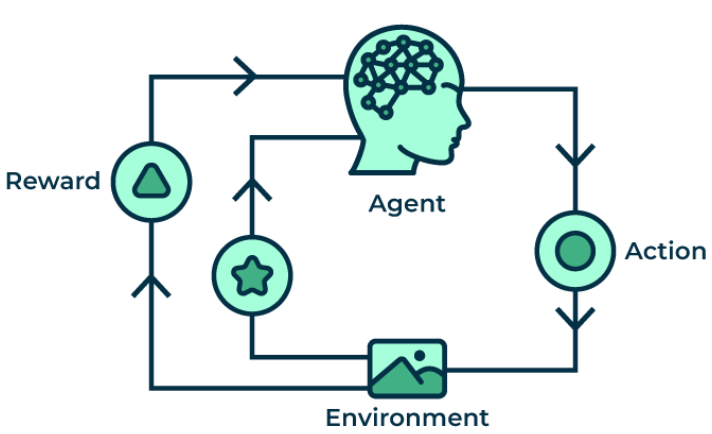


Figure 2.9: Reinforcement Learning [32]

$$Q(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a, \pi\right] \tag{2.5}$$

Where:

- s: Current state;
- a: Action taken in that state;
- r_t: Reward received at time step t;
- γ: Discount factor (determines the importance of future rewards relative to immediate rewards);
- π: Policy used to select actions.

Through repetition, the agent learns the best strategies. The most popular Reinforcement Learning models are:

- Q-learning: is an algorithm that maps states to actions. The Q-function estimates the expected reward for a given action in a given state.
- *State-Action-Reward-State-Action* (SARSA): is another RL model that learns the Q function. However, this model updates the Q-function to the action that was taken instead of the optimal action.
- Deep Q-learning: this model is a combination of Q-learning and deep learning. Q-learning uses a neural network to display the Q function, which allows it to learn complex relationships between states and actions.

RL algorithms are common in video game development and are used to teach robots to reproduce human tasks.

2.2.5 Self-Supervised Learning

Self-Supervised Learning (SSL) allows models to train themselves with unlabeled data, instead of needing labeled data in bulk. These algorithms learn one part of the input from another part, automatically generating labels and transforming unsupervised problems into supervised problems [31].

There are two main tasks in SSL: the first, the pretext task, aims to guide the model to learn intermediate representations of data, which is very useful for understanding the underlying structural meaning. The downstream task is the process of transferring pretext knowledge to a specific task. This type of task is provided with a smaller amount of tagged data [33].

These types of algorithms are used in computer vision and *Natural Language Programming* (NLP), where the amount of labeled training data needed to train models can be exceptionally large.

2.3 Artificial Neural Network

The *Artificial Neural Network* (ANN), also known as a neural network, is a computational structure inspired by the workings of the human brain. Neural networks are composed of layers of several nodes, containing an input layer, one or more hidden layers, and an output layer [34]. Each node, or artificial neuron, connects to others and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node will be activated, sending data to the next network layer [35]. These artificial neurons, also known as perceptrons, are the basic processing units in an ANN. Equation (2.6) defines the output of the percepton.

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right) \tag{2.6}$$

Where:

- y is the output of the perception;
- f is the activation function;
- w_i are the weights associated with each input feature x_i;
- x_i are the input features;
- b is the bias term.

Neural networks rely on training data to learn how to improve their accuracy over time. ANN are algorithms that can perform tasks like recognizing human speech or images in just minutes, which could take hours for human specialists. Figure 2.10 illustrates the architecture of an artificial neural network. One of the most famous Artificial Neural Networks is the Google search engine.

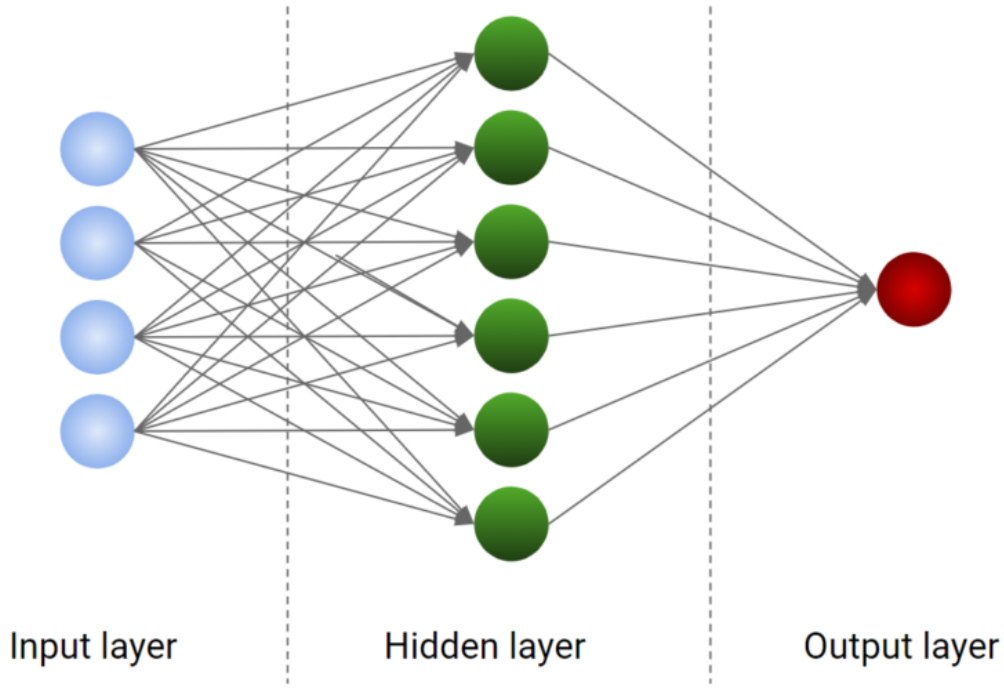


Figure 2.10: Artificial Neural Network architecture [36]

After defining the input layer, the weights are defined. These weights help determine the importance of any given variable, with the largest contributing significantly to the output compared to the other inputs. All the inputs are then multiplied by their respective weights and then added together. The output is then transmitted via an activation function, which determines the output. If this output exceeds a certain level, it will trigger (or activate) the node, transmitting data to the next layer in the network. This results in the output of one node becoming the input of the next node. This process of transmitting data from one layer to the next defines this neural network as a feedforward network [37].

The goal is to minimize the cost function to ensure the correctness of the fit concerning any given observation. As the model adjusts its weights and biases, it uses the cost function and reinforcement learning to reach the point of convergence, or local minimum. The process in which the algorithm adjusts its weights is by gradient descent, allowing the model to determine the direction to take to reduce errors (or minimize the cost function). With each training example, the model parameters are adjusted to gradually converge on the minimum.

2.3.1 Deep Neural Networks

A *Deep Neural Network* (DNN) is a much more complex network than an Artificial Neural Network. Figure 2.11 demonstrates the complexity of DNN. They have many more layers than ANN, which makes them more likely and quicker to learn information and apply it correctly [38].

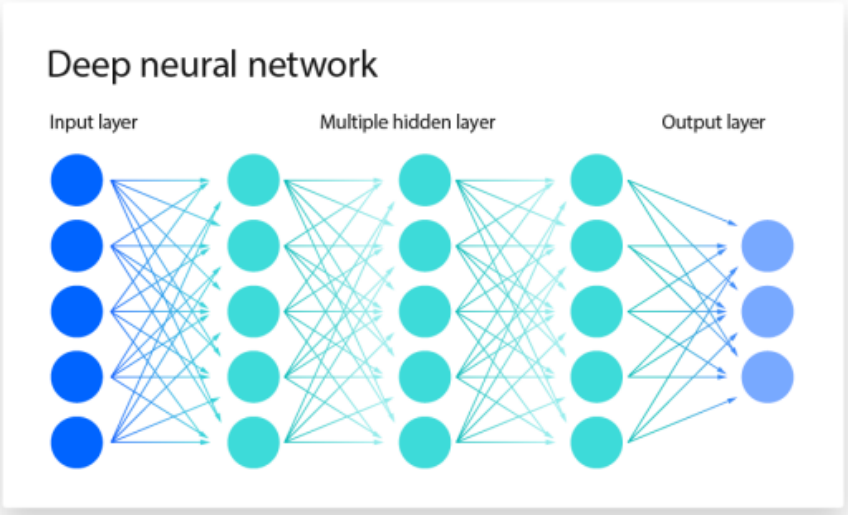


Figure 2.11: Deep Neural Network [34]

Convolutional Neural Network

Convolutional Neural Network (CNN) is a class of deep neural networks that specializes in processing data with a grid-like topology, such as an image. A digital image is a graphical representation of visual data, containing a series of pixels arranged in a grid-like pattern that contains pixel values to indicate the brightness and color of each pixel [32].

The human brain processes an enormous amount of information the moment an image is seen, each neuron working in its receptive field and connected to other neurons to cover the entire visual field [39]. Each neuron in the CNN processes only its receptive field, the layers are arranged in such a way that they first detect simpler patterns and only then more complex ones. Figure 2.12 shows the Convolutional Neural Network architecture.

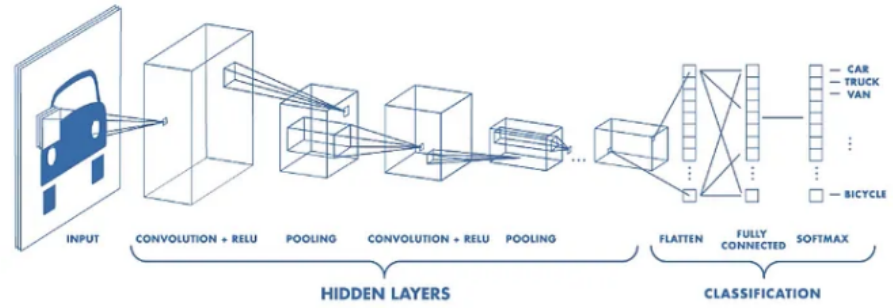


Figure 2.12: Convolutional Neural Network [40]

This type of network does not require manual feature extraction. CNN filters represent several receptors that can correspond to different characteristics, and the activation functions mimic the function in which only neural electrical signals above a certain threshold can be sent to the next neuron [41]. CNN has several advantages over fully connected systems:

- Local connections - Each neuron is no longer connected to all the neurons in the previous layer, but only to a small number of neurons, which helps to reduce parameters and speed up convergence.
- Sharing weights - when a set of connections shares the same weights, the parameters are reduced even further.
- Dimension reduction through downsampling - a pooling layer uses the notion of local image correlation to downsample the resolution of an image, which can minimize the amount of data while retaining important information. This can also reduce the amount of parameters by excluding unnecessary features.

Four components are needed to develop a CNN model. Convolution is the process of extracting features. Padding to increase the input value to 0. Stride is used to modulate the convolution density. Finally, pooling to eliminate redundancies [41].

Feed-Forward Neural Network

Feedforward Neural Networks represent one of the simplest architectures. This type of network directs its information unidirectionally, i.e. from input to output only, as shown in Figure 2.13 [42].

By not having cyclical connections between its nodes, it ensures that the information only flows in one direction, from the input layers through the hidden layers and finally to the output layers. In this way, no feedback can affect the inputs.

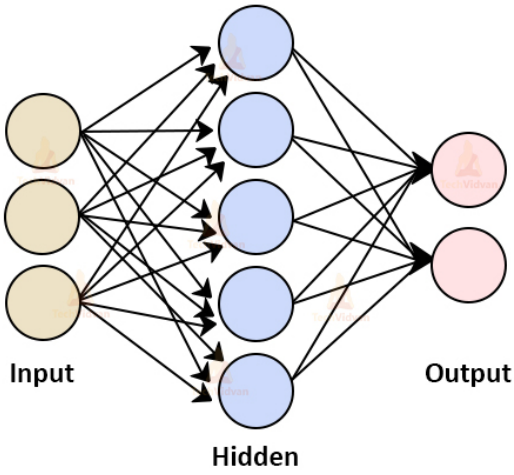


Figure 2.13: Feed-Forward Neural Network [43]

Recurrent Neural Networks

A *Recurrent Neural Network* (RNN) is a neural network that uses sequential or time series data. The RNN takes into account information from previous inputs to influence the current input and output, as can be seen in Figure 2.14. Another distinctive feature of this type of network is that they share parameters in each network layer, i.e. the weights have the same weight within each network layer [44]. These weights are adjusted using backpropagation and gradient descent to facilitate reinforcement learning.

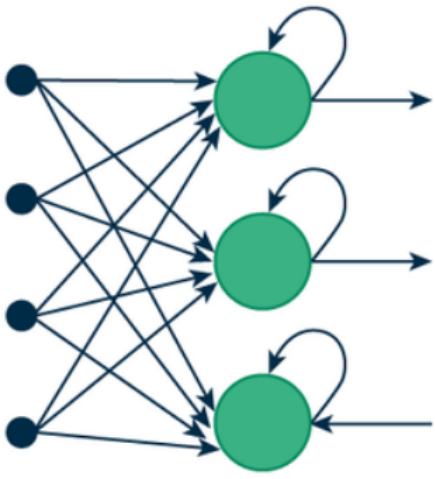


Figure 2.14: Recurrent Neural Networks [44]

This type of algorithm often faces two problems. One of them is known as fading gradients, which results when the gradient is very small, it keeps decreasing, updating the weight parameters until they become insignificant, i.e. 0. As a result, the algorithm stops learning. Explosive gradients occur when the gradient is too

large, creating an unstable model. In this case, the weights of the model will become too large and will give rise to undefined results. The solution to these problems is to reduce the number of layers in the neural network, but this will eliminate much of its complexity [45].

Long Short-Term Memory

Long Short Term Memory (LSTM) neural networks are a type of sequential neural network that allows information to persist over time, making it particularly effective for tasks involving sequential data [46]. The key to this type of network lies in its great ability to selectively retain or forget information over long sequences, thus mitigating the fading gradient problem.

Each cell in the LSTM network has three phases known as gates, as shown in Figure 2.15.

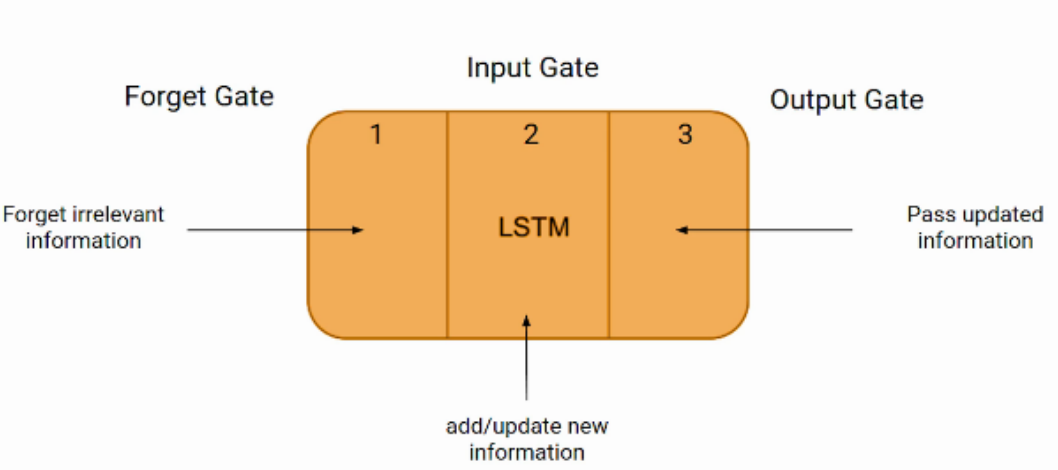


Figure 2.15: LSTM Architecture [47]

These phases known as gates place the flow of information into and out of the LSTM memory. The first gate is called the forget gate, this gate decides whether to keep or forget the information that was passed on by the previous cell. Equation (2.7) shows the output of the forget gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.7}$$

Where:

- f_t is the output of the forget gate at time t;
- σ is the sigmoid activation function;
- W_f is the weight matrix for the forget gate;
- h_{t-1} is the previous hidden state;

- x_t is the current input;
- b_f is the bias for the forget gate.

The input gate is used to quantify the importance of the information in the cell. The result will be a number between zero and one, with zero being less important and one being more important. Equation (2.8) shows the output of the input gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.8}$$

Where the terms are similar to the forget gate.

The Candidate Cell State is an intermediate representation that reflects the new information proposed to be added to the state of the memory cell. The Candidate Cell State is defined by Equation (2.9):

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.9}$$

Where:

- $\tilde{C}_t$ is the proposed update to the cell state at time t;
- tanh is the hyperbolic tangent function;
- W_c is the weight matrix for the cell state update;
- b_c is the bias for the cell state update.

The Cell State Update in an LSTM unit is a crucial step in maintaining and modifying the internal state of the memory cell over time. At each time step, this cell needs to decide how much of the previous information should be forgotten and how much should be added or updated. Equation (2.10) is used to update the state of the cell:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.10}$$

- C_t is the updated cell state at time t;
- i_t is the input gate vector at time t.

The Output Gate is responsible for controlling how part of the memory cell state is used to calculate the current hidden state, which is the output of the LSTM unit. In the Output Gate Equation (2.11), the calculation is done as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.11}$$

Where:

- o_t is the output of the output gate at time t;
- σ is the sigmoid activation function;
- W_o is the weight matrix for the output gate;
- b_o is the bias for the output gate.

The Hidden State is a representation of the information that is transmitted to the next processing stage in the neural network. This hidden state contains important information and encapsulates the knowledge learned by the LSTM unit over time. In the LSTM unit, the calculation of the Hidden State is determined by the equation (2.12):

$$h_t = o_t \cdot \tanh(C_t) \tag{2.12}$$

Where:

- h_t is the hidden state at time t.

The bidirectional LSTM consists of two layers, processing information back and forth, which unlike the traditional LSTM only flows in one direction. This allows the network to access information from past and future steps simultaneously. Incorporating these two functions improves the model's ability to capture long-term dependencies and make more accurate predictions on more complex sequential data [47].

2.4 Deep Learning

Deep Learning is a subset of machine learning, where it is essentially a neural network with three or more layers and can carry out feature analysis. The main goal of these neural networks is to simulate the behavior of the human brain, allowing them to learn from large amounts of data. Although a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize performance [48].

Deep Learning differs from classic machine learning in the type of data it works with and the methods it uses, as shown in Figure 2.16.

Machine learning algorithms use structured and labeled data to make precisions, which means that specific features are defined from the model's input data and organized into tables. This means that if unstructured data are used, they usually go through some pre-processing to organize them into a structured format [49].

Deep learning is a type of machine learning that involves multiple layers of non-linear processing units that extract and transform features. The lower layers, which are closer to the input data, learn simple features, while the higher layers learn more

complex features that are derived from the lower layer features. This architecture creates a hierarchical and powerful feature representation, which makes deep learning ideal for analyzing and extracting useful knowledge from large datasets and data from various sources [50].

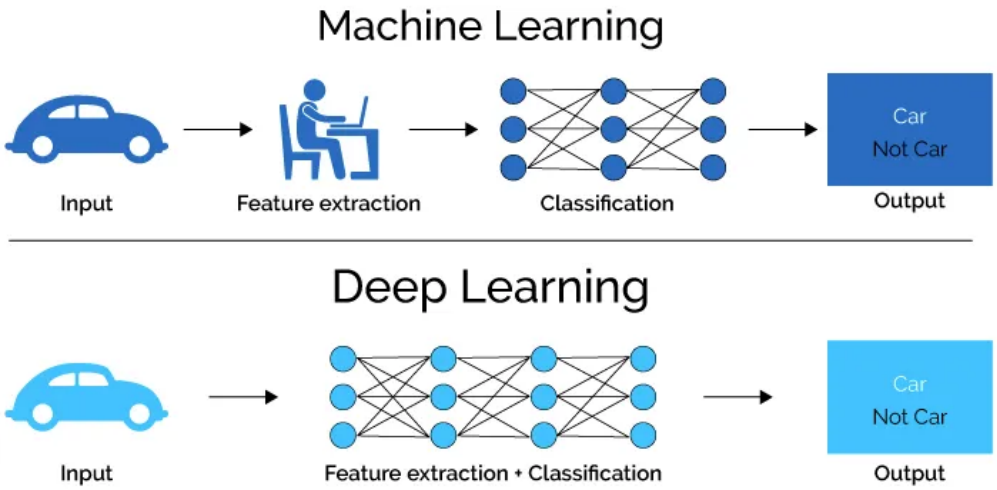


Figure 2.16: Machine Learning vs Deep Learning [51]

This type of learning eliminates some of the data pre-processing that is normally involved in machine learning. These algorithms can process unstructured data such as text and images and automate feature extraction, removing some of the dependence on human experts. To demonstrate an example, let's say we want to classify a set of animal photos and we want to categorize them by "dog" and "cat", Deep Learning algorithms can determine which features are most important to distinguish one animal from the other. Using gradient descent processing and back propagation, deep learning adjusts its accuracy to be as effective as possible. In machine learning, a human expert usually establishes this hierarchy of features [49].

Deep Learning models are also capable of different types of learning, such as supervised and unsupervised learning.

This type of model consists of several layers of interconnected nodes, each built on the previous layer to refine and optimize prediction or categorization. The input and output layers are called visible layers [52]. The input layer is where the deep learning model ingests the data for processing and the output layer is where the final prediction or classification is made.

The process called backward propagation uses algorithms, such as gradient descent, to calculate the errors in the predictions and then adjusts the weights and biases of the function by moving backward through the layers to train the model. Forward propagation and backward propagation allow a neural network to make

predictions and correct any errors accordingly. Over time, the algorithm gradually becomes more accurate.

Deep learning requires an enormous amount of computing power. High-performance GPUs are ideal because they can handle a large flow of calculations on several cores with plenty of memory available.

2.5 Web Scraping

Web Scraping is the process of collecting and parsing raw data from the Web. It is a solution for accessing web data in an automated way [53]. The process usually involves writing a script that accesses a web page, analyzes its content, and extracts the relevant data. The main use cases for web scrapping are to obtain datasets for training artificial intelligence.

Web Scraping is very similar to manually copying and pasting information from a website, it performs the same function only on a large scale and in an automated way. The real power of this technology lies in its ability to build and power some of the most revolutionary commercial applications in the world [54]. Figure 2.17 explains the processes involved in this method of obtaining data from websites.

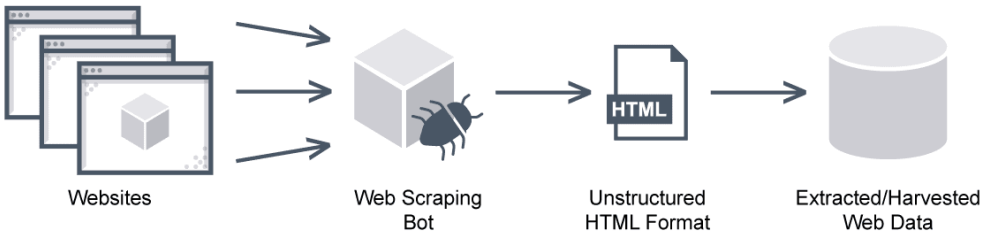


Figure 2.17: Web Scrapping process [55]

The principles of Web Scraping lies in two principles: a web crawler and a web scrapper. The web crawler is an artificial intelligence app that browses the Internet to index and search for content, following links and exploring content as if it were a real person [56]. A web scraper is a specialized tool designed to extract data from web pages accurately and quickly. This tool varies greatly in design and complexity depending on the project. An essential part of a web scraper are the data locators used to find the data to be extracted from the page. This parameter varies from page to page, so it is necessary to know the structure of *HyperText Markup Language* (HTML) pages to be able to adapt the web scraper to the specific web page. In more popular terms, the web crawler works like a horse and the web scraper is like a carriage. The crawler drives the scrapper around the Internet where it extracts the data [57].

Typically, the Web Scraping process goes through the following stages:

- Identifying the target website.
- Collect URL of the pages from which the data is to be extracted.
- Request for these URL to obtain the HTML of the page.
- Use localizers to find the HTML data.
- Saving the data in a *JavaScript Object Notation* (JSON), *Comma-Separated Values File* (CSV), or other structured format.

Web Scrapping must be used ethically and legally. Certain websites prohibit scraping in their terms of service and extracting data from them without permission can result in legal consequences [58].

There are several ways to exploit Web Scraping. There are browsers that support data extraction, there are also browser extensions and some websites also provide information on how to use this tool. There are also online services that provide such a tool, such as Diffbot, Octoparse, Import.io, and ParseHub. Programming languages such as Python are very suitable for this purpose.

2.5.1 Web Scraping in Python

Python is one of the most popular programming languages for Web Scraping due to its simplicity, wide range of libraries, and open-source code. The most widely used libraries for Web Scraping are BeautifulSoup, Scrappy, Selenium, and Playwright.

BeautifulSoup

BeautifulSoup is a popular Python library for analyzing HTML and *Extensible Markup Language* (XML) documents. This library provides convenient ways to extract data from these documents, making the web scraping process easier and more efficient. Asynchronous processing is supported, which is advantageous when dealing with large amounts of data. BeautifulSoup allows to navigate through the structure of the document, moving from element to element until you find the code where the desired extraction is located. It provides functions to extract text, attributes, and content using simple commands. It also allows to manipulate the data by removing or adding new elements.

Scrappy

Scrapy is a Python framework designed exclusively for web scraping, providing a powerful and flexible architecture for extracting data from large-scale websites. Its modular architecture allows pipelines, and items to be developed and reused.

Scrappy automatically manages the crawling process, handling HTTP requests, processing responses, and track reads. It also supports asynchronous processing, which is very advantageous when dealing with big data and allows to increase the crawling speed. It provides visualization tools that can make studying the site much more intuitive. This framework provides plenty of tools for web scraping that can be applied to various projects with different data sizes and different levels of processing.

Selenium

Selenium is a tool designed to automate interactions with web browsers. It is normally used for test automation in web applications, but it is also used for web scraping when interaction with JavaScript is required or when the web page has dynamic web technologies that can make the traditional scraping task more difficult. This tool allows to automate and control interaction with web browsers and supports various browsers such as Google Chrome, Firefox, and Opera. It is supported not only in the Python programming language but also in Java, C, Ruby, and JavaScript. This library is very interesting because it can support more complex web pages than other libraries.

Playwright

Playwright is a browser automation library that allows to control and interact with web browsers programmatically. Developed by Microsft, this library offers a simple and effective API for automating tasks in browsers such as Chrome, Firefox, and Safari. Playwright allows you to control all browser functions, including page handling, cookies, screen capture, and JavaScript execution. It is possible to run the browser headless (without a graphical interface), which makes it easier to extract information in terms of resources. Playwright is also supported in several languages such as TypeScript and Java.

2.6 Papers analyzed

This section analyzes existing machine-learning projects involving basketball.

2.6.1 Predicting the outcome of NBA games with Machine Learning

The study "Predicting the outcome of NBA games with machine learning" [59] compared predictions based on team and player statistics. The task has been successfully completed, firstly by web scraping the Synergy Sports website, an NBA statistics website that is paid for and only accessible through universities or companies. Synergy Sports has access to various statistics that would not normally be found on

common NBA data systems, as they have cameras all over the court that give them more detailed information about the game. The authors of this work scraped twelve seasons from the website.

One of the great challenges of the work was to clean up the data obtained from the web scrapping since they obtained the data of each team's players and team data. The task was to clean up the data by removing invalid data or information that was not considered relevant to the work. Once the data had been cleaned, it was also saved in a CSV file so that it could be used.

To improve prediction accuracy, data features were added, including team form parameters for the last ten games, Elo rating, individual player stats for the last ten games and overall performance, and a player-only rating system based on points, assists, and other statistics to rank the best players in the league.

The dataset was divided into 80 % for training and 20 % for testing. Two models were used: Logistic Regression and Random Forest. The model used for RandomForest obtained an accuracy of 67.15 %, which was slightly higher than the model used by Logic Regression.

Player statistics were used to predict NBA match results. 80 % of the dataset was used for training and the remaining 20 % for testing. Linear regression obtained the best results with an accuracy of 58.66 % which was a significantly lower value than the team statistics.

2.6.2 Predicting the Outcome of NBA Games

In thesis [18], the author compares various Machine Learning methods to see which ones give the best results to predict NBA games. Collective team statistics are used as a dataset. The dataset for this work was obtained from an open-source Python library known as the "nba_api package". Through this library, it is possible to retrieve data from the NBA's official website. The dataset used includes the 2018 season up to and including the 2021 season, thus accounting for four seasons.

Exploratory analysis was conducted to understand which characteristics were significant for the model. The lower the value of the left-out features, the higher the odds of winning the game.

To improve the prediction model's accuracy, the author recommends adding more variables like the team's recent performance in the last ten games. The Elo rating is also crucial for creating a separate ranking based on this formula. These two metrics will provide complementary information to the model.

The model was trained on a dataset split into 75:25 ratio for training and testing. 30 features were used, and 6 algorithms were tested including Logistic Regression, Random Forest Classifier, K Neighbors Classifier, SVC, Gaussian NB, and XGB Classifier. The model was evaluated using five classification options.

The best classification model was Gaussian Naive Bayes with 65 % accuracy. The author suggests that tweaking the parameters of some techniques could improve their performance. A prediction accuracy above 60 % is considered good since home teams in the NBA statistically win about 60 % of their games.

2.6.3 Predicting the Outcome of NBA Games with Machine Learning Algorithms

The thesis [60] was developed to test various machine learning algorithms for perverting the results of basketball games, specifically NBA games. A simple betting system was built from the results of the model. One of the aims of this work was to exploit the inefficiencies of the betting market and take advantage of the market to make a profit.

