Imperial College London
Department of Computing

MSci Joint Mathematics and Computer Science
Individual Project Final Report

Spotting The Wisdom In The Crowds

Liam Williams
ldw08@ic.ac.uk

Supervisor:
Dr. William Knottenbelt

Second Marker:
Dr. Giuliano Casale

June 21, 2012

Abstract

The growing popularity of websites on which hundreds of "tipsters" provide sporting tips has resulted in a large influx of sports betting advice in the public domain. But how do we know which of these tips are "reliable" and which should not be trusted? How do we spot the wisdom which may be hiding in the crowds?

The aim of this project is to investigate to what extent it is possible to extract such reliable advice from a large group of tipsters by observing the historical betting patterns of the population. Previous research into the performance of sporting tipsters has been limited to much smaller population sizes, for example those with regular newspaper columns.

We consider a simulated world, which is used to investigate a variety of potential scenarios regarding the behaviour of the tipsters in the population. In this controlled environment, we are able to develop and validate methods for detecting and extracting the wisdom. We then look at a particular real world tipping website which considers only the tennis win market. Applying the techniques developed whilst investigating the simulated world, we find that although high precision rates are achievable, it is much more of a challenge to obtain a significant positive return on investment.

Acknowledgements

I would like to thank my supervisor Dr. William Knottenbelt for his support and guidance over the course of the project, as well as his level-headedness on the numerous occasions I exclaimed "I've cracked it, we're rich!" only to find out that something was amiss.

Thanks also go to the staff at Active Capital Management, whose initial ideas and suggestions helped to give the project the kick start it needed.

Finally I would like to thank my friends and especially my family for their patience and support these past four years and without whom I would not be where I am today.

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, we will look at the problems considered in the project and the approach taken to tackling them. Section 1.1 gives an overview of what the scope of the project is and why it is worth undertaking, as well as the challenges we face in arriving at a solution. Section 1.2 lists the main contributions of the project and finally, Section 1.3 provides a brief summary of the report organisation.

1.1 Motivation

Before the advent of the Internet, perhaps the most common public source for sporting tips was that offered by professional tipsters published in newspapers. In recent years, websites on which "expert" tipsters provide public tips have grown in popularity, see e.g. TennisInsight[1] and OLBG[2]. As a result of this, there is a much greater volume of tips in the public domain than there has been in the past.

The question remains: is it possible to tell which of these tips are "reliable" and which are not; is it possible to spot the wisdom which may be hiding in the crowds?

The main aim of this project is to perform an analysis of the sporting tips given on such tipping websites to investigate how well tipsters perform when considered individually and also as a group. Additionally, we develop a "meta-tipster" which aims to identify correlations in tipsters betting patterns which are then used to provide a classification for future tips.

We begin by constructing a simulated world, where some of the desirable properties of the real-world data are artificially made present, for example giving a subset of tipsters a consistent edge in predicting events. This is done to show that, if such phenomena were to occur, it would be possible to pick up on this information by considering the betting patterns of the tipsters.

Ultimately, any new tip offered by the tipsters is to be classified according to whether the prediction is expected to become a reality or not. We consider several approaches

[1] `http://www.tennisinsight.com/`
[2] `http://www.olbg.com/`

to making this binary classification, using techniques from the areas of statistics and machine learning.

We consider several different configurations of the simulated world and evaluate the performance of the different approaches to classification by the meta-tipster to identify which kind of classifier is most appropriate given the underlying structure of the data.

Performance is evaluated against a naive majority voting approach and selection using the bookmakers implied odds to select the most probable outcome of an event according to the bookmakers, or "favourite" selection. Outperforming favourite selection may indicate that, collectively, the tipsters have private information that the public odds information alone does not encompass.

The performance of the meta-tipster is then evaluated using historical data collected from publicly available tip histories on TennisInsight. The TennisInsight data set comprises of approximately half a million tips made between May 2008 and December 2011 by approximately two thousand tipsters.

The period of time over which to consider historical data for training a classifier is also considered; tipster's models are subject to change in a real world situation, so looking at the entire past year's data may not be a good idea if tipsters tend to adapt their models more frequently than this. In practise, one might expect to have to periodically re-train classifiers to keep up with recent trends. Another point is that not all tipsters will stay active on a site indefinitely; there is no point considering tipsters who are no longer providing new tips.

Finally we examine the potential return on investment (ROI) that could have been achieved by following the advice of each of the classifiers over a period of time spanning 2010 and 2011. Several stake sizing strategies are considered, including a Kelly betting strategy, flat stake and crowd sourcing the amount to stake.

1.2 Contributions

The main contributions of the project are:

- A simulated world which can be used to generate different populations of tipsters which tip events with markets that have a finite number of possible outcomes.
- Using the simulated world to examine the effect deviations from a baseline population have on the predictive ability of the various classifiers considered.
- An investigation into the real-world TennisInsight data set, including making use of historical data to evaluate the performance of the meta-tipster and leading on to a final evaluation which looks at the potential return on investment which could have been achieved, had the advice of the meta-tipster been followed.

1.3 Report Organization

The report is set out as follows:

Chapter 2 provides **background** information on the tools and techniques used, as well as considering existing **related work** in the context of the project.

Chapter 3 goes into some of the details of the **implementation**, including data collection and storage, the simulated world and use of existing technology to form the various classifiers.

Chapter 4 investigates the **simulated world**, which provides a controlled environment for finding out which kinds of situation make it possible to spot the wisdom. Starting with a baseline population, which is used as a frame of reference when evaluating the performance of the meta-tipster, we make a number of changes to the baseline population and evaluate their effect on the predictive performance of the meta-tipster.

Chapter 5 investigates the **real world** data obtained from TennisInsight, first attempting to get an idea of the typical behaviour of the population of tipsters. We then apply the meta-tipster approach to the real world data, considering the effect of different "look-back" period lengths and then performing a **final evaluation** of both the predictive performance of the meta-tipster and the potential return on investment that could have been achieved.

Chapter 6 provides a **discussion** of the main findings of the project, drawing comparisons between the simulated populations considered and the real world data. Finally, we suggest a plan for possible **future work**.

2

Background

This chapter contains the relevant background information that was considered to arrive at a solution. Section 2.1 discusses previous related work in the area of forecasting sporting events and how this project will cover new ground. Section 2.2 provides a brief reference for some terminology which will be used throughout the report. In Section 2.3, several possible problem representations considered, as well as appropriate normalisation methods. Section 2.4 explains the approach taken in constructing the simulated world. Section 2.5 gives background information on the tipping websites which were considered as sources of data. Section 2.6 discusses the approaches for data storage that were considered. Section 2.7 considers different ways of achieving dimensionality reduction. Section 2.8 explores several different types of binary classifier. Finally, a number of different stake sizing strategies are considered in Section 2.9.

2.1 Related Work

Previous work in assessing the performance of experts in the context of forecasting sporting events was found to be limited. The general consensus for judgemental forecasting appears to be that "in nearly all cases where the data can be quantified, the predictions of the [statistical] models are superior to those of the expert"[23], apart from the case when "broken-leg cues"[1] are available to the expert.

Forrest and Simmons[15] examined the performance of newspaper football tipsters when forecasting the results of English league matches; their analysis was limited to just three tipsters. The tipsters in this case were found, individually, to perform poorly; even the simple strategy of always picking the home team (which has been shown to have an advantage over the away team previously[8]) was found to yield a better strike rate, however all of them demonstrated some underlying knowledge of football above just picking results at random.

Considering each tipster's forecasts individually against a simple statistical model constructed using public information, only one of the three tipsters was found to be making

[1] A broken-leg cue refers to an unusual important piece of information whose presence would dramatically alter judgement compared to a model of that judgement[38]

use of some form of private or semi-public information. However, when combining the tipsters predictions and taking a consensus (i.e. when two or more tipsters agreed), the conclusion was that the consensus forecast *did* add some explanatory power above just using the public variables.

In later work by Forrest et al[14], the performance of the odds-setters (i.e. bookmakers) as forecasters is considered against a benchmark statistical model, concluding that *all* of the British bookmakers considered appear to individually be privy to, and make effective use of, information not included in the benchmark model.

Andersson et al[2, 3] look at the performance and confidence of experts and laypeople in forecasting outcomes of the football World Cup, finding that both groups surveyed had similar levels of performance (which was better than chance) in predicting match outcomes. Unfortunately, a simple rule following world rankings was found to outperform the majority of participants. In predicting more complex outcomes such as full time scores and ball possession, however, the experts were found to outperform the naïve participants. In all cases, the experts were found to be significantly more confident about their forecasts than the laypeople.

