Utrecht University

BACHELOR THESIS 7.5 ECTS

NBA Game Predictions based on a Team Composition Model

Author:
Ibrahim EL GARMOUHI

Supervisor:
dr. G.A.W. VREESWIJK

Second Reader:
dr. D. DODER

A thesis submitted in fulfilment of the requirements for the degree of Artificial Intelligence

April 19, 2020

Abstract

Over the years the sports industry has grown into a multi-billion dollar industry in which technology plays an essential role. As the industry started collecting more data, the research into analyzing that data advanced simultaneously. This thesis will focus on developing a model that is going to be used by several learning algorithms to predict the game outcome of future games. The model will extend upon the results of Singh [16] to measure the effect of adding personalized statistics for each player and use team-compositions to predict the outcome of each game. Research shows that Machine learning algorithms perform better than traditional statistical models when trying to capture a latent context or in this case team-composition. Therefore, the learning is done by a set of supervised learning algorithms which consist of Logistic Regression, Neural Networks and Linear Support Vector Machines.

Contents

1 Introduction

Over the last decade, we shifted to a so-called "Data-Driven Society". Back in 2017, The Economist published a story titled, "The world's most valuable resource is no longer oil, but data." Many companies have realized this and are trying to profit from it in every way. The use of Artificial Intelligence (AI) is playing an essential role in this development and has been gaining a lot of attention over the years. One of the industries that is fully taking advantage of this field is the sports industry.
George William James, a famous baseball statistician, founded a system called "sabermetrics" back in 1977. The system is based on an empirical analysis of baseball statistics that measure in-game activity. For long he was ridiculed for looking at a sport from a mathematician's point of view. However, this system revolutionized the way we look at baseball today and ignited several other sports to do the same. The NBA is one of these organizations and developed a similar system in the early 90s called "APBRmetrics", which takes its name from the acronym *APBR* (Association for Professional Basketball Research). This lead to an outburst of quantitative analytics research in basketball. This thesis is inspired by these developments and will combine the *APBRmetrics* system with current AI techniques to predict game outcomes for the NBA competition.

The next part of this introduction will start with explaining the relevance of this research, subsequently, the related work done in this field and lastly quantify the research questions of this thesis in more detail.

1.1 Problem Context

The aim for every NBA team is to win as many as games as possible. To achieve this goal, several elements play a role from the team-budget to the strategy a team plays on the field. The performance of each team and player is quantified by using a set of metrics specific to the player and team. Besides measuring the performance, these statistics can have several other purposes. One of the less mature areas in which this data is being applied is the predictive analytics field. The predictive analytics field tries to make predictions based on historical data. Machine Learning, a subgroup within AI, uses this same approach. The data availability and quality are crucial for these techniques to succeed. The NBA, which will be focusing on in this research provides a readily available database dating as far as 60 years back. Consisting of accurate metrics specific to each team and player making the data quality of a high standard. With databases of this magnitude, you need inventive ways to organize the data so that the learning can happen within acceptable time limits.

The predictions that result from these models, can be used to alter team strategy and optimize player selection beforehand to maximize the win probability through these models. The sports gambling industry highly profits from these findings by adjusting the odds ratios based on forecasted results to maximize the profit for each game.

As shown, the confidence in predictive models has increased excessively and many NBA-teams nowadays employ full-time data scientist as part of their core staff. Something that 15 years ago would be considered absurd. As this is still a rather new research area there is still a lot of value to be gained within the field and other sections of the data mining process.

1.2 Related work

NBA games are known for their difficulty to predict due to the relatively high level of entropy and average score difference of only eight points per game. In a high pace sport like basketball, where scores usually end up above a hundred points, an eight-point difference is not considered high. However, many still ventured into the field of building models based on historical data to predict the outcome of a game.

Cao [2] built a model based on a Naive Bayes Classifier resulting in an accuracy of 65.8% per season. He used multiple consecutive seasons as the training set and singles seasons as the test set. As Artificial Neural Networks (ANN's) become more accessible over the years, they became the subject of many researchers. ANN's were also used to try and predict NBA games in an article written by Bernard Loeffelholz and Earl Bednar [11]. They used a set of 620 games to train a selection of neural networks consisting of feed-forward, radial basis, probabilistic and generalized regression neural networks. They compared the results to the predictions of expert NBA journalists and had significantly better results. The best ANN's were able to predict the correct winner of 74.3% of the games, where the journalist had an average prediction accuracy of 68.7%. Compared to the NFL (National Football League), one of the most analyzed professional sports out there, these are relatively high accuracy's. A research done by Purucker [14] for the NFL, using similar neural structures resulted in accuracy's around 61% compared to an experts score of 72%. Most of these models relied on the box-score statistics of each game to serve as feature-data to train their models.

Bunker and Thabtah [1] took a more general approach and developed a data mining framework using Machine Learning techniques to predict sports results. They determined that the data preparation and feature extraction can have a significant effect on the actual result. Instead of mostly relying on the team identity and box-score statistics they compared work that used expert-selected features [7], to work that used several different subsets of feature-combinations to reach and optimal model [12]. Bunker and Thabtah [1] clearly showed that machine learning models highly depend on the quality of their features, feature-combinations used and the granularity level of the training data.

1.3 Research Questions

The motivation and related work discussed in the previous paragraph is what led to finding an interesting project done by two master students at Stanford University [16]. Sing and Wang developed an interesting way of prepossessing the data before inserting the data into the learning algorithms. The technique is based on using the identity of the players present in each game to serve as the features for the Machine Learning models. This thesis will use the same prepossessing technique and extend

it by adding personalized player-statistics for each player. In this thesis, the technique will be referred to as the team-composition modelling technique. The goal is to give a more accurate description of the player's strengths and weaknesses, and use that to model the win probability.

The feature set used by Sing and Wang [16] was relatively large and no feature selection algorithm was applied on this set. Bunker and Thabtah [1] clearly showed the relevance of this step, especially in the case of large feature sets. Therefore, this thesis will apply selection algorithms on the feature sets created and select the optimal models. The following research questions resulted from these findings and led to this research:

- Does the addition of player-statistics to the team composition modelling technique improve the predictive performance in comparison to only using the player identity?
- Which classification algorithm achieves the highest performance with the team modelling technique out of Multivariate Logistic Regression, Neural Networks and Support Vector Machines?

The techniques selected in the second question were selected on the bases of our problem description and the related work discussed in the previous section. All three techniques differ in their approach but perform well on binary classification problems. Since we are trying to predict the game-outcome (win/lose) of NBA-games these techniques fitted well with the goal of this research.

1.4 Outline

The outline of this thesis will be as follows. Chapter 2 will start by discussing the data mining approach and will include the theoretical basis of the learning techniques used for training. In chapter 3, the data that is collected and preprocessed into the right format will be discussed extensively. Chapter 4 will continue by explaining the experimental setup and methodology used to perform each analysis. Subsequently, chapter 5 will discuss and analyse the results of our experiments. Lastly, chapter 6 will contain the discussion and conclusion, which will answer the research questions, and discuss potential improvements for future work.

2 Theory

In this chapter, an overview of the theory used in this research is given and elaborated. We will start by giving a brief overview of the data mining process. Afterwards, we will continue by discussing the theory behind the learning algorithms used for training. And lastly, we will present the measures used to evaluate and compare the several models in this research.

2.1 Data Mining Process

"The objective of data mining is to identify valid novel, potentially useful, and understandable correlations and patterns in existing data" [3]. It is a process of knowledge discovery in data that is not telling in its raw form. The data mining process can be roughly divided into two high-level steps: the data preparation and the actual data mining.