The dataset spans 15 years, covering around 20 000 matches from the 2003-04 to 2017-18 season.

Using the dataset, various features were created using Feature Engineering. Features such as averages of the last 10, 7, or 5 games were added, and the Elo Rating was also added. A feature that shows the squad fit for each game, as well as players who were elected to the best starting five of last season. This information can additionally indicate to the algorithm the best players available for each game.

Several machine learning algorithms were tested, but Decision Tree obtained an accuracy of 68.6 %. In terms of betting accuracy, a return on investment of 4 % was obtained, which proves that despite the advances made in Artificial Intelligence, the creation of a compact model for betting purposes is still some way off.

2.6.4 Injury Analysis Based on Machine Learning in NBA Data

Study [61] focuses on understanding the causes of injuries in the NBA world. The authors of this paper study how game statistics can contribute to analyzing why injuries happen. Since the NBA has 82 games for each team per season, it is very common for injuries to occur due to the overload of games.

The data are from the pro sports transaction website and includes injury data from 2011 to 2019. Only injuries that occurred during games were taken into account. The game statistics were obtained from the Basketball Reference website.

The variables chosen to be used in the model are measured according to their importance to the algorithm being studied. The algorithm has access to the complete dataset and will see which variables have the greatest weight for injury prediction. The algorithm used for these predictions was RandomForest.

The paper shows that match overload is an essential factor in contributing to injuries. Statistics such as minutes played during the season, the number of times you shoot from 3 points, or even the players with the best defensive performance

are more at risk of injury. There has been a lot of discussion about reducing the number of games per season and it could be a good opportunity to move towards a plan that could implement this measure and also maintain the NBA's financial model.

2.6.5 Algorithms Used

Table 2.1 details the algorithms used in each article studied.

Table 2.1: Algorithms of the Works Analyzed

Article	**Algorithm**
[59]	RandomForest Linear Regression Logistic Regression
[18]	Gradient Boosting K-Nearest Neighbors Logistic Regression RandomForest Support Vector Machine XGBoost
[60]	XGBClassifier Bagging Classifier Logistic Regression KNeighbors Classifier SVC RandomForestClassifier MLPClassifier AdaBoostClassifier DecisionTreeClassifier SGDClassifier
[61]	RandomForest

Chapter 3

Architecture and Feature Analysis

This chapter discusses the structure of the work, describing the stages using a block diagram, justifying the selection of the dataset, and specifying the metrics and the value of feature engineering in order to obtain the best predictions.

3.1 Block diagram

The work will focus on the block diagram represented in Figure 3.1. Although all the processes are essential, feature engineering will be an essential step in obtaining the best results. Below is a brief description of the processes depicted:

- Web Scraping: The data will be scraped through an NBA statistics website, using the Python programming language.
- Data Processing: After obtaining the data, it is necessary to process it, remove unnecessary information and filter the necessary information for a dataset.
- Feature Engineering: New features will be obtained from other features. It is important to simplify the information as much as possible so that machine learning has access to the relevant features it needs.
- Machine Learning: Different machine learning techniques and two different datasets will be tested, one from the NBA and the other from the WNBA.

- Results: Analysis of results and subsequent conclusions of the study.

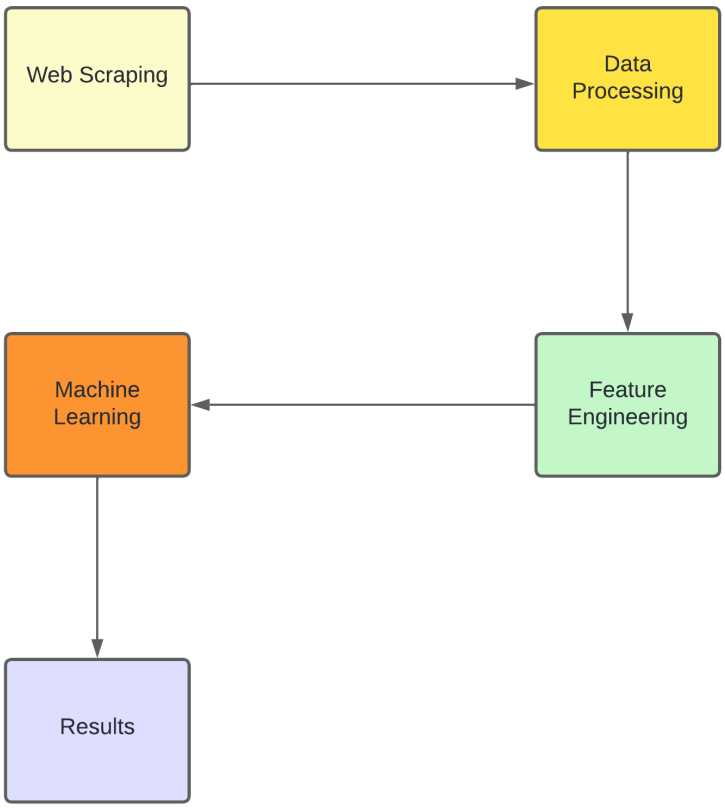


Figure 3.1: Diagram of Process

3.2 Metrics

Initially, it is necessary to define the seasons from which the data will be collected. Figure 3.2 demonstrates the impact that the metrics related to the 3 points have gained too much importance not to be a key factor when it comes to obtaining data [62]. As can be observed, from the 2009-2010 season to the 2017-2018 season, the number of triples per season has almost doubled. As an example, in the 2024 season, the team in first place in the Western Conference is the one with the highest shooting efficiency from the three-point line in the entire league. It might not be worth considering earlier seasons because the statistics related to 3-pointers would not have the desired impact when training the model. Data was then taken from the 2015-2016 season to the recent 2023-2024 season. In this way, it will be possible to ensure when training the model that the statistics related to the 3 points will have the proper impact, and the model will have eight seasons to train which conveys robustness concerning the dataset.

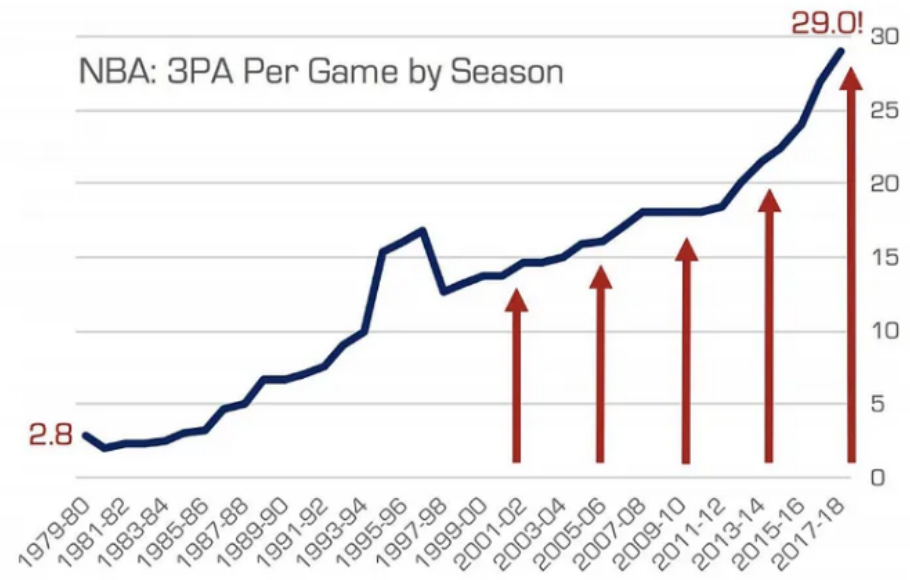


Figure 3.2: Evolution of the 3-point shot [62]

The website that will be used to carry out the scrapping is Basketball Reference, which is complete in terms of statistics on games and players. The article [59] mentions that the Basketball Reference website is a valid alternative to train machine learning models.

Basketball statistics are quite different from other sports, as it is considered a fast-paced sport in the sense that the score fluctuates a lot during the game, and it turns out to be crucial to analyze all the metrics available during the game. Figure 3.3 shows a normal box score of an NBA game, 33 metrics will be considered from this website.

	Basic Box Score Stats																		
Starters	**MP**	**FG**	**FGA**	**FG%**	**3P**	**3PA**	**3P%**	**FT**	**FTA**	**FT%**	**ORB**	**DRB**	**TRB**	**AST**	**STL**	**BLK**	**TOV**	**PF**	**PTS**	**+/-**
Anthony Davis	40:37	16	24	.667	0	0		9	13	.692	5	15	20	5	0	4	4	2	41	
LeBron James	34:52	10	21	.476	0	1	.000	4	5	.800	1	10	11	4	2	0	6	5	24	
Cam Reddish	33:19	4	7	.571	1	3	.333	0	0		2	1	3	1	0	3	3	5	9	
D'Angelo Russell	31:53	6	15	.400	0	3	.000	1	2	.500	1	3	4	7	1	1	2	1	13	
Taurean Prince	18:59	2	3	.667	1	2	.500	1	1	1.000	0	3	3	2	0	0	1	4	6	
Team Totals	**240**	**47**	**88**	**.534**	**2**	**13**	**.154**	**27**	**35**	**.771**	**12**	**43**	**55**	**25**	**5**	**10**	**18**	**25**	**123**	

	Advanced Box Score Stats														
Starters	**MP**	**TS%**	**eFG%**	**3PAr**	**FTr**	**ORB%**	**DRB%**	**TRB%**	**AST%**	**STL%**	**BLK%**	**TOV%**	**USG%**	**ORtg**	**DRtg**	**BPM**
Anthony Davis	40:37	.690	.667	.000	.542	16.9	34.1	27.2	21.0	0.0	8.8	11.9	32.8	134	97	
LeBron James	34:52	.517	.476	.048	.238	3.9	26.5	17.4	16.6	2.6	0.0	20.5	33.1	94	100	
Cam Reddish	33:19	.643	.643	.429	.000	8.2	2.8	5.0	3.5	0.0	8.0	30.0	11.9	98	107	
D'Angelo Russell	31:53	.409	.400	.200	.133	4.3	8.7	6.9	27.8	1.4	2.8	11.2	22.2	100	106	
Taurean Prince	18:59	.872	.833	.667	.333	0.0	14.6	8.7	12.1	0.0	0.0	22.5	9.2	141	109	
Team Totals	**240**	**.595**	**.545**	**.148**	**.398**	**34.3**	**82.7**	**63.2**	**53.2**	**4.7**	**18.5**	**14.8**	**100.0**	**116.6**	**103.3**	

Figure 3.3: Basketball Reference Box Score

Table 3.1 shows a detailed description of the box score statistics.

Table 3.1: Basic Box Score Status

Abbreviation	**Meaning**	**Description**
MP	Minutes Played	Minutes played by the team
FG	Field Goals	Number of shots attempted by the team that produced points
FGA	Field Goals Attempts	Number of shots attempted by the team
FG%	Field Goal Percentage	Percentage of baskets that produced points for the team
3P	3-Point Field Goal	Number of shots attempted by the team
3PA	3-Point Field Goal Attempted	Number of 3-point shots attempted
FT	Free Throw	Number of Free Throws made
FTA	Free Throw Attempts	Number of free throws attempted
FT%	Free Throw Percentage	Percentage of free throws made
ORB	Offensive Rebounds	Number of offensive rebounds
DRB	Defensive Rebounds	Number of defensive rebounds
TRB	Total Rebounds	Number of rebounds
AST	Assists	Number of assists made by the team
STL	Steals	Number of steals made by the team
BLK	Blocks	Number of blocks made by the team
TOV	Turnovers	Number of turnovers (lose ball) made by the team
PF	Personal Faults	Number of faults made by the team
PTS	Points	Points made by the team

All of these metrics are important, but some end up being more prominent among popular analysts of the game of basketball. The Field Goal, which indicates the team's shooting efficiency, is an important statistic, especially when evaluating individual players. Due to the popularity of the 3-point shot, statistics related to

this particular feature of the game have gained importance in the last few years. All the parameters related to rebounds, mainly indicate the aggressiveness that the team has in winning second balls, this metric turns out to be very relevant because it allows teams to obtain points through second chances or to win possession from the opposing team.

3.2.1 Advanced Stats

The advanced box stats has more detailed information used by sports experts and analysts. They provide hidden information about the game. They need to be explained in more detail because they are metrics related to mathematical formulas and their impact is highly undervalued.

True Shooting Percentage

True Shooting Percentage (TS%) is an advanced statistic that measures the team's efficiency at shooting the ball. It is intended to more accurately calculate a team shooting than field goal percentage, free throw percentage, and three-point field goal percentage taken individually. Equation (3.1) shows the formula for this advanced metric.

$$\text{TS\%} = \frac{0.5 \times \text{PTS}}{\text{FGA} + 0.475 \times \text{FTA}} \tag{3.1}$$

Where:

- Points Scored (PTS);
- Field Goal Attempts (FGA);
- Free Throw Attempts (FTA).

When a team has consistently high True Shooting Percentages (60% or above), demonstrates their ability to efficiently score points on their games.

Effective Field Goal Percentage

Effective Field Goal Percentage (eFG%) is a measurement of how successful the team is in the field. This metric provides a more complete picture of the game situation than standard field goal percentages because three-point shots are given extra weight. In Equation (3.2) it is possible to observe a three-pointer is worth one and a half times as much as a made two-pointer. Since the NBA's 3-point situation has a considerably high weighting, this formula turns out to be very important because it pays special attention to the 3-point shot.

$$\text{eFG\%} = \frac{\text{FGM} + 1.5 \times \text{3PM}}{\text{FGA}} \tag{3.2}$$

Where:

- Field Goals Made (FGM);
- Three-Point Field Goals Made (3PM);
- Field Goal Attempts (FGA).

The Effective Field Goal Percentage can tell at a glance which team is having more success from the field. The team with the higher percentage is scoring more effectively from the field.

Three Point Attempt Rate

The *Three Point Attempt Rate* (3PAr) refers to the frequency with which a basketball team attempts three-point shots relative to their total field goal attempts. It is a metric used to assess a team's tendency to shoot three-pointers compared to two-pointers. The formula for this parameter is obtained from the Equation (3.3).

$$3P\% = \frac{3PA}{FGA} \times 100 \tag{3.3}$$

Where:

- 3PA: Three-Point Field Goal Attempts;
- FGA: Field Goal Attempts.

This metric is used to evaluate the team's shooting strategy and efficiency when it comes to 3-point shooting. Teams with a high percentage are teams that take the 3-point shot as one of their main strategies and achieve it with high efficiency.

Free Throw Attempt Rate

The *Free Throw Attempt Rate* (FTr) is a metric used in basketball to measure how often a team attempts free throws in comparison to their field goal attempts. Equation (3.4) can help determine a team's ability to draw fouls and get to the free throw line, which may reflect their level of aggressiveness and effectiveness in driving to the basket or operating in the low post.

$$FTAR = \frac{FTA}{FGA} \times 100 \tag{3.4}$$

Where:

- FTA: Free Throw Attempts;
- FGA: Field Goal Attempts.

Teams with high numbers in this stat often have a high-caliber attacking player that can only be stopped by a foul.

Offensive Rebound Percentage, Defensive Rebound Percentage and Total Rebound Percentage

Offensive Rebound Percentage (ORB%) (Equation (3.5)) measures the percentage of available offensive rebounds a team grabs in a game. It indicates the player's ability to secure missed shots by their team, leading to second-chance scoring opportunities.

$$\text{OREB\%} = \frac{\text{OREB}}{\text{OREB} + \text{DREB Opp}} \times 100 \tag{3.5}$$

Where:

- OREB: Offensive Rebounds;
- DREB Opp: Opponent Defensive Rebound.

Defensive Rebound Percentage (DRB%) (Equation (3.6)) is a metric that measures the percentage of available defensive rebounds that a team secures during a game. It indicates a team's ability to end the opposing team's possession by securing their missed shots.

$$\text{DREB\%} = \frac{\text{DREB}}{\text{DREB} + \text{OREB Opp}} \times 100 \tag{3.6}$$

Where:

- DREB: Defensive Rebounds;
- OREB Opp: Opponent Offensive Rebounds.

Total Rebound Percentage (TRB%) (Equation (3.7)) combines offensive and defensive rebounds to measure the percentage of total rebounds a team grabs while in the game, giving a comprehensive picture of their rebounding ability.

$$\text{TREB\%} = \frac{\text{TREB}}{\text{TREB} + \text{TREB Opp}} \times 100 \tag{3.7}$$

Where:

- TREB: Total Rebounds;
- TREB Opp: Opponent Total Rebounds.

Rebounding is an important category in the game, it influences the ability of teams to keep possession of the ball or recover it through missed shots by the opposition.

Assist Percentage

Assist Percentage (AST%) is a basketball statistic that measures the percentage of teammate field goals a player assists while they are on the court. Equation (3.8) provides insights into player's ability to create scoring opportunities for their teammates by passing the ball effectively.

$$\text{Assist Percentage } (\%) = \frac{\text{AST}}{\text{FGM}} \times 100 \tag{3.8}$$

Where:

- AST: Assists;
- FGM: Field Goals Made.

Teams with high Assist Percentage values often have primary playmakers who excel at passing and court vision.

Steal Percentage

Steal Percentage (STL%) is a basketball statistic that calculates the percentage of opponent possessions that a team steals while they are on the court. Equation (3.9) gives an insight into how well a team performs defensively by interrupting the opposing team's offense through interceptions or turnovers.

$$\text{Steal Percentage } (\%) = \frac{\text{STL}}{\text{Poss}} \times 100 \tag{3.9}$$

Where:

- STL: Steals;
- Poss: Opponent Possessions.

Steal Percentage is a valuable metric for evaluating a team's defensive ability to generate turnovers and disrupt the opponent's offense. Teams with high Steal Percentage often have players with defensive instincts, quick hands, and anticipation.

Block Percentage

Block Percentage (BLK%) is calculated using Equation (3.10). In basketball is a measure of a team's ability to disrupt opponents' shots near the basket. It tells us the percentage of opponent two-point field goal attempts that a team successfully blocks while they are on the court.

$$\text{Block Percentage } (\%) = \frac{\text{BLK}}{\text{2PA}} \times 100 \tag{3.10}$$

Where:

- BLK: Blocks;
- 2PA: Opponent 2-Point Field Goal Attempts.

Turnover Percentage

Turnover Percentage (TOV%) is a basketball statistic that quantifies the rate at which a team turns the ball over, relative to the number of possessions they are involved in. Equation (3.11) provides insight into a team's ball-handling and decision-making skills, as well as their ability to maintain possession and avoid costly turnovers.

$$\text{Turnover Percentage (\%)} = \frac{\text{TO}}{\text{Poss}} \times 100 \tag{3.11}$$

Where:

- TO: Turnovers;
- Poss: Possessions.

Usage Rating

Usage Rating (USG%) is one of the metrics that will not be used for this work. Equation (3.12) represents how this metric can be calculated. This statistic measures the percentage of a team's possessions that a player uses while on the court. Essentially, it quantifies how involved a player is in the offense by looking at how often they end a possession through a field goal attempt, free throw attempt, or turnover. As team statistics will be used, the Usage Percentage will be 100% for all the games of all the teams.

$$\text{Usage Rate} = \frac{\text{FGA} + \text{TO} + 0.44 \times \text{FTA} + \text{AST}}{\text{Team's Total Possessions}} \times 100 \tag{3.12}$$

Where:

- FGA: Field Goal Attempts;
- TO: Turnovers;
- FTA: Free Throw Attempts;
- AST: Assists;
- Team's Total Possessions: The total number of team possessions.

Offensive Rating and Defensive Rating

Offensive Rating (ORtg) and *Defensive Rating* (DRtg) are crucial basketball statistics that help evaluate a team's offensive and defensive performance. These ratings are calculated by considering the number of points scored and allowed per 100 possessions, providing a standardized measure to compare teams, regardless of the game's pace.

ORtg is a metric that measures a team's offensive efficiency by taking into account factors such as shooting accuracy, turnovers, and offensive rebounds, using Equation (3.13).

$$\text{Offensive Rating (ORtg)} = \frac{\text{PTS}}{\text{Poss}} \times 100 \tag{3.13}$$

Where:

- PTS: Points Scored;
- Poss: Possessions.

DRtg measures a team's defensive efficiency. Equation (3.14) reflects how effectively a team prevents its opponents from scoring, considering factors such as field goal percentage defense, turnovers forced, and defensive rebounds.

$$\text{Defensive Rating (DRtg)} = \frac{\text{Opp PTS}}{\text{Opp Poss}} \times 100 \tag{3.14}$$

Where:

- Opp PTS: Opponent Points Allowed;
- Opp Poss: Opponent Possessions.

3.3 Feature Engineering

Feature engineering is the process of transforming raw data into features that fit the machine learning model. It is a method of selecting, extracting, or transforming the most relevant features from the available data to make the machine learning model as accurate as possible [63].

The success of machine learning models depends heavily on the quality of the features used to train them. Figure 3.4 shows where the feature engineer fits into the machine learning process. This technique helps to highlight the most important patterns and the relationship between the data, which will make the model learn about the data more effectively.

It is important to explain the features that will be created based on feature engineering so that it is clear what they represent.

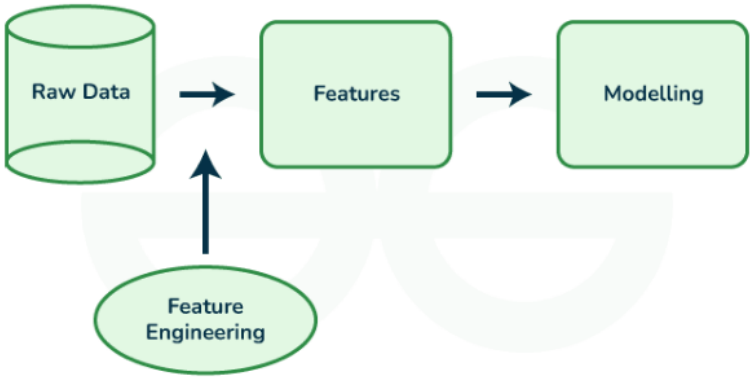


Figure 3.4: Feature Engineering in Machine Learning [63]

Table 3.2 indicates the features that will be introduced with a brief description. The meaning of these features, which will be introduced through Feature Engineering, will be described in more detail below.

Table 3.2: Description of the features introduced

Feature	**Description**
Last Games	Average statistics from the last few games
Rest Days Between Matches	Team time off between matches
Information About the Next Game	Opponent statistics
Elo Rating	Team rankings

3.3.1 Last Games

Basketball is a high-volume scoring sport with very short breaks between games compared to other sports [64]. This feature will explore how the teams arrive at the game to be predict. Since basketball is a sport of momentum, there will be times during the season when it seems that nothing goes right for the team or other times when everything goes perfectly. Victories are a very important moral aspect, but player injuries can sometimes have a big impact on form and give the model accurate information about the team's state.

In several of the works discussed, it's common to use only the last 10 games as the basis for this feature, an average of various statistics from the last 10 games. However, the number of games will be varied to see what impact they will have on the algorithm.

3.3.2 Rest Days Between Matches

The physical condition of teams is very important for sporting results. The NBA is widely criticized for the number of games each team plays per season, which is at least 82. Playing 82 games from the end of October to the beginning of April is quite a task. To give out an idea of a comparison with soccer, a team plays at least 30 games a season, starting in mid-August and ending in May, which gives an idea of how brutal the NBA schedule can sometimes be. There are also Back-to-backs, pairs of games that happen on sequential days, which are said to be grueling ordeals that put teams to the test [65]. Much more tired than normal, players often compete in different cities with not a day of rest in between. It is logical for teams to perform worse in the second game of back-to-backs compared to other games.

The machine learning algorithm will be provided with information about the interval of days between matches to understand the physical condition of the players before they take to the field.

3.3.3 Information About Next Game

Information about the game to predict will be transmitted to the algorithm. It would be impossible to understand this information from the last match's statistics alone. There is a lot of relevant information that can be sent, such as home-court advantage. This small detail has long been a topic of discussion for the NBA. Past statistics have shown that the home team will win around 60 % of the time [66]. This statistic makes it relevant for the algorithm to know which team is playing at home and which team is playing away.

In American sports, teams playing away games do not have the same allocation of tickets for their fans as in European sports. Consequently, the atmosphere in the stadium tends to be more favorable for home fans. Additionally, teams playing away often engage in a series of away games, leading to an extended period of time away from home and family.

Statistics on past matches will also be broadcast, especially on the next opponents. We need to be careful when transmitting this information to the Machine Learning model because we do not want to transmit information that only happens during the game, as that would be feeding the model with information that has not happened yet. Statistics such as who the next opponent is and their recent performance can be known before the game starts.

3.3.4 Elo Rating

Elo Rating is the most effective method for relativizing a team's current form and performance in the NBA. The Elo Ratings are calculated as follows: all teams start with a score of 1500 and over the course of the games, points are added or subtracted

based on the final result of each game. The exact formula for calculating the elo rating is defined in Equation (3.15).

$$\text{Elo Rating (Ri+1)} = k \times (S\_team - E\_team + Ri) \tag{3.15}$$

Where:

- k: constant representing the maximum adjustment possible in a single game;
- S_team: the actual score of the team participant (usually 1 for a win, 0 for a loss);
- E_team: represents the expected win probability of the team;
- Ri: former elo rating.

Equation (3.16) represents the probability of the team winning.

$$E_A = \frac{1}{1 + 10^{(Oppelo - Ri)/400}} \tag{3.16}$$

Where:

- Opp_elo: Opponent elo rating.

The Elo Rating is maintained from season to season. Teams tend to remain constant over time, and if a team is a title contender it tends to remain so for a while before starting to fall. So the formula for calculating the Elo Rating at the start of the season is given by Equation (3.17).

$$E_A = (R \times 0.75) + (0.25 \times 1505) \tag{3.17}$$

Where:

- R: final Elo of a team in one season.

As the NBA teams have a large number of games and a single game does not have a big impact during the season, k tends to have a lower value compared to sports that have fewer games during the season. The k value for this type of work usually ranges from 16 to 32 and is chosen based on the desired sensitivity for calculating the Elo Rating.

Chapter 4

Web Scraping and Data Organization

This chapter shows how the NBA and WNBA datasets were extracted using web scraping.

4.1 Version Control

To ensure that there were no compatibility problems, a new virtual environment was created. Table 4.1 shows the versions of the libraries used, Python and Jupyter Notebook. A study was carried out to ensure that the versions of these five components used were released on similar dates so that no incompatibilities were found during the web scraping process.

Table 4.1: Anaconda Virtual Environment Versions

Name	**Version**
Python	3.8.18
Jupyter Notebook	6.4.12
Playwright	1.25.1
Beautifulsoup4	4.12.2
Pandas	2.0.3

4.2 Web Scraping Basketball-Reference

This section explains how the dataset was obtained to training and testing the machine learning model by scraping the Basketball-Reference website.

Initially, it was important to understand what relevant information would be used to train the model. Figure 4.1 shows in the red rectangle what data is required for the dataset.

	Basic Box Score Stats																		
Starters	MP	FG	FGA	FG%	3P	3PA	3P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	+/-
Anthony Davis	40:37	16	24	.667	0	0		9	13	.692	5	15	20	5	0	4	4	2	41	
LeBron James	34:52	10	21	.476	0	1	.000	4	5	.800	1	10	11	4	2	0	6	5	24	
Cam Reddish	33:19	4	7	.571	1	3	.333	0	0		2	1	3	1	0	3	3	5	9	
D'Angelo Russell	31:53	6	15	.400	0	3	.000	1	2	.500	1	3	4	7	1	1	2	1	13	
Taurean Prince	18:59	2	3	.667	1	2	.500	1	1	1.000	0	3	3	2	0	0	1	4	6	
Team Totals	240	47	88	.534	2	13	.154	27	35	.771	12	43	55	25	5	10	18	25	123	

	Advanced Box Score Stats														
Starters	MP	TS%	eFG%	3PAr	FTr	ORB%	DRB%	TRB%	AST%	STL%	BLK%	TOV%	USG%	ORtg	DRtg	BPM
Anthony Davis	40:37	.690	.667	.000	.542	16.9	34.1	27.2	21.0	0.0	8.8	11.9	32.8	134	97	
LeBron James	34:52	.517	.476	.048	.238	3.9	26.5	17.4	16.6	2.6	0.0	20.5	33.1	94	100	
Cam Reddish	33:19	.643	.643	.429	.000	8.2	2.8	5.0	3.5	0.0	8.0	30.0	11.9	98	107	
D'Angelo Russell	31:53	.409	.400	.200	.133	4.3	8.7	6.9	27.8	1.4	2.8	11.2	22.2	100	106	
Taurean Prince	18:59	.872	.833	.667	.333	0.0	14.6	8.7	12.1	0.0	0.0	22.5	9.2	141	109	
Team Totals	240	.595	.545	.148	.398	34.3	82.7	63.2	53.2	4.7	18.5	14.8	100.0	116.6	103.3	

Figure 4.1: Desired Data for the Dataset

4.2.1 NBA

The first step in retrieving the data is to create a function that accesses the page from which data is to be retrieved. Playwright will be used for this function. This library will emulate a browser, so it is possible to access the page and extract the code through an HTML selector. This function will work asynchronously, allowing the program to perform multiple tasks without having to wait for one task to finish before starting the next [67]. In Web Scraping, there are often numerous HTTP requests and parsing responses, which can sometimes take too long. With this type of function, it is possible to speed up the process of obtaining data and extract it more efficiently. Figure 4.2 shows a time comparison between synchronous and asynchronous functions.