Player name recognition has also been considered as a method in which lay predictions can be made. Pachur and Biele[27] find that a recognition heuristic agreed with lay forecasts for the European Football Championships 2004 in 90% of cases, suggesting that laypeople do tend to use player name recognition to make their predictions. The heuristic was found to perform significantly better than chance, but was outperformed by a model based on direct indicators of team strength.

Scheibehenne and Broder[32] look into player name recognition by laypeople for predicting the 2005 Wimbledon Gentlemen's tennis competition. Predictions based on this were found to perform at least as well as predictions based on official rankings, however online betting odds led to more accurate forecasts.

Bruno Deschamps and Olivier Gergaud[12] investigate the originality (i.e. excessive variation from the public information given their private information) of 35 professional French horse racing tipsters, finding that the tipsters were indeed more original than not. However, it is found that forecasts would have been more accurate if the tipsters had deviated less from the public information.

Something which does not appear to have been investigated previously is how the interactions between tipster's betting patterns may be a good indicator of the true outcome of an event. Consensus voting is a simple example of these kind of interactions, but it is suspected that there is more underlying structure due to the conflicting models which tipsters use to make their predictions, which is a further motivation for this investigation.

To illustrate, each tipster can be assumed to have a model which they use to choose which tips to bet on. This model may be backed by an actual statistical model or could be something less tangible, such as player name recognition or some other heuristic method.

Some of these models may completely dominate others, i.e. model A dominates model B if A predicts correctly in all of the cases which B does, as well as in additional cases where B does not predict correctly.

Certain models may be better at predicting different kinds of cases; one model may be good at predicting the "usual" continuous cases, but may completely fail when something unexpected happens, take the literal case of a player breaking their leg. Another model, from a tipster who pays more attention to such discontinuities would be the advice to follow in this case, however this advice may *not* be the popular opinion of all tipsters.

2.2 Terminology

Site
A tipping website (e.g. TennisInsight) where tipsters make tips.

Sport
A sport can be either a literal sport (e.g tennis) or a contrived sport (e.g. OLBG offers betting on "specials" such as the outcome of awards such as the Golden Globes).

Match
An instance of a sporting event (e.g. a single football match).

Market
A market for a particular sport consisting of several possible outcomes (referred to as "runners") which depend on the outcome of a particular match (e.g. a tennis match has a win market and a set betting market).

Runner
A possible outcome of a market. (e.g. for the tennis win market, the runners are the competing players and for the football full time result market, the runners are the competing teams and the draw outcome).

Tipster
An "expert" who makes tips on a site (e.g. a member of TennisInsight).

Tip
A selection of a particular runner for a certain market and match which a tipster is recommending. Includes the stake (i.e. amount of money bet) and odds taken by the tipster (odds are offered by bookmakers that are independent of the site the tip is placed on).

Selection
A selection refers to the choice of a single runner for a particular market and match. The term is used since different tipsters can make tips for the same selection.

2.3 Problem Representation

For each tip offered by a tipster for a particular match, market and runner (referred to as a selection), there is data available on stake size and odds taken by each tipster. This can be represented as the feature vector:

$$\begin{pmatrix} s_1 & o_1 & \dots & s_n & o_n \end{pmatrix}$$

where n is the number of tipsters, o_j represents the odds taken and s_j the stake laid out by tipster j.

As an example, suppose we have two tipsters, Alice and Bob, who have each placed a stake of 10 units on a particular player to win a tennis match, however Alice has taken odds of 1.2 and Bob has taken odds of 1.1. Additionally a third tipster, Charlie, is also considered, but did not happen to place any bets at all for this particular match. Then the corresponding feature vector is:

$$\overbrace{\begin{pmatrix}10 & 1.2\end{pmatrix}}^{\text{Alice}} \overbrace{\begin{pmatrix}10 & 1.1\end{pmatrix}}^{\text{Bob}} \overbrace{\begin{pmatrix}0 & 0\end{pmatrix}}^{\text{Charlie}}$$

For a collection of m selections, the corresponding feature vector for each selection can be written as the rows of the matrix:

$$\begin{pmatrix} s_{11} & o_{11} & \dots & s_{1n} & o_{1n} \\ \vdots & \vdots & & \vdots & \vdots \\ s_{m1} & o_{m1} & \dots & s_{mn} & o_{mn} \end{pmatrix}$$

Here, o_{ij} represents the odds taken and s_{ij} the stake laid out by tipster j for selection i (with $o_{ij} = s_{ij} = 0$ if the tipster did not tip this selection).

Each selection has a class associated with it, corresponding to whether the result of the associated match resulted in this selection being the correct result for the associated market or not.

It is possible that the number of tipsters to be considered will be large, perhaps thousands, which means a large number of dimensions for the feature vectors.

This potentially high dimensional space is problematic. This is due to both practical reasons (e.g. computation begins to become intractable) and also because, since it is possible that $m \ll n$, such that the amount of available data will be sparse, the so-called "curse of dimensionality". Possible ways to reduce the dimensionality of the problem are discussed in Section 2.7.

2.3.1 Alternatives

Clearly, this is not the only way to formulate the problem and several variations of this model will be considered.

The stake size and odds taken are considered important since it seems likely that tipsters will "bet their beliefs" and size their bets according to the edge that they consider to have over the odds offered by the bookmaker.

It may not only be important to consider the details of who did or did not support a particular selection, but also, in the case that a particular tipster did *not* support the selection, if they instead supported a different selection or perhaps supported none at all.

Note that tipsters are assumed to only be "allowed" to support one selection per match and market (something which is enforced on TennisInsight and OLBG).

In light of this, the original feature vector can be extended:

$$\begin{pmatrix} s_1 & o_1 & s_1^* & o_1^* & \dots & s_n & o_n & s_n^* & o_n^* \end{pmatrix}$$

where the additional variables s_i^* and o_i^* represent the stake laid out and odds taken by the tipster if they supported a different runner for the same match and market as the selection in question, with $s_i^* = o_i^* = 0$ if the tipster did not support a different runner.

So if $s_i = o_i = s_i^* = o_i^* = 0$ for a particular tipster, this represents that they did not have a tip for this match and market. Representations which include this type of information will be referred to as "and not" versions.

This representation makes more sense if we are considering markets with more than two possible outcomes. Suppose there were three possible outcomes for the market that Alice and Bob had bet on, then the corresponding feature vectors for all three selections would be:

$$\begin{array}{c} \overbrace{}^{\text{Alice}}\ \overbrace{}^{\text{Bob}}\ \overbrace{}^{\text{Charlie}} \\ \begin{pmatrix} 10 & 1.2 & 0 & 0 & 10 & 1.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 1.2 & 0 & 0 & 10 & 1.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 1.2 & 0 & 0 & 10 & 1.1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

The benefit of this is that there is now a distinction between not betting on anything at all (corresponding to all zeros for that tipster as in the case of Charlie) and betting on something which was *not* this (see e.g. the second row for Alice).

Another representation is to simply have a binary indicator variable, corresponding to whether or not the tipster supported the selection, instead of the stake size and odds variables; the feature vector is just:

$$\begin{pmatrix} t_1 & t_1^* & \dots & t_n & t_n^* \end{pmatrix}$$

where $t_i = 1$ if the tipster did support the selection and $t_i^* = 1$ if the tipster supported a different selection. If $t_i = t_i^* = 0$, this again represents that they did not have a tip for this match and market.

A benefit of all of the representations considered is that they are all able to cope with markets containing any number of outcomes and even a variable number of outcomes (which could be useful in representing, say, horse racing markets which have a variable number of runners per race).

2.3.2 Normalisation

Clearly, each of the features are not all the in the same units (e.g odds, stake size) and so appropriate normalisation of each feature can be carried out as a pre-processing step.

One way to do this is to mean centre and scale to unit variance by dividing by the empirical mean and standard deviation. A problem with this is that, since the data will be sparse (i.e. it is unreasonable to expect that *all* of the tipsters will provide a tip for *every* match), this transformation will destroy the sparsity since each feature will typically have a different normalisation constant.

Another way to handle this in the case of stake sizes is to represent the stake size as a percentage of the previously observed maximum stake size for the tipster in question, which does preserve sparsity and normalises stake size to the range $[0, 1]$.

Odds can also can be normalised to the range $[0, 1]$ by representing them using the corresponding implied probabilities (see Appendix A.1.4).

