A DYNAMIC BAYESIAN NETWORK TO PREDICT THE TOTAL POINTS SCORED IN NATIONAL BASKETBALL ASSOCIATION GAMES

by

Enrique M. Alameda-Basora

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Industrial Engineering

Program of Study Committee:
Sarah Ryan, Major Professor
Dan Nettleton
Sigurdur Olafsson

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

I dedicate this work to my beloved parents, Eloiris Basora-Cintron and Rafael A. Alameda-Rojas. Thank you for always loving me unconditionally and instilling within me the values of hard work and dedication. Before embarking on my journey to Iowa State University, I had to cope with the decision not to see you as often, and that was probably the hardest decision of my life. I could imagine how difficult it was to take care of four children. I vividly recall times when you would spend a whole day of your busy lives making sure everyone completed their science fair project due that week. I want you to know that all the sacrifices you made for us were not in vain. Incredibly, we have all grown up and become adults with promising careers and/or graduate degrees. Our success is a testament to all the hours of hard work you put into raising us and making sure we believed in our abilities to succeed. Thank you for teaching me the value of education and life-long learning. Without you, I would not be where I am today. I will forever be grateful and indebted to you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This thesis was completed due to the collective efforts of many individuals. First and foremost, I want to thank my committee members: Dr. Sigurdur Olafsson, Dr. Dan Nettleton, and Dr. Sarah Ryan for being willing to serve as my advisors and mentors. I want to thank specifically my major professor, Dr. Sarah Ryan, for her continued patience throughout this whole process and for taking me in when I was a junior in my undergraduate program and introducing me to the wonderful and incredibly rewarding world of research. Thanks to her, I was able to choose a topic of my interest and define a research problem that would test my intellect and expand my knowledge of statistics and machine learning.

Next, I wish to thank two dear friends and colleagues that helped me get motivated to solve my research problem: Dr. Hieu T. Pham and Gorkem Emirhuseyinoglu. Hieu, thank you for always asking about my progress and being genuinely interested in my topic. Also, thank you for sharing your views on the Data Science community with me and answering some of the questions regarding the profession. Gorkem, thank you for staying up all night with me all winter break working on your research while I worked on mine. If it was not for seeing how hard you worked in a span of three weeks, I would have probably not finished as quickly as I did. Keep up that good work ethic!

Lastly, I want to thank all those people who in one way or another understood how tedious this five year program was for me and sacrificed their time with me, allowing me to finish my research. Thank you Courtney Geiken, Pierre Alameda-Basora, Valeria Alameda-Basora, and Rafael Alameda-Basora. I promise that I will make up all the time I missed with you.

ABSTRACT

Bettors on National Basketball Association (NBA) games commonly place wagers concerning the result of a game at time points during that game. We focus on the Totals (Over/Under) bet. Although many forecasting models have been built to predict the total number of points scored in an NBA game, they fail to provide bettors engaged in live-betting with predictions that are based on the game currently being played. We construct an Expert Bayesian Network to sequentially, as the game progresses, update the probability that the total points scored by both teams will exceed that set by the oddsmakers, and then use this probability to influence our wager at the end of the first, second, and third quarters. Research methods include data collection of team statistics over the last five NBA seasons, discretization of features, filter-based feature selection and specification of the network structure using domain knowledge and statistical tests. We compare the profit of our live-betting strategy against amateur betting strategies, wagers informed by a Naïve Bayes classifier, and wagers informed by a Bayesian Network whose structure is specified using a greedy search algorithm. When applied to games played during the early 2018-2019 NBA regular season, the Expert Bayesian Network and the Naïve Bayes model provide the most accurate predictions. Wagers informed by these two models yield profits of over 10% and 6%, respectively, but the other models and strategies are not profitable.

CHAPTER I

INTRODUCTION

1.1 Background and Motivation

With a total of 1,230 games each regular season, there are far more opportunities to wager on a National Basketball Association (NBA) game than most other sports leagues. The outcome of NBA games is moderately predictable and, therefore, betting on the conclusion of a game yields little to no profit due to low-risk wagers (Stern, 2008). The three most common bets placed on an NBA game are Point Spread, Moneyline and Totals (bettingexpert, 2018). Point Spread is defined as the differential of the points scored between the two teams, whereas the Moneyline is a simple win/lose bet. For Totals (also known as the Over/Under), oddsmakers (also known as bookmakers) set a total number of points for any given NBA game and bettors place their wagers on whether the combined teams' scores are more points (over) or fewer points (under) than the number of points set by the oddsmakers. Although intuition suggests that bets should be placed on the end result only at the beginning of the game, bettors commonly place wagers during specific time points in the game such as the ends of quarters (Williams 2010). With the risk of wagering on the total points in basketball games, one can see how predictive models can aid bettors in decision making. To better illustrate the three common types of betting options in an NBA game, we provide an example in **Table 1** of a hypothetical game matchup between the Los Angeles Lakers (LAL) and the Toronto Raptors (TOR).

Table 1: Typical Example of the Three Betting Options

	Team	**Spread**	**Win**	**Total**
10/14/2016	LAL	-5.0 (-115)	-250	O 169.0 (-125)
7:00PM	TOR	+5.0 (-125)	+170	U 169.0 (-115)

In **Table 1**, at the beginning of the game, the bettor can wager on any of the three options. For point spread, the bettor can wager on whether the underdog, as denoted by the positive (+) sign, Toronto Raptors will win the game or lose by fewer than five points, or the Los Angeles Lakers will win the game by more than five points. For the Moneyline option, the bettor can choose to either bet that the Los Angeles Lakers will win the game or, if they want to maximize their potential earnings (and risk), bet that the Toronto Raptors will win the game. For the Over/Under option, the oddsmakers set the total points for both teams to be 169 and the bettor has the option to wager on whether the total points scored by both teams will be Over or Under that number. Unless specified by the bookmakers, the payout for an accurate wager is, if denoted by the positive (+) sign, how much money the bettor wins if they wager $100 and, if denoted by the negative (-) sign, how much money the bettor must wager to win $100.

Due to basketball's high and volatile scoring nature, Point Spread and Totals betting approaches are more difficult to predict than Moneyline; however, the payoff is larger in most cases (Williams, 2010). In fact, the difficulty with predicting winning probabilities (Moneyline) is well-known as there is a lack of context within the game, no measure of prediction uncertainty and no publicly available data sets or models against which researchers and analysts can compare their results (Ganguly & Frank, 2018). Due to the risk involved in wagers, a tool for bettors that estimates the joint probability distribution of scoring totals given a set of variables, uses this distribution to estimate the probability that the score is greater than that value set by the oddsmakers, and is updated as the game proceeds would be valuable to users.

1.2 Research Problem

Although many forecasting models have been built to predict the total number of points, these models' predictions are primarily based on data from previously completed games. This

method fails to provide bettors with predictions also based on the current game being played. Specifically, there does not exist a publicly available model that estimates the probability that the score total is greater than the total number of points set by the oddsmakers using in-game data. In this thesis, we aim to sequentially update this probability through a machine learning-based network and use it to make wagering decisions at the end of each of the first three quarters as the game progresses.

1.3 Proposed Solution

The machine learning-based network we propose is a Bayesian network (BN). Given a set of conditional probability tables, computed on random variables, the BN captures all existing knowledge about its inputs (random variables of interest) and converts it into a directed acyclic graph (Jensen, 2009). This is a graph which consists of a set of nodes, a set of directed arcs that pair distinct nodes to each other and contains no cycle (Bertsekas, 1998). The knowledge is then used to predict outcomes or diagnose causal effects (if the structure is known), or to discover causal relationships (if the structure is unknown). As with every sport, basketball possesses a vast array of statistics that are collected in every game which are correlated and can be used as inputs, or predictors, within the Bayesian network.

1.4 Organization of Thesis

The remaining chapters of this thesis are organized as follows. In Chapter II, we review the related studies in the literature. In Chapter III, we give an extensive overview of Bayesian Belief Networks and describe how one can be used to solve the research problem presented in this thesis. In Chapter IV, we provide a description of the data collected for both the training and testing sets and how feature selection was conducted. In Chapter V, we detail the experiment which was designed in hopes of building a profitable Bayesian Network which can estimate the probability

that the score total is greater than the total number of points set by the oddsmakers. In Chapter VI, we evaluate the networks constructed in the study and compare them to amateur betting strategies as well as a Naïve Bayes classifier. Finally, concluding remarks are provided in Chapter VII.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

In this chapter, we review the sources and documentation that relate to the topic and pave the way for future research work. While reviewing them, we give a critical evaluation of these works with respect to the research problem being investigated. This whole review consists of two main parts. The first is an overview of data mining followed by the role it plays in sports predictive modeling. Then, we review the data mining techniques applied to making predictions in basketball.

2.2 Data Mining and its Role in Sports Predictive Modeling

2.2.1 Data Mining Concept

Aggarwal (2015) describes data mining, a complex and multistage process, as "the study of collecting, cleaning, processing, analyzing and gaining useful insights from data." Colloquially, data mining starts when a method to collect data is employed and ends when results and recommendations for a specific system are given through the analysis of said data. An interdisciplinary process, data mining requires fluency in the quantitative disciplines of statistics, mathematics, decision science and computer science (Dhar, 2013). Not to be confused with the data extracted from statistics, the data often collected in the process of data mining are heterogeneous and unstructured. One may argue that the most important part in the process of data mining is transforming these conglomerated data into a standardized format one can comprehend more easily (Mikut & Reischl, 2011). After the data collection and pre-processing step, it is important to specify a training and a testing data set. Typically, most of the data collected are used as a training set from which the model can learn, and a smaller portion of data is then used for testing and evaluating the corresponding data mining model.

After the data sets have been determined, the analysis phase begins. One of the most common ways of analyzing the data is using a machine learning algorithm. These algorithms, as explained by Alpaydin (2014), "involve collecting a large sample of data and programming computers to optimize a performance criterion using these samples." Essentially, the goal of machine learning is to "teach" the computer to extract an algorithm for a specific task (Alpaydin, 2014). There are two types of machine learning algorithms: supervised learning and unsupervised learning. Although the machine learning concept remains the same, the clear distinction between these types involve whether the data collected combine input variables with an associated output variable (supervised learning) or if it is a collection of input data with no corresponding output data (unsupervised learning).

In supervised machine learning the goal is to learn, from a training data set, all the input variables and their corresponding output variable in such a way that when given new input data, the machine can predict the value of the output variable for the data. If this output variable is discrete, then it becomes a classification problem. On the other hand, if the output variable is a continuous real value, it is referred to as a regression problem (Kotsiantis, 2007). The machine learning-based network in this thesis is an example of a supervised machine learning algorithm. To be specific, we focus on a classification problem where, given a set of input variables such as effective field goal percentage and pace, we estimate the total points scored by both teams, our output variable. Then, using the constructed network, we find the probability that the total number of points is greater than the value specified by the oddsmakers.

2.2.2 Sports Predictive Modeling

Sports predictive modeling, also known as sports analytics, is an emerging field that involves data management, predictive models and information systems to predict specific sport-related outcomes in hopes of improving sporting performance (Gerrard, 2014). Due to the vast amount of statistics and metrics collected for each player and team every game, the practicality of data mining tools and techniques in sports analytics is evident. To name a few examples, if sports organizations can perform machine learning algorithms on the data collected from every game they would be able to correctly predict which of their players will be stars, successful coaching and training strategies and how well they will do in the upcoming season.

Because Operations Research in sports has been around for more than 50 years (Wright, 2009), using data mining tools and techniques to make predictions is not a novel idea (Haghighat et al., 2013). Forecasting models have been used to determine the outcome of sports for years. What drives a lot of these models, just like the one constructed in this thesis, is the sports betting market. In Section 2.3, we examine some of these models. Specifically, we describe those models used to predict the outcome or find the joint probability distribution of total points scored in a basketball game, whether it be a National Collegiate Athletic Association (NCAA) basketball game or an NBA game. For an in-depth comparison that explains the similarities and differences of predictions in both leagues, refer to Zimmerman (2016).

2.3 Data Mining Techniques Applied to Basketball Predictive Modeling

2.3.1 Naïve Bayes Classifier

Naïve Bayes (NB) is a simple machine learning algorithm used for classification. Before using the NB algorithm, one must understand its underlying assumptions. It assumes that all the input variables are equally important and that they are all independent of one another. These

assumptions are quite strong and, therefore, it is difficult to find a data set where they hold true. Nevertheless, the NB classifier works surprisingly well despite its unrealistic assumptions. Essentially the NB classifier is centered on Bayes' theorem. Bayes' theorem, depicted in **Eq. (2.1)**, describes how to update the probability of a hypothesis (H) when given evidence (E). It follows the axioms of conditional probability and it is a common technique on which many modern machine learning algorithms, including Bayesian Networks, rely. Using Bayes' Theorem, NB finds the most likely hypothesis (H_{ML}) given the data and its evidence.

$$p[H|E] = \frac{p[E|\,H]\,p[H]}{p[E]} \tag{2.1}$$

Rigorously, given a problem instance to predict, represented by a vector $X = (x_1, \ldots, x_n)$ representing *n* input variables, the NB assigns to this instance a probability $p(Y_k|x_1, \ldots, x_n)$ for each of k possible outcomes for an output variable Y_k. Using Bayes' theorem to calculate the posterior probability, one can reformulate the model with X as our evidence and Y_k as our hypotheses to make it more manageable for computing probability tables involving a large number of input variables such that $p(Y_k|X) = \frac{p(Y_k)p(X|Y_k)}{p(X)}$. Afterwards, following the assumption that all the input variables are independent of one another, one can assume each feature x_i is independent of every other feature. This means that the probability of a feature given the other features and its output variables becomes simply the probability of the feature given the output variables, that is $p(x_i|x_{i+1}, \ldots, x_n, Y_k) = p(x_i|Y_k)$. Finally, by creating a joint model, one can then calculate each hypothesis' maximum likelihood such that $H_{ML} = \underset{k\in\{1,\ldots,K\}}{argmax}\, p(Y_k) \prod_{i=1}^{n} p(x_i|Y_k)$ and make their prediction (Raschka, 2014).

In their research study, Miljković et al. (2010) use the NB classifier to build a model and predict the outcome of 778 NBA games. They are interested in the point spread betting option.

The system has an accuracy of 10%, as only 78 out of the 778 games' spread (or point difference) are correctly determined (in the sense of the point spread wager in **Table 1**). As mentioned previously, due to the volatility of scoring, it is extremely difficult to predict the point differential of an NBA game. In fact, de Saá Guerra et al. (2011) studies the volatility of these games thoroughly and proves how dynamic and how dependent on phase transitions the scoring of an NBA game is. However, Miljković et al. clearly violate one of the assumptions of the NB classifier, as the input variables they use are obviously dependent upon one another. An example of this dependence is when they use field goals made and field goals attempted as one cannot make more field goals than those attempted. Although the authors use input variables that violate the assumptions of the NB, they claim that the results are satisfactory and in line with expectations. One important limitation is that Miljković et al. use end-game summary statistics to build their model. Although a common practice, it limits the practicality of their model as it can only be applied after the game has concluded, when bets have already been decided, and cannot be used for live-betting.

2.3.2 Logistic Regression

Developed by statistician David Cox (1958), logistic regression is an example of a supervised machine learning algorithm where the output variable is categorical. A clear distinction to make is that logistic regression in itself is not a classification algorithm. It only becomes a classification algorithm when combined with a decision rule that associates the output variables' outcomes with dichotomous predicted probabilities. There are different types of logistic regression such as binary logistic regression, multinomial logistic regression where the dependent variable has more than two outcome categories, ordinal logistic regression where there are multiple ordered (in terms of nature of information within the values assigned to variables) categories and many

more (Norris et al., 2006). Nonetheless, all types of logistic regression models can be used to estimate the probability distribution of one or more random variables following a cumulative logistic distribution. These algorithms measure the relationship between the categorical dependent variable and the independent variables by estimating probabilities of outcomes using the standard logistic function depicted in **Eq. (2.2).**

$$F(t) = \frac{1}{1+e^{-t}} \; where \; t \in \mathbb{R} \qquad \textbf{(2.2)}$$

The best way to understand logistic regression is that one is finding the β (intercept and slope) parameters that best fit the data and describe the relationship between the dichotomous characteristic of interest and the input variables. The regression tactic itself, just like in simple linear regression, generates the coefficients, standard error and significance levels of a formula to predict a logit transformation of the probability of the output variable. That is, given a vector of n input variables $X = (x_1, \ldots, x_n)$, logistic regression intends to generate a formula of the form $logit(p) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n + \epsilon$ where p is the probability that the output variable takes on a particular value and ϵ is the standard error. The logit transformation itself can be defined in terms of log odds by solving for t in **Eq. (2.2)** such that, $odds = \frac{p}{1-p}$ and $logit(p) = \ln(\frac{p}{1-p})$. Rather than minimizing the sum of squared errors, logistic regression chooses parameters that maximize the likelihood of observing the sample values and uses this likelihood to make its prediction (Gortmaker et al., 1994). Just like the NB classifier, before modeling the data using a logistic regression algorithm, one must understand its underlying assumptions. These assumptions are not as strong as the NB ones (independence between variables) but can hamper performance if violated. Logistic regression assumes that all the observations in the data set are independent of

one another and that the independent variables are linearly correlated with the logistic odds, $logit(p)$.

Kvam and Sokol (2006) developed a logistic regression/Markov chain model to predict the outcome (win/loss) of the NCAA basketball tournament matches. They used a Markov chain to build the transition probability matrix within games and considered the location of the game and the margin of victory in previous games. Their model yielded promising results as it outperformed the most prevalent ranking systems and paved the way for future research work combining the two methods.

In their research work, Štrumbelj and Vračar (2012) use a multinomial logistic regression algorithm in combination with a Markov model approach to estimate the outcome (win/loss) of NBA games. Their main goal is to see if they can simulate and calculate the winning percentage over time. Their model correctly predicts the outcome of about 70% of the games they sample. Although it yields fruitful results, the authors acknowledge the model's limitations. The proposed Markov model used is homogeneous, meaning that the transition probabilities do not change as the match progressed, which is a strong assumption. The authors emphasize the importance of having an unbiased model that is non-homogeneous and conclude that, for future research work, the transitional conditional probabilities should be more focused on the point spread and the specific time point in the game.

Lopez and Matthews (2015) built two logistic regression models to help predict the outcome (win/loss) of NCAA basketball tournament matches. The first was a point-spread-based logistic regression model. The second was an efficiency model built using logistic regression on game outcomes. The authors later combined these two models using an ensemble, where individual produced models are merged using a weighted average, based on a Log Loss score. Their combined

model won the first college basketball competition hosted by Kaggle, a website that organizes analytics and modeling competitions, as it correctly predicted more tournament matches than the other models submitted. The authors proved that by using the right input variables, point differential, and efficiency metrics, they could outperform complex models that were shown to be more accurate than the logistic regression. The authors emphasized the need to include these predictors in more advanced predictive models that used Bayesian statistics.