It is necessary to set several retries and a time interval between the scraping of the website. This chosen sleep interval is very important because sometimes scraping websites too quickly can result in a server ban, which causes the scraping to stop working because the website can no longer be accessed. Equation (4.1) shows how to get around this if it happens. Suppose the program has to try to retrieve the data from the page. In that case, it means that there has been an error such as a temporary ban or a connection error, so it is important to have a waiting period

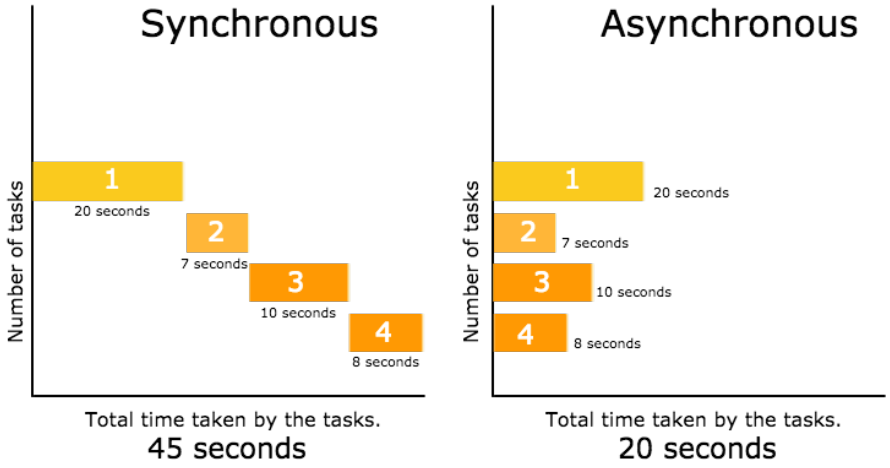


Figure 4.2: Synchronous versus Asynchronous [68]

between each attempt so that if this temporary ban occurs, it is possible to wait and re-enter the page when the suspension has been lifted.

$$\mathrm{WT} = S \times R \tag{4.1}$$

Where:

- WT: Waiting time;
- S: Sleeping Time;
- R: Retries;

The number set for the S parameter is 5 and for the R parameter, it is 3. This means that if the scraping fails initially, it will wait 5 seconds, if it fails again it will wait 10 seconds and finally on the third attempt it will wait 15 seconds before re-executing the scraping.

BeautifulSoup will be the library used to parse the site, converting the HTML into a navigable tree of Python objects. Figure 4.3 shows the structure of the website for a given time. If looking at the site individually by season, it is clear that the matches are organized by month and each month has its table with links to the statistics for each match. It will therefore be necessary to navigate through the months to obtain the tables of where the matches are. For example, when entering the website in the 2022-2023 season, there are boxes with the months of the season, clicking on these objects enters a link where the box score table contains the statistics of all the matches that took place in that month.

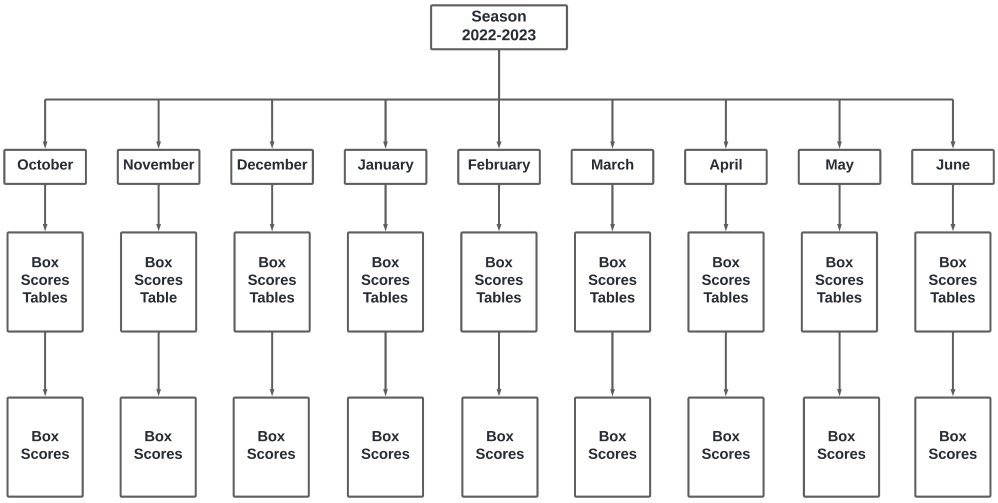


Figure 4.3: Season Page Structure

It is necessary to pass HTML parameters to the library. The aim is just to select what is needed to parse the boxscores. Figure 4.4 shows the parameters needed when inspecting the NBA Reference 2022-2023 season page. The "filter" parameter is a reference to the boxes with the months. Here it becomes necessary to make a loop so that after scraping the page for one month it moves on to another, and so on. As for the other parameter "all_schelude", it indicates the boxscore tables. Here are all the games from the respective months, each of which has a link to its statistics page; once all the tables have been obtained, the next step is to extract the individual game statistics.

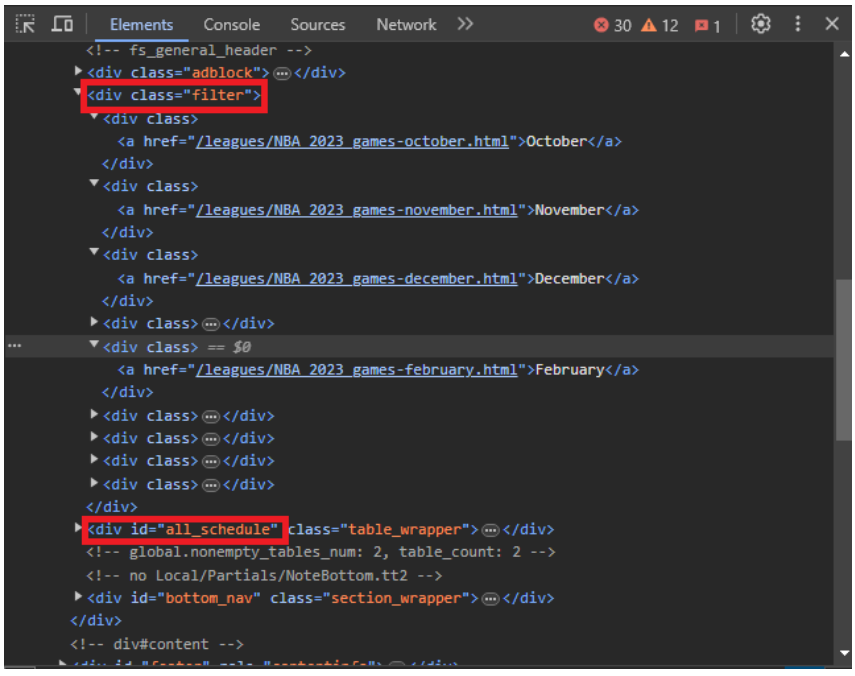


Figure 4.4: Season Page HTML Code

After obtaining the boxscore tables, individual game statistics can be scraped. However, it is necessary to understand what is contained in the table. In addition to game data, it provides information such as location, game videos, and so on, so it is necessary to analyze the HTML structure to understand where the link to the respective game is. Once this parameter has been found on the page, it is possible to obtain the match data from the NBA-Reference website. A lot of data will be extracted, so it will be necessary to organize the information first and then pass it on to the dataset.

Figure 4.5 shows the general flow of the algorithm. After accessing the site, the month will be selected and the statistics of each game will be taken from the box scores table, and so on until all the games in the table have been obtained. After that, another month is selected, and extracted the statistics from the box scores, and so on for all the seasons from 2015/2016 to 2023/2024.

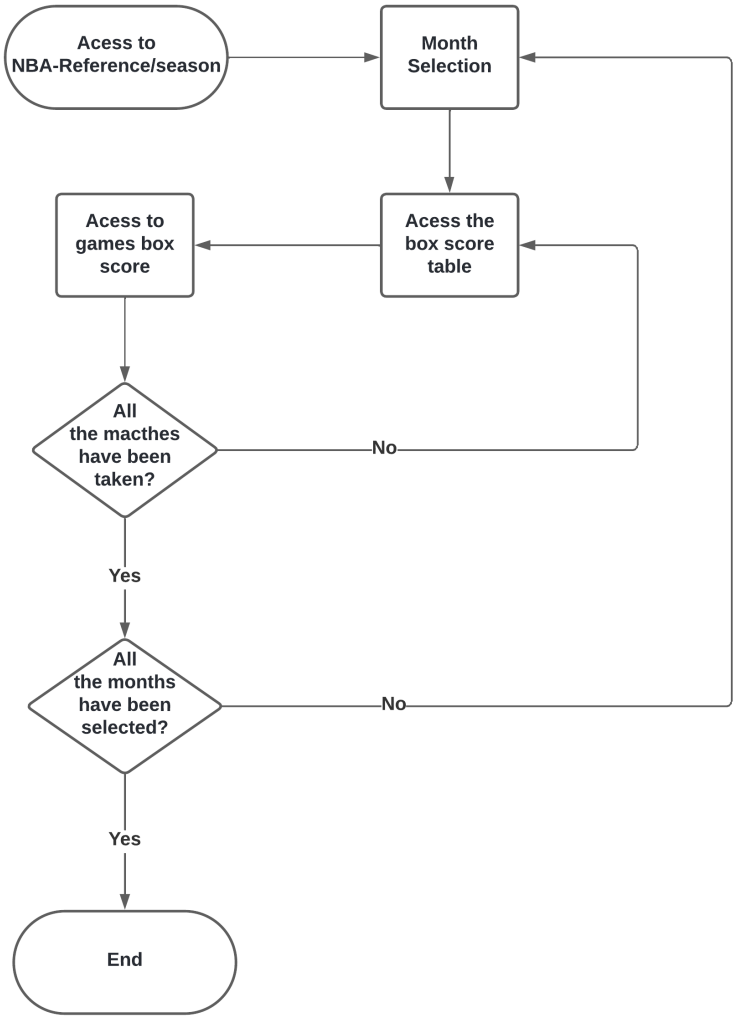


Figure 4.5: Web Scrapping Flowchart

Scraping is a very time-consuming process. On average, each season took about five hours. Once completed, the data needs to be processed to be converted into a dataset.

4.2.2 WNBA

The process is very similar to scraping the NBA. The site was the same, Playwright was used as the browser to access the website and BeautifulSoup was used to scrape the data. However, there are structural differences in the WNBA site. Because there are fewer games in this competition, the site thought it was not necessary to organize by month, so when entering the page for each season, access is immediately gained to the table of box scores with all the statistics for the respective games in that season. Once all the HTML elements for the table and the data for each game have been obtained, it is possible to start scraping the box scores.

Once all the HTML elements for the table and the data for each game have been obtained, it is possible to start scraping the box scores. Figure 4.6 shows the output after starting this process of scraping the table containing the box scores. For the WNBA, data was obtained from the 2016 season to the 2023 season. It is not possible to obtain data for the 2023-2024 season as it only starts in July. The Web Scraping process was also time-consuming. Although, because there were fewer games, the total time to obtain a season was on average one hour.

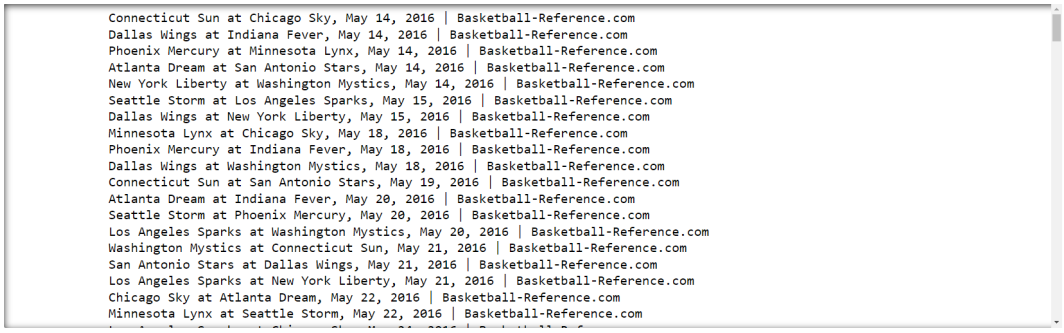


Figure 4.6: Output of Scraping

4.3 Data Organization

After obtaining the individual statistics pages for each game, they need to be organized and stored in a dataset to be used later by a machine learning model. Figure 4.7 shows the structure of the page obtained, which is quite different from the one on the Basketball-Reference website because when the HTML pages are scraped, they do not come with CSS customization.

Basic Box Score Stats

Starters	MP	FG	FGA	FG%	3P	3PA	3P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	+/-
Anthony Davis	40:37	16	24	.667	0	0		9	13	.692	5	15	20	5	0	4	4	2	41	
LeBron James	34:52	10	21	.476	0	1	.000	4	5	.800	1	10	11	4	2	0	6	5	24	
Cam Reddish	33:19	4	7	.571	1	3	.333	0	0		2	1	3	1	0	3	3	5	9	
D'Angelo Russell	31:53	6	15	.400	0	3	.000	1	2	.500	1	3	4	7	1	1	2	1	13	
Taurean Prince	18:59	2	3	.667	1	2	.500	1	1	1.000	0	3	3	2	0	0	1	4	6	
Reserves	**MP**	**FG**	**FGA**	**FG%**	**3P**	**3PA**	**3P%**	**FT**	**FTA**	**FT%**	**ORB**	**DRB**	**TRB**	**AST**	**STL**	**BLK**	**TOV**	**PF**	**PTS**	**+/-**
Austin Reaves	27:33	9	15	.600	0	3	.000	10	12	.833	0	2	2	3	0	0	0	1	28	
Max Christie	19:29	0	1	.000	0	1	.000	2	2	1.000	0	2	2	1	1	0	1	1	2	
Jarred Vanderbilt	15:46	0	1	.000	0	0		0	0		0	4	4	1	1	2	0	1	0	
Rui Hachimura	8:37	0	1	.000	0	0		0	0		2	0	2	0	0	0	0	3	0	
Jaxson Hayes	7:23	0	0		0	0		0	0		1	3	4	1	0	0	1	1	0	
Jalen Hood-Schifino	0:46	0	0		0	0		0	0		0	0	0	0	0	0	0	1	0	
Maxwell Lewis	0:46	0	0		0	0		0	0		0	0	0	0	0	0	0	0	0	
Christian Wood	Did Not Play																			
Team Totals	240	47	88	.534	2	13	.154	27	35	.771	12	43	55	25	5	10	18	25	123	

Advanced Box Score Stats

Starters	MP	TS%	eFG%	3PAr	FTr	ORB%	DRB%	TRB%	AST%	STL%	BLK%	TOV%	USG%	ORtg	DRtg	BPM
Anthony Davis	40:37	.690	.667	.000	.542	16.9	34.1	27.2	21.0	0.0	8.8	11.9	32.8	134	97	
LeBron James	34:52	.517	.476	.048	.238	3.9	26.5	17.4	16.6	2.6	0.0	20.5	33.1	94	100	
Cam Reddish	33:19	.643	.643	.429	.000	8.2	2.8	5.0	3.5	0.0	8.0	30.0	11.9	98	107	
D'Angelo Russell	31:53	.409	.400	.200	.133	4.3	8.7	6.9	27.8	1.4	2.8	11.2	22.2	100	106	
Taurean Prince	18:59	.872	.833	.667	.333	0.0	14.6	8.7	12.1	0.0	0.0	22.5	9.2	141	109	
Reserves	**MP**	**TS%**	**eFG%**	**3PAr**	**FTr**	**ORB%**	**DRB%**	**TRB%**	**AST%**	**STL%**	**BLK%**	**TOV%**	**USG%**	**ORtg**	**DRtg**	**BPM**
Austin Reaves	27:33	.690	.600	.200	.800	0.0	6.7	4.0	16.7	0.0	0.0	0.0	29.1	155	112	
Max Christie	19:29	.532	.000	1.000	2.000	0.0	9.5	5.7	5.2	2.3	0.0	34.7	5.8	91	105	
Jarred Vanderbilt	15:46	.000	.000	.000	.000	0.0	23.4	14.0	6.5	2.9	11.3	0.0	2.5	59	92	
Rui Hachimura	8:37	.000	.000	.000	.000	31.8	0.0	12.8	0.0	0.0	0.0	0.0	4.6	104	113	
Jaxson Hayes	7:23					18.6	37.5	29.9	13.8	0.0	0.0	100.0	5.4	75	102	
Jalen Hood-Schifino	0:46					0.0	0.0	0.0	0.0	0.0	0.0		0.0	0	112	
Maxwell Lewis	0:46					0.0	0.0	0.0	0.0	0.0	0.0		0.0	0	114	
Christian Wood	Did Not Play															
Team Totals	240	.595	.545	.148	.398	34.3	82.7	63.2	53.2	4.7	18.5	14.8	100.0	116.6	103.3	

Figure 4.7: Box Score Obtained

The main objective is to extract the totals from the table containing the basic statistics of the teams and the table containing the more advanced statistics. However, the page provides other relevant useful information such as which team played at home, the season in which it took place and the date. This last piece of information will be essential to be able to organize the dataset chronologically later on. If the aim is to make predictions, it is important to maintain a chronological order and use past matches to predict future matches.

The statistics provided indicate a lot of parameters, but it is only possible to see who won based on the points scored and conceded. It is also necessary to provide this information to the dataset so that the model can identify winning and losing patterns when it is trained. Duplicate columns that are in both tables will only be scraped from the first one and statistics such as "bpm" that only provide individual information will be removed.

Basketball is a team sport in which only five players can be on the court at the same time from each team. Due to this limited number of players, the impact that one player can have on the game ends up being greater than in a sport like soccer. where you have eleven players from each team. It, therefore, becomes necessary to obtain the numbers of the player who stands out from each team in each statistic. These data will have a similar name to the other statistics but with the addition

of "_max" in the name. For example, the points parameter has the name "Points", which indicates the points that the team has scored, but "Points_max" indicates the number of points that the player with the most points from that team has scored. In addition to being based on team statistics, an extra factor is added in case a team has a key player who makes a difference in the game but translates mathematically into the statistics (which is quite common), which the model will be able to identify. Also, sometimes there are injuries, and this parameter going down or up can indicate the absence or return from injury of a key player.

It is also important to include all the opponent's parameters in the match statistics. When trying to predict whether team x will win or not, the model also needs to know everything about the opponent. In this way, the model can have information not only about the team we want to predict but also about the other team, thus greatly reducing the probability of predicting that both teams will win or both teams will lose.

The libraries used for the final data extraction and organization were BeautifulSoup and Pandas. BeautifulSoup's HTML selector allows to select the parts of the file to extract using HTMlL elements, which is very important when the goal is extract parts of a page. Pandas is responsible for organizing the data. Obtaining statistics such as the individual maximum of each team or converting the data so that they all have the same scale will be very important for the machine learning model. This library is essential because it allows the data obtained from the HTML files to be processed and included in the dataset. Figure 4.8 shows the dataset that was obtained after extracting and organizing the relevant data.

	mp	mp.1	fg	fga	fg%	3p	3pa	3p%	ft	fta	...	tov%_max_opp	usg%_max_opp	ortg_max_opp	drtg_max_opp	team_opp	total_opp
0	240.0	240.0	39.0	81.0	0.481	6.0	20.0	0.300	14.0	18.0	...	22.8	29.0	178.0	111.0	DAL	95
1	240.0	240.0	36.0	100.0	0.360	7.0	31.0	0.226	16.0	19.0	...	50.0	32.6	152.0	111.0	ATL	98
2	240.0	240.0	37.0	85.0	0.435	8.0	19.0	0.421	17.0	23.0	...	20.0	30.9	148.0	116.0	SAS	107
3	240.0	240.0	41.0	89.0	0.461	8.0	21.0	0.381	17.0	19.0	...	28.6	30.9	138.0	118.0	MIN	99
4	240.0	240.0	27.0	86.0	0.314	6.0	26.0	0.231	15.0	20.0	...	16.8	30.9	157.0	90.0	MEM	92
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
22357	240.0	240.0	49.0	92.0	0.533	18.0	40.0	0.450	13.0	16.0	...	50.0	37.8	223.0	134.0	MIL	94
22358	240.0	240.0	37.0	80.0	0.463	10.0	32.0	0.313	19.0	21.0	...	50.0	28.1	215.0	124.0	SAS	101
22359	240.0	240.0	39.0	82.0	0.476	11.0	36.0	0.306	12.0	14.0	...	33.3	29.4	165.0	122.0	HOU	103
22360	240.0	240.0	43.0	92.0	0.467	20.0	49.0	0.408	17.0	19.0	...	23.3	32.6	267.0	142.0	UTA	107
22361	240.0	240.0	41.0	86.0	0.477	15.0	32.0	0.469	10.0	11.0	...	14.7	41.1	180.0	128.0	BOS	123

22362 rows × 150 columns

Figure 4.8: NBA Dataset

There are structural differences between the NBA and WNBA data pages, such as

IDs of certain HTML elements or different aspects of the HTML page. It is important to analyze the HTML page carefully and rigorously to obtain the parameters as efficiently as possible. Figure 4.9 shows the scheme of the data processing program.

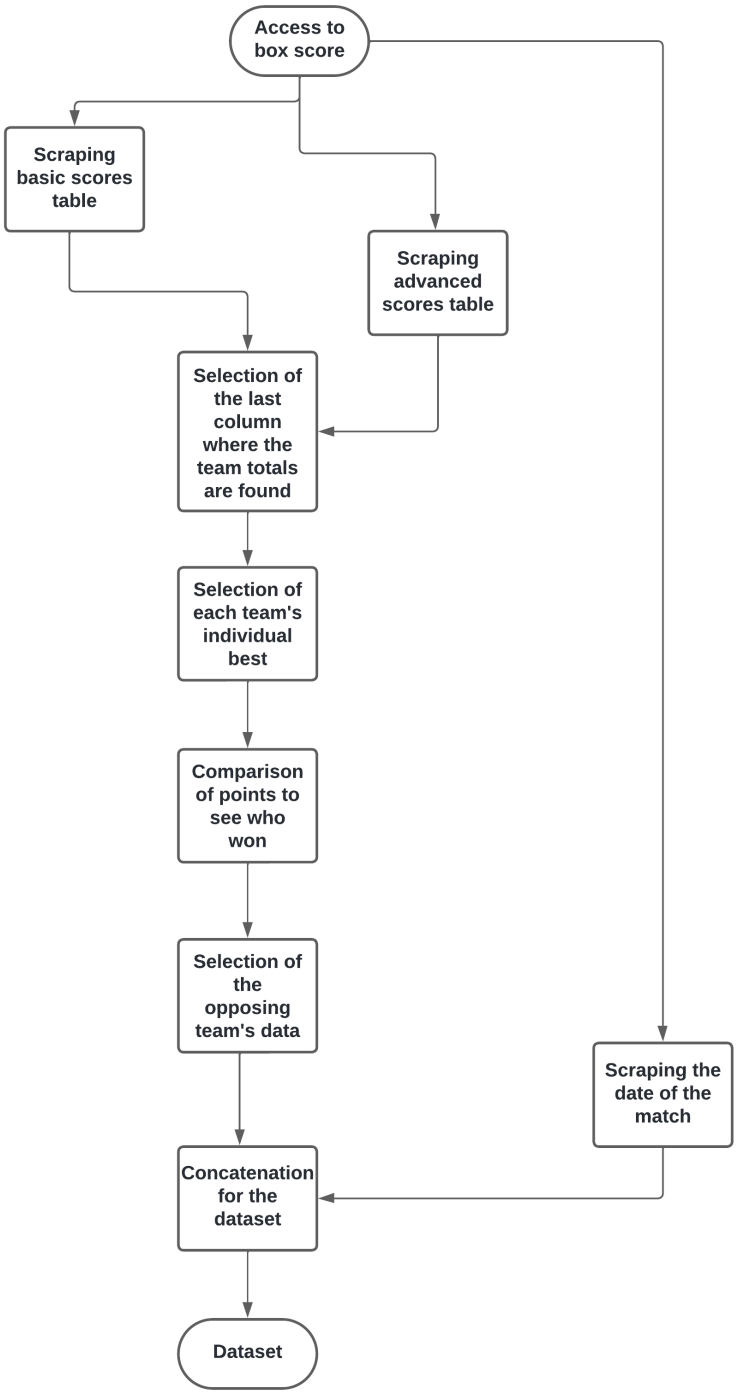


Figure 4.9: Web Scrapping Data Scheme

After accessing the respective box score of the match, the box score will be

scraped with the basic and advanced statistics and also the date on which the match took place. Then, using the Pandas library, select the last line of each table where the team statistics are. It will also analyze the two tables and select the individual maximum of each statistic, to obtain the maximum in each statistic of each game. Then the opponent's statistics are obtained and the points between the teams are compared to see who won. Once all these parameters have been obtained, everything is combined to form the data set.

4.3.1 Dataset

The analysis of the data obtained is important. Understanding how many games are there and how they are divided in each dataset can be very important later on when analyzing the data.

Analyzing NBA

Figure 4.10 shows the distribution of games in the NBA dataset by season. There are more than 2500 games per season except in three of the seasons. The 2019/2020 and 2020/2021 seasons were years affected by COVID-19. The NBA has reduced the number of games in these two years to protect players from contracting this terrible virus that has affected everyone season. As for the 2023/204 season, is still underway and the data obtained was for March. This season will be used exclusively for testing.

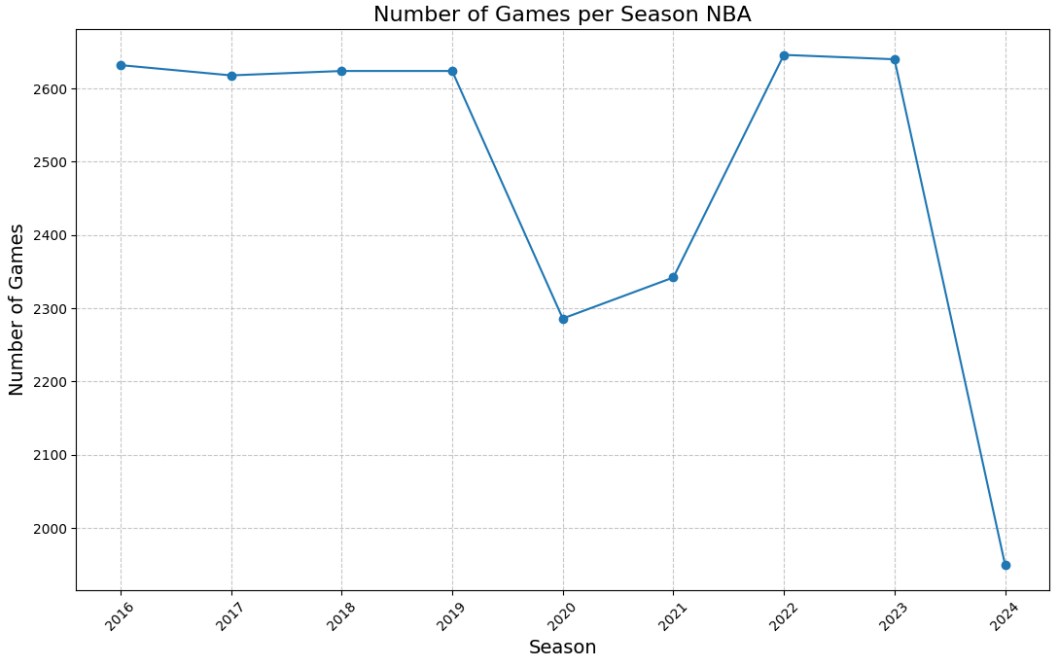


Figure 4.10: NBA Games Distribution

Analyzing WNBA

The variation of games in the WNBA dataset is shown in Figure 4.11. You can see that there are more than 400 games per season, but in one of the seasons, 2020, there are substantially fewer games. As happened in the NBA, COVID-19 has also affected this league and preventive measures have been taken to reduce the number of games. Note that in the 2022 and 2023 seasons there was a significant increase in games due to the Commissionary Cup, which introduced a tournament within the league that resulted in a greater number of games in those seasons. The 2024 season has not started yet, so there are no data available to qualify for the games.

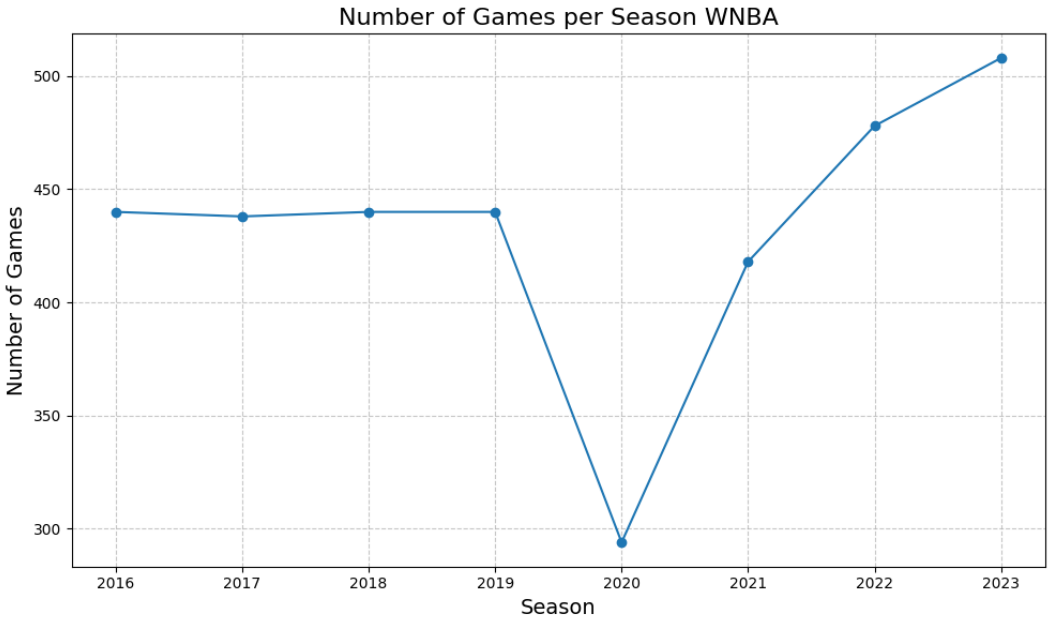


Figure 4.11: WNBA Games

NBA vs WNBA

There are significant differences in the number of matches between the two datasets. Figure 4.12 shows the average number of games played per season in the two datasets. The WMNA has only 12 teams while the NBA has 32 and in the regular season, there are only 40 games, while in the NBA there are 82.

