NTNU – Trondheim
Norwegian University of
Science and Technology

The Betting Machine

Thomas Rene Andresen
Damian Dubicki

Master of Science in Computer Science
Submission date: June 2013
Supervisor: Helge Langseth, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Thomas R. Andresen and Damian Dubicki

The Betting Machine

Master Thesis, Spring 2013

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

The popularity of football has increased excessively since it originated, and in later years, systems to predict football results have become a popular area of research. Based on information found during a specialization project, the most common way of predicting future matches is based on making observations of previous match results. In this thesis we aim to develop and test a model that is able to utilize more of the available data than what has been previously attempted. In addition we also aim to develop a framework that is able to automate most of the involved processes - from data mining to prediction to betting.

Sammendrag

Fotball har hatt en stadig økning i populæritet siden spillet først oppstod og i senere år har det å prøve å forutse kampresultater blitt en populær forskningsgren. Basert på informasjon funnet i et fordypningsprosjekt, er den mest vanlige måten å spå fremtidige resultater basert på tidligere kampresultater. I denne masteroppgaven har vi hatt som mål å utvikle og teste en modell som er i stand til å utnytte mer tilgjengelig data enn det som tidligere har vært forsøkt. I tillegg har vi også som mål å utvikle et rammeverk som er i stand til å automatisere de involverte prosessene - fra datainnsamling til resultatspådom til å satse pengebeløp.

Preface

This paper is the master thesis of Thomas R. Andresen and Damian Dubicki and is the conclusion of our masters degree in computer science. We are two students from the Norwegian University of Science and Technology (NTNU), Department of Computer and Information science and the Artificial Intelligence group. In this thesis we have developed a model for predicting results of football matches in a closed league. We have also developed a framework that gathers data, simulate matches and suggests profitable bets. The framework supports publishing of predictions and other internal data to a web page.

We would like to thank Helge Langseth, our supervisor, for constructive feedback, discussions and the help he have provided during the period. In addition we would thank the AI-Masters course and all the people involved for helping us out with the more formal tasks.

Thomas R. Andresen and Damian Dubicki
Trondheim, June 20, 2013

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The following chapter introduces our master thesis written in the spring of 2013 at NTNU. We present the background and motivation for the project. In section 1.2 we define our goal more formally and present the research questions to which this thesis aims to answer. Section 1.3 explains the general structure of the thesis.

1.1 Background and Motivation

Soccer or associated football (will be referred to as football for the rest of this paper) is one of the world's most popular sports - particularly in Europe, where it is the most dominant spectator sport. Football is an excellent game for different types of betting. As a result, researchers have taken great interest in the sport - in attempts of reliably predicting the outcome of matches. A variety of models have been proposed, many of which look promising. Obtaining accurate predictions, however, is still considered a difficult problem. This is due to the sheer complexity of the game and the many factors that needs to be accounted for. As with other modelling tasks concerned with the real world, abstraction is key. Many attempts have been made in trying to identify the key aspects of the game - home ground effect, player skill, form, injuries and different psychological effects to mention some.

Models are based on statistical data alone, or include subjective knowledge (usually supplied by domain experts) - commonly referred to as expert models/systems. In order to keep complexity at a feasible level, each model is usually concerned with a limited number of parameters only. The availability of data has traditionally also been considered a constraint, limiting the scope and fullness of many state-of-the-art statistical models. With advances being made in both data availability and data processing ability, we may soon witness richer and more descriptive models.

The focus in this thesis is mainly on developing a model for prediction of matches in a round-robin tournament. Round-robin tournaments have a fixed match schedule where all teams play against each other, in turn, a predefined number of times. The resulting match structure presents a trivial modelling task. Most European football leagues use this type of scheduling, and this thesis looks at the English Premier League in particular.

Wrapping the model is a custom developed framework that aims to automate all the processes associated with prediction and betting. A detailed view of how the developed model and framework works and the key differences between this system and other systems of similar research will be given. Before discussing the models and implementations any further, a closer look at the mathematical theories behind is needed. This will be given in chapter 2.

1.2 Goals and research questions

A preface to this thesis, was a specialization project in the fall of 2012. Research on the topic of how to predict football results was conducted. Table 2.2 lists the findings of the project - such as what (mathematical) types of models has been attempted and what kind of data is consumed by them. A basic model and the naked spine of a framework was presented. This thesis continues where the specialization project left off and the goal is to develop a new model and a functioning framework with the capability of automating the processes of prediction and betting. The following concludes the goal of the thesis:

Goal *Extend the previous model to utilize more statistical data and develop a framework that can fully automate the processes of prediction and betting.*

The previous model is described in section 4.1. A description of how it works and what kind of data it consumes is given. It is a relatively simplistic model whose only data dependency is the results (goals scored) of previous matches. This is used to estimate attack- and defence strengths of each team. These (hidden) parameters are learned and used to estimate future results. The goal is to extend the model to utilize more statistical data and to investigate the impacts of this. The research questions of this thesis are defined as follows:

Research question 1 *What kind of statistical data would be wise to use in a prediction model?*

Research question 2 *How will this data impact a model's predicting ability*

1.3 Thesis Structure

This thesis is divided into 6 chapters, with subsections. Chapter one is an introduction if the thesis, including motivations for the thesis, goals and research questions. Chapter two follows with an background theory about the mathematical theory needed to grasp the concepts used by the model. In addition a description of the money management used by the framework and a brief section about previous work in the field. Chapter three introduces the framework developed during this thesis, and describing the different parts of the framework. Chapter four introduces with a description of a model developed in the specialization project, which was a preface to this thesis. Followed by the description of the model developed during this thesis. In chapter five, the experimental set-up and the results are presented. Concluding this thesis with chapter six, an evaluation of the results a long with a conclusion answering the research questions and comparing the goal to what have been achieved.

Chapter 2

Background Theory

The following chapter describes a brief introduction to the mathematical theory that is needed to get a grasp of how JAGS(Just Another Gibbs Sampler) works and how one can apply this theory to predict a football match. A long with an brief introduction to the strategy of the betting agent. Also as mentioned, some previous work in the field is presented.

2.1 Bayesian Networks

A Bayesian network is a graphical data structure used to represent dependencies among variables and a knowledge in an uncertain domain. Such a network is simply built up of nodes, which represent random variables with directed links that connects them. These links or edges, represents a statistical dependence between the nodes or variables. Bayesian networks are graphic in nature, and thus easier to read and understand than a mathematical formulated model. When a node has an directed link from itself X_j to another node X_i, then the value taken by X_i is dependent on the value of X_j. The node X_j is then often referred to as the parent node. Parent nodes have an direct influence of it's children nodes. To quantify this effect, each node X_i have a conditional probability distribution $P(X_i|Parents(X_i))$ of it parent node.

An example of two simple networks is shown in figure 2.1. This figure shows that X does not have any links from or to it, and therefore it is completely independent of the other variables. This means that it does not influence any of the other variables directly nor is it directly influenced by the other variables in the given example. Y and Z on the other hand, do have a directed link each from A. This shows that they are conditionally independent given A. What this means, is that Y and Z are independent of each other, but still both have a connection to A, and they are both influenced by A. In this example, A is the parent node of both Y and Z.

To define what a network means, the joint distribution over all the variables needs to be specified. A joint distribution is a table that denotes the probabilities of all the combinations of the values given, for example $P(X, Y)$ will give a table with all the possible combinations of X and Y. The conditional probabilities implied by the joint distribution are needed to define the network. To do this, the following formula is presented:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)) \tag{2.1}$$

Where $P(x_I|parents(X_i))$ are the conditional probabilities implied by the joint distribution.

2.2 Constructing a network

To correctly construct a Bayesian network, one needs to construct it in such a way that the nodes and arcs give a good representation of the domain one want

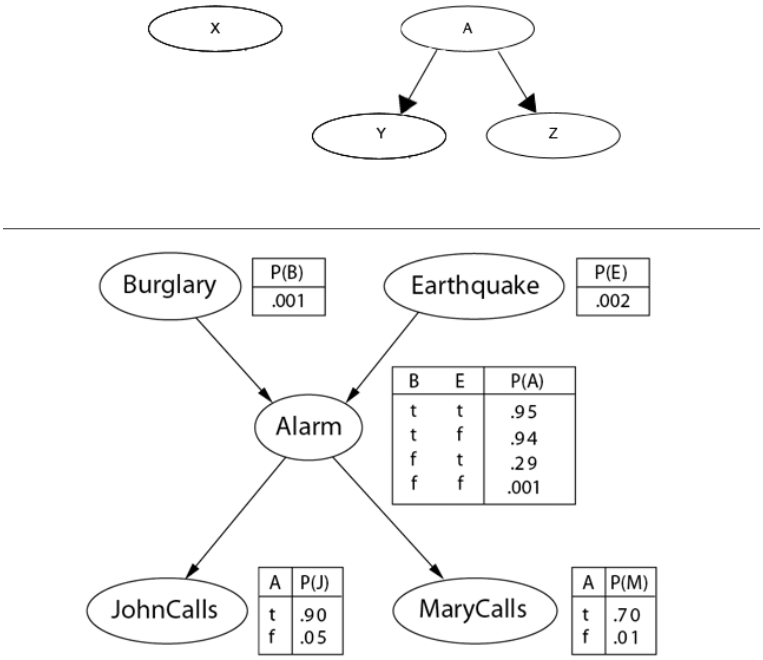

Figure 2.1: Example of two simple network. (Bottom figure is adapted from Russel and Norvig [2010])

to describe. A good representation is an network that is as compact as possible while being able to give as accurate representation of the domain as possible. Equation 2.1, shows relations that helps in constructing a network. If the joint distribution is rewritten using the product rule:

$$P(x_1, ..., x_n) = P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}, ...x_1) \tag{2.2}$$

And then repeat the process, the following big product will be the result:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1) \tag{2.3}$$

Comparing this to equation 2.1 shows that the joint distribution is equivalent to the general assertion that for every variable X_i

$$P(X_i|X_{i-1}, ..., X_1) = P(X_i|Parents(X_i)) \tag{2.4}$$

Given that ordering of the nudes is done in such a way that they are consistent with the partial order given in the structure of the network. What equation 2.4 states, is that a network is representing the domain correctly only if each node is conditionally independent of its other predecessors in the node ordering, given its parents. To achieve this, the first step is to determine how many variables or nodes are needed to model the domain. Then comes the ordering of the nodes, preferably in such a way that causes precede effects, this will ensure that the network will become more compact. The next step is to select links between the nodes. To accomplish this, a minimal set of parents for X_i is chosen, to satisfy equation 2.4, and a link is inserted between each parent and X_i. For the optimal result, one want the parent node X_i to contain every node that directly influence X_i. To finalize the network, adding a conditional probability table, $P(x_i|Parents(X_i))$ is advisable.