2.3.3 Neural Networks

According to machine learning principles, it is often worth understanding models that are known to be wrong. However, we must remember what makes these models wrong to begin with. By learning and understanding what the model predicted incorrectly, we can then make more accurate assessments. That is what artificial neural network algorithms hope to accomplish. A neural network consists of elements called neurons (or nodes) and directed weighted arcs. Each neuron receives input from a prior node or data input (depending on the number of hidden layers) and then an activation function is computed based on the weighted arc (Mao, 1996). These weights, as well as the activation of the nodes, can be modified by specific learning rules that help the algorithm make its prediction. What makes neural networks popular is their ability to perform both supervised and unsupervised learning. For the purpose of this thesis, we will restrict ourselves to an overview of supervised learning neural networks as it is the only one that relates to the problem at hand.

One can think of an artificial neural network as a typical function, $f: X \to Y$, which maps a set of inputs (X) to their corresponding output (Y). We proceed to define this function $f(x)$ as a composition of other functions $g_i(x)$ where g is a vector of functions, $g = (g_1, g_2, \ldots, g_n)$, that can further be decomposed into other functions. Then we represent these functions as a network

structure with arcs depicting the dependencies between functions. The most widely used type of composition is the nonlinear weighted sum (Kennedy & Chua, 1988), where $f(x) = K(\sum_i w_i g_i(x))$ and K, the activation function, is some predefined function. This activation function provides a smooth transition to a new corresponding output as input values change. To learn from this network, we then define a cost function $C: F \to \mathbb{R}$ where F is a function space such that, for the optimal solution f^*, $C(f^*) \leq C(f)\ \forall f \in F$. Learning algorithms then search through the feasible region to find a function that has the smallest possible cost. There are many cost functions; however, a commonly used one is the mean-squared error (Kennedy & Chua, 1988).

Although an extremely powerful machine learning approach, neural networks have their share of disadvantages. For one, artificial neural networks' prediction performance is inconsistent as there is a plethora of decisions that must be made within the network such as: the number of layers, the number of nodes in each layer and the activation function (Tu, 1996). These vary slightly for each iteration and, thus, the final result may not always be the same. Another disadvantage is the need for lots of data. Although this is a drawback of most algorithms, it is especially relevant to artificial neural networks because of the vast number of weights and connections within them (Tu, 1996).

In their research work, Loeffelholz et al. (2009) use the statistics of 620 NBA games to train a variety of neural networks. They later fuse these networks, using Bayesian strategies, to predict the winning team for NBA games that have yet to be played. They are most interested in the Moneyline betting market, and they successfully predict 72% of the NBA games on which they test their model. The authors make a significant breakthrough by showing that neural networks are capable of using common box score statistics to accurately classify the outcome of a game that has yet to be played. Although insightful, their model does not change its prediction as the game is

being played (as this was not the authors' intent), and one can theoretically just predict the winningest team and have an accuracy of 67% (Lopez et al., 2017). However, their model can still be used to predict upsets and win a substantial amount of money by picking the underdog.

2.3.4 Review of Bayesian Networks Applied to Sports

Exhaustive details on BNs will be discussed in-depth in Chapter III. For now, we limit our discussion on the relevant literature involving BNs in the context of sports predictive modeling. As previously mentioned, applying data mining techniques to perform sports analytics has been around for more than 50 years. Therefore, applying Bayesian Networks to make predictions in sports games is not a novel approach. Although the resources are scarce when it comes to building a Bayesian Network to make predictions on the outcome of basketball games, there have been numerous attempts of constructing such a network in the context of other sports.

Min et al. (2008) develop a framework (which they call Football Result Expert System) for making predictions in soccer games using rule-based reasoning in combination with a BN approach. Due to the highly stochastic nature of the sport, the authors combine this framework with an in-game time-series approach for the model to yield more accurate and realistic predictions. In essence, the rule-based reasoner determines a teams' strategy and simulates the role of the head coach while the BNs sample the stages of the game progression. This is a significant breakthrough as the authors can successfully predict the outcomes of the games (win or lose) more realistically and with higher accuracy than models that do not use in-game statistics. The authors are also among the first to take into consideration multiple factors as opposed to just the score of the game to make their predictions all while displaying the versatility of BNs and how they can be used in situations where there is insufficient data. Min et al. conclude that such a framework can be readily applied to other sports but acknowledge that their method is lacking a more formal

machine learning approach and that the BN can be further used for parameter learning methods to tune their system automatically.

In their research study, Joseph et al. (2006) compare the performance of an expert constructed BN to other machine learning techniques when predicting the outcome (win, lose, or draw) of football matches. The authors use their specialty domain knowledge to determine the structure of the BN and compare its performance to a decision tree learner, the Naïve Bayes classifier, the k-nearest neighbor algorithm, and a BN which learns (presumably) a different structure from the data itself. The authors' contribution yields some fruitful results as they show that the BN whose structure is specified using their domain knowledge outperforms all other machine learning algorithms used in the study. Perhaps more insightful is the fact that their Expert BN has an overall average accuracy of 59.21%, which is significantly higher than the accuracy of the BN that learns the structure from the data (39.69%). The authors show that having expert knowledge is crucial when building a reliable BN. It is important to note that, when given data of full seasons, the BNs perform roughly the same. However, when the models are given less data, the data-driven BN does not learn a structure as reliable as the one built using domain knowledge. Moreover, the Expert BN predicts the outcomes significantly faster than the one that learns the structure from the data.

Like Min et al. in 2008, Constantinou et al. (2013) present a novel BN model called pi-football for forecasting soccer match outcomes. This model encompasses both objective (learned on the data) and subjective (using specialty domain knowledge) information to make its predictions, in which time-dependent data is weighted using degrees of uncertainty. The predictions are made on the games before they are played and evaluated on both accuracy and profitability measurements. The authors build the first publicly available model that demonstrates

profitability against all of the available published bookmaker odds. Furthermore, they show that the specialty domain knowledge (not acquired from the data) improves the forecast capability of their model. Their results once again suggest that BNs, coupled with domain knowledge, are extremely potent and sophisticated machine learning algorithms. They further show that their model beats all the bookmakers' odds over a long period of time when it comes to profitability. The authors acknowledge that their model might not be the most precise as they are not the most-informed experts and that the results are inconsistent. To remedy this inconsistency, the authors claim that they should add more subjective information on team strength in their future work. Nevertheless, the authors' inconsistency may result from using past games' data as opposed to in-game data. This inconsistency illustrates our point that using in-game summary statistics to train our model is crucial when making betting decisions.

2.3.5 Conclusion and Research Gap

Although there have been numerous machine learning algorithms applied to basketball games, these models were built to predict the outcome (win/loss) or point spread of the game as opposed to the total points scored by both teams. All these models have avoided predicting the team-winning probability or point differential at the end of the game sequentially as it is being played. This lack hinders their practicality when it comes to sports betting as, in basketball, one is allowed to bet at specific time points in the game such as the ends of quarters. Additionally, some authors (Štrumbelj & Vračaras in 2012 and Lopez & Matthews in 2015) have even hinted at using a Bayesian model to determine the point spread of a game. The Bayesian Network models described in Section 2.3.4 do not have the capabilities of updating predictions (although they do update other phenomena) as the game progresses, but these models do show the versatility of Bayesian Networks as they focus more on framework approaches. Although we restrict ourselves

to determining the total points scored by both teams using a Bayesian Network, one could also use a variation of the approach detailed in this thesis to predict the total point differential as the game is being played.

CHAPTER III

BAYESIAN NETWORK DETAILS AND JUSTIFICATION

3.1 Introduction to Bayesian Networks

Bayesian (belief) networks are a class of graphical models that structure probabilistic information in a systematic and intuitive way using graphs (Darwiche, 2010). As mentioned in the Introduction, BNs consist of a set of conditional probability tables computed on all random variables and use these tables to form a directed acyclic graph (DAG). Specifically, given a set of random variables $Z = \{Z_1, Z_2, \ldots, Z_n\}$ describing the quantities of interest, a DAG is created such that each node is associated with one variable Z_i and the arcs that connect them represent direct probabilistic dependencies between the variables. It is reasonable to assume that if there is no arc connecting two nodes, the corresponding variables are either independent or conditionally independent ($\perp\!\!\!\perp$) given a subset of the remaining variables (Scutari & Denis, 2014). The corresponding graph is then used to predict outcomes or diagnose causal effects (if the structure is known), or to discover causal relationships (if the structure is unknown). One can think of Bayesian networks as an extension of the Naïve Bayes classifier. Whereas the NB classifier assumes all (input) variables to be independent of one another and, thus, makes the technique somewhat impractical for applications where variables are highly dependent on one another such as in sports, BNs can model the dependencies between these variables and estimate the joint probabilities whenever variables are not independent.

3.2 The Importance of the Directed Acyclic Graph Structure

Before understanding the importance of the DAG, there is some terminology that requires clarification. A node is called an ancestor of another node if it is connected to it by a direct path. On the other hand, a node is called a descendant (δ) of another node if it can be reached on a direct

path from that other node. A node is said to be the parent (ρ) of another node if it immediately precedes it on the path from the root node to this other node. Because Bayesian networks model the dependencies using a directed acyclic graph, one is guaranteed that there is no node that can be its own ancestor or its own descendant. This, in turn, guarantees that computation of probability values converges. The BN uses this structure to aid in computing the conditional probability tables over all random variables. Specifically, a Bayesian Network assumes that each variable is conditionally independent of its non-descendants in the graph given the state of its parents. Thus, the unique joint probability distribution (JPD) of the random variables can be computed by following **Eq. (3.1)** for each random variable Z_i and its corresponding realization z_i.

$$P(z_1, \ldots, z_n) \equiv P(Z_1 = z_1, Z_2 = z_2, \ldots, Z_n = z_n) = \prod_{i=1}^{n} P\left(z_i \middle| \{z_j\}_{j \in \rho(i)}\right) \qquad \textbf{(3.1)}$$

Then one can just repeatedly apply the probability rule relating joint and conditional probabilities to calculate the conditional probability tables. Given two random variables Z_1 and Z_2, $P(Z_1 = z_1, Z_2 = z_2) = P(Z_2 = z_2 | Z_1 = z_1) P(Z_1 = z_1) = P(Z_1 = z_1 | Z_2 = z_2) P(Z_2 = z_2)$.

3.3 Bayesian Network Learning

There are two characteristics that one can learn regarding a Bayesian Network. The first, structure learning, involves learning the structure of the corresponding DAG associated with the network. This implies specifying which random variables are conditionally dependent on one another and which are conditionally independent given the state of the parents of the corresponding random variables. The task of structure learning from a given data set can be accomplished either by performing constraint-based algorithms or score-based algorithms (Scutari & Denis, 2014). Score-based algorithms represent the application of heuristic optimization techniques to the problem of learning the structure of a BN. This approach first defines how well the BN fits the

overall data and then searches over the space of the DAG for a structure with the maximal score. On the other hand, constraint-based algorithms specify which pair of variables are connected by an arc, regardless of its direction, and then identify the connection between two adjacent nodes to see if they are not conditionally independent on a third one among all the pairs of non-adjacent nodes with a common neighbor (Scutari & Denis, 2014). Although specifying the DAG structure is often impossible, one who has specialty domain knowledge can accurately model the phenomenon and completely specify the graph structure. The network built following this approach is known as an Expert BN. For this thesis, the author has specialty domain knowledge acquired through numerous (11+) years of being an avid basketball, specifically NBA, fan and thus the structure can already be specified when the attributes to learn from have been selected.

The second characteristic that one can learn from a BN is called parameter learning. This involves, after learning the structure of the BN, decomposing the joint distribution of the random variables into local distributions associated with each observed sample to update and estimate the corresponding parameters (Faltin & Kenett, 2007). Maximum likelihood estimation is the most common approach used in the literature to accomplish this task. Given a set of n random variables $(Z_1, \ldots, Z_n)$ and a directed acyclic graph, one can compute the associated joint distribution by following **Eq. (3.1)**. Thus, the parameter vector θ that needs to be learned is the conditional distributions of the specific realization of the random variable given its parent nodes; that is, $\theta_{z_i|\{z_j\}_{j\in\rho(i)}} = P\left(z_i \middle| \{z_j\}_{j\in\rho(i)}\right)$. Specifying a training data set, S, with s independent observations; that is, $S = \{(z_1^r, \ldots, z_n^r), r = 1, \ldots, s\}$, the log-likelihood function can be defined as seen in **Eq. (3.2).**

$$
\begin{aligned}
l(\theta|S) &= log(P(S|\theta)) \\
&= \sum_{r=1}^{s} \log(P(z_1^r, \ldots, z_n^r)|\theta) \\
&= \sum_{r=1}^{s} \log(\prod_{i=1}^{n} P\left(z_i^r \middle| \{z_j^r\}_{j\in\rho(i)}\right)|\theta) \\
&= \sum_{r=1}^{s} \sum_{i=1}^{n} \log\left(\theta_{z_i^r|\{z_j^r\}_{j\in\rho(i)}}\right) \qquad \textbf{(3.2)}
\end{aligned}
$$

3.4 Why a Bayesian Network?

Bayesian networks are ideal when dealing with larger data sets, missing values, discrete variables, many variables and where there exist dependencies between the random variables (Faltin & Kenett, 2007). All these qualities suggest that constructing a BN for the task at hand is the best option. Not only are we working with a large data set (which will be explained in Chapter IV), but there exists a dependency between some of the random variables (i.e., effective field goal percentage and total points scored). Moreover, there is a need to predict, given the observed values of multiple random variables, the probability that the total points scored in a game is greater than the value set by the oddsmakers. It is also worth noting that we are interested in a probability value and not the most likely value. In this regard, the BN is one of the few machine learning algorithms that can be used for the intended purpose. As previously mentioned, BNs allow one to learn the JPD over all the variables in a data set. This, in turn, provides a more versatile model where one can run queries conditioned on multiple predictor variables. It is this JPD learning which interests us as we can use this distribution to aid in the decision-making process.

3.5 Bayesian Network Limitations

BNs are extremely powerful and potent in addressing inferential processes. However, just like every other machine learning algorithm, BNs have their fair share of limitations. For one, learning the structure of a BN is an NP-complete problem and computationally expensive (Jensen,

2009). Additionally, if a categorical predictor variable has a new category which was not observed as part of the training data set, then the model will assign a zero probability. Thus, one must be extremely careful when selecting attributes for the training data set.

3.6 Simple Bayesian Network Example

To give an illustration as to how one can use a BN to predict the probability that the total points in a basketball game is greater than the value set by the oddsmakers, we consider a hypothetical training data set of ten game instances as depicted in **Table 2**. The data set consists of four variables: **3P%**, **eFG%**, **PACE** and the **TOTAL POINTS** scored by one team. **3P%** is associated with how many 3-pointers were made in the game divided by the number of 3-pointers attempted by that team. The **eFG%** of a team is an adjusted value of total shots made divided by total shots attempted by a team that takes into consideration the fact that 3-point field goals are worth 50 percent more than 2-point field goals. Moreover, **PACE** is a statistic that represents the number of possessions per 48 minutes of a team. It is worth noting that **3P%**, **eFG%**, and **PACE** can take infinite values as they are all continuous variables and can cause problems when building the simple Bayesian Network. To maneuver this, it is not unreasonable to bin these values and make them categorical as we do not want to reduce the effects of minor observation errors.

Table 2: Data Used for Simple Bayesian Network Example

Game	3P%	eFG%	PACE	TOTAL POINTS
1	[0.20-0.25)	[0.45-0.50)	[100-105)	99
2	[0.20-0.25)	[0.45-0.50)	[100-105)	102
3	[0.30-0.35)	[0.55-0.60)	[100-105)	122
4	[0.50-0.55)	[0.60-0.65)	[100-105)	121
5	[0.30-0.35)	[0.45-0.50)	[95-100)	90
6	[0.30-0.35)	[0.45-0.50)	[95-100)	102
7	[0.40-0.45)	[0.50-0.55)	[110-115)	131
8	[0.25-0.30)	[0.55-0.60)	[110-115)	140
9	[0.25-0.30)	[0.45-0.50)	[105-110)	109
10	[0.35-0.40)	[0.50-0.55)	[105-110)	116

As previously mentioned **eFG%** is a function of how many three-pointers a team makes in a game; thus, one can assume that this variable is dependent on **3P%**. Moreover, for this example, let us assume **PACE** is conditionally independent of the aforementioned variables (**eFG%** and **3P%**). Finally, it is common knowledge that these variables affect the total points scored by the team as the better a team shoots and the more possessions they have, the more likely the team is to score. By possessing this specialty domain knowledge, one can specify the BN structure. After specifying the BN's structure, it is time to learn the conditional probability distributions. In this example, we model total points as a continuous variable, as opposed to in our experimentation where we model it as a discrete variable that has over 100 distinct values and all the other random variables as discrete. Thus, we have created what we call a Hybrid Bayesian Network. The most natural way to build these kinds of networks is to start with the categorical variables (**3P%**, **eFG%**, and **PACE**) and make the distribution of the continuous variables (**TOTAL POINTS**) depend on it. Again, this is not a representation of the network built in this thesis; it is just for the readers to have an idea on how to build a Hybrid Bayesian Network as it

will not be covered later on. For this example (given that we have a limited data set) let us assume that **TOTAL POINTS** always follow a Normal Distribution. The BN's structure along with the dependence relationships linking the variables can be seen in **Figure 1**.