In the Algorithms that were discussed in chapter 2.1, training-data was mentioned several times. Training-data can have many different forms and rarely is equal to the raw data that you import. This restructuring is done according to rules that you define beforehand. These rules are defined within your feature definitions. A feature is an input column for your model to use for training. The generation of a feature definition is mainly done by using domain knowledge. The goal is to develop features that have a positive impact on your model performance. This information is not known beforehand and therefore it is a discovery process to whether a feature has a positive effect in the model.

In order to avoid that our final model has to process a large number of irrelevant features, we apply some form of feature selection. Feature selection is done according to metrics that give an accurate description of the model's performance. This area alone is an extensive research field from which complex techniques resulted to efficiently select the best features [4]. In this research, we will use a relatively simple technique called: recursive feature elimination. This technique recursively combines and eliminates features according to a feature-importance metric which can be manually defined. The metric has to indicate the predictive performance of the feature in question. The set of features that result from the selection will then be used in the final model and the irrelevant ones will be deleted.

2.2 Machine Learning

Machine learning algorithms get a lot of attention nowadays. An interesting project that Google started in 2007, is the Scikit-learn project [13]. The Scikit-learn project consists of a large library of several AI-techniques and other statistical approaches implemented and optimized for your hardware. The library also leaves a lot of room for parameter customization and optimization, which makes the library generally

applicable for several use-cases. In our case, we are looking at a binary classification problem for which we chose the following algorithms. The choice was mainly based on the wide use of the algorithms by the scientific community in related problem contexts.

2.2.1 Multivariate Logistic Regression

Logistic regression is part of the supervised learning techniques which rely on labelled data to fit the sample points in a data-set. To fully understand logistic regression, we will give a brief overview of linear regression. Linear regression is the simplest and most extensively used statistical technique for predictive modelling. In short, the goal is to fit a line that is as close as possible to the given sample points. The most common method for fitting this line is the Ordinary Least Squares(OLS) approach. The goal of this approach is to minimize the sum of squared errors (SSE) between the real value (y_i) and the predicted value ($\hat{y}_i$). Figure 2.1 shows an example of a fitted line with the errors visualized.

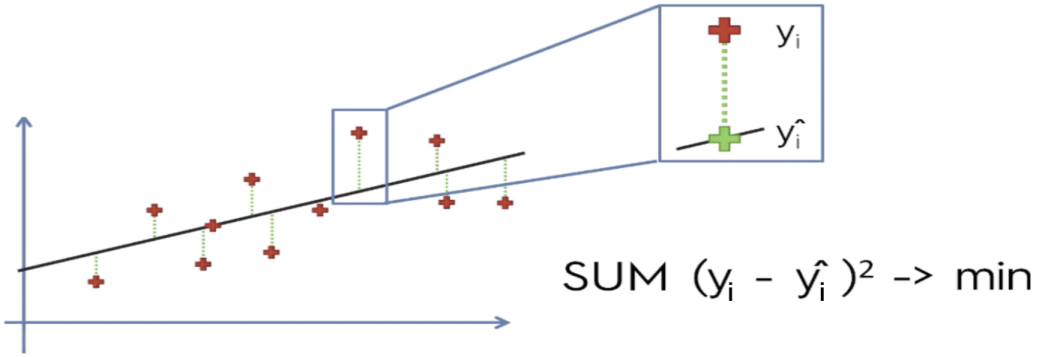

FIGURE 2.1: Simple Linear Regression

A variation of the regression model explained above is the logistic regression model. This model predicts the probability that a given value belongs to a certain category. In the case of binomial logistic regression, the point can belong to two categories, either a "1" or a "0". The main difference with linear regression is that instead of fitting a line to the given sample points, logistic regression tries to fit the sigmoid function (2.1).

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

For the function to become applicable in classification problems a threshold between 0 and 1 must be specified. This threshold is determined according to the goal of the classification. A trade-off between precision and recall is often made to optimize the ideal threshold for a specific problem context. In our case, the threshold is set to $P = 0.5$ since we are looking at a win or lose scenario in a basketball match. If we were using linear regression the function would look like (2.2) in which h denotes the hypothesis function which is the same as the predicted variable. The β variables are the regression coefficients, which we will want to estimate.

$$h(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} = \beta^T x_i \tag{2.2}$$

The result of the following equation would likely consist of continues values. Therefore we modify the sigmoid function (2.1) to the following:

$$p(x_i) = g(\beta^T x_i) = \frac{1}{1 + e^{-\beta^T x_i}} \tag{2.3}$$

As the x-term approaches infinity g approaches 1, likewise as x gets smaller g approaches zero, creating the odds function contained between zero and one. Equation 2.3 can be manipulated to:

$$log(\frac{p(x_i)}{1 - p(x_i)}) = \beta^T x_i \tag{2.4}$$

This value is called the log-odds which has a linear relation to x_i. As with a linear model the β^T gives a representation of the y-change when increasing x_i by a certain amount. However, with logistic regression $p(x_i)$ is not linearly related to x_i. The amount of change to $p(x_i)$ will depend on the values of x_i. There is still a positive dependence between these variables, only not linear. The coefficient vector is still unknown and needs to be estimated. The general method used in this case is the maximum likelihood technique. The conditional probabilities in case of binomial logistic regression will be given by:

$$P(y_i = 1|x_i; \beta) = p(x_i) \tag{2.5}$$

$$P(y_i = 0|x_i; \beta) = 1 - p(x_i) \tag{2.6}$$

$$P(y_i|x_i; \beta) = (p(x_i))^{y_i}(1 - p(x_i))^{1-y_i} \tag{2.7}$$

The likelihood will be a product of all samples in the data-set resulting in the following definition:

$$L(\beta) = \prod_{i=1}^{n} P(y_i|x_i; \beta) \tag{2.8}$$

The likelihood function can be defined as the likelihood that our model will correctly predict the desired y value. Therefore, the goal is to maximize this likelihood by estimating β. For computational reasons we use the following log-likelihood function for the maximization:

$$l(\beta) = log(L(\beta)) = \sum_{i=1}^{n} y_i log(p(x_i)) + (1 - y_i) log(1 - p(x_i)) \tag{2.9}$$

The maximizing can be done with several techniques. The Scikit-learn library offers a selection of iterative solvers with their built-in logistic regression algorithm. The most commonly used solver is stochastic gradient descent (SGD). The solver we will be using this research is based on Newton's method for parameter estimation [9]. This method maximizes function (2.9) by finding the partial derivatives of $l(\beta)$ and setting them equal to 0. At this point, we are looking for the critical point of the partial derivatives. This critical point will be the maximum of our log-likelihood.

2.2.2 Feedforward Neural Networks

An artificial neural network (ANN) consists of multiple nodes named perceptrons. The perceptron was invented in 1957 by Frank Rosenblatt [15] at Cornell University.

The perceptron is formally defined by the following equation:

$$y = sign(\sum_{i=1}^{N} w_i x_i - \theta) = sign(w^T x - \theta) \tag{2.10}$$

where w is the weight vector and θ the bias-term/threshold. The input is defined by the x vector which is multiplied by the weights. This produces the output y. In a binary classification problem, this single perceptron is initiated by randomly allocating the weight values. The perceptron accepts the input-data (x) and multiplies it by the weights and confirms whether the y-label is similar to the predicted y. The prediction is done according to the activation threshold, which is defined by θ. For example, if θ is zero the weighted output must produce a positive value to classify as a 1 and a negative value to classify as a zero. If the prediction is correct the weights are considered satisfactory and no changes are made. In the event of mislabeling the input data, the weights get adjusted to reduce the error. This process continues until the function reaches a point where every point is classified correctly. This algorithm is often used to classify linear separable classes.