As an example, returning to the basic stake and odds representation with the Alice and Bob example, the feature vector

$$\big(\overbrace{10 \quad 1.2}^{\text{Alice}} \quad \overbrace{10 \quad 1.1}^{\text{Bob}}\big)$$

might be normalised to

$$\big(\overbrace{\tfrac{10}{100} \quad \tfrac{1}{1.2}}^{\text{Alice}} \quad \overbrace{\tfrac{10}{10} \quad \tfrac{1}{1.1}}^{\text{Bob}}\big)$$

if the maximum stake size for Alice were 100 units and the maximum stake size for Bob were 10 units.

2.4 Simulated World

To construct the simulated world, we need three main components: the matches, the odds setter and the population of tipsters who have the opportunity to place bets on the markets associated with these matches according to the odds set by the odds setter.

2.4.1 Generating "True" Probabilities

Given a match which has a market with n runners, the distribution of winning runners can be said to be categorically distributed, where the probability of runner x_i winning is p_i, constrained by:

$$\sum_{i=1}^{n} p_i = 1$$

The "true" probabilities p_i for each of the n runners are generated by sampling $\mu_i \in [0, 1]$ from the uniform distribution on $[0, 1]$, and then constructing the corresponding p_i as:

$$p_i = \frac{\mu_i}{\sum_{j=1}^{n} \mu_j}$$

For example, with $n = 2$, if μ_1, μ_2 are randomly selected as $\mu_1 = 0.4, \mu_2 = 0.8$ then $p_1 = \frac{0.4}{0.4+0.8} \approx 0.33$ and $p_2 = \frac{0.8}{0.4+0.8} \approx 0.66$.

2.4.2 Estimation Strategies

A tipster has estimates $\hat{p}_i$ (summing to 1) which they hope correspond to the true p_i (i.e. $|\hat{p}_i - p_i| \in [0, 1]$ is small for all i). The process of finding these $\hat{p}_i$ is referred to as an estimation strategy.

Random

A random estimation strategy forms estimates $\hat{p}_i$ using the same method in which the "true" probabilities were distributed, so that $\hat{p}_i$ can vary completely from the true p_i.

Smart

Tipsters can be artificially instilled with wisdom such that their estimates do not differ a great deal from the true probabilities of each runner in the market.

To introduce this error, each true probability p_i is adjusted by adding a random value r_i sampled from the uniform distribution on $[-rp_i, r(1-p_i)]$ for a fixed $r \in [0,1]$. The resulting adjusted probabilities will not sum to 1, so the estimates $\hat{p}_i$ are taken as:

$$\hat{p}_i = \frac{p_i + r_i}{\sum_{j=1}^{n}(p_j + r_j)}$$

Now r can be used to change how reliable the estimates are; increasing r allows the estimates to vary more from the true probabilities, where at $r = 1$ each of the probabilities have a chance of being adjusted to estimates wildly different from the true probabilities, whilst at $r = 0$, $\hat{p}_i = p_i$ and the estimations are perfect. This is illustrated in Figure 2.1.

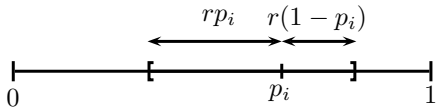

Figure 2.1: Smart Probability Adjustment

2.4.3 Specialist Tipsters

To model the idea that tipsters may have a certain type of match that they are a "specialist" on, sets of tipsters, T_i, are assigned a subset of all matches that they will be "smart" on, M_i, which is disjoint from all other sets of matches assigned to the other sets of tipsters (i.e. $M_i \cap M_j = \emptyset \; \forall i \neq j$).

For all matches $m \notin M_i$, which the tipsters T_i are not specialists on, the tipsters may either decide to simply behave randomly, or alternatively, if they are aware that they are not a specialist on a particular match, may simply abstain from tipping that match.

Variations of this can be considered, for example where the sets of assigned matches M_i are no longer disjoint and so there is some overlap in the matches which the groups of tipsters are specialists for.

2.4.4 Copycat Tipsters

Some groups of tipsters will follow the advice of a "pack leader" and copy the leader's actions, regardless of whether the leader's decision happens to be a good one or a bad one. This can be used to model the idea that a popular opinion may not always be the

correct one. Having a large number of copycat tipsters may have a significant influence on the consensus of the population.

2.4.5 Odds Offered

Tipsters are offered odds by an odds-setter, who uses a smart estimation strategy with (typically) small r to set these odds. The odds set are taken as the reciprocal of the probability estimates (the estimates correspond to the implied probabilities which can be calculated from offered odds and may include over-round in a real world situation, see Appendix A.2.2), denoted $\tilde{p}_i$. Here we assume no over-round is introduced.

2.4.6 Selection Strategies

Given that a simulated tipster has a set of estimates $\hat{p}_i$ for each runner x_i, they can now use these to decide which runner to select, if any.

Most Edge

The tipster can calculate the discrepancy d_i between their estimate $\hat{p}_i$ and the implied probability of the odds offered by the odds-setter $\tilde{p}_i$:

$$d_i = \hat{p}_i - \tilde{p}_i \in [-1, 1]$$

Positive discrepancy indicates that the tipster's estimate of the runner winning is greater than the odds-setter's estimate; in other words, the tipster is more confident than the odds-setter that this runner will win. A positive discrepancy of 1 would indicate that the tipster has estimated that this runner is certain to win, whilst the odds-setter has estimated that they are certain to lose.

Similarly, negative discrepancy indicates that the tipster is *less* confident than the odds-setter that this runner will win. A negative discrepancy of -1 would indicate that the tipster has estimated that this runner is certain to lose, whilst the odds-setter has estimated that they are certain to win.

It is therefore in the best interests of the tipster to only make a selection when there exists a runner x_i with $d_i > d_{min}$, where $d_{min} \in (0, 1]$ is the minimum positive edge the tipster is prepared to place a bet on. When such a runner does exist, if there are multiple such runners, the one with the greatest discrepancy value is chosen.

Ignore The Market

Alternatively, a tipster may decide to take no notice of the implied probability of the odds offered and instead simply choose the runner with the largest predicted probability of winning $\hat{p}_i$. In this case we will take $d_i = \hat{p}_i$ for simplicity.

2.4.7 Stake Sizing Strategies

Given that a tipster has selected a runner x_i with corresponding discrepancy $d_i > d_{min}$, they must now decide how much to stake on this runner. The process of deciding on a stake size is referred to as a stake sizing strategy.

Fixed

The tipster always bets the same amount, say B, regardless of the edge they consider they have.

Scaled

The tipster bets an amount $f(d_i) \cdot B_{max}$ where f is monotonically increasing on $[0, 1]$.

For example, $f(d_i) = d_i^{\lambda}$ could be used for some fixed $\lambda \geq 1$.

When $\lambda = 1$ we will refer to this as a "proportional" stake sizing strategy.

2.4.8 Experiments

By combining tipsters with different underlying estimation, selection and stake sizing strategies, different experiments can be set up to simulate data from different populations of tipsters. Chapter 4 goes on to investigate the effects changes in the population have on being able to spot the wisdom in the crowds.

2.5 Real World Data

Two tipping websites were considered as a source of real world data for the project, TennisInsight and OLBG. Ultimately only TennisInsight data was used for analysis, due to problems in collecting data from OLBG, discussed in Section 3.3.

2.5.1 TennisInsight

TennisInsight allows tipsters to tip on the win market for professional tennis matches using an unlimited amount of virtual money. Odds are offered by tracking the best odds offered by various bookmakers.

The site provides specialist information in the form of variety of statistics for the tennis players (i.e. indicators of their past performance) which is only accessible to paid subscribers.

Current tips made by tipsters are also only visible by paid subscribers, with the exception of the next upcoming match. Tip histories from 2008 to the present are publicly available for all non-current matches.

Leader boards for tipsters recent performance exist (e.g. ROI, strike rate, amount earned), which could be seen as a source of motivation for the tipsters.

The investigation into the TennisInsight data can be found in Chapter 5.

2.5.2 OLBG

OLBG allows tipsters to tip on a variety of markets for a number of different sports, such as tennis, football, rugby, darts, and horse racing. Like TennisInsight, odds are offered by tracking the best odds offered by various bookmakers. Tipsters have a limited amount of virtual money, which is supplemented periodically to represent income.

There is no form of paid subscription. Tip histories are publicly available for the past 45 days only. No form of specialist information is given for the various sports.

The site offers "free" bets which are sponsored by bookmakers and cash prizes in monthly competitions as incentives for tipsters to perform well.

2.6 Data Storage

2.6.1 Relational Databases

A relational database can be used to store data as a "set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables"[2] .