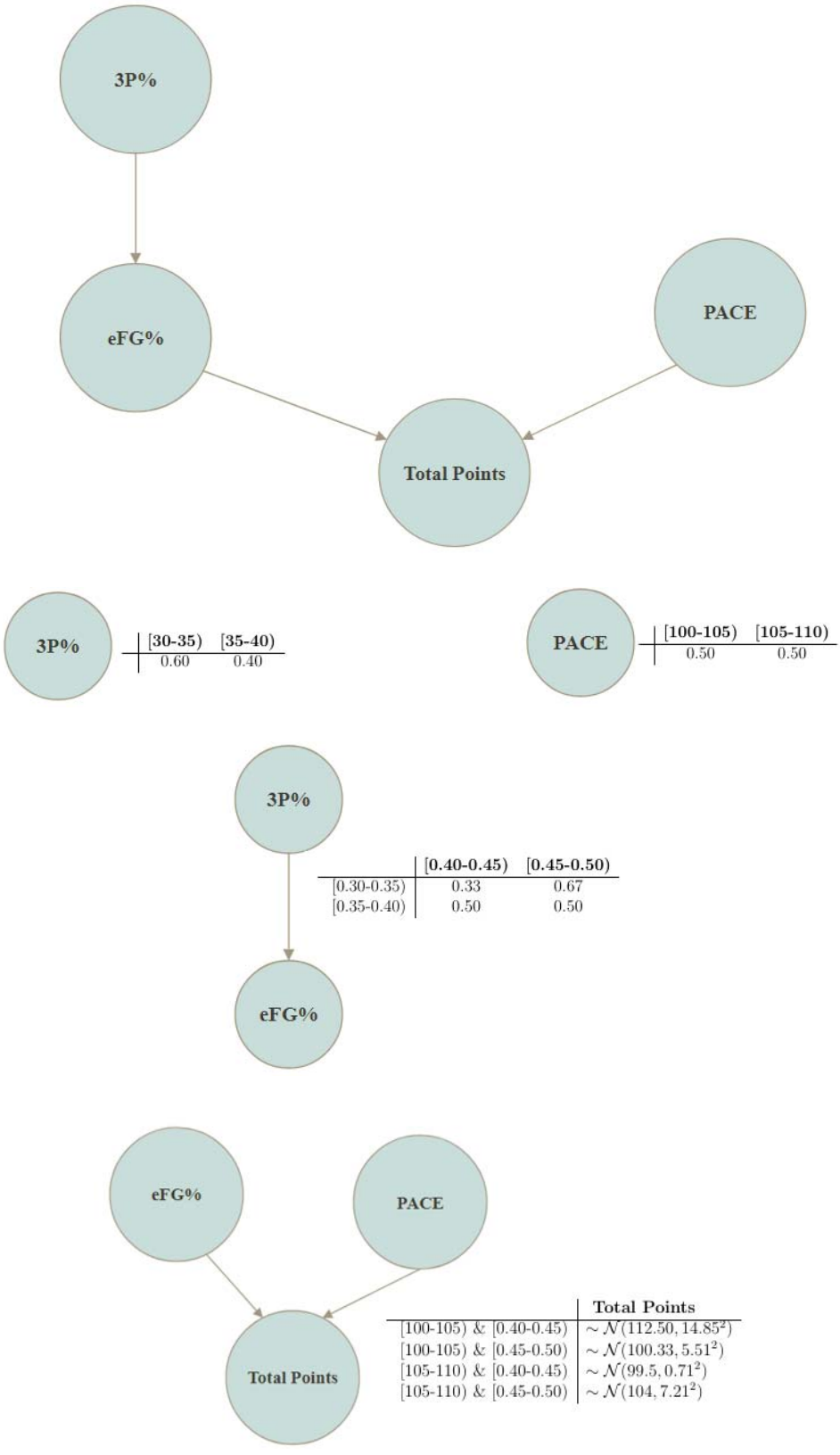

Figure 1: Simple Bayesian Network Example

Now suppose that we are trying to bet on the outcome of a game between the Indiana Pacers and the Cleveland Cavaliers and the oddsmakers set the betting line, after the third quarter, for the total points (**TOTAL POINTS**) scored to 106.5. This leaves the bettor with no option other than to choose to bet Over or Under the points set by the oddsmakers. Moreover, let us assume that, currently, the **PACE** of the game is 103.3, the **3P%** is 0.38, and the **eFG%** is 0.42. Then one can calculate, using the conditional probability distributions depicted in **Figure 1**, the probability that the score is greater than that set by the oddsmakers by following **Eq. (2.3)** to get the JPD of both outcomes normalizing the values.

$$P(\boldsymbol{PACE} = [100 - 105), \boldsymbol{3P}\% = [0.35 - 0.40), \boldsymbol{eFG}\% = [0.40 - 0.45), \boldsymbol{TP} > 106.5)$$

$$= (0.5)(0.4)(0.5)\left(1 - P\left(\frac{106.5-112.50}{14.85}\right)\right)$$

$$= (0.5)(0.4)(0.5)(0.65725)$$

$$= 0.065725$$

$$P(\boldsymbol{PACE} = [100 - 105), \boldsymbol{3P}\% = [0.35 - 0.40), \boldsymbol{eFG}\% = [0.40 - 0.45), \boldsymbol{TP} \leq 106.5)$$

$$= (0.5)(0.4)(0.5)\left(P\left(\frac{106.5-112.50}{14.85}\right)\right)$$

$$= (0.5)(0.4)(0.5)(0.34275)$$

$$= 0.034275$$

Thus, the probability that the total points is greater than that set by the oddsmakers is $\frac{0.065725}{0.065725+0.034275} = 0.65725$. Likewise, the probability that the total points is less than or equal to that set by the oddsmakers is $\frac{0.034275}{0.065725+0.034275} = 0.34275$. Note that these probabilities sum up to one. Therefore, our decision at this point in time would be to bet Over the value set by the oddsmakers. In an all-continuous case, BNs act as simple regression models; thus, they are

rendered obsolete in such applications when compared to the already existing regression models. In an all-discrete case, such as the one built in this thesis, we would compute the probabilities of both teams scoring the distinct values of **TOTAL POINTS** and, given the value set by the oddsmakers, would split the probabilities (into Over/Under that value) and add them together before normalizing them.

3.7 Guide on Computing Probabilities in this Study

As mentioned in the Introduction, our goal is to sequentially update the probability that the total points scored by both teams is greater than the value set by the oddsmakers and be able to use this probability to make live wagering decisions at the end of each of the first three quarters. In Section 3.2, we described the importance of the DAG structure and why it is needed to compute the conditional/joint probability distributions of the in-game statistics used in our model. We hope to define a data set on which to train our BN to compute and store the conditional probability distributions of specific in-game statistics before the game is played. Then, at the end of each of the first three quarters, the bettor can collect the in-game statistics' values and, along with the value set by the oddsmakers, input them into our model to quickly estimate the joint probability distribution by using the stored conditional probability distributions. If the model predicts the joint probability distribution quickly enough, this procedure would allow the bettor to participate in live betting.

CHAPTER IV

DATA COLLECTION AND PREPARATION

4.1 Collection of Data Sets and Additional Features Constructed

In this section, we detail the procedure for collecting the data for our training and testing sets. Additionally, we provide an exhaustive list of the features that were scraped and constructed to compose our data sets. For the features constructed, we briefly explain why we chose to construct them. Throughout this section and the rest of the thesis, we use the terms "attributes" and "features" interchangeably.

4.1.1 Training Data Set

Having the objective of placing wagers at the end of the first, second and third quarters, 54 in-game team statistics or features for each team (108 total features) recorded at the end of each of the first three quarters were scraped from the official NBA website. The scraping was conducted using these Python packages: *Selenium*, *Pandas*, and *NumPy*. The in-game team statistics were collected for every non-overtime game in the five most recent regular seasons of the NBA (2013-2014 through 2017-2018). Overtime games were not considered because they accounted for only 7.804% of the games played during this stretch and, if included, could bias our performance. This bias is because the oddsmakers do not change their value for total points after the fourth quarter has ended, causing these games typically to go Over the value set. Only five recent regular seasons were selected for the training set because of the paradigm shift that has occurred in the association. In recent years, there has been an increase in the pace (average number of possessions for both teams scaled to 48 minutes) of the game and three-point shots attempted (3PA) as shown in **Figure 2**. The game is now shifting to lineups where all players can space the floor (not clog the area near

the basket of the basketball court) and shoot. Despite these restrictions, our training set consisted of 17,361 instances or 5,787 games per each of the three quarters.

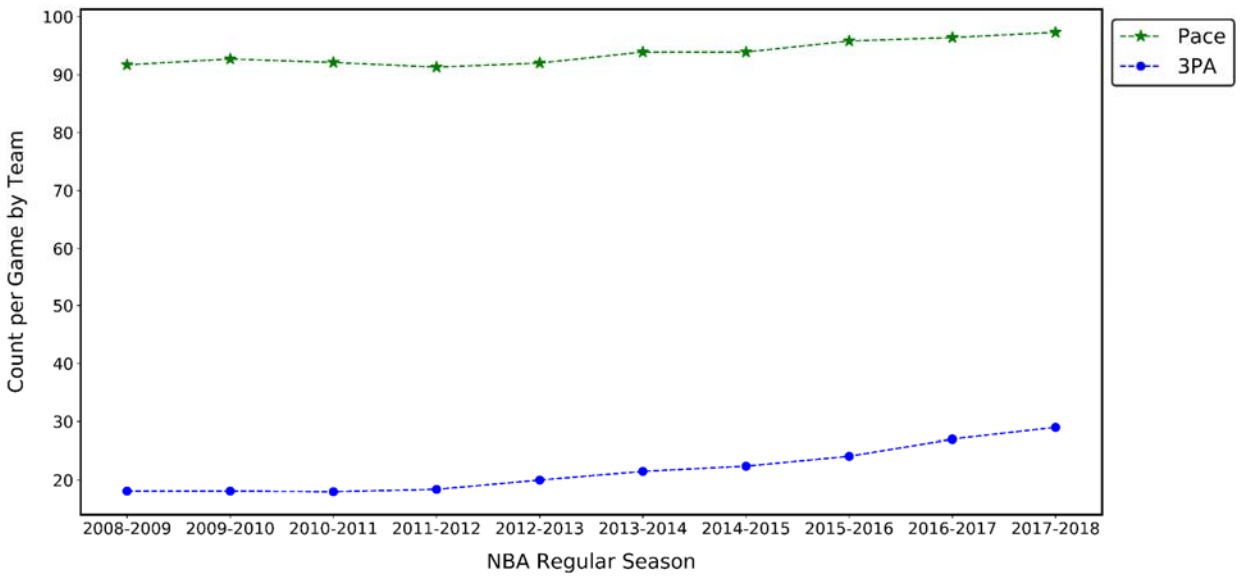

Figure 2: Average Pace and Three-Point Shots Attempted for Individual Teams Last 10 NBA Regular Seasons (Source data from: https://www.basketball-reference.com/leagues)

4.1.2 Test Data Set

As with the training set, 54 in-game statistics were collected for each team (108 total features) by scraping the official NBA website. However, unlike the training set, the games collected to compose the test set were of the unseen (by the BN) 2018-2019 NBA regular season. This was done to have a completely independent and, therefore, unbiased test set on which we can evaluate our network after fitting the probability distributions from our training data. In total, 100 early non-overtime regular season game statistics were scraped at the end of the first, second and third quarters. Thus, our test set consisted of 300 instances or 100 games for each of the three quarters of interest. Moreover, the value set by the oddsmakers for the total points scored by both teams and the Over/Under odds for these values at the end of each of the first three quarters were

collected manually from Bovada, one of the most trusted and efficient gambling websites (Sports Betting Dime, 2018), as each game was being played. Although, as aforementioned, the test set should be roughly 50% as large as the training set, our test set is limited as the oddsmakers do not publish historical odds, and collecting the data manually takes a considerable amount of time. Additionally, having a large data set on which to train our model may give it enough predictive power to outweigh the higher variability we may observe in performance estimation by having a smaller test set (Kotsiantis, 2007). Nevertheless, because the oddsmakers try their best to set 50/50 odds, we consider 300 instances in our test set enough to provide an accurate measure of performance.

4.1.3 Features Scraped and Additional Constructed Features

After collecting both the training and test data sets, we looked into constructing additional features. Most of the in-games statistics that were collected to predict the probability that the total points scored by both teams is greater than the value set by the oddsmakers pertain to individual teams. This is by no means detrimental to our model as we want to consider the flexibility of using both teams' individual statistics to better predict these probabilities. Not having any features, including the total points scored at the end of the game, that describe the game as it pertains to the collective effort of both teams may sway our predictions undesirably. Thus, we constructed seven attributes, in both the training set and the test set, that consider the collective efforts of both teams. All the attributes scraped (S) and constructed (C) are depicted in **Table 3**. Attributes pertaining to teams are recorded both for the home team (no suffix) and the visiting opponent (denoted with _OPP). The interpretations of the in-game statistics depicted in **Table 3** are elaborated in **Table A1** of Appendix A.

Table 3: Exhaustive List of Attributes in the Data Sets

MINUTES PLAYED (S),	**MINUTES PLAYED_OPP** (S),	**2PT FGA %** (S),	**2PT FGA %_OPP** (S),
3PT FGA % (S),	**3PT FGA %_OPP** (S),	**2PT PTS %** (S),	**2PT PTS %_OPP** (S),
2PT MR PTS % (S),	**2PT MR PTS %_OPP**(S),	**3PT PTS %** (S),	**3PT PTS %_OPP** (S),
FBPS PTS % (S),	**FBPS PTS %_OPP** (S),	**FT PTS %** (S),	**FT PTS %_OPP** (S),
PTS OFF TO % (S),	**PTS OFF TO %_OPP** (S),	**PITP %** (S),	**PITP %_OPP** (S),
2FGM % AST (S),	**2FGM % AST_OPP** (S),	**2FGM % UAST** (S),	**2FGM % UAST_OPP** (S),
3FGM % AST (S),	**3FGM % AST_OPP** (S),	**3FGM % UAST** (S),	**3FGM % UAST_OPP** (S),
FGM % AST (S),	**FGM % AST_OPP** (S),	**FGM % UAST** (S),	**FGM % UAST_OPP** (S),
PTS OFF TO (S),	**PTS OFF TO_OPP** (S),	**2ND CHANCE PTS** (S),	**2ND CHANCE PTS_OPP** (S),
PITP (S),	**PITP_OPP** (S),	**FBPS** (S),	**FBPS_OPP** (S),
PTS (S),	**PTS_OPP** (S),	**FGM** (S),	**FGM_OPP** (S),
FGA (S),	**FGA_OPP** (S),	**FG %** (S),	**FG %_OPP** (S),
3PM (S),	**3PM_OPP** (S),	**3PA** (S),	**3PA_OPP** (S),
3P% (S),	**3P%_OPP** (S),	**FTM** (S),	**FTM_OPP** (S),
FTA (S),	**FTA_OPP** (S),	**FT%** (S),	**FT%_OPP** (S),
OREB (S),	**OREB_OPP** (S),	**DREB** (S),	**DREB_OPP** (S),
REB (S),	**REB_OPP** (S),	**AST** (S),	**AST_OPP** (S),
TO (S),	**TO_OPP** (S),	**BLK** (S),	**BLK_OPP** (S),
STL (S),	**STL_OPP** (S),	**PF** (S),	**PF_OPP** (S),
+/- (S),	**+/-_OPP** (S),	**OFFRTG** (S),	**OFFRTG_OPP** (S),
DEFRTG (S)	**DEFRTG_OPP** (S)	**NETRTG** (S)	**NETRTG_OPP** (S),
AST% (S)	**AST%_OPP** (S)	**AST/TO** (S)	**AST/TO_OPP** (S),
AST RATIO (S)	**AST RATIO_OPP** (S)	**OREB%** (S)	**OREB%_OPP** (S),
DREB% (S)	**DREB%_OPP** (S)	**REB%** (S)	**REB%_OPP** (S),
TO% (S)	**TO%_OPP** (S)	**eFG%** (S)	**eFG%_OPP** (S),
TS% (S),	**TS%_OPP** (S),	**PACE** (S),	**PACE_OPP** (S),
PIE (S),	**PIE_OPP** (S),	**FTRATE** (S),	**FTRATE_OPP** (S),
TPQ (C),	**TPPQ** (C),	**TP2QsP** (C),	**AWAY TEAM** (C),
HOME TEAM (C),	**PSUTP** (C),	**TOTAL POINTS** (C)	

4.2 Discretization of Data Sets

Some of the features scraped are continuous quantities. As aforementioned, Bayesian Networks work best with discrete attributes because they simplify the computation of probabilities. If the data set includes only continuous attributes, the supervised learning problem becomes a regression problem which renders a BN impractical when compared to the existing regression models. Thus, discretization is imperative to ensure a successful model.

When discretizing the variables, it is crucial that the same discretization, in terms of bins used for the continuous attributes, is used in both the training set and the test set. This is because

the levels present in the test set had to be observed when the model was learning the distribution from the training set to be predicted. Moreover, to ensure our model does not assign a zero probability to any values of **TOTAL POINTS**, the bins must be wide enough to encompass the values of the features in the test set. This makes discretizing the features one of the most difficult steps in our data pre-processing as we cannot observe the instances in the independent test set to discretize our attributes and, thus, have no idea how to effectively discretize them. At the same time, we need to discretize our features before selecting the attributes we are going to use in the model as it is going to be composed of these discretized features.

Some sophisticated discretization methods have been developed and were considered, such as the Class-Attribute Interdependence Maximization (CAIM) method (Kurgan & Cios, 2004). The goal of this method of discretization is to maximize the dependence relationship between the class (attribute/feature we are trying to predict; response feature) labels and the continuous-valued attributes while minimizing the number of levels (discrete intervals) of each of these attributes. This method uses the advantage of knowing how the attributes affect the class attribute in our training set to perform the discretization. However, this method (along with many others) was ultimately not used because it did not fit the model we are trying to build. The CAIM discretization algorithm is primarily used to determine singular values for the class attribute, which in our case is **TOTAL POINTS**, and transforms the training data accordingly. However, in our model, we want to see all the values that this feature can take and use the value set by the oddsmakers to compute the probability that it is greater than that value. Performing the CAIM discretization method yielded too many levels for each feature which would have made it difficult to find the probability of more than one value of **TOTAL POINTS** for a game. Thus, discretization was performed by observing the minimum as well as the maximum value each feature took in the

training set and using our specialty domain knowledge to determine how wide the intervals should be, which yielded a more desirable number of levels for the features. All features were discretized into equal interval widths of length five from their minimum value to their maximum value except for those features depicted in **Table 4**, which were chosen to be of less or greater width due to their maximum/minimum value being too small or too large, respectively. **Table 5**, in Section 4.3.3, depicts the discretization intervals as well as the minimum and maximum values of the features ultimately selected to train our model.

Table 4: Attributes Discretized into Different Bin Widths than Five

Features	Width of Each Bin
OREB	2
DREB	2
REB	2
AST	2
TO	2
BLK	2
STL	2
PF	3
+/-	4
OFFRTG	10
DEFRTG	10
NETRTG	10
AST/TO	2
PACE	10
PIE	10

4.3 Feature Selection

Although collecting a vast number of attributes is generally beneficial, it may be worthwhile to train a model on only a subset of these (Aggarwal, 2015). This is especially true in this work, as a Bayesian Network assumes each variable is conditionally independent of its non-descendants in the graph given the state of its parents. As part of our job is to specify the structure,

it would be impractical to determine which attributes are conditionally independent of one another when there are so many. Not only will the network have an enormous number of arcs, but computing the conditional probabilities will be computationally expensive and, thus, impractical as we need to predict the probabilities as the game is being played. Moreover, although not common knowledge, a lot of the attributes that have been scraped are duplicates such as **MINUTES PLAYED** and **MINUTES PLAYED_OPP**; **OFFRTG** and **DEFRTG_OPP;** and **PACE** and **PACE_OPP**. Many of these attributes are also related such as **FGM** and **FGA** with **FG%.** Involving only one member of each such set of features is enough as those selected features provide our model with the same kind of information as the whole collection. Including redundant features may bias our Bayesian Network's performance.