The playoff model is also different between the two leagues. The NBA has implemented the best-of-7 in all its rounds, while the WNBA until the 2021 season used the best-of-3, and only in the 2022 and 2023 seasons is a hybrid model implemented, in which the first round of the playoffs is the best-of-3 and the rest of the best-of-5, which means that there are more games to be analyzed.

The fact that the 2024 season is also an important factor because it allows the NBA dataset to have one more season, although it is not complete, it has a relevant number of games.

Both competitions are very competitive, but the model in which the NBA is run means that are more data to be obtained than for the WNBA.

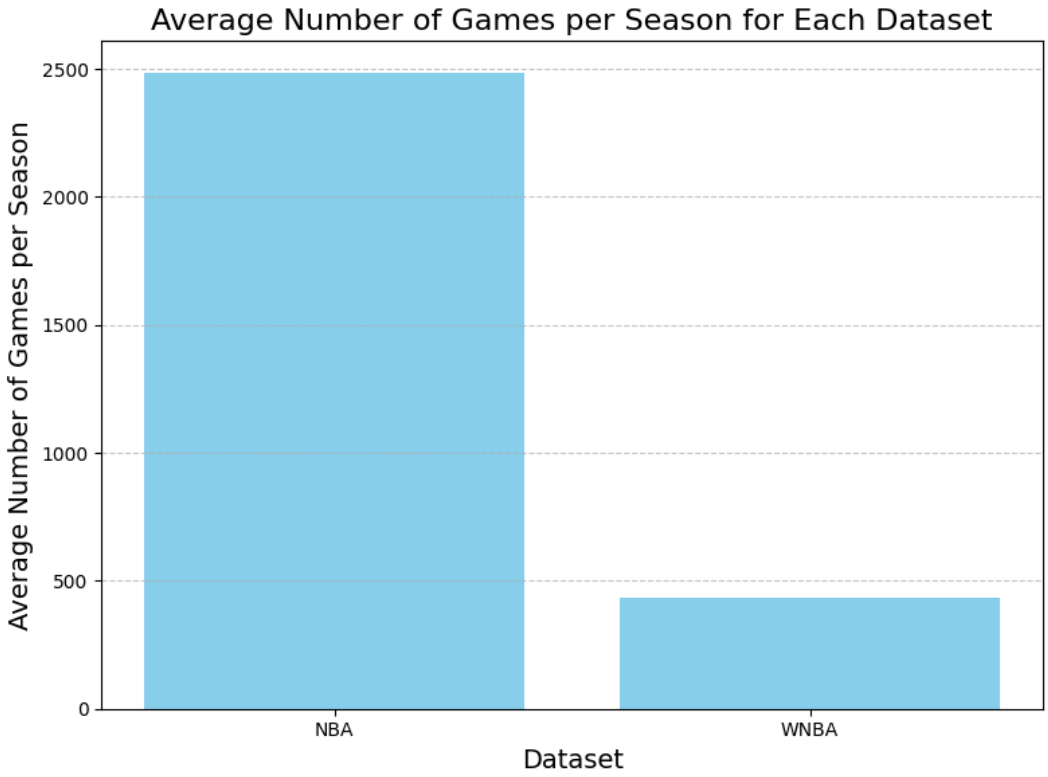


Figure 4.12: Comparison of games per season between NBA and WNBA dataset

Chapter 5

Machine Learning

This chapter will describe the preparation of the datasets obtained through web scraping, the creation of the features through Feature Engineering, and how the Machine Learning process will be carried out.

5.1 Data Preparation

Initially, when the dataset is opened, it is not organized in chronologically. It is essential for the machine learning model and for creating new features through feature engineering that the model contains a temporal order. When making predictions, it is necessary to make sure that there is information from the past to predict the future. It would be cheating to use information from future games to predict past games, so it is necessary to maintain this condition of temporal order to validate the results. Figure 5.1 shows the dataset organized chronologically.

Repeated columns will also be eliminated as they only confuse the machine learning model. Columns like "mp1" and "mp_opp" provide the same information as the "mp" column: the minutes played. The mp1 column is a repetition of the mp column since the scaped basic score and advanced score tables both contained information on minutes played. As for "mp_opp", it is information about the minutes played by the opposing team, which is the same as the minutes played by the other team, so it does not need to be passed on to the machine learning model.

fg	fga	fg%	3p	3pa	3p%	ft	fta	...	tov%_max_opp	usg%_max_opp	ortg_max_opp	drtg_max_opp	team_opp	total_opp	home_opp	season	date	won
41.0	96.0	0.427	9.0	30.0	0.300	20.0	22.0	...	37.5	38.9	201.0	120.0	NOP	95	0	2016	2015-10-27	True
37.0	87.0	0.425	7.0	19.0	0.368	16.0	23.0	...	30.4	29.0	138.0	105.0	CLE	95	0	2016	2015-10-27	True
38.0	94.0	0.404	9.0	29.0	0.310	10.0	17.0	...	53.2	34.6	162.0	104.0	CHI	97	1	2016	2015-10-27	False
37.0	96.0	0.385	12.0	29.0	0.414	20.0	26.0	...	57.1	33.8	258.0	121.0	ATL	94	1	2016	2015-10-27	True
37.0	82.0	0.451	8.0	27.0	0.296	12.0	15.0	...	33.3	23.6	132.0	104.0	DET	106	0	2016	2015-10-27	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34.0	72.0	0.472	11.0	30.0	0.367	21.0	25.0	...	20.0	41.1	222.0	122.0	MIN	118	0	2024	2024-03-12	False
48.0	94.0	0.511	12.0	33.0	0.364	10.0	14.0	...	50.0	34.5	149.0	142.0	LAC	100	1	2024	2024-03-12	True
43.0	92.0	0.467	20.0	49.0	0.408	17.0	19.0	...	23.3	32.6	267.0	142.0	UTA	107	1	2024	2024-03-12	True
40.0	79.0	0.506	14.0	40.0	0.350	12.0	15.0	...	20.0	32.0	162.0	132.0	PHI	79	0	2024	2024-03-12	True
41.0	86.0	0.477	15.0	32.0	0.469	10.0	11.0	...	14.7	41.1	180.0	128.0	BOS	123	0	2024	2024-03-12	False

nns

Figure 5.1: Dataset Organized by Game Date

The machine learning algorithm aim to predict the result of the next game based on the information in the previous game. Defining a variable that tells the team's result in the next match is necessary, as this will be the "target" variable. This variable will be the model's training target, that is, the machine learning model will be trained to predict the result of the target variable. When analyzing a team's performance in a season, it is important to acknowledge that there may be periods of both highs and lows. To make an accurate prediction about the outcome of the next game, it is recommended to take into account the statistics from the previous game. This approach can yield a more realistic representation of the current state of the team. By utilizing this approach, a balanced view of the team's performance can be achieved, which can lead to a more insightful prediction of their future performance.

The dataset needs to be divided into teams to create the target column. To be sure that the target variable will apply to that team and not to different teams from different games, each team will have its data frame, which will create a copy of the won column and move it up one position to get the result of the next game. The machine learning model can only interpret numerical values, so converting the trues and falses to 1 and 0 is necessary. There will be null values since at the end of each team's data frame there will be no result for the next game because the next game does not exist. However, it is necessary to convert this value to a number which will be 2 to be interpreted by the machine learning model. Figure 5.2 shows the data frame of the Philadelphia 76ers ("PHI") team with the target column where it can be seen that the column contains the result of the next game. This same column will be the target of the machine learning prediction process.

%	3p	3pa	3p%	ft	fta	ft%	...	total_opp	team_elo	opp_elo	days_since_last_game	days_since_last_game_opp	home_opp	season	date	won	target
10	7.0	22.0	0.318	20.0	23.0	0.870	...	112	1490.000000	1510.000000	0	0	1	2016	2015-10-28	False	0
02	6.0	15.0	0.400	27.0	37.0	0.730	...	99	1480.287744	1500.287744	2	2	0	2016	2015-10-30	False	0
47	7.0	24.0	0.292	17.0	23.0	0.739	...	107	1470.854500	1519.983681	3	3	0	2016	2015-11-02	False	0
20	6.0	28.0	0.214	13.0	15.0	0.867	...	91	1461.691413	1492.217889	2	2	1	2016	2015-11-04	False	0
19	10.0	23.0	0.435	10.0	15.0	0.667	...	108	1452.789576	1538.564691	2	2	1	2016	2015-11-06	False	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
38	12.0	28.0	0.429	15.0	19.0	0.789	...	112	1522.096181	1432.285163	2	1	1	2024	2024-03-05	False	0
32	13.0	35.0	0.371	18.0	21.0	0.857	...	115	1511.461058	1422.363225	1	2	0	2024	2024-03-06	False	0
38	9.0	37.0	0.243	16.0	20.0	0.800	...	103	1501.131302	1574.169374	2	3	0	2024	2024-03-08	False	1
38	9.0	30.0	0.300	8.0	14.0	0.571	...	73	1511.098741	1537.697905	2	2	1	2024	2024-03-10	True	0
75	8.0	33.0	0.242	11.0	16.0	0.688	...	106	1500.779402	1546.617110	2	2	1	2024	2024-03-12	False	2

Figure 5.2: PHI Data Frame with Target Variable

In NBA and WNBA games there is always a winner. If there are draws at the end of regulation time, the game goes to overtime, which is four minutes per part. If after four minutes no winner has been found, the game goes into another overtime and so on until a team wins. Figure 5.3 shows that the dataset contains as many wins as losses which proves that there are no missing matches in the chosen time frame.

```
[16]: df["won"].value_counts()

[16]: False    11181
      True     11181
      Name: won, dtype: int64
```

Figure 5.3: Number of Wins and Losses in the Dataset

It will also be necessary to remove the null values. Since ML models do not work well with null values. The Pandas library was used to check the columns with null values.

Figure 5.4 shows columns with null values. These statistics refer to the maximum individual minutes played by both the team and the opponent. This statistic is not relevant and can therefore be eliminated. As for +/-, it is a statistic used individually. It represents the difference in points since the player has been on the pitch. As the focus is on team stats, it is not necessary either. So these six columns will be removed. The columns removed from the WNBA dataset are similar to those of the NBA, however, the +/- is not counted in women's basketball so it is not necessary to remove it.

```
[22]: nulls[nulls > 0]

[22]: +/-             22362
      mp_max          22362
      mp_max.1        22362
      +/-_opp         22362
      mp_max_opp      22362
      mp_max_opp.1    22362
      dtype: int64
```

(a) Nulls Values NBA

```
[20]: nulls[nulls > 0]

[20]: mp_max          3456
      mp_max.1        3456
      mp_max_opp      3456
      mp_max_opp.1    3456
      dtype: int64
```

(b) Nuls Values WNBA

Figure 5.4: Null Values in the Two Datasets

5.2 Feature Engineering

The following section delves into the creation of features through Feature Engineering. The primary goal of these features is to assist the machine learning model in visualizing statistics that may not be fully comprehensible using the statistics obtained solely from the current dataset.

5.2.1 Elo Rating

This feature aims to add a ranking of the teams to the dataset. Initially, in the dataset, all the teams start with an elo of 1500, then depending on whether they lose or win, the formulas of the Elo Rating described in Chapter 3 are used. The k chosen for calculating the Elo Rating will be 20. The NBA often features high-scoring games, and teams experience both good and challenging periods throughout the season. The k for Elo Rating for the NBA ranges from 16-24, so 20 was chosen to balance out the stability and responsiveness of the system. This allows for gradual changes in team ratings over time, while still being sensitive enough to capture major changes. Figure 5.5 shows the teams with the elo before and after the match.

To check that the elo was being upgraded correctly, a column called team elo before has been added. This column will be used to check the elo before the game. This way it will be possible to be sure that the Elo Rating is being carried out correctly.

To calculate the teams' Elo Rating, it is necessary to group the dataset by team and by date. The aim is to iterate through each team's dataset to obtain the wins and losses of each specific team. Also the division of each team's dataframe by date to guarantee a correct chronological order. After obtaining the correct data frame for each team, the Elo Rating is calculated over the dataset. The elo will be calculated using the points, if the team for which the elo is to be calculated has more points than the opposing team, 1 will be awarded to Equation (3.15), otherwise, 0 will be awarded. Figure 5.6 shows the variation of the elo over the dataset for the Boston Celtics team. It is possible to see the elo increasing with wins and decreasing with losses.

	mp	mp.1	fg	fga	fg%	3p	3pa	3p%	ft	fta	...	drtg_max_opp	team_opp	total_opp	home_opp	season	date	won	team_elo_before	team_elo
0	240.0	240.0	37.0	82.0	0.451	8.0	27.0	0.296	12.0	15.0	...	104.0	DET	106	0	2016	2015-10-27	False	1500.000000	1490.000000
1	240.0	240.0	35.0	83.0	0.422	6.0	18.0	0.333	19.0	27.0	...	104.0	GSW	111	1	2016	2015-10-27	False	1500.000000	1490.000000
2	240.0	240.0	41.0	96.0	0.427	9.0	30.0	0.300	20.0	22.0	...	120.0	NOP	95	0	2016	2015-10-27	True	1500.000000	1510.000000
3	240.0	240.0	38.0	94.0	0.404	9.0	29.0	0.310	10.0	17.0	...	104.0	CHI	97	1	2016	2015-10-27	False	1500.000000	1490.000000
4	240.0	240.0	37.0	96.0	0.385	12.0	29.0	0.414	20.0	26.0	...	121.0	ATL	94	1	2016	2015-10-27	True	1500.000000	1510.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57	240.0	240.0	34.0	72.0	0.472	11.0	30.0	0.367	21.0	25.0	...	122.0	MIN	118	0	2024	2024-03-12	False	1591.670925	1579.091993
58	240.0	240.0	30.0	82.0	0.366	7.0	39.0	0.179	27.0	35.0	...	104.0	SAC	129	1	2024	2024-03-12	False	1572.572129	1560.513192
59	240.0	240.0	48.0	89.0	0.539	9.0	28.0	0.321	16.0	19.0	...	124.0	OKC	111	1	2024	2024-03-12	True	1524.750470	1534.039297
60	240.0	240.0	43.0	99.0	0.434	14.0	37.0	0.378	11.0	15.0	...	119.0	IND	121	0	2024	2024-03-12	False	1629.642592	1616.075228
61	240.0	240.0	40.0	79.0	0.506	14.0	40.0	0.350	12.0	15.0	...	132.0	PHI	79	0	2024	2024-03-12	True	1531.628017	1540.720196

2 rows × 153 columns

Figure 5.5: Elo Rating Initially

To complement the information for the dataset, information about the opponent was also added. To do this, the name of the opposing team and the date of the match are checked, and, based on this information, the Elo Rating of the opposing team is looked up in the dataset, and the elo before is also added beforehand to check that it is being calculated correctly.

```
all_teams_df[all_teams_df["team"] == "BOS"]
```

fg	fga	fg%	3p	3pa	3p%	ft	fta	...	ortg_max_opp	drtg_max_opp	team_opp	total_opp	home_opp	season	date	won	team_elo_before	team_elo
39.0	85.0	0.459	8.0	24.0	0.333	26.0	27.0	...	120.0	123.0	PHI	95	0	2016	2015-10-28	True	1500.000000	1510.000000
32.0	85.0	0.376	7.0	26.0	0.269	32.0	41.0	...	231.0	101.0	TOR	113	0	2016	2015-10-30	False	1510.000000	1499.712256
35.0	98.0	0.357	6.0	29.0	0.207	11.0	14.0	...	126.0	101.0	SAS	95	0	2016	2015-11-01	False	1499.712256	1489.720538
35.0	83.0	0.422	10.0	27.0	0.370	18.0	22.0	...	145.0	110.0	IND	100	1	2016	2015-11-04	False	1489.720538	1480.016319
44.0	97.0	0.454	12.0	30.0	0.400	18.0	23.0	...	171.0	119.0	WAS	98	0	2016	2015-11-06	True	1480.016319	1490.590862
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39.0	93.0	0.419	15.0	38.0	0.395	11.0	12.0	...	175.0	122.0	CLE	105	1	2024	2024-03-05	False	1690.364813	1675.375261
41.0	91.0	0.451	11.0	38.0	0.289	16.0	25.0	...	227.0	119.0	DEN	115	1	2024	2024-03-07	False	1675.375261	1660.716654
46.0	94.0	0.489	15.0	39.0	0.385	10.0	13.0	...	180.0	130.0	PHO	107	1	2024	2024-03-09	True	1660.716654	1666.394810
49.0	100.0	0.490	15.0	40.0	0.375	8.0	12.0	...	207.0	142.0	POR	99	1	2024	2024-03-11	True	1666.394810	1671.941005
43.0	92.0	0.467	20.0	49.0	0.408	17.0	19.0	...	267.0	142.0	UTA	107	1	2024	2024-03-12	True	1671.941005	1677.360148

ns

Figure 5.6: Elo Rating from the BOS Team

The Elo Rating is not reset season after season, there is a continuity to this feature. To check that the formula was being applied correctly, the following command from Figure 5.7 was executed. It can be seen that the team's link in the last game

of the 2016 season was 1239. However, at the start of the 2017 season, that same team's elo will start with a value of 1306 due to the application of Equation (3.17).

```
mp                        240.0
mp.1                      240.0
fg                         37.0
fga                        81.0
fg%                       0.457
                            ...
won                       False
team_elo_before    1243.549512
team_elo           1239.829614
opp_elo_before     1486.134053
opp_elo            1496.532936
Name: 2457, Length: 154, dtype: object
```

(a) Elo Final Day of Season 2016

```
mp                        240.0
mp.1                      240.0
fg                         36.0
fga                        84.0
fg%                       0.429
                            ...
won                       False
team_elo_before    1306.122211
team_elo           1301.187318
opp_elo_before     1560.086477
opp_elo            1568.374087
Name: 2656, Length: 154, dtype: object
```

(b) Elo First Day of the Season 2017

Figure 5.7: Elo Rating Comparison between the Last Day of the 2016 Season and the First Day of the 2017 Season

To observe the impact of the Elo Rating, the following study was carried out to make a comparison with reality. Figure 5.8 shows the respective Elo Rating of three teams chosen at random from the dataset. Analyzing the performances of the three teams separately, what is clear is that TOR initially shows quite high elo values in the dataset, which is completely in line with reality, this team even won the 2019 title, however, their period with the most elo ended up being when they had a record of 53-19 and ended up reaching the conference finals, which was a very positive season. However, from 2021 onwards, there will be a sharp decline in the team's record. TOR, after having an excellent season in 2020, showed a huge drop in form, finishing in twelfth place. However, in the 2021 season, they performed more reasonably and ended up in sixth place in the conference, which justifies the increase in the elo at this stage. From this point onwards there is a clear drop in the elo, which is justified by the fact that TOR has one of the worst records in the league at the moment.

For the Boston team, there is a noticeable consistency in their performance, reflected in their track record of reaching several conference finals and an NBA final. However, there are two peaks, one negative and the other positive, which should be analyzed. The negative peak was when BOS finished in seventh place in the conference, but despite this, they managed to reach the conference finals, which justifies that small rise at the end of the season. The second peak and this one is already positive, is in the 2024 season when this same team has the best record in the league and is one of the main contenders for the title.

Finally, to analyze the NOP team, their starting elo is not very famous, even having a lot of difficulties in reaching 1500. However, in the 2018 season it presented a peak because they won a series of games in a row, since there was a fall practically in the same season. In 2020 they had one of the worst records in the league, finishing

bottom of their respective conference, which justifies the negative spike in 2020. As for the negative peak in 2021, it is again due to a bad season by this team with a record of 31-41. In 2024 it is already possible to see a paradigm shift for this team, which is justified by their fourth-place finish in the Western Conference.

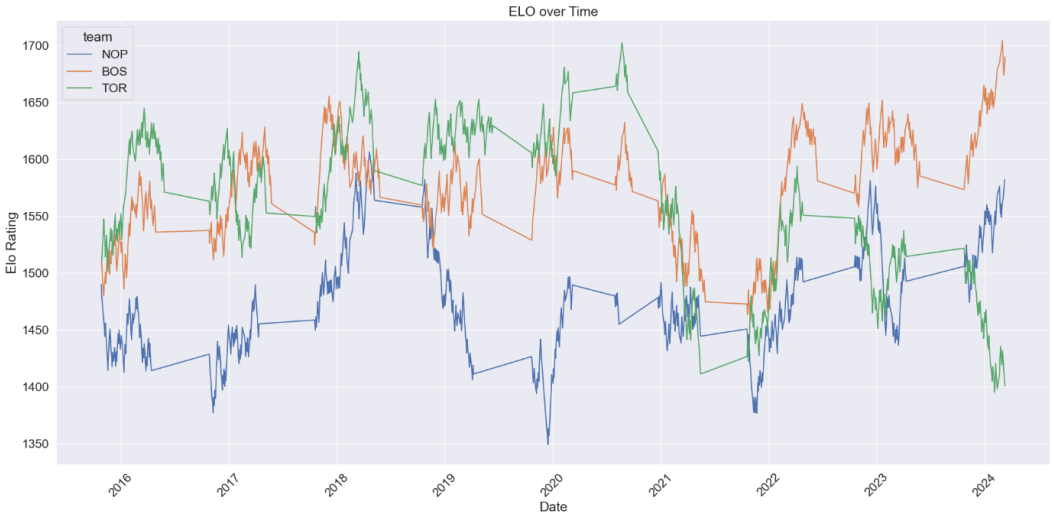


Figure 5.8: Visualization of the Elo Rating of 3 Teams

Figure 5.9 shows the final dataset after adding the Elo Rating. It can be seen that two more columns have been added to the original dataset.

	mp	mp.1	fg	fga	fg%	3p	3pa	3p%	ft	fta	...	ortg_max_opp	drtg_max_opp	team_opp	total_opp	team_elo	opp_elo	home_opp	seaso
0	240.0	240.0	37.0	82.0	0.451	8.0	27.0	0.296	12.0	15.0	...	132.0	104.0	DET	106	1490.000000	1510.000000	0	201
1	240.0	240.0	35.0	83.0	0.422	6.0	18.0	0.333	19.0	27.0	...	206.0	104.0	GSW	111	1490.000000	1510.000000	1	201
2	240.0	240.0	41.0	96.0	0.427	9.0	30.0	0.300	20.0	22.0	...	201.0	120.0	NOP	95	1510.000000	1490.000000	0	201
3	240.0	240.0	38.0	94.0	0.404	9.0	29.0	0.310	10.0	17.0	...	162.0	104.0	CHI	97	1490.000000	1510.000000	1	201
4	240.0	240.0	37.0	96.0	0.385	12.0	29.0	0.414	20.0	26.0	...	258.0	121.0	ATL	94	1510.000000	1490.000000	1	201
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
22357	240.0	240.0	34.0	72.0	0.472	11.0	30.0	0.367	21.0	25.0	...	222.0	122.0	MIN	118	1579.312418	1589.349213	0	202
22358	240.0	240.0	30.0	82.0	0.366	7.0	39.0	0.179	27.0	35.0	...	166.0	104.0	SAC	129	1572.320905	1544.578578	1	202
22359	240.0	240.0	48.0	89.0	0.539	9.0	28.0	0.321	16.0	19.0	...	233.0	124.0	OKC	111	1524.935644	1615.462853	1	202
22360	240.0	240.0	43.0	99.0	0.434	14.0	37.0	0.378	11.0	15.0	...	149.0	119.0	IND	121	1615.462853	1524.935644	0	202
22361	240.0	240.0	40.0	79.0	0.506	14.0	40.0	0.350	12.0	15.0	...	162.0	132.0	PHI	79	1546.617110	1500.779402	0	202

22362 rows × 152 columns

Figure 5.9: Dataset with Elo Rating

This feature will play an essential role in the machine learning model, as it will act as a sort of classification system for teams throughout the dataset. Teams with a higher Elo Rating will be teams that tend to fight for the title and are therefore more likely to win their matches.

5.2.2 Rest Days Between Matches

This variable will add information about the number of rest days between each match. In the NBA it is quite common to have very few rest days between games, sometimes there are even two games on two consecutive days. This feature can provide some information about the team's physical condition for the game.

Figure 5.10 shows how this feature was implemented. First, the data in the date column had to be converted into a date and time format that can easily be used by the Pandas library to manipulate and analyze this data set. After this, the dataset is separated into teams and by date to ensure that the data is sorted correctly for subsequent manipulation.

It then calculates the difference in days between the current date and the date of the team's next match. The values are then converted to integers so that they can later be interpreted by machine learning algorithms. A function has also been created that checks if there has been a change of season and resets the rest days to 0 if this happens. It would not make sense to count the rest days that fall during the change of season because it is practically the same for all teams and would interfere with the normalization scales. The opponent's rest days will also be calculated later. The lambda function is applied to find the corresponding line for the opponent on the same date and extracts the opponent's rest days for that match.

```
df = pd.read_csv("all_teams_data_with_elo.csv", index_col=0)
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values(by=['team', 'date'])
df['days_since_last_game'] = df.groupby('team')['date'].diff().dt.days.fillna(0).astype(int)

def reset_rest_days(group):
    season_changed = group['season'].ne(group['season'].shift())
    group['days_since_last_game'] = group['days_since_last_game'].where(~season_changed, 0)
    return group

df = df.groupby('team').apply(reset_rest_days)

df['days_since_last_game_opp'] = df.apply(lambda row: df.loc[(df['team'] == row['team_opp']) &
                                                             (df['team_opp'] == row['team']) &
                                                             (df['date'] == row['date']),
                                                             'days_since_last_game'].values[0], axis=1)
```

Figure 5.10: Feature Code of Rest Days Between Matches

Figure 5.11 shows the rest periods between two randomly selected teams between January and February of the 2019 season. It can be seen that CLE has two more back-to-backs (two games in two days) than the LAL team. CLE played more games with a two-day break than the other team and played fewer games with a three-day break. The CLE team played 15 games while the LAL team played 10, which could be a decisive factor in the results between games. The fact that one team played 5 more games than the other during the same period means that there will be fewer rest days between games for the team that played more games.

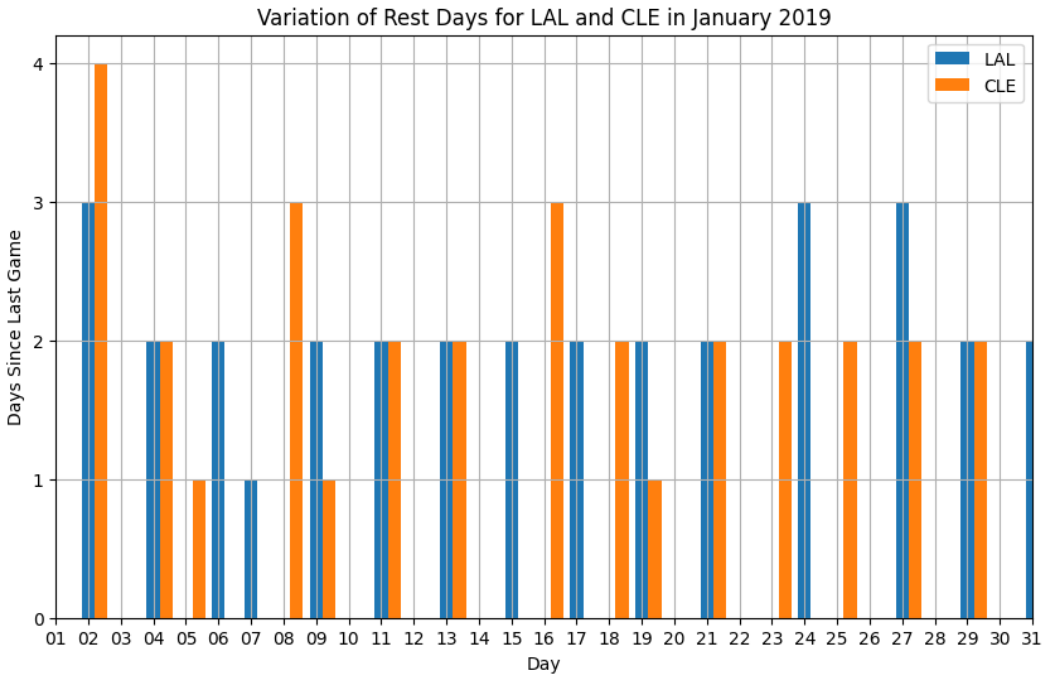


Figure 5.11: Rest Days between Two Teams

5.2.3 Team Averages

This feature represents a significant addition by Feature Engineering, introducing a series of new features for machine learning. These features are derived from the statistical averages calculated by the team.

