Predicting Basketball Game Outcomes using Machine Learning Methods

Nicholas Cannon, 22241579

October 19, 2020

Abstract

Accurately predicting the outcome of a sporting game has become a popular research topic as the sports betting industry has become increasingly popular. Due to advancements in technology, professional sporting leagues, such as the National Basketball Association (NBA), have been able to improve their data collection and analytical systems and thus, large volumes of high quality in-game sporting data is now available online for free. Previous research has utilised this available data to train advanced machine learning and statistical models and two techniques have been established. However, a formal definition of these techniques and how they work has not yet been put forward. This study explores the two established techniques for basketball game prediction and provides a formal definition as well as various implementations using machine learning models such as: Logistic Regression, Random Forests & Neural Networks. Previous research has also largely ignored the use of the now available advanced basketball statistics provided by the NBA as training data. In hopes of improving on previous basketball prediction models, this study also uses the advanced statistics as training data, making comparisons to the traditional statistics commonly used by earlier research.

Acknowledgements

I'd like to thank the Computer Science & Software Engineering department at the University of Western Australia and Dr Du Huynh for assisting me with this research.

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

The ability to accurately predict a sporting game outcome has become increasingly important for both punters and companies in the sports betting industry. Advancements in technology has made the ability to wager on sporting matches incredibly easy, and as a result, the sports betting industry has seen a meteoric rise in popularity. The Australian Gambling Research Centre found that over half a million Australian adults regularly wagered on sports with a monthly expenditure of around $1000 in 2015 [1]. Various sporting leagues, such as the National Basketball Association (NBA), have also capitalised on the advancement of technology to improve their data collection & analytical systems. The NBA has implemented a sophisticated video tracking system in every arena, that has captured 25 frames per second of every game played since 2012-13 season. From this large amount of data generated, the NBA extracts various advanced statistics such as amount of times players touch & pass the ball, the speed at which the game is being played and other advanced metrics such as effective field goal percentage and offensive & defensive ratings, which are metrics adapted for basketball specifically. When combining the now available advanced statistics with the traditional statistics, the game of basketball can be described in a level of detail not possible before. The NBA provides both the advanced and traditional statistics online for free.

It has also become common for sporting franchises to start utilising data and analytics to gain insights into their teams and to aid with decision making. A famous example being Billy Beanes extreme management techniques when rebuilding the Oakland A's baseball team, in which rigorous statistical analysis on in-game statistics were used to value player performance instead of the opinions of expert scouts [10]. This has since become the norm in Major League Baseball and the use of data and analytics has carried over to other sporting leagues such as the NBA, where it is now common for teams to hire data scientists to stay competitive. The data and analytics movement has also had a large impact on the way in which basketball is being played in the NBA, the most obvious example being the increase in three point shot attempts after a simple analysis revealed

the efficiency of the shot. This is also known as the NBAs three point revolution and is discussed by Goldsberry in his book titled Sprawlball [8]. As such, the ability to gain actionable insights from sporting statistics and to predict future game outcomes is advantageous not only for both sides of the betting industry but also for the coaching and team staff of the franchises competing in professional sporting leagues.

With the explosion and availability of in-game team sporting statistics and the ability to financially benefit from sports betting, using advanced statistical and machine learning methods to predict game outcomes has become a popular research topic, with the NBA and the National Football League (NFL) being popular leagues for experimentation. The problem of predicting the outcome of a match can be defined as a classification problem or more specifically, a binary classification problem if the sport does not allow games to end in a draw. Many machine learning and statistical models [3, 6, 15, 13, 9] have been implemented to try and solve this classification task for many team sports. Models such as Logisitc Regression, Support Vector Machines, Random Forests and Neural Network techniques have been applied to a variety of professional sports such as American football [13, 9], Australian football & rugby [11] and basketball [12, 3, 14, 15].

Previous research has established two modelling techniques when using machine learning methods to predict games, in this thesis I refer to these two techniques as *modelling matchups* and *modelling teams*. These two techniques have been implemented using various classification models to predict game outcomes by predicting which team will win based on previous statistics collected. However, all previous research has proposed prediction models that implement only *one* of these techniques and as such a comparison between the two techniques does not exist. The techniques have also not yet been formally defined in anyway. This has made previous research difficult to interpret and reproduce as the requirements regarding the training and testing of the proposed models that implement these techniques has not been explicitly stated. The NBAs advanced statistics have also not yet been used as training data for basketball prediction models. This is in part due to the availability of the advanced statistics, as the NBA only began collection in 2012. However, recent research [14] has chosen not to include the advanced statistics when training basketball prediction models, due to it adding complexity to the research. As such, the predictive value of the advanced statistics has largely been ignored in previous research.

In this thesis, both of the established sports prediction techniques are implemented for basketball prediction using three machine learning models: Logistic Regression, Random Forests and Neural Networks. The models are trained using

two datasets, the first containing only traditional basketball statistics and the second, only advanced basketball statistics. This thesis aims to

- provide a formal definition of the two established sports prediction techniques in hopes of making clear the processes and requirements of each and to provide a comparison of the techniques when applied to basketball prediction, and
- improve on previous machine learning models for basketball prediction by exploring the use of advanced statistics as training data for the models, the proposed models are also trained with the traditional statistics for comparison.

This thesis is organised as follows: Chapter 2 will cover the previous research in machine learning & statistical methods for sports prediction and basketball prediction. Chapter 3 will formally outline the two established techniques applied in this paper for basketball prediction, provide an outline of the two datasets used for training and explain how the techniques implemented are evaluated and tested. Chapter 4 will provide the results for the baseline models and implemented machine learning models for both techniques and datasets. Finally, conclusions and future work is discussed in Chapter 5.

CHAPTER 2

Literature Review

2.1 Machine Learning in Sports Prediction

2.1.1 Neural Networks in American Football Predictions

One of the earliest studies that applied neural networks to predict match results in the National Football League (NFL) was published by Purucker [13] in 1996. Purucker used a basic multilayer perceptron (MLP) model with 5 input neurons, a hidden layer of 2 neurons and 1 output neuron trained with back propagation. The data set used was very minimal consisting of 5 features calculated from the first eight rounds of the season. The model did not focus on team match ups and instead modelled the performance of individual teams over the previous rounds. The output of the MLP would give the relative strength of each team which was then used to rank the teams. In an upcoming match up, the team with the higher rank is predicted to win. Given the small number of features and the basic model, Purucker was still able to achieve an accuracy of 61% compared to the 72% accuracy of domain experts at the time.

Building off the work of Purucker [13], Kahn [9] achieved a better model which outperformed NFL expert predictions. Kahn gathered data from the 2003 NFL season up to week 14 with a total of 208 instances. Kahn hypothesised that differential statistics, that is the difference between the offensive and defensive team statistics, had predictive value and compiled a list that were chosen as most indicative of a winning football team. Interestingly, Kahn did not include any statistics related to scoring as he suggested it would bias the network towards a single feature. Game outcomes were labelled as home team wins with a $+1$ and away team wins with a -1. The optimal network was a back propagation network with a learning coefficient of 0.01, momentum of 0 and a 10-3-2 network structure. The output neurons defined the probability of the home and away teams winning the game, the neuron with the higher value was taken as the winner of that game.

Kahn tested the network with two different prediction sets from weeks 14 and 15 of the 2003 NFL season. One set contained season averages (averages calculated from all previous games) and the other contained averages calculated from the past 3 games only. The results showed that season averages produced better prediction results. Kahn suggested that this was due to teams being fairly consistent in performance over the season. Other research [5] has suggested the same idea, using more games in calculating feature averages leads to better results. Comparing this network with predictions made by eight experts from the Entertainment and Sports Programming Network (ESPN) found that the neural network on average was 3% more accurate. Although Kahn did not train with any previous season data, he suggests it would be valuable for training as the modes in which teams win games in the NFL is not likely to change between seasons.

2.1.2 Neural Networks in Australian Football, Rugby & Soccer Predictions

McCabe and Trevathan [11] proposed an MLP model for predicting sport outcomes given only basic information. The model was applied to multiple major sports leagues including the Australian Football League (AFL), the Australian National Rugby League (NRL), Super Rugby and the English Premier League (EPL). McCabe and Trevathan compared back propagation and conjugate gradient descent learning algorithms and found that although back propagation took longer to converge with root mean square (RMS) error, it arrived at a more optimal value. The MLP consisted of three layers, an input layer with 19 inputs, one hidden layer with 10 neurons and a single output neuron. The output of the network was defined as the probability that the team would win its next game. An output of close to 1 gives high confidence that the team will win their upcoming game and hence low values give lower confidence.

The data set used to train their MLP consisted of 13-38 rounds (depending on the league) with 19-20 features. Round by round cumulative team statistics were calculated from this raw data. McCabe and Trevathan experimented with a player availability feature that attempted to capture which star players were available for a team in a given round, this feature was abandoned as it did not improve model accuracy and was complicated to process.

The MLP model was trained using previous rounds including rounds from previous seasons. Given a game between two teams in the next round, the MLP was used to give the probability of each of those teams winning their upcoming game in that round, the team with the higher probability was chosen as the

winner of that game. The model was tested live on all four leagues for at least three seasons and the average, best and worst accuracies are shown in Table 2.1 below.

League	**Best**	**Worst**	**Average**
AFL	68.1%	58.9%	65.1%
NRL	67.2%	52.2%	63.2%
Super Rugby	75.4%	58.0%	67.5%
EPL	58.9%	51.8%	54.6%

Table 2.1: MLP live prediction accuracy results extracted from [11] over the 4 seasons tested.

It is clear from Table 2.1 that some sporting leagues are harder to predict than others, for example the EPL league is much harder to predict than the Super Rugby league, when comparing the average accuracies over the 4 seasons tested. This table also shows that some seasons are much harder to predict than others, this is clear when comparing the Super Rugby leagues best and worst accuracies which differ by almost 18%.

2.2 Basketball Prediction

Much like the previous sports covered, basketball is also a team sport which follows a similar format to NFL, Australian football & Rugby. Basketball games do not end in draws, instead extra time is added (overtime) until one of the teams scores more points than the other. This means that predicting basketball games can be defined as a binary classification problem.

2.2.1 Statistical Methods

Cheng et al. [6] proposed a maximum entropy model called NBAME that fits to discretized NBA statistics for predicting game outcomes. The problem was formalized as a two class classification problem. Each game had 28 features describing the performance of the two teams (14 features per team) and the labelled outcome of the game indicating if the home team won. Only common statistics, such as *points scored*, *assists*, *blocks*, *turnovers*, etc., were used to describe a game. Features were discretized using partition around medoids (PAM) with $k = 3$. The data set consisted of 10,271 instances from all games from the 2007-08 to the 2014-15 NBA seasons. The mean of the most recent 6 games prior to

the game being predicted was calculated for each feature and unlike Kahn [9] no season average was tested with the NBAME model.

Cheng et al. [6] did not entertain the idea that data from previous seasons would be valuable in training the NBAME model unlike other research [9, 4, 11, 3]. The data set was sorted by date and separated by season, training and testing a new NBAME model per season. The authors explored the idea of adjusting the classification threshold value and discussed its impact on model accuracy. For example, setting a threshold value of 0.6 meant the model made predictions defined by Equation (2.1):

$$h(x) = \begin{cases} 1 \text{ (win)}, & \text{score}(x) \geq 0.6 \\ 0 \text{ (lose)}, & \text{score}(x) \leq 0.4. \end{cases} \tag{2.1}$$

Setting a high threshold value in Equation (2.1) helps to increase the model's accuracy but fewer games can be predicted because, games falling inside the range of $0.4 < \text{score}(x) < 0.6$ are ignored. The threshold value can thus be considered as a trade-off between accuracy and number of predictions. Whether or not this is valuable for sports prediction is up for discussion.