Generally, there are two types of methods one can use when performing feature selection: filter-based methods and wrapper-based methods (Dhar, 2013). Filter-based methods are independent of a learning algorithm and are based on individual scores such as statistical correlation and information gain in relation to the class attribute. On the other hand, wrapper-based methods involve creating multiple models using a learning algorithm with different subsets of features and adding/subtracting some features until the best model is found. This task is essentially a search problem. There have been well-regarded wrapping approaches for feature selection using Bayesian Networks (Inza et al., 2000 and Drugan, M. M., & Wiering, 2010). Although, if directly related to the learning algorithm, wrapper-based methods generally perform better than filter-based approaches, the number of features in our training data set makes them extremely time-consuming because the wrapper-based method must consider all possible combination of features. Thus, we perform feature selection using a combination of filter-based methods and specialty domain

knowledge. The filter-based methods used are the information gain ratio and the chi-square test of independence.

4.3.1 Information Gain Ratio

Originally developed to select the attribute at each node in the decision tree, information gain describes the amount of information gained about an attribute from observing another attribute (Karegowda et al., 2010). Rapidly becoming one of the most widely-used filter-method approaches for feature selection in discrete sets (classification), information gain is independent of the learning algorithm, computationally fast and can be used for a large dimensional data set. Rigorously, let S be a data set consisting of s instances with c distinct classes/values (which can be mapped to integers). The entropy of the whole data set is then given by **Eq. (4.1)**, where p_i is the probability that a sampled instance belongs to a specific class i and is estimated by $\frac{s_i}{s}$ where s_i is how many times that specific class value appears in the data set.

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{4.1}$$

Then, given a specified attribute denoted by A, let A_v denote the instances in the data set where A takes value v. To compute the information gained from the attribute on a specific class i, follow **Eq. (4.2)**. Specifically, let $H(S_A)$ be the entropy of a specific value for the attribute found by partitioning the data to instances where A takes value v.

$$Gain(S, A) = H(S) - \sum_{v \in A} \frac{s_{A_v}}{s} H(S_A) \tag{4.2}$$

Although insightful, information gain is biased towards the number of distinct values each attribute can take in a data set (Quinlan, 1986). Thus, information gain ratio takes this into account

by normalizing the information gain of the attributes according to the intrinsic value formulation depicted in **Eq. (4.3).**

$$IV(S_A) = -\sum_{v \in A} \frac{S_{Av}}{S} \log_2\left(\frac{S_{Av}}{S}\right) \quad \textbf{(4.3)}$$

After acquiring the intrinsic value formulation for each attribute and calculating its information gain for the whole data set, the information gain ratio for each attribute can be computed by following **Eq. (4.4)**.

$$Gain\ Ratio\ (S, A) = \frac{Gain(S,A)}{IV(S_A)} \quad \textbf{(4.4)}$$

This technique was applied to our data set after the features had been discretized. The 25 features with the highest information gain ratio in relation to our class attribute are depicted in **Figure 3**. As can be seen in the chart, some of the attributes that we constructed are the top features by this measure. It is also important to note that some features provide our model with the same information and, thus, the figure includes multiple redundant features.

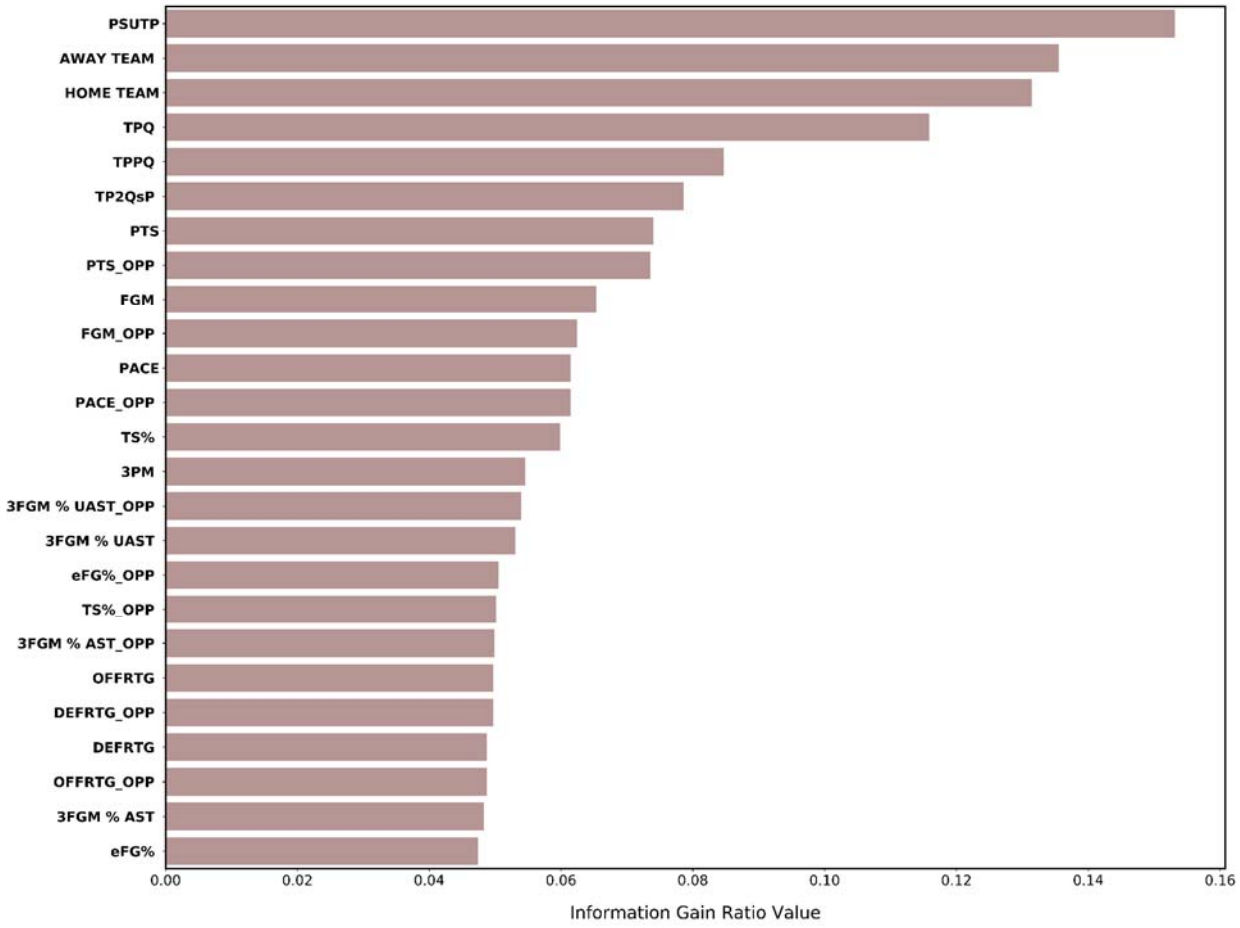

Figure 3: Top 25 Features by Information Gain Ratio in Relation to **TOTAL POINTS**

4.3.2 Chi-Square Test of Independence for Feature Selection

The chi-square test is a statistical test that measures the dependence or association between categorical or nominal data. Generally, the chi-square statistic compares the tallies or counts of observed categorical responses in a data set between two or more independent groups. Then, these categorical responses' expected counts are approximated by assuming independence. By quantifying how much the observed responses deviate from the expected ones, a statistical test can be conducted to infer whether the two variables are related (Chernoff & Lehmann, 1954). In the context of feature selection, the independent groups for which we approximate the expected count are the different values for a specific feature in a data set and the different values for a specific

class in the data set. Thus, we essentially test whether the values the feature takes and the values the class takes are independent (Guyon & Elisseeff, 2003). We calculate the chi-square statistic between each attribute and our class attribute by following **Eq. (4.5).** Given a data set S with s instances and c distinct classes and an attribute A with v distinct levels, we compute O_{ij} as the number of observations where class i and feature value j coincide in the data set. We also compute $E_{ij} = \frac{s_i s_j}{s}$, which is the expected number of observations if feature value j were independent of class i, where s_i, s_j are the number of times class value i and feature value j appear in the data set, respectively. We then use the limiting distribution, χ^2 with $(c-1)(v-1)$ degrees of freedom, to acquire a p-value and help us determine if that feature is independent of the class attribute by setting this independence assumption as our null hypothesis. If it is, then we can discard the feature. On the other hand, if the feature and the class attribute are dependent on each other, the feature is selected in the model.

$$\chi^2_{Statistic} = \sum_{i=1}^{c} \sum_{j=1}^{v} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad \textbf{(4.5)}$$

When the two events are independent, the observed count is close to the expected count; thus, a small $\chi^2_{statistic}$ value is obtained. A high value of this quantity indicates that the hypothesis of independence is incorrect. Therefore, the higher the chi-square statistic, the more likely the feature should be selected for model training (Sarkar & Goswami, 2013). **Figure 4** depicts the 25 attributes that had the highest chi-square statistic score in relation to the class attribute **TOTAL POINTS**. We can again observe that the constructed features are among the top. However, just as when using information gain, many features listed are redundant and do not provide more information to our model.

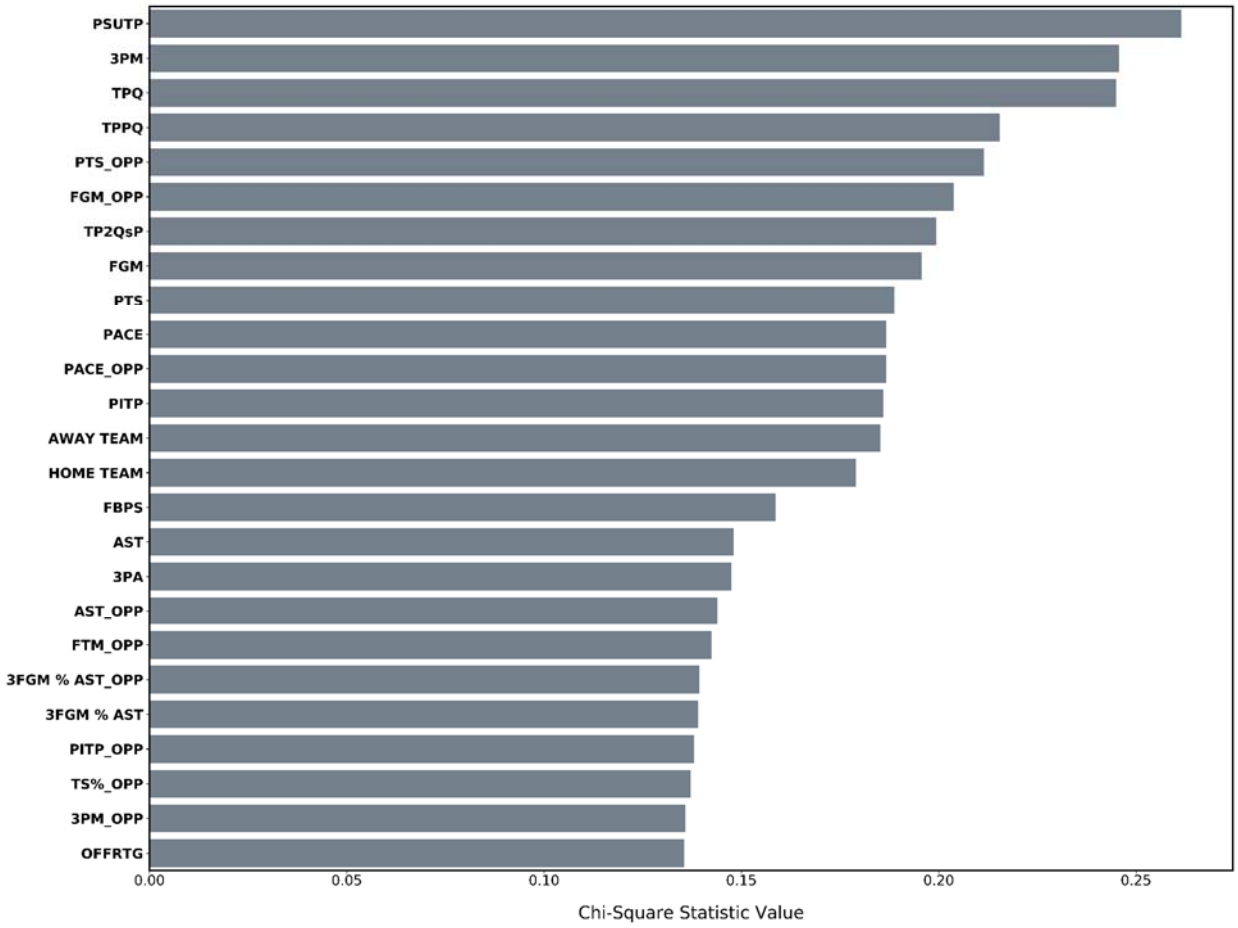

Figure 4: Top 25 Features by Chi-Square Statistic in Relation to **TOTAL POINTS**

4.3.3 Final Learning Features and Validation

When performing feature selection using filter-based methods, it is important to conduct more than one of these methods. Specifically, we always want at least one statistical correlation method, such as the chi-square statistic, and a method that relies solely on the data set we are trying to learn from and not on theoretical quantities, such as information gain ratio. By learning from these two methods to perform feature selection, we avoid selecting attributes based on only one metric. As can be seen in the results in **Figure 3** and **Figure 4**, both methods list the constructed attributes for points by both teams in the current quarter and previous quarters to be important. This is not coincidental, as our class attribute itself is a value that is acquired from adding these

attributes. Nevertheless, by observing which features were selected by both methods and using domain knowledge to determine if the feature was redundant or if any other features were important, we selected the 17 attributes to be used for our model. These attributes along with their discretization bins are depicted in **Table 5**. Each bin's right value is not inclusive unless it includes the overall maximum value (i.e. The bins for **3PM** are [0,5), [5,10), [10,15), [15,20) and [20,25]). Note **MINUTES PLAYED** was already a factor/discrete attribute with three levels: 12, 24 and 36. These represent the minutes played at the end of the first three quarters, respectively.

Table 5: Description and Discretization of Features Selected

Features	Description	Value Limits for Binning	Width of Each Bin
MINUTES PLAYED (S)	Amount of time that has passed in the game	{12, 24, 36}	Each Distinct Value
FGM (S)	Number of shots made by home team	[0, 50]	5
FGM_OPP (S)	Number of shots made by away team	[0, 45]	5
3PM (S)	Number of three-point shots made by home team	[0, 25]	5
3PM_OPP (S)	Number of three-point shots made by away team	[0, 25]	5
eFG% (S)	Home team's shooting efficiency; accounts for worth of **3PM**	[5, 105]	5
eFG%_OPP (S)	Away team's shooting effiency; accounts for worth of **3PM_OPP**	[5, 110]	5
TS% (S)	Same as **eFG%** but also accounts for free throws made	[15, 105]	5
TS%_OPP (S)	Same as **eFG%_OPP** but also accounts for free throws made	[15, 105]	5
OFFRTG (S)	Total points of the home team per 100 possessions	[25, 190]	10
DEFRTG (S)	Total points of the away team per 100 possessions	[25, 180]	10
PACE (S)	Avg. number of possessions for both teams scaled to 48 minutes	[75, 130]	10
TPQ (C)	Total points scored by both teams in quarter	[15, 90]	5
TPPQ (C)	Total points scored by both teams in previous quarter	[0, 90]	5
TP2QsP (C)	Total points scored by both teams two quarters previously	[0, 90]	5
PSUTP (C)	Total points scored by both teams up to that point	[20, 220]	10
TOTAL POINTS (C)	Total points scored by both teams at the end of the game	{134, 140, . . . , 275}	Each Distinct Value

Perhaps the most surprising decision with regard to the attributes selected is the inclusion of **MINUTES PLAYED**, as neither of our two methods identifies it as a top 25 attribute. It is important to remember that, when using filter-based methods, we only consider the attributes in relation to our class attribute, **TOTAL POINTS**. When collecting data, the time remaining in the game does not affect the total points scored by both teams according to our measures, as it will list the same value of **TOTAL POINTS** for the game regardless of the quarter being played. But, the

time remaining impacts the total points scored by both teams significantly because the pace of the game slows down with each passing quarter as the players get more fatigued and more concentrated on the defensive side of the game. Also, if we do not include this attribute it is very unlikely that our model will update its predictions as the game is being played. Moreover, the attributes selected accumulate over time, which means they have a strong dependence with **MINUTES PLAYED** and it is important to model this dependency.

Features that were not included and may surprise some are the **AWAY TEAM** and **HOME TEAM** for the game. Although selected as top 25 attributes using both measures, having the probabilities dependent on teams itself is quite detrimental to our model. Each team plays 82 games a season and, as soon as which team is playing becomes a factor when computing the conditional probabilities, we have fewer games to learn from for each team. Additionally, in every season teams trade players, players leave to go to other teams, or new star players emerge. Although some teams remained consistent the last five seasons, as is the case with the Golden State Warriors, most teams fluctuate in terms of performance and overall statistics every year, like the Miami Heat did when LeBron James returned to the Cleveland Cavaliers in the 2014-2015 regular season. For these reasons, we believe that the in-game statistics will be enough to make our model profitable without having to specify the teams matched up in the game.

Although the need for feature selection was evident and it was performed to the best of our knowledge, it must be validated to ensure that the model in fact performs better by learning from only the selected features than from all of them. Therefore, we created two Bayesian Networks using the R *bnlearn* package's (Scutari, 2010) hill-climbing algorithm. A score-based algorithm, the hill-climbing method is a greedy search on the whole space of complete directed graphs. Given a starting point, it adds, deletes, or reverses a possible arc in a graph and computes the score of the

graph successively until it optimizes the graph for a specific scoring measure. Clearly, there are too many attributes to set each arc and test all possible combinations until finding the optimal one. Thus, the algorithm follows a heuristic approach where it uses score caching, score decomposability and score equivalence to reduce the number of performance tests (Daly and Shen, 2007).