In basketball, it is generally quite common to look at small game samples to understand the team's physical and psychological state. The usual approach is to evaluate the team's statistics over the last 10 games which is the way most used by the NBA to convey the team's recent form. But, sometimes the last 10 games may not be a true sample of what is happening. The team may be turning its form around. For example, in the last 10 games a team winning 5 and losing 5 may not seem so good, but if in the previous 5 games of those 10 it has won all 5, it may mean something else that only averages of statistics related to the last 10 games would not be perceptible. So the decision was taken not only to use statistical averages from the last 10 games but also from the last 5 and 7 to identify patterns that would significantly impact the machine learning algorithm.

Figure 5.12 represents the code used to take the averages of the last 5, 7, and 10 games. The "find_team_averages" functions will calculate the average of the last 10, 7, and 5 games respectively. The "groupby" method is used to group the data by team and by season to obtain the averages of the respective teams in the correct chronological order. The average functions will be divided by season. It is quite normal for teams to change a lot from season to season, from coaches to players, and

even team strategy. So it would not make sense to pass on the continuity of averages from one season to the next, when the season ends this variable is reset and started again. After that, the functions are applied, but only the value resulting from the average functions is to be passed on. It is therefore necessary to set "groupkey=false" so as not to pass on the keys that are used to perform the averages, such as the team and the season.

```
def find_team_averages_10(team):
    rolling = team.rolling(10).mean()
    return rolling

def find_team_averages_7(team):
    rolling = team.rolling(7).mean()
    return rolling

def find_team_averages_5(team):
    rolling = team.rolling(5).mean()
    return rolling


df_rolling_10 = df_rolling.groupby(["team", "season"], group_keys=False).apply(find_team_averages_10)
df_rolling_7 = df_rolling.groupby(["team", "season"], group_keys=False).apply(find_team_averages_7)
df_rolling_5 = df_rolling.groupby(["team", "season"], group_keys=False).apply(find_team_averages_5)
```

Figure 5.12: Code for Calculating the Averages

With the implementation of the statistic of averages of previous matches, it will not be possible to predict the first 10 matches of each season, because it is not possible to calculate the average of these matches, as there have not been enough matches for this calculation to be made. Figure 5.13 shows the dataframe of the average of the games played, where it can be seen null values for the initial lines because there are not enough games played to calculate the average.

	mp_10	fg_10	fga_10	fg%_10	3p_10	3pa_10	3p%_10	ft_10	fta_10	ft%_10	...	ortg_max_opp_10	drtg_max_opp_10	total_opp_10	team_elo
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
22357	0.000	0.506522	0.417647	0.503110	0.493103	0.468182	0.487886	0.327907	0.285714	0.756943	...	0.514692	0.565882	0.465179	0.155
22358	0.000	0.400000	0.436765	0.355024	0.424138	0.486364	0.404869	0.302326	0.253968	0.787048	...	0.509953	0.491765	0.385714	0.531
22359	0.000	0.569565	0.473529	0.535167	0.468966	0.465152	0.459739	0.330233	0.268254	0.833372	...	0.385308	0.595294	0.413393	0.710
22360	0.050	0.467391	0.416176	0.454785	0.393103	0.439394	0.409382	0.376744	0.328571	0.767211	...	0.312322	0.522353	0.377679	0.677
22361	0.025	0.508696	0.469118	0.461483	0.424138	0.503030	0.389905	0.304651	0.269841	0.746208	...	0.554976	0.470588	0.518750	0.099

22362 rows × 142 columns

Figure 5.13: Dataframe of the Avarages with Null Values

It is necessary to rename the columns containing the average values to avoid overlap with other columns and ensure that the columns are properly added.

Figure 5.14 shows how the averages dataset will look with the columns renamed and the rows with null values removed. It is important to mention that columns

previously added as Feature Engineering will also be the target of the latest match averages.

	mp_10	fg_10	fga_10	fg%_10	3p_10	3pa_10	3p%_10	ft_10	fta_10	ft%_10	...	ortg_max_opp_5	drtg_max_opp_5	total_opp_5	team_elo_5
241	0.000	0.447826	0.357353	0.482775	0.320690	0.328788	0.430641	0.346512	0.295238	0.792649	...	0.314692	0.515294	0.350000	0.584683
247	0.000	0.506522	0.414706	0.506699	0.420690	0.392424	0.493349	0.395349	0.357143	0.737806	...	0.333649	0.517647	0.308929	0.627488
257	0.075	0.417391	0.427941	0.383014	0.296552	0.312121	0.422565	0.351163	0.307937	0.738273	...	0.450237	0.430588	0.273214	0.485386
258	0.000	0.380435	0.397059	0.358373	0.251724	0.287879	0.381710	0.409302	0.344444	0.782497	...	0.353555	0.414118	0.300000	0.479557
260	0.000	0.365217	0.330882	0.396172	0.268966	0.266667	0.420071	0.374419	0.358730	0.687981	...	0.257820	0.411765	0.260714	0.511301
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
22357	0.000	0.506522	0.417647	0.503110	0.493103	0.468182	0.487886	0.327907	0.285714	0.756943	...	0.545024	0.562353	0.451786	0.168737
22358	0.000	0.400000	0.436765	0.355024	0.424138	0.486364	0.404869	0.302326	0.253968	0.787048	...	0.423697	0.411765	0.337500	0.516418
22359	0.000	0.569565	0.473529	0.535167	0.468966	0.465152	0.459739	0.330233	0.268254	0.833372	...	0.385782	0.585882	0.410714	0.708744
22360	0.050	0.467391	0.416176	0.454785	0.393103	0.439394	0.409382	0.376744	0.328571	0.767211	...	0.357346	0.562353	0.425000	0.666926
22361	0.025	0.508696	0.469118	0.461483	0.424138	0.503030	0.389905	0.304651	0.269841	0.746208	...	0.592417	0.480000	0.433929	0.098723

19932 rows × 426 columns

Figure 5.14: Dataframe of the Avarages Completed

5.2.4 Information About Next Game

Despite all the information already given about the team, it is also necessary to provide information about the next match, which is the match to be predicted. Information about the opponent, whether the game is played at home or away can be crucial in determining results. This is information that can be known in advance of the match and can be passed on to the machine learning model.

Figure 5.15 shows how the first part of this feature was carried out. First, a function is created that will pass the information from the bottom row of the dataset to the top, meaning it will be moved up one position; then, another function is created that will group the dataset by the team and allow it to choose which column it is necessary to move up one position. Lastly, the necessary columns are chosen, whether the game is played at home or not. This information is very relevant because a quick analysis of the dataset shows that the teams that play at home win 57% of the time, which is a figure that must be beaten in the machine learning model. Another relevant piece of information will be who the next opponent is and the date of the next game. These two pieces of information will be needed to be able to gather statistical information about the next opposing team in the next step. It is necessary to be very careful when creating these features because it is important to ensure that information about the game to predict is not passed on, but simply information known before the game started.

Once the information has been obtained about who the next opponent is and on what date the game is being played, it is possible to add information about the statistics of the next opponent. Since this is the game to be predicted, it would be an excellent addition to having information about the opposing team's statistics. The "lambda" function is utilized to determine the upcoming opponent for the team's next match. This prediction is based on various factors such as the date of the

next match, the opponent, and all available information about that team, including statistics from their last match, Elo Rating, the duration of days without matches, and the averages of their last 5, 7, and 10 matches. In this way, it will be possible to pass on to the machine learning model a very complete view of the opponent of the match being predicted.

```python
def shift_col(team, col_name):
    next_col = team[col_name].shift(-1)
    return next_col

def add_col(df, col_name):
    return df.groupby("team", group_keys=False).apply(lambda x: shift_col(x, col_name))

df["home_next"] = add_col(df, "home")
df ["team_opp_next"] = add_col(df, "team_opp")
df["date_next"] = add_col(df, "date")
```

Figure 5.15: Code on the Next Opponent's Information

Figure 5.16 represents the dataset with these features already added, where it is possible to see a total of 1003 features. All statistics with the prefix "y" belong to the next opponent, while those with the prefix "x" belong to the team that is being predicted.

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	total_opp_5_y	team_elo_5_y	opp_elo_5_y	days_since_last_game
0	0.00	0.456522	0.500000	0.375598	0.379310	0.348485	0.483373	0.441860	0.396825	0.730455	...	0.291071	0.546433	0.450146	0.01
1	0.00	0.326087	0.250000	0.413876	0.310345	0.257576	0.509501	0.511628	0.412698	0.827305	...	0.367857	0.423179	0.517467	0.01
2	0.25	0.478261	0.558824	0.356459	0.068966	0.212121	0.131829	0.325581	0.238095	0.927655	...	0.351786	0.579808	0.487285	0.01
3	0.25	0.521739	0.544118	0.416268	0.413793	0.454545	0.419240	0.186047	0.142857	0.883314	...	0.317857	0.541567	0.448435	0.01
4	0.50	0.391304	0.455882	0.330144	0.482759	0.515152	0.437055	0.372093	0.412698	0.568261	...	0.341071	0.527320	0.500481	0.01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
19799	0.00	0.478261	0.544118	0.368421	0.551724	0.636364	0.413302	0.232558	0.190476	0.820303	...	0.398214	0.672099	0.575572	0.01
19800	0.00	0.260870	0.294118	0.284689	0.310345	0.393939	0.356295	0.162791	0.206349	0.499417	...	0.280357	0.578290	0.567785	0.01
19801	0.00	0.608696	0.647059	0.437799	0.482759	0.651515	0.353919	0.186047	0.142857	0.883314	...	0.405357	0.668114	0.542578	0.01
19802	0.00	0.434783	0.485294	0.358852	0.482759	0.500000	0.448931	0.209302	0.222222	0.611435	...	0.398214	0.375698	0.446723	0.01
19803	0.00	0.652174	0.588235	0.528708	0.517241	0.545455	0.445368	0.162791	0.174603	0.611435	...	0.530357	0.414470	0.499380	0.01

19804 rows × 1003 columns

Figure 5.16: Dataset with Opponent Information

5.2.5 Dataset Handling

After creating the new features through Feature Engineering, there will be columns that cannot be passed to the machine learning model because they are of the object data type. To ensure accurate interpretation, it is important to provide the model with only numerical data. Columns such as next team, next match date, or even season cannot be sent to train the model because it is not possible to switch to a numerical format.

The dataset depicted in Figure 5.17 will be utilized to train and validate the model. It contains 993 features, and to manage this large number, a feature selector

will be employed to choose the most appropriate features for each machine learning algorithm.

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	ortg_max_opp_5_y	drtg_max_opp_5_y	total_opp_5_y	team_elo_5_y	opp_elo_5_y	days_since_last
0	240.0	40.0	94.0	0.426	11.0	27.0	0.407	20.0	26.0	0.769	...	157.8	117.6	96.6	1526.310287	1471.784465	
1	240.0	34.0	77.0	0.442	9.0	21.0	0.429	23.0	27.0	0.852	...	154.2	109.6	105.2	1456.513340	1509.907134	
2	265.0	41.0	98.0	0.418	2.0	18.0	0.111	15.0	16.0	0.938	...	144.6	115.6	103.4	1545.209702	1492.815805	
3	265.0	43.0	97.0	0.443	12.0	34.0	0.353	9.0	10.0	0.900	...	192.0	118.8	99.6	1523.554816	1470.815331	
4	290.0	37.0	91.0	0.407	14.0	38.0	0.368	17.0	27.0	0.630	...	164.8	111.2	102.2	1515.486990	1500.288428	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
19799	240.0	41.0	97.0	0.423	16.0	46.0	0.348	11.0	13.0	0.846	...	204.0	124.2	108.6	1597.473088	1542.811223	
19800	240.0	31.0	80.0	0.388	9.0	30.0	0.300	8.0	14.0	0.571	...	159.8	117.4	95.4	1544.350445	1538.401626	
19801	240.0	47.0	104.0	0.452	14.0	47.0	0.298	9.0	10.0	0.900	...	169.2	118.2	109.4	1595.216113	1524.127160	
19802	240.0	39.0	93.0	0.419	14.0	37.0	0.378	10.0	15.0	0.667	...	182.6	127.8	108.6	1429.625229	1469.845737	
19803	240.0	49.0	100.0	0.490	15.0	40.0	0.375	8.0	12.0	0.667	...	213.4	129.4	123.4	1451.581278	1499.664726	

19804 rows × 993 columns

Figure 5.17: Dataset of the NBA that will be Used for Machine Learning

The same Feature Engineer process was carried out for the WNBA. Figure 5.18 represents the dataset that will be used for the WNBA.

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	ortg_max_opp_5_y	drtg_max_opp_5_y	total_opp_5_y	team_elo_5_y	opp_elo_5_y	days_since_last_game_5_y
0	200	29	59	3.571	5	12	1.416	20	25	4.250	...	199.0	110.6	81.2	1492.482397	1484.113415	2.8
1	200	30	73	4.790	5	22	1.929	11	11	5.000	...	157.8	108.4	87.8	1452.513340	1507.524569	3.0
2	200	24	68	2.758	9	24	1.250	29	34	4.257	...	171.0	129.4	71.4	1572.210588	1479.111005	3.8
3	200	37	72	5.344	3	12	0.900	16	21	4.700	...	154.4	120.8	79.6	1499.966059	1500.360588	3.2
4	200	27	67	4.261	7	21	1.400	27	30	6.472	...	171.8	116.0	73.4	1572.210588	1488.574574	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2535	200	32	69	3.335	9	29	1.921	9	13	3.667	...	154.6	123.0	76.0	1719.486228	1535.906325	5.0
2536	200	37	70	3.717	13	29	2.471	17	17	5.000	...	177.0	118.0	89.0	1646.349046	1637.710128	3.4
2537	200	26	72	3.403	8	35	1.508	16	22	4.881	...	159.4	125.4	77.2	1723.895052	1564.737447	4.8
2538	200	33	63	3.292	13	30	2.983	8	16	2.500	...	155.6	120.2	78.0	1724.217238	1586.635226	4.2
2539	200	23	69	1.635	7	22	1.033	20	23	4.600	...	172.2	118.6	88.2	1644.371888	1663.392685	3.8

2540 rows × 951 columns

Figure 5.18: Dataset of the WNBA that will be Used for Machine Learning

It is evident that the dataset contains 951 features and fewer games compared to the NBA dataset. The reason for this discrepancy is that certain WNBA statistics, like defensive rebounds and plus/minus (+/-), are not included.

5.3 Machine Learning Architecture

The following section provides insight into the development and operation of the machine learning model. It is essential to adhere to the chronological order of the data to ensure that the algorithm does not receive information about the future it is intended to predict.

5.3.1 Scikit-Learn

Scikit-learn is an open-source library widely used in machine learning for Python. It is currently the most commonly used library in this domain. With a wide range of functionalities, from classification and regression algorithms to clustering and data pre-processing, Scikit-Learn has become an essential tool for data scientists and developers.

This library is built on libraries such as NumPy, SciPy and Matplotlib. It also provides plenty of data pre-processing tools such as normalization, feature selection, model selection, and model evaluation.

Figure 5.19 shows the sequence used in the work on predicting basketball matches. The concepts are explained as follows:

- Normalization: Used for all the values in the dataset to have the same scale to facilitate interpretation by the Machine Learning model.
- Feature Selection: Used to select the most relevant features to prevent the target to prevent overfitting.
- Machine Learning Models: Testing various machine learning algorithms.
- Model Evaluation: Evaluating the accuracy of each Machine Learning model.

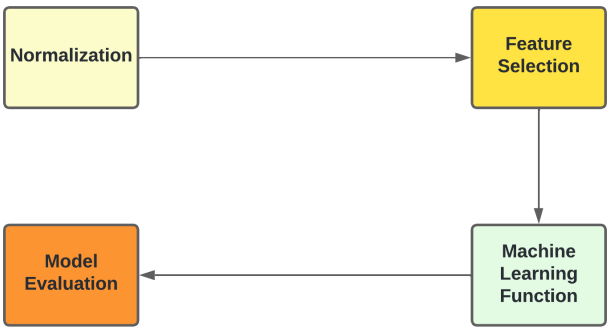


Figure 5.19: Machine Learning Architecture

5.3.2 Normalization of Values

Many machine learning algorithms require the normalization of values as the 993 variables may have different evaluation scales. For instance, a team may score 120 points but only have a 33 % accuracy on the three-point line. To eliminate the effect of different evaluation scales throughout the features, it is necessary to normalize the scales.

Normalization is crucial for this work because it avoids feature dominance. In datasets where features have different scales, those with numerically larger values can dominate over the others during model training, which would result in a model biased towards larger scales and ignoring features with smaller scales. It makes interpretation easier because by assigning the same scale to all the characteristics, they will be equally comparable in relation to the variable we want to predict.

Two types of normalization can be used: normalization by standard deviation and min-max normalization.

Standard Deviation

Standard deviation normalization is a statistical method used to scale data so that it has a mean of zero and a standard deviation of one. First, the mean and standard deviation are calculated for the variable and then the data is normalized using the Equation (5.1). This method is not suitable for data sets that do not follow a normal distribution.

$$\text{Normalized data} = \frac{\text{Original data} - \text{Mean}}{\text{Standard Deviation}} \tag{5.1}$$

Min-Max Normalization

Min-max normalization, one of the most popular normalization techniques, aims to restrict the range of variables between 0 and 1 (or -1 to 1, if there are negative values), adjusting the data accordingly. This approach, described by Equation (5.2), is particularly useful when the data does not follow a normal distribution or when the standard deviation is low. However, it is important to note that min-max normalization can be sensitive to extreme values, undermining its effectiveness in data sets with pronounced outliers.

$$\text{Normalized data} = \frac{\text{Original data} - \text{Min}}{\text{Max} - \text{Min}} \tag{5.2}$$

The data will be normalized using this method and the Python Scikit-learn package. Features such as Team Averages were first normalized and only then was the feature created. This makes it possible to reduce the influence of outliers. Without normalization, outliers can distort the average or other features. Normalization before creating certain statistics can result in more balanced features and more robust models.

Figure 5.20 shows how the dataset for the NBA looks like after normalization with Min-Max.

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	ortg_max_opp_5_y	drtg_max_opp_5_y	total_opp_5_y	team_elo
0	0.00	0.431818	0.500000	0.375598	0.392857	0.348485	0.483373	0.441860	0.396825	0.730455	...	0.238364	0.437229	0.269663	0.54
1	0.00	0.295455	0.250000	0.413876	0.321429	0.257576	0.509501	0.511628	0.412698	0.827305	...	0.212976	0.264069	0.430712	0.42
2	0.25	0.454545	0.558824	0.356459	0.071429	0.212121	0.131829	0.325581	0.238095	0.927655	...	0.145275	0.393939	0.397004	0.58
3	0.25	0.500000	0.544118	0.416268	0.428571	0.454545	0.419240	0.186047	0.142857	0.883314	...	0.479549	0.463203	0.325843	0.54
4	0.50	0.363636	0.455882	0.330144	0.500000	0.515152	0.437055	0.372093	0.412698	0.568261	...	0.287729	0.298701	0.374532	0.52
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
19799	0.00	0.454545	0.544118	0.368421	0.571429	0.636364	0.413302	0.232558	0.190476	0.820303	...	0.564175	0.580087	0.494382	0.67
19800	0.00	0.227273	0.294118	0.284689	0.321429	0.393939	0.356295	0.162791	0.206349	0.499417	...	0.252468	0.432900	0.247191	0.58
19801	0.00	0.590909	0.647059	0.437799	0.500000	0.651515	0.353919	0.186047	0.142857	0.883314	...	0.318759	0.450216	0.509363	0.67
19802	0.00	0.409091	0.485294	0.358852	0.500000	0.500000	0.448931	0.209302	0.222222	0.611435	...	0.413258	0.658009	0.494382	0.37
19803	0.00	0.636364	0.588235	0.528708	0.535714	0.545455	0.445368	0.162791	0.174603	0.611435	...	0.630465	0.692641	0.771536	0.41

19804 rows × 993 columns

Figure 5.20: Dataset Normalized

5.3.3 Feature Selection

In datasets with a high number of features, it is necessary to use a Feature Selector before using the machine model. The NBA dataset has 993 features and WNBA has 951. Passing this number of features to a machine learning model risked several problems that could jeopardize its accuracy in predicting NBA games. Dimensionality reduction is very important in datasets with many features, some of which may be irrelevant or even detrimental to the model's performance. By removing these irrelevant or harmful features, the feature selector can improve the model's performance by reducing the noise in the data and focusing on the irrelevant features.

Complex models can fit the training data quite well, which is called overfitting, capturing the noise in the data. A feature selector helps mitigate overfitting by removing features that do not contribute significantly to the model's ability to generalize. Fewer features also mean less training time, which is very beneficial in terms of computational performance.

ElasticNetCV is an algorithm that is utilized to determine the most critical features in machine learning. This method combines regression techniques such as Lasso and Ridge, which are beneficial for feature selection. ElasticNet employs cross-validation to ascertain the optimal parameters for the model.

By combining techniques such as Lasso regression (L1 regularization) which penalizes the sum of the absolute coefficient values, this technique is useful for selecting characteristics, as it can reduce some coefficients to zero. Ridge regression (L2 regularization) penalizes the sum of the squares of the coefficients. It is useful for dealing with multicollinearity but does not perform trait selection, as it tends to keep all coefficients different from zero. The ElasticNET formula is defined by:

$$\left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \alpha \left(\rho \sum_{j=1}^{p} |\beta_j| + \frac{(1-\rho)}{2} \sum_{j=1}^{p} \beta_j^2 \right) \right\} \tag{5.3}$$

- y_i: Response variable for the i-th observation;
- β_0: Intercept term;
- x_{ij}: Value of the j-th predictor for the i-th observation;
- β_j: Coefficient for the j-th predictor;
- n: Number of observations;
- p: Number of predictors;
- α: Overall regularization strength (a non-negative parameter);
- ρ: Mixing parameter between L1 (Lasso) and L2 (Ridge) regularization.

Cross-validation is a widely used method in machine learning to assess the model's ability to generalize. This method divides the dataset into subsets, which are known as training and test subsets. However, when working with time series, it is important to respect the temporal order of the data. Using "TimeSeriesSplit" instead of cross-validation is essential due to the sequential nature of time series data, as shown in Figure 5.21. In time series, the order of the data is crucial because each data point is related to the previous point and the next point. In this case, cross-validation can scramble the data by breaking the temporal order, which would be cheating because we would be going into the future to find out the relevant features. "TimeSeriesSplit", by maintaining the temporal order, allows us to split the training and test data to find the most relevant features. It is guaranteed that the training data precedes the test data, thus maintaining a temporal order.

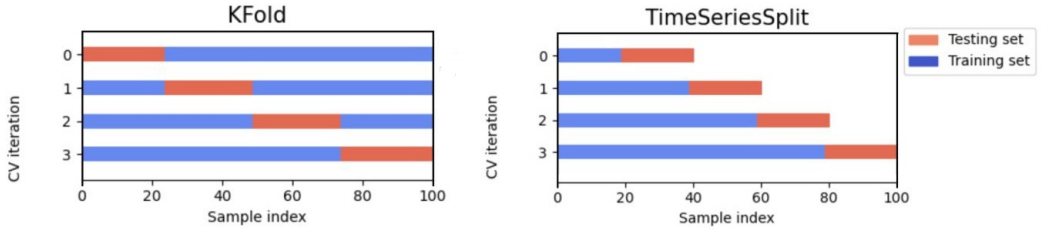


Figure 5.21: Cross-Validation vs TimeSeriesSplit

Pearson's correlation was also used to understand the correlation between the variables. Understanding correlation is essential as highly correlated variables, despite their significance, may provide redundant information to the algorithm. Pearson's correlation is defined by Equation (5.4). A correlation close to the coefficient of 1 indicates that the features are highly similar. Consequently, in instances where two features are highly similar, they may not provide as much relevant information

to the machine learning algorithm due to their overlapping significance. It is important to remove one of the features that is highly correlated to avoid redundancy between the chosen variables.

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \tag{5.4}$$

Where:

- X_i and Y_i: The individual sample points of variables X and Y.
- $\bar{X}$ and $\bar{Y}$: The means of X and Y, respectively.

Figure 5.22 shows a schematic of how the Feature Selector was developed. Initially, ElasticNetCV is used to remove all features that have an importance value of 0 for the "Target" variable, thus eliminating unnecessary variables that would not be useful.

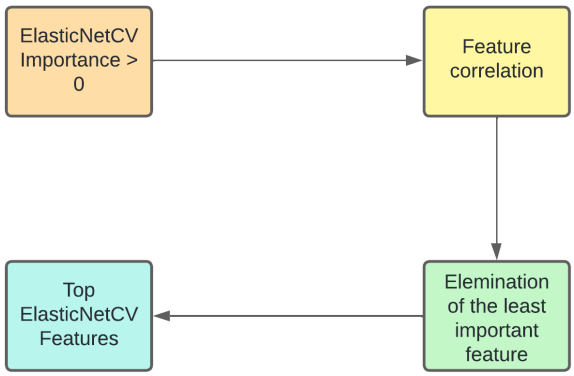


Figure 5.22: Feature Selection Scheme

After obtaining the features that have some importance, Pearson's correlation is used to eliminate the highly correlated variables, as shown in Figure 5.23. The pairwise correlation of all the variables is then calculated; those with more than 90 % correlation are compared to the "Target" variable and the one with the least correlation to this variable is eliminated, thus keeping the feature that can most help predict the final result of the game.

Following the application of ElasticNetCV, we will compile a comprehensive list of the most significant features for implementation in our machine learning models. We anticipate having 100 variables available for the NBA and 17 for the WNBA. It appears that establishing relationships between the "Target" variable and the features in the WNBA dataset may present a greater level of complexity.

```python
correlation_matrix = full[selected_features].corr()


high_corr_pairs = []
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.9:
            high_corr_pairs.append((correlation_matrix.columns[i], correlation_matrix.columns[j]))


features_to_remove = []
for (feature1, feature2) in high_corr_pairs:
    corr1 = abs(full[[feature1, 'target']].corr().iloc[0, 1])
    corr2 = abs(full[[feature2, 'target']].corr().iloc[0, 1])
    if corr1 > corr2:
        features_to_remove.append(feature2)
    else:
        features_to_remove.append(feature1)

features_to_keep = [feature for feature in selected_features if feature not in features_to_remove]
```

Figure 5.23: Pearson Correlation Code

This approach is of great importance as it helps simplify the model, enhance interpretability, and potentially improve predictive performance by focusing on the most significant features in the dataset.

5.3.4 Machine Learning Function

Chronological order is very important. It is essential to ensure that historical data is used to predict future data. This approach is quite common in predictive modeling, where the model is trained on historical data up to a certain point and tested on future data to evaluate its performance. The training data allows the model to learn patterns about the historical data. After training the model, it is tested on future data to make predictions. The predictions will be compared with the actual data values to calculate metrics such as accuracy, providing information about the model's ability to make accurate predictions about the data.

It was then necessary to divide the data by season. Figure 5.24 illustrates the division of data for training and testing works. Initially, the first two seasons contained in the 2015/2016 and 2016/2017 datasets will only be used for training and to make forecasts for the 2017/2018 season. After that, the 2015/2016, 2016/2017 and 2017/2018 seasons will be used to make forecasts for the 2018/2019 season and so on until the 2023/2024 season. This ensures that the training data are past events concerning the test data. If this chronological division were not carried out, better results would probably be obtained. However, it would be wrong to use training data from future matches to test past matches, as this is not how the real world works. When making predictions, past data is always taken into account to predict future events, not the other way around.

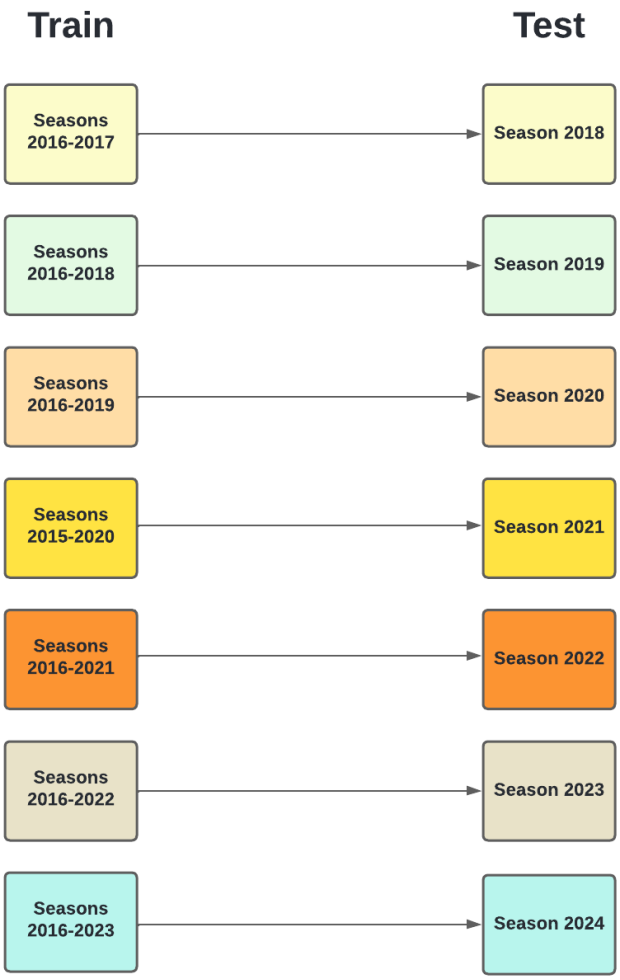


Figure 5.24: Machine Learning Dataset Splitting

5.3.5 Model Evaluation

Several parameters could be evaluated, but the most predominant is the accuracy of the model. When trying to predict something like the outcome of a sports game such as basketball, in which there are two possible outcomes such as winning or losing, accuracy ends up being the most important. The accuracy for machine learning is represented by Equation (5.5). The time it takes to train and validate each algorithm will also be indicated to understand its complexity and computational weight.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{False Positives} + \text{True Negatives} + \text{False Negatives}} \tag{5.5}$$

Where:

- True Positives: Number of cases correctly classified as positives;

- True Negatives: Number of cases correctly classified as negatives;
- False Positives: Number of cases incorrectly classified as positives;
- False Negatives: Number of cases incorrectly classified as negatives.