Trawiński [16] proposed a fuzzy classification system for predicting basketball games in the Spanish ACB league. The motivation behind using a fuzzy rule learning system was because they had soft decision boundaries that are not always axis parallel. The fuzzy systems were trained with the 2008-09 ACB season only as Trawiński wanted to avoid capturing changes between seasons. The single season contained 238 instances.

The first experiment Trawiński conducted was an intuitive naive approach which only considered data from the past 3 games. 6 features were used in the prediction which consisted of the difference in points between the two teams for the past 3 games. The results of this prediction system were not satisfactory and did not outperform a linear regression model which achieved an accuracy of 66.7%.

The second experiment was a more advanced approach that incorporated 15 features including averages of traditional basketball statistics (points scored, assists per game etc) and considered home and away perspectives with a binary flag. The inability for fuzzy classification systems to handle large feature sets motivated Trawiński to perform feature analysis on these 15 features using multiple feature selection algorithms. The results showed that the average number of points scored by the home team was the most influential followed by opponents average number of assists.

2.2.2 Machine Learning Methods

The results from Trawiński [16] described above are quite different when predicting NBA games, as reported by Thabtah et al. [15]. For NBA games, the most influential feature was found to be a team's defensive rebounds. Thabtah et al. explored machine learning techniques for making NBA game predictions using a feature subset generated from feature selection algorithms. The authors goal was to identify the most significant features of a basketball game when predicting game outcomes using artificial neural networks, naive bayes and logistic model trees. Data on 11 teams from the NBA finals from years 1980 through to 2017 was collected with a total of 430 instances. Each game had 21 features that consisted of standard basketball statistics that were totalled instead of averaged.

The feature selection process included a total of 3 different algorithms including 2 filtering methods (correlation based filtering & multiple regression) and 1 rule induction method (RIPPER algorithm). The results indicate that the defensive rebounds feature is highly influential as it appears in all three feature subsets. Models trained on the RIPPER feature subset achieved better performance in both accuracy & F1 metrics, the best model being the logistic model trees. Interestingly the ANN model performed better in accuracy & F1 when trained on the full feature set rather than the reduced feature set produced by the RIPPER algorithm. Overall there was a 2-4% increase in prediction accuracy on the other models when training with the RIPPER feature set.

Beckler et al. [3] published a paper researching machine learning techniques for decision guidance and game outcome prediction for the NBA. Beckler et al. applied 4 binary classification techniques including linear regression, support vector machines (SVM), logistic regression & artificial neural networks. The data set was built from raw individual player statistics from the 1992-93 to 1996-97 NBA seasons. The data was then accumulated for each team over all games to produce cumulative statistics for every game of the season. Games had a total of 30 features which include 14 traditional basketball statistics per team as well as each team's accumulated totals from last season, bringing the total number of features to 60. Beckler et al. found that taking the ratio of each team's features provided a better comparison as it shows one team's relative advantage over the other.

The linear regression model applied was standard, consisting of a weighted sum of all input features, a single bias value and a threshold of 0.5. The logistic regression model was a Matlab-based regression classifier that used maximum likelihood estimation (MLE) to optimize the weight vector which converged within 200-300 iterations of gradient descent using an epsilon threshold value of 0.01. The SVM model used was an open-source implementation called SVMLite.

Beckler et al. suggested the SVM solution would produce a higher test set accuracy than both linear and logistic regression due to its ability to maximise the margin of classification. A number of kernels were experimented with including basic linear, polynomial and radial-basis function with the simple basic linear kernel performing the best. Finally a Matlab based feed forward back propagation neural network model was applied to the data. The input layer had one node for each input feature, the single hidden layer had 20 nodes each using a sigmoid activation function and finally a single node output layer that used a threshold of 0.5 for classification. Beckler et al. were hesitant to increase the complexity of the network due to concerns that it would overfit the data.

Two data sets were used for training across all 4 classifiers. The first used only the previous seasons data to predict games in the current season and the second used both the previous season and all available data in the current season (games up until the game being predicted). Both experiments used 100-fold cross-validation. The average increase in model accuracy (across all 4 models) when using both previous and current season data was approximately 1%. The insignificant increase in accuracy shows the importance of previous season data in predicting game outcomes.

	Linear		**Logistic**		**SVM**		**ANN**	
	P	P+C	P	P+C	P	P+C	P	P+C
Mean	0.6991	0.7009	0.6744	0.6876	0.6596	0.6791	0.6478	0.6536

Table 2.2: Mean prediction accuracy results when trained with and without current season data extracted from [3].

The mean accuracy results across all seasons for each model can be found in Table 2.2. The best performing model was the linear regression model and the worst was the ANN model. Beckler et al. suggest that ANN struggled due to the lack of exploratory tuning which led to overfitting of the training data. To evaluate these models, the authors made comparisons to other prediction systems including a naive majority vote classifier that picked the team that had the fewest losses as the winner of an upcoming game, this model achieved an accuracy of 62%. The machine learning models were also compared to expert NBA predictions which averaged up to 71% accuracy. In conclusion Beckler et al. suggest using simple models for predicting NBA game outcomes.

2.2.3 Statistical Methods vs Machine Learning Methods

Cheng et al. discusses the shortcomings of machine learning models in this prob-

lem domain which highlights the motivation for the maximum entropy model put forward in [6]. It was suggested that prior research that had applied neural networks and random forests to solve this problem had overfit the training data due to a limited data set size. Cheng et al. argues a maximum entropy model overcomes this limitation by making use of little known facts and making no assumptions about the unknown. It was also suggested that the lack of independence between features in sports statistics is a major issue, using the naive bayes model from [12] as an example. Cheng et al. said that the maximum entropy model was an attempt to overcome the feature independence limitations. Cheng et al. also briefly discussed the SVM's inability to output probability values which makes results difficult to explain, this inability was not discussed by Beckler et al. when applying the model in [3].

Cheng et al. compared the NBAME model performance to a set of classical machine learning models including naive bayes, logistic regression, back propagation neural networks and random forests. The results showed that the NBAME model outperformed all classical machine learning models with a prediction accuracy of 74.4% with the highest classical machine learning model (a random forest) measuring an accuracy of 70.6%. However the implementation details of the classical machine learning models was left out of the study. The NBAME model was the best (only outperformed in two seasons) followed closely by the random forest model. The naive bayes model had the lowest average prediction accuracy of 60%. The neural network model fluctuated the most in prediction accuracy between seasons however outperforming the NBAME model for one season. On the other hand the logistic regression model had the most stable prediction accuracy between seasons.

The comparison made by Cheng et al. between statistical and machine learning models showed a correlation between model accuracies across different seasons. This correlation could indicate that some seasons are just harder to predict than others due to unanticipated natural factors. Beckler et al. [3] also found correlations in model accuracy when comparing multiple machine learning classifiers predicting games from multiple NBA seasons.

CHAPTER 3

Methodology

There are two machine learning techniques that can be used for predicting sporting game outcomes, more specifically, basketball game outcomes. I refer to these two techniques in this thesis as *modelling matchups* and *modelling teams*. The two techniques differ in the perspective they take on the game, the assumptions they make and what they ultimately end up predicting. Although these techniques differ, the predictions they provide are still used to solve the same underlying binary classification problem. Another important similarity between the two techniques is that they are both restricted to only using statistics from previous games to predict the next game. In this chapter, I will cover how both of these techniques work when predicting basketball games and how they can be trained and tested. I will also be covering the various machine learning models selected to be the underlying binary classifiers for these techniques, the dataset used and the pre-processing steps required for both techniques and finally how the techniques can be evaluated and the metrics used in the evaluation process.

3.1 Modelling Matchups Technique

In order to understand how the matchup modelling technique works when being applied to basketball prediction, it is important to first understand how games are organised in the NBA regular season. The NBA is made up of 30 teams across 2 divisions (East & West) and each team plays 82 games in the regular season, 41 of those games are played on that team's home court, the other 41 are played on their opponent's home court's. When a team is playing on their home court, they are referred to as the *home team* and their opponent is referred to as the *away team*. A team's home court is located in the city in which the team is based, for example, the New York Knick's home court is at Madison Square Garden in New York city. This means the home team has numerous benefits including not having to travel to the city where the game is held. Each team will play every other team during the regular season (the amount of games played

depends on the divisions of the two teams but this is not important), both teams will play each other on their home court an equal amount of times to reduce any bias. A game will always be played on one of the competing team's home courts. In other words, there will not exist a game where both teams are away teams.

Knowing this, a regular season NBA game can be (and most often is) described in terms of the home team and away team. The matchup modelling technique takes this perspective when trying to predict game outcomes. More specifically, the goal of the matchup modelling technique is to predict the probability of the home team winning. As this is a binary classification problem, predicting the probability of the home team winning can also be viewed as predicting the probability of the away team losing. Choosing to predict the probability of the home team winning has become the unofficial standard in the literature and allows for easier comparisons to previous work, however in the end, this decision does not impact final prediction results.

The matchup modelling technique, denoted MT_1, when given some measure of the home team's abilities f_H and the away team's abilities f_A (feature vectors), for any regular season game g, gives a probability p of the home team winning that game. Mathematically, this can be defined as follows:

$$\mathrm{MT}_1(g, f_H, f_A) = p, \text{ where } 0 \leq p \leq 1. \tag{3.1}$$

The class label can then be given by running p through a step function, denoted by SF, with a threshold value of 0.5:

$$\mathrm{SF}(p) = \begin{cases} 1 \text{ (home win)}, & \text{if } p \geq 0.5 \\ 0 \text{ (home lose)}, & \text{otherwise} \end{cases} \tag{3.2}$$

In the above definition, MT_1 can be any machine learning method capable of binary classification and both f_H and f_A can be any feature vectors that are indicative of the respective team's performance. In this thesis, we take the feature vectors f_H and f_A as the averages of the team's statistics over the past N games, not including the game being predicted. It is important to exclude the team statistics for the current game in the feature vectors, as this information is not available prior to the game being played. This makes sense because if we had the team statistics for the game we are trying to predict, we can see how many points both of the teams have scored in that game and therefore we would know the winner of that game (the team who scored more). We can define the feature vectors mathematically with the function FV, that returns the feature vector for a team t for their n^{th} game of the season g_n, averaged over N games:

$$\mathrm{FV}(g_n, t, N) = \frac{1}{N} \sum_{i=g_{n-1-N}}^{g_{n-1}} \mathrm{stats}_{ti} \tag{3.3}$$

In the above equation, stats_{ti} denotes the game statistics vector for team t in their i^{th} game of the regular season not the i^{th} game of the season. In this thesis we take the game statistics to be either the *advanced statistics* or the *traditional statistics* (see Tables 3.4 & 3.5 for the list of traditional and advanced statistics used). Since the home and away teams are both described using the same statistics, we can reduce the number of features by performing an element wise subtraction of the away features from the home features. This vector will essentially describe the home team's relative advantage over the away team for that game. The final feature vector f, for the regular season game g where t_H and t_A are the home and away teams for game g, can be defined as:

$$f_g = f_H - f_A = \mathrm{FV}(g, t_H, N) - \mathrm{FV}(g, t_A, N). \tag{3.4}$$

This would then simplify the definition of MT_1 in Equation (3.1) to:

$$\mathrm{MT}_1(g, f_g) = p, \text{ where } 0 \leq p \leq 1. \tag{3.5}$$

In order to train a model using this technique, the feature vector f_g and the outcome of game g (a boolean value denoting if the home team won game g), is required for all games in the training data set. In this thesis, we train this technique with all games of the season previous to the season used for testing. As discussed in Section 3.5, we use the 2018-19 NBA regular season for testing. This means the 2017-18 NBA regular season is used for training (see Section 3.4.1 for an in depth explanation of this dataset). Likewise, testing this technique requires the feature vectors f_g of the test season to be run through the classifier MT_1 and the step function SF to provide the final predictions. The final predictions are then combined with the ground truth to compute the evaluation metrics described in Section 3.5.