The score we used to perform the hill-climbing algorithm and, then, compare the networks learned from different set of features was the Bayesian Information Criterion (BIC). Originally developed by Schwarz, BIC acts as a model selection tool. If a model is built on a training set, the BIC score gives an estimate of the model performance on an unseen testing set (Schwarz, 1978). The criterion reduces the risk of over-fitting by introducing a penalty term that grows with the number of parameters to filter out unnecessarily complicated models. BIC is usually given by the formula depicted in **Eq. (4.6)**, where *n* is the number of parameters (features) we are trying to fit, *s* is the number of instances in our training set and $l(\theta|S)_{max}$ is the maximized value of the log-likelihood function of the model previously described in Section 3.3. A smaller BIC indicates a better model; however, the R package *bnlearn*'s BIC is given by the formula depicted in **Eq. (4.7)** which is the classical definition rescaled by -2 and, thus, a higher (less negative) BIC indicates a better model.

$$BIC_{classic} = -2l(\theta|S)_{max} + n\,(\ln(s)) \quad \textbf{(4.6)}$$

$$BIC_{bnlearn} = l(\theta|S)_{max} - n\frac{\ln(s)}{2} \quad \textbf{(4.7)}$$

The BIC for the Non-Expert Bayesian Networks that were constructed by not using feature selection and by using feature selection can be compared in **Table 6**. As aforementioned, these networks' structures were both specified by the hill-climbing algorithm and, therefore, we can

compare their scores without bias. It is important to note that the BIC itself is calculated using the training set and it gives an estimation for model performance on the test set.

Table 6: BIC Comparison of Non-Expert Bayesian Networks

	BIC
Structure Specified using All Features	**-1,361,596.46**
Structure Specified using Features Selected	**-406,404.10**

As depicted in **Table 6**, the BIC of the Non-Expert Bayesian Network whose structure was specified using the features selected is significantly greater (by 70%) than that of the network that uses all the attributes to specify its structure. Although not surprising, it means that our initial assumption of having too many redundant features was most likely the case. Nevertheless, this proves, quite crudely, that the 17 features selected do a better job at mirroring the dependence structure of the data (Scutari & Denis, 2014). The DAG associated with the Non-Expert Bayesian Network for the features selected, which from now on will be referred to as NEBN_FS, can be seen in **Figure 5**.

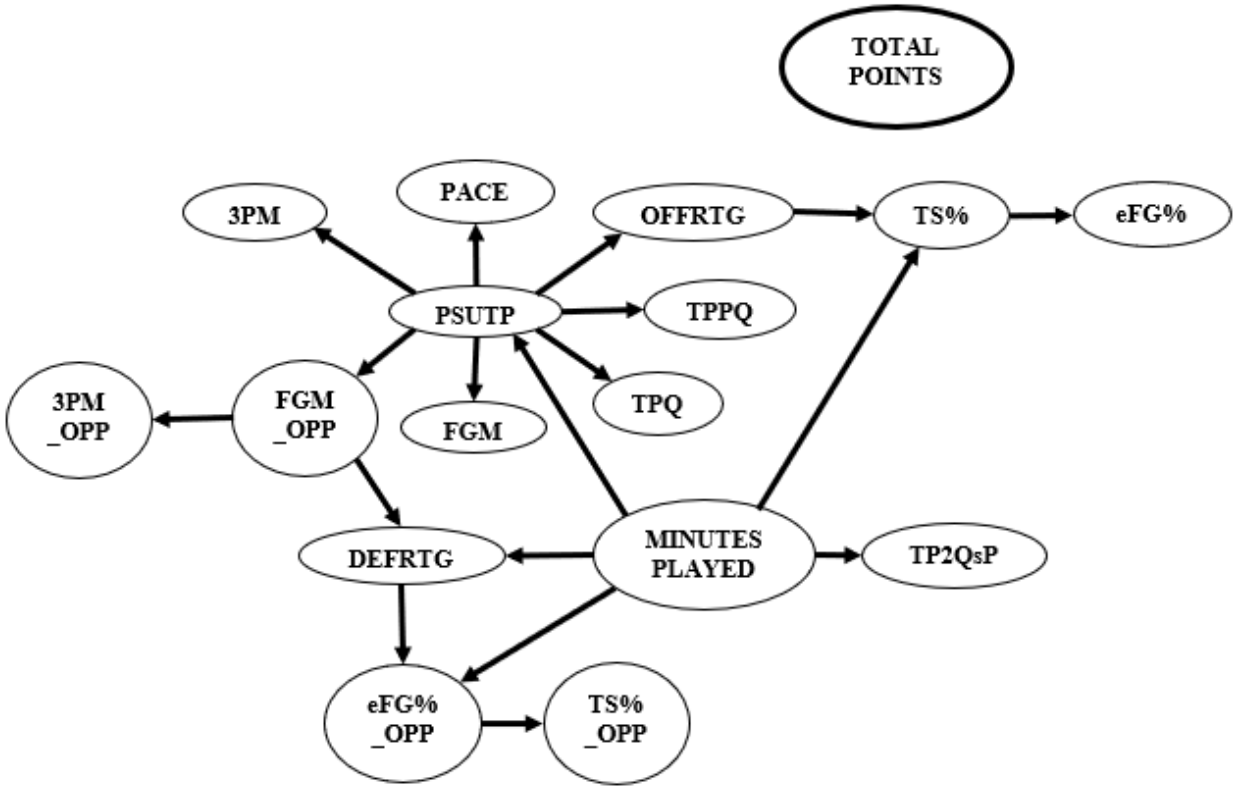

Figure 5: Directed Acyclic Graph of the Non-Expert Bayesian Network

CHAPTER V

EXPERIMENTAL DESIGN

5.1 Introduction

In this chapter, we explain how we use an association measure derived from the chi-square test to help us model, along with our specialty domain knowledge, the dependence of the directed acyclic graph constructed in this study. We detail the statistical test used to specify the structure of the Expert Bayesian Network and make an initial comparison to the network built in Section 3.3.4 using the BIC measure detailed in the same section. Finally, we explain how we compute the probability that the total points scored by both teams is greater than the value set by the oddsmakers before evaluating the model in Chapter VI.

5.2 Cramer's V Measure of Association

Cramer's V is a measure of association between two discrete variables. It is derived from the chi-square statistic described in Section 3.3.3 and it assigns a value of association between the two variables of interest within the range $[0,1]$ where 0 indicates no association and 1 indicates that, if we know the value of an attribute, we can perfectly predict the other corresponding attribute's value (Cramer, 1946). Although it provides the same information as the chi-square statistic, it allows us to compare intuitively how much more an attribute is dependent on another one (like a correlation measure). Cramer's V measure of association between two attributes can be computed by following **Eq. (5.1)** where $\chi^2_{Statistic}$ is obtained from two attributes by following **Eq. (4.5)**, s is the number of instances in a data set S and v_1, v_2 are the number of distinct values for the two attributes, respectively.

$$V = \sqrt{\frac{\chi^2_{Statistic}}{s \min(v_1, v_2)}} \tag{5.1}$$

Figure 6 depicts a heat map of Cramer's V measure of association between the features selected to construct the Expert Bayesian Network. As can be seen, **MINUTES PLAYED** has a comparatively strong association with some of the other features; therefore, it shows that our assumption for why we selected it to construct our final model was adequate. This heatmap helps show the dependency between the features and was one of the main tools used to specify the structure of the BN.

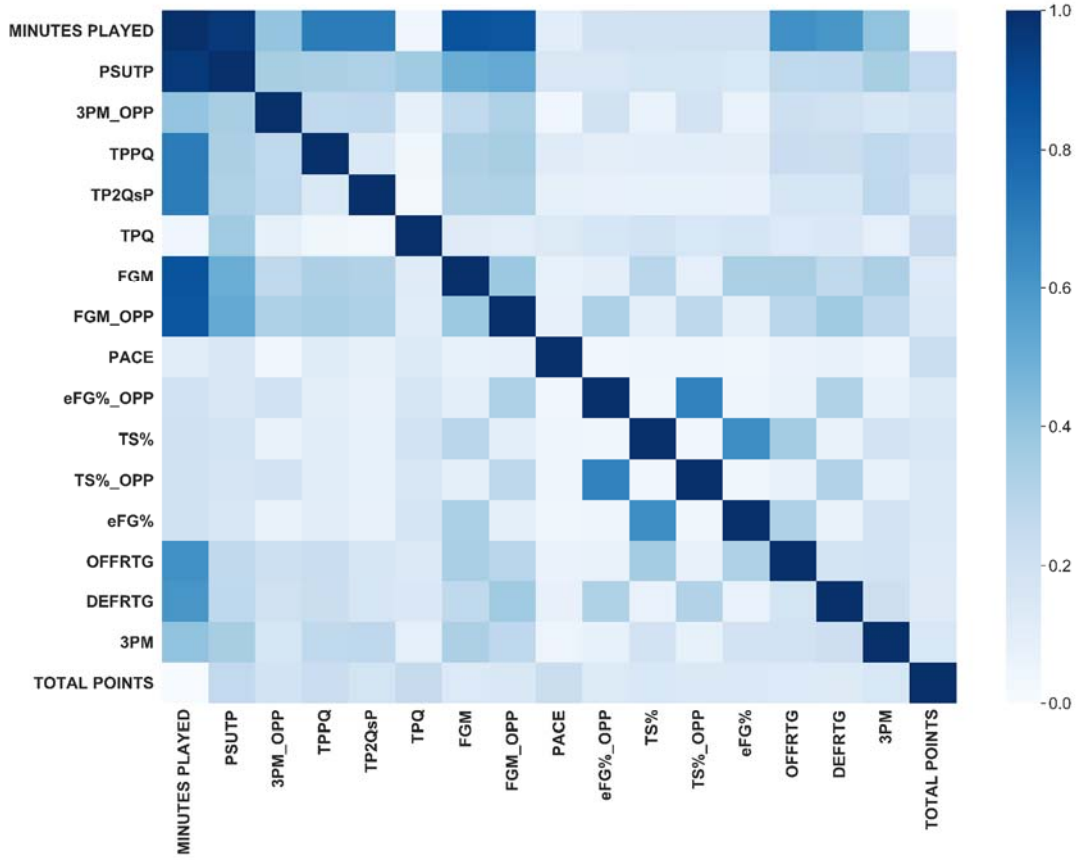

Figure 6: Cramer's V Measure of Association Between Pairs of Selected Features

5.3 Chi-Square Test for Conditional Independence

We modeled Bayesian Networks by assuming that each node is conditionally independent of its non-descendants given the state of its parents. Thus, it would be wise to run tests of conditional independence to make sure we are not violating this assumption. By running conditional independence tests on the arcs of the Bayesian Network, we can assess the probabilistic dependence encoded in each arc and determine whether this dependence is supported by the data (Scutari & Denis, 2014). Namely, we define the two attributes (nodes) connected by a directed arc as A_1 and A_2, where A_1 is the tail and A_2 is the head, and let $P = \{\rho_1(A_2), \ldots, \rho_n(A_2)\}$ be a set containing all the other n parent nodes or features of A_2, thus $A_1 \notin P$. We can then set up a test using a variation of the chi-square test of independence previously detailed in Section 4.3.2 to test for conditional independence between the two nodes. By setting our null hypothesis to test whether A_2 is conditionally independent of its non-descendants given the state of its parents (not including A_1), that is, $H_o: (A_2 \perp\!\!\!\perp \mathrm{A}_1)|P$ and having the alternative hypothesis be the opposite, that is, $H_A: \left(A_2 \not\perp\!\!\!\perp A_1\right)|P$, we can determine whether an arc should be included in the directed acyclic graph. Clearly, we want to reject the null hypothesis with respect to each arc to make sure all the children of the nodes in the network are dependent on their parents.

We compute the $\chi^2_{Statistic}$ for conditional independence testing by following **Eq. (5.2)**. For the context of this thesis, let us assume again that there is an arc directed from A_1 to A_2. Specifically, given a data set S with s instances, attributes A_1 and A_2 with v_1 and v_2 distinct levels, respectively, and all τ of the configurations of all possible combinations of the parent variables of A_2 (not including A_1), we compute O_{ijk}, which is the number of instances where A_1 value i, A_2 value j and the kth configuration of the parent nodes of A_2 are observed in the data set. We also

compute $E_{ijk} = \frac{\sum_{j=1}^{v_2} s_{ijk} \sum_{i=1}^{v_1} s_{ijk}}{\sum_{i=1}^{v_1} \sum_{j=1}^{v_2} s_{ijk}}$, which is the expected number of observations if A_2 value j were independent of A_1 value i given the kth configuration of A_2's parent nodes where s_{ijk} is the number of times A_1 value i, A_2value j, and this kth configuration appear in the data set.

$$\chi^2_{Statistic} = \sum_{i=1}^{v_1} \sum_{j=1}^{v_2} \sum_{k=1}^{\tau} \frac{(O_{ijk} - E_{ijk})^2}{E_{ijk}} \quad \textbf{(5.2)}$$

This test, just like the previous chi-square test, has an asymptotic χ^2 distribution under the null hypothesis (Scutari & Denis, 2014). Unlike the previous one, however, the degrees of freedom of this test also depend on the parent nodes of the attribute to which the arc is directed. Namely, let the parent nodes, $\rho_1, \ldots, \rho_n$, of A_2 have $u_1, \ldots, u_n$ levels. The degrees of freedom for the test will then be $(v_1 - 1)(v_2 - 1)(\prod_{i=1}^{n} u_i)$. Thus, by setting a significance level, we can find the critical values of the distribution and then use the $\chi^2_{Statistic}$ to test the dependence of the attributes connected by the arcs.

5.4 Expert Bayesian Network

5.4.1 Methodology

With a foundation as to which attributes are dependent on each other through Cramer's V measure of association, we constructed the Bayesian Network with a specified structure. By running the conditional independence test on the arcs illustrated in **Figure 5**, we found out that all the parent-child pairs of the NEBN_FS are indeed dependent. The maximum p-value of the test detailed in Section 5.3 over all the arcs was $< 2.2 * 10^{-16}$; therefore, we reject the null hypothesis, that these nodes are independent of their parent nodes given all their other parent nodes, as there is enough evidence to suggest that they are dependent on them. After verifying that the hill-climbing algorithm constructed reasonable arcs, we examined these arcs and the heatmap in

Figure 6 to specify the Expert Bayesian Network's structure. After considering many combinations and tinkering with the arcs to match our assumptions, the structure of its DAG was finalized as illustrated in **Figure 7.**

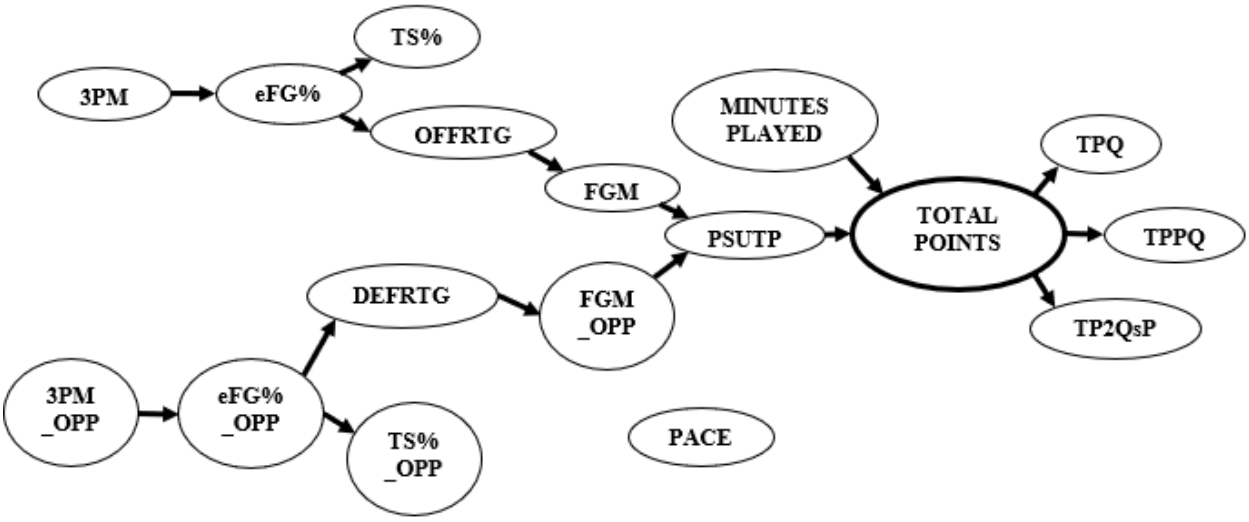

Figure 7: Directed Acyclic Graph of the Expert Bayesian Network

5.4.2 Initial Comparison to NEBN_FS

First, the same conditional independence test was performed on all arcs in the Expert Bayesian Network, and they all gave a p-value $< 2.2 * 10^{-16}$. Therefore, as was the case for NEBN_FS, we reject the null hypothesis, that each node at the head of an arc is independent of that parent node given the state of all their other parent nodes. When comparing the arcs of the Non-Expert Bayesian Network using feature selection whose structure was specified using the hill-climbing algorithm (shown in **Figure 5**) to our Expert Bayesian Network's arcs (seen in **Figure 7)**, some similarities but also some major differences are noted.

Some arcs are included in both networks but reversed in the Expert Bayesian Network, as is the case with the arcs from **DEFRTG** to **FGM_OPP**, **eFG%** to **TS%**, **FGM** to **PSUTP** and

FGM_OPP to **PSUTP**. The direction of these arcs was reversed based on specialty domain knowledge, knowing how attributes depend on each other. The arc from **eFG%_OPP** to **TS%_OPP** stayed the same. A major difference in the structure of the networks results from the decision to remove all the arcs from **MINUTES PLAYED** to the other variables and add an arc from **MINUTES PLAYED** to **TOTAL POINTS** in the Expert Bayesian Network. Adding the latter arc is not surprising as we want **TOTAL POINTS** to be directly dependent on the end of the quarter when we are making our wager, but the removal of the others can be questioned. Although one of the primary reasons to include **MINUTES PLAYED** in our final set of features was because of the dependence of other attributes on it, the data proved, through statistical tests, that an arc from **MINUTES PLAYED** to these other attributes is not needed, as these attributes are independent of the time given their new parents. One last, unorthodox, approach to specifying our structure was that, instead of having the points at the end of each quarter be dependent on **PSUTP**, we had the total points scored by both teams at the end of the game, **TOTAL POINTS**, depend on **PSUTP** and these attributes be dependent on **TOTAL POINTS**. This was done because of how we calculate the probabilities to aid in our decision-making process, as we are finding the probability distribution of **TOTAL POINTS** instead of just predicting the attribute's value. This was the only change to the network's structure driven by the purpose of the BN rather than the data.

We computed the BIC of both networks, seeing how this measure was used to validate feature selection, to see which network was better supported by the data. The BIC values for both networks can be seen in **Table 7**. Unsurprisingly, the BIC of the Non-Expert Bayesian Network is greater than that of the Expert Bayesian Network, which indicates that the latter is a better model

for our data. Obviously, because the Non-Expert Bayesian Network was constructed using an algorithm that maximizes this score, we expect it to perform better according to this metric.