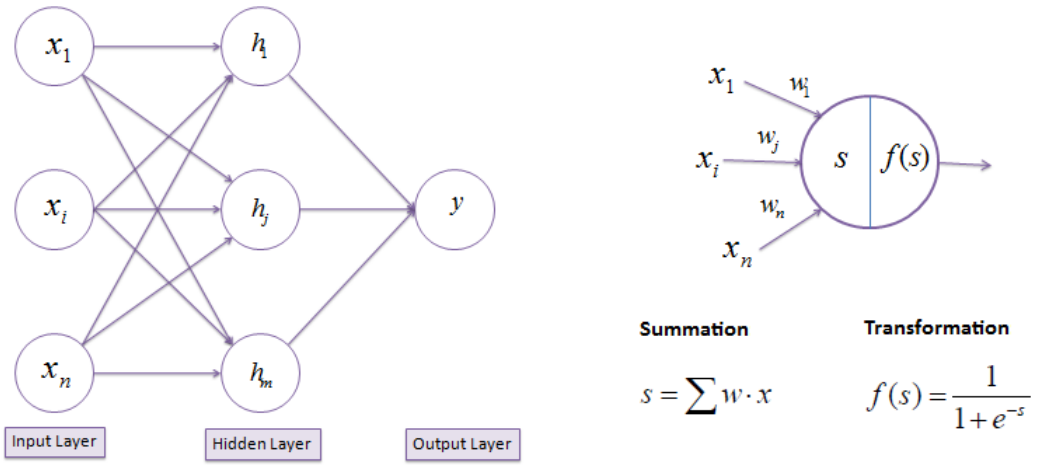

FIGURE 2.2: Multilayer Perceptron Schematic

The neural network we are using in this research is a multi-layer perceptron (MLP), with instead of a linear threshold we will use a sigmoid activation function. This model consists of at least 3 nodes. The nodes are arranged in layers which consist of the input, hidden and output-layer (figure 2.2). Each node in the hidden-layer(s) is a single perceptron which connect to the next perceptron or output-node. The training process of an MLP is done through the backpropagation algorithm. This algorithm consist of two phases named the forward phase and backward phase. In the forward phase, the desired output corresponding to the given inputs are evaluated. In the backward phase, partial derivatives of a cost function with respects to the different parameters are propagated back through the network. This process continues until the error is at the lowest value (convergence).

2.2.3 Support Vector Machines

Support vector machines (SVM's) are responsible for finding the hyper-plane that separate different kind of classes and maximize the margin. The margins are the perpendicular distances of the points closes to the line. Figure (2.3) illustrates an example of a linear separable data-set in a 2-dimensional space. The two main requisites that any SVM must follow are the following:

1. Only look into lines (hyperplanes) that classify the classes correctly.
2. Pick the line that maximizes the margin

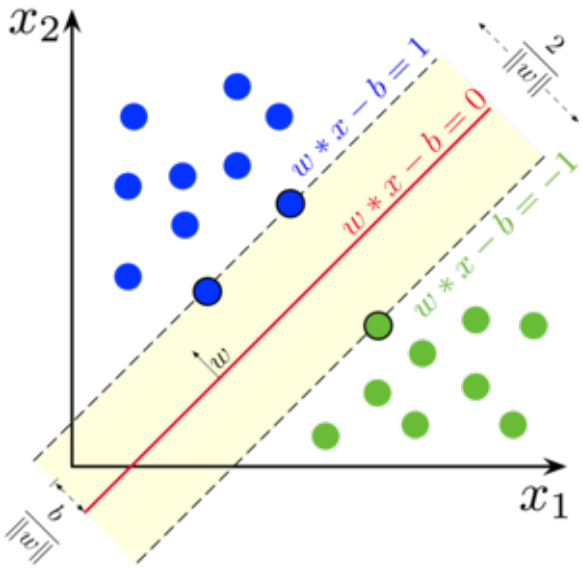

FIGURE 2.3: Support Vector Machine with visualized decision boundary and margin $||w||$.

In the example of figure 2.3, the data is linearly separable and the margin can be found by mathematically finding the perpendicular distance from each point to the median (red line). The smallest distances on both sides form your margin-points, which determine your decision boundaries (dotted lines). This all works fine with the example of figure 2.3. In data where that is not linearly separable, we apply the so-called kernel-trick. The idea behind the kernel-trick is to model data that is not linearly separable in the n'th dimension your working, to a higher dimension where that is the case. To intuitively grasp this figure 2.4 shows an example of a data-set that is not linearly separable in the second dimension but is in the third dimension. Te make this mathematically feasible we do not compute the exact transformation of each point to a higher dimension. Like with figure 2.3, we only need to find the inner product in that higher dimension [6]. This makes it computationally more scalable and a good candidate for complex data-sets.

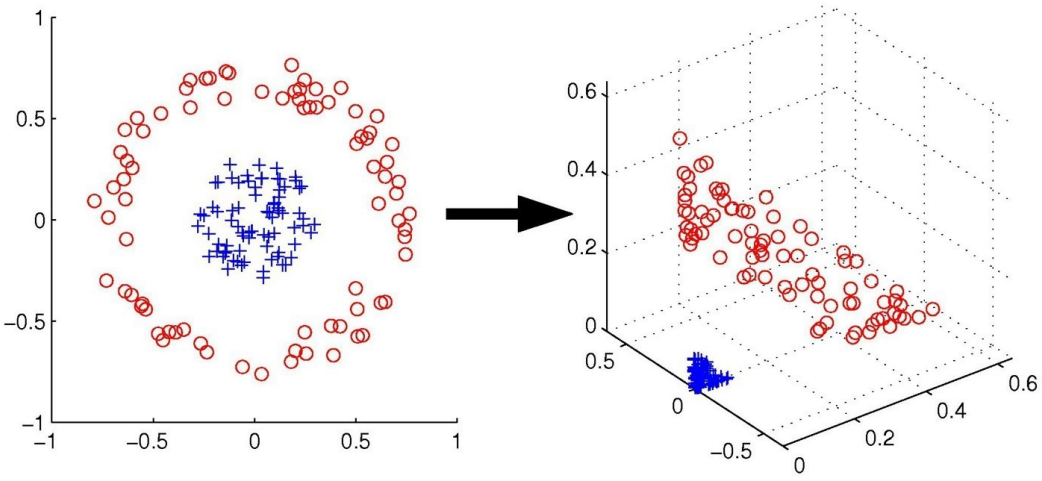

FIGURE 2.4: Kernel trick applied for transformation from 2D-space to 3D-space

2.3 Measures

As discussed in paragraph 2.1 the evaluation of each model is done according to performance measures. Each problem context has its own set of customized performance measures. For example, a regression problem uses different measures than a classification problem. Since we are dealing with a binary (win/lose) classification problem in this research, our focus will be on measures relevant for that specific category.

		Predicted Class	
		True	False
Correct Class	True	True Positive (TN)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

TABLE 2.1: Confusion matrix binary classification

The confusion matrix of a binary classification problem is shown in table (2.1). From this table, you can deduct several more refined measures that can be used in the performance evaluation of a model. The metrics that will be used in his research are listed below:

- **Accuracy:** One of the most intuitive metrics is the accuracy. The metric is defined as: $(TP + TN)/(TP + TN + FN + FN)$ which informally means that the accuracy describes the number of points that were correctly classified as a ratio of the total points that were classified. The accuracy is only effective in a balanced data-set. For example, in a data-set where the majority of the samples is classified as one of the classes the accuracy can give you an imprecise display of your model's performance.

- **Recall:** The recall is a metric that measures the amount of correctly classified positive points. So in a data-set where you have to classify if someone has cancer. The recall tells us the proportion of patients that have cancer and were classified by the model as having cancer. The formal definition is: $TP/(TP + FN)$.