2.3 Markov Chains

To describe a Markov chain, a sequence of random variables is defined. The collection of the variables is the state space and each variable is a given state. The Markov chain is the sequence or chain of how the next sample is sampled from this state space. In addition there is an assumption that the next state is only dependent on a finite fixed number of previous states. The simplest Markov chain is called first-order Markov chain. Here the current state is only dependent on the previous state. This gives the formula:

$$P(X_t|X_{0:t-1}) = P(X_t|X_{t-1}) \tag{2.5}$$

Where X_t represent the unobservable variables. The notation $X_{0:t-1}$ denotes the set of variables from X_0 to X_{t-1}. The transition model is $P(X_t|X_{t-1})$, while a second order Markov process have $P(X_t|X_{t-2}, X_{t-1})$ as transition model. A transition model represents how a domain evolves. It also specifies the probability distribution over the newest state variable, given the previous value. The two transition models above correspond to figure 2.2, which shows the Bayesian networks for the first and second order Markov chain. Instead of specifying a new distribution for each time step, an assumption that changes in the domain state are caused by stationary process is made.

Let us consider an example (figure 2.3): You want to calculate a teams strength, based only on a match result, if the result is positive for the team,

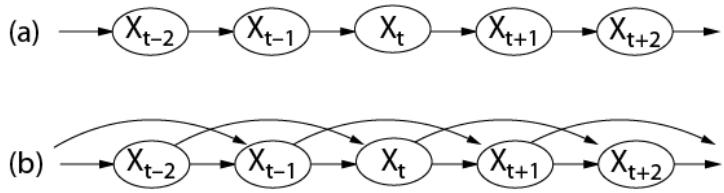

Figure 2.2: (a) Bayesian network of a first order Markov chain (b) Second order Markov chain (adapted from [Russel and Norvig, 2010])

the strength is updated with a positive value, if the result is negative it is updated with a negative value. Of course this is an very simplified example just to show the concept. For the given example, the conditional probability of the result is the same for all t, $P(R_t|R_{t-1})$, and therefore it is only needed to specify a conditional probability table once.

After specifying the transition model, it is needed to specify how evidence variables get their values. This is called the sensor model, and a sensor Markov assumption is made:

$$P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t) \tag{2.6}$$

Where E_t represents the observed variables and $P(E_t|X_t)$ is the sensor model. Figure 2.3 shows the transition and sensor model for the example above. It shows that the state of the world causes the sensor to take on values. Match result causes the strength to get a new value, positive or negative. In addition to the two models that are specified, the prior probability distribution, $P(X_0)$ also needs to be specified.

$$P(X_{0:T}, E_{1:T}) = P(X_0) \prod_{t=1}^{T} P(X_t|X_{t-1})P(E_t|X_t) \tag{2.7}$$

The right hand side is simply the initial state and the transition and sensor model. Figure 2.3 easily shows that the example is an first order Markov chain.

The Markov assumption in combination with the Ergodic theorem ensures that the chain gradually converges to a unique stationary distribution independent of its initial state. In simple words the Ergodic theorem in use with a Markov chain states that the chain has the possibility to go from any state to any other state, also the chain does not repeat an identical cycle. This ensures non repeating samples. This stationary distribution is denoted by $\phi(.)$ The period before

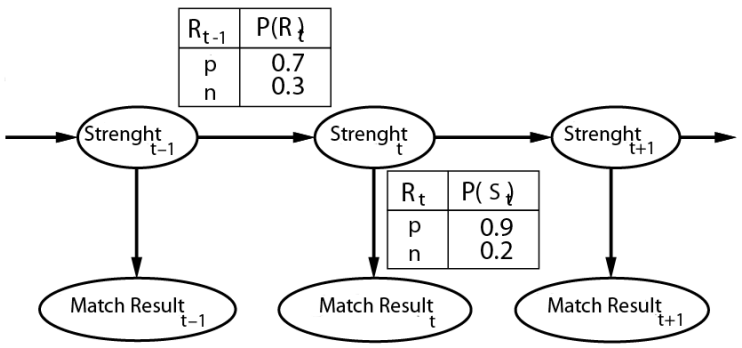

Figure 2.3: Bayesian network structure from simple strength example above, with transition and sensor model as described in the text

the chain converges is usually referred to as burn-in. The samples drawn during this period is seldom of any use, because they are still dependent (correlated) on the chains initial state. The preferred samples are the ones that are correlated with the stationary distribution only. An estimator for the expectation $E[f(X)]$ is given by:

$$f = \frac{1}{n-m} \sum_{t=m+1}^{n} \tag{2.8}$$

Where m denotes the number of burn-in samples to be discarded and n denotes the number of samples to calculate the expectation over.

2.4 Markov Chain Monte Carlo

Calculating the conditional probabilities and more importantly, the posterior distribution of the Bayesian model in figure 2.1 of the previous section is not very complicated, compared to calculating it for a model consisting of hundreds and even thousands of such connected nodes - with some of them even describing multivariate probability densities (depending on multiple variables and spanning multiple dimensions).

Markov Chain Monte Carlo (often abbreviated MCMC, as it will be done in this paper) is a methodology and a family of algorithms developed in order to estimate and analyse otherwise computationally infeasible problems. Having

such a broad range of applications and a simplistic nature, it acts as a framework - offering generic solutions to complex problems.

2.4.1 Use in Bayesian inference

Bayesian statistics revolves around a probability model, as section 2.1 visualized as an Bayesian network, containing variables that are all considered random quantities and are either observed (known) or unobserved (unknown). The figure 2.4 shows an example of this, were two teams with the observed variables attack and defence strengths and the unobserved variable for the goal probability is presented. Let D denote the set of observed data (evidence) and θ denote the set of unobserved variables of the network. $P(\theta)$ is then called the priors of the model and $P(D|\theta)$ the likelihood. $P(D)$ is called the marginal likelihood or model evidence. The full probability model is given by equation 2.9.

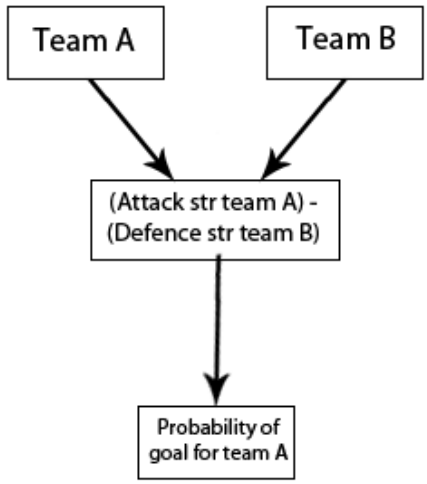

Figure 2.4: Bayesian network with observed and unobserved variables

$$P(D, \theta) = P(D|\theta)P(\theta) \tag{2.9}$$

The variable D have been observed, so by applying Bayes theorem the conditional probability is shown:

$$P(\theta, D) = \frac{P(\theta)P(D|\theta)}{P(D)} = \frac{P(\theta)P(D|\theta)}{\int P(\theta)P(D|\theta)d\theta} \propto P(\theta)P(D|\theta) \tag{2.10}$$

This conditional probability is known as the posterior distribution of θ, and is the key to all Bayesian inference (probability of θ observing D). $P(\theta)P(D|\theta)$ is proportional to $P(\theta|D)$, but the sum of probabilities of mutually exclusive events in this proportional distribution will not add up to one. Thus calculating the normalizing constant is needed.

In Bayesian inference, important properties of a distribution can be obtained by taking posterior expectations over functions of the priors θ. This can be expressed as

$$E[f(\theta)|D] = \frac{\int f(\theta)P(\theta)P(D|\theta)d\theta}{\int P(\theta)P(D|\theta)d\theta} \tag{2.11}$$

2.4.2 Monte Carlo integration

Monte Carlo integration is a special case of Monte Carlo simulation being applied to the task of obtaining expectations over functions of probability densities. This task usually involve the solving of integrals as the previous section shows. The expectation of a function $f(X)$ where $X = \{X_t, ..., X_{t+n}\}$ is a set of samples drawn from an arbitrary distribution $\pi(.)$ can be denoted:

$$E[f(X)] \approx \frac{1}{n}\sum_{t=1}^{n} f(X_t) \tag{2.12}$$

The population mean of $f(x)$ is the preferred, but an estimation from the sample mean is an good approximation given that the samples are independent and identically distributed(IDD) and the sample size n is large enough.

More effective methods for solving integrals numerically exists and they should generally be considered as a first choice. The problem however arises when the number of dimensions get big. It is no longer feasible to draw independent samples from the distribution.

This problem is easily solved by instead of sampling directly from the target distribution $\pi(.)$, samples are drawn from a distribution that is proportional to it. One way of doing this is to construct a Markov chain having $\pi(.)$ as its equilibrium or stationary distribution.

2.5 Metropolis-hastings Algorithm

Section 2.3 shows how easily a Markov chain is constructed and that it eventually will converge to a stationary distribution as long as it is ergodic. The problem was constructing a chain with a stationary distribution $\phi(.)$ equal to that of the distribution of interest $\pi(.)$. To justify the introductory comment on the simplistic nature of MCMC, the pseudo code for the Metropolis algorithm is provided in algorithm 1, which is used in Bayesian statistics to estimate posterior densities.

$X_0 \leftarrow 0$
for *t in 0:N* **do**
  $Y \leftarrow Sample[q(.|X_t)]$
  $U \leftarrow Sample[uniform(0,1)]$
  if $U <= a(X_t, Y)$ **then**
    $X_{t+1} \leftarrow Y$
  else
    $X_{t+1} = X_t$
  end
end

Algorithm 1: Metropolis-Hastings pseudo code

Here $q(.|.)$ refers to an arbitrary distribution called the proposal (or jump) distribution and $uniform(x, y)$ refers to the uniform distribution. Let $a(X, Y)$ denote the probability of accepting a move from X_t to a new candidate point Y. A popular choice for the proposal is a Gaussian with $q(.|.) \sim N(X_t, .)$. The target distribution is denoted by $\pi(.)$. The acceptance probability $a(X|Y)$ is then given by

$$a(X, Y) = min(1, \frac{\pi(Y)q(X|Y)}{\pi(X)q(Y|X)}) \tag{2.13}$$

Evaluating the expression contained in $a(X, Y)$, some of the symbols are recognized from the section about constructing Markov chains. Recall that (with the move being accepted (2.14))

$$Y = X_{t+1} \textbf{ and } X = X_t \tag{2.14}$$

Figure 2.5 illustrates the moves in the direction of a coordinate axis if the candidate is accepted.

It can be shown, due to detailed balancing, that $a(X, Y)$ (equation 2.13) will actually converge to the target density $\pi(.)$.