In the context of the data that will be simulated or collected, the relational database is an appropriate storage solution since the data will be analysed in a variety of different (and potentially unforeseen) ways, which will require querying the data in different ways. If a flat file approach to storing the data were taken, it would be more difficult to achieve this, whereas a database solution keeps things flexible.

2.6.2 Serialization

Serialization in the context of object oriented programming languages refers to the process of converting an in-memory object to a serialized form which can be persisted to a file (or even to a database) and read back into memory at a later date. This may be a suitable technique for caching transformed data which takes a significant amount of time to compute. For example this could be useful for persisting trained classifiers.

2.7 Dimensionality Reduction

It is possible that we will be dealing with datasets consisting of thousands of tipsters, corresponding to a very high dimensional feature space, which could be problematic

[2] `http://searchsqlserver.techtarget.com/definition/relational-database`

both in terms of computational feasibility and efficiency of the classifiers. As such, we consider several ways to reduce the dimensionality of the problem.

2.7.1 General Techniques

If there is not a lot of data for a particular feature, it may makes sense to exclude that feature. In this context, this would mean excluding tips by tipsters who have made less than some threshold number of tips.

2.7.2 Principal Component Analysis

Principal Component Analysis (PCA)[1, 20, 28] is a technique used for finding patterns in high dimensional data.

An orthogonal transformation is used to transform data to a new coordinate system which better explains the variance in the data, such that a small number of the basis vectors (or principal components) in the new coordinate system may explain the majority of the variance in the data. By projecting data onto the space spanned by a subset of the basis vectors, the dimensionality of the data can be reduced.

PCA can be a useful technique for visualising high dimensional data, by projecting data using just the first two or three principal components, but this should not be taken as an end in itself[41], but more as a guide for further investigation.

A strong assumption made by PCA is that large variances have important structure[34] which is something which may be incorrect in practice, depending on the particular problem (see Figure 2.2).

2.7.3 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA)[13, 16] is, similar to PCA, used to find patterns in high dimensional data.

Unlike PCA, the goal of LDA is to find a new coordinate system which best explains how to discriminate between two or more classes, where each data point is associated with a class. This can be useful when the discriminatory information is not contained in the variance of the data, but in the mean (see Figure 2.2).

A fundamental assumption of LDA is that the independent variables are normally distributed, which may be unreasonable to assume in this context, however it is still a technique worth investigating since the underlying structure of the data is not yet known.

It is important to note that, since we are dealing with a two class problem, the number of non-zero eigenvectors will be just one; we will be reducing the dimensionality of the data down to just one dimension, which may have limited applicability.

A more practical way to use LDA as a dimensionality reduction technique is considered, where the magnitude of the factor loadings is used to rank the importance of each of the original features and then the top L of these factors are kept, where L is the

dimensionality we wish to reduce to.

Details of LDA can be found in Appendix B.2.

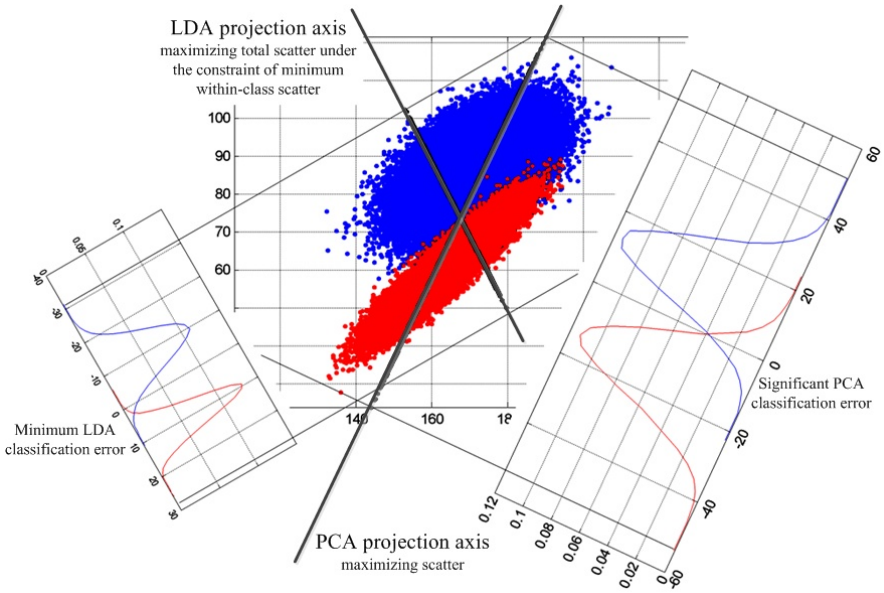

Figure 2.2: Comparison of PCA and LDA[3]

2.8 Binary Classification

The aim of a binary classifier in this context is to take a feature vector (see Section 2.3) and determine which class it belongs to. In other words, given a particular selection, the classifier will predict whether it expects the selection to become a reality (i.e. to win) or not by assigning it to one of these two classes.

2.8.1 Simple Majority Voting

A simple majority voting classifier counts the number of tipsters who supported a selection and compares this to the count of tipsters for all other selections for the same match and market. The classification is then based on whether the selection in question has the support of the majority of tipsters or not.

2.8.2 Favourite Strategy

Another simple strategy which only makes use of the public odds information is to simply consider whether the selection in question is the "favourite" or not and make the classification accordingly. The term favourite refers to the selection for a market which has the smallest odds amongst all other selections for that market.

[3] `http://www.ait.gr/ait_web_site/faculty/apne/Images/WhyLDA.jpg`

2.8.3 Linear Separability

Two sets of points in an n-dimensional space, for example points belonging to one of two classes, are linearly separable if they can be separated by a hyperplane in $(n-1)$ dimensions.

For example, in the two dimensional case, this corresponds to a single line which completely separates the two sets of points.

Clearly, it is not always the case that data will be linearly separable and so a variety of different classifiers are considered, some which attempt to linearly separate data and some that perform classification in a non-linear manner.

2.8.4 Linear Discriminant Analysis

Having performed LDA to obtain the coordinate system which best explains the separation of the classes, this space can now be used directly to classify new data points.

The data point to be classified, $\mathbf{x}$, is projected to the LDA space $\mathbf{x} \mapsto \hat{\mathbf{x}}$ and then the Euclidean distance between $\hat{\mathbf{x}}$ and each projected mean point $\hat{\mathbf{x}}_c$ for each class c, is computed.

The point $\mathbf{x}$ is then assigned the class c which minimises the distance $\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_c\|$.

2.8.5 Naive Bayes

The Naive Bayes classifier attempts to model the probability distribution over the class variable $P(C \mid T_1, \ldots, T_N)$ by considering the conditional probabilities of $P(T_i \mid C)$ for the feature variables T_i, which can be estimated using the sample frequencies observed in the training data.

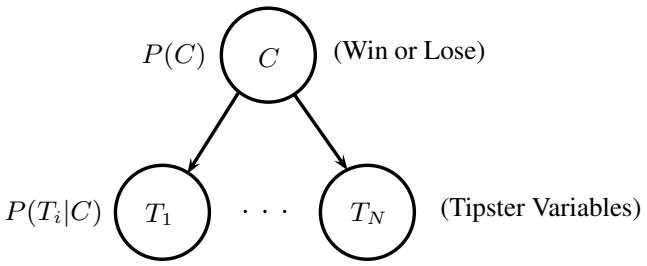

Figure 2.3: Naive Bayes Classifier

The idea here is that, given a selection is destined to win or lose, this will influence the betting behaviour of the tipsters, as shown in Figure 2.3. But we get to observe this behaviour *before* the match has taken place; it may be possible to predict whether the selection will win or lose using historical information on how the betting behaviour varies given the true classification.

The tipster variables (e.g. stake) must be categorical here, so we will need to decide on how many "bins" to quantise our normalised features to. This will be determined using

cross validation on the training set as discussed in Section 2.8.12.

A more in depth discussion of the method can be found in Section B.1.

2.8.6 Logistic Regression

Also known as the logit model, logistic regression[17, 29] is a regression technique which can be used to predict the probability of an event occurring by fitting explanatory variables to the logistic function:

$$f(y) = \frac{e^y}{e^y + 1} = \frac{1}{1 + e^{-y}}$$

where

$$y = \beta_0 + \sum_{i=1}^{k} \beta_k x_k.$$

Here, β_0 is the "intercept" and $\beta_1 \dots \beta_k$ are the regression coefficients of $x_1 \dots x_k$. The x_i represent the explanatory variables of the model such that $f(y)$ is the probability of the dependent event occurring. In this case, the x_i are the features corresponding to tipster's support or non support for a particular selection and $f(y)$ is the probability that the selection wins.