- **Precision:** Using the same cancer diagnostic example as before. The precision tells us what ratio of patients that we're diagnosed by the model as having cancer, actually have cancer. So the amount of positives that were predicted correctly. The formal definition is therefore: $TP/(TP + FP)$.

- **F1-score:** The F1-score was introduced with the idea of capturing both the precision and recall into one defining measure. Since some cases require the precision and other recall, the F1-score solves this problem by combining those. The F1-score has, therefore no intuitive definition. The formal definition is the following:$2 * ((Precision * Recall)/(Precision + Recall))$. A model that makes perfect predictions will have an F1-score of 1 and a poor classifier will be closer to zero.

- **AUC-score:** AUC stands for Area Under ROC Curve. The ROC curve shows the performance of the classifier at various thresholds. The horizontal and vertical axes in the ROC function are represented by the true-positive rate (precision) and the false-positive rate respectively. An ideal ROC-curve pulls towards the left corner, which means that the larger the AUC the better the classifying performance. The AUC can be interpreted as the probability that the model classifies a random positive point more highly than a random negative sample point. An ideal AUC is close to 1, meaning that the model almost makes perfect classification. The AUC is insensitive to unbalanced data-sets, making the AUC a highly effective metric to use.

Model comparison can then be done using one of the aforementioned metrics. The statistical significance between these scores, however, is not always clear. Thomas Dietterich [5] wrote a paper on using statistical hypothesis tests to compare supervised classifiers. The recommendation for paired nominal data was a fairly new statistic called McNemar. The McNemar test uses the contingency table 2.2 of two binary classifiers to calculate the p-values. It is reporting on how many correct and incorrect predictions the classifiers made with regards to the label values. By setting an alpha boundary, the McNemar statistic can then be used to compare models for statistical significance.

		Classifier 2	
		Correct	Incorrect
Classifier 1	Correct	Correct/Correct	Correct/Incorrect
	Incorrect	Incorrect/Correct	Incorrect/Incorrect

TABLE 2.2: Contingency table of two binary classifiers

3 Data

This chapter will discuss the steps relevant for converting the raw data we import to the structured feature data that is being used by the learning models. We will start by showing the unstructured data we import and subsequently the preprocessing will be discussed in paragraph 3.2.

3.1 Data Import

The data that is being used in this research consists of game-data and player-data. The game-data consists of the home-team, away-team, the score and the date at which the game took place (Appendix A.1). The game data was filtered out to only include regular-season games since players often play and behave differently during the playoffs. The player-data consists of advanced statistics specific to each player aggregated over a specific season. In Appendix A.2 an overview of the advanced statistics for a random is player presented. The statistics are based on a one-season data-set and are generated by the NBA. As an addition to the advanced statistics of each player, we also imported the season totals for each player (A.3). These are the total scores for a set of metrics aggregated over one season. From these total scores, two additional statistics were computed and used in the model training. The data was aggregated utilising a web-scraper that uses `https://www.basketball-reference.com/` as its source. The complete set of data that was imported consisted of the seasons 2013/2014 till 2017/2018 for both the game and player data.

The advanced player statistics data and the game data ware mostly ready to use and only needed to be structured in the right way. The highlighted statistics (appendix A.2) are part of the final set that was used to train our models. The selection was made based on expert-knowledge, to prevent the models from getting too much irrelevant data. A more comprehensive explanation will be given in the methodology. The final set of statistics that were computed and used in the models are described down below:

- **minutes_played:** the number of minutes played in a season for a specific team
- **points_scored:** an average of the amount of points scored per game over the course of one season in specific team
- **usage_percentage:** usage percentage is an estimate of the percentage of team plays used by a player while he was on the floor
- **player_efficiency_rating (PER):** PER strives to measure a per-minute performance of each player based on a detailed formula derived by John Hollinger in 1971. The league-average PER is always 15.00, which allows a performance comparison for each player over multiple seasons.

- **win_shares:** an estimate of the percentage of team plays used by a player while he was on the floor (for a specific team over the course of one season)
- **assist_percentage:** an estimate of the percentage of teammate field goals a player assisted while he was on the floor (for a specific team over the course of one season)
- **block_percentage:** an estimate of the percentage of opponent two-point field goals attempts blocker by the player while he was on the floor (for a specific team over the course of one season)
- **effective_field_goal_percentage (eFG%):** The eFG% was introduced to give an accurate measure to account for the fact that three-point field goals way heavier than normal field goals. It is calculated by the following formula:

$$eFG\% = \frac{field\_goals\_made + (0.5 * 3point\_field\_goals\_made)}{field\_goal\_attempts} \tag{3.1}$$

These metrics are imported and calculated for each player and each team that the player played for in a specific season.

3.2 Preprocessing

All the necessary data is imported at this point and will be restructured according to the team-composition technique described by Sing [16]. A few additional adjustments will be made to improve the quality of the training data.

3.2.1 Data Restructuring

The input data for the models in the sci-kit learn library need to be in the form of a 2-dimensional array. Each row has to have at least one column and depending on the amount features more. In this array, each column forms a feature for the model to train with. The y-labels (actual values) will consist of one column where each entity is a single binary value. The team-composition technique represents a team by making one large horizontal array of all the players in the NBA. Each player has it designated location in that array and will be used to indicate the presence of that player in a game. This is done by setting all values to zero except for the players that are present in that game. In a game where team A plays against team B, both teams have their array which describes their team. The concatenated arrays of both teams form the input for one sample in our training data (figure 3.1).

Figure 3.1 represents the data if we only model the identity of the players. This is represented by the ones in the arrays. One of the main objectives of this research to add player-statistics to the model. Our approach will be to replace the ones in the array by a personal metric of the player. The metrics that were discussed in 3.1 will be used for this purpose. The results will be a set of matrices X like in figure 3.1. Every matrix will be represented by a different statistic. The hypothesis function 3.2 in our regression model will then be a concatenation of these matrices multiplied by their regression coefficient vector. Each statistic has its input-array which is equal to the amount of players times two.

$$y(x_{stat}) = \beta^T x_{stat_1} + \beta^T x_{stat_2} + ... + \beta^T x_{stat_n} \tag{3.2}$$

$$
\begin{array}{c} \\ x_i^0 \\ x_i^1 \\ x_i^2 \\ \dots \\ x_i^n \end{array}
\begin{array}{c}
\begin{array}{ccccc} f_1 & f_2 & f_3 & \dots & f_p \end{array} \\
\left[\begin{array}{ccccc}
0 & 1 & 0 & \dots & 1 \\
0 & 1 & 0 & \dots & 0 \\
0 & 0 & 1 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots \\
0 & 0 & 0 & \dots & 1
\end{array}\right]
\end{array}
\begin{array}{c} \\ y^0 \\ y^1 \\ y^2 \\ \dots \\ y^n \end{array}
\begin{array}{c}
\begin{array}{c} y_i \end{array} \\
\left[\begin{array}{c}
0 \\ 1 \\ 1 \\ \dots \\ 0
\end{array}\right]
\end{array}
$$

FIGURE 3.1: The input-matrix X and the y-labels which represent win(1) or lose(0). The length of x_i, p is the length of the player-array times two. The number of training samples is equal to the number of rows (n).

3.2.2 Data Cleaning and Transformation

The goal is to get a representation of the team-players and their statistics for each unique game. During the data exploration phase, we discovered that a large chunk of the players switch teams during each season. This can even happen multiple times in one season for some players. Since the data is imported per season, you can get duplicate player files since our statistics are always calculated for each team over the course of one season. Sing [16] did not account for this problem, which can lead to inaccurate predictions later on. In the final data set we used, this problem is resolved by retrieving the player-roster of each unique game and defining the identity array with that information.