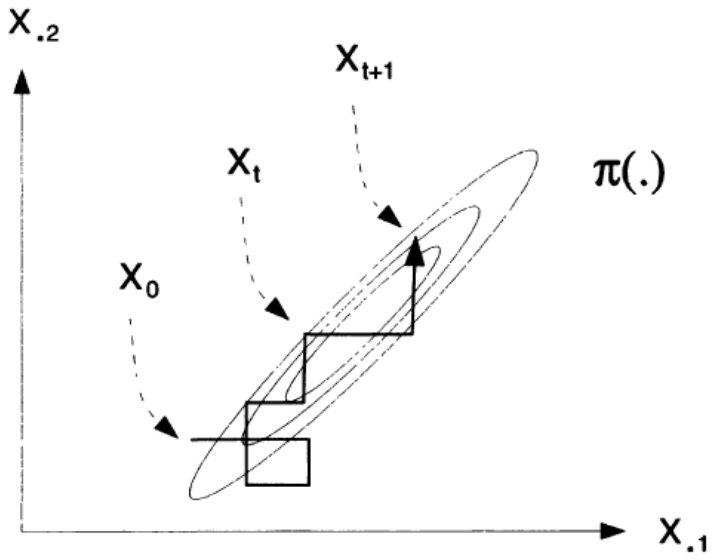

Figure 2.5: Single-component Metropolis-Hastings applied to a bivariate target distribution (adapted from Gilks and Richardson [2000])

$$\pi(X_t)P(X_{t+1}|X_t) = \pi(X_{t+1}P(X_t|X_{t+1})) \tag{2.15}$$

Obtaining equation 2.15, which represents the detailed balancing from equation 2.13 it can be integrate (equation 2.16) and proving that if X_t is from $\pi(.)$ then X_{t+1} will also be.

$$\int \pi(X_t)P(X_{t+1}|X_t)dX_t = \pi(X_{t+1}) \tag{2.16}$$

Then drawing samples will still be possible even with highly complex target distribution.

2.6 Gibbs sampling in Bayesian networks

Gibbs sampling is a special case of the Metropolis Hastings algorithm. Generating the next state with Gibbs sampling is done by randomly selecting a value for one of the non evidence variables X_i. The current values of the variables in the Markov blanket of X_i are used for sampling X_i (a Markov blanket consists of

parents, children and children's parents of a variable.). The algorithm will therefore randomly flip one variable at a time, keeping the evidence variables fixed.

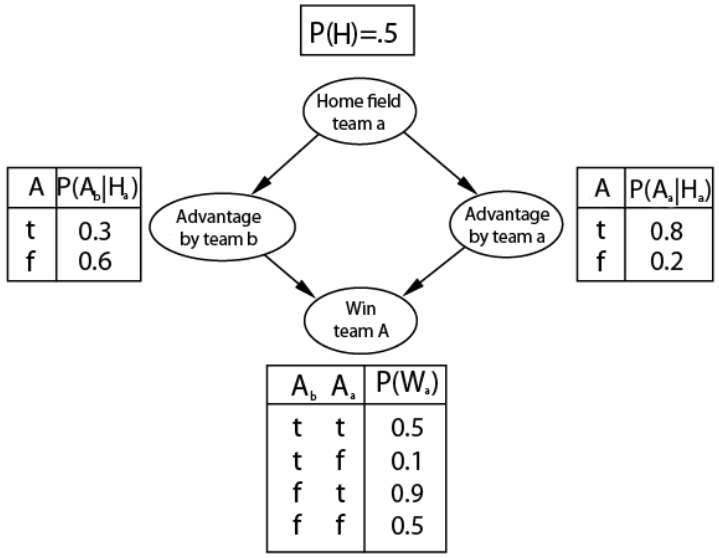

Figure 2.6: Example network with conditional probability tables

Figure 2.6 provides an example to explain this. For simplicity let Home field team a be denoted to $homefield_a$, advantage by team a to $advantage_a$, advantage by team b to $advantage_b$ and win team a to win_a. Further the assumption that the team on the home field have a greater chance to have an advantage over the other team, due to a psychological effect that they are playing at their home field. Then considering the problem of estimating $P(advantage_a|advantage_b = true, win_a = true)$. The Gibbs sampler will then initialize $advantage_a = true$ and $homefield_a = true$ randomly, giving the initial state:

$$[homefield_a = true, advantage_b = true, advantage_a = true, win_a = true].$$

At the next step a variable is selected randomly from $[advantage_a, homefield_a]$, since these two are the non evidence variables. If $homefield_a$ is sampled from $P(homefield_a|advantage_b = true, advantage_a = true)$ and we suppose that the result is false, the new current state will be: $[false, true, true, true]$

Further on, if $advantage_a$ is sampled, from $P(advantage_a|homefield_a = false, advantage_b = true, win_a = true)$ and the result will be false, the new state will be: $[false, true, false, true]$. As described, we see that the current values in the Markov blanket, are used for sampling a new value.

2.7 Money managment

Making money in the betting market is about more than just making good predictions. Determining what games to bet on and how much to bet is just as important. This section presents some of the theories behind making profitable bets.

2.7.1 The basics

When mixing money and uncertain events, such as the outcome of a football match or a stock's movement on the exchange, a new kind of currency is often needed to determine the real value of a potential investment. Investors are always interested in knowing the expected return of their investments - how much money they are expected to make. This value is usually taken as a probability-weighted expectancy over the different possible scenarios of the investment. An investment in the realm of sports betting refers to the action of placing a bet and the same principles apply.

Let P_h denote the probability of a home win in a given match. Let O_h denote the odds provided by a bookmaker on that outcome (home win). For simplicity we restrict ourselves to unit sized bets. Provided that we win the bet, our winnings are O_h and the return simply $R_{hw} = O_h - 1$. If we lose the bet, however, the return is $R_{hl} = -1$. We see that a bet is an investment over exactly two scenarios - win or lose. Given the probability of the different outcomes or scenarios of the match, we can calculate the expected return on the bet. Let $E[R_h]$ denote the expected return on a home win bet.

$$E[R_h] = P_h * R_{hw} + (1 - P_h)R_{hl} \tag{2.17}$$

Bets with a positive expected return are called profitable bets, because they are expected to make a profit. However, with uncertainty comes also risk. A method for measuring the uncertainty of a bet is by calculating it's variance. Variance is defined as the sum of the probability weighted squared deviations from the expected returns and is measured over the different scenarios of the bet (win/lose).

$$\sigma_h^2 = P_h * (R_{hw} - E[R_h])^2 + (1 - P_h)(R_{hl} - E[R_h])^2 \tag{2.18}$$

The expected return and variance of bets are the basic building blocks of many betting strategies. The following subsections explain how these measures can be used in practice.

2.7.2 Maximize return - minimize variance

With the theory from the previous section, a concrete implementation of a betting strategy can be built. This exact strategy is implemented in the framework under the name $MinVariance$. The goal is obvious - place bets so to obtain the highest possible expected return while keeping the variance low. Define $\beta_{i,j}$ where j denote the outcome of match i as:

$$\beta_{i,j} = max(0, \frac{E[R_{i,j}]}{2 * \sigma_{i,j}^2}) \tag{2.19}$$

Then select the outcome j from match i with maximal $\beta_{i,j}$. This ensures that only one bet is placed on each match i. Define a new set O containing the optimal bets. If the utility is to bet a fixed fraction of the current bankroll at a time, the bet amount of each bet can be found by dividing each optimal bet beta by the sum of all optimal bet betas and multiplying it with a total amount A.

$$b_i = \frac{\beta_{i,j}}{\sum \beta_{i,j}} * A, \beta_{i,j} \in O \tag{2.20}$$

2.7.3 Portfolio pricing - Modern Portfolio Theory

Portfolio pricing is a method that stems from the financial theory of Modern Portfolio Theory. It is mainly used in finance to minimize the overall variance of a stock portfolio. This is done by analysing how the stocks move in relation to each other - how they are correlated. The concept is that a collection of investments have a lower combined variance than a single asset. This effect is called diversification. The goal is to find the best possible risk-expected return profile, which simply is the best trade off alternative between risk and return. The expected return and variance of a portfolio is given by:

$$E(R_p) = \sum_i w_i E(R_i) \tag{2.21}$$

$$\sigma_p^2 = \sum_i \sum_j w_i w_j \sigma_{i,j} \tag{2.22}$$

Here R_p is the return on the portfolio while R_i is the return on asset i. w_i and w_j are weights, determining the amount of each asset in the portfolio. $\sigma_{i,j}$ is the covariance between asset i and j. The covariance measures the degree in which

σ_i^2	$\sigma_{i,j}$	$\sigma_{i,k}$
$\sigma_{i,j}$	σ_j^2	$\sigma_{j,k}$
$\sigma_{i,k}$	$\sigma_{i,k}$	σ_k^2

Table 2.1: An example of a variance-covariance matrix

assets move together through scenarios and is defined as the probability-weighted sum of the cross-product of deviations from the expected returns. When assets move in different directions given an event, we say that the assets are negatively correlated. Negative correlation between assets mean a negative covariance and this is responsible for the diversification effect.

$$\sigma_{i,j} = \sum_n \pi_n (R_{n,i} - E[R_i])(R_{n,j} - E[R_j]) \tag{2.23}$$

A more approachable solution to calculating the portfolio variance is to construct a variance-covariance matrix. An example is shown in table 2.1. By summing over all rows and columns of the matrix, we obtain the portfolio variance. Van der Wijst [2012]

The wanted risk-return profile is then found by adjusting each asset's corresponding weight w_i in accordance to some predefined utility. This non-trivial task can be solved by using an optimization algorithm. The implementation of the framework uses a *SteepestAscentOptimizer* together with a fitness function to find it's optimal risk-return profile.

The motivation behind introducing Modern Portfolio Theory to sports betting is that both markets share certain similarities. It is easy to picture matches and outcomes as assets in a portfolio. By allowing bets to be placed on more than one outcome per match, the experiment was to see if the diversification effect could improve the risk-aversive abilities of the betting agent. We know that the outcomes of a match are mutually exclusive and hence negatively correlated. However, it is still considered unwise to adapt a betting strategy like this. It is a known fact to the average better that the bookies claim a cut of all bets by "shorting" the odds. The results of this experiment is presented in chapter 5.

2.8 Previous work in the field

As mentioned, the preface to this thesis was an specialization project, where an structural literature review (SLR) was done, to try to establish what other work have been done in this field. In the SLR, 11 papers were selected and studied

further. And among other things, information about what kind of methods and what kind of data that were used was extracted. Table 2.2 shows what methods and what kind of data were used by each of the selected studies to predict match results. While table 2.3 shows the overview of the papers.

Qualified Study	**Methods used**	**Data used**
QS1	Bayesian dynamic generalized linear model and Markov chain Monte Carlo. Temporal model.	Result of previous games with a cap of 5 goals and a magnitude of the psychological effect of winning/losing "big". Uses first half of season as training set and the second half as test set.
QS2	Bivariate Poisson and Ordered probit	Win-draw-loose and score results from a 25 years old English league football data set.
QS3	Bayesian hierarchical Poisson-log normal distributed model.	Predefined vector of attack and defense abilities. Takes home advantage in to account.
QS4	Ordered logit regression	Match results of 14 seasons from 1993/94 2007/08 from Premiership, championship, league one and League two.
QS5	Dynamic generalized linear model with independent Poisson variables. MCMC for inference.	Previous match results with an evolution parameter.
QS6	Poisson log-linear	Greek First division of season 1997-98. Where data is categorized Goals scored by team A against team B, playing on ground C(Home advantage)
QS7	Bayesian Network	Home, draw and away results of all English Premier League matches from seasons 1993/94-2009/10. And subjective information about form, psychology and fatigue of a team.
QS8	Bivariate Poisson	All premier league match results from the past 9 seasons and home advantage.