The β_i can be estimated using the standard maximum likelihood methods used to solve generalized linear models. These estimates can be computed numerically by using iteratively re-weighted least squares, which makes use of a Newton-Raphson iterative optimization.

Having computed the coefficients β_i, the logistic model can now be used for classification. The point to be classified, $\mathbf{x}$, is used to compute the probability of the event occurring, $f(y)$. If $f(y) > 0.5$, $\mathbf{x}$ is assigned to the class corresponding to the event occurring and otherwise $\mathbf{x}$ is assigned to the class corresponding to the event not occurring.

A benefit of having this probability is that, since it can be thought of as the confidence the model has that the point belongs to the class, a more paranoid approach could be taken for classification by considering $f(y) > p_{min}$ for some $p_{min} \in (0.5, 1]$.

Logistic regression is also able to work with discrete and categorical data, unlike LDA[30] since it makes no assumptions about the underlying distribution of the explanatory variables. This means that, for example, using indicator variables as discussed in Section 2.3.1 is a possibility.

Model Construction

Likelihood ratio tests can be used to determine whether a restricted model with maximised likelihood function L_R is significantly different than the corresponding full model with maximised likelihood function L_F using the test statistic:

$$-2 \log \left(\frac{L_R}{L_F} \right) = -2 \left[\log(L_R) - \log(L_F) \right]$$

which is χ^2 distributed with degrees of freedom equal to the number of variables which need to be restricted in the full model to form the restricted model.

By performing these tests, it may be possible to reduce the dimensionality of the problem by restricting the full model to the variables which are significant in explaining the data, which may help the model when making generalisations about new data.

2.8.7 Support Vector Machines

In the linearly separable case, the support vector machine (SVM)[9,18] aims to minimise the margin $\|\mathbf{w}\|$ between the hyperplanes $\mathbf{w}.\mathbf{x} - b = -1$ and $\mathbf{w}.\mathbf{x} - b = 1$ (see Figure 2.4).

These hyperplanes are chosen such that $\mathbf{w}.\mathbf{x}_i - b \geq 1$ for all $\mathbf{x}_i$ belonging to the first class and $\mathbf{w}.\mathbf{x}_i - b \leq -1$ for all $\mathbf{x}_i$ of the second class. This can be rewritten as $y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1$ where $y_i = 1$ for points $\mathbf{x}_i$ of the first class and $y_i = -1$ for points $\mathbf{x}_i$ of the second class.

The hyperplanes do not depend on all training points but only on the so-called "support vectors" which lie on these hyperplanes.

Finding the weights $\mathbf{w}$ and b is a quadratic programming optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 \\ &\text{subject to} && y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1 \qquad \forall i = 1, \ldots, n \end{aligned}$$

In the case where data is non-linearly separable, additional terms $\xi_i > 0$ which measure the degree of misclassification of data points $\mathbf{x}_i$ can be added to introduce a "soft margin" which aims to split the examples as cleanly as possible: $y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1 - \xi_i$. This soft margin is also useful to prevent over fitting in the linearly separable case, since, a single outlier without using soft margins could determine the entire boundary.

This introduces a hyper-parameter which we will denote C, which is used to determine how much to penalise the error terms ξ_i, leading to the optimization problem:

$$\begin{aligned} &\text{minimize} && \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right\} \\ &\text{subject to} && y_i(\mathbf{w}.\mathbf{x}_i - b) \geq 1 \\ & && \xi_i \geq 0 \qquad\qquad \forall i = 1, \ldots, n \end{aligned}$$

which is again a quadratic programming problem.

Further details on how these quadratic programming problems are solved in practise can be found in the LIBSVM paper[7].

The hyper-parameter C will be learned using cross validation on the training set, as discussed in Section 2.8.12.

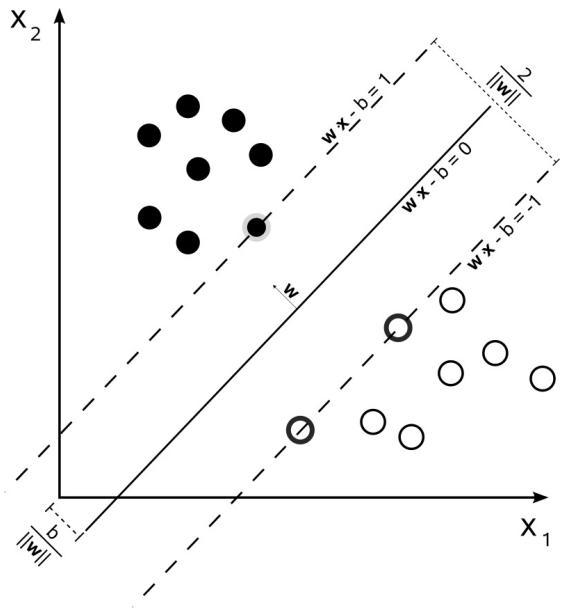

Figure 2.4: Linear Support Vector Machine[4]

2.8.8 K Nearest Neighbour Algorithm

The K nearest neighbour algorithm (KNN) makes classifications by considering the closest training examples in the feature space, according to some distance metric. The classification boundaries are non-linear and can identify multiple clusters of classes in the feature space.

Training examples are stored with the corresponding class label and classifications are made by considering the "closest" K neighbours to the test point and classifying the test point as the most frequently occurring class amongst these K neighbours. For the two class case, K can be chosen as an odd number so that there will always be a majority.

For example in Figure 2.5, the green circle test point is classified as a red triangle if $K = 3$ and as a blue square if $K = 5$.

Alternatives to the basic majority voting classification exist, such as weighting each neighbour according to its distance from the test point, which can help if the dataset is imbalanced (i.e. contains a large number of one class compared to the other classes).

Some disadvantages[10] are the poor run time performance if the training set is large and KNN is very sensitive to irrelevant or redundant features since *all* features contribute to the distance.

The value of K to use will be found using cross validation on the training set, as discussed in Section 2.8.12.

[4] `http://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`

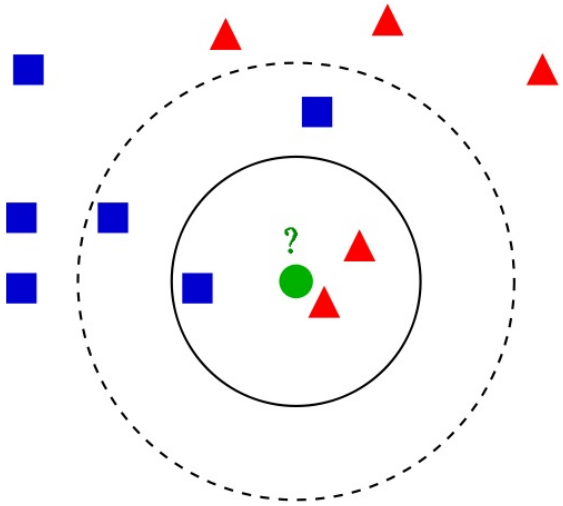

Figure 2.5: K Nearest Neighbour Classification[5]

2.8.9 Kernel Methods

In some cases, it may be possible to map data which is not linearly separable into a higher (possibly infinitely) dimensional space, where it *is* possible to linearly separate the data[6]. See for example Figure 2.6, which shows how a function ϕ is used to map data to a space where the data is linearly separable, using an SVM approach in this case.

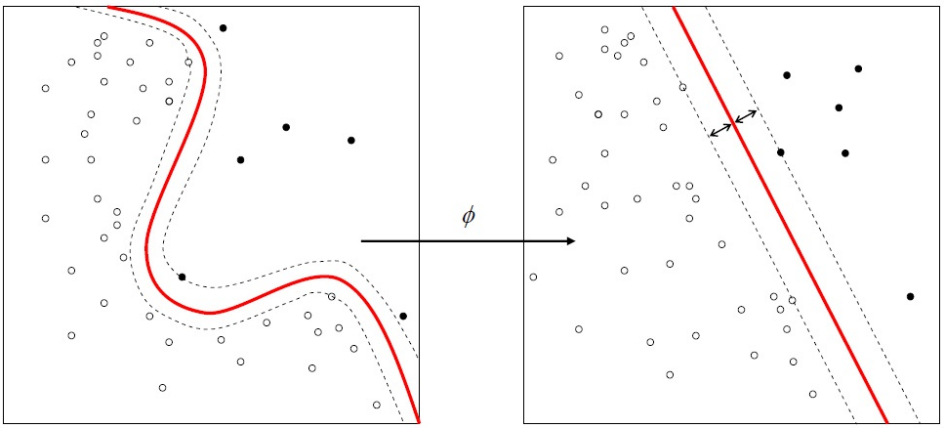

Figure 2.6: Non-linear Support Vector Machine[7]

It is possible to operate in this higher dimensional space without ever explicitly mapping the original data to the space by performing operations solely on inner products in the inner product space of this higher dimensional space, the so-called "kernel trick"[33].