Since we are trying to predict the outcome of games according to historical data, the use of current season player-statistics is not an option. The statistics that are being used to describe the performance of the players are the statistics of the last known season. Which means that for new players in a season, we do not have any known statistics. According to the official NBA website, a team makes an average of 9 player transaction per season. These are not all rookies(new players in the NBA), so for most of these players there is some historical data available. However, the average of rookies per season in our data-set is around 88. This about 1/10 of the total amount of players in the NBA for a specific season. In the statistic arrays, this will lead to around 1/10 of statistics missing over the entire player-set. To overcome this potential problem we tested two methods to fill the missing data and see whether the missing data influences the eventual results at all. This was done by training all three learning algorithms for each feature (player-statistic) on our development set. The first method we tried was a simple team-average approach, in which the missing values are filled by the team average of the metric in question. The McNemar test was used to evaluate whether the models with filled averages had a significantly higher performance. The models with the filled averages had no significant improvements on our development-set, which lead us to discard this method.

The second approach was inspired by Koren [10] who used a matrix factorization technique for recommender systems. In figure 3.2 an example is shown of a problem where we are trying to fill the missing values of each user/rating relation. As described in Koren's article [10], matrix factorization performs relatively well in this problem context. The idea for our approach was to replace the user and item variables by team-players and calculate a ratio between each combination in the table to represent a score. You would end with some ratio's missing and we would able to solve that by applying this technique. Computationally this would lead to no

problems and the result would be a filled table with all the missing values resolved. However, the essence of matrix factorization in this context, is that tries to model some latent features that a user has with the item in question. In our case, both the x and y variables are the same. Meaning you are trying to model features that players have with themselves, which theoretically is not possible. This led us to abandon this approach as it requires more research to make it functional. At this point, we allowed the missing in the final set to exist and accept the potential inaccuracies.

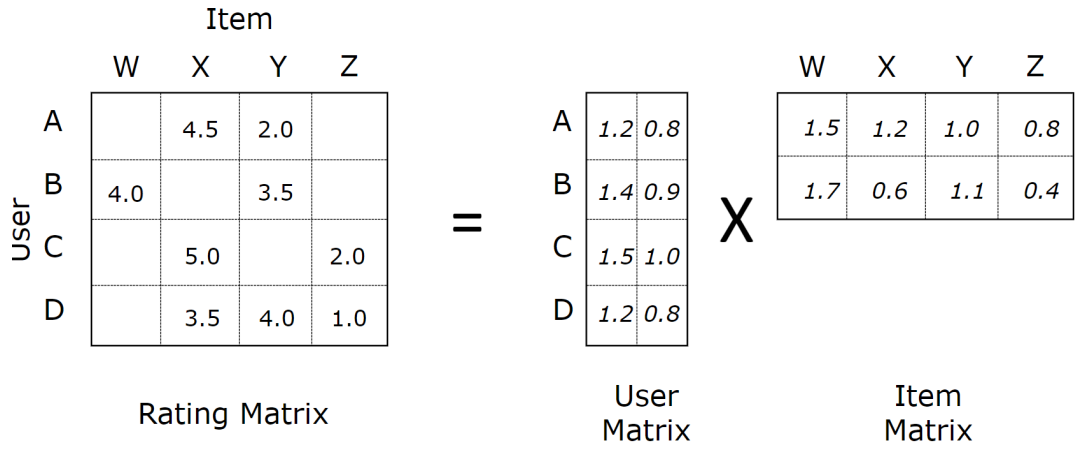

FIGURE 3.2: Matrix factorization example on Netflix user rating problem

Finally, the data is now organized and structured in the arrays as described. In a multi-feature data set like ours, machine learning models tend to weigh certain features heavier in case of range differences [8]. For example, if all features consist of percentages, the range would be the same. However, in cases where one feature is ranged between 0 and 1000, this feature would get an unfair advantage. In some cases where features are dependent on each other, this allowed. However, this is not the case in our context. To prevent this from happening normalization is applied. The normalization technique we used is called min-max (3.3) and is one of the most commonly used normalization techniques. This was done for every player-metric used, therefore eliminating the unfair advantage a feature might have.

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{3.3}$$

4 Methodology

This chapter will describe the procedures taken to perform the experiments in this research. All experiments were conducted in Python 3.7 with the use of some additional packages. The sci-kit learn package provided the machine learning algorithms which then were customized to our problem context. The importing of the raw data was done using a scraper which uses https://www.basketball-reference.com/ as its source. All other steps were done manually and can be found in appendix A.2.

4.1 Training and testing

Before training and testing can begin we must split up the data and define a domain in which both can take place. Since we are dealing with NBA-competition data, the complete data-set will be split into separate seasons to keep that domain boundary intact, as opposed to Sing [16] who shuffled the complete data-set and split it into training and testing. This can lead to biased predictions based on the use of future data which is not available in a realistic scenario. The training data must consist of historical games with regards to the test data. Therefore, testing will be done on complete seasons using one or more preceding season for training. For example, if we are trying to predict the 2017/2018 season, one measurement will use the 2016/2017 season for training and the other measurement will be based on all preceding seasons available. This allows us to compare whether a larger historical training set performs better than a smaller both more recent training set.

4.2 Feature selection and parameter tuning

The preprocessed data contains a set of features consisting of 8 player-statistic based features and 1 identity-feature. Each of these features is organized with the same technique, which means they all have the length of the player-array (867) times two. The learning algorithms technically view each feature as 1734 single features. In the combined 9-feature data, this results in an excessively large data-set, which can significantly degrade your model [4]. The essence of the modelling technique we are using lies in the representation of the full player-array filled with a variety of metrics. Therefore the decision was made to keep the low-level feature columns intact and select only from the 9 high-level features. We applied a recursive feature elimination technique, simultaneously combined with a grid search parameter tuning technique. This allows us to keep the 9 high-level features intact and only select between them. The idea behind this approach is to optimize the parameters for each combination of features we test. To make this computationally feasible we used logistic regression for this process. Since we are dealing with a balanced data-set the accuracy was used as the ranking metric in this selection process. The fitting time of the logistic regression model was significantly lower allowing us to use this model to run an exhaustive grid search with each combination of features.

The best performing feature-combination that resulted from the elimination process, was used to grid-search optimize the parameters for the neural network and the support vector machine. The parameters used in the grid searches are selected by trying out a few sample parameters to see which range would be best to run the grid-search on. The grid-search technique we used, exhaustively generates candidates from the grid of parameter values we determined (table 4.1) and runs the model. To prevent us from picking an overfitted candidate a 5-fold cross-validation is applied with every iteration. The models that resulted from this process were used to fit and predict the scores on the complete data set. The baseline measurements which only include the identity-feature used the same parameter optimization as the best performing feature model. A comparison between those models can then be made to answer the research questions of this paper.

Algorithm	**Grid**
Neural Network	hidden_layer_sizes: (100,100),(100,80),(100,50),(100,20), (50,50),(50,40),(50,20),(100,1),(50,1) alpha: 0.0001,0.001 activation: 'logistic', 'identity', 'tanh' solver : 'adam', 'sgd'
Logistic Regression	C: 1, 0.5, 0.1 solver : 'newton-cg', 'lbfgs', 'sag', 'saga', 'liblinear'
Support Vector Machine	C: 0.1, 0.3, 0.7, 1, 1.2, 1.5

TABLE 4.1: Parameter grids used in the grid search with C and alpha as the regularization parameters

5 Analysis

In this chapter, we will analyze the results accumulated from applying the steps discussed in the methodology chapter. We will start by showing the results of the feature selection and parameter optimization process, which lead to our final models. Next, we will continue by using the optimized parameters to perform our baseline measurements and compare those to the best feature model scores. The last step will be to determine which learning algorithm had the highest performance on our data.