Table 7: BIC Comparison of Bayesian Networks

	BIC
Expert Bayesian Network Score	**-530,496.75**
NEBN_FS Score	**-406,404.10**

Although the BIC score of the Non-Expert Bayesian Network is greater than that of the network whose structure we specified, this is just one measure used to compare the networks. Moreover, the BIC score considers all the attributes in the data set and how likely we are to predict all those attributes given a new unseen data set (Scutari & Denis, 2014). For our purpose, we are interested in seeing how well we can predict only one of those attributes, namely **TOTAL POINTS**, given an unseen set. If we use a different measure that is not directly related to how either of the networks were built and is therefore unbiased, we can see that our Expert Network does a better job at predicting the value of **TOTAL POINTS.** Using the same package (*bnlearn*) as before, with the same rescaling factor of -2 from the classical definition, we computed the Akaike Information Criterion (AIC) according to **Eq. (5.3)**.

$$AIC_{bnlearn} = l(\theta|S)_{max} - n \qquad \textbf{(5.3)}$$

Table 8 shows that, using this unbiased measure on the specific node for **TOTAL POINTS**, the Expert Bayesian Network is a better predictor of this attribute than the Non-Expert Bayesian Network. Although the Non-Expert Bayesian Network yields a more favorable overall BIC score, the Expert Bayesian Network yields a more favorable (higher valued) AIC score for

the attribute in question. While the improvement in AIC for the class attribute is encouraging, the real question is whether either network can provide information for profitable betting.

Table 8: AIC Comparison on the Class Node

	AIC
Expert Bayesian Network Score	**-74,998.75**
NEBN_FS Score	**-75,907.90**

5.5 Calculating the Probabilities

To answer the question posed in Section 5.4.2, we needed to see which network was better in terms of performance. This, as aforementioned, involves estimating the probability that **TOTAL POINTS** is greater than the value set by the oddsmakers. We modeled this task as a classification problem where the class attribute, **TOTAL POINTS** (Y), could take any of over 100 different values. Given the value of our in-game statistics, $\{z_1, \ldots, z_n\}$, and the value for **TOTAL POINTS** set by the oddsmakers, y^*, we estimated $\phi_G = P(Y > y^* | z_1, \ldots, z_n)$ using the Bayesian Network constructed. To accomplish this, we obtained the JPD of all the attributes, including the class attribute, and extracted the probabilities with the values of $z_1, \ldots, z_n$ fixed. Then, we summed up the probabilities for those values of **TOTAL POINTS**, denoted by y, that were greater than the value set by the oddsmakers; that is $\hat{\phi}_G = \sum_{y>y^*} p(z_1, \ldots, z_n, y)$. Next, we estimated ϕ_L by $\hat{\phi}_L = \sum_{y \leq y^*} p(z_1, \ldots, z_n, y)$. Finally, we calculated the probability that **TOTAL POINTS** is greater than the value set by the oddsmakers by normalizing $\hat{\phi}_G$ and $\hat{\phi}_L$ to sum to one. For our decision process, if the derived probability of **TOTAL POINTS** being greater than the value set by the oddsmakers, $\widehat{\Psi}_G$ depicted in **Eq. (5.4)**, was greater than 0.5, we wagered Over. If not, we wagered Under.

$$\widehat{\Psi}_G = \frac{\hat{\phi}_G}{\hat{\phi}_G + \hat{\phi}_L} \quad \textbf{(5.4)}$$

CHAPTER VI

MODEL EVALUATION

6.1 Results

The results of the models were evaluated according to both the accuracy and profitability of bets placed. Moreover, the time it took for the BN to predict the probability that the total points scored by both teams at the end of the game was greater than the value set by the oddsmakers was examined. In this chapter, we provide an in-depth discussion on the comparison of results for the BNs constructed, a Naïve Bayes classifier, and three amateur betting strategies.

6.1.1 Accuracy Results

Before evaluating the Non-Expert Bayesian Network built with the features selected and the Expert Bayesian Network, we wanted to simulate typical amateur betting strategies. Specifically, for each matchup, we calculated the average points scored by each team during the previous 3, 5, and 15 games and summed the two teams' averages to decide our wagering decision for those bets at the end of each of the first three quarters. If the total of the average points scored in the last n games by each team was greater than the value set by the oddsmakers, we would wager Over. Otherwise, we would wager Under for the respective amateur betting strategy. It is important to note that the BNs' probability changed as the game progressed because the value set by oddsmakers usually shifted at the end of each quarter, while the amateur betting strategies' prediction for the total points scored by both teams did not. Thus, our comparison is not fair, but there is no better way to compare these amateur betting strategies to the BNs.

Moreover, to compare the BNs to another model that could update the probability that the total points scored by both teams was greater than the value set by the oddsmakers, a Naïve Bayes classifier was built using the *bnlearn*'s package *naive.bayes* function on the final discretized

training set, which contained only the 17 features selected as detailed in Section 4.3.3. This function constructs the star-shaped DAG form of a Naïve Bayes classifier, which contains only arcs directed from the class attribute (which needs to be specified within the function), **TOTAL POINTS**, towards each of the other features. The Naïve Bayes classifier's performance was evaluated in the same way as the BNs, which is described in Section 5.5.

Figure 8 depicts the results of the overall accuracy (if the favorable option decided by the strategy was correct) of Over/Under bets at the end of all three quarters for all the different methods applied to the test set games. For the Expert Bayesian Network and the Naïve Bayes classifier, there were some instances (four for the Expert BN and two for the NB classifier) where the model returned an "N/A" value for the probability that the total points scored by both teams would be greater than that set by the oddsmakers. We decided to omit these instances by assuming the bettor would not wager on a game if they did not obtain a probability value.

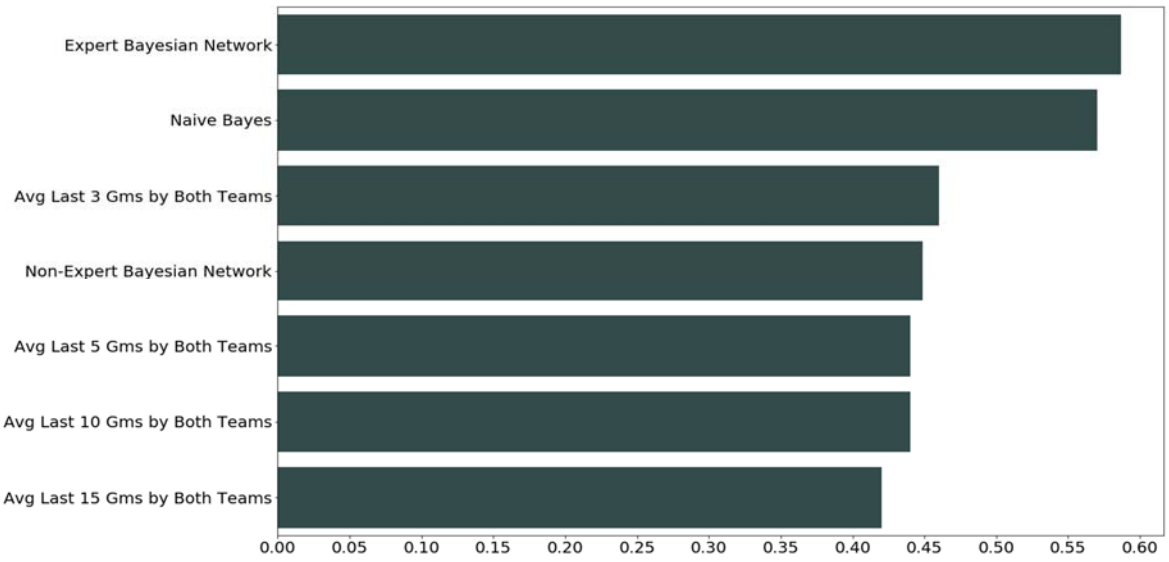

Figure 8: Overall Accuracy of Over/Under Prediction (%)

As can be seen in **Figure 8**, the Expert BN provides, only marginally (compared to the Naïve Bayes classifier), the best betting strategy when it comes to predicting Over/Under. We can

observe in **Figure 8** that the Expert BN and Naïve Bayes classifier yield promising results. But, perhaps more importantly, the Non-Expert BN's accuracy is worse than one of the amateur betting strategies. This is, presumably, because the Non-Expert BN did not have any parents of **TOTAL POINTS** identified. Thus, it did not provide any useful information on **TOTAL POINTS** beyond its empirical distribution based on the training set. Moreover, for the sum of average points scored by each team during the last n games (or amateur betting strategies), it appears that the fewer games we average over, the more accurate is the prediction.

6.1.2 Profitability Results

The odds for the Over/Under corresponding to the value for total points scored by both teams set by the oddsmakers were collected at the end of each of the first three quarters for each game in the test set. For this thesis, we assumed that the bettor either wagered \$100, if a prediction was obtained from the BNs, on the more favorable outcome (Over or Under) or wagered \$0, if an "N/A" value was obtained for the probability that the total points scored by both teams was greater than the value set by the oddsmakers. **Table 9** depicts all the values observed for the odds (whether it be Over or Under) with their implied probability percentage and profit amount if the bettor were to wager \$100 for the corresponding odds and win. If the bettor wagered on a bet and lost, their net loss would be -\$100.

Table 9: Description of Odds When Wagering \$100

Oddsmaker's Odds	Implied Probability Percentage	Wager Profit Amount
-105	51.22%	+ \$95.24
-110	52.38%	+ \$90.91
-115	53.49%	+ \$86.96
-120	54.55%	+ \$83.33
-125	55.56%	+ \$80.00

As evident in **Table 9**, all of the values for the Over/Under odds collected make it impossible to obtain a profit if our model is less than 50% accurate as we would wager more money than what we would be paid if we won a bet. Because of this, we restrict our analysis of quarterly predictions to the Expert BN and Naïve Bayes model as they were the only potentially profitable models. **Table 10** shows the monetary results for each of the three quarters as well as the overall profit amount of the Expert Bayesian Network. Again, we decided to omit those instances where the probability value returned was "N/A" by assuming the bettor would not wager any money on a game if they did not obtain a probability value.

Table 10: Total Profit Amount Overall and per Quarter (Expert BN)

	Number of Wagers Placed	Total Amount Wagered	Total Payout	Amount Won	Profit%
First Quarter	99	$9900	$10471.36	$571.36	5.77%
Second Quarter	100	$10000	$11807.41	$1807.41	18.07%
Third Quarter	97	$9700	$10291.93	$591.93	6.10%
Overall	**296**	**$29600**	**$32570.70**	**$2970.70**	**10.04%**

As seen in **Table 10**, our Expert Bayesian Network is quite profitable as it yields a 10.04% profit margin for 296 predicted instances. For comparison, the average savings account has a measly 0.09% annual percentage yield (Moon, 2019). It appears that betting after the second quarter, in particular, is significantly more profitable (and presumably more accurate) than the first and third quarters. Although further sensitivity analysis is needed, this may be because our model is not as sensitive to new information about the game when compared to the models used by the oddsmakers to set the value of total points scored by both teams. Although more information is indeed obtained after the third quarter, the uncertainty is not as great, and the value set by the oddsmakers at the end of that quarter is probably more accurate. It may also just be a coincidence given the small number of games observed. Nevertheless, our model is profitable in all three

quarters which is an indication that using our Expert Bayesian Network to make live-betting decisions is a viable option.

Table 11 shows the monetary results for each of the three quarters as well as the overall profit amount of the Naïve Bayes classifier model. As with the BN, we decided to omit those instances where the probability value returned was "N/A" by assuming the bettor would not wager any money on a game if they did not obtain a probability value.

Table 11: Total Profit Amount Overall and per Quarter (Naïve Bayes Classifier)

	Number of Wagers Placed	Total Amount Wagered	Total Payout	Amount Won	Profit%
First Quarter	100	$10000	$10101.07	$101.07	1.01%
Second Quarter	100	$10000	$11051.67	$1051.67	10.52%
Third Quarter	98	$9800	$10669.18	$869.18	8.87%
Overall	**298**	**$29800**	**$31821.92**	**$2021.92**	**6.78%**

As seen in **Table 11**, the Naïve Bayes model is quite profitable as it yields a 6.78% profit margin for 298 predicted instances. Just like our Expert BN, it appears that betting after the second quarter, in particular, is more profitable than betting after the first and third quarters. Although promising, our Naïve Bayes model is outperformed by the Expert BN, as its overall profit margin (and accuracy) is less even though the model returns a probability value in more instances. This is not surprising, as the BN can model dependencies between attributes while the Naïve Bayes classifier assumes all attributes are independent of each other. However, even with this handicap, the Naïve Bayes performs only slightly worse than the Expert BN. This may be an indication that the final attributes constructed provide enough information to predict the total points scored by both teams without having to consider the interactions with other attributes. Nevertheless, because the Expert BN dominates the Naïve Bayes model in both accuracy and profitability measures, we restrict our other analyses to only the Expert BN.

6.1.3 Time Results

Because the whole purpose of this thesis is to provide a tool to aid in making wagering decisions in real-time, the time it takes for the Expert BN to compute the joint probability distributions and use these to estimate the probability that the total points scored by both teams is greater than that value set by the oddsmakers is critical. Because of this, we analyzed the time it took for the Bayesian Network to predict an outcome (Over/Under) at the end of each of the first three quarters for a given game. **Table 12** shows the mean, standard deviation, and maximum time it took the BN to predict the probability that the total points scored by both teams is greater than that value set by the oddsmakers over all 300 instances.

Table 12: Summary Statistics of Prediction Time Over All 300 Instances

Mean	Standard Deviation	Maximum
2.14 seconds	0.55 seconds	3.32 seconds

As evident by **Table 12**, it took the Expert BN only a total of 642.32 (2.14 multiplied by 300 instances) seconds to estimate the probability that the total points scored by both teams was greater than the value set by the oddsmakers for all 300 instances. On average, this equates to 2.14 seconds per instance. Moreover, the most amount of time the BN took to provide a probability estimate was 3.32 seconds. The average and maximum values, coupled with the small standard deviation value of 0.55 seconds, ensures that the bettor using this tool has plenty of time between the end of each quarter and the subsequent quarter (which is around two minutes) to collect the in-game statistics and input them into the model.

6.2 Discussion of Results

To draw further conclusions from our model, a more in-depth analysis is needed. Specifically, we must understand how the proportions of Over vs. Under outcomes differ for each quarter between our model's predictions and the value set by the oddsmakers. **Table 13** depicts the confusion matrices by quarter for the Expert Bayesian Network constructed in the study.

Table 13: Expert Bayesian Network's Confusion Matrices by Quarters

		True Outcome		% Predicted Correctly
First Quarter	**Predicted Outcome**	Over	Under	**Overall: 56.57%**
	Over	14	4	77.78%
	Under	39	42	51.85%
	Actual Game Outcomes	54	46	**Prediction on 99 Samples**
Second Quarter	**Predicted Outcome**	Over	Under	**Overall: 63.00%**
	Over	18	5	78.26%
	Under	32	45	58.44%
	Actual Game Outcomes	50	50	**Prediction on all 100 Samples**
Third Quarter	**Predicted Outcome**	Over	Under	**Overall: 56.70%**
	Over	14	7	66.67%
	Under	35	41	53.95%
	Actual Game Outcomes	51	49	**Prediction on 97 Samples**

As illustrated in **Table 13**, it appears that our model predicts Under more frequently at the end of each quarter than Over. Moreover, it is worth noting that, overall, the oddsmakers are performing their job splendidly, as the outcome proportions when using their estimated value, 51.67% Over and 48.33% Under, is close to their goal of a 50% even split for each outcome. Although a larger sample would be needed to make a statistically significant conclusion, one can infer that the reason there is a slight discrepancy in proportions in the first quarter (54% Over and 46% Under) is due to a lack of information, as not enough time has elapsed in the game at that point. Although our model was consistent in predicting the game outcomes at the end of the first and third quarters, it was significantly more accurate (as presumed in Section 6.1.1) when placing

wagers at the end of the second quarter. Again, this may be due to the oddsmakers' models used to set the value for total points being more sensitive to the in-game statistics than our model, which also could explain their inconsistent outcome proportions at the end of the first quarter. Although one may be tempted to conclude that our model is drastically overfitting, it is important to analyze the training and test data sets further to understand the reason we are predicting Under more frequently than Over. The distributions of **TOTAL POINTS** in the training and test data sets are depicted in **Figure 9**.

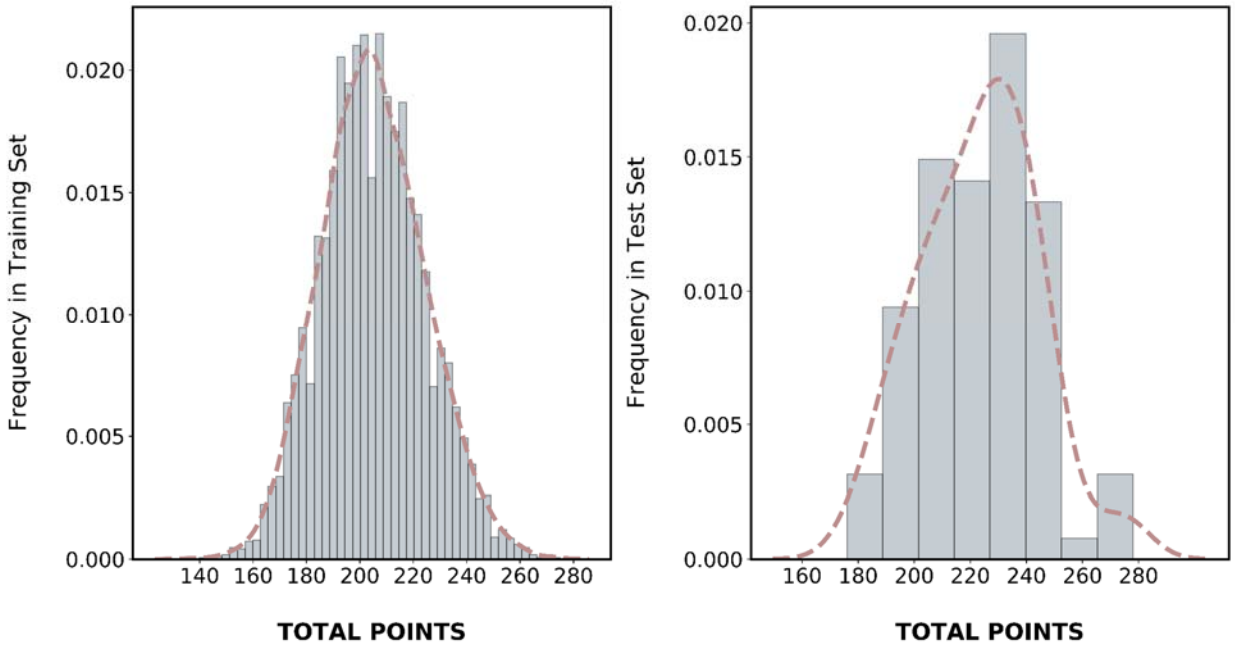

Figure 9: Distribution of TOTAL POINTS in the Training (left) and Test (right) Data Sets

As seen in **Figure 9**, the distributions of the total points scored by both teams for the training and test data sets differ greatly from one another. Specifically, the training data set's distribution is centered around a mean of 205.29 total points while the test set's distribution is centered around a mean of 223.14 total points. Although one might attribute this to the test set only being comprised of 100 games, it is important to note that the standard deviation for both set's

distribution was roughly the same (19.33 for training and 20.91 for test). This means that we cannot attribute the difference of the distributions merely to the difference in the number of games between both sets. Additionally, when observing our results in **Table 13**, we can see that our Expert Bayesian Network is significantly more accurate when it predicts Over (74.19%) than when it predicts Under (54.70%). Therefore, we are not necessarily overfitting. For comparison, the Naïve Bayes model correctly predicts Over 67.09 % of the time and Under 53.42% of the time.