Although accuracy is the most important measure for the classification of this work, other measures can be evaluated, such as precision, which is represented by Equation (5.6). Precision in this context is the proportion of correct predictions of wins (true positives) concerning the total number of predictions of wins made by the model (true positives plus false positives). This metric allows the question to be answered of all the times the model predicted a win, and how many of those predictions were correct. This parameter makes it possible to assess the accuracy of the model's predictions of victories.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.6}$$

Recall is another metric that can be used in the basketball classification model. This metric, defined by Equation (5.7), measures the model's ability to detect all the real wins (i.e. those that actually happened) in the official data set. This metric allows to answer the question of how many of the real wins the model could predict correctly. A high recall means that the model is good at identifying the majority of real wins.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.7}$$

The F1-Score defined by Equation (5.8) is a performance metric that considers both precision and recall. It is useful for finding a balance between these two aspects, especially when there is an imbalance between the classes.

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.8}$$

We will conduct rigorous testing with varying numbers of features, ranging from 15 to 100, to comprehensively analyze their impact on the accuracy of the Machine Learning model for the NBA. Because the WNBA has fewer relevant features, a study will be carried out with the 7 and the 17 best features to understand the impact of feature variation. The objective is to meticulously evaluate different algorithms and their combinations to achieve optimal results.

The behavior of the algorithm with the best metrics will also be studied throughout the test seasons to understand the levels of accuracy that would be obtained for each season.

An exclusive forecast for the 2024 season will also be tested, in which the previous seasons will be the scope of the training set and the 2024 season the test set. This

will make it possible to understand which algorithm is best for predicting the season we are currently in.

Chapter 6

Results

This chapter, we will be presenting the outcomes of the diverse algorithms and features that have been tested for the NBA and WNBA models.

6.1 NBA

First, the NBA results will be presented. Subsequently, various algorithms will be tested using the top 15, 30, 50, 75, and 100 features selected by the Feature Selector. The metrics employed will be those outlined in Chapter 5.

The algorithms covered are *Logistic Regression* (LR), *Ridge Regression Classifier* (RR), *Random Forest* (RF), *Naives Bayes* (NB), KNN, SVM, *Stacking Classifier* (STACK), *Bagging Classifier* (BAG), *Multi-Layer Perceptron* (MLP), *AdaBoost Classifier* (AB), and *XGBoost Classifier* (XGB).

Table 6.1 indicates the meaning of the letters in the statistics that will be selected by the Feature Selector.

Table 6.1: Explanation of the Name of Statistics

Name	**Meaning**
x	Statistics on the team we want to predict
y	Statistics on the opponent of the match we want to predict
opp	Opponent statistics
5	Average of the last 5 statistics
7	Average of the last 7 statistics
10	Average of the last 10 statistics

The parameterization of the algorithms was also varied. Table 6.2 shows how this variation occurred and which parameters were chosen for each algorithm. Parameters that are omitted have the default value. The "Parameter Variation" column displays the range of values over which each parameter has been varied. Given that the STACK algorithm is a composite of multiple algorithms, various combinations were thoroughly tested, and the final choice was based on the combination that yielded the best results.

Table 6.2: Table with Algorithm Parameterization

Algorithm	**Parameter Variation**	**Parameter Chosen**
LR	$1000 \leq$ max iter ≤ 3000 $1 \leq$ verbose ≤ 5 $0 \leq$ random_state ≤ 100	max_iter = 2000 verbose = 2 random_state = 42
RR	$0.1 \leq$ alpha ≤ 5	alpha = 0.8
RF	$50 \leq$ n_estimator ≤ 500 $1 \leq$ max_deph ≤ 10 $1 \leq$ min_samples_split ≤ 20 $1 \leq$ min_samples_leaf ≤ 10 $0 \leq$ random_state ≤ 100	n_estimator = 200 max_deph = 5 min_samples_split = 10 min_samples_leaf = 5 random_state = 42
NB	$0.1 \leq$ priors ≤ 0.9	priors = 0.5; 0.5
KNN	$3 \leq$ n_neighbors ≤ 400	n_neighbors = 275
SVM	$0.1 \leq$ C ≤ 15 $0 \leq$ random_state ≤ 200	C = 0.3 random_state = 100
BAG	$0 \leq$ n_estimators ≤ 100 $0 \leq$ random_state ≤ 100	n_estimators = 11 random_state = 43
MLP	$0 \leq$ hidden_layer_size ≤ 100 $100 \leq$ max_iter ≤ 2000 $0.0001 \leq$ alpha ≤ 0.1	hidden_layer_size = 30; 15 max_iter = 1000 alpha = 0.001
AB	$50 \leq$ n_estimator ≤ 300 $0.0001 \leq$ learning_rate ≤ 0.1 $0 \leq$ random_state ≤ 100	n_estimators = 100 learning_rate = 0.09 random_state = 0
XGB	$50 \leq$ n_estimators ≤ 200 $0.0001 \leq$ learning_rate ≤ 0.1 $1 \leq$ max_depth ≤ 10 $1 \leq$ min_child_weight ≤ 10 $0.01 \leq$ gamma ≤ 1 $0.1 \leq$ subsample ≤ 1 $0 \leq$ colsample_bytree ≤ 2 $0 \leq$ random_state ≤ 100	n_estimators = 100 learning_rate = 0.05 max_depth = 2 min_child_weight = 2 gamma = 0.05 subsample = 0.8 colsample_bytree = 0.9 random_state = 42
STACK	Parameters used to test the other algorithms	RR RF KNN LR MLP

6.1.1 Top 15 Features

Figure 6.1 shows the Top 15 Features most related to the final result by ElasticNetCV. It appears that all the selected features have been derived from feature engineering. The two most important features are "team_elo" and "team_elo_5_y", which is an average of the last 5 matches of the opposing team's elo. We can also discern three statistics derived from the three-point shot, underscoring the significance of this parameter in today's NBA.

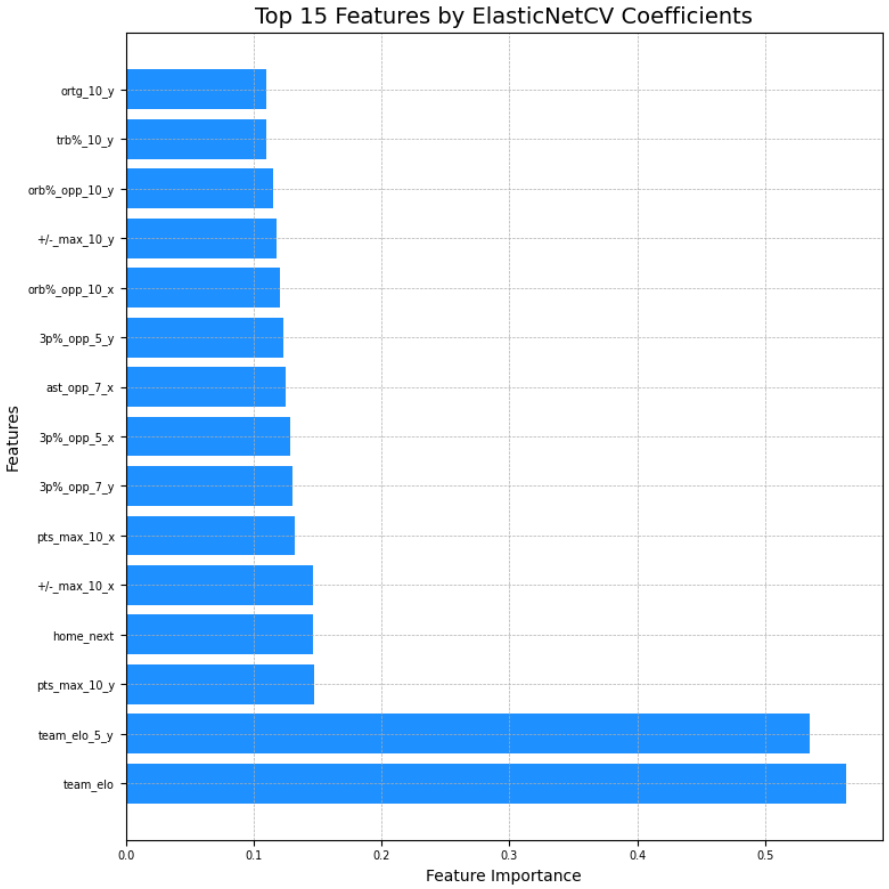


Figure 6.1: Top 15 Features by Feature Selector

Table 6.3 shows that the algorithms perform relatively similarly, with small variations in the metrics analyzed. The BAG algorithm showed the highest accuracy (65.06 %) and one of the best F1-Scores (64.66 %), indicating a good balance between precision and recall. This performance suggests that BAG is effective in correctly classifying game results, balancing detecting true positives, and minimizing false positives.

LR also showed competitive results, with an accuracy of 65.01 % and an F1-Score of 64.69 %. Although slightly inferior to BAG, LR proved to be robust and consistent.

The MLP stood out for its greater precision (66.08 %), suggesting that this model effectively avoids false positives. However, its recall (62.96 %) was slightly lower than BAG, which means that there may be more false negatives.

On the other hand, Naive Bayes had the lowest accuracy (63.91 %) and recall (58.21 %), indicating an inferior performance compared to the other algorithms. This result can be attributed to the conditional independence assumption made by Naive Bayes, which may not be well suited to the complexity of the interactions between variables in NBA data.

Table 6.3: Table with the Metrics of the Top 15 Features

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**
BAG	65.06 %	65.40 %	63.92 %	64.66 %
LR	65.01 %	65.40 %	63.74 %	64.69 %
STACK	64.89 %	65.40 %	63.20 %	64.28 %
RR	64.88 %	65.27 %	63.59 %	64.42 %
MLP	64.83 %	66.08 %	60.92 %	63.40 %
AB	64.82 %	64.95 %	64.36 %	64.66 %
XGB	64.73 %	64.98 %	63.85 %	64.40 %
SVM	64.53 %	65.09 %	62.67 %	63.86 %
KNN	64.50 %	64.95 %	62.96 %	63.94 %
RF	64.33 %	65.25 %	61.29 %	63.21 %
NB	63.91 %	65.68 %	58.21 %	61.72 %

6.1.2 Top 30 Features

The algorithm can leverage a diverse array of combinations for the top 30 features, as demonstrated in Figure 6.2. The inclusion of features remains an integral aspect of all feature engineering creations. However, the importance of features is increasingly being reduced, which can lead to lower accuracy percentages. New metrics concerning the launch of the three-point line are added such as “3p_max_opp” and “3pa_max_opp”. These variables are important for understanding how the team handles against other teams that have star players who specialize in the three-point shot. In a league where the three-point line plays a dominant role, it is not uncommon for certain players to have the ability to significantly influence the game on a particularly good day. The algorithm needs to comprehend how teams manage and respond to this type of performance.

Table 6.4, shows enhanced accuracy for the top 30 features in comparison to the previous table. This improvement can be attributed to the relevance of these new features, which contain highly pertinent information for the prediction task. Additionally, some of these features capture intricate interactions that were not apparent with a smaller number of features. Additional features up to a certain value can provide a variety of information that helps the model to better understand the data.

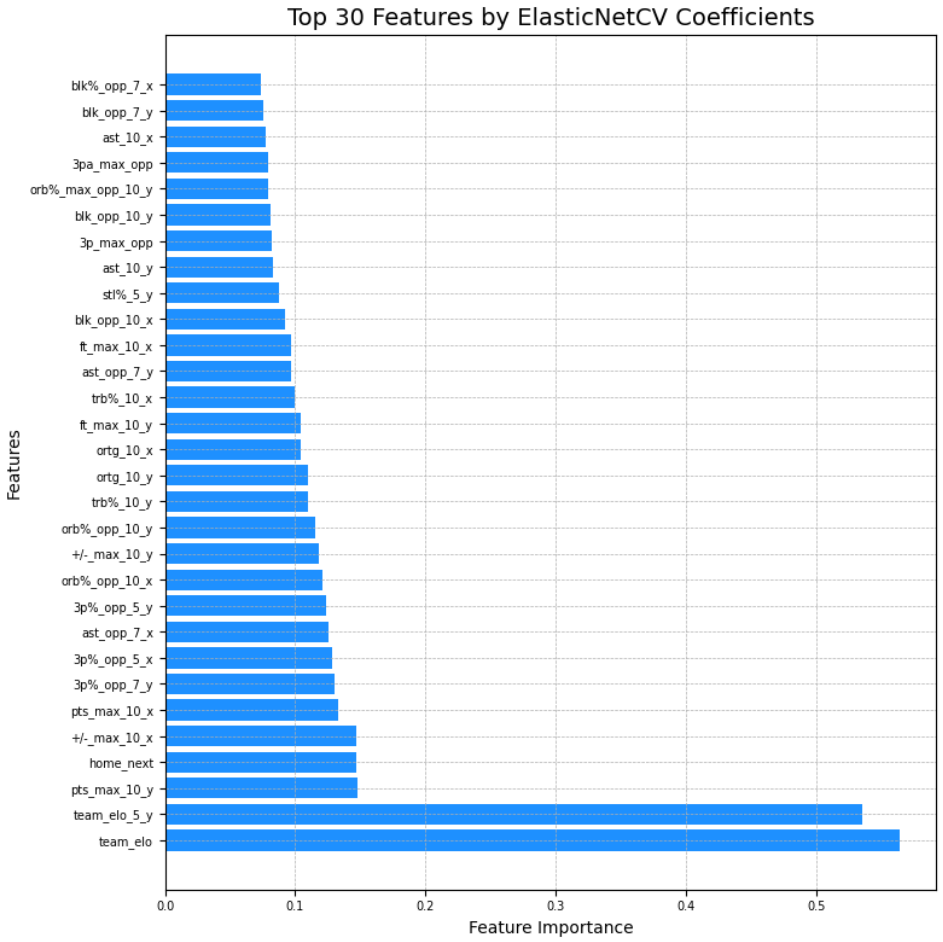


Figure 6.2: Top 30 Features by Feature Selector

The BAG model again shows the highest percentage of accuracy with 65.18 %, which is an increase in its accuracy with the Top 15 Features. Its F1-Score of 65.19 % shows an excellent balance between precision and recall.

The STACK algorithm showed an accuracy of 65.04 % and F1-Score of 65.07 %, standing out as one of the algorithms with the best overall performance. Stacking combines predictions from multiple base models to improve predictive performance, and its results indicate a good balance between the various evaluation metrics.

Although MLP accuracy is the second highest (65.07 %), its precision and recall are slightly lower than most of the other algorithms. This difference may be due to the way MLP handles data and its generalization characteristics. The algorithm has the potential to learn patterns that generally lead to increased accuracy. However, it may not always effectively improve precision and recall, particularly in instances where there are unbalanced classes or a certain amount of noise in the data.

LR and RR also showed competitive results, with an accuracy of 65.04 % and F1-Score of 65.07 % and 65.04 %, respectively. The data indicates a higher percentage compared to the Top 15 Features, suggesting that the new features have contributed

to the algorithm's improved understanding of previously unnoticed patterns.

Table 6.4: Table with the Metrics of the Top 30 Features

Algorithm	Accuracy	Precision	Recall	F1-Score	P-Accuracy (Top 15)
BAG	65.18 %	65.16 %	65.22 %	65.19 %	65.06 %
MLP	65.07 %	64.31 %	64.24 %	64.77 %	64.83 %
LR	65.04 %	65.01 %	65.13 %	65.07 %	65.01 %
STACK	65.04 %	65.00 %	65.14 %	65.07 %	64.89 %
RR	65.04 %	65.03 %	65.06 %	65.04 %	64.88 %
XGB	64.84 %	64.93 %	64.49 %	64.71 %	64.73 %
SVM	64.69 %	64.72 %	64.53 %	64.63 %	64.53 %
RF	64.65 %	64.74 %	64.31 %	64.52 %	64.33 %
AB	64.64 %	64.70 %	64.39 %	64.55 %	64.82 %
KNN	64.36 %	64.69 %	63.10 %	63.89 %	64.50 %
NB	63.74 %	63.78 %	63.55 %	63.67 %	63.91 %

P-Accuracy: Accuracy with the Top 15 Features.

6.1.3 Top 50 Features

The visualization in Figure 6.3 illustrates the Top 50 Features extracted by ElasticNetCV. It is noteworthy that feature-engineered features continue to dominate; however, it is becoming progressively challenging to establish a clear relationship between these features and the "Target" variables.

Table 6.5 presents the results for the Top 50 Features, showing an increase in in the MLP algorithm from 65.07 % to 65.49 %, and the highest recall seen so far of 66.09 %. This means that the model has a high ability to identify the games that were correctly the games that were actually positive, i.e. in the context of the NBA the games that were correctly predicted as wins.

BAG and STACK also showed competitive results, with accuracies of 65.39 % and 65.28 % respectively, which shows an increase in the results with the number of features used previously. They also have relatively high F1-Scores of 65.32 % and 65.40 % respectively, demonstrating their robustness and consistency in terms of predictions.

LR and RR continue to increase their level of accuracy. XGB showed an accuracy of 65.22 %, which is among the highest among the algorithms analyzed. This performance demonstrates that XGB is effective in predicting NBA games, managing to correctly predict the majority of game outcomes. XGB was one of the algorithms with the biggest increase within the Top 5 in accuracy with 50 features, which suggests that including more information can help improve the model's performance.

The Support Vector Machine also reaches 65.00 %, demonstrating that it can be effective in predicting NBA games. Its recall is slightly lower (64.33 %), indicating that the model may be missing some cases of real wins, i.e. generating false negatives. The F1-Score of 64.76% suggests a good balance between precision and recall, but

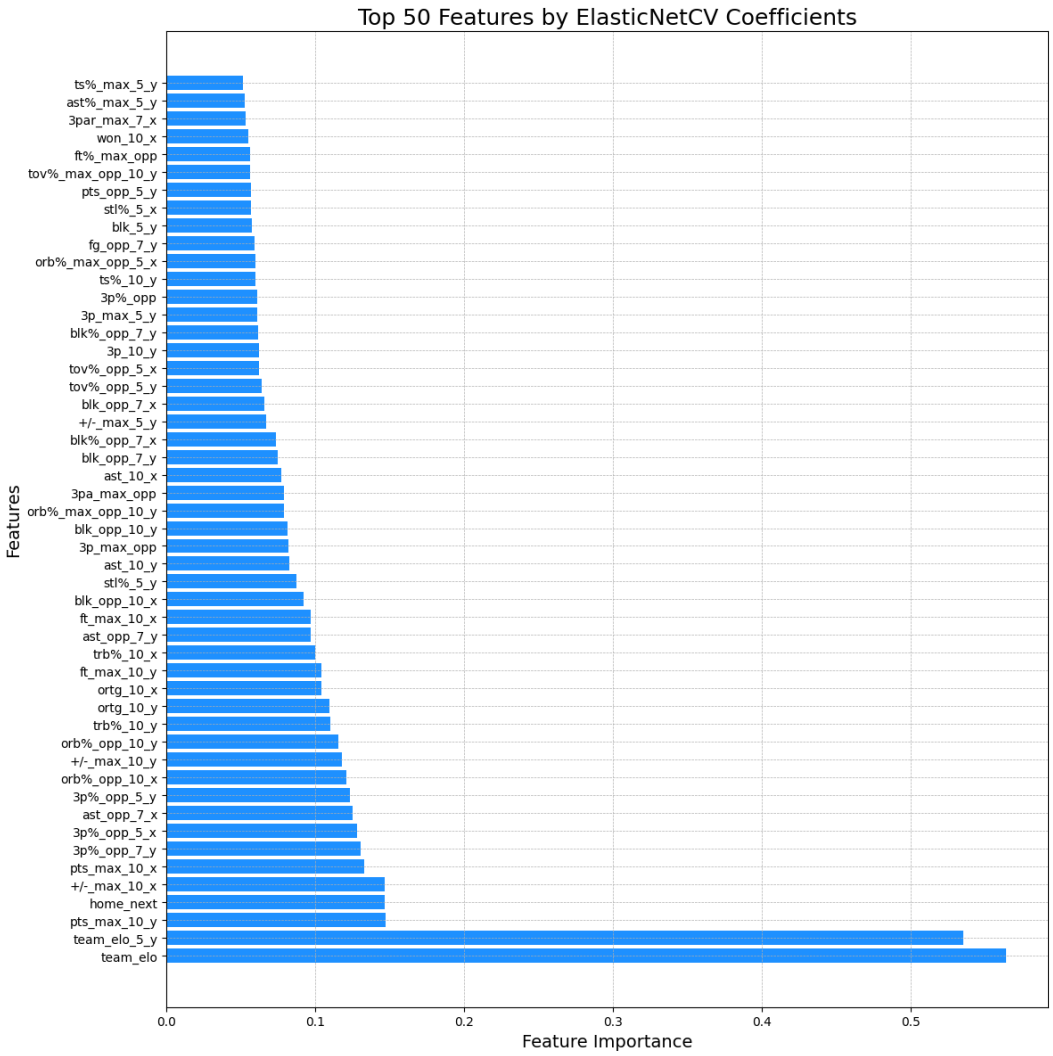


Figure 6.3: Top 50 Features by Feature Selector

with a slight tendency to be more precise than sensitive, which can be useful in scenarios where it is more critical to avoid false positives than to capture all the real ones.

6.1.4 Top 75 Features

The increasing complexity of the features in the Top 75 is apparent, as illustrated in Figure 6.4. The features developed by the Feature Engineers continue to prevail in the chosen categories.

In the Top 75 features, the algorithm exhibiting the highest level of accuracy is STACK. Table 6.6 shows that LR and STACK have slightly higher ratings than MLP with Top 50 Features. Although their accuracy is similar, they exhibit varying values in other metrics. STACK has a fractionally higher recall (66.03 % against 65.87 %), indicating that it is slightly more effective at identifying real wins. However, LR

Table 6.5: Table with the Metrics of the Top 50 Features

Algorithm	Accuracy	Precision	Recall	F1-Score	P-Accuracy (Top 15|30)
MLP	65.49 %	65.30 %	66.09 %	65.69 %	64.83 % | 65.07 %
BAG	65.39 %	65.44 %	65.21 %	65.32 %	65.06 % | 65.18 %
STACK	65.28 %	65.17 %	65.63 %	65.40 %	64.89 % | 65.04 %
LR	65.27 %	65.33 %	65.05 %	65.19 %	65.01 % | 65.04 %
XGB	65.22 %	65.32 %	64.86 %	65.09 %	64.73 % | 64.84 %
RR	65.21 %	65.24 %	65.08 %	65.16 %	64.88 % | 65.04 %
SVM	65.00 %	65.20 %	64.33 %	64.76 %	64.53 % | 64.69 %
RF	64.47 %	64.84 %	63.18 %	64.00 %	64.34 % | 64.65 %
KNN	63.89 %	63.84 %	64.00 %	63.92 %	64.50 % | 64.34 %
AB	63.67 %	64.76 %	64.31 %	64.53 %	64.82 % | 64.64 %
NB	63.61 %	63.78 %	63.55 %	63.67 %	63.91 % | 63.74 %

P-Accuracy: Accuracy with the Top 15, and 30 Features.

has a relatively higher precision (65.38 % against 65.32 %), suggesting that it makes fewer incorrect predictions of wins. The tie-breaker is in the F1-Score metric; the STACK algorithm has this metric marginally better than LR (65.68 % vs. 65.62 %), reflecting a significantly better balance between precision and recall.

RR and BAG are also excellent candidates in terms of predicting NBA results, with percentages of 65.41 % and 65.40 % respectively. The fact that BAG has a higher recall of close to 1 % indicates that it is more effective at capturing real wins, which is reflected in the F1-Score parameter.

MLP shows a decrease of more than 0.5 % in its accuracy compared to the previous measurement. More complex models such as neural networks may have greater capacity for non-linear and complex data relationships. However, this also means that they are more susceptible to overfitting when there are too many irrelevant or noisy features for the algorithm. The MLP stands out for its high recall of 69.61 %, beating even algorithms with higher accuracy. This is because there is a kind of trade-off with precision; this parameter, which stands at 63.13 %, suggests that MLP makes more predictions of wins, which is why it has such a high recall.

SVM and XGB also have percentages above 65.00 %, but these are two very different cases. SVM has slightly increased its accuracy concerning the Top 50 Features, which indicates the algorithm's ability to deal with high dimensionality. While XGB is already declining, this is because the features introduced are no longer very relevant to the algorithm, which leads to a drop in accuracy.

6.1.5 Top 100 Features

In Figure 6.5, the Top 100 Features are displayed. It is worth noting that some of these features may not be particularly relevant to the average NBA viewer. Additionally, certain parameters exhibit minimal importance to the "Target" variable, which could potentially impact the accuracy of certain algorithms due to the challenges in identifying relationships between variables.

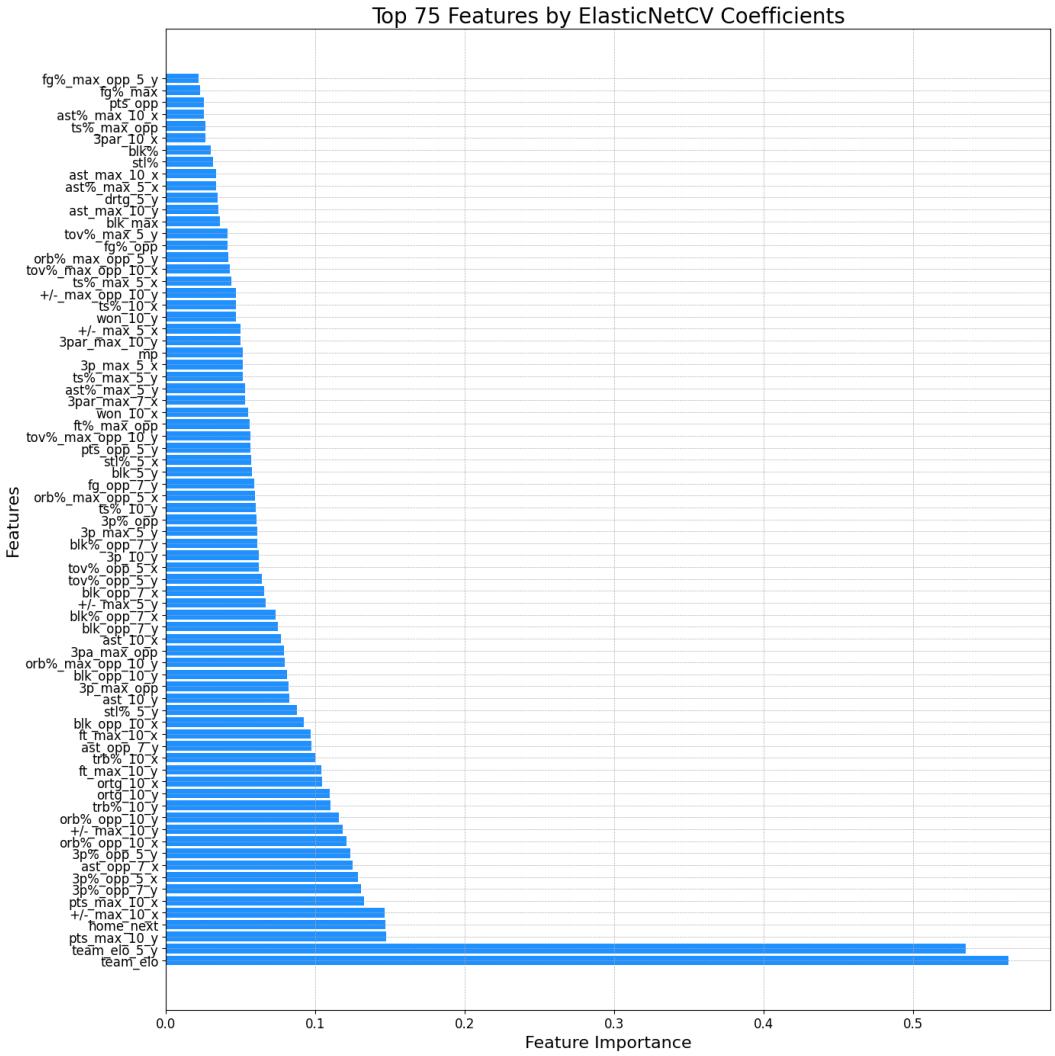


Figure 6.4: Top 75 Features by Feature Selector

Table 6.6: Table with the Metrics of the Top 75 Features

Algorithm	Accuracy	Precision	Recall	F1-Score	P-Accuracy (Top 15|30|50)
STACK	65.50 %	65.32 %	66.03 %	65.68 %	64.98 % | 65.04 % | 65.28 %
LR	65.50 %	65.38 %	65.87 %	65.62 %	65.01 % | 65.04 % | 65.27 %
RR	65.41 %	65.28 %	65.83 %	65.55 %	64.88 % | 65.04 % | 65.21 %
BAG	65.40 %	65.24 %	65.91 %	65.57 %	65.06 % | 65.18 % | 65.39 %
SVM	65.13 %	65.02 %	65.46 %	65.23 %	64.53 % | 64.69 % | 65.00 %
XGB	65.08 %	65.13 %	64.86 %	65.00 %	64.73 % | 64.84 % | 65.22 %
MLP	64.96 %	63.13 %	69.61 %	66.21 %	64.83 % | 65.07 % | 65.49 %
AB	64.77 %	64.80 %	63.63 %	64.71 %	64.82 % | 64.64 % | 63.67 %
RF	64.00 %	63.88 %	64.40 %	64.14 %	64.34 % | 64.65 % | 64.47 %
KNN	63.83 %	63.53 %	64.88 %	64.20 %	64.50 % | 64.34 % | 63.89 %
NB	63.13 %	62.67 %	64.92 %	63.77 %	63.91 % | 63.74 % | 63.61 %

P-Accuracy: Accuracy with the Top 15, 30 and 50 Features.