5.1 Feature selection and parameter optimization

The parameter optimization was done on a subset of the complete data-set. As discussed in the methodology, our goal is to make predictions using complete seasons for training and testing. The development set we used, therefore, consisted of the complete 2014/2015 season of which the feature selection was based on the scores for the 2015/2016 season. The result of the feature elimination is shown in the first row of table 5.1. From the 9 features, 511 permutations were generated in which the identity array is included. For all 511 permutations, an exhaustive grid-search was executed with the parameter grid from table 4.1. The result was a set of 511 different feature models optimized and cross-validated on the 2014/2015 season data (see score-logs appendix A.2). These models were used to predict the 2015/2016 season from which the best performing model is shown in table 5.1. The feature-set of this final model was reduced to three features consisting of usage_percentage, win_shares and the assists_percentage.

Algorithm	Best parameters	Score 2015/2016 season
Logistic Regression	solver : newton-cg C: 0.1	0.656
Neural Network	alpha : 0.0001 activation : logistic hidden_layer_sizes : (50,40)	0.650
Support Vector Machine	C : 0.1	0.658

TABLE 5.1: Accuracy Results of grid-search parameter tuning using the 2014/2015 for fitting with the following feature-set: [*usage_percentage*, *win_shares*, *assists_percentage*]

The best performing feature combination was used to run an exhaustive grid-search for the neural network and support vector machine (table 5.1). As with our logistic regression model fitting, was done on the 2014/2015 season with a 5-fold cross-validation. The 2015/2016 season was used to measure the predictive performance of these optimized models. A comparison between these models for significance will be done in paragraph 5.2 and 5.3.

5.2 Feature performance

In table 5.2 till 5.7 a comparison between the baseline models and the best scoring feature models is shown on the complete data-set. This data-set set consists of five seasons starting at the 2013/2014 season. The model performance is illustrated using three metrics discussed in paragraph 2.3. The metrics were determined using the confusion-matrix of each classifier. As the performance of the model is measured by using the test-cores, the training-scores will be shown in the extended results (appendix A.1). The *p*-value between each pair of models is calculated with the use of the McNemar test statistic. The alpha-value (significance level) was set to 0.05.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2014/2015	0.609	0.708	0.571	0.606	0.707	0.566	0.683
2015/2016	0.640	0.730	0.600	0.659	0.741	0.617	0.024*
2016/2017	0.625	0.729	0.577	0.623	0.733	0.569	0.752
2017/2018	0.605	0.723	0.551	0.613	0.729	0.560	0.235

TABLE 5.2: Logistic Regression scores using **one** preceding season for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2015/2016	0.646	0.729	0.612	0.659	0.735	0.627	0.060
2016/2017	0.622	0.722	0.578	0.631	0.727	0.589	0.508
2017/2018	0.604	0.715	0.556	0.599	0.704	0.558	0.805

TABLE 5.3: Logistic Regression scores using **all** preceding seasons available for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

Table 5.2 displays that the logistic regression models using the composition technique perform better than a random model. However, the additional features that were implemented show no significant improvement to the model. The only model that showed a significant improvement by adding the features was the one tested on the 2015/2016 season. Table 5.3 shows the results of using a multi-season training-set to fit the models. Like with the single-season training, there are no significant improvements by using the feature data.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2014/2015	0.615	0.714	0.577	0.617	0.701	0.589	0.807
2015/2016	0.642	0.711	0.620	0.650	0.716	0.629	0.368
2016/2017	0.615	0.698	0.586	0.628	0.712	0.596	0.275
2017/2018	0.609	0.715	0.565	0.626	0.726	0.583	0.010*

TABLE 5.4: Feedforward neural network scores using **one** preceding season for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2015/2016	0.659	0.732	0.631	0.666	0.728	0.646	0.396
2016/2017	0.624	0.715	0.586	0.621	0.713	0.584	0.922
2017/2018	0.597	0.707	0.551	0.594	0.693	0.558	0.888

TABLE 5.5: Feedforward neural network scores using **all** preceding seasons available for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

Table 5.4 depicts the results of our neural network performance. By only assessing the accuracy, F1-score and the AUC-score the feature seems to consistently perform better. However, after applying the McNemar test the only model that a significant improvement was the 2016/2017 trained model. The models that were trained on multiple season in table 5.5 showed almost identical performance scores. Between the baseline models and the feature models, there are no significant improvements present.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2014/2015	0.614	0.703	0.582	0.616	0.703	0.584	0.784
2015/2016	0.660	0.737	0.628	0.658	0.730	0.630	0.740
2016/2017	0.631	0.720	0.594	0.625	0.720	0.585	0.403
2017/2018	0.604	0.716	0.555	0.621	0.726	0.575	0.055

TABLE 5.6: Support vector machine scores using **one** preceding season for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

	Baseline scores			**Feature scores**			
Season	**Accuracy**	**F1-score**	**AUC**	**Accuracy**	**F1-score**	**AUC**	***p*-value**
2015/2016	0.647	0.723	0.618	0.663	0.730	0.640	0.335
2016/2017	0.623	0.718	0.582	0.633	0.718	0.593	0.246
2017/2018	0.609	0.712	0.567	0.588	0.682	0.555	0.195

TABLE 5.7: Support vector machine scores using **all** preceding seasons available for training. The p-values with an * depict a significant difference between the base-model and the feature-model.

The results of our support vector machine classifier are shown in table 5.6 and 5.7. Like with the previous models, the feature models showed no significant improvement in comparison to the baseline models. The scores are almost identical to each other with both the single-season trained models and the multi-season trained models.

Overall, the models performed better than a random model which would have an accuracy of around 0.5 with a large enough sample size. The models consistently showed accuracy's higher than 0.6 However, our feature models which included player-statistics to enhance the performance did not yield any positive results. Of the complete set of measurements we did, only two models yielded a statistically significant improvement. The inclusion of player-statistic features, therefore, did not perform consistently over the full data-set. A potential justification for this result could be based on the idea behind the team-compositing technique. The values for

each feature are put in an array which largely consists of zeros. As explained in paragraph 4.2, each feature technically consists of 1734 feature-columns during the actual training phase. Around 96% of those columns are zeros. With a ratio this shifted, the values of the 4% of data that is filled in can be saturated by the overflow of zero's. Suggesting that the actual value is not more important than the presence of a value in that location. For player-identity modelling, this can be ideal. However, when adding more expressive values instead of just ones, this technique can lead to a saturation of that value. The two models that did have a significant effect, can be seen as fluctuations. The *p*-values of these two models were moderately close to our 0.05 threshold, implying that the statistical significance is not irrefutable.

5.3 Model comparison

The results discussed in paragraph 5.2 are visualized in figure 5.1 and 5.2 to compare the learning algorithms used in this research. According to the McNemar-test, the three algorithms did not show any significant differences and performed consistently over the different data-samples. Therefore, the type of learning algorithm had no impact on the performance of the classification.

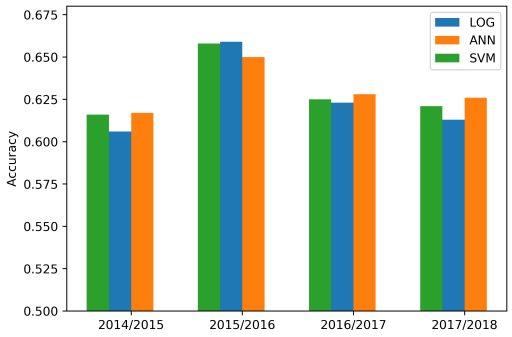

FIGURE 5.1: Accuracy scores for all learning algorithms, using a single preceding season for fitting

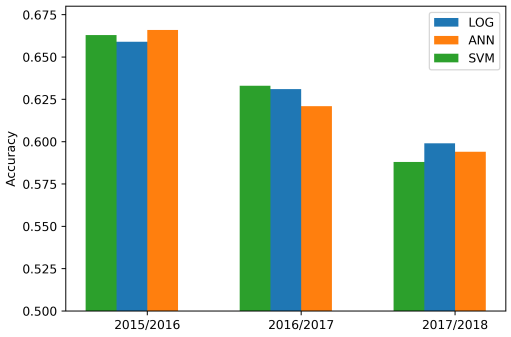

FIGURE 5.2: Accuracy scores for all learning algorithms, using multiple seasons for fitting

6 Conclusion and Discussion

In this thesis we applied the team-composition model to NBA game-data to answer the following main research questions:

- Does the addition of player-statistics to the team composition modelling technique improve the prediction performance in comparison to only using the player identity?
- Which classification algorithm achieves the highest performance with the team modelling technique out of Multivariate Logistic Regression, Neural Networks and Support Vector Machines?