As aforementioned, in recent years, there has been an increased volume in the pace of the game. The 2018-2019 NBA regular season, in particular, has been historically unprecedented as of the writing of this thesis. Not only are the pace per game, number of three-point shots attempted (3PA) per game by individual teams, and points per game scored (PPG) by individual teams at an all-time high, but the average increase from the previous year is significantly higher than it was in the past. **Figure 10** shows these statistics' increase and peak over the last six regular seasons (recall that games played during the 2013-2014 through 2017-2018 season compose the training set, and games played during the 2018-2019 season compose the test set).

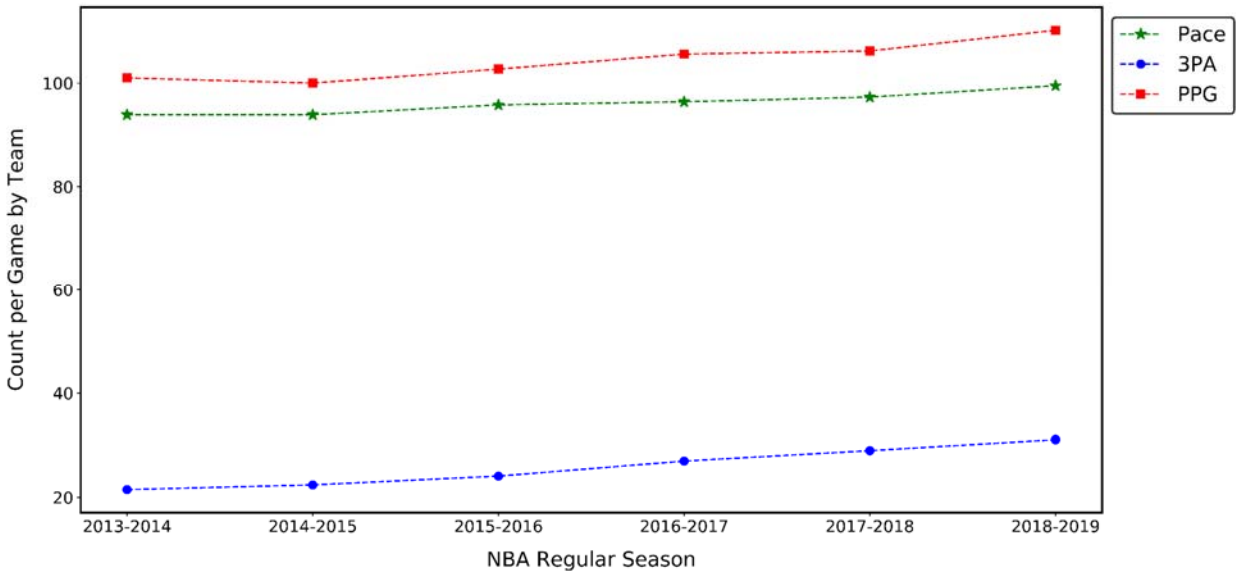

Figure 10: Offense-Driven Statistics' Average over the Last Six Regular Seasons

From our analysis, we can conclude that the reason our model predicts Under much more frequently than Over is because our training set is not an adequate set to learn from that will generalize to the unseen (by the BN) instances in the test set. This also explains why we only obtained predictions for 296 out of the 300 instances on which we tested our model. Although we assumed the bettor would not wager on these games, the difference between the training and test sets may mean that the values of the quarterly in-game statistics of the games in the test set were not observed in the corresponding quarter of the games in our training set. Thus, the model assigned a zero probability for $\hat{\phi}_G$ and $\hat{\phi}_L$ (detailed in Section 5.5). Then, when we normalize these values to obtain the probability that the score is greater than that value set by the oddsmakers, a "N/A" resulted. Nevertheless, we are profitable in all quarters, and our model exhibited a strong overall accuracy performance (58.78%). Without ignoring the limitations of our model, we deem our model successful.

CHAPTER VII

CONCLUSIONS

Although many forecasting models have been built to predict the total number of points scored in NBA games, these models' predictions are primarily based on data from previously completed games. They fail to provide bettors with predictions also based on the current game being played. In turn, this lack hinders their practicality when it comes to sports betting as, in basketball, one is allowed to bet at specific time points in the game such as the end of quarters. We presented a new classification approach to estimate and sequentially update the probability that the total points scored by both teams is greater than the total number of points set by the oddsmakers (inspired by the Totals betting strategy) given a set of in-game statistics as input variables. We estimated this probability to make wagering decisions at the end of each of the first three quarters by calculating the joint probability distribution of scoring totals. This was achieved through a Bayesian Network constructed by using filter-based feature selection, conditional independence tests, and domain knowledge.

The Bayesian Networks and Naïve Bayes models built in the study were trained on every non-overtime game in the last five regular seasons of the NBA (2013-2014 through 2017-2018) and evaluated on 100 early non-overtime games of the 2018-2019 NBA regular season. Specifically, the models were evaluated through collecting the same in-game statistics as in the training set, the value set by the oddsmakers for total points at the end of each of the first three quarters, and the Over/Under odds corresponding to these values. The Expert Bayesian Network and, to a lesser extent, the Naïve Bayes model demonstrated an ability to beat the oddsmakers, generating a profit of over 10% and over 6%, respectively, by providing estimates for use in making wagering decisions in real-time despite an unprecedentedly high-scoring 2019 NBA

regular season. Specifically, our Expert BN's average (2.14 seconds), maximum (3.32 seconds), and standard deviation (0.55 seconds) statistics for computation time over all 300 instances ensured that the bettor using this tool has plenty of time between quarters to collect the in-game statistics, as well as the value set by the oddsmakers, and input them into our model. We showed, in line with previous work, that a BN's structure works best when domain knowledge is incorporated, as the Expert BN had a higher overall accuracy (58.78%) than the BN (44.8%) that learned the structure from the training data set. Moreover, we showed that the amateur betting strategies did not generate a profit when applied to the values set by the oddsmakers at the end of each quarter in our test set. Lastly, we showed that, even with all the dependencies present between the in-game statistics, the Naïve Bayes model was successful but was dominated by the Expert BN. Although we deemed the Expert BN successful for its high profitability and strong overall accuracy performance, the model is not without its flaws.

The training data set used for our model was inadequate as we did not foresee the increase in point production of the new NBA season. Although we tried to mitigate the risk by considering only the last five NBA regular seasons, it might have been wise to use part of our testing set or to collect more data from the 2018-2019 NBA regular season and use it as a validation set to tune our DAG. Another limitation of our model is that it does not incorporate historical odds, as the oddsmakers do not archive them, to make its predictions. If we can somehow find a correlation between the in-game statistics and the change in the value set by the oddsmakers for total points scored by both teams at the end of each of the first three quarters, our model can make better-informed decisions. Moreover, although it is common in the literature to not consider overtime games for these types of problems, it is important to note that when the bettor is wagering on a game in real-time, they do not know whether the game will go to overtime. Therefore, it is likely

(given how much more frequently our Expert BN predicted Under than Over) that our estimates for total overall accuracy were biased in our favor (though not by much, as when manually collecting data only 5 out of 105 games were discarded due to overtime). Nonetheless, all these limitations provide us with great opportunities for future research.

Future research aims to improve upon the limitations of our Expert Bayesian Network mentioned above. One idea is to construct an ensemble model that incorporates the Expert BN along with other probabilistic models (such as logistic regression and the random forest) and obtains a weighted average of these probabilities (preferably the other models will be biased towards predicting Over) to influence our wagering decisions. Additionally, we might want to consider, as previously mentioned, how the value set by the oddsmakers will shift in relation to the change of the in-game statistics at the end of each quarter by collecting more data and estimating conditional probability distributions for these shifts. Currently, our model is dynamically updating as it possesses information on the total minutes played in the game and the accumulation of the in-game statistics but cannot model the same game progressively (that is, it does not store in-game statistics of previous quarters to make its predictions). This is a crude way of making the BN time-dependent and, therefore, it may be worthwhile to build a multi-stage BN that consists of three stages (one for the end of each quarter) and feeds information from one stage to another for our model to learn game progression. Lastly, one of the future research ideas lies within the realm of decision analysis. In our model, we assumed that the bettor would wager either \$100 or \$0 on the more favorable outcome, but this may not be the best way to go about making wagering decisions. In the future, we might explore obtaining steady-state conditional probabilities for the result of the game given the outcome predicted at the end of each quarter by the Expert Bayesian Network to build a multi-stage decision model that maximizes a bettor's utility function

obtained from an analysis of their risk aversion. Instead of a static decision to wager a fixed amount each quarter, this model could allocate a fixed budget per game to wager at the end of each of its first three quarters. A conference paper summarizes the methodology described in this thesis along with some of the results (Alameda-Basora & Ryan, 2019).

BIBLIOGRAPHY

Advanced Box Scores. (n.d.). In *NBA Advanced Stats*. Retrieved November 7, 2018, from https://stats.nba.com/teams/boxscores-advanced/

Advanced Box Scores. (n.d.). In *NBA Advanced Stats*. Retrieved November 7, 2018, from https://stats.nba.com/teams/boxscores-four-factors/

Advanced Box Scores. (n.d.). In *NBA Advanced Stats*. Retrieved November 7, 2018, from https://stats.nba.com/teams/boxscores-misc/

Advanced Box Scores. (n.d.). In *NBA Advanced Stats*. Retrieved November 7, 2018, from https://stats.nba.com/teams/boxscores-scoring/

Advanced Box Scores. (n.d.). In *NBA Advanced Stats*. Retrieved November 7, 2018, from https://stats.nba.com/teams/boxscores-traditional/

Aggarwal, C. C. (2015). *Data Mining: The Textbook*. *Springer International Publishing* (pp. 746). https://doi.org/10.1007/978-3-319-14142-8

Alameda-Basora, E., & Ryan, S. (2019). *Application of Bayesian Network to Total Points in NBA Games*. Proceeds of the 2019 IISE Annual Conference, Orlando, FL.

Alpaydin, E. (2014). *Introduction to machine learning*. *Methods in Molecular Biology* (Vol. 1107, pp. 105–128). https://doi.org/10.1007/978-1-62703-748-8-7

Basketball. Bovada. n.d. Retrieved October 16th, 2019 through October 31st, 2019 from www.bovada.lv/sports/basketball

Bertsekas, D. (1998). *Network Optimization: Continuous and Discrete Models*. *Methods* (Vol. C, pp. 51-114). https://doi.org/10.1.1.86.448

Bovada Review (October 19, 2018). In *Sports Betting Dime*. Retrieved from https://www.sportsbettingdime.com/sportsbooks/bovada/

Chernoff, H. & Lehmann, E. L. (1954). The use of maximum likelihood estimates in χ^2 tests for goodness of fit. *The annals of mathematical statistics*, *25*(3), 579-586.

Constantinou, A. C., Fenton, N. E., & Neil, M. (2013). Profiting from an inefficient association football gambling market: Prediction, risk and uncertainty using Bayesian networks. *Knowledge-Based Systems*, *50*, 60–86. https://doi.org/10.1016/j.knosys.2013.05.008

Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, *20*(2), 215–242. https://doi.org/10.1007/BF03180993

Cramér, Harald. 1946. *Mathematical Methods of Statistics*. Princeton: Princeton University Press, page 282 (Chapter 21. The two-dimensional case). ISBN 0-691-08004-6

Daly R, Shen Q (2007). "Methods to Accelerate the Learning of Bayesian Network Structures." In Proceedings of the 2007 UK Workshop on Computational Intelligence. Imperial College, London.

Darwiche, A. (2010). Bayesian networks. *Communications of the ACM*, *53*(12), 80. https://doi.org/10.1145/1859204.1859227

De Saá Guerra, Y., Martín Gonzalez, J. M., Sarmiento Montesdeoca, S., Rodriguez Ruiz, D., Arjonilla López, N., & García-Manso, J. M. (2013). Basketball scoring in NBA games: An example of complexity. *Journal of Systems Science and Complexity*, *26*(1), 94–103. https://doi.org/10.1007/s11424-013-2282-3

Dhar, V. (2013). Data science and prediction. *Communications of the ACM*, *56*(12), 64–73. https://doi.org/10.1145/2500499

Drugan, M. M., & Wiering, M. A. (2010). Feature selection for Bayesian network classifiers using the MDL-FS score. *International Journal of Approximate Reasoning, 51*(6), 695-717. https://doi.org/10.1016/j.ijar.2010.02.001

Faltin, F., & Kenett, R. (2007). Bayesian Networks. *Encyclopedia of Statistics in Quality & Reliability*, *1*(1), 4. https://doi.org/10.1002/wics.48

Ganguly, S., & Frank, N. (2018). The Problem with Win Probability. *MIT Sloan Sports Analytics Conference*. http://www.sloansportsconference.com/wp-content/uploads/2018/02/2011.pdf

Gerrard, B. (2014). Sports Analytics: A Guide for Coaches, Managers and Other Decision Makers, B.C. Alamar. Columbia University Press, New York (2013). *Sport Management Review*, *17*(2), 240–241. https://doi.org/10.1016/j.smr.2013.06.005

Gortmaker, S. L., Hosmer, D. W., & Lemeshow, S. (1994). Applied Logistic Regression. *Contemporary Sociology*, *23*(1), 159. https://doi.org/10.2307/2074954

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157-1182.

Haghighat, M., Rastegari, H., & Nourafza, N. (2013). A Review of Data Mining Techniques for Result Prediction in Sports. *Advances in Computer Science*, *2*(5), 7–12.

How to Bet on Basketball. (n.d.). In *bettingexpert.* Retrieved March 14th, 2018, from https://www.bettingexpert.com/academy/betting-on-different-sports/basketball

Huggins, J., Hammant P., Bevan, J., & Stewart, S. Selenium WebDriver, Google Test Automation Conference (2009).

Inza, I., Larrañaga, P., Etxeberria, R., & Sierra, B. (2000). Feature Subset Selection by Bayesian network-based optimization. *Artificial Intelligence*, *123*(1–2), 157–184. https://doi.org/10.1016/S0004-3702(00)00052-7

Jensen, F. V. (2009, November). Bayesian networks. *Wiley Interdisciplinary Reviews: Computational Statistics*. https://doi.org/10.1002/wics.48

John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

Joseph, A., Fenton, N. E., & Neil, M. (2006). Predicting football results using Bayesian nets and other machine learning techniques. *Knowledge-Based Systems*, *19*(7), 544–553. https://doi.org/10.1016/j.knosys.2006.04.011

Karegowda, A. G., Manjunath, A. S., & Jayaram, M. A. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, *2*(2), 271-277.

Kennedy, M. P., & Chua, L. O. (1988). Neural Networks for Nonlinear Programming. *IEEE Transactions on Circuits and Systems*, *35*(5), 554–562. https://doi.org/10.1109/31.1783

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, *31*, 249–268. https://doi.org/10.1115/1.1559160

Kvam, P., & Sokol, J. S. (2006). A logistic regression/Markov chain model for NCAA basketball. *Naval Research Logistics*, *53*(8), 788–803. https://doi.org/10.1002/nav.20170

L. A. Kurgan and K. J. Cios, "CAIM discretization algorithm," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 2, pp. 145-153, Feb. 2004. doi: 10.1109/TKDE.2004.1269594

Leagues. Basketball Reference. n.d. Retrieved March 14th, 2019 from basketball-reference.com

Loeffelholz, B., Bednar, E., & Bauer, K. W. (2009). Predicting NBA Games Using Neural Networks. *Journal of Quantitative Analysis in Sports*, *5*(1), 1-17. https://doi.org/10.2202/1559-0410.1156

Lopez, M. J., & Matthews, G. J. (2015). Building an NCAA men's basketball predictive model and quantifying its success. *Journal of Quantitative Analysis in Sports*, *11*(1), 5–12. https://doi.org/10.1515/jqas-2014-0058

Lopez, Michael & Matthews, Gregory & Baumer, Ben. (2017). How often does the best team win? A unified approach to understanding randomness in North American sport.

Mao, J. (1996). Why artificial neural networks? *Communications*, *29*, 31–44. https://doi.org/10.1109/2.485891

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22. URL http://www.jstatsoft.org/v35/i03/.

Marco Scutari, Jean-Baptiste Denis. (2014) *Bayesian Networks with Examples in R*. Chapman and Hall, Boca Raton. ISBN 978-1-4822-2558-7.

Mikut, R., & Reischl, M. (2011). Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(5), 431–443. https://doi.org/10.1002/widm.24

Miljković, D., Gajić, L., Kovačević, A., & Konjović, Z. (2010). The use of data mining for basketball matches outcomes prediction. In *SIISY 2010 - 8th IEEE International Symposium on Intelligent Systems and Informatics* (pp. 309–312). https://doi.org/10.1109/SISY.2010.5647440

Min, B., Kim, J., Choe, C., Eom, H., & (Bob) McKay, R. I. (2008). A compound framework for sports results prediction: A football case study. *Knowledge-Based Systems*, *21*(7), 551–562. https://doi.org/10.1016/j.knosys.2008.03.016

Moon, Chris (2019). *Average Bank Interest Rates in 2019: Checking, Savings, Money Market, and CD Rates*. Retrieved March 14th , 2019, from https://www.valuepenguin.com/banking/average-bank-interest-rates

Norris, C. M., Ghali, W. A., Saunders, L. D., Brant, R., Galbraith, D., Faris, P., & Knudtson, M. L. (2006). Ordinal regression model and the linear regression model were superior to the logistic regression models. *Journal of Clinical Epidemiology*, *59*(5), 448–456. https://doi.org/10.1016/j.jclinepi.2005.09.007

Python Software Foundation. Python Language Reference, version 3.6.6. Available at http://www.python.org

Quinlan, J. Ross. "Induction of decision trees." Machine learning 1.1 (1986): 81:106.