There are a number of linear classifiers which can be adapted to take advantage of this to form non-linear classifiers, including LDA[24] and the SVM[9]. PCA[25] can also be

[6] http://www.youtube.com/watch?v=3liCbRZPrZA
[7] http://en.wikipedia.org/wiki/File:KnnClassification.svg
[7] http://en.wikipedia.org/wiki/File:Kernel_Machine.png

adapted to use this technique.

Unfortunately, the problem of finding a good kernel to use for a problem is non-trivial. A common initial choice of kernel is a Gaussian radial basis function:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

which has a single parameter $\gamma > 0$.

2.8.10 Conservative Classifiers

An important thing to note is that the problem formulation allows classifiers to predict that more than one selection will win. Clearly this is impossible and so in situations like this, we will classify both selections as lose. This means that the recall of the classifier will decrease, but the precision will almost certainly increase as a result, which is well suited to the problem since we are most interested in the precision. This conservative attitude can be thought of as a kind of sanity check each classifier makes - "do I agree with myself?".

2.8.11 Measuring Performance

Recall and Precision

In the case of the binary classifier predicting whether selections win or lose, there are four cases for the result of a classification, explained in Table 2.1.

Type	Symbol	Description
False Positive	FP	Predicted win, actual loss
False Negative	FN	Predicted loss, actual win
True Positive	TP	Predicted win, actual win
True Negative	TN	Predicted loss, actual loss

Table 2.1: Classification Cases

The precision and recall quantities[26] are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad \text{and} \quad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}.$$

Precision represents the proportion of all win classifications made that were actual wins.

Recall represents the proportion of all actual wins that were identified as wins by the classifier.

Perhaps the most important of these in a betting scenario is Precision, since it takes into account the false positive. The false positive is the only case where, if the advice of the classifier is taken, money is staked which will be lost. Precision of 1 corresponds to *all* of the win classifications made being actual wins, whereas Precision of 0 corresponds to *none* being actual wins.

However, Recall is also important since a higher Recall rate would correspond to a larger number of opportunities for winning bets to be placed.

Over-fitting

All of the classifiers considered use a training set to "learn" the structure of the data. Over-fitting refers to when the classifiers begin to become biased towards the training set; effectively learning the specifics of the random noise in the sample rather than learning a generalised model. As an extreme example, if number of parameters is the same as the number of examples, the classifier could end up "memorizing" the training data completely.

Equally, it is possible for a classifier to under-fit by being *too* general and not learning any of the important structure. This could happen if too few training examples were used for example.

As a result of this, rather than using a single training set and single testing set to evaluate performance, it makes more sense to use a technique such as cross validation to help avoid the possibility that classifiers may appear to be performing well by chance.

In cross validation, the entire set of examples is randomly split into N "folds". A single fold is then used as the testing set, whilst the remaining $(N-1)$ folds are used as the training set. This is repeated for each fold in turn and then the average of the performance measures (such as Precision and Recall) can be used to compare classifiers.

2.8.12 Learning Classifier Hyper-Parameters

Of the classifiers discussed, some have additional "tuning" parameters which must be chosen. For example, the linear SVM requires the soft margin parameter to be chosen.

One way to make effective use of data available is to perform an axillary cross validation step within the main cross validation loop. For instance, when performing N fold cross validation of a data set, for each of the N training sets, we can split this into M folds which are used to perform M fold cross validation for each of the possible hyper-parameter configurations. The best performing configuration is then trained on the whole M folds and the outer N fold cross validation proceeds as usual. This is shown in Figure 2.7.

The hyper-parameters to be determined for each classifier are shown in Table 2.2.

Classifier	**Parameters**
KNN	Number Of Neighbours
Linear SVM	Soft Margin
Kernel SVM	Soft Margin, Kernel Parameters
Naive Bayes	Number Of Bins

Table 2.2: Summary Of Classifier Hyper Parameters

2.8.13 Summary

Both linear (LDA, SVM, Naive Bayes) and non-linear (logistic regression, KNN, kernel LDA, kernel SVM) methods for binary classification have been considered since it is

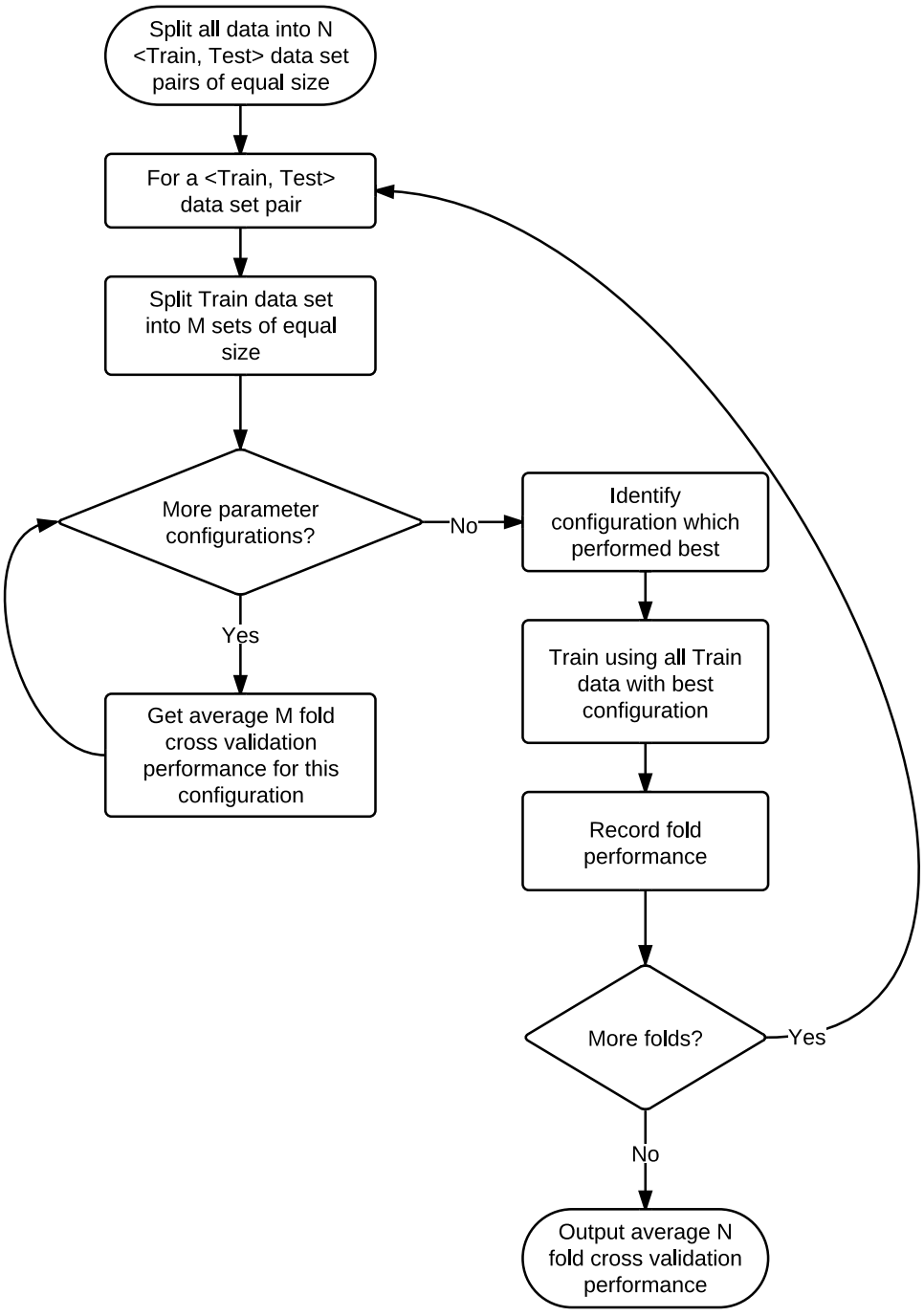

Figure 2.7: Auxiliary Cross Validation For Choosing Classifier Parameters

unclear which approach will perform best with the data being investigated.

The computational complexity of these methods for training and classification has not been taken into account at this stage. This is because the primary concern is to produce an accurate classifier which can be trained and classify new examples in a "reasonable" time, as opposed to having explicit upper bounds on the time that the classifier must

take to be trained and classify new examples. However, if this does become a problem with the empirical data which is to be investigated, the issue will be revisited.