QS9	Ordered probit and Monte Carlo	Results of two previous seasons of English league matches and William Hill bookmarker prices (for comparison). Also home advantage and distance between clubs is used.
QS10	Poisson	Match results from English Premier division and Italian Serie A of the season 1997-98.
QS11	Stochastic dynamic paired comparison model. Simulated maximum likelihood. Metropololis-Hastings.	Results from the 2008-2009 season of Italian Serie A are used to estimate team strengths (using two different models).

Table 2.2: Methods and data used by each of the Qualified studies from the SLR.

As table 2.2 shows, most of the data used to predict matches in previous studies that the SLR found, have only been based on match results. Some exceptions as subjective data about form, psychology and team fatigue and an home field advantage have also been used. Therefore the main goal with this thesis is to include more statistical data to predict results.

Qualified Study	**Title**	**Author**
QS1	Prediction and retrospective analysis of soccer matches in a league	Rue & Salvesen
QS2	Regression models for forecasting goals and match results in association football	Goddard
QS3	Bayesian hierarchical model for the prediction of football results	Baio & Blangiardo
QS4	Using ELO ratings for match result prediction in association football	Hvattum & Arntzen
QS5	Dynamic Bayesian forecasting models of football match outcomes with estimation of the evolution variance parameter	Owen
QS6	Statistical modeling for soccer games: The Greek league	Karlis & Ntzoufras
QS7	Pi-football: A Bayesian network model for forecasting Association Football match outcomes	Constantinou, Fenton & Neil
QS8	A dynamic bivariate Poisson model for analyzing and forecasting match results in the English Premier League	Koopman & Lit
QS9	Predicting bookmaker odds and efficiency for UK football	Grahm & Stott
QS10	On modeling Association Football data	Karlis & Ntzoufras
QS11	Stochastic dynamic Thurstone-Mosteller models for sports tournaments	Cattelan, Varin & Firth

Table 2.3: Papers selected for study in SLR

Chapter 3

Framework

This chapter presents the framework that have been extended and improved from the specialization project. A brief overview of the framework is given, followed by a more detailed description of the framework's major components. Section 3.2 describes how the system gathers the data necessary to supply the new model. In Section 3.3 the season simulator is presented. The automated betting abilities are described in section 3.5. Section 3.4 is concerned with presenting the workings of the betting agent places its bets. Finally, an explanation of how to effectively use the framework is given.

3.1 Framework: Overview

The framework is built on the .NET platform using the C# language. It is mostly comprised of injectable dependencies and plugins written in Java- and CoffeeScript. This was an early design choice made to simplify the process of updating components in case of sudden changes to the web-pages needed for data collection. The framework is supplied with a very thin console application that enables interaction with the underlying code library. The framework relies heavily on the open exchange format JSON for communication with the outside world.

The framework use extensive data mining to fetch all the statistical data needed to perform predictions and betting. The data is mined from web-pages such as `www.whoscored.com` and `www.oddsportal.com` and include data about fixtures, match statistics and odds for both historic and future matches. The framework has the ability to do predictions and betting in real-time, with fetching of new data and scheduling automatically maintaned. The framework also features a multi-threaded wrapper around the JAGS software that enables a fast and intuitive interface for generating and parsing samples from models written in the JAGS language. The betting strategies described in 2.7 are also fully implemented. An extension to allow fully automated betting was also developed, but is currently broken due to changes made on the bookmaker's website.

3.2 Framework: Data mining

The framework relies heavily on data mining to gather all the data needed by the model. Direct data feeds exists, but are very expensive. The framework taps into publicly available data by using a wrapper to communicate with a web testing framework called PhantomJS. PhantomJS is a scriptable, headless web browser that enables DOM-manipulation and script injection on websites (only local changes). Small interchangeable scripts, called manuscripts, are used to control the behaviour of the browser and enables the betting framework to extract data from virtually any website. Figure 3.1 shows some of the data that is available via `www.whoscored.com`.

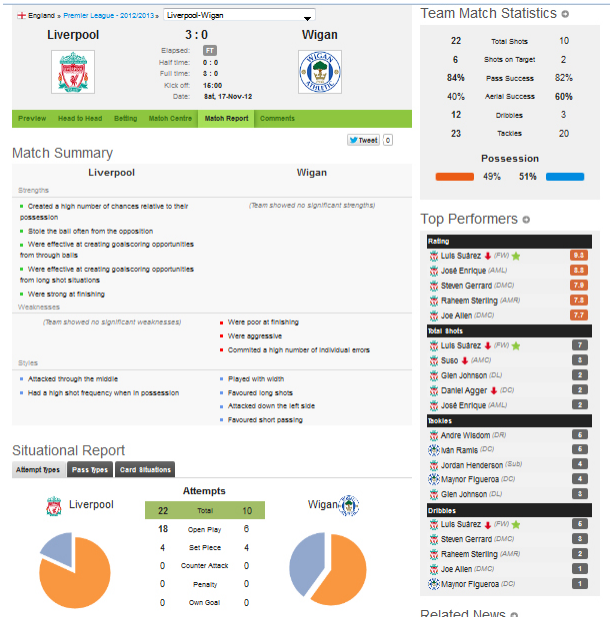

Figure 3.1: Match statistics from `www.whoscored.com`

Data
Team ID
Blocked Scoring Attempts
Post Scoring attempts
Total Passes
Accurate Passes
Total Tackles
Total Offsides
Won Contests
Shots Off Target
On Target Scoring Attempts
Total Scoring Attempts
Aerials Won
Aerials Lost
Fouls
Throws
Corners
Ball Possession

Table 3.1: Data fetched from `www.whoscored.com`

3.3 Framework: Season simulator

One of the most important components of the framework is the season simulator. Provided with a match schedule (list of fixtures) it handles clustering of matches, predictions and delegates tasks to other components of the system at the right times. Clustering in this context refers to the task of determining what matches should be predicted next. This operation is essential because the JAGS software is very sensitive with regards to the ordering and structure of data. Logic implemented in the simulator aims to make this scheduling as effective as possible. The simulator also has support for real-time simulation of ongoing seasons. Internal scheduling is responsible for fetching new data and running predictions at times calculated from the match schedule.

3.4 Framework: Betting Agent

The betting agent use the predictions of the model combined with odds from a bookmaker to suggest profitable bets from a set of matches. This is done by using the betting strategies described in 2.7. The betting agent is assigned a specific betting style that effects how much the agent is willing to bet. Three different styles are implemented - conservative, moderate and aggressive.

3.5 Framework: Automated betting

Using the same mechanisms involved in data mining, the framework aims to implement fully automated betting. Provided with an account on a betting site, the agent is able to automatically log in and bet on the user's behalf. Sadly this functionality is now broken as the bookmaker of choice decided to completely change the structure of their website.

3.6 Framework: Reporting

The framework also has a component for generating reports from simulations. The reports are in JSON-format and enables instant posting of results to e.g. a website.

Chapter 4

Description of the Model

This chapter consists of a description of the JAGS model developed during this thesis. Section 4.1 includes a description of the model developed during the specialization project. Which is the base of the version 2.1, which is described in section 4.2.

4.1 Model v1.0

This model is was developed during our specialization project fall 2012. The full model is found in appendix A. The model is based on the Rue and Salvesen model from their paper [Rue and Salvesen, 2000]. This version is however simplified, it does not include freak results (matches that have goals equal or higher to 5 goals for each team) or the mixture model adapted from [Dixon and Coles, 1997] that gives higher probabilities for 0-0 and 1-1 results on the cost of 1-0 and 0-1. The most important variables in the model are the attack and defence strengths of each team. These are represented by random variables a(attack) and b(defence). High values of these mean good attack and defence strengths. The variable $e_A = (a, d)_A$ denotes the properties of team A and similarly for team B. Further on the prior mean $\mu_{a,A}$ and variance $\sigma^2_{a,A}$ for a_A and similarly for defence strengths and for team B is found. The model is used to predict match results in the future.

4.1.1 The goal model

A result is denoted as $(X_{A,B}, Y_{A,B})$. And it is needed to specify how it depends on the properties of home team A and away team B. The assumption that the number of goals scored $(X_{A,B})$ by team A is dependent on the teams attack strength and team B's defence strength. And similarly the other way around with goals scored $(Y_{A,B})$ by team B. Let $\Delta_{AB} = \frac{(a_A + d_A - a_B - d_B)}{2}$ denote to the difference in strength between the teams. In addition the following assumption is made (equation 4.1):

$$\begin{aligned} x_{A,B}|(e_A, e_B) &= X_{A,B}|a_A - d_B - \gamma\Delta_{AB} \\ y_{A,B}|(e_A, e_B) &= X_{A,B}|a_B - d_A + \gamma\Delta_{AB} \end{aligned} \tag{4.1}$$

Where γ represents the psychological effect that team A may underestimate team B if the are a weaker team. Since both teams play in the same league the reasonable expectation of the effect is that $\gamma > 0$. The opposite effect($\gamma < 0$) is not expected, because that would mean that team A would be much more superior than team B, an that is not likely to occur in the same league. Further one Rue and Salvesen [Rue and Salvesen, 2000] found that previous results in over 900 matches a long with the nature of the game, suggests to a first approximation of a Poisson law for $x_{A,B}$ and $y_{A,B}$. And therefore they assume that the number of goals conditioned on teams properties is Poisson distributed with mean $\lambda^{(x)}_{A,B}$ and $\lambda^{(y)}_{A,B}$ where

$$\begin{aligned} \log(\lambda_{A,B}^{(x)}) &= c^{(x)} a_A - d_B - \gamma \Delta_{AB} \\ \log(\lambda_{A,B}^{(y)}) &= c^{(y)} a_B - d_A - \gamma \Delta_{AB} \end{aligned} \tag{4.2}$$

Where $\lambda_{A,B}^{(x)})$ and $\lambda_{A,B}^{(y)})$ is the mean, followed by the constraints $c^{(x)}$ and $c^{(y)}$ that roughly describe the logarithm of the empirical mean of the home and away goals.

4.1.2 The time model

The model is dynamic in regards of the attack and defence strengths. Which is maintained by a time constant. An assumption that t' and $t'' > t'$ are two time points within a common reference point where team A plays a match. The attack strength of team A at the time t'', $a_A^{t''}$ is then dependent on the previous strength $a_A^{t'}$. To specify these dependencies, Rue and Salvesen [Rue and Salvesen, 2000] used a Brownian motion to tie the attack strength with the two time points t' and t'' as equation 4.3 shows.

$$a_A^{t''} = E a_A^{t'} + \{B_{a,A}(\frac{t''}{\tau}) - B_{a,A}(\frac{t'}{\tau})\} \frac{\sigma_{a,A}}{\sqrt{\{1-\gamma(1-\gamma/2)\}}} \tag{4.3}$$

The parameter γ is the same for all teams and represents the inverse loss of memory rate for $a_A^t, var(a_A^{t''} - a\ t') \propto \sigma_{a,A}^2/\tau$. Although the process will give and variance that increases towards ∞ with increased time, the conditioning on match results will ensure smooth posterior realizations of the properties and reasonable assumptions into the near future.