Table 6.7 shows some difficulty in reaching highest accuracy values when increasing to 100 features. It is often observed that algorithms may experience a decrease in

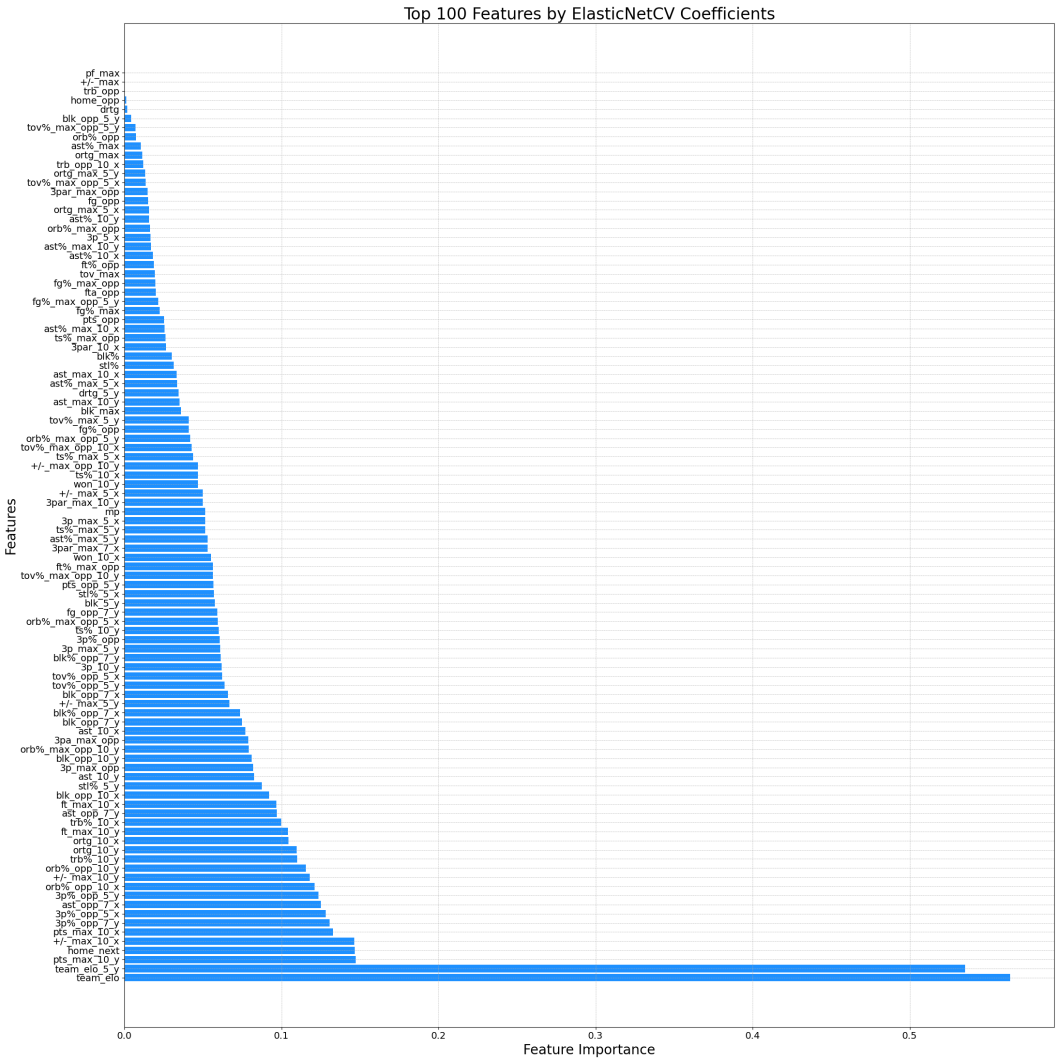


Figure 6.5: Top 100 Features by Feature Selector

performance, with accuracy increasing up to a certain number of features and then tapering off. This can happen due to various factors such as the relevance of the features; initially, adding more features usually includes relevant information that helps the model to better capture the patterns in the data, increasing accuracy. This happens up to a certain point, after which additional features can be irrelevant or redundant, introducing noise instead of useful information. This noise can confuse the model and reduce its ability to generalize.

BAG turned out to be the algorithm with the highest accuracy, although there was a decrease from 64.40 % to 65.37 %. MLP is an algorithm that stands out for its high recall, indicating an excellent ability to identify real wins. However, the fact that it has lower precision suggests that the algorithm generates more false positives. This trade-off can be advantageous in contexts where identifying wins is critical and penalizing false positives is less of a problem.

Overall, it can be seen that most of the algorithms that performed relatively well are falling with this number of features, which means that the Top 75 Features is ideal for most of them.

Table 6.7: Table with the Metrics of the Top 100 Features

Algorithm	Accuracy	Precision	Recall	F1-Score	P-Accuracy (Top 15|30|50|75)
BAG	65.37 %	65.18 %	65.93 %	65.56 %	65.06 % | 65.18 % | 65.39 % | 65.40 %
LR	65.33 %	65.23 %	65.63 %	65.43 %	65.01 % | 65.04 % | 65.27 % | 65.50 %
RR	65.14 %	65.02 %	65.50 %	65.26 %	64.88 % | 65.04 % | 65.21 % | 65.41 %
STACK	65.10 %	64.80 %	66.04 %	65.42 %	64.98 % | 65.04 % | 65.28 % | 65.50 %
XGB	65.07 %	65.03 %	65.15 %	65.09 %	64.73 % | 64.84 % | 65.22 % | 65.08 %
SVM	65.01 %	64.89 %	65.37 %	65.13 %	64.53 % | 64.69 % | 65.00 % | 65.13 %
MLP	64.87 %	63.13 %	69.61 %	66.21 %	64.83 % | 65.07 % | 65.49 % | 64.96 %
AB	64.73 %	64.73 %	64.69 %	64.71 %	64.82 % | 64.64 % | 63.67 % | 64.77 %
RF	63.99 %	63.88 %	64.40 %	64.14 %	64.34 % | 64.65 % | 64.47 % | 64.00 %
KNN	62.17 %	63.53 %	64.88 %	64.20 %	64.50 % | 64.34 % | 63.89 % | 63.83 %
NB	62.72 %	62.22 %	64.72 %	63.44 %	63.91 % | 63.74 % | 63.61 % | 63.13 %

P-Accuracy: Accuracy with the Top 15, 30, 50 and 75 Features

Figure 6.6 represents the performance of the top algorithms over the variation in features. It can be seen that the accuracy increases up to a certain number of features (in most algorithms 75 features, in MLP 50 features). This proves that adding new features, even if they are important to the feature selection algorithm, will only be relevant up to a certain point. After that, these features may not contribute relevant information or the algorithm may overfitting and cause a decrease in accuracy.

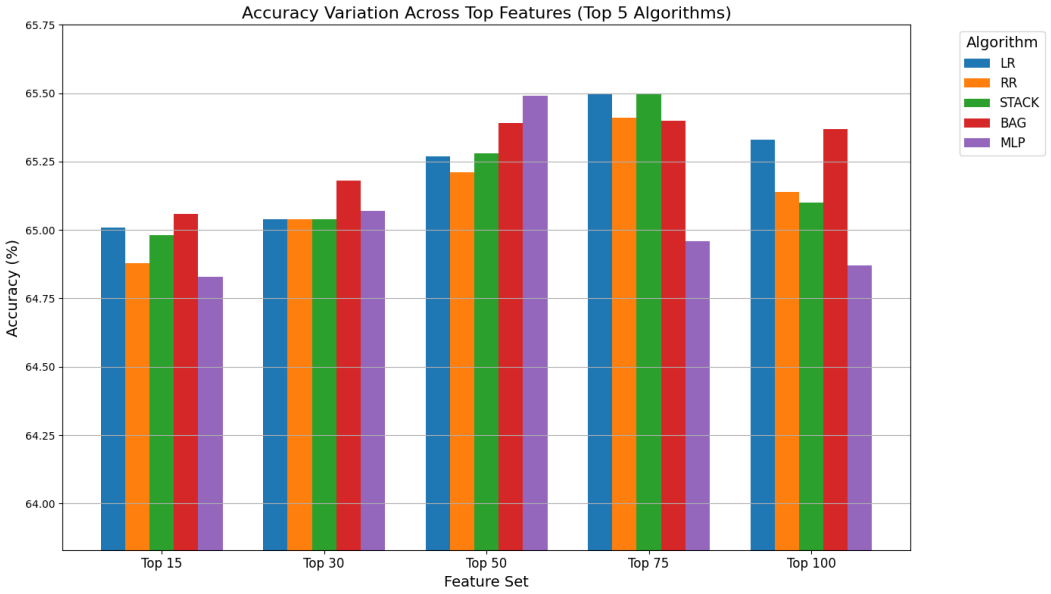


Figure 6.6: Accuracy of the Top 5 Machine Learning Algorithms by Varying the Features

6.1.6 Analysis by Season

To understand the accuracy of this algorithm in each season contained in the dataset, from 2016 to 2024, a study was carried out to try to understand its variation during each season. The algorithm chosen was the Stacking Classifier with 75 features because it was the one that obtained the best performance. This algorithm stands out compared to Logic Regression, also with 75 features, because it had a better recall and F1-Score.

Figure 6.7 shows the accuracy obtained over the seasons. It can be seen that the seasons with the best levels of accuracy are 2017/2018 and 2018/2019. A slightly sharp decline is observed in the following seasons and may be justified by COVID-19. The 2019/2020 and 2021/2022 seasons were highly affected by the pandemic that affected the whole world. The 2019/2020 season was stopped halfway through and resumed months later in a kind of resort where the teams that had a chance of making the playoffs gathered to finish the season. The result was that there were no spectators in the stands and all the games were played in the same place, which meant that the home factor was eliminated.

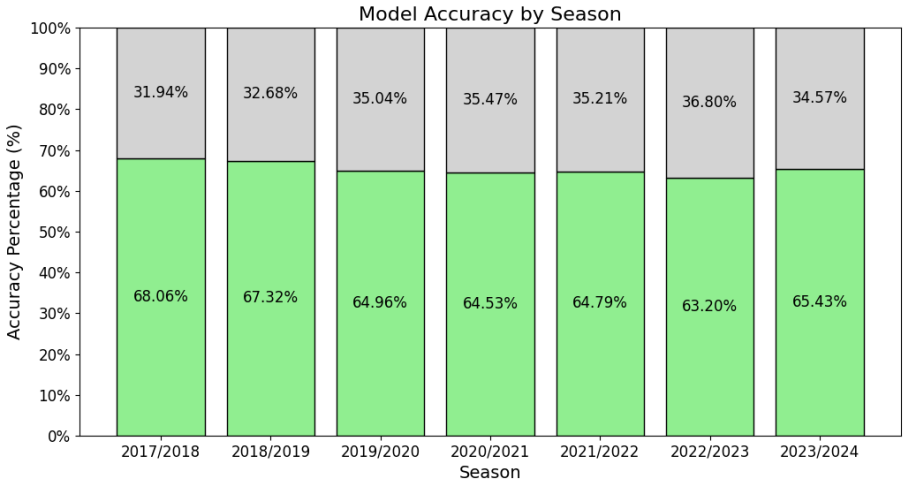


Figure 6.7: Accuracy by Season

Statistically, NBA teams win 58 % of the games they play at home, which poses a significant limitation for the machine learning algorithm. The pandemic also severely affected the 2020/2021 and 2021/2022 seasons. In the first season, regular-season games were reduced from 82 to 72. Despite playing in their arenas, teams did not have fans, which diminished the home-court advantage. Unvaccinated players faced various restrictions on their ability to perform, as some states did not allow these athletes to participate in certain events. When one player contracted COVID-19, several other players on the same team had to quarantine, forcing teams to use players who were not part of the regular rotation.

In the 2021/2022 season, the NBA returned to 82 games in the regular season, but COVID-related protocols continued to have an impact. Fans began to return to the stadium, but full capacity was not allowed. In the 2022/2023 season, everything returns to normal, but the ability to predict this season is severely affected by previous seasons in which there was a pattern of abnormality in the world and the league.

In the 2023/2024 season, the league's predictive capacity increases, and statistics such as the home factor begin to gain the right weight for machine learning algorithms.

Figure 6.8 indicates the confusion matrix of the Stacking Classifier algorithm with the Top 75 features, where it is possible to see that its success is slightly greater in victories than in defeats. Although the difference is not significant, this may be due to the features chosen by ElasticNETCV being more informative or discriminative for predicting wins than losses. This could mean that the patterns associated with victories are easier for the model to identify.

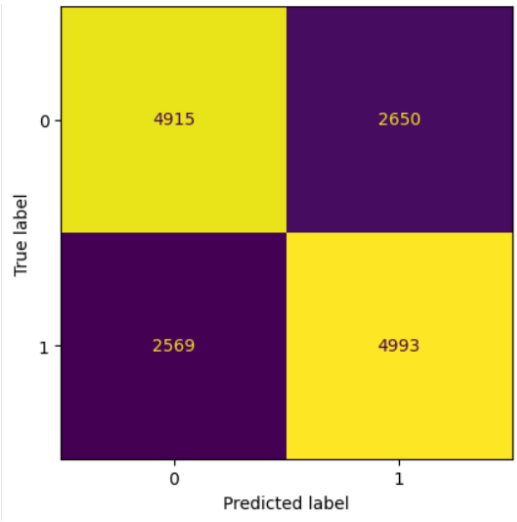


Figure 6.8: Confusion Matrix

6.1.7 Season 2023/2024

Upon careful examination, it was observed that although the 2023/2024 season contains the most data for training, it does not necessarily yield the highest accuracy. Subsequently, a comprehensive study was initiated to evaluate the top five ML and two DL learning algorithms to understand if these algorithms could be effective in better predicting the current NBA season.

The parameters employed for the DL models are presented in Table 6.9. In the DL algorithms, due to their randomness in certain parameters and as the objective was to obtain a precise study, ten attempts were made for LSTM and CNN. In

these ten attempts, the different feature variations were evaluated and the variation that showed the best results was chosen. Subsequently, the averages of these ten attempts were calculated and compared with those of the other ML algorithms.

Table 6.8: Table of DL Algorithm Parameters

Algorithm	**Parameter Chosen**
LSTM	LSTM units (1st layer) = 200 Dropout rate (1st layer) = 0.5 LSTM units (2nd layer) = 64 Dropout rate (2nd layer) = 0.2 Dense units = 32 Learning rate = 0.0003
CNN	Conv1D filters (1st layer) = 32 Kernel size (1st layer) = 2 Dropout rate (1st layer) = 0.5 Conv1D filters (2nd layer) = 64 Kernel size (2nd layer) = 2 Dropout rate (2nd layer) = 0.5 Dense units (1st dense layer) = 64 Dropout rate (1st dense layer) = 0.2 Dense units (2nd dense layer) = 32 Learning rate = 0.0003

Although the LSTM has a slightly higher dispersion than the CNN due to its higher standard deviation, it ends up obtaining better results in predicting the results of the 2023/2024 season as shown in Table 6.9.

Table 6.9: DL Algorithm Metrics

Algorithm	**Min**	**Max**	**Average**	**Standard Deviation**
LSTM (Top 75)	64.46 %	65.62 %	65.26 %	0.31 %
CNN (Top 30)	64.53 %	65.25 %	64.97 %	0.25 %

Table 6.10 shows that the STACK and MLP algorithms, with 75 and 50 features respectively, have the best percentages of accuracy. However, STACK has an advantage because it has a higher recall and F1-Score, which suggests a better balance between identifying true positives and the accuracy of positive predictions. In a context applied to the NBA, it is better at capturing the majority of wins, even if it accepts a small number of wrong predictions. The other algorithms analyzed have slightly lower prediction capabilities than STACK.

LSTM achieved highly positive metrics, competing closely with the two machine learning algorithms that had the highest prediction accuracies. By obtaining a prediction percentage of 65.26 % and in one attempt obtaining better accuracy values than STACK, it managed to reach 65.62 %. This is due to the weights of

the neural networks, which are initialized randomly, and also the way in which the learning rate is adjusted over time can vary between runs.

Table 6.10: Table with Metrics for the 2023/2024 Season

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**
STACK (Top 75)	65.43 %	65.12 %	66.55 %	65.83 %
MLP (Top 50)	65.43 %	65.72 %	64.61 %	65.16 %
LSTM (Top 75)	65.26 %	65.62 %	64.21 %	64.89 %
LR (Top 75)	65.13 %	65.24 %	64.85 %	65.05 %
RR (Top 75)	65.07 %	65.24 %	64.61 %	64.92 %
CNN (Top 30)	64.97 %	65.09 %	64.76 %	64.89 %
BAG (Top 75)	64.89 %	64.82 %	65.21 %	65.02 %

In light of the challenges posed by irregular patterns arising during the pandemic, it was decided to exclude the specific time periods representing the peak of COVID-19 activity (2019/2020, 2020/2021, and 2021/2022 seasons) from the training set. This was done in order to ensure the effectiveness of the ML algorithms.

Table 6.11 shows the results, where we can see both the increase and decrease in accuracy by the algorithms. MLP achieves the highest accuracy of 65.74%, and even the highest for the 2023/2024 season. MLP appears to be more robust to the noise present in COVID-19 data. Removing this data allows the model to focus on more consistent and predictable patterns. Removing atypical data makes the dataset more homogeneous and consistent, which leads to increased accuracy. The BAG algorithm also showed improvements in removing data affected by the pandemic.

The other ML and DL algorithms showed slight drops in accuracy. Although the seasons affected by the pandemic are different in many ways from the others, they have relevant information about the game's evolution. These three years out of the eight used for training are already significantly reduced, as certain algorithms rely on a large volume of data to capture complex variations. Also, because prediction methods are based on time series, erasing a period of history may not be appropriate, leading to the removal of accuracy from these algorithms.

Table 6.11: Table with Metrics for the 2023/2024 without the Seasons affected by COVID

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**C-Accuracy**
MLP (Top 50)	65.74 %	65.51 %	66.55 %	66.03 %	65.43 %
STACK (Top 75)	65.31 %	65.30 %	65.45 %	65.38 %	65.43 %
BAG (Top 75)	65.25 %	65.03 %	66.27 %	65.64 %	64.89 %
LR (Top 75)	64.95 %	64.68 %	65.94 %	65.31 %	65.13 %
RR (Top 75)	64.88 %	64.71 %	65.58 %	65.14 %	65.07 %
LSTM (Top 75)	64.80 %	65.03 %	64.24 %	64.59 %	65.26 %
CNN (Top 30)	64.71 %	65.17 %	63.38 %	64.24%	64.97 %

C-Accuracy: Accuracy with the COVID seasons in the training set.

6.2 WNBA

A comparative study mirroring that of the NBA has been undertaken for the WNBA. However, it is important to note that the dataset for the WNBA is considerably smaller and lacks one season compared to the NBA, as the 2024 season has not yet started.

Although the WNBA dataset is smaller than the NBA dataset, it was clear that it was necessary to increase the number of iterations for ElasticNetCV not to reach the limit on the number of iterations. Iterations in the context of ElasticNetCV refer to the number of steps the optimization algorithm takes to find the model that minimizes the cost function. The reasons for ElasticNetCV needing more iterations and being more computationally exhaustive for the WNBA dataset are as follows:

- **Less Information**: In smaller datasets, there is less data available to inform parameter estimation. This can make it difficult to identify clear patterns and, consequently, require more iterations for the algorithm to find a stable minimum of the cost function.
- **Greater Variability**: With less data, the variability in the estimators can be greater. This means that coefficient updates can be less accurate, resulting in an optimization process that needs more steps to stabilize and converge.
- **Less Effective Regularization**: In a smaller dataset, the effects of regularization can be less pronounced. Regularization helps to avoid overfitting and simplifies the model, but with less data, the balance between adjustment and regularization can be more difficult to achieve, requiring more iterations to find the optimal coefficients.

The features chosen by ElasticNetCV and Pearson's Correlation comprised 17 elements. Among these, the 7 and 17 features demonstrated varying levels of importance for the "Target" variable.

Table 6.12 presents the varied parameters and the selected values for the ML algorithms, analogous to the approach taken for the NBA model.

6.2.1 Top 7 Features

In the Top 7 Features shown in Figure 6.9, it can be seen that the Feature Engineer dominates this category, although the “fg” field goal is quite important. The field goal is a ratio between the number of baskets hit and missed. This metric tells us how effective the teams are at shooting the basket. As in the WNBA there are far fewer points per game than in the NBA, this indicates that there are fewer shots on target. Therefore, it becomes an essential parameter for team efficiency.

Table 6.12: Table with Algorithm Parameterization for the WNBA

Algorithm	Parameter Variation	Parameter Chosen
LR	1000 ≤ max iter ≤ 3000 1 ≤ verbose ≤ 5 0 ≤ random_state ≤ 100	max_iter = 2000 verbose = 2 random_state = 42
RR	0.1 ≤ alpha ≤ 5	alpha = 1
RF	50 ≤ n_estimator ≤ 500 1 ≤ max_deph ≤ 10 1 ≤ min_samples_split ≤ 20 1 ≤ min_samples_leaf ≤ 20 0 ≤ random_state ≤ 100	n_estimator = 100 max_deph = 2 min_samples_split = 15 min_samples_leaf = 10 random_state = 42
NB	0.1 ≤ priors ≤ 0.9	priors = 0.4; 0.6
KNN	3 ≤ n_neighbors ≤ 400	n_neighbors = 175
SVM	0.1 ≤ C ≤ 15 0 ≤ random_state ≤ 200	C = 1 random_state = 100
BAG	0 ≤ n_estimators ≤ 100 0 ≤ random_state ≤ 100	n_estimators = 11 random_state = 43
MLP	0 ≤ hidden_layer_size ≤ 100 100 ≤ max_iter ≤ 3000 0.0001 ≤ alpha ≤ 0.1	hidden_layer_size = 30; 15 max_iter = 2000 alpha = 0.001
AB	50 ≤ n_estimator ≤ 300 0.0001 ≤ learning_rate ≤ 0.1 0 ≤ random_state ≤ 100	n_estimators = 100 learning_rate = 0.09 random_state = 0
XGB	50 ≤ n_estimators ≤ 200 0.0001 ≤ learning_rate ≤ 0.1 1 ≤ max_depth ≤ 10 1 ≤ min_child_weight ≤ 10 0.01 ≤ gamma ≤ 1 0.1 ≤ subsample ≤ 1 0 ≤ colsample_bytree ≤ 2 0 ≤ random_state ≤ 100	n_estimators = 50 learning_rate = 0.05 max_depth = 2 min_child_weight = 0.05 gamma = 0.05 subsample = 0.8 colsample_bytree = 0.9 random_state = 42
STACK	Parameters used to test the other algorithms	RR RF KNN LR MLP

Table 6.13 shows the results of the ML algorithms with the Top 7 Features. MLP is the best algorithm for predicting WNBA results, as it has the highest percentages in all statistics. BAG also performs very competitively, with an accuracy of 67.27 %, which is slightly lower than MLP.

RF shows the lowest performance using the 7 best features selected by Elastic-NETCV, suggesting that it may not be the most suitable model for this dataset, which shows less variability compared to the dataset previously used for predicting NBA games.

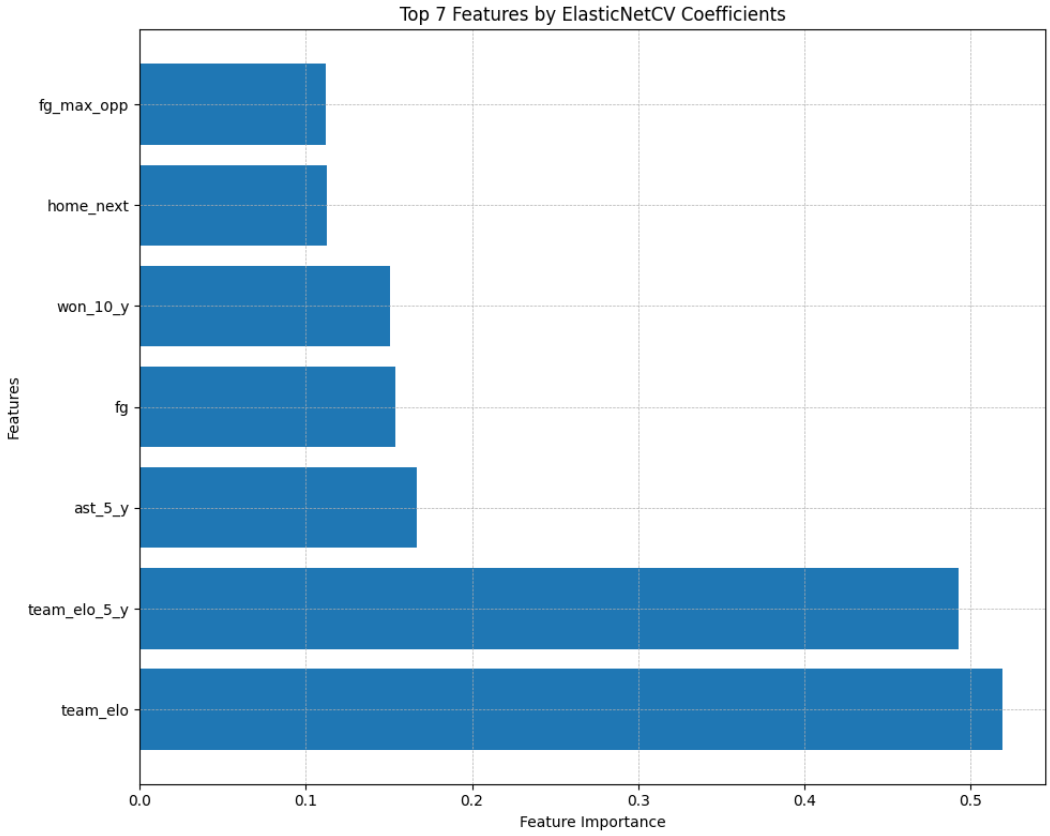


Figure 6.9: Top 7 Features by Feature Selector WNBA

Table 6.13: Table with the Metrics of the Top 7 Features

Algorithm	Accuracy	Precision	Recall	F1-Score
MLP	67.48 %	67.86 %	66.36 %	67.09 %
BAG	67.27 %	67.91 %	65.40 %	66.63 %
LR	67.16 %	67.72 %	65.50 %	66.59 %
STACK	67.00 %	67.62 %	65.18 %	66.38 %
RR	67.00 %	67.62 %	65.19 %	66.38 %
NB	66.52 %	68.18 %	61.90 %	64.89 %
XGB	66.31 %	67.07 %	64.02 %	65.51 %
SVM	65.68 %	65.81 %	65.19 %	65.49 %
KNN	65.52 %	65.12 %	66.77 %	65.93 %
AB	65.25 %	65.48 %	64.44 %	64.96 %
RF	64.36 %	65.70 %	60.00 %	62.72 %

6.2.2 Top 17 Features

For the Top 17 Features chosen by ElasticNetCV, no additional Feature Enginner features have been added, but more statistics from the previous game seem to have a greater impact, as can be seen in Figure 6.10. A metric related to the three-point shot is highlighted, which shows that in the WNBA this game strategy is not given as much importance as it is in the NBA.