The overall goal of this technique is to learn the relationships between the various teams in the league. This technique attempts to pick up which teams play better against other teams, which teams thrive when playing on another teams home court and which teams do not. The main assumption of this technique is that a team's performance as a whole does not change significantly between seasons. Yes, players are traded between teams and the overall rosters and coaching staff can change but this technique assumes those changes have minimal effect between seasons and on the overall team performance.

3.2 Modelling Teams Technique

The second modelling technique explored in this thesis takes an entirely different perspective. Instead of attempting to learn how two teams perform against one another, the team modelling technique attempts to learn how a single team will perform in their next game based only on their performances in previous games. A game is still described in terms of the two teams playing but instead of training a model to predict if the home team of that game will win, we instead train a model to predict (separately) the probabilities of those two teams winning that game and then take the team with the higher probability as the winner. A team's probability of winning their next game can be thought of as some kind of quantitative ranking up to that game in the season.

We can define this technique mathematically with MT_2, where g is the upcoming game, f_t is some measure of team t's performance up to but not including game g and p is the probability of team t winning game g:

$$\mathrm{MT}_2(g, f_t) = p, \text{ where } 0 \leq p \leq 1. \tag{3.6}$$

The above equation defines the feature vector f_t quite loosely, as it can be any measure that is indicative of team t's performance in the season. Again, like the matchup modelling technique, the feature vector f_t is under the same restrictions as the f_H and f_A feature vectors, in that statistics from future games played by team t can not be incorporated into the measure. We can actually use the same function FV, defined in Equation (3.3), to obtain the feature vector f_t for team t in game g. However, unlike the matchup modelling technique, the feature vector output from FV is used directly by MT_2 and is not combined with the opposing team's feature vector. This can be defined as follows:

$$f_t = \mathrm{FV}(g, t, N). \tag{3.7}$$

The output probability p alone does not supply sufficient information to predict the outcome of game g. In order to predict game g we also need the other team's probability of winning game g. However, if we define the opponent of team t in game g as t', then the probability of t' winning game g is *not* equal to the complement of p:

$$\mathrm{MT}_2(g, f_{t'}) \neq 1 - \mathrm{MT}_2(g, f_t) \tag{3.8}$$

Since the above is true, it is easier to think of the output of MT_2 as a ranking function. To obtain the winner of game g, all we need to do is run both probabilities p_t and $p_{t'}$ (the probabilities of team t and team t' winning game g), through

a function that returns the team with the higher probability. This function can be defined as follows:

$$W(p_t, p_{t'}) = \begin{cases} t, & \text{if } p_t > p_{t'} \\ t', & \text{if } p_t < p_{t'}. \end{cases} \tag{3.9}$$

Note that, there could exist a game in which the outputs for p_t and $p_{t'}$ are both equal. This would essentially mean that both teams are equally ranked or equally likely to win the next game. To deal with this situation we can instead define t as the home team of game g and then when p_t is equal to $p_{t'}$, we can select team t as the winner because team t has the home court advantage, this is better than randomly choosing. This can be defined with the function W_2:

$$W_2(p_t, p_{t'}) = \begin{cases} t, & \text{if } p_t \geq p_{t'} \\ t', & \text{if } p_t < p_{t'} \end{cases}, \text{ where } t \text{ is the home team of game } g. \tag{3.10}$$

So we use W_2 instead of a step function such as SF given in Equation (3.2) to obtain the prediction results. This requires that the underlying binary classifier used by MT_2 be able to output class probabilities.

To train this binary classifier, we require a dataset that contains, for each team t, the feature vector f_t for all games g played by t and a boolean flag that describes whether or not t won g. Since there are two teams involved in a game, this dataset will have two descriptions for each game. For example, suppose that teams t and t' both played in game g and team t won that game, in the data there would exist both the [g, f_t, 1] and [g, $f_{t'}$, 0] vectors, where the last element in the vector is the label denoting if that team had won.

In this thesis the dataset has not been limited to only one season like the matchup technique dataset. Instead, we take *all* available seasons previous to the test season and use the function FV, defined in Equation (3.3), to compute the feature vector f_t for all teams over all games in the previous seasons. It is best to think of this process as individually selecting a date ordered list of all games played by t and rolling the function FV over this dataset with a window size of N for all teams t and then concatenating this dataset together.

The testing process is similar in that the feature vectors f_t are computed over the test season for each team, then for each game of the season we grab both of the competing team's features up to the interested game, which is then used to compute the probability of that team winning, those probabilities are then run through a function such as W_2 to obtain the final prediction. All the final predictions are then paired with the ground truths to compute the relevant metrics.

Unlike the matchup modelling technique which looks at the relationships between various teams in the league, the team modelling technique takes a much more individualistic approach to modelling teams. The technique attempts to pick up how teams perform overtime and identify when teams are likely to go on hot or cold streaks based on their recent performances. This technique does *not* assume a team's performance is stable between seasons (this is the main assumption of matchup modelling), instead it tries to model those changes between seasons and between games. However, this technique does assume that changes in opponents have minimal impact on the team's performance and it makes no attempt to learn those relationships.

3.3 Proposed Models

As discussed above, the techniques MT_1 and MT_2 both require a binary classification model, with the restriction that MT_2 requires one that can output class probabilities. For this thesis, three binary classification models have been implemented: Logistic Regression, Random Forest & Neural Networks. All three are sufficiently diverse in the way they work and are all capable of outputting class probabilities. These models also differ in levels of complexity, with the Logistic Regression model being less complex when compared with the Neural Network. Using these models allows for comparison between classical machine learning, ensemble learning and deep learning approaches to basketball game prediction.

The Support Vector Machine (SVM) model was excluded from this study as it does not natively output class probabilities, this makes the model unsuitable for the team modelling technique (MT_2) and also makes classifications harder to interpret. Cheng et al. [6] raised concerns with the SVM model and its use in sports prediction and chose not to include it when making comparisons to their proposed Max Entropy model. Beckler et al. [3] implemented an SVM model for NBA game predictions and hypothesised it would perform the best. However, the results found that the SVM model performed the second worst out of the 4 models implemented.

3.3.1 Logistic Regression

The first model explored in this thesis is a simple Logistic Regression model. It was selected because of its simplicity and effectiveness in binary classification tasks. A Logistic Regression model defines a weighted sum of statistics that is run through a Sigmoid function to produce a probability that an instance

	Matchups		Teams	
Hyper-parameter	**Advanced**	**Traditional**	**Advanced**	**Traditional**
C	0.8	1.0	0.5	0.5
Penalty	Lasso			

Table 3.1: Optimal hyper-parameters for the Linear Regression models for both modelling techniques using both advanced and traditional statistics.

belongs to the positive class. In the context of basketball prediction, the Logistic Regression model will compute a weighted sum of the f_g and f_t feature vectors for the matchup and team modelling techniques respectively.

Table 3.1 breaks down the hyper-parameter values for each Logistic Regression model implemented over the two modelling techniques. The hyper-parameter values were tuned with a 5 fold cross-validation process. The various types of penalties experimented with include: Ridge (ℓ_2), Lasso (ℓ_1) and no penalty, the best performing penalty for all models was the Lasso penalty. However the amount of regularization between each of the models varied, each model was tested with a C value of 1.0, 0.8 and 0.5. When modelling matchups, the optimal C value was 0.8 and 1.0 when training with the advanced and traditional statistics respectively. Interestingly, when modelling teams the Logistic Regression models were more regularized with an optimal C value of 0.5 for both datasets.

The optimisation algorithm (solver) used by all Logistic Regression models in this thesis is the Library for Large Linear Classification solver (Liblinear solver). The Liblinear solver uses a coordinate descent algorithm that minimizes a multivariate function by successively solving uni-variate optimization problems. This means that the path to the minimum moves along one axis at a time. The Liblinear solver was selected as it is well suited for small datasets with high dimensionality.

3.3.2 Random Forests

The Random Forest model was the next model selected to be the underlying classifier for both modelling techniques because it is an ensemble learning method that has the ability to compute feature importances. This makes the Random Forest an ideal for this thesis as its learning technique is sufficiently diverse from the other two models implemented and the feature importance values will provide valuable insights into the underlying classification problem. The Random Forest model works by training multiple Decision Trees on various subsets of the training data to produce an ensemble of diverse models. Upon inference, the instance to

	Matchups		Teams	
Hyper-parameter	**Advanced**	**Traditional**	**Advanced**	**Traditional**
N-Estimators	500	500	500	300
Min-samples Leaf	1		3	
Bin Strategy	K-Means		Quantile	
N-Bins	3			

Table 3.2: Optimal hyper-parameters for the Random Forest models for both modelling techniques using both advanced and traditional statistics.

be predicted is passed through each of the Decision Trees in the ensemble and the statistical mode of all the predictions is then taken as the final prediction. By taking advantage of the predictions from many diverse and independent models with uncorrelated errors, the Random Forest is able to overcome any overfitting issues commonly experienced when training a single Decision Tree model.

The number of estimators and the minimum samples per leaf hyper-parameters were tuned for the Random Forest models using 5 fold cross-validation. Table 3.2 provides an overview of the hyper-parameter tuning results for all Random Forests in this thesis. All Random Forest models were tested with 300 and 500 estimators. The optimal number of trees across all models was found to be 500, apart from the team modelling Random Forest trained using the traditional statistics, which had 300 trees. The minimum number of samples per leaf however was different only between the two techniques, a minimum number of 1 and 3 samples per leaf was optimal for the matchup and team modelling techniques respectively.

The Random Forest models also performed better on discretized input data. For this, Scikit-Learns *KBinsDiscretizer* was used to discretize the data before training and testing. Bin sizes of 3 and 5 were explored as well as the K-Means and quantile binning strategies. The matchup data performed best when discretizing the data into 3 bins using the K-Means strategy. As for the team data, it to was also discretized into 3 bins, however the optimal binning strategy was the quantile strategy.

Other hyper-parameter values for the Random Forest models were experimented with, however changes to these hyper-parameters either made no difference to the final evaluation metrics or they negatively impacted it. This includes limiting the maximum depth of the trees during training, which meant that the depth of the tree could grow as deep as necessary. Any attempts to regularize the trees by limiting their depth did not improve the performance of the models. The trees were also trained with bootstrapping to introduce more diversity into

the subsets used for training the individual models in the ensemble.

3.3.3 Artificial Neural Networks

The final model implemented in this thesis is an Artificial Neural Network (ANN) model. The ANN model was selected because of its ability to solve difficult non-linear classification tasks. The model has also been a popular choice in previous literature surrounding sports prediction. In this thesis, a Multilayer Perceptron (MLP) model is used as the underlying classification model for both modelling techniques. An MLP is a feed-forward ANN consisting of an input layer, some number of hidden layers and then an output layer. All nodes (except for the input nodes) are given a weighted sum of input values which is then run through a non-linear activation function to provide a single output value for the node. The combination of non-linear activation functions and multiple layers of neurons allows the MLP to classify data that is not linearly separable.