4.1.3 The full model

After elaborating the two components, the full model can be build. If we look at an example of four teams that play 3x2 matches at three different times, and the fourth match is the one to be predicted. To predict the match, one does look at the previous results, which are given as $e_A^{t_0}, e_B^{t_0}, e_C^{t_0}, e_D^{t_0}$ at time t_0 and similarly for t_1 and so on. Then θ is written for all the variables in the model, and equation 4.4 is presented to predict the results at t_3.

$$\begin{aligned} \pi(\theta) =& \pi(e_A^{t_0})\pi(e_B^{t_0})\pi(e_C^{t_0})\pi(e_D^{t_0}) \\ & \times \pi(x_{A,B}^{t0}, y_{A,B}^{t_0} | e_A^{t_0}, e_B^{t_0}) \pi(x_{C,D}^{t0}, y_{C,D}^{t0} | e_C^{t_0}, e_D^{t_0}) \\ & \times \pi(e_A^{t_1} | e_A^{t_0}) \pi(e_B^{t_1} | e_B^{t_0}) \pi(e_C^{t_1} | e_C^{t_0}) \pi(e_D^{t_1} | e_D^{t_0}) \\ & \times \pi(x_{A,C}^{t1}, y_{A,C}^{t_1} | e_A^{t_1}, e_C^{t_1}) \pi(x_{B,D}^{t1}, y_{B,D}^{t1} | e_B^{t_1}, e_D^{t_1}) \\ & \times ... \end{aligned} \tag{4.4}$$

To elaborate equation 4.4, one can look at figure 4.1, that represents the basic structure of the model. The first line of the equation, represents the prior density for each team. The next line represents the goal model, were the calculations of each teams parameters are set up against each other to predict the scores. The next line uses the Brownian motion to adjust the prior parameter with the next time slice and so on for the rest of the model.

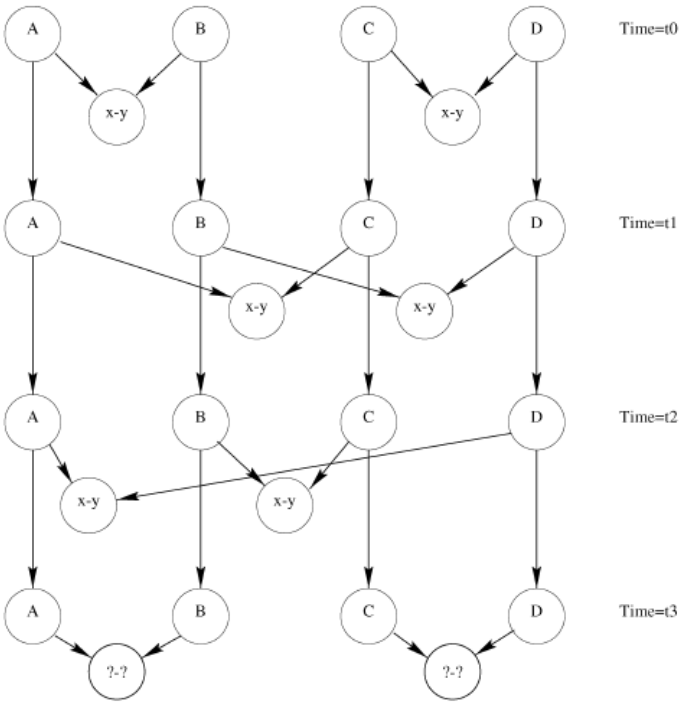

Figure 4.1: The structure of the full model with four teams and eight matches (adapted from Rue and Salvesen [2000])

4.2 Model v2.1

The model developed during this master thesis is found in its full in appendix B. And this section describes it. The major changes from the previous model lay in the usage of more statistical data. As section 2.8 shows, our founding did not indicate that it was common to use additional statistical data to predict results,

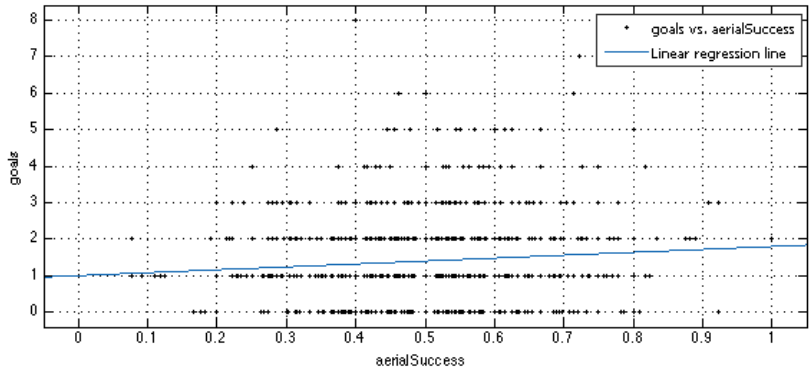

Figure 4.2: Correlation between aerial success and goals scored

which was the main reason to include these. After the decision to include more data, some research to find out what kind of data to would include was done. Figures 4.2 through 4.7 shows an correlation between the statistical data that was tested with number of goals scored in a match. And these correlations are the reason for the particular data chosen. Table 4.1 shows what variables the model use.

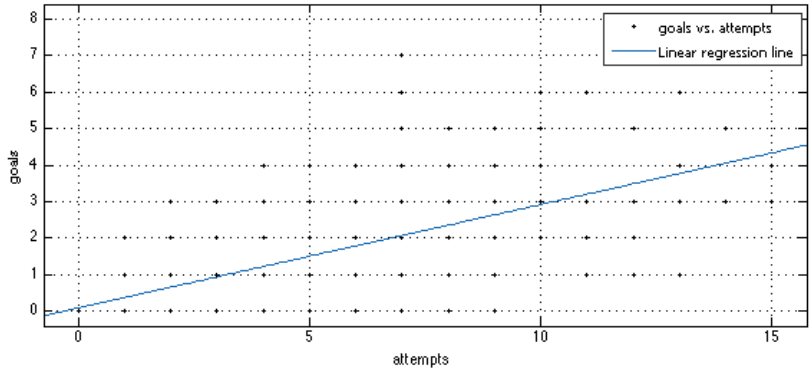

Figure 4.3: Correlation between attempts and goals scored

As we can se, each of the figures that represent the correlation between the different parameters and goals scored, does impact on goals scored in different levels. The table 4.1 show us an overview of the variables used in the model. More detailed description of these variables are found in the table 4.1 through

Observable variables	**Description**	**Calculations of variables**
goalsScored	How many goals a team have scored	Direct data from source
passSuccess	The success rate of a teams passes	Accurate passes divided with total passes
aerialSuccess	The success rate of a teams aerial duels	Aerial duels won divided with duels lost and won
contestSuccess	The success rate of a teams contest	Contests won for home team divided with contests won for home team with contests won for away team
possession	The teams possession	Direct data from source
attempts	The teams number of attempts	Direct data from source

Table 4.1: Data used in the model

4.2.3 below.

4.2.1 The goal model

Similarly to version 1.0 a result is denoted as $(X_{A,B}, Y_{A,B})$. And it is needed to specify how it depends on the properties of home team A and away team B. An the assumption that the number of goals scored $(X_{A,B})$ by team A is dependent of teams attack strength and team $B's$ defence strength, and similarly the other way around. The difference from the previous model, is that the attack and defence strengths consist of more data. Let $Skill_{Att,A}$ denote the attach strength of team A, and $Skill_{Def,A}$ denote the defence strength of team A. Let $\Delta_{AB} = \frac{(Skill_{Att,A}+Skill_{Def,A}-Skill_{Att,B}-Skill_{Def,B})}{2}$ denote the difference in strength between the teams. Given the Poisson fits shown in figures 4.8 and 4.9, an assumption is made that the number of goals made by each team is Poisson

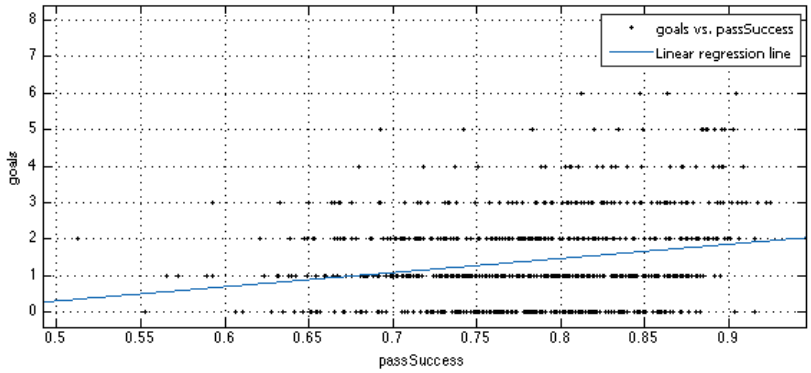

Figure 4.4: Correlation between pass success and goals scored

distributed with mean $\lambda_{A,B}^{(x)}$ and $\lambda_{A,B}^{(y)}$ where

$$\begin{aligned} \log(\lambda_{A,B}^{(x)}) &= Skill_{Att,A} - Skill_{Def,A} - \gamma\Delta_{AB} \\ \log(\lambda_{A,B}^{(y)}) &= Skill_{Att,B} - Skill_{Def,A} - \gamma\Delta_{AB} \end{aligned} \tag{4.5}$$

Further on an elaboration of the parameters $Skill_{Att,A}$ and $Skill_{Def,A}$ is needed, since it is here were the additional data lays. Within the parameter $Skill_{Att,A}$ we find the following:

$$\begin{aligned} &(passSucces_A - passSuccess_B)+ \\ &(aerialSuccess_A - aerialSuccess_B)+ \\ &(contestSuccess_A - contestSuccess_B)+ \\ &(possession_A - possesion_B)+ \\ &(attempts_A - saves_B) \end{aligned} \tag{4.6}$$

While the parameter $Skill_{Def_A}$ include:

$$\begin{aligned} &(passSucces_A - passSuccess_B)+ \\ &(aerialSuccess_A - aerialSuccess_B)+ \\ &(contestSuccess_A - contestSuccess_B)+ \\ &(possession_A - possesion_B)+ \\ &(saves_A - attempts_B) \end{aligned} \tag{4.7}$$