Raschka, S. (2014). Naive Bayes and Text Classification I - Introduction and Theory. *ArXiv Preprint ArXiv: 1410.5329*, 20. Retrieved from http://arxiv.org/abs/1410.5329

Sarkar, S. D., & Goswami, S. (2013). Empirical study on filter based feature selection methods for text classification. *International Journal of Computer Applications*, *81*(6), 38-43.

Søren Højsgaard (2012). Graphical Independence Networks with the gRain Package for R. Journal of Statistical Software, 46(10), 1-26. URL http://www.jstatsoft.org/v46/i10/.

Štrumbelj, E., & Vračar, P. (2012). Simulating a basketball match with a homogeneous Markov model and forecasting the outcome. *International Journal of Forecasting*, *28*(2), 532–542. https://doi.org/10.1016/j.ijforecast.2011.01.004

Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, *49*(11), 1225–1231. https://doi.org/10.1016/S0895-4356(96)00002-9

Schwarz, G. (1978). Estimating the dimension of a model. Annals of Statistics 6(2), pp. 461-464.

Stern, H. S. (2008). Point spread and Odds Betting: Baseball, Basketball, and American Football. In *Handbook of sports and lottery markets* (pp. 223-237). Elsevier.

Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Williams, J. (2010). Momentum and Sports Betting. *Available at SSRN 1553150*.

Wright, M. B. (2009). 50 years of OR in sport. *Journal of the Operational Research Society*, *60*(SUPPL. 1). https://doi.org/10.1057/jors.2008.170

Zimmermann, A. (2016). Basketball predictions in the NCAAB and NBA: Similarities and differences. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *9*(5), 350-364.

APPENDIX A: Interpretations of In-Game Statistics

Table A1: Interpretation of In-Game Statistics (from: https://stats.nba.com/help/glossary)

MINUTES PLAYED: Total minutes played in the game by the respective team

2PT FGA %: Percentage of shots taken by the respective team that are two-point field goals

3PT FGA %: Percentage of shots taken by the respective team that are three-point field goals

2PT PTS %: Percentage of points scored by the respective team that are from two-point field goals

2PT MR PTS %: Percentage of points scored by the respective team from shots made between the paint and three-point areas

3PT PTS %: Percentage of points scored by the respective team that are from three-point field goals

FBPS PTS %: Percentage of points scored by the respective team that are scored while on fast break

FT PTS %: Percentage of points scored by the respective team that are from free throws

PTS OFF TO %: Percentage of points scored by the respective team that are from the opponent's turnovers

PITP %: Percentage of points scored by the respective team that are from the paint area

2FGM % AST: Percentage of made two-point field goals by the respective team that are assisted

2FGM % UAST: Percentage of made two-point field goals by the respective team that are not assisted

3FGM % AST: Percentage of made three-point field goals by the respective team that are assisted

3FGM % UAST: Percentage of made three-point field goals by the respective team that are not assisted

FGM % AST: Percentage of made field goals by the respective team that are assisted

FGM % UAST: Percentage of made field goals by the respective team that are not assisted

PTS OFF TO: Amount of points scored by the respective team that are from the opponent's turnovers

2ND CHANCE PTS: Amount of points scored by the respective team that are scored after an offensive rebound

PITP: Amount of points scored by the respective team that are from the paint area

FBPS: Amount of points scored by the respective team that are scored while on fast break

PTS: Amount of points scored by the respective team

FGM: Number of shots made by the respective team

FGA: Number of shots attempted by the respective team

FG %: Percentage of shots made by the respective team

3PM: Number of three-point shots made by the respective team

3PA: Number of three-point shots attempted by the respective team

3P%: Percentage of three-point shots made by the respective team

FTM: Number of free throws made by the respective team

FTA: Number of free throws attempted by the respective team

FT%: Percentage of free throws made by the respective team

Table A1: (continued)

OREB: The number of rebounds collected by the respective team while they were on offense

DREB: The number of rebounds collected by the respective team while they were on defense

REB: The number of rebounds collected by the respective team

AST: The number of passes by the respective team that lead directly to a made basket

TO: The number of times the respective team loses the ball to the defense

BLK: The number of times the respective team tips the ball from the opponent's shot, negating their chance to score

STL: The number of times the respective team takes the ball from the opponent on offense, causing them a turnover

PF: The number of fouls committed by the respective team

+/-: The point differential of the respective team

OFFRTG: The total points of the respective team per 100 possessions

DEFRTG: The total points of the opposing team per 100 possessions

NETRTG: The respective team's point differential per 100 possessions

AST%: The percentage of shots the respective team assisted on

AST/TO: The respective team's total assist divided by their total turnovers

AST RATIO: The respective team's total assist per 100 possessions

OREB%: The percentage of rebounds by the respective team that are offensive rebounds

DREB%: The percentage of rebounds by the respective team that are defensive rebounds

REB%: The percentage of total rebounds in the game by the respective team

TO%: The percentage of plays by a respective team that end in a turnover

eFG%: The respective team's shooting efficiency; accounts for the worth of making three-point field goals

TS%: Same as **eFG%** but also accounts for the worth of made free throws

PACE: Average number of possessions for both teams scaled to 48 minutes

PIE: Measures the respective team's overall statistical contribution against the total statistics in the game

FTRATE: The number of free throw attempts of a respective team divided by the field goal attempts of that team

TPQ: The total points scored by both teams in the quarter

TPPQ: The total points scored by both teams in the previous quarter

TP2QsP: The total points scored by both teams two quarters previously

AWAY TEAM: The team that is not playing in their city for that game

HOME TEAM: The team that is playing in their city for that game

PSUTP: The total points scored by both teams up to that point in the game

TOTAL POINTS: The class attribute; total points scored by both teams at the end of the game

APPENDIX B: R Program to Estimate Probabilities for All Game Instances (BNs)

Final_Thesis_Work.R

```r
#This code was written by Enrique M. Alameda-Basora at 10:20 am on 12/20/2018

#Loading the necessary packages

library(data.table)
library(dplyr)
library(lubridate)
library(bnlearn)
library(gRbase)
library(gRain)

#Setting work directory
setwd("C:/Users/Enriq/Downloads")



#Loading the Discretized Data sets with Feature Selection Already Performed
Training_Data<-fread("Discretized_Final_Training_Set.csv", stringsAsFactors = T)

Test_Data<-fread("Discretized_Final_Test_Set.csv", stringsAsFactors = T)

VSBO<-fread("VSBO.csv") #csv file containing the 300 values set by the oddsmakers



#Converting the problem into a classification problem
Test_Data[,c(1:17)]<-lapply(Test_Data[,c(1:17)], factor)  ## as.factor() could also be used
Training_Data[,c(1:17)]<-lapply(Training_Data[,c(1:17)], factor)  ## as.factor() could also be used
```

```r

#Changing Troublesome Network Names
names(Training_Data)[names(Training_Data) == "TIME INDICATOR"] = "MINUTES_PLAYED"
names(Test_Data)[names(Test_Data) == "TIME INDICATOR"] = "MINUTES_PLAYED"
names(Training_Data)[names(Training_Data) == "eFG%_OPP"] = "effective_opponent"
names(Test_Data)[names(Test_Data) == "eFG%_OPP"] = "effective_opponent"
names(Training_Data)[names(Training_Data) == "eFG%"] = "effective"
names(Test_Data)[names(Test_Data) == "eFG%"] = "effective"
names(Training_Data)[names(Training_Data) == "3PM"] = "Three_Point"
names(Test_Data)[names(Test_Data) == "3PM"] = "Three_Point"
names(Training_Data)[names(Training_Data) == "3PM_OPP"] = "Three_Point_Opponent"
names(Test_Data)[names(Test_Data) == "3PM_OPP"] = "Three_Point_Opponent"
names(Training_Data)[names(Training_Data) == "TS%"] = "True_shooting"
names(Training_Data)[names(Training_Data) == "TS%_OPP"] = "True_shooting_opponent"
names(Test_Data)[names(Test_Data) == "TS%"] = "True_shooting"
names(Test_Data)[names(Test_Data) == "TS%_OPP"] = "True_shooting_opponent"
names(Training_Data)[names(Training_Data) == "FINAL_SCORE"] = "TOTAL_POINTS"
names(Test_Data)[names(Test_Data) == "FINAL_SCORE"] = "TOTAL_POINTS"

#Making a Non_Expert BN and calculating its score for further verification
DAG_NE<-hc(Training_Data)
Non_Feature_Selection_HC_Score<- score(DAG_NE,data=Training_Data) #-406,404.1
```

```r

#Running conditional independence tests on each directed arc
arc.strength(DAG_NE,data=Training_Data, criterion="x2")


#Lets compute accuracy of Non-expert BN

Non_Expert_Fitted<-bn.fit(DAG_NE,Training_Data)

Non_Expert_fitted.grain<- as.grain(Non_Expert_Fitted) #fitting it in grain package to calculate
                                                      #exact conditional probabilities

#Pre-allocating vectors
Vector_NE<-{}
Full_Vector_NE<-{}
Full_Vector_1_NE<-{}
Full_Vector_2_NE<-{}
P_Greater_VSBO_NE<-{}
P_less_VSBO_NE<-{}


#Give me all the values for TOTAL POINTS
values_of_TOTAL_POINTS<-as.numeric(as.vector(unique(Training_Data$TOTAL_POINTS)))
```

```
for (j in c(1:dim(Test_Data)[1]))

  for (i in values_of_TOTAL_POINTS)
  {
    {
    #Finding the joint probability distribution using the test sets values
    bnet.ev<-setEvidence(Non_Expert_fitted.grain, nodes = c("PSUTP","TOTAL_POINTS","TP_SCORED_IN_PREV_2_QTRS",
                                                            "TP_SCORED_IN_PREVIOUS_QTR","TP_SCORED_IN_QTR","MINUTES_PLAYED",
                                                            "FGM","FGM_OPP","PACE","effective_opponent","effective",
                                                            "OFFRTG","DEFRTG","Three_Point","Three_Point_Opponent","True_shooting","True_shooting_opponent"),
                         states = c(as.character(Test_Data$PSUTP[j]),i,as.character(Test_Data$TP_SCORED_IN_PREV_2_QTRS[j]),
                                    as.character(Test_Data$TP_SCORED_IN_PREVIOUS_QTR[j]),as.character(Test_Data$TP_SCORED_IN_QTR[j]),
                                    as.character(Test_Data$MINUTES_PLAYED[j]),as.character(Test_Data$FGM[j]),as.character(Test_Data$FGM_OPP[j]),
                                    as.character(Test_Data$PACE[j]),as.character(Test_Data$PACE_OPP[j]),as.character(Test_Data$effective_opponent[j]),
                                    as.character(Test_Data$effective[j]),as.character(Test_Data$OFFRTG[j]),
                                    as.character(Test_Data$DEFRTG[j]),as.character(Test_Data$Three_Point[j]),
                                    as.character(Test_Data$Three_Point_Opponent[j]),as.character(Test_Data$True_shooting[j]),
                                    as.character(Test_Data$True_shooting_opponent[j])))
      Vector_NE[i]<-pEvidence(bnet.ev)
      Full_Vector_NE<-as.data.frame(Vector_NE[values_of_TOTAL_POINTS])
      rownames(Full_Vector_NE)=values_of_TOTAL_POINTS
      colnames(Full_Vector_NE)='pEvidence'
      print(j)
    }

```

```
#Getting the probability, given the evidence, that is is greater than some value

      Full_Vector_1_NE<-sum(Full_Vector_NE[rownames(Full_Vector_NE)>VSBO$VSBO[j],])
      Full_Vector_2_NE<-sum(Full_Vector_NE[rownames(Full_Vector_NE)<=VSBO$VSBO[j],])

      P_Greater_VSBO_NE[j]<-Full_Vector_1_NE/(Full_Vector_1_NE+Full_Vector_2_NE)
      P_less_VSBO_NE[j]<-Full_Vector_2_NE/(Full_Vector_2_NE+Full_Vector_1_NE)
      Sys.time()
}

#Storing results in a csv file for analysis later on
Non_Expert_Results<-matrix(nrow=300,ncol=4)
Non_Expert_Results<-as.data.frame(Non_Expert_Results)
Non_Expert_Results$V1<-VSBO$VSBO
Non_Expert_Results$V2<-Test_Data$TOTAL_POINTS
Non_Expert_Results$V3<-P_Greater_VSBO_NE
Non_Expert_Results$V4<-P_less_VSBO_NE
colnames(Non_Expert_Results)<-c("VSBO","FS","Prob_G","Prob_L")
write.csv(Non_Expert_Results,"NExp_Results.csv",row.names=FALSE)

```

```
#Specifying Structure of Expert BN

DAG_E<-empty.graph(nodes=c("PSUTP","TOTAL_POINTS","TP_SCORED_IN_PREV_2_QTRS",
                           "TP_SCORED_IN_PREVIOUS_QTR","TP_SCORED_IN_QTR","MINUTES_PLAYED",
                           "FGM","FGM_OPP","PACE","effective_opponent","effective",
                           "OFFRTG","DEFRTG","Three_Point","Three_Point_Opponent",
                           "True_shooting","True_shooting_opponent"))



# Optimal Bayesian Network
DAG_E<-set.arc(DAG_E, from="PSUTP", to="TOTAL_POINTS", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="TOTAL_POINTS", to="TP_SCORED_IN_PREV_2_QTRS", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="TOTAL_POINTS", to="TP_SCORED_IN_PREVIOUS_QTR", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="TOTAL_POINTS", to="TP_SCORED_IN_QTR", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="effective_opponent", to="DEFRTG", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="Three_Point_Opponent", to="effective_opponent", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="effective_opponent", to="True_shooting_opponent", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="effective", to="OFFRTG", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="Three_Point", to="effective", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="effective", to="True_shooting", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="OFFRTG", to="FGM", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="DEFRTG", to="FGM_OPP", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="FGM_OPP", to="PSUTP", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="FGM", to="PSUTP", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)
DAG_E<-set.arc(DAG_E, from="MINUTES_PLAYED", to="TOTAL_POINTS", check.cycles = TRUE, check.illegal = TRUE, debug = FALSE)

```

```
#Calculating Expert BN's score for further verification
Feature_Selection_Expert_Score<- score(DAG_E,data=Training_Data) #-530,496.75

#Running conditional independence tests on each directed arc
arc.strength(DAG_E,data=Training_Data, criterion="x2")

#Running AIC on TOTAL_POINTS for both networks to compare

Feature_Selection_Expert_Score<- score(DAG_E,data=Training_Data, "aic", by.node=TRUE)
Feature_Selection_Expert_Score["TOTAL_POINTS"] #-74,998.75

Feature_Selection_Non_Expert_Score<- score(DAG_NE,data=Training_Data, "aic", by.node=TRUE)
Feature_Selection_Non_Expert_Score["TOTAL_POINTS"] #-75,907.90


#Lets compute accuracy of Expert BN

fitted = bn.fit(DAG_E, Training_Data)

Expert_fitted.grain<- as.grain(fitted)

P_Greater_VSBO_E<-vector(mode="numeric",length=300)
P_less_VSBO_E<-vector(mode="numeric",length=300)
Vector_E<-{}
Full_Vector_1_E<-{}
Full_Vector_2_E<-{}
Full_Vector_E<-{}
```

```
#Preallocate time vectors
Time_of_game<-{}
Start.time<-{}
Time<-{}

#Start the time outside the loop
Start.time=Sys.time()

for (j in c(1:dim(Test_Data)[1]))
  for (i in values_of_TOTAL_POINTS)
  {
   {
     bnet.ev<-setEvidence(Expert_fitted.grain, nodes = c("PSUTP","TOTAL_POINTS","TP_SCORED_IN_PREV_2_QTRS",
                                                         "TP_SCORED_IN_PREVIOUS_QTR","TP_SCORED_IN_QTR","MINUTES_PLAYED",
                                                         "FGM","FGM_OPP","PACE","effective_opponent","effective",
                                                         "OFFRTG","DEFRTG","Three_Point","Three_Point_Opponent",
                                                       "True_shooting","True_shooting_opponent"),
                          states = c(as.character(Test_Data$PSUTP[j]),i,as.character(Test_Data$TP_SCORED_IN_PREV_2_QTRS[j]),
                                     as.character(Test_Data$TP_SCORED_IN_PREVIOUS_QTR[j]),as.character(Test_Data$TP_SCORED_IN_QTR[j]),
                                     as.character(Test_Data$MINUTES_PLAYED[j]),as.character(Test_Data$FGM[j]),as.character(Test_Data$FGM_OPP[j]),
                                     as.character(Test_Data$PACE[j]),as.character(Test_Data$effective_opponent[j]),
                                     as.character(Test_Data$effective[j]),as.character(Test_Data$OFFRTG[j]),
                                     as.character(Test_Data$DEFRTG[j]), as.character(Test_Data$Three_Point[j]),
                                     as.character(Test_Data$Three_Point_Opponent[j]),
                                     as.character(Test_Data$True_shooting[j]),as.character(Test_Data$True_shooting_opponent[j])))
     Vector_E[i]<-pEvidence(bnet.ev)
     Full_Vector_E<-as.data.frame(Vector_E[values_of_TOTAL_POINTS])
     rownames(Full_Vector_E)=values_of_TOTAL_POINTS
```

```
       colnames(Full_Vector_E)='pEvidence'
     }

     #Getting the probability, given the evidence, that is is greater than some value

       Full_Vector_1_E<-sum(Full_Vector_E[rownames(Full_Vector_E)>VSBO$VSBO[j],])
       Full_Vector_2_E<-sum(Full_Vector_E[rownames(Full_Vector_E)<=VSBO$VSBO[j],])

       P_Greater_VSBO_E[j]<-Full_Vector_1_E/(Full_Vector_1_E+Full_Vector_2_E)
       P_less_VSBO_E[j]<-Full_Vector_2_E/(Full_Vector_2_E+Full_Vector_1_E)
       if(j==1){
         #Set the first element
         Time[j]=Sys.time()
         Initial_Time=Sys.time()-Start.time
         Time_of_game[j]=Initial_Time
       } else{
         Time[j]=Sys.time()
         Time_of_game[j]=Time[j]-Time[j-1]
       }

       print(j) #just to see how it progresses
   }
```

```
229
230 #Storing results in a csv file for analysis later on
231 Expert_Results<-matrix(nrow=300,ncol=4)
232 Expert_Results<-as.data.frame(Expert_Results)
233 Expert_Results$V1<-VSBO$VSBO
234 Expert_Results$V2<-Test_Data$TOTAL_POINTS
235 Expert_Results$V3<-P_Greater_VSBO_E
236 Expert_Results$V4<-P_less_VSBO_E
237 colnames(Expert_Results)<-c("VSBO","FS","Prob_G","Prob_L")
238 write.csv(Expert_Results,"ExpertBN_Results.csv",row.names=FALSE)
239 write.csv(Time_of_game,"Time_for_Games_Results.csv",row.names=FALSE)
```