One of the frameworks considered (see Section 3.1.2) has implementations for the majority of the classifiers mentioned and so it was practical to experiment with almost all of the classifiers, however ultimately I decided to focus my investigation towards six types of classifier: linear discriminant analysis (LDA), Naive Bayes (NB), logistic regression (LR), weighted K nearest neighbour (WKNN) using an inverse distance weighting, and finally two support vector machine variants, with a linear kernel (LSVM) and Gaussian kernel (GSVM).

Many other types of binary classifier exist that have not been discussed, for example neural networks, decision trees and mixture models. These methods were not looked into further since they either appeared to be less suited to the problem or are known to be prone to over-fitting.

2.9 Sizing Bets

Given we have a way of assigning a classification to each new selection, this begs the question - how much should I bet when the classification is positive? One option is to bet a fixed amount every time. Another option could be to follow the crowd's opinion on how much to bet, by using the average stake which was placed on the selection in question.

However, if we truly do have an edge over the odds being offered, there may be a more intelligent way to size our bets.

2.9.1 The Kelly Criterion

The Kelly criterion[21, 35] is a formula used to determine the optimal stake size in a series of bets. The strategy will perform better than any essentially different strategy in the long run, where the objective is increasing wealth.

Essentially, the strategy is equivalent to, given a choice of bets, to choose the one with the highest geometric mean of outcomes[4].

In the case of simple bets with two outcomes, either losing the stake amount or winning a profit according to the odds, the fraction of the current bankroll to stake is:

$$f^* = \frac{p(v+1)-1}{v}$$

where p is the win probability and v are the fractional odds being offered ("v to 1").

This can be thought of as "expected profit divided by profit if you win".

It is possible that f^* is negative (when the expected profit is negative) which means that no bet would be placed in this case.

Alternatives to the Kelly strategy include a fractional variant, where the proportion of the bankroll wagered is $\frac{f^*}{c}$ for some positive integer c. This can help reduce the short

term volatility[37] of the strategy and also protect against bad estimates being made for p, but at the cost of a reduced potential maximum return.

2.9.2 Return On Investment Simulations

Considering this, a novel way to look at the performance of the meta-tipster on the real world data could be to simulate a series of bets given an initial bankroll, where the classifier sizes its bets according to the (fractional) Kelly strategy.

This requires that the probability of winning is provided by the classifier, which is only available in the case of logistic regression and the Naive Bayes classifiers. For other classifiers, the probability of winning could simply be set to a fixed value, however this may not be very effective.

3

Implementation

In this chapter, details of the implementation side of the project are presented. Section 3.1 gives an overview of the various technologies used. Section 3.2 covers the approach taken to data storage and persistence. Section 3.3 explains how the real world data was collected. An overview of the simulated world system can be found in Section 3.4. Section 3.5 considers the implementation of the analysis side of the project and finally Section 3.6 discusses the approach taken to testing.

3.1 Technology Used

Throughout the course of the project I made use of a number of different technologies to facilitate the implementation. The following is a brief summary of how each was used.

3.1.1 Programming Language

The language I settled on to use was C# (C Sharp)[1]. I made this decision because I found several libraries I thought would be useful to use which happened to be written in C# and this was judged to be a better alternative than attempting to port such libraries to another language I was more familiar with.

I also made use of R[2] towards the end of the project, due to its extensive library of statistics packages and as a means to present data using graphs and charts.

3.1.2 The Accord.NET Framework

The Accord.NET framework[3] is a C# framework which extends the AForge.NET framework[4]. It contains implementations of many of the machine learning and statistics

[1] `http://msdn.microsoft.com/library/z1zx9t92`
[2] `http://www.r-project.org/`
[3] `http://accord-net.origo.ethz.ch/`
[4] `http://www.aforgenet.com/framework/`

techniques that were explored, including PCA, LDA, SVM, kernel variants, logistic regression and hypothesis testing.

3.1.3 Math.NET Numerics

The Math.NET Numerics C# library[5] contains implementations for a variety of different discrete and continuous distributions, some of which are not yet implemented in Accord.NET.

3.1.4 ALGLIB

The ALGLIB numerical analysis library[6] contains implementations for PCA, LDA and logistic regression. It was considered as a possible alternative to Accord.NET, however it does not have as much functionality, for example no kernel variants of the PCA and LDA algorithms.

3.1.5 Emgu CV

Emgu CV[7] is a C# wrapper for the OpenCV[8] image processing library. It was used as an alternative to the Accord.NET SVM implementation after it was found that it was poorly optimised. The SVM implementation in OpenCV is based on LIBSVM[9].

3.1.6 NHibernate

NHibernate[10] is an object-relational mapper for C# which can be used to abstract away the database access layer. This is used as an alternative to interacting directly with the database so that the implementation code is not directly tied to the underlying database used.

In addition, Fluent NHibernate[11] was used to avoid using XML files to define mappings so that the mapping code remained compile safe.

3.1.7 SQLite

SQLite[12] is a library which implements a self contained SQL database engine, allowing a databases to be stored to a single portable file. The advantage of this is that, for example, the different data sets that will be simulated or collected could each be persisted to an

[5] http://www.mathdotnet.com/
[6] http://www.alglib.net/
[7] http://www.emgu.com/wiki/index.php/Main_Page
[8] http://opencv.willowgarage.com/wiki/
[9] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[10] http://nhforge.org/Default.aspx
[11] http://www.fluentnhibernate.org/
[12] http://www.sqlite.org/

individual database file. It is also possible to use SQLite in memory, which helps facilitate unit testing of database code. SQLite was used as the back end storage mechanism.

3.1.8 Html Agility Pack

The Html Agility Pack[13] is a HTML parser library for C# which can be used for parsing the web pages scraped as part of the data collection process. This is used as an alternative to, for example, direct use of regular expressions to parse HTML, which can lead to unpredictable, hard to maintain code[14].

3.1.9 NUnit

NUnit[15] is a C# unit-testing framework which was used to write unit tests for the implementation, an important part of any software development.

3.2 Persistence

One of the first steps I took was to figure out how best to store the data that I would be both simulating and collecting from tipping websites. I decided to use a relational database for this, which meant I had to first figure out the design and then consider how to interface with C# .

3.2.1 Database Design

Figure 3.1 shows the relationships between the main data entities. These were designed so that each entity corresponds to a table in the database and also to a class in the object oriented implementation. The design allows any "sport" imaginable to be represented; a sport is assumed to be a type of event for which matches are played by participants of the sport, referred to as runners. For each match, a number of different markets may be available. For a particular available market, there will be a number of possible outcomes for that market, referred to as selections, each corresponding to a runner for the sport in question.

To put this in perspective, for the case of tennis, specifically the tennis win market, the runners are the tennis players and for each tennis match played there will be a win market available for the match, for which the two participants in the tennis match will be the possible outcomes for the market. Tipsters who belong to a site, say TennisInsight members, can then tip matches according to who they predict will win each match.

[13] `http://htmlagilitypack.codeplex.com/`
[14] `http://www.codinghorror.com/blog/2009/11/parsing-html-the-cthulhu-way.html`
[15] `http://nunit.org/`

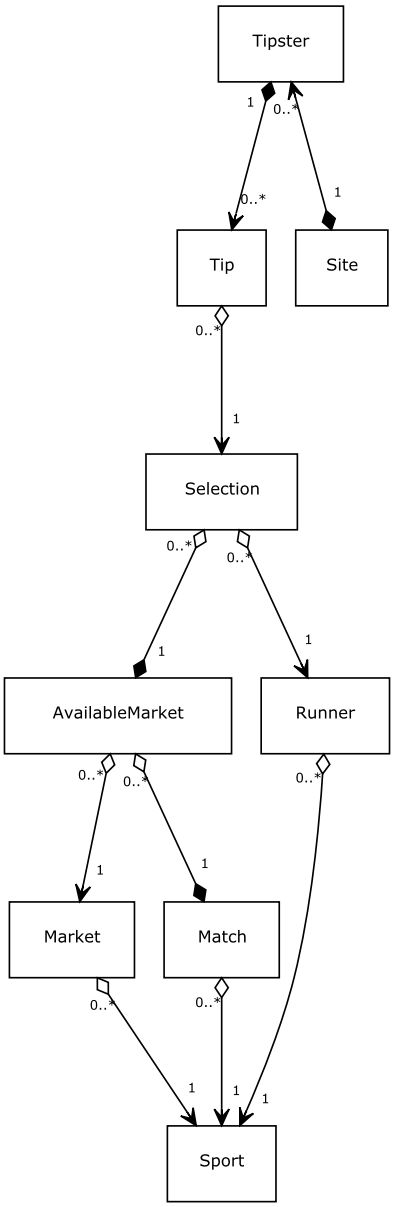

Figure 3.1: Relationships Between Main Data Entities

3.2.2 Persistence Layer

I was careful to make sure that the layer between the database and the programming language was not tightly coupled; I wanted to make it easy to use a different storage implementation in the future. To achieve this, I used NHibernate.