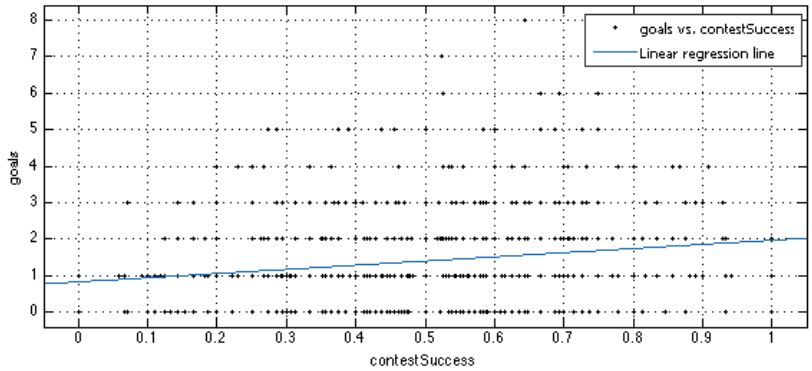

Figure 4.5: Correlation between contest success and goals scored

4.2.2 The time model

The time model is similar to the one found in the previous model, described in section 4.1.2. But due to usage of the beta distribution, we implemented a scaler to mimic the behaviour of Brownian motion in a beta distribution. This scaler is defined as following:

$$Scaler = 10 - (B^t - B^{t-1}) \tag{4.8}$$

Were B^t denotes days since last match

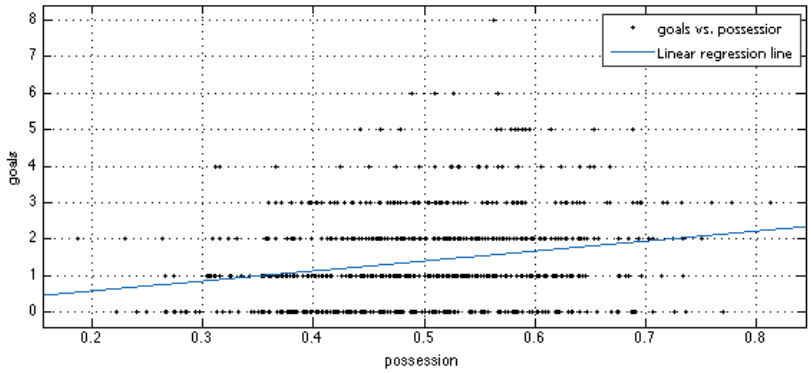

Figure 4.6: Correlation between possession and goals scored

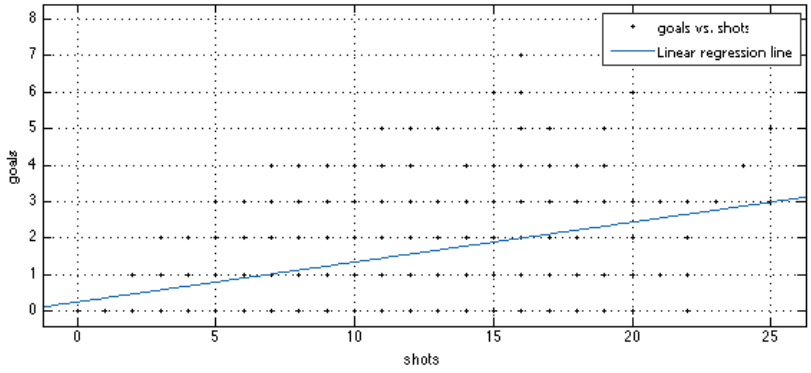

Figure 4.7: Correlation between shots and goals scored

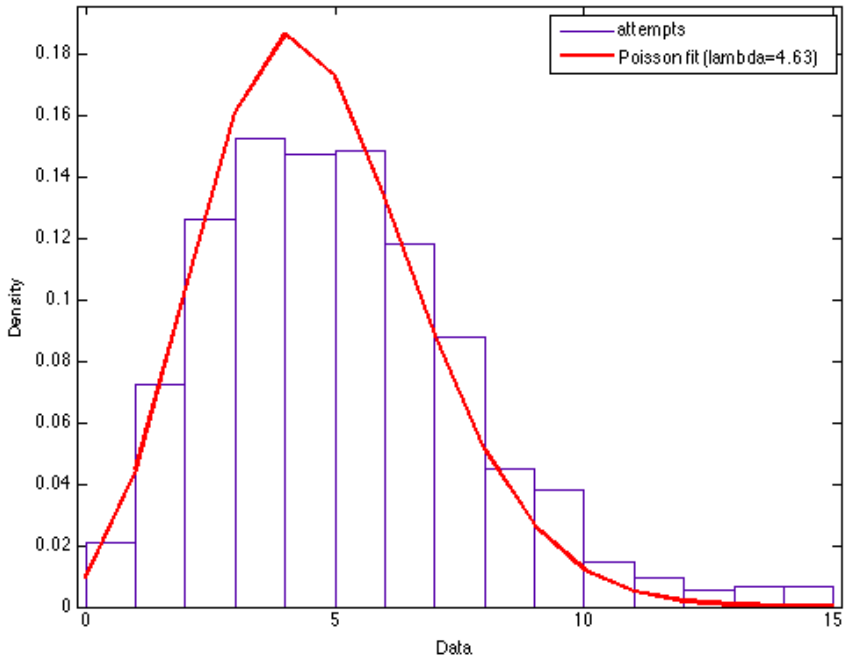

Figure 4.8: Poisson fit on attempts

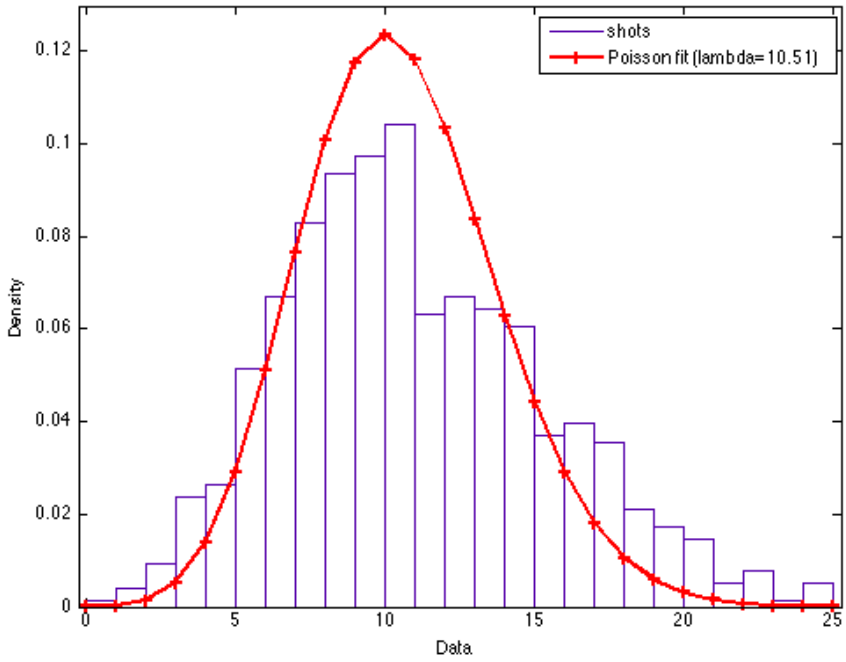

Figure 4.9: Poisson fit on shots

4.2.3 The full model

The implementation of the full model is similar to the implementation found in the previous version of the model, found in section 4.1.3

Chapter 5

Experimental set-up and Results

This chapter explains the experimental set-up of the presented model, in addition the results of the model running with the presented system to predict match results and bet on matches is presented.

5.1 Experimental set-up

To se how well the presented model is performing, the choice of profitability of the model while betting on matches was an natural selection. The model is using the first half of the Premier League season 2011/2012 to train, and then predicts the second half of the season. Using odds provided from `www.oddsportal.com` the system uses either MPT or Min variance (both described in 2.7). The following settings are used for the model:

- Burn-in: 1000
- Chains: 3
- Samples: 1000

The data above, is backed up by the trace samples shown in figure 5.1. Were it shows that the parameters in the model are independent of each other.

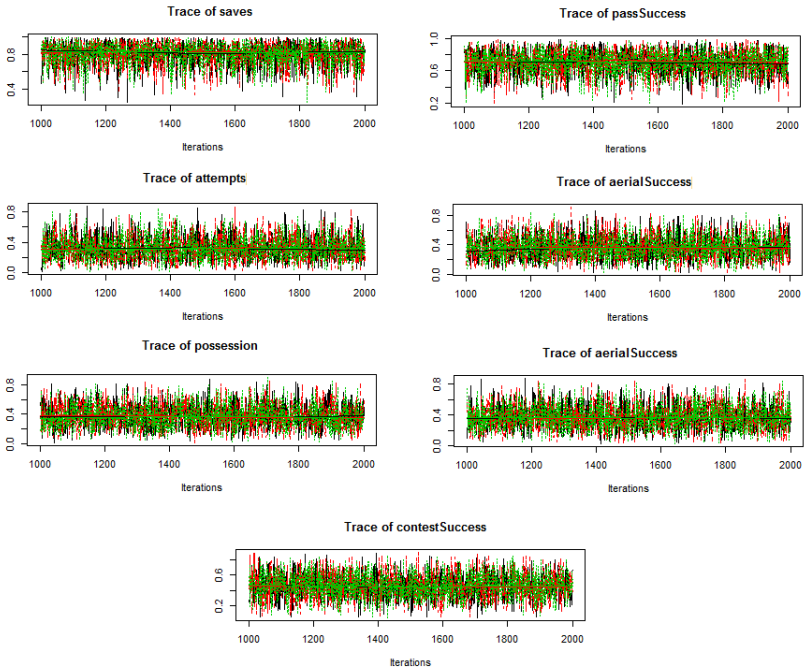

Figure 5.1: Trace of variables used in the model

5.2 Results

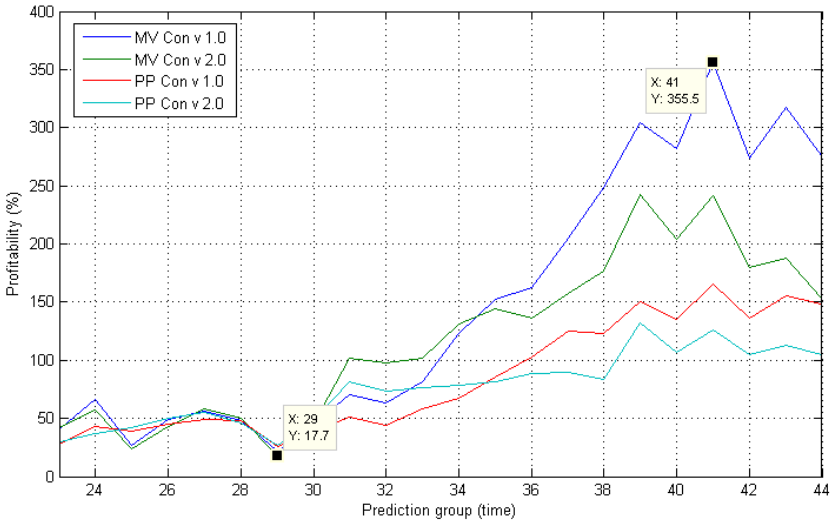

Figure 5.2: Profitability with the conservative betting agent

Figures 5.2, 5.3 and 5.4 shows the results the described system outputs. Each figure shows four different plots. Two for each model and two for each betting strategy. One of them (PP in graphs) based on the MPT described in section 2.7.3. The other strategy is Min variance (MV in graphs), which is based on the strategy used in [Rue and Salvesen, 2000] and described in section 2.7.2. The three figures represent how aggressive the betting agent is with its bets, the three different styles are conservative, moderate and aggressive. Each of the different betting styles, does simply place higher bets based on how aggressive it is.