When exploring the Neural Network models for both techniques, changes to the hyper-parameters and network architectures were explored manually. From the experimentation, it was found that simpler architectures outperformed more complex ones upon comparison of the final evaluation metrics. In general, wider networks with more nodes in each layer, performed worse than skinnier deeper networks. All Neural Network models were trained with Adam optimization with a learning rate of 10^{-3}, decay of 10^{-4} and a binary cross-entropy loss function. Other optimizers such as RMSProp were experimented with, as well as varying learning rate values (10^{-2} and 10^{-4}), however these changes did not impact final model results. Early stopping was also utilised to prevent overfitting of the models, if the model's F1 score (see Section 3.5) did not improve after 10 training epochs, the training was halted and the model weights from the best epoch were used as the final weights. Training without early stopping for 100 epochs, had no positive impact on the model and lead to overfitting in most cases. Finally, prior to training the networks, the input data was scaled between a range of 0 and 1. An overview of the hyper-parameter values for all Neural Network models implemented in this study across both techniques can be seen in Table 3.3.

The optimal network structure when modelling matchups included two hidden layers with 16 and 8 nodes, both using a Scaled Exponential Linear Unit (SELU) activation function. The input layer had 75 and 74 nodes for the advanced and traditional datasets respectively, one for each input feature. The output layer consisted of a single node that utilized a sigmoid activation function. Batch Normalisation was used after the input layer and both hidden layers. To regularize the model, 20% dropout was used after the two hidden layers during training.

	Matchups		Teams	
Hyper-parameter	**Advanced**	**Traditional**	**Advanced**	**Traditional**
Input neurons	75	74	25	18
Neuron layout	input - 16 - 8 - 1		input - 8 - 4 - 1	
Dropout	20%		None	
# hidden layers	2			
Hidden activation	SELU			
Output activation	Sigmoid			
Learning Rate	0.001 with 10^{-4} decay			
Optimizer	Adam			
Batch Norm	Yes			

Table 3.3: Network structure and optimal hyper-parameter values of the Neural Network models for both the matchup and team modelling techniques trained on both datasets.

Similarly when modelling teams, a network structure with two hidden layers were used. The input layer consisted of 25 and 18 nodes for the advanced and traditional statistics respectively. The two hidden layers contained 8 and 4 nodes with a SELU activation function and Batch Normalisation implemented after. Unlike the matchup modelling architecture, this one did not perform better with drop out regularisation. Finally, the output layer consisted of a single node with a sigmoid activation function.

3.4 Data Collection & Preparation

The data used in this project was collected from the NBAs advanced statistical website (*https://stats.nba.com*). Data from the 2013-14 NBA regular season up to the 2018-19 regular season was collected, data prior to the 2013-14 season was not considered due to the availability of advanced statistics. A Python script was written that automated the collection of game statistics from the website for all regular season games in the desired seasons. Playoff games were not considered because it would cause an imbalance in the dataset, as only the top 16 teams of the season get to compete in the playoffs. The way in which teams approach and strategise playoff games is also different when compared to regular season games.

The main motivation behind building the data set using the NBAs advanced statistical website was to collect both the traditional and advanced statistics shown in Tables 3.4 and 3.5, which are only available on this website. In addition to the traditional box scores, the advanced, miscellaneous, four factor and player

Statistics	Description
FGM, FGA, FG_PCT	Field goals made, attempted & percentage
FG3M, FG3A, FG3_PCT	3 point field goals made, attempted & percentage
FTM, FTA, FT_PCT	Free throws made, attempted & percentage
OREB, DREB, REB	Offensive, Defensive & total rebounds
AST	Assists
STL	Steals
BLK	Shots blocked
TO	Turn overs
PF	Personal fouls
PTS	Points scored
PLUS_MIN	Plus minus

Table 3.4: List of traditional basketball statistics collected for this project.

tracking box score statistics were also collected. It is important to note that some statistics can be derived from others, for example a teams shooting performance can be described in terms of the number of shots made and the number of shots attempted. It can also be described using a percentage which is derived from the number of shots made and attempted. Tables 3.4 and 3.5 have related statistics on the same row. Tests showed that the models performed better using the less granular features such as the shooting percentages.

Each game required 7 different requests to fetch the various box score statistics and game information in JSON format, the raw JSON files were then saved to disk for later processing. Fetching each season (1230 games) concurrently completed within 3 hours. After successful fetching of the raw game data, another two Python scripts were written to extract the relevant statistics from the raw data to CSV files in the format required for the proposed modelling techniques.

3.4.1 Matchup Data

To model the home and away matchups of NBA games, data from both the home and away team perspectives is required. A script was written to extract the home and away statistics for each game from the raw data. These statistics were then combined into a single CSV file separated by season, where each row corresponds to an individual game. A game is made up of team features (either the advanced or traditional statistics) from both the home and away teams, which are prefixed with '$H\_$' or '$A\_$' respectively. Each game is labelled with either a 1 or 0 that denotes whether or not the home team won that game. The 2017-18 NBA regular season was used for training, previous seasons could have been

Statistic	Description
OFF_RATING, DEF_RATING	Offensive & defensive rating
FTA_RATE	Free throw attempt rate
TM_TOV_PCT	Team turnover percentage
EFG_PCT	Effective field goal percentage
TS_PCT	True shooting percentage
PACE	Number of possessions per 48 minutes
PTS_OFF_TO	Points off turnovers
PTS_2ND_CHANCE	Second chance points
PTS_FB	Fast break points
PTS_PAINT	Points in the paint
DIST	Distance run (miles)
ORBC, DRBC, RBC	Offensive, defensive, total rebound chances
TCHS	Touches
PASS	Passes made
CFGM, CFGA	Contested field goals made & attempted
UFGM, UFGA	Uncontested field goals made & attempted

Table 3.5: List of advanced basketball statistics collected for this project.

used as well, however there was no increase in model performance when doing so. Before any pre-processing, this data set consisted of 84 columns and 1230 rows per season (7380 rows across all seasons).

Before being fed into the models for training, the matchup data was pre-processed with a series of transformations using a custom transformer built with Pandas and Scikit-learn (see Appendix B). This transformer applies all the necessary transformations for both the numerical and categorical features. The transformer first fits a one-hot encoder that encodes the home team and away team IDs as one-hot vectors. It is important that the team IDs be encoded as we do not want the models to make any associations with the numerical value of a team's ID. After encoding the team IDs, the transformer uses historical facts in the data to compute features that describe if the home team won their last game and if the home team is the series leader. Then either the advanced or traditional features are dropped before computing team averages.

In order to compute each team's averages over the past N games before the game being predicted, the data must be temporarily grouped by both team IDs and sorted by date whilst retaining the original data frame index. This allows rolling transformations to be applied once per team in isolation. Essentially applying a transformation on a subset of games a team is involved in over the season, with the first row being the first game they played and the last row the

Game ID	DF index	H ID	A ID	H stats	A stats	H WIN
g_n	5	6	5	...	...	1
g_{n+1}	12	6	8	...	...	0
g_{n+2}	23	6	12	...	...	0
...	...	...	...	...	...	...
g_{n-1}	34	6	13	...	...	1

Table 3.6: Subset of the matchup rows after temporarily grouping by team ID and sorting by date, this example shows the group with home team ID 6.

last game they played as shown in Table 3.6. This is necessary to compute the rolling average of a team's features over the current season.

The rolling average window size (N) defines how many games previous to the current game are to be used to compute the average. Multiple window sizes were experimented with as well as an expanding window which uses all available previous games in the average (essentially a season average). Averaging the features of a team's previous 5 games was found to give the best results when predicting games. The averages are then shifted down by one row inside the group, this ensures that only the mean up to the previous game played is used to predict the current game. The data is then ungrouped by team and sorted by the original index such that it is back in the same ordering as before the transformation. Finally the home team averages are subtracted from the away team averages for each row to give the relative advantage (or disadvantage) the home team has over the away team. After all transformations, the final dataset consisted of 76 features (including the label) and 1063 rows.

3.4.2 Team Data

Modelling a team's probability of winning their next game requires a dataset for each team that comprises of all the games that team has played over all available seasons. This is unlike the matchup data described in Section 3.4.1 as it describes a game from only one of the team's perspectives, not from both the home and away team perspectives.

A script was written to extract all the games that each team had played (both at home and away) over all seasons from the raw game data. The extracted data was saved to a separate CSV file. Each row in the CSV files corresponds to a single game played by that team, with features that describe how well that team did in that game. This essentially means that there exists two rows in two different CSV files that describe the same game, but from two different perspectives. Across

Game	Season	ID	Team Stats	Team_A Stats	Home	Win
g_n	2013	x	...	...	1	1
g_{n+1}	2013	x	...	...	0	1
g_{n+2}	2013	x	...	...	0	0
...	...	...	...	...	...	...
g_{n-1}	2019	x	...	...	1	1

Table 3.7: A representation of the team dataset format, this table shows all games (home & away) played by team with ID x over the available seasons.

the 2013-14 and 2018-19 NBA regular seasons, each team had played a total of 492 games.

For each game, both the traditional and advanced statistics for the interested team are recorded. Other features that provide context to the game were computed, such as a feature that describes whether or not the interested team was playing on their home court for that game. To capture how well opponents played against the interested team, basic statistics (points, rebounds, assists, steals & blocks) earned by the opposing team were also recorded for each game. The opponent features are post-fixed with '$\_A$' (A for against). Each row is labelled with a 1 or 0 denoting whether or not the interested team won or lost that game respectively. In total, there are 51 features (including the label) that describe the game from one of the team's perspectives. A summarised version of the team dataset format can be seen below in Table 3.7, where the "Team Stats" and "Team_A Stats" represent either the advanced or traditional statistics for the interested team and the opponent team respectively. Note how Table 3.7 contains games starting from the 2013 season up to the latest 2019 season.

As opposed to the matchup data described in Section 3.4.1, the team data does not require complex groupings before calculating the team statistic averages as the data is already grouped by team. For each team's CSV file, four transformations are applied:

1. First the label is shifted up one row, this is to ensure that the outcome of a team's next game is described only by features previous to that game.

2. After shifting of the label, either the advanced or traditional features are dropped.

3. Then an averaging scheme is applied to the remaining team features (and opponent features). A rolling mean and expanding mean (season average) were experimented with, tests showed that a rolling mean with a window size of 5 performed the best when predicting games.

4. Finally all rows containing missing values were dropped, in total 180 rows were dropped (6 rows for each team).

After each team's data was processed independently, all the CSV files were concatenated together. All games from the 2018 season were then filtered out of this dataset to produce the test dataset, the rest of the games made up the training data. The final training set contained data on 12150 games from the 2013-14 to 2017-18 NBA seasons (remembering that there are two perspectives of each game) and the test data set contained 2460 games (2 perspectives for 1230 games).

3.5 Evaluation Techniques

The techniques proposed in this project are tested using the 2018-19 NBA regular season. This season was the latest full season available at the time this study was conducted. As mentioned in previous research [6, 3], the performances of models can fluctuate depending on the season being tested. For this reason, both the team and matchup techniques were tested using the same season. By using the 2018-19 season for testing (the latest), the team modelling technique described in Section 3.2 is able to be trained on a larger dataset.

The F1 score is used as the main scoring and evaluation metric for the models proposed in this project, as opposed to accuracy like previous research [6, 3, 15, 16]. The main motivation behind excluding accuracy from the evaluation techniques was due to the imbalanced nature of the datasets. This unbalance is a consequence of the home court advantage that is present in the game of basketball. This advantage is well known and can be seen when looking at the distribution of home wins and away wins for any season, in which there will exist more instances of home wins than away wins.

The F1 metric is defined by Equation (3.11) below. It is the harmonic mean of precision and recall and is a suitable measure for imbalanced binary classification tasks. Precision is defined as the proportion of true positives in the total positive classifications returned by the algorithm. Recall is defined as the proportion of positive instances in the data that are correctly classified by the algorithm. The F1 score ranges between 0 and 1, with 1 meaning perfect precision and recall. The benefit of using F1 over accuracy is that it gives equal importance to precision and recall, both of which are valuable when predicting basketball games.