For the first question, we implemented 8 additional features based on the player statistics to improve the prediction accuracy. The feature selection process resulted in an elimination of 6 of those features, making the final data-set considerably smaller. However, the results were rather disappointing as the final model did not show any significant improvements in comparison to only using the player identity. Neglecting a few fluctuations which were on the boundary, this led to the conclusion that the addition of player-statistics did not result in a higher predictive performance.
The second question is based on using the results of our final model and comparing the different learning algorithms to each other. The results showed that the learning algorithms performed similarly on different data-sets. The variation in our results was mostly accredited to the data used for training and testing and not the algorithms used. This was the case for the best performing feature model and the baseline identity model. Therefore leading us to the conclusion that the type of learning algorithm used in this context did not affect the performance in a significant way.

This thesis clearly showed the obstacles a machine learning research, in general, might have. The importance of preprocessing and the quality of the data was highlighted in chapter 3 where several steps were taken to improve this quality. Nonetheless, there is still room for improvement. One of the steps could be to find an accurate method to fill the missing data in the player-statistics. A first step was taken in this research by using averages and matrix factorization to fill these missing values. We did not succeed in applying the second approach and several arguments can be made against the accuracy of using averages.

During the analysis of the results, potential flaws of the team-composition technique came forward. The technique can be seen as reliable in the case of only representing the identity of players. However, when player-statistics are added the dimension of the training-data grows excessively and can lead to saturation of the player-statistic values. In future work, this potential problem can be resolved by applying some form of dimensionality reduction on the feature-level, whereas we only selected between the high-level features in this research. Additionally, with feature-sets this large, models are more susceptible to overfitting. In this research, some measures were taken by applying techniques like cross-validation to counteract this development. However, several more sophisticated approaches were left unexplored.

Lastly, the technique itself may have to be altered at its fundamentals to allow for a more compact representation of player-data in each game. Since the goal of the technique is to represent the data in an uncomplicated manner, it does not offer an adequate approach to the addition of personalized statistics as Sing [16] suggested. Future work may include defining a new approach to representing the statistics of each player for each game. An approach that should be able to represent the player identity and personalized statistics in a more condensed sapce.

A Appendix

A.1 Tables

	start_time	away_team	away_team_score	home_team	home_team_score
0	2013-10-29 23:00:00+00:00	ORLANDO MAGIC	87	INDIANA PACERS	97
1	2013-10-30 00:00:00+00:00	CHICAGO BULLS	95	MIAMI HEAT	107
2	2013-10-30 02:30:00+00:00	LOS ANGELES CLIPPERS	103	LOS ANGELES LAKERS	116
3	2013-10-30 23:00:00+00:00	BROOKLYN NETS	94	CLEVELAND CAVALIERS	98
4	2013-10-30 23:00:00+00:00	BOSTON CELTICS	87	TORONTO RAPTORS	93
5	2013-10-30 23:00:00+00:00	MIAMI HEAT	110	PHILADELPHIA 76ERS	114
6	2013-10-30 23:30:00+00:00	WASHINGTON WIZARDS	102	DETROIT PISTONS	113
7	2013-10-30 23:30:00+00:00	MILWAUKEE BUCKS	83	NEW YORK KNICKS	90
8	2013-10-31 00:00:00+00:00	INDIANA PACERS	95	NEW ORLEANS PELICANS	90
9	2013-10-31 00:00:00+00:00	ORLANDO MAGIC	115	MINNESOTA TIMBERWOLVES	120
10	2013-10-31 00:00:00+00:00	CHARLOTTE BOBCATS	83	HOUSTON ROCKETS	96
11	2013-10-31 00:30:00+00:00	MEMPHIS GRIZZLIES	94	SAN ANTONIO SPURS	101
12	2013-10-31 00:30:00+00:00	ATLANTA HAWKS	109	DALLAS MAVERICKS	118
13	2013-10-31 01:00:00+00:00	OKLAHOMA CITY THUNDER	101	UTAH JAZZ	98
14	2013-10-31 02:00:00+00:00	PORTLAND TRAIL BLAZERS	91	PHOENIX SUNS	104
15	2013-10-31 02:00:00+00:00	DENVER NUGGETS	88	SACRAMENTO KINGS	90
16	2013-10-31 02:30:00+00:00	LOS ANGELES LAKERS	94	GOLDEN STATE WARRIORS	125
17	2013-11-01 00:00:00+00:00	NEW YORK KNICKS	81	CHICAGO BULLS	82
18	2013-11-01 02:30:00+00:00	GOLDEN STATE WARRIORS	115	LOS ANGELES CLIPPERS	126
19	2013-11-01 23:00:00+00:00	CLEVELAND CAVALIERS	84	CHARLOTTE BOBCATS	90

TABLE A.1: Sample of Game-data 2013/2014 season

age	23
assist_percentage	10.1
block_percentage	3.9
box_plus_minus	5
defensive_box_plus_minus	3.7
defensive_rebound_percentage	19.2
defensive_win_shares	0.1
free_throw_attempt_rate	0.571
games_played	7
minutes_played	61
name	Quincy Acy
offensive_box_plus_minus	1.2
offensive_rebound_percentage	9.5
offensive_win_shares	0.1
player_efficiency_rating	17.2
positions	[SMALL FORWARD]
slug	acyqu01
steal_percentage	3.4
team	TORONTO RAPTORS
three_point_attempt_rate	0.357
total_rebound_percentage	14.3
true_shooting_percentage	0.542
turnover_percentage	10.2
usage_percentage	14.5
value_over_replacement_player	0.1
win_shares	0.2
win_shares_per_48_minutes	0.188

TABLE A.2: Advanced player statistics of Quincy Acy for the 2013/2014 season

age	23
assists	4
attempted_field_goals	14
attempted_free_throws	8
attempted_three_point_field_goals	5
blocks	3
defensive_rebounds	10
games_played	7
games_started	0
made_field_goals	6
made_free_throws	5
made_three_point_field_goals	2
minutes_played	61
name	Quincy Acy
offensive_rebounds	5
personal_fouls	8
positions	[SMALL FORWARD]
slug	acyqu01
steals	4
team	TORONTO RAPTORS
turnovers	2

TABLE A.3: Season totals Quincy Acy for season 2013/2014

A.2 Code

Link to all the code and score-logs used in this research:
https://github.com/brahimeg/NBA-thesis.git