Table 6.14 shows the results for the Top 17 Features. STACK ended up showing

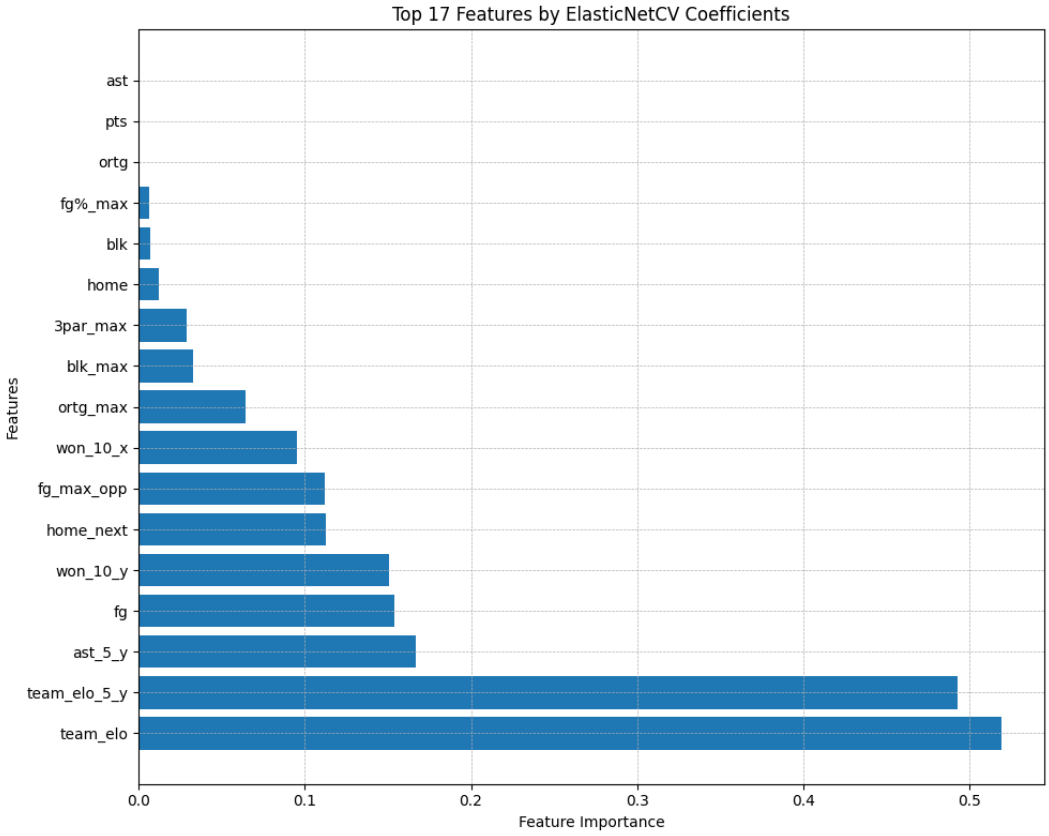


Figure 6.10: Top 17 Features by Feature Selector WNBA

an increase in accuracy to 67.32 % while also maintaining the highest precision, which means that the model is reliable in correctly identifying positive examples, such as wins while minimizing defeats.

The MLP shows a sharp drop from 67.48 % to 62.61 %, indicating what can be considered a "curse of dimensionality", a phenomenon where an increase in the number of features can lead to an exponential increase in the volume of the feature space, making it more difficult for the ML model to generalize well.

The SVM algorithm has demonstrated a commendable accuracy of 66.47 %. Additionally, it achieved the highest recall among all the algorithms, at 66.56 %. Recall measures the algorithm's ability to correctly identify all actual positive cases.

Figure 6.11 shows the variation in accuracy as the number of features changes. MLP suffers the most as the number of features increases, decreasing by almost 5%. The other algorithms maintain fairly close percentages, varying between 66.5% and 67.3%. The fact that there are not as many features as in the NBA does not allow for such a detailed study of the impact of the number of features.

Table 6.14: Table with the Metrics of the Top 17 Features

Algorithm	Accuracy	Precision	Recall	F1-Score	P-Accuracy (Top 7)
STACK	67.32 %	67.80 %	65.93 %	66.85 %	67.00 %
RR	67.11 %	67.77 %	65.19 %	66.45 %	67.00 %
LR	66.90 %	67.62 %	64.76 %	66.16 %	67.16 %
BAG	66.90 %	67.66 %	64.65 %	66.13 %	66.63 %
SVM	66.47 %	66.42 %	66.56 %	66.49 %	65.68 %
XGB	65.42 %	65.60 %	64.76 %	65.18 %	66.31 %
RF	65.31 %	65.59 %	64.34 %	64.96 %	64.36 %
AB	65.20 %	65.41 %	64.44 %	64.93 %	65.25 %
NB	65.10 %	65.31 %	64.34 %	64.82 %	66.52 %
MLP	62.61 %	63.90 %	57.88 %	60.74 %	67.48 %
KNN	60.55 %	61.25 %	57.35 %	59.23 %	65.52 %

P-Accuracy: Accuracy with the Top 7 Features.

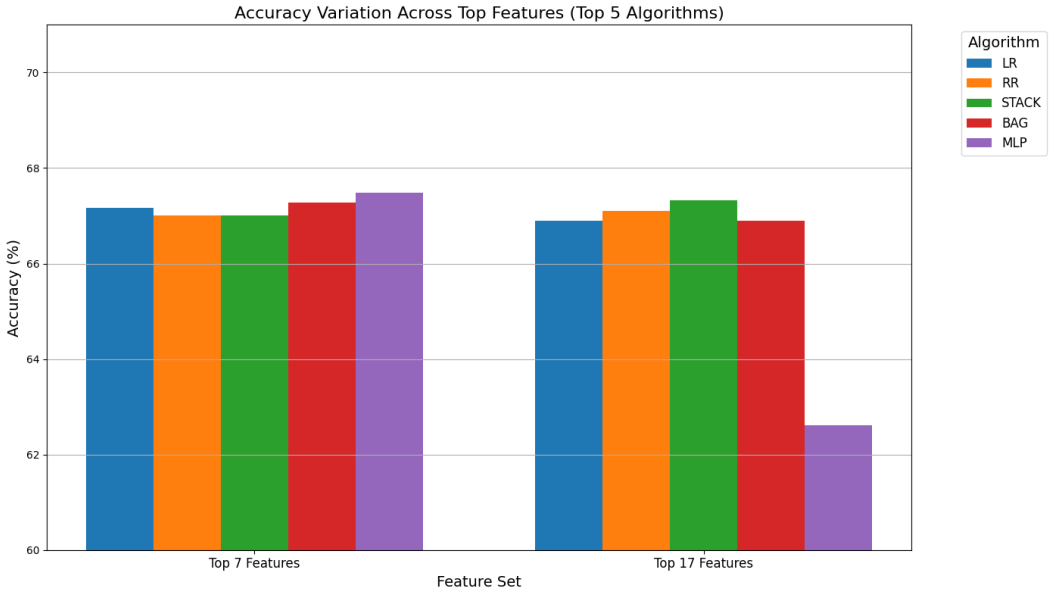


Figure 6.11: Accuracy of the Top 5 Machine Learning Algorithms by Varying the Features WNBA

6.2.3 Analysis by Season

A study was undertaken using the algorithm with the strongest predictive capacity to assess its performance across different seasons. In the context of the NBA, it has been noted that the pandemic had an impact on the league. The objective was to analyze and understand the extent of this impact on the WNBA.

Figure 6.12 shows the results of the MLP algorithm with the Top 7 Features in each WNBA season studied. It is possible to see a sharp decrease in the 2020 season at the very time that COVID-19 decelerated. As in the NBA, there have been several changes at all levels in the WNBA that are responsible for this decrease in percentage. In the 2020 season, the number of games in the regular season was cut from 36 games to 22, which means that there is less information. That season was

also played in a "bubble" at the IMG Academy in Bradenton, Florida to minimize the risk of contagion, very similar to what happened in the NBA. The "home_next" feature is one of the most important features for the feature selector algorithm, and the fact that there is no crowd and all the games are played on a neutral field means that one of the most important features no longer has an influence. Some players have also chosen not to participate due to health concerns, which means that the information provided may be less concise than this season.

The 2021 season will see a return to more normal percentages. The season has returned to 32 games, still low compared to the traditional 36. The matches were also held in the teams' usual stadiums, which means that the "home_next" feature is once again becoming more relevant, although the arenas are still limited to a certain number of spectators.

The pandemic had a huge impact on this league, just as it had on the NBA. The advantage of the WNBA is that it takes place exclusively in the summer, avoiding the more controversial periods experienced by other leagues due to COVID-19.

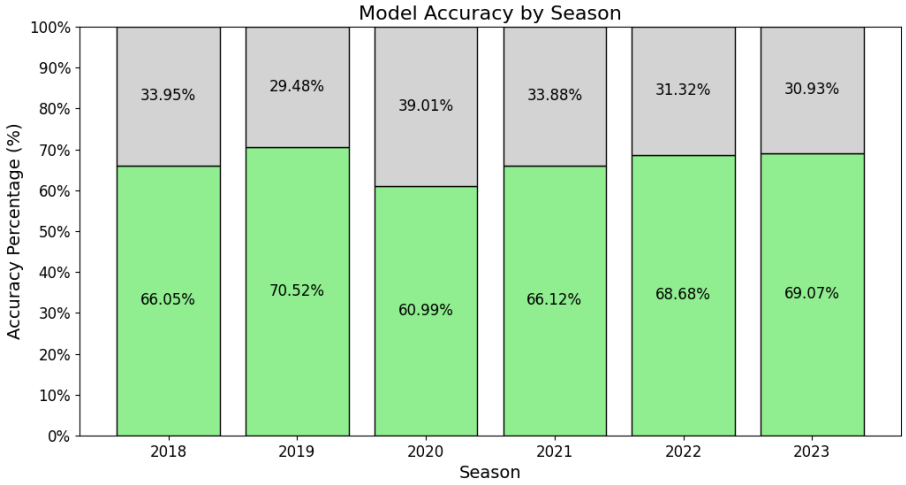


Figure 6.12: Accuracy by Season WNBA

Figure 6.13 displays the confusion matrix with MLP with the Top 7 features, showing that the algorithm is slightly better at identifying losses than wins. The difference is minimal, but unlike the NBA, the features that ElasticNetCV chose for the WNBA may relate more to losses, making them easier to identify.

6.2.4 Season 2023

A recent study was conducted on the WNBA for the data available from the 2023 season, which is the most current season for which data is available. All seasons before this are used for training purposes. The top 5 ML algorithms with the best previous performance and two DL techniques, LSTM and CNN, are used to test the

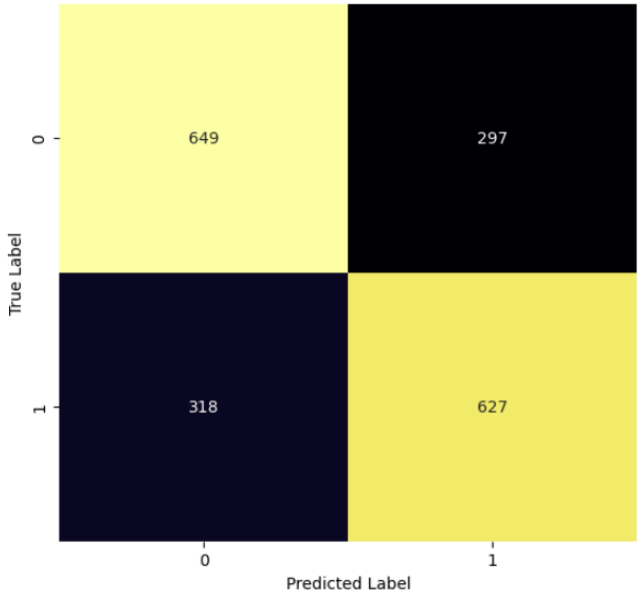


Figure 6.13: Confusion Matrix WNBA

capacity of these algorithms for this type of prediction. The parameters used for the WNBA DL models are the same as those used for the NBA. The set of features that obtained the best results was also used and ten attempts were made to obtain the average metrics of the DL algorithms.

Table 6.15 shows the metrics of the DL algorithms, in which we see a much higher standard deviation of CNN compared to LSTM, which can lead to more variations in the results.

Table 6.15: DL Algorithm Metrics WNBA

Algorithm	**Min**	**Max**	**Average**	**Standard Deviation**
LSTM (Top 7)	68.56 %	69.33 %	69.05 %	0.29 %
CNN (Top 17)	65.46 %	69.59 %	67.97 %	1.19 %

LR was the method that obtained the best classification for this season, as shown in Table 6.16. Despite having the same accuracy as BAG, LR stands out for having a better F1-Score, which indicates a better harmonic mean relationship between precision and recall.

The LSTM algorithm obtained excellent percentages of predictions, and although the result was 69.05 %, in the ten attempts made there were values which equaled the 69.33 % of the top ML algorithms. Despite the fact that CNN had the worst average accuracy in its ten attempts, it ended up with a value of 69.59 % in its best attempt, which beats all the ML algorithms. This is because the weights of neural networks are usually initialized randomly. Each run starts with a different

set of weights, which can lead to different training paths and, consequently, different performance levels.

Table 6.16: Table with the Metrics for the 2023 season

Algorithm	Accuracy	Precision	Recall	F1-Score
LR (Top 7)	69.33 %	70.88 %	66.15 %	68.44 %
BAG (Top 7)	69.33 %	70.65 %	66.67 %	68.06 %
MLP (Top 7)	69.07 %	70.05 %	67.18 %	68.59 %
LSTM (Top 7)	69.05 %	70.00 %	67.33 %	68.61 %
RR (Top 17)	68.81 %	69.68 %	67.18 %	68.41 %
STACK (Top 17)	68.30 %	69.35 %	66.15 %	67.72 %
CNN (Top 17)	67.78 %	69.14 %	64.87 %	66.92 %

Table 6.17 shows the results with the 2020 season removed from the training set to understand the impact that this season, which was severely affected by the pandemic, had on the algorithm predictions. There were some increases, especially in the RR algorithm, which reached 69.59 % and was previously at 68.81 %.

The change in metrics concerning the removal of the season affected by COVID-19 is related to the nature of the data and the way the algorithms learn from this data, as it is possible to observe both positive and negative changes in the metrics of the algorithms studied.

Table 6.17: Table with Metrics for the 2023 Season without the Season Affected by COVID

Algorithm	Accuracy	Precision	Recall	F1-Score	C-Accuracy
RR (Top 17)	69.59 %	71.27 %	66.15 %	68.62 %	68.81 %
LSTM (Top 7)	69.38 %	70.23 %	67.90 %	69.02 %	69.05 %
LR (Top 7)	69.33 %	70.00 %	68.21 %	69.10 %	69.33 %
STACK (Top 17)	69.33 %	70.65 %	66.67 %	68.60 %	68.30 %
BAG (Top 7)	69.07 %	69.84 %	67.69 %	68.75 %	69.33 %
MLP (Top 7)	68.30 %	70.02 %	64.10 %	67.02 %	69.07 %
CNN (Top 17)	67.99 %	69.28 %	65.28 %	67.20 %	67.78 %

C-Accuracy: Accuracy with the COVID season in the training set.

6.3 Comparison of Results

An assessment was undertaken to compare the outcomes with the studies referenced in Chapter 2. The results obtained in this thesis and the articles covered are presented in Table 6.18.

The prediction percentages of this thesis are slightly lower than two of these studies, which may be due to a variety of factors. The quality of the dataset is very important to allow ML algorithms to capture relationships that may be important for making predictions; different datasets with different statistical parameters will lead to different results. The features created by Feature Engineering are extremely

Table 6.18: NBA Results Comparison

Paper	**Algorithm**	**Accuracy**
[60]	DecisionTreeClassifier	68.60%
[59]	RF	67.15 %
This Thesis	STACK	65.50 %
[18]	NB	65.00 %

important because they facilitate the analysis of information for ML models. The feature selector employed in this dissertation primarily selects these features. Also, and most importantly, it is related to the periods studied. The articles analyzed refer to previous seasons, with the most recent being up to the 2020/2021 season. The NBA is a league that is always evolving and there may be more competitive or less competitive seasons, which will certainly influence the ability to predict. COVID-19 also had a significant impact on the seasons during which it was present, and its effects may still be felt at various levels within the league today.

How the training and test data are divided is also very important. In the study for this dissertation, a different approach was tried: dividing the dataset by seasons and using the seasons before the one being tested for training. The approach taken by these articles involves dividing the dataset into a training set and a testing set. This allows for a portion of a season to be utilized for testing within the training set, thereby potentially leading to improved results as the training incorporates patterns specific to that season.

An additional consideration worth noting is the use of cross-validation in certain articles for feature selection. However, it may not be suitable for this type of study due to the random division of the dataset, which could result in future data being used in the training set and past data in the test set. As a result, the features selected may not align with realistic criteria, considering that the real world does not operate in this manner.

The highest accuracy obtained in this thesis was 65.50 %, which is in line with the results that have been observed in the scientific community.

As for the WNBA, there are no studies that can serve as a basis for comparing our results. The highest prediction result obtained over the dataset was 67.48 % with MLP, which beats the results obtained on the NBA dataset in this dissertation. While we have fewer games and slightly fewer statistics for the WNBA, it is notable that the algorithm finds it easier to capture relationships between past data and predict future data in this context. This is perhaps due to the heightened level of competition in the NBA compared to its women's league.

Chapter 7

Conclusions

This chapter will address the findings of the work and propose potential areas for future improvement.

7.1 Discussion

In summary, this thesis has effectively showcased the capability of ML and DL algorithms to forecast the outcomes of basketball games within the NBA and WNBA leagues. Despite the existence of datasets for the NBA, WebScrapping was used to obtain a more up-to-date dataset so that it could be compared with the results of other articles on predicting NBA games using ML. The league has been significantly impacted by the pandemic, resulting in complex patterns that are challenging to interpret. The best result obtained for this league was 65.50 %, which is very competitive according to the studies carried out and proves the unpredictability and competitiveness of the best basketball league in the world.

For the WNBA, WebScapping is a necessary tool, as there are no datasets with relevant information for this study. While there are no other results available for comparison, achieving a 67.48 % success rate in predicting the outcomes of sporting events is considered a notable accomplishment, given the volatility of such events. The WNBA proves to be less unpredictable and less competitive than the NBA, which is to be expected as it is a league that is still several years behind in terms of both financial and human resources.

In conclusion, the initial objective has been successfully attained. Two distinct ML models have been developed: one for forecasting the outcomes of NBA games and the other for predicting the outcomes of WNBA games, both yielding highly satisfactory results.

7.2 Future Work

For future work, several parameters could be explored over time to make the study more up-to-date and possibly improve the results in terms of the accuracy of the two models.

Hyperparameter optimization techniques such as GridSearch and RandomSearch could be used to find the ideal combination of hyperparameters and features to maximize the model's performance. However, these two techniques are computationally intensive, which could require other types of computational resources that are not available at the moment. The datasets used were obtained in mid-March, but it would also be possible in the future to obtain more current datasets with the 2023/2024 season practically complete and some games from the 2024 WNBA season. The addition of a database indicating the available players and their impact could be very useful since this is a five-on-five sport and one really good player can have a huge impact on the game. Exploring new feature engineering techniques used in other sports and applying them in this context could potentially enhance the results. Exploring DL algorithms in more depth will also be an excellent future development, as the aim of this thesis was to demonstrate that these algorithms can be viable alternatives to the ML algorithms studied.

References

[1] "What are the most watched sports in the world?." Available at https://blog.gwi.com/chart-of-the-day/worlds-most-popular-sports/. (Last accessed in 22/03/2024). [Cited on pages ix and 2]

[2] "What is the nba play-in tournament, how does it work, when is it and which teams will participate in it?." Available at https://www.marca.com/en/more-sports/2021/05/17/60a254e0268e3e9f6e8b45e4.html/. (Last accessed in 22/03/2024). [Cited on pages ix and 3]

[3] "Nba advanced stats." Available at https://www.nba.com/stats/teams/. (Last accessed in 22/03/2024). [Cited on pages ix and 4]

[4] R. R. Nadikattu, "Implementation of new ways of artificial intelligence in sports," *Journal of Xidian University*, vol. 14, no. 5, pp. 5983–5997, 2020. [Cited on page 9]

[5] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, 2018. [Cited on page 9]

[6] "What is artificial intelligence (ai)?." Available at https://www.ibm.com/topics/artificial-intelligence/. (Last accessed in 22/03/2024). [Cited on page 10]

[7] "The history of artificial intelligence." Available at https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/. (Last accessed in 22/03/2024). [Cited on page 10]

[8] "What is the history of artificial intelligence (ai)?." Available at https://www.tableau.com/data-insights/ai/history/. (Last accessed in 22/03/2024). [Cited on pages 10 and 11]

[9] "History of ai." Available at www.humanoid.waseda.ac.jp/booklet.html. (Last accessed in 22/03/2024). [Cited on pages ix and 10]

[10] "Artificial intelligence history." Available at Artificial Intelligence History. (Last accessed in 22/03/2024). [Cited on page 10]

[11] "Artificial intelligence." Available at `https://www.researchgate.net/artificialintelegence`. (Last accessed in 22/03/2024). [Cited on pages ix and 11]

[12] "Semi-supervised learning, explained with examples." Available at `https://www.altexsoft.com/blog/semi-supervised-learning/`. (Last accessed in 22/03/2024). [Cited on pages ix, 12, and 20]

[13] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *2016 3rd international conference on computing for sustainable global development (INDIACom)*, pp. 1310–1315, Ieee, 2016. [Cited on pages ix and 13]

[14] "What are classification problems?." Available at `https://www.educative.io/answers/what-are-classification-problems`. (Last accessed in 28/03/2024). [Cited on page 13]

[15] D. Maulud and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 140–147, 2020. [Cited on page 14]

[16] "Linear regression in machine learning." Available at `https://www.javatpoint.com/linear-regression-in-machine-learning`. (Last accessed in 28/03/2024). [Cited on pages ix and 14]

[17] "What is linear regression?." Available at `https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/`. (Last accessed in 28/03/2024). [Cited on page 14]

[18] M. Houde, "Predicting the outcome of nba games," 2021. [Cited on pages ix, 15, 34, 36, and 108]

[19] P. A. Flach and N. Lachiche, "Naive bayesian classification of structured data," *Machine learning*, vol. 57, pp. 233–269, 2004. [Cited on page 17]

[20] P. Viswanath and T. H. Sarma, "An improvement to k-nearest neighbor classifier," in *2011 IEEE Recent Advances in Intelligent Computational Systems*, pp. 227–231, IEEE, 2011. [Cited on page 17]

[21] Y. Xu, Q. Zhu, Z. Fan, M. Qiu, Y. Chen, and H. Liu, "Coarse to fine k nearest neighbor classifier," *Pattern recognition letters*, vol. 34, no. 9, pp. 980–986, 2013. [Cited on page 17]

[22] "O que é e como funciona o algoritmo randomforest." Available at `https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-randomforest/`. (Last accessed in 22/03/2024). [Cited on page 18]

[23] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016. [Cited on page 18]

[24] "What is a random forest?." Available at `https://www.spotfire.com/glossary/what-is-a-random-forest`. (Last accessed in 22/03/2024). [Cited on pages ix and 18]

[25] "What is a support vector machine? working, types, and examples." Available at `https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/`. (Last accessed in 22/03/2024). [Cited on page 19]

[26] A. C. Lorena and A. C. De Carvalho, "Uma introdução às support vector machines," *Revista de Informática Teórica e Aplicada*, vol. 14, no. 2, pp. 43–67, 2007. [Cited on page 19]

[27] "Gradientboostingclassifier." Available at aklearn. (Last accessed in 29/03/2024). [Cited on page 19]

[28] "What is unsupervised learning?." Available at `https://www.ibm.com/br-pt/topics/unsupervised-learning`. (Last accessed in 22/03/2024). [Cited on page 20]

[29] "What is unsupervised learning?." Available at `https://cloud.google.com/discover/what-is-unsupervised-learning`. (Last accessed in 22/03/2024). [Cited on page 20]

[30] Y.-F. Li and D.-M. Liang, "Safe semi-supervised learning: a brief introduction," *Frontiers of Computer Science*, vol. 13, pp. 669–676, 2019. [Cited on page 20]

[31] "Five machine learning types to know." Available at `https://www.ibm.com/blog/machine-learning-types/`. (Last accessed in 22/03/2024). [Cited on pages 20 and 22]

[32] "Types of machine learning." Available at `https://www.geeksforgeeks.org/types-of-machine-learning/`. (Last accessed in 22/03/2024). [Cited on pages ix, 21, and 24]

[33] "Self-supervised learning and its applications." Available at `https://neptune.ai/blog/self-supervised-learning`. (Last accessed in 22/03/2024). [Cited on page 22]

[34] "What is a neural network?." Available at `https://www.ibm.com/topics/neural-networks`. (Last accessed in 22/03/2024). [Cited on pages ix, 22, and 24]

[35] "Artificial neural network." Available at `https://www.databricks.com/glossary/artificial-neural-network`. (Last accessed in 22/03/2024). [Cited on page 22]

[36] "How many neurons for a neural network?." Available at YourDataTeacher. (Last accessed in 22/03/2024). [Cited on pages ix and 23]

[37] "Introduction to artificial neural networks." Available at Artificial-Neural-Networks. (Last accessed in 22/03/2024). [Cited on page 23]

[38] D. Durstewitz, G. Koppe, and A. Meyer-Lindenberg, "Deep neural networks in psychiatry," *Molecular psychiatry*, vol. 24, no. 11, pp. 1583–1598, 2019. [Cited on page 24]

[39] "What are convolutional neural networks?." Available at IBM. (Last accessed in 22/03/2024). [Cited on page 24]

[40] "Convolutional neural networks, explained." Available at Towards-Science. (Last accessed in 22/03/2024). [Cited on pages ix and 25]

[41] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021. [Cited on page 25]

[42] "Introduction to feed-forward neural network in deep learning." Available at Feed-Foward. (Last accessed in 22/03/2024). [Cited on page 25]

[43] P. Sharma, "Introduction to feed-forward neural network in deep learning." Available at `https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/`, 2024. (Last accessed in 24/04/2024). [Cited on pages ix and 26]

[44] "Introduction to recurrent neural network." Available at `https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/`. (Last accessed in 22/03/2024). [Cited on pages ix and 26]

[45] S. Grossberg, "Recurrent neural networks," *Scholarpedia*, vol. 8, no. 2, p. 1888, 2013. [Cited on page 27]

[46] "Understanding lstm networks." Available at `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. (Last accessed in 22/03/2024). [Cited on page 27]

[47] "What is lstm? introduction to long short-term memory." Available at Analyticsvidhya. (Last accessed in 22/03/2024). [Cited on pages ix, 27, and 29]

[48] "What is deep learning?." Available at `https://www.oracle.com/br/artificial-intelligence/machine-learning/what-is-deep-learning/`. (Last accessed in 22/03/2024). [Cited on page 29]

[49] "What is deep learning?." Available at `https://www.ibm.com/topics/deep-learning`. (Last accessed in 22/03/2024). [Cited on pages 29 and 30]

[50] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6, IEEE, 2018. [Cited on page 30]

[51] "Machine learning vs deep learning." Available at `https://www.linkedin.com/posts/prof-tarig-mohamed-ahmed-55947499_activity-6428853875553374208-Vq2P/?trk=public_profile_like_view`. (Last accessed in 18/05/2024). [Cited on pages ix and 30]

[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Cited on page 30]

[53] "Web scraping." Available at Imperva. (Last accessed in 22/03/2024). [Cited on page 31]

[54] "How to scrape data from a website." Available at `https://www.techtarget.com/whatis/feature/How-to-scrape-data-from-a-website`. (Last accessed in 22/03/2024). [Cited on page 31]

[55] "Web scraping definition." Available at `https://avinetworks.com/glossary/web-scraping/`. (Last accessed in 22/03/2024). [Cited on pages ix and 31]

[56] B. Zhao, "Web scraping," *Encyclopedia of big data*, vol. 1, 2017. [Cited on page 31]

[57] "Web scraping – saiba o que é e para que serve." Available at `https://pplware.sapo.pt/internet/web-scraping-saiba-o-que-e-e-para-que-serve/`. (Last accessed in 22/03/2024). [Cited on page 31]

[58] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and ethics of web scraping," 2020. [Cited on page 32]

[59] J. Weiner, "Predicting the outcome of nba games with machine learning." Available at NBA-Machine Learning, 2021. (Last accessed in 19/03/2024). [Cited on pages 33, 36, 39, and 108]

[60] L. M. Lunelli, "Previsão de resultados de jogos da nba com algoritmos de machine learning." Available at `https://run.unl.pt/bitstream/10362/93006/`

1/TGI0277.pdf, 2019. (Last accessed in 23/04/2024). [Cited on pages 35, 36, and 108]

[61] W. Wu, "Injury analysis based on machine learning in nba data," *Journal of Data Analysis and Information Processing*, vol. 8, no. 4, pp. 295–308, 2020. [Cited on pages 35 and 36]

[62] "The rise of 3-point shot attempts." Available at Medium. (Last accessed in 02/04/2024). [Cited on pages ix, 38, and 39]

[63] "What is feature engineering?." Available at https://www.geeksforgeeks.org/what-is-feature-engineering/. (Last accessed in 04/04/2024). [Cited on pages ix, 46, and 47]

[64] M. Stump, "Statistical analysis of momentum in basketball," 2017. [Cited on page 47]

[65] Kevin, "Analyzing the impact of back-to-backs on nba team performance." Available at https://bleacherreport.com/articles/1520496-how-important-is-home-court-advantage-in-the-nba, 2024. (Last accessed in 04/04/2024). [Cited on page 48]

[66] K. BELHUMEUR, "How important is home-court advantage in the nba?." Available at https://bleacherreport.com/articles/1520496-how-important-is-home-court-advantage-in-the-nba, 2013. (Last accessed in 04/04/2024). [Cited on page 48]

[67] M. Chojnowska, "Decoding asynchronous programming in python: Understanding the basics." Available at https://sunscrapers.com/blog/python-async-programming-basics/, 2023. (Last accessed in 07/04/2024). [Cited on page 52]

[68] P. Kumar, "Implementing async features in python - a step-by-step guides." Available at https://www.velotio.com/engineering-blog/async-features-in-python, 2023. (Last accessed in 07/04/2024). [Cited on pages ix and 53]