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.11}$$

Other measures used for evaluation include: confusion matrices, receiver operating characteristic curves (ROC curve) and ROC area under curve (ROC AUC). These measures allow for further understanding of the predictions each model is making, where it may be struggling and how it acts when its threshold value is adjusted.

CHAPTER 4

Results & Discussion

4.1 Baselines

Before implementing any machine learning techniques to model the data, a set of 3 baseline models were used to establish prediction results using little to no intelligence. These baseline models only rely on historical observations from the data. Comparisons made between the machine learning models and the following baseline models allowed a quantitative value to be calculated that shows the machine learning technique's relative advantage over simple naive approaches. The baseline results also provided context to the test dataset and reasoning for the exclusion of accuracy as an evaluation metric. The test set F1 scores for the three baseline models is displayed in Table 4.1. The confusion matrices for the three baseline models can be seen in Figure 4.1, the percentages are normalised across the rows (true labels). The PR and ROC curves have been omitted as the baseline models do not produce the probability values that are required to plot the curves.

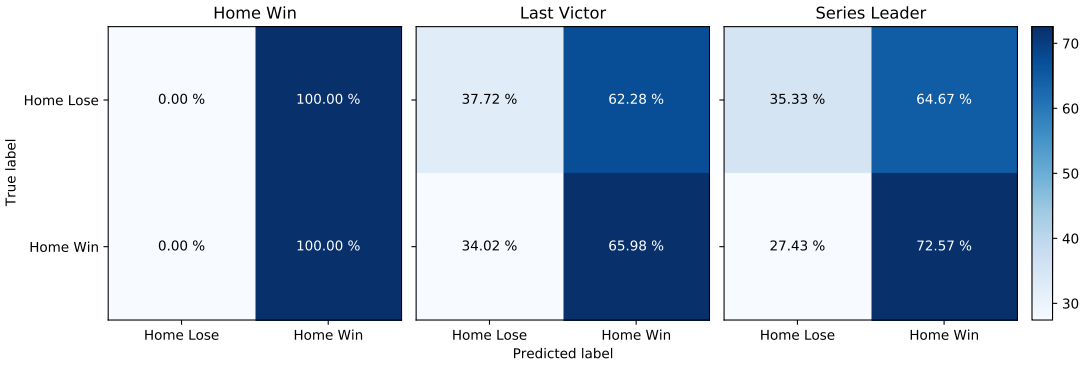

Figure 4.1: Test set prediction results for the three baseline models implemented. The confusion matrix percentages are normalized across the rows (true values).

	Home Win	**Last Victor**	**Series Leader**
F1 Score	0.441	0.539	**0.561**

Table 4.1: Baseline model F1 scores on the test set, with the best performing baseline score in bold.

4.1.1 *Home win* baseline model

The simplest approach to predicting basketball games is to always select the home team as the winner. Always choosing the home team rather than randomly selecting one of the teams yielded better results due to the home court advantage present in basketball. It is well known in basketball that teams playing on their home courts have an advantage over their opponents, this can be seen by looking at the distribution of home wins and home losses in which more instances of home wins will be observed for any given season. The naive baseline model, denoted by BL_1, simply assigns the home team as the winning team for every game. Mathematically, BL_1 defines the chosen label between the home team t_H and the away team t_A for their n^{th} game g_n in the series:

$$\mathrm{BL}_1(g_n, t_H, t_A) = 1 \text{ (home win)}. \tag{4.1}$$

From the results it is clear how imbalanced the data set is, the model achieved an accuracy of 59.3%, when it predicted 100% of the games where the home team lost incorrectly. Scoring this model using the F1 metric achieved a score of 0.441, which is a more suitable measure of the performance of this approach.

4.1.2 *Last victor* baseline model

Building off the previous approach, historical facts in the data were then used to improve the baseline model results. More specifically, the team who won the last game in the series was taken as the winner of the next i.e., the series leader is always chosen as the winner of the game. In other words, given a game between team t and team t' find the last game where t played t' and select the winner of that game as the winner of the next, if no previous game between t and t' exists then pick the home team. In terms of the home team t_H and the away team t_A, for a given game g_n in the series of games between t_H and t_A, this can be defined as:

$$\mathrm{BL}_2(g_n, t_H, t_A) = \begin{cases} 1 \text{ (home win)}, & \text{if } t_H \text{ won game } g_{n-1} \\ 1 \text{ (home win)}, & \text{if } g_{n-1} \text{ does not exist} \\ 0 \text{ (home lose)}, & \text{otherwise.} \end{cases} \tag{4.2}$$

Using this information the model was able to predict $\approx 38\%$ of games correctly where the home team had lost. However, there are a significant amount of games ($\approx 34\%$) in which this approach incorrectly predicted that the home team would lose when they won. This could be because the home court is alternated for each game in the series, for example if the first game g_1 between teams t and t' was at t's home, the next game g_2 would be at t''s home and so on. It is possible that t beat t' in the first game because t had the home court advantage which they won't have in the next game against t' on t''s home court. The reason for a high number of incorrect home win predictions could be explained by the lack of initial historical data available, in which the model resorts back to the most naive approach of always selecting the home team as the winner. This approach was still an improvement on the previous one, with an F1 score of 0.539 during the 2018-19 season.

4.1.3 *Series leader* baseline model

The final approach implemented selected the team who had won the majority of the games in the series as the winner i.e. always choosing the series leader as the winner. For example if team t won two out of the three games against team t', then select team t as the winner of the next game against t'. If there were no previous games between t and t' or they are both tied for series leader, resort back to selecting the home team as the winning team. In terms of the home team t_H and the away team t_A, in the n^{th} game of the series denoted g_n, BL_3 defines the chosen label as follows:

$$\text{BL}_3(g_n, t_H, t_A) = \begin{cases} 1 \text{ (home win)}, & \text{if } t_H \text{ is the series leader} \\ 1 \text{ (home win)}, & \text{if series is tied or } g_{n-1} \text{ does not exist} \\ 0 \text{ (home lose)}, & \text{otherwise} \end{cases} \tag{4.3}$$

This approach still struggled with classifying home wins as home losses, however not as bad as the previous *last victor* approach. With an F1 score of 0.561 on the 2018-19 season, this was the best baseline approach implemented using only historical facts.

4.2 Matchup Modelling Results

Table 4.2 contains the F1 test set results for the machine learning models implementing the matchup modelling technique for both the advanced and traditional

	Advanced (F1)	Traditional (F1)
Logistic Regression	0.636	0.621
Random Forest	0.613	0.626
Neural Network	0.624	0.611

Table 4.2: F1 metric results for the three machine learning models using the matchup modelling technique on the test set.

	P-value	t Statistic
Logistic Regression	0.572	-0.604
Random Forest	0.636	0.503
Neural Network	0.758	-0.325

Table 4.3: 5 x 2 cross validated t-test results between the advanced and traditional models using the matchup modelling technique.

features of 75 and 74 dimensions respectively (Table 3.3). Given that the best baseline F1 score is 0.561, the results show that all of the models implemented at least outperform the baseline models when predicting games during the 2018-19 season. Furthermore, the results from this table suggest that all models but the Random Forest model performed better when trained using the advanced statistics. The model with the best F1 score was the Logistic Regression model trained with the advanced statistics (0.636), and the worse was the Neural Network trained using the traditional statistics (0.611).

A 5 x 2 cross validated t-test was conducted to test if the performance of the advanced and traditional models were statistically significant. The 5 x 2 cross validate t-test was proposed by Dietterich [7] to address the shortcomings of other model comparison methods such as the re-sampled paired t-test and k-fold cross validated t-test. The null hypothesis for the t-test was taken as the two models having equal performance, and therefore, the null hypothesis must be rejected for there to be a statistical significance between the performance of the models. Table 4.3 shows the p-value and t statistics for each of the models, with a significance level of $\alpha = 0.05$, we are unable to reject the null hypothesis for all models and can conclude that there is not a statistical significance between the advanced and traditional model performance for the matchup modelling technique.

4.2.1 Logistic Regression

The confusion matrices and ROC curves for both Logistic Regression models are displayed in Figure 4.2. Both Logistic Regression models perform equally

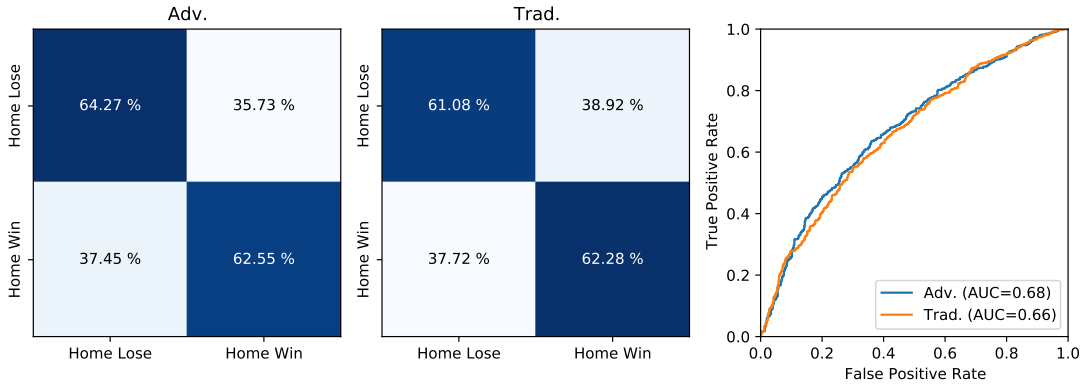

Figure 4.2: Results from the Logistic Regression method on the test set for the matchup modelling technique using both advanced and traditional statistics.

well when predicting home wins, with $\approx 62\%$ of home wins correctly predicted. However, since the dataset is imbalanced with many home win instances (due to the home court advantage), the other $\approx 37\%$ of home wins incorrectly predicted as home losses, account for a large part of the errors made in both models. The performances of the models mostly differ when dealing with home losses, in which the model trained with the advanced statistics is able to do a better job of correctly predicting when a home team will lose, improving on the traditional statistics by $\approx 3\%$, but this difference was not found to be statistically significant.

4.2.2 Random Forest

The Random Forest results differ quite a bit from the Logistic Regression and Neural Network results. Unlike the other models, the Random Forest models were the only models found to perform better on the traditional statistics when using the modelling matchups technique. From the confusion matrices in Figure 4.3, it is clear that both Random Forest models do well when predicting home wins, with $\approx 72\%$ and $\approx 75\%$ of home wins correctly predicted for the advanced and traditional models respectively. This is a significant increase when compared to both of the Logistic Regression models which managed $\approx 62\%$. However, both Random Forest models perform very poorly when predicting home losses, not being able to predict even half of the home losses correctly. In fact, the majority of the predictions the Random Forest models make are home wins. Therefore, it is possible that the Random Forests have not learnt anything meaningful about the underlying problem, and have instead learnt to just predict home wins due to there being so many in the dataset. Even if this is true, the Random Forest

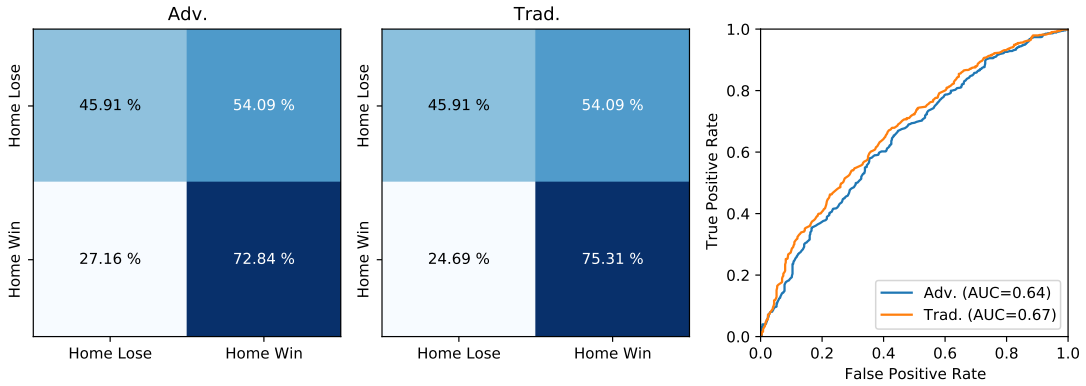

Figure 4.3: Results from the Random Forest method on the test set for the matchup modelling technique using both advanced and traditional statistics.

model trained with the traditional dataset still outperforms the best baseline results in Figure 4.1. Unlike the other models, there is quite a large difference in AUC between the two Random Forests and much less overlap between the ROC curves, with the traditional model's ROC curve consistently outperforming the advanced model's ROC curve. Again, like the Logistic Regression models, the paired t-test did not find the difference between the Random Forests statistically significant.