Baseline measurements [identity_array] LOGR													Scores with best feature-set LOGR									
Training	Testing	Training Accur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test	<- p-value ->	Training	Testing	TrainingAccur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test
2013/2014	2014/2015	0.708	0.609	0.791	0.825	0.759	0.708	0.692	0.571	0.683	2013/2014	2014/2015	0.704	0.606	0.817	0.829	0.762	0.707	0.682	0.566
2014/2015	2015/2016	0.709	0.640	0.788	0.827	0.757	0.730	0.695	0.600	0.024	2014/2015	2015/2016	0.708	0.659	0.809	0.836	0.761	0.741	0.690	0.617
2015/2016	2016/2017	0.718	0.625	0.818	0.865	0.773	0.729	0.696	0.577	0.752	2015/2016	2016/2017	0.714	0.623	0.840	0.889	0.776	0.733	0.687	0.569
2016/2017	2017/2018	0.677	0.605	0.792	0.890	0.741	0.723	0.654	0.551	0.235	2016/2017	2017/2018	0.680	0.613	0.825	0.897	0.750	0.729	0.651	0.560
all before	2014/2015	0.708	0.609	0.791	0.825	0.759	0.708	0.692	0.571	0.683	all before	2014/2015	0.704	0.606	0.817	0.829	0.762	0.707	0.682	0.566
all before	2015/2016	0.707	0.646	0.788	0.807	0.757	0.729	0.692	0.612	0.060	all before	2015/2016	0.705	0.659	0.806	0.804	0.760	0.735	0.687	0.627
all before	2016/2017	0.711	0.622	0.799	0.840	0.762	0.722	0.693	0.578	0.508	all before	2016/2017	0.705	0.631	0.809	0.841	0.761	0.727	0.684	0.589
all before	2017/2018	0.703	0.604	0.797	0.858	0.757	0.715	0.684	0.556	0.805	all before	2017/2018	0.700	0.599	0.811	0.822	0.759	0.704	0.678	0.558
Baseline measurements [identity_array] ANN											Scores with best feature-set ANN									
Training	Testing	Training Accur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test	<- p-value ->	Training	Testing	TrainingAccur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test
2013/2014	2014/2015	0.719	0.615	0.810	0.833	0.770	0.714	0.701	0.577	0.807	2013/2014	2014/2015	0.710	0.617	0.770	0.779	0.755	0.701	0.698	0.589
2014/2015	2015/2016	0.720	0.642	0.750	0.747	0.754	0.711	0.714	0.620	0.368	2014/2015	2015/2016	0.715	0.650	0.748	0.750	0.751	0.716	0.709	0.629
2015/2016	2016/2017	0.731	0.615	0.783	0.760	0.774	0.698	0.720	0.586	0.275	2015/2016	2016/2017	0.728	0.628	0.797	0.787	0.775	0.712	0.713	0.596
2016/2017	2017/2018	0.685	0.609	0.783	0.846	0.743	0.715	0.665	0.565	0.010	2016/2017	2017/2018	0.678	0.626	0.779	0.857	0.738	0.726	0.658	0.583
all before	2014/2015	0.719	0.615	0.810	0.833	0.770	0.714	0.701	0.577	0.807	all before	2014/2015	0.710	0.617	0.770	0.779	0.755	0.701	0.698	0.589
all before	2015/2016	0.716	0.659	0.783	0.790	0.761	0.732	0.704	0.631	0.396	all before	2015/2016	0.711	0.666	0.773	0.760	0.755	0.728	0.699	0.646
all before	2016/2017	0.717	0.624	0.776	0.809	0.761	0.715	0.706	0.586	0.922	all before	2016/2017	0.712	0.621	0.783	0.805	0.760	0.713	0.698	0.584
all before	2017/2018	0.706	0.597	0.789	0.838	0.758	0.707	0.690	0.551	0.888	all before	2017/2018	0.704	0.594	0.771	0.789	0.752	0.693	0.691	0.558
Baseline measurements [identity_array] SVM											Scores with best feature-set SVM									
Training	Testing	Training Accur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test	<- p-value ->	Training	Testing	TrainingAccur.	Testing Accur.	recall train	recall test	F1 train	F1 test	AUC train	AUC test
2013/2014	2014/2015	0.706	0.614	0.783	0.796	0.755	0.703	0.691	0.582	0.784	2013/2014	2014/2015	0.703	0.616	0.782	0.793	0.754	0.703	0.688	0.584
2014/2015	2015/2016	0.709	0.660	0.785	0.811	0.756	0.737	0.696	0.628	0.740	2014/2015	2015/2016	0.710	0.658	0.786	0.787	0.757	0.730	0.696	0.630
2015/2016	2016/2017	0.715	0.631	0.812	0.815	0.771	0.720	0.695	0.594	0.403	2015/2016	2016/2017	0.717	0.625	0.815	0.825	0.772	0.720	0.696	0.585
2016/2017	2017/2018	0.676	0.604	0.792	0.864	0.741	0.716	0.653	0.555	0.055	2016/2017	2017/2018	0.676	0.621	0.795	0.867	0.741	0.726	0.652	0.575
all before	2014/2015	0.706	0.614	0.783	0.796	0.755	0.703	0.691	0.582	0.784	all before	2014/2015	0.703	0.615	0.782	0.793	0.754	0.703	0.688	0.584
all before	2015/2016	0.707	0.647	0.784	0.783	0.756	0.723	0.693	0.618	0.335	all before	2015/2016	0.707	0.663	0.786	0.772	0.756	0.730	0.692	0.640
all before	2016/2017	0.711	0.623	0.793	0.823	0.761	0.718	0.694	0.582	0.246	all before	2016/2017	0.710	0.633	0.797	0.808	0.762	0.718	0.693	0.593
all before	2017/2018	0.702	0.609	0.794	0.834	0.756	0.712	0.684	0.567	0.195	all before	2017/2018	0.702	0.588	0.796	0.764	0.757	0.682	0.684	0.555

FIGURE A.1: Extended final scores of the baseline model and best feature model

Bibliography

[1] Rory P Bunker and Fadi Thabtah. "A machine learning framework for sport result prediction". In: *Applied computing and informatics* 15.1 (2019), pp. 27–33.

[2] Chenjie Cao. "Sports data mining technology used in basketball outcome prediction". In: (2012).

[3] H Michael Chung and Paul Gray. "Data mining". In: *Journal of management information systems* 16.1 (1999), pp. 11–16.

[4] Manoranjan Dash and Huan Liu. "Feature selection for classification". In: *Intelligent data analysis* 1.3 (1997), pp. 131–156.

[5] Thomas G Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[6] Martin Hofmann. "Support vector machines-kernels and the kernel trick". In: *Notes* 26.3 (2006).

[7] Josip Hucaljuk and Alen Rakipović. "Predicting football scores using machine learning techniques". In: *2011 Proceedings of the 34th International Convention MIPRO*. IEEE. 2011, pp. 1623–1627.

[8] T Jayalakshmi and A Santhakumaran. "Statistical normalization and back propagation for classification". In: *International Journal of Computer Theory and Engineering* 3.1 (2011), pp. 1793–8201.

[9] Peter K Kitanidis and Robert W Lane. "Maximum likelihood parameter estimation of hydrologic spatial processes by the Gauss-Newton method". In: *Journal of Hydrology* 79.1-2 (1985), pp. 53–71.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems". In: *Computer* 42.8 (2009), pp. 30–37.

[11] Bernard Loeffelholz, Earl Bednar, and Kenneth W Bauer. "Predicting NBA games using neural networks". In: *Journal of Quantitative Analysis in Sports* 5.1 (2009).

[12] Dragan Miljković et al. "The use of data mining for basketball matches outcomes prediction". In: *IEEE 8th International Symposium on Intelligent Systems and Informatics*. IEEE. 2010, pp. 309–312.

[13] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Michael C Purucker. "Neural network quarterbacking". In: *IEEE Potentials* 15.3 (1996), pp. 9–15.

[15] Frank Rosenblatt. "Perceptron simulation experiments". In: *Proceedings of the IRE* 48.3 (1960), pp. 301–309.

[16] Prastuti Singh and Bai Yang Wang. "NBA Game Predictions based on Players' Chemistry". In: *Department of Physics, Stanford, CA 94305* ().