Looking at the figures, the best combination is the moderate better of model version 1.0 with the MPT strategy (—). Which is the one that gives the best profitability if the actual bets were placed on the odds provided from `www.oddsportal.com`, with the total profit being close to 500 percent at the end of the season, and peeking at over 1100 percent. On the other end of the scale, the worst combination is found using model version 2.0 combined with the aggressive style and the Min variance strategy(—). This combination gives a loss of about 50 percent.

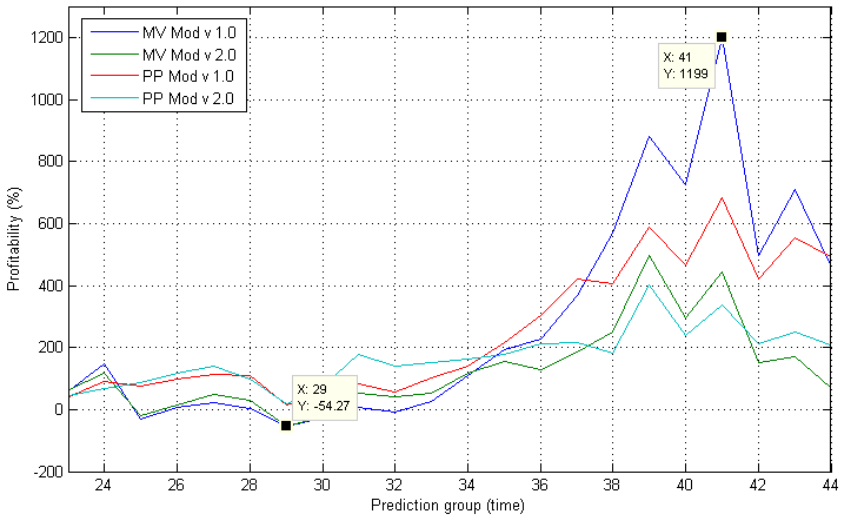

Figure 5.3: Profitability with the moderate betting agent

Analysing each of the figures in more details, shows that model version 1.0 (— and —) is the most profitable model overall. Looking into more details, figure 5.2 shows that of the four combinations, (model version 1.0 and 2.0, with MPT or Min variance) with the conservative style, the most profitable outcome is model version 1.0 with Min variance strategy (—). While the least profitable is model version 2.0 combined with MPT (—). While the moderate style, in figure 5.3 shows that MPT and model version 1.0 (—) gives the best profitability. Where Min variance with model version 2.0 (—) is the least profitable. The aggressive better shown in figure 5.4, shows that the best outcome, is again MPT combined with model 1.0 (—). While the worst is Min variance combined with model 2.0 (—). Surprisingly though, model version 1.0 combined with the Min variance (—), is very close to being the worst case in this scenario.

Further analysing shows that the all over, the conservative style is, not surprisingly, the safest style. Where non of the combinations ever become negative in terms of profit. And all over, the model version 1.0 combined with MPT (—) have the most stable progress. With very few high losses or gains over all three betting styles. While model version 1.0 combined with Min variance (—) and the aggressive style has the most unstable progress, leaping from over 1000 percent profit to a loss of 25 percent.

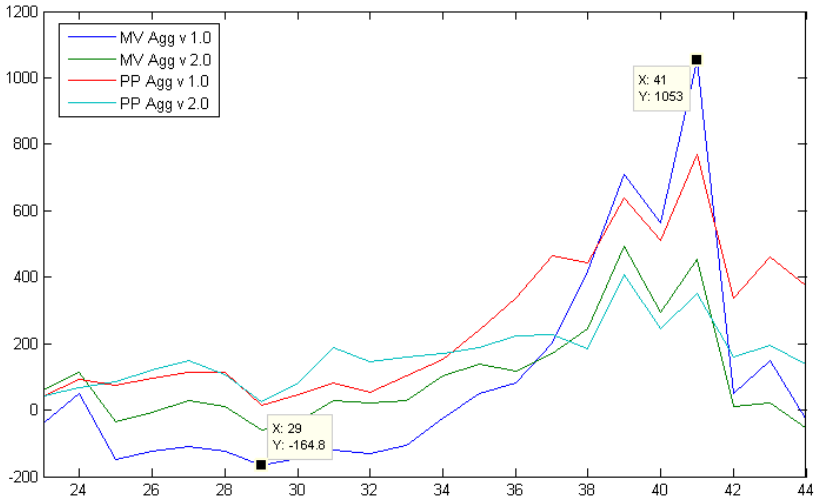

Figure 5.4: Profitability with the aggressive betting agent

In addition to profitability, figures 5.5 and 5.6 shows how the two models, version 1.0 and 2.0 respectively, compares their predicted outcome to the odds. The plots only shows for 40 games, to make them readable, compared to 190 matches. The optimal model would of course be one where the probability of the actual outcome would be as close to 100 percent as possible. This is of course an task that is nearly impossible, but an model that at average has better probability of the outcome, or as close as possible compared to the odds would do very good. Comparing figure 5.5 and 5.6, both models perform similarly in terms of how close they were compared to the odds. But a closer look reveals that model version 1.0 performs slightly better than model version 2.0 for that given span of game numbers.

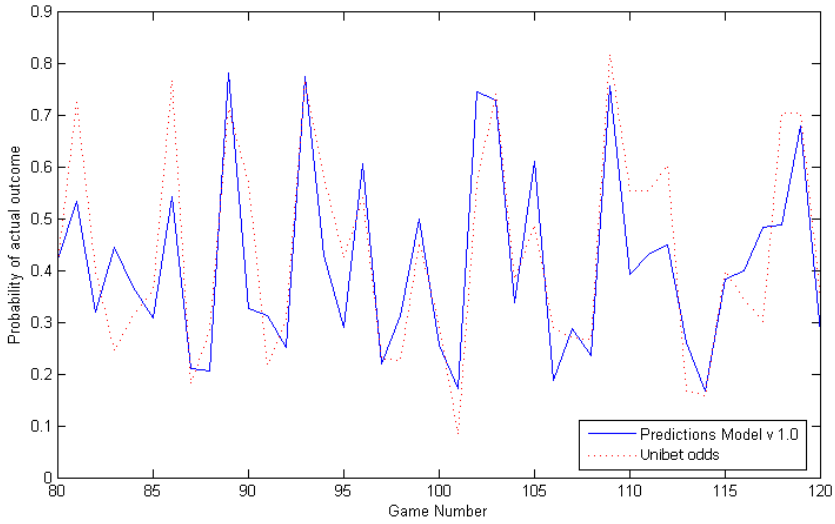

Figure 5.5: Predictions versus odds in model version 1.0, game 80 through 120

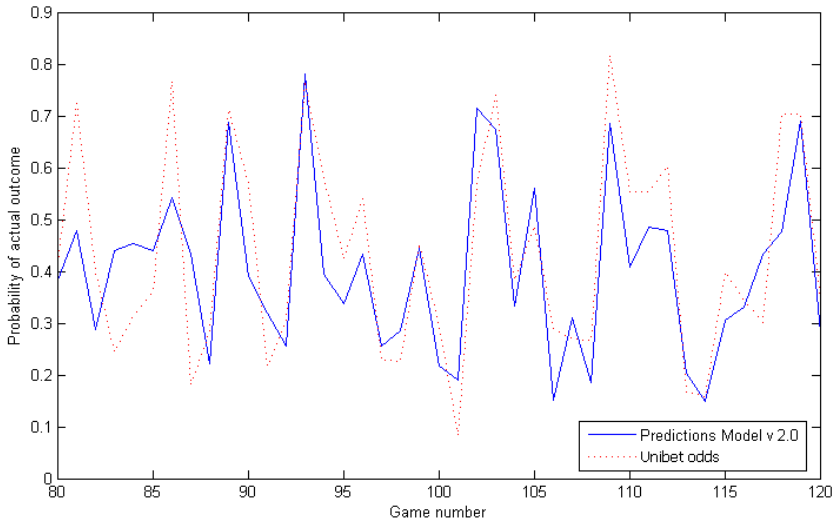

Figure 5.6: Predictions versus odds in model version 2.0, game 80 through 120

Chapter 6

Evaluation and Conclusion

In this chapter, a discussion is presented about the result of the previous chapter. In addition a conclusion and final words about this thesis.

6.1 Evaluation

Looking back at section 5.2, it shows that overall the first version of the model performs better than the model developed during this thesis. The results may be somewhat surprising, considering the figures presented in chapter 4. Were an correlation between the added parameters in the new model and goals scored are shown. Considering the added parameters should have an impact to the accuracy of the model, it may have been other factors than just the fact that adding more data does not help improve the accuracy of an model to predict results. One of the different factors that may have caused the presented results may be over-fitting. Over fitting is a term used in machine learning and statistic, and occurs when one does have to many parameters or when a model is to complex. And this leads to random errors and noise instead of the actual relationship with data. As described in chapter 4, model version 2.0 is fairly more complex than version 1.0. Thus having a greater chance of over-fitting the data. Another consideration to take into account is that the new statistical data used in the model version 2.0, may have been used in a wrong way. Comparing the statistical data within the JAGS model may have been done differently, also different settings of the same model would give different results. For example weighting the statistical data in another way could have improved the model. As mentioned, model version 2.0 is fairly more complex than the first version, increasing the chance for having an mathematical error or an misinterpretation while reading the JAGS manual. In addition, i may be possible that the data are simply reflected good enough by the outcome of the match, and that the singular data on its own do not impact as much as one should imagine. Last but not least, the possibility that the implementation of the framework may have an error, can impact largely on the output of the model. Thus, extensive debugging have been done to ensure this is not the case, one can never be absolutely sure when a system reaches such an extent as the described system.

The presented goal in this thesis was as mentioned to utilize more statistical data in a model, and to develop a framework that can fully automate the process of prediction and betting. As presented in section 4.2, the model does utilize more data than were used in the previous version of the model. In addition, one of the research questions was about what kind of statistical data that would be wise to use in a prediction model. The answer to this, is described in section 4.2, were correlation figures of the impact of data compared to goals are presented. These findings present that some data does impact more than others on the final result of a match. The second reaserch question was about how these data would impact on the models prediction ability. The results in section 5.2, speak for them self, but as mentioned above, other factors may have interfered and given

false results, thus we believe that this the results are representable and give an accurate description for the two models. The second part of this thesis goal, was about the framework. And as presented in chapter 3, looking aside from the placing of the actual bets with real money, the system is as automated as it can be. Everything from scraping data to predictions and placing imaginary bets is done within the framework, thus making it as the goal intended.