One of the main motivations behind using a Random Forest model in this thesis was because of its ability to calculate feature importances which provide valuable insights into the underlying problem. The top ten features for both datasets can be seen in Table 4.4. For the advanced features, it was found that the most important feature was a team's *offensive rating* followed closely by their *defensive rating*. The offensive rating statistic is calculated by measuring how many points a team scores per 100 possessions, conversely, the defensive statistic is how many points allowed per 100 possessions. Although the difference in importance between the offensive and defensive rating statistics is small (0.002), this could still possibly suggest a team's offensive abilities are more important than their defensive abilities. For the traditional statistics, the most important feature was found to be the *plus minus* statistic, exceeding the second most important feature (points scored) by 0.02. The plus minus statistic measures the point differential when a player is on the court playing, when aggregating the plus minus for all players in a team, the metric gives the point differential when the team is playing. If a team is very good and they score more points than their opponent, that team will have a positive plus minus and vice versa. This is interesting because the plus minus statistic has a lot in common with the offensive & defensive rating statistics, in that it captures a team's offensive and defensive

Feature	Importance	Feature	Importance
OFF_RATING	0.046	PLUS_MIN	0.061
DEF_RATING	0.044	PTS	0.042
TM_TOV_PCT	0.040	FT_PCT	0.039
PACE	0.039	OREB	0.039
CFGM	0.036	FG_PCT	0.038
UFGA	0.036	STL	0.038
CFGA	0.035	BLK	0.038
EFG_PCT	0.033	FG3_PCT	0.038
UFGM	0.033	TO	0.038
TS_PCT	0.032	PF	0.038

Table 4.4: Advanced (left) and traditional (right) feature importance values rounded to 3 decimal places for the Random Forest matchup modelling technique.

abilities, taking into account the performance of the other team. In other words, both statistics are zero sum, for a team to have a large positive plus minus or offensive rating, the opponent must allow them to score more points. This could possibly suggest that measuring the performance of a team in isolation makes no sense, as the performance is highly dependent on the opposing team.

4.2.3 Neural Network

Finally, the Neural Network matchup modelling results are displayed in Figure 4.4. Like the Logistic Regression model, the Neural Network is able to classify the majority of both classes correctly ($> 60\%$ of instances for each class). Interestingly, the Neural Network models make slightly different errors. The advanced model struggles more with classifying home losses correctly, whereas the traditional model struggles with classifying home wins correctly. Again, similar to the Logistic Regression model, the large number of incorrect home win predictions accounts for the majority of the model's error. This is backed up by the fact that the model with the worse F1 score, the Neural Network trained with the traditional dataset, also has the most incorrect home win predictions, at 39.51% of home wins being incorrectly predicted. Even with the increased complexity of the Neural Network model, the performance of the models was unable to exceed that of the simple Logistic Regression model when classifying both home wins and losses. Given that the simpler Neural Network architectures outperformed the more complex ones and both Logistic Regression models outperformed the Neural Network models, the results could suggest that more complex models are

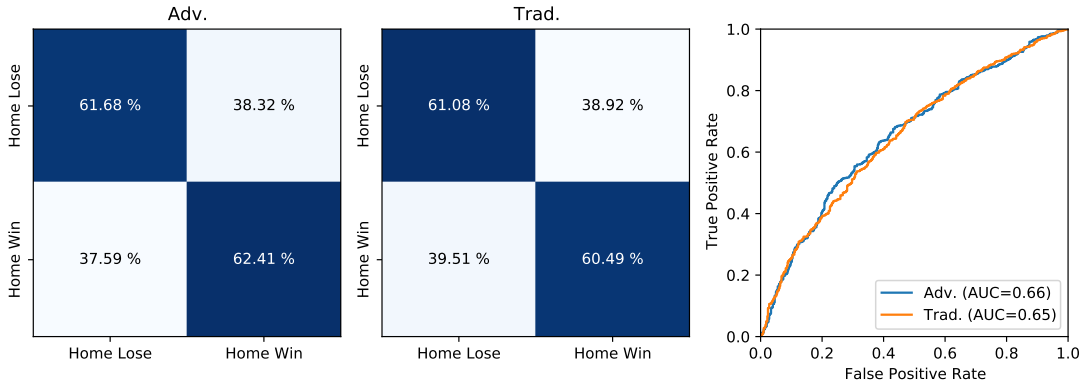

Figure 4.4: Results from the Neural Network method on the test set for the matchup modelling technique using both advanced and traditional statistics.

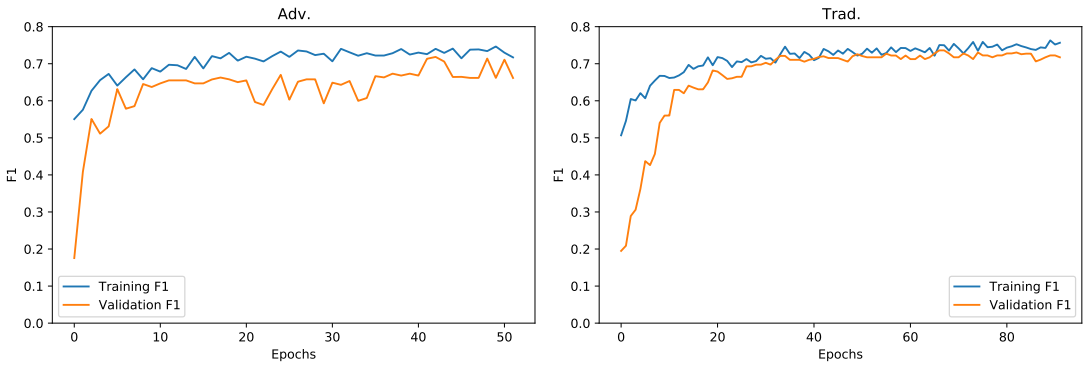

Figure 4.5: Training and validation curves for the Neural Network matchup modelling technique for both advanced and traditional statistics when predicting if the home team will win.

not suitable for modelling basketball matchups.

Figure 4.5 shows the training and validation learning curves for both Neural Network models. 10% of the training data was used as the validation set during training. The traditional datatset model was able to train longer (92 epochs) before the early stopping kicked in, whereas the advanced dataset model was cut off much earlier at 52 epochs. However, the advanced dataset model improved much faster than the traditional dataset model, plateauing at $\approx$ 10 epochs compared to $\approx$ 20 epochs. The advanced model's learning curves does not show any signs of overfitting, as the validation curve consistently stays below the training curve, finishing with an F1 score of 0.662, which is slightly higher than the recorded test set F1 metric of 0.624. However, the traditional model's learning curves are

much closer, coming in contact a few times during training. The validation curve also finished with an F1 score of 0.717, much higher than the test set F1 score of 0.611. It is possible that the traditional model could benefit from a more strict early stopping rule, such that training is cut off earlier to introduce more bias into the model.

4.3 Team Modelling Results

	Advanced (F1)	**Traditional (F1)**
Logistic Regression	0.637	0.628
Random Forest	0.602	0.589
Neural Network	0.635	0.624

Table 4.5: F1 metric results for the three machine learning models using the team modelling technique.

The test set F1 scores for the models implementing the team modelling technique are displayed in Table 4.5. Remembering that machine learning models from the team modelling technique output the probability of a single team winning their next game, not the winner of that game. These F1 scores are calculated from the final predictions, after both team's probability of winning has been obtained from the model and used to obtain the winner of that game. Recall that from the team modelling, the advanced and traditional features have, respectively, 25 and 18 dimensions (Table 3.3). From the results, the best model was the Logistic Regression model trained using the advanced statistics, with an F1 score of 0.637. The worst model was the Random Forest model trained with the traditional statistics, recording an F1 score of 0.589 on the test set, beating the best baseline model's F1 score by 0.028. Unlike the matchup technique models, which had the Random Forest model perform better with the traditional statistics, all the models implementing the team modelling technique were found to perform better when trained on the advanced statistics.

Like the matchup models, the same 5 x 2 cross validated t-test was conducted to test if the difference between the advanced and traditional models performance was statistically significant. The results for this t-test are displayed in Table 4.6. At a significance level of $\alpha = 0.05$, we are unable to reject the null hypothesis for all models, which states that the two models have equal performance. Therefore, we can conclude that there is no statistical significance between the performance of the models when trained on the advanced or traditional datasets for the team modelling technique.

	P-value	t Statistic
Logistic Regression	0.236	-1.348
Random Forest	0.131	-1.807
Neural Network	0.222	-1.394

Table 4.6: 5 x 2 cross validated t-test results between the advanced and traditional models using the team modelling technique.

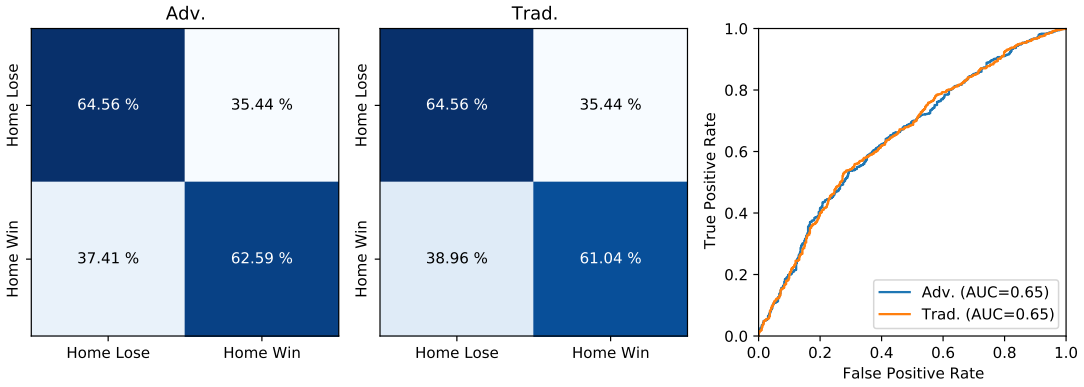

Figure 4.6: Results from the Logistic Regression method on the test set for the team modelling technique using both advanced and traditional statistics.

4.3.1 Logistic Regression

The confusion matrices and ROC curve for the Logistic Regression models is displayed in Figure 4.6. Both models appear to perform just as well as each other when predicting home losses, outperforming home win predictions by around $\approx$ 2% to 3%. Again, the test set used to obtain these results is imbalanced, containing more home win instances than home lose instances and therefore, similar to the Logistic Regression matchup models, the large amount of home wins incorrectly predicted ($\approx$ 37% and $\approx$ 39%) as home losses, account for the majority of the errors made in both models. Overall, the advanced and traditional Logistic Regression models appear to be fairly similar, having a similar AUC score and identical performance when classifying home losses. Since the conducted t-test was unable to reject the null hypothesis at $\alpha = 0.05$, we are unable to conclude that the slight difference in the home win performance of the models is statistically significant.