6.2 Conclusion

Based on the section 6.1, we do not want to conclude that adding more statistical data will lead to an worse prediction model. As mentioned, the results may be compromised due to different factors, we do however feel confident that our findings are fairly accurate. And given enough time to do eave more extensive testing and research an model with more statistical data still can perform well. The model version 2.0 did return a positive profit in with both the conservative and moderate betting style, thus we can conclude that adding more data do not make an prediction model completely unusable. Section 6.1 does also discuss our goal and research questions. We feel that our main goal have been reached, since we did manage to build a model using more data, that is profitable and fairly accurate compared to bookmarker odds. Also a framework that is as automated as it gets, excluding the real betting, is build and presented. The research question have also been answered as good as they can be given our time frame. And we feel confident that we have presented and developed a good framework that is very well suitable for using with JAGS. In addition to the main goal and the research question, we want to point out the idea of implementing MPT. We feel that this betting strategy may be worth looking into, since it gave fairly good results as shown in section 5.2.

Bibliography

M. J. Dixon and S. G. Coles. Modelling association football scores and inefciencies in the football betting market. pages 265–280, 1997.

W. R. Gilks and S. Richardson. *Markov Chain Monte Carlo in Practice*. 2000.

Hvard Rue and O. Salvesen. Prediction and retrospective analysis of soccer matches in a league. 2000.

Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. 2010.

Dominicus Van der Wijst. Ti4146: Finance for science and technology students [compendium]. 2012.

Appendix A

Appendix B: JAGS model v1.0

```
### TBM v1.0
#####################################################

data {
  # Set C^(x) and C^(y) as constants
  # Values reported by Rue & Salvesen
  homeGoalAvg <- 0.395
  awayGoalAvg <- 0.098
}

model {
  ### Time model
  ################################################

  tau ~ dgamma(10, 1)
  precision ~ dgamma(10, 0.1)

  # Loop through all teams and all timeslices
  for(t in 1:noTeams) {
    # Initial distribution of attack/defense strength
    attack[t, 1]  ~ dnorm(0, precision)
    defense[t, 1]  ~ dnorm(0, precision)

    # Evolve the attack/defense strength over time with Brownian motion
        (Wiener process).
```

```
  ## The parameters are normally distributed with mean equal to the
      parameter value of
  ## the previous timeslice (round). Loss-of-memory effect provided by
      variance parameter.
  for(s in 2:noTimeslices) {

    attack[t, s]  ~ dnorm(
                attack[t, (s-1)],
                (
                  ((abs( days[t,s] - days[t,s-1] )) / tau) * precision
                )
              )

    defense[t, s]  ~ dnorm(
                defense[t, (s-1)],
                (
                  ((abs( days[t,s] - days[t,s-1] )) / tau) * precision
                )
              )
  }
}

### Goal model
##################################################

# Params:
## attack[t, s] / defense[t, s] = attack/defense strength for team t at
    given timeslice s
## team[i, 1] = home team in game i
## team[i, 2] = away team in game i
## timeslice[i, 1] = timeslice(round/games played so far) of home team
    in game i
## timeslice[i, 2] = timeslice(round/games played so far) of away team
    in game i
## goalsScored[i, 1] = goals scored by home team in game i
## goalsScored[i, 2] = goals scored by away team in game i

# Give the delta-parameter some room to move
gamma ~ dunif(0, 0.1)

# Loop through all games in correct order
for(i in 1:noGames) {

  # delta param for psychological effect of underestimating
  delta[i]     <-  (
```

```
                attack[team[i, 1], timeslice[i, 1]] + defense[team[i,
                    1], timeslice[i, 1]] -
                attack[team[i, 2], timeslice[i, 2]] - defense[team[i,
                    2], timeslice[i, 2]]
            ) / 2

    log(homeLambda[i]) <-  (
                homeGoalAvg +
                (
                  attack[team[i, 1], timeslice[i, 1]] -
                  defense[team[i, 2], timeslice[i, 2]] -
                  gamma * delta[i]
                )
              )

    log(awayLambda[i]) <-  (
                awayGoalAvg +
                (
                  attack[team[i, 2], timeslice[i, 2]] -
                  defense[team[i, 1], timeslice[i, 1]] +
                  gamma * delta[i]
                )
              )

    goalsScored[i, 1] ~ dpois( homeLambda[i] )
    goalsScored[i, 2] ~ dpois( awayLambda[i] )

  }

}
```

Appendix B

Appendix C: JAGS model v2.0

```
### TBM v2.0
#####################################################

data {
  # Set C^(x) and C^(y) as constants
  # Values reported by Rue & Salvesen[2000]
  homeGoalAvg <- 0.395
  awayGoalAvg <- 0.098

  # Small constant to avoid sampler getting stuck
  C <- 0.0001

  # Scaling-specific parameter for beta distributions
  S    <-  10

  # Loss of memory effect for beta distributions
  eps   <- 3
  rho   <- 10
}

model {
  ### Time model
  ################################################

  tau     ~  dgamma(10, 1)
```

```
precision  ~  dgamma(10, 0.1)

# Loop through all teams and all timeslices
for(t in 1:noTeams) {
  # Initial distribution of attack/defense strength
  attack[t, 1]  ~  dnorm(0, precision)
  defense[t, 1]  ~  dnorm(0, precision)

  # Initial distribution of other skill parameters
  # Add C to avoid sampler getting stuck at inf.
  # Truncate the distributions to avoid [0, 1] - <0,1> is much nicer.

  passSuccess[t, 1]  ~  dbeta(
                0.7 * S,
                0.3 * S
              )T(C, 1-C)

  aerialSuccess[t, 1]  ~  dbeta(
                0.5 * S,
                0.5 * S
              )T(C, 1-C)

  contestSuccess[t, 1] ~ dbeta(
                0.5 * S,
                0.5 * S
              )T(C, 1-C)

  possession[t, 1]  ~  dbeta(
                0.5 * S,
                0.5 * S
              )T(C, 1-C)

  attempts[t, 1]    ~  dbeta(
                0.5 * S,
                0.5 * S
              )T(C, 1-C)

  saves[t, 1]       ~  dbeta(
                0.7 * S,
                0.3 * S
              )T(C, 1-C)

  # Evolve the attack/defense strength over time with Brownian motion
      (Wiener process).
```

```
## The parameters are normally distributed with mean equal to the
    parameter value of
## the previous timeslice (round). Loss-of-memory effect provided by
    variance parameter.
## Implements a custom scaling tecnique for beta-distribution to
    mimic Brownian motion
for(s in 2:noTimeslices) {

  motion[t, s] <- ((abs( days[t,s] - days[t,s-1] )) / tau) * precision

  attack[t, s] ~ dnorm(
            attack[t, (s-1)],
            motion[t, s]
          )

  defense[t, s] ~ dnorm(
            defense[t, (s-1)],
            motion[t, s]
          )

  betaScale[t, s]  <-  S - (((abs(days[t, s] - days[t, (s-1)])) / rho)
      * eps)

  passSuccess[t, s]  ~  dbeta(
                (passSuccess[t, (s-1)] + C) * betaScale[t, s],
                ((1 - passSuccess[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)

  aerialSuccess[t, s]  ~  dbeta(
                (aerialSuccess[t, (s-1)] + C) * betaScale[t, s],
                ((1 - aerialSuccess[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)

  contestSuccess[t, s] ~ dbeta(
                (contestSuccess[t, (s-1)] + C) * betaScale[t, s],
                ((1 - contestSuccess[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)

  possession[t, s]  ~  dbeta(
                (possession[t, (s-1)] + C) * betaScale[t, s],
                ((1 - possession[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)

  attempts[t, s]    ~  dbeta(
                (attempts[t, (s-1)] + C) * betaScale[t, s],
```

```
                ((1 - attempts[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)

    saves[t, s]      ~ dbeta(
                (saves[t, (s-1)] + C) * betaScale[t, s],
                ((1 - saves[t, (s-1)]) + C) * betaScale[t, s]
              )T(C, 1-C)
  }
}

### Goal model
################################################

# Params:
## attack[t, s] / defense[t, s] = attack/defense strength for team t at
    given timeslice s
## team[i, 1] = home team in game i
## team[i, 2] = away team in game i
## timeslice[i, 1] = timeslice(round/games played so far) of home team
    in game i
## timeslice[i, 2] = timeslice(round/games played so far) of away team
    in game i
## goalsScored[i, 1] = goals scored by home team in game i
## goalsScored[i, 2] = goals scored by away team in game i

# Give the delta-parameter some room to move
gamma ~ dunif(0, 0.1)

# Scalers for limiting the impact of att/def-skill
attSkillScaler ~ dunif(0, 0.25)
defSkillScaler ~ dunif(0, 0.25)

# Loop through all games in correct order
for(i in 1:noGames) {
  # Home team open play skill
  playSkillDiff[i] <- (
              (   passSuccess[team[i, 1], timeslice[i, 1]]   -
                  passSuccess[team[i, 2], timeslice[i, 2]] ) +
              (   aerialSuccess[team[i, 1], timeslice[i, 1]] -
                  aerialSuccess[team[i, 2], timeslice[i, 2]] ) +
              (   contestSuccess[team[i, 1], timeslice[i, 1]] -
                  contestSuccess[team[i, 2], timeslice[i, 2]] ) +
              (   possession[team[i, 1], timeslice[i, 1]]    -
                  possession[team[i, 2], timeslice[i, 2]]   )
            )
```

```
# Home team attack-specific skill
hAttSkill[i] <- (
          playSkillDiff[i] +
          (attempts[team[i, 1], timeslice[i, 1]] - saves[team[i, 2],
              timeslice[i, 2]])
        ) / 5
# Home team defense-specific skill
hDefSkill[i] <- (
          playSkillDiff[i] +
          (saves[team[i, 1], timeslice[i, 1]] - attempts[team[i, 2],
              timeslice[i, 2]])
        ) / 5

# Away team attack specific skill
aAttSkill[i] <- 0 - hDefSkill[i]

# Away team defense specific skill
aDefSkill[i] <- 0 - hAttSkill[i]

# Attack/defense node mutators
wHomeAtt[i] <- (hAttSkill[i] * attSkillScaler) + attack[team[i, 1],
    timeslice[i, 1]]
wHomeDef[i] <- (hDefSkill[i] * defSkillScaler) + defense[team[i, 1],
    timeslice[i, 1]]

wAwayAtt[i] <- (aAttSkill[i] * attSkillScaler) + attack[team[i, 2],
    timeslice[i, 2]]
wAwayDef[i] <- (aDefSkill[i] * defSkillScaler) + defense[team[i, 2],
    timeslice[i, 2]]

# delta param for psychological effect of underestimating
delta[i] <- (
         wHomeAtt[i] + wHomeDef[i] -
         wAwayAtt[i] - wAwayDef[i]
       ) / 2

# Home lambda parameter
homeLambda[i] <- exp(
             homeGoalAvg +
             (
               wHomeAtt[i] -
               wAwayDef[i] -
               gamma * delta[i]
             )
```

```
            )

   # Away lambda parameter
   awayLambda[i] <- exp(
               awayGoalAvg +
               (
                 wAwayAtt[i] -
                 wHomeDef[i] +
                 gamma * delta[i]
               )
             )

   goalsScored[i, 1] ~ dpois( homeLambda[i] )
   goalsScored[i, 2] ~ dpois( awayLambda[i] )

  }

}
```