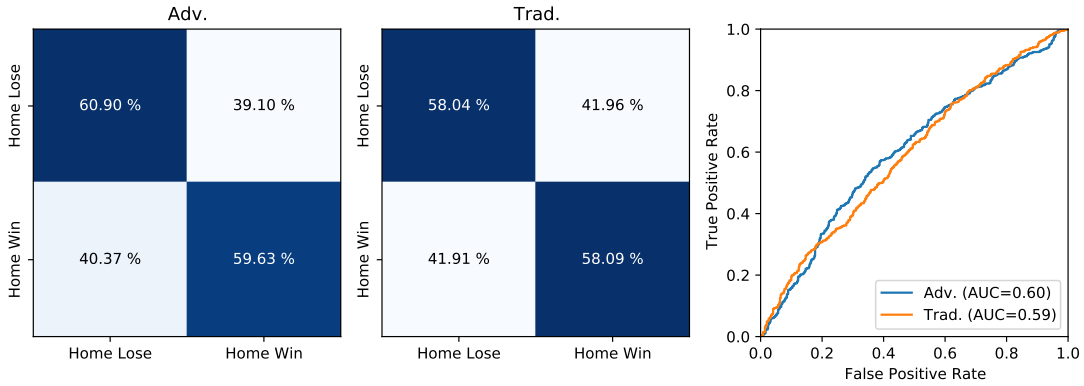

Figure 4.7: Results from the Random Forest method on the test set for the team modelling technique using both advanced and traditional statistics.

4.3.2 Random Forest

Unlike the results from the matchup modelling Random Forests in Figure 4.3, the team modelling Random Forests did not suffer from a large amount of incorrect home lose predictions, as seen from the confusion matrices in Figure 4.7. The team modelling Random Forest results are much more comparable to the team modelling Logistic Regression results in Figure 4.6, in that a strong diagonal in both confusion matrices exists, meaning that the model was able to classify the majority ($> 50\%$) of both classes correctly, however not as well as the Logistic Regression model. It is no coincidence that the model with the worst-test set F1 score, the Random Forest trained with the traditional statistics, also recorded the largest amount of incorrect home win predictions out of all the models in this thesis, at 41.91% of home wins incorrectly predicted. Although the traditional statistics performed worse than the advanced statistics, the t-test results in Table 4.6 show that this is not statistically significant given $\alpha = 0.05$.

An analysis of the Random Forest model's feature importances can be seen in Table 4.7, where the top ten most important features for both datasets is displayed. It is important to note that these are the most important features when predicting the probability that a team will win their next game, not the winner of the game itself. Therefore, the results are not directly comparable to the results in Table 4.4 due to the differing techniques. Interestingly, the most important advanced statistic was found to be a team's *pace*, which measures the number of *possessions* (i.e. the number of times a player of the team had the ball) per 48 minutes of game time. One possible interpretation of this could be that teams that gain possession of the ball frequently have more chances to score points, but the pace statistic doesn't take into account whether or not points were

Feature	Importance	Feature	Importance
PACE	0.048	FG3_PCT	0.068
PTS_OFF_TO	0.048	STL	0.067
PTS_PAINT	0.047	TO	0.066
AST_A	0.046	AST	0.063
PTS_2ND_CHANCE	0.046	FT_PCT	0.062
DIST	0.046	PTS_A	0.062
STL_A	0.046	BLK	0.060
PLUS_MIN	0.044	AST_A	0.059
PTS_A	0.044	STL_A	0.059
REB_A	0.044	FG_PCT	0.058

Table 4.7: Advanced (left) and traditional (right) feature importance values rounded to 3 decimal places for the Random Forest team modelling technique.

scored off that possession. As such, a team who frequently gains possession of the ball but does not score points off that possession, is not better off than a team who can frequently score of fewer possessions. Neither the offensive or defensive rating statistics (the most important features found in the matchup modelling technique) made the top 10 features for the advanced dataset, neglecting these features as important could give reason to the poor performance of the advanced model. For the traditional statistics, the most important feature found was a team's three point field goal percentage, which describes the percentage of three point shots made by a team. Given the rise in popularity of the three point shot and its efficiency, this seems like a suitable metric to measure a team's offensive success, however, the statistic largely ignores other offensive factors.

4.3.3 Neural Network

Lastly, the team modelling Neural Network results are shown in Figure 4.8. Much like the other models, the Neural Networks have a strong diagonal in the confusion matrices, meaning the majority of both classes were able to be correctly predicted. Again, the large amount of incorrect home win predictions accounts for the majority of the model's errors. Even though the advanced model shows a small improvement over the traditional model when classifying home wins, this difference was not found to be significant from the t-test results in Table 4.6. Given the increased complexity of the Neural Network models, the performance was not able to surpass that of the much simpler Logistic Regression models for both the advanced and traditional datasets, much like the matchup modelling technique. Therefore, like the matchup technique, it would be suitable to recom-

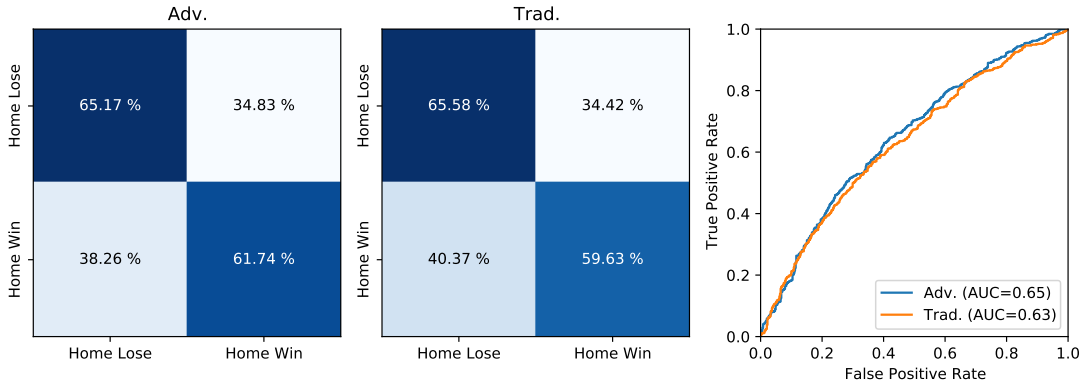

Figure 4.8: Results from the Neural Network method on the test set for the team modelling technique using both advanced and traditional statistics.

mend simpler models such as Logistic Regression over the more complex Neural Network models for modelling teams.

Figure 4.9 shows the training and validation curves for both of the team modelling Neural Networks. The curves are not directly comparable to the curves in Figure 4.5, as the the team modelling Neural Networks are trying to predict if a team will win their next game given their performance in recent games, not predicting the winner of a game between the home and away team. However, from the training and validation curves, the models do not seem to be learning which causes the early stopping to halt the training at around 15 to 20 epochs. The early stopping rule was removed to see if the models were able to learn after training for more epochs. Removing the early stopping callback allowed the models to train for 100 epochs. Unfortunately, after 100 epochs, the models were not better off, the advanced data set ended with a validation F1 score of 0.5549. Although the models do not seem to improve much over training, they still perform reasonably well when compared to the other models. Deeper experimentation with the Neural Network architectures to facilitate better learning could be an opportunity for improving the overall performance of this technique.

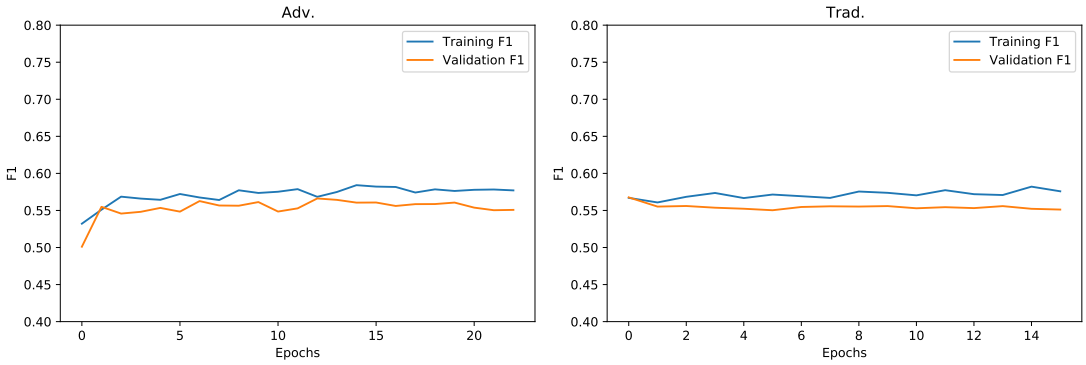

Figure 4.9: Training and validation curves for the Neural Network team modelling technique for both advanced and traditional statistics when predicting if a team will win their next game in the season. Confusion matrices are normalized across the row.

CHAPTER 5

Conclusions & Future Work

Applying machine learning techniques to predict basketball games has become a popular research topic, and as such, many models and techniques have been put forward that utilise traditional basketball statistics to accurately provide game outcome predictions. This thesis has explored the two main sports prediction techniques, namely modelling matchups and modelling teams and applied various machine learning classifiers trained with both advanced and traditional basketball statistics. From the results presented in this thesis, conclusions on the two techniques, the machine learning models and the datasets can be made.

Techniques This thesis presents the two techniques established in previous research and provides a formal definition of how they work and what data they require. The results indicate that neither technique performs significantly better than the other when predicting basketball games. Given that the performance of the techniques is similar and the data used between the models is more or less the same, a possible conclusion could be that either technique is fine for future work. However, the modelling matchups and modelling teams techniques require a different process to get from raw data to final predictions and from an engineering perspective, the matchup modelling technique is far more testable and maintainable than the team modelling technique. This is largely because the matchup modelling technique outputs the winning team directly, unlike the team modelling technique which requires another step to check which team has a higher winning probability. This extra step made the process of testing a costly and error prone operation, as extra pre-processing steps were required to input the correct team features for the game being tested. The matchup modelling technique also allows for easier integration with common machine learning workflows, such as the Scikit-learn API, which makes development much easier. In conclusion, since the results indicate that neither technique outperformed the other, the matchup modelling technique is suggested due to it being a more straight forward, testable and maintainable approach when compared to the team modelling technique.

Machine Learning Models The three machine learning models explored in thesis vary in overall complexity and learning approach, with one being an ensemble learner. While the results differ between the two techniques, the overall conclusion that can be made for both of the techniques is that the more simpler models, such as the Logistic Regression model, are better suited than more complex models for predicting basketball game outcomes. Although the results between the Logistic Regression models and the Neural Network models aren't significantly different, the Logistic Regression model requires far fewer resources to train than a Neural Network. As such, there is no benefit from using a more complex method if the results are not positively impacted from doing so. In other words, why train a complex Neural Network when the same performance could be achieved with much less effort. Another conclusion that can be made with regards to the machine learning models in general, is that the models do provide value over methods that strictly use only historical facts present in the data. This is evident in the fact that *all* machine learning models were able to outperform the baseline models for both techniques.

Datasets Unlike previous research that has neglected the use of the advanced statistics provided by the NBA, this thesis has explored the use of them as training data for basketball prediction models, in hopes of improving the prediction performance. However, even though the results show that the majority of the models performed better using the advanced statistics on the test set, a 5x2 cross validated t test was conducted, comparing the performance of the models trained with the advanced and traditional datasets. The results of the t test show that there is no statistical significance between the performances of the models, and therefore, can conclude that the advanced statistics are as good as the traditional statistics when predicting basketball games. Given this conclusion however, due to the difficulty of obtaining the advanced statistics (available only from the NBA's advanced statistical website) and the extra pre-processing they require, it would be beneficial to instead use the traditional statistics in future works.

Based on the conclusions above, there still exists multiple opportunities for improvement by future works in this area. One such improvement could be the exploration of incrementally training models during the season, allowing the models to adapt quickly to changes during the season. One way of achieving this could be to batch all the games played within a week, retraining the model with this batch. Experimentation would need to be done to find the optimal way of incrementally training, such that the model does not change dramatically between weeks. Another opportunity for improvement could be to take advantage of the time series nature of basketball statistics by training a Neural Network that works on sequential data. A Recurrent Neural Network (RNN) could possibly

learn how a team's performance changes during the season, which could provide a better prediction when predicting the outcome of their next game. A final suggestion could be to stop aggregating the statistics by team and instead look at the player level data. This would require more complex pre-processing and transformations, however, it would be possible to capture the performance of a team at a much higher level of detail. For example, changes to a team roster could be captured as well as which players are injured and which are available to play for a given game. Looking at the player level data will also allow for player impacts to be calculated, allowing the model to adjust predictions if a team has no high impact players available.

References

[1] A.R. Armstrong and M. Carroll. *Sports betting in Australia*. Australian Gambling Research Centre, Australian Institute of Family Studies, Melbourne, 2017.

[2] Kurt Badenhausen. Nba team values 2019: Knicks on top at $4 billion, Feb 2019. URL https://www.forbes.com/sites/kurtbadenhausen/2019/02/06/nba-team-values-2019-knicks-on-top-at-4-billion/.

[3] Matthew Beckler, Hongfei Wang, and M Papamichael. NBA Oracle. *Zuletzt besucht am*, 1(1995):15213, 2013. URL http://www.mbeckler.org/coursework/2008-2009/10701{_}report.pdf.

[4] Rory P. Bunker and Fadi Thabtah. A machine learning framework for sport result prediction. *Applied Computing and Informatics*, 2019. ISSN 22108327. doi: 10.1016/j.aci.2017.09.005.

[5] D. Buursma. Predicting sports events from past results "towards effective betting on football matches". In *Conference Paper, presented at 14th Twente Student Conference on IT, Twente*, Holland, 2011.

[6] Ge Cheng, Zhenyu Zhang, Moses Ntanda Kyebambe, and Nasser Kimbugwe. Predicting the outcome of NBA playoffs based on the maximum entropy principle. *Entropy*, 18(12), 2016. ISSN 10994300. doi: 10.3390/e18120450.

[7] Thomas G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 1998. ISSN 08997667. doi: 10.1162/089976698300017197.

[8] K. Goldsberry. *SprawlBall: A Visual Tour of the New Era of the NBA*. HMH Books, 2019. ISBN 9781328765031. URL https://books.google.com.au/books?id=1yaDDwAAQBAJ.

[9] J. Kahn. Neural network prediction of nfl football games. *World Wide Web Electronic Publication*, pages 9–15, 2003. URL www.scopus.com. Cited By :15.

[10] M. Lewis. *Moneyball: The Art of Winning an Unfair Game*. W. W. Norton, 2004. ISBN 9780393066234. URL https://books.google.com.au/books?id=oIYNBodW-ZEC.

[11] Alan McCabe and Jarrod Trevathan. Artificial intelligence in sports prediction. In *Proceedings - International Conference on Information Technology: New Generations, ITNG 2008*, 2008. ISBN 0769530990. doi: 10.1109/ITNG.2008.203.

[12] Dragan Miljković, Ljubiša Gajić, Aleksandar Kovačević, and Zora Konjović. The use of data mining for basketball matches outcomes prediction. In *SIISY 2010 - 8th IEEE International Symposium on Intelligent Systems and Informatics*, pages 309–312, 2010. ISBN 9781424473946. doi: 10.1109/SISY.2010.5647440.

[13] Michael C. Purucker. Neural network quarterbacking. *IEEE Potentials*, 15 (3):9–15, 1996. URL `www.scopus.com`. Cited By :19.

[14] Russell Valenzuela. *Predicting National Basketball Association game outcomes using ensemble learning techniques*. PhD thesis, California State University, Long Beach, 2018.

[15] Fadi Thabtah, Li Zhang, and Neda Abdelhamid. NBA Game Result Prediction Using Feature Analysis and Machine Learning. *Annals of Data Science*, 2019. ISSN 2198-5804. doi: 10.1007/s40745-018-00189-x.

[16] Krzysztof Trawiński. A fuzzy classification system for prediction of the results of the basketball games. In *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 2010. ISBN 9781424469208. doi: 10.1109/FUZZY.2010.5584399.

APPENDIX A

Original Honours Proposal

Background

The National Basketball Association (NBA) is a North American professional basketball league founded in New York City on June 6 1946. The NBA is one of four major professional sports leagues in the United States and Canada with a total revenue of 8.76 billion US dollars in 2019 according to Forbes magazine [2]. Data and analytics has had a big impact on professional sports with the most famous example being Billy Beanes Moneyball which is an approach to building a competitive baseball team on a small budget by utilising statistics to identify undervalued players (and overvalued players). The NBA has embraced the data and analytics movement (possibly surpassing its use in Major League Baseball) as it's the first American sports league to install player tracking systems in every arena to capture vasts amounts of data for every player in every game. The league also hosts an annual analytics hackathon.

Machine learning has become increasingly popular and with the explosion of data, lower costs in data storage and new technologies its become more feasible to apply these methods to solve a number of problems. The combination of machine learning and sports is quite popular and there has been a number of papers that have already studied it's applications with NBA data. One of which is the NBA Oracle [3], which used aggregated team box scores and four different binary classification algorithms (Support vector machines, linear regression, logistic regression and artificial neural networks) to classify the winning team with up to 73% accuracy. However the NBA Oracle was published in 2013 and doesn't take advantage of the more advanced box score statistics that are available today from the NBA. Another study [6] has taken a different approach and used the principle of maximum entropy to predict game outcomes. This study was able to predict outcomes with up to 74.4% accuracy in 2016 and boasts better performance than any other classical machine learning algorithms.

Predicting the outcome of a game is important for a number of reasons. One

of which is sports betting, which has become quite popular since becoming legal. Another more useful reason would be to discover what features are more important when winning a game. Is it better for a team to be more defensively skilled? Or will taking more shot attempts on offense be the difference between winning and losing? This knowledge is incredibly valuable and has a lot of influence on the game and how the players play. A good example would be the 3 point revolution. When the 3 point line was first introduced in 1979 the efficiency of the shot was largely ignored with an average of 2.8 three point attempts per game, fast forward to the 2018 season the average 3 point attempts per game is 29. Coaches and players realised they can improve their offensive efficiency by changing their shot selection which led to scoring more points. This is a simple example of how basic math has fundamentally altered the game and how it's played. Data and machine learning could uncover less obvious relationships like this between team features and winning games.

Aim

This project aims to improve game outcome predictions by using modern NBA game statistics and machine learning techniques. More specifically to improve on the 73% accuracy produced from the linear regression model (the highest performing one) published in the NBA Oracle paper. I also aim for this project to produce classical machine learning models and deep learning models that outperform the maximum entropy model published in 2016.

This project should also produce insights on the team features that are more important when deciding the outcome of a game. I expect the advanced box score data to produce models that give better results as it provides a more detailed description of team performance than traditional box scores. However I do expect to discover that most features will be redundant when when classifying teams. I also don't expect to develop a model that produces perfect solutions as the game of basketball (and sports in general) has a lot of uncertainty. The end goal is to get the most out of the available data and be able to make the most educated prediction.

Another application of this project could be forecasting / classifying playoff teams throughout the season. A team makes the playoffs if their win loss ratio is in the top 8 for their conference i.e. the teams who win the most games make the playoffs. Combining the prediction model with the upcoming game schedules could provide in season forecasts of playoff teams.

Method

The tasks for this project will follow that of a typical machine learning project.

1. The first stage will be constructing the data set which I plan on scraping from the official NBA statistical website(stats.nba.com).
2. The second stage will be the analysis stage. The purpose of this is to give me a deeper understanding of the data set, potential relationships between features and insight into what pre-processing might be required before training.
3. The final stage of the project will require me to select, train and tune various machine learning models. This will be quite an iterative process as it will require experimenting with each model, tuning various hyper parameters and reporting results. The goal is to produce the best model possible by understanding what worked and what didn't and why.

Software and Hardware Requirements

This project will use Python 3 and a variety of packages such as Pandas, Numpy, Scitkit-learn and Tensorflow. Special hardware includes Nvidia GPUs (access provided by Dr Du Huynh) and super computer / cloud computing access for training models (if required).

APPENDIX B

Scikit-Learn Matchup Transformer

```
class ProcessFeatures(BaseEstimator):
    def __init__(self, avg_cols, return_df=False, window=5):
        self.team_enc = OneHotEncoder(sparse=False)
        self.return_df = return_df
        self.window = window
        self.avg_cols = avg_cols

    def fit(self, X, y=None):
        self._check_df(X)
        self.team_enc.fit(X[['H_ID', 'A_ID']])
        self.t_cols = [
            *('{}_h'.format(idx) for idx
                in self.team_enc.categories_[0]),
            *('{}_a'.format(idx) for idx
                in self.team_enc.categories_[1])
        ]
        return self

    def transform(self, X):
        self._check_df(X)
        X = X.copy() # make a copy to prevent changes being made

        # compute features
        h_won_last = X.apply(
            self._home_won_last, args=[{}], axis=1)
        h_leader = X.apply(self._home_series_leader,
                           args=[defaultdict(list)], axis=1)
        teams = self.team_enc.transform(X[['H_ID', 'A_ID']])
        teams = pd.DataFrame(data=teams, columns=self.t_cols)

        # compute feature averages for home / away teams
        h_avgs = (
            X.groupby('H_ID')
                [['H_{}'.format(col) for col in self.avg_cols]]
                .rolling(self.window).mean()
                .groupby('H_ID')
                .shift(1)
```

```
                .droplevel(0)
                .sort_index()
        )
        a_avgs = (
            X.groupby('A_ID')
                [['A_{}'.format(col) for col in self.avg_cols]]
                .rolling(self.window).mean()
                .groupby('A_ID')
                .shift(1)
                .droplevel(0)
                .sort_index()
        )

        # compute differential stats and rename
        diff = a_avgs - h_avgs.values
        diff.columns = self.avg_cols

        # combine all features
        data = pd.concat([
            diff,
            h_won_last.rename('H_WON_LAST'),
            h_leader.rename('H_LEADER'),
            teams,
            X[['HOME_WIN']]
        ], axis=1)
        self.columns = data.columns

        if self.return_df:
            return data
        else:
            return data.dropna().values

    def fit_transform(self, X, y=None):
        self._check_df(X)
        return self.fit(X).transform(X)

    def _check_df(self, X):
        if not isinstance(X, pd.DataFrame):
            raise ValueError('X must be a pandas DataFrame')

    """PANDAS TRANSFORMATION METHODS"""
    @classmethod
    def _home_series_leader(cls, row, series_history):
        """Is home team the series leader"""
        matchup_id = tuple(sorted([row['H_ID'], row['A_ID']]))

        series = series_history.get(matchup_id, None)
        if series is None:
            home_leader = 1
```

```
        else:
            home_leader = int(
                series.count(
                    row['H_ID']) >= series.count(row['A_ID']))

        series_history[matchup_id].append(
            row['H_ID'] if row['HOME_WIN'] else row['A_ID'])
        return home_leader

    @classmethod
    def _home_won_last(cls, row, last_games):
        """Did home team win last matchup. """
        matchup_id = tuple(sorted([row['H_ID'], row['A_ID']]))

        last_winner = last_games.get(matchup_id, None)
        if last_winner is None:
            home_won = 1
        else:
            home_won = 1 if row['H_ID'] == last_winner else 0

        last_games[matchup_id] = (
            row['H_ID'] if row['HOME_WIN'] else row['A_ID'])
        return home_won
```