I found getting used to NHibernate a challenge at first and it took a lot longer than expected to get everything "just working". However, after overcoming this, aided by the excellent NHibernate 3.0 Cookbook[11], I found that the persistence layer did exactly what I wanted it to and I could now talk about my entities in terms of objects in the programming language, without having to worry about the intricacies of how they would

actually be persisted.

3.2.3 Storage Mechanism

I decided to make use of the SQLite database engine as the back end storage mechanism, mainly for the convenience of not having to run a full blown database server instance, although a different database could be "swapped in" at any time due to the fact that the persistence layer abstracts away the underlying storage mechanism. Another benefit of SQLite is that it allows you to use entirely in-memory database instances, which was useful for running simulations which did not need to be persisted, as well as for unit testing purposes.

3.3 Scrapers

After setting up the persistence later, the next step was to collect the real-world data. I did this early on in the project since it was possible that the websites would later become unavailable.

I encountered problems when attempting to gather data from OLBG, since the site had a particularly sensitive flood detection, which meant it was not feasible to scrape the data in good time. In addition, only 45 days of tip history were available, whereas TennisInsight had tip histories going as far back as 2008 available. As such, I focused on TennisInsight, where I did not encounter any problems collecting the data.

3.3.1 TennisInsight

The main data to be scraped from TennisInsight was the tip histories of all of the tipsters. These tip history pages used plain HTML which made it possible to learn the structure of the pages and extract the relevant information. An example of a tip history page is shown in Figure 3.2.

I decided to use a HTML parser library rather than directly attempting to use regular expressions. This was for two reasons, firstly regular expressions can be hard to maintain ("what does that do again..?"). Secondly, if pages did not appear how I expected them to, using a library allowed me to use exception handling to make sure I was not populating the database with garbage! On several occasions this appeared to be a good decision; some pages served up contained bad structure, for example missing closing HTML tags.

Unexpectedly, I also managed to discover a *major* security flaw on the site which meant that *any* user's password and personal details were publicly viewable in plain text! The issue was reported to the site owner and has since been fixed.

The scraper made use of a separate worker thread for saving the downloaded pages to disk so that disk access was not a bottleneck. Scraped pages to save were queued up in a string representation and the worker thread would continuously check for new work on the queue to be saved.

Figure 3.2: TennisInsight Tip History Page Example

3.4 Simulated World

With my scraper now populating the database with the thousands of tips on TennisInsight, I now turned my attention to implementing the simulated world. The idea was that simulations would be persisted in the same way that the real-world tips were, so that the analysis side of the implementation is completely decoupled from the method which was used to generate/collect the data to be analysed.

3.4.1 System Overview

A simplified overview of the simulation system is shown in Figure 3.3. The SimulationConstructor uses a SimulationSpecification to construct the simulation, which it may then persist if required. The design of the simulated world in the background section translated almost directly; each tipster has a strategy comprising of three components, estimation, selection and stake.

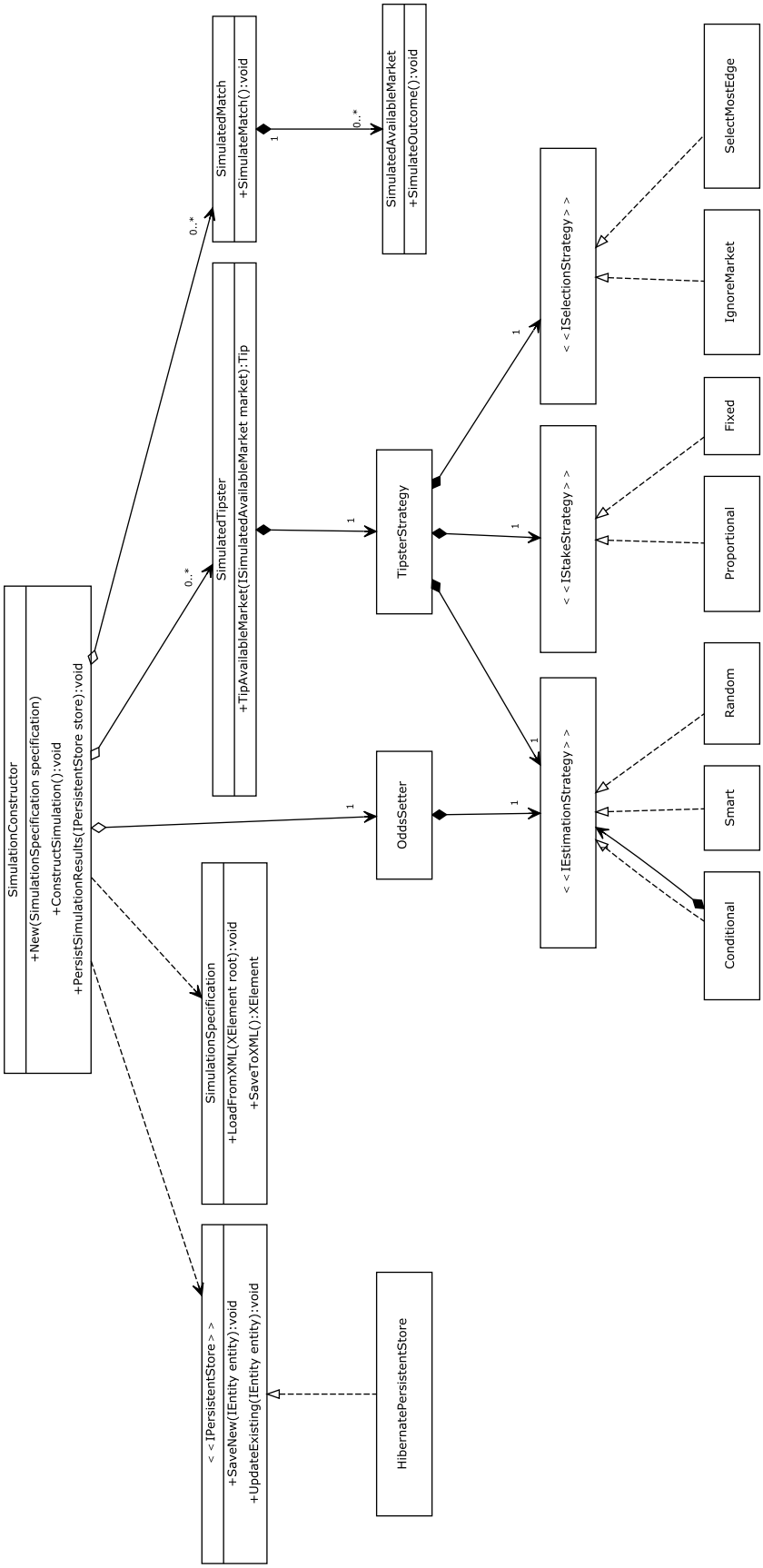

Figure 3.3: Simulation System Overview

3.4.2 XML Specification

The SimulationSpecification also supported loading and saving to a human readable XML format, which is flexible enough to be used to design different populations of tipsters to simulate. By specifying a seed for the random number generator in the specification, this meant that the XML file would describe the simulation completely. The benefit of this is that, for example, instead of having to transfer a whole persisted simulation to a third party, we can now just send them a small XML file which they can use to simulate the same data set.

3.5 Analysis

Now I had a source of data, I moved on to the next step, analysing the data and attempting to extract information from it. I used existing implementations for the classifiers where possible and implemented a cross validation framework which could be used to evaluate performance. In addition to this, I implemented an experimentation framework, allowing experiments to be designed which would allow fixing all but one variable of interest constant to investigate the effect of the controlled variable.

3.5.1 Classifiers

Many of the classifiers relied on an underlying matrix structure, corresponding to the feature structures first described in Section 2.3. I first wrote transformation code which transformed the incoming data which was in terms of selections, tips and tipsters to the corresponding matrix form. This led to a generic SelectionToMatrix classifier, which wraps a classifier which deals with the matrix representation and performs the transformation of incoming data into the matrix representation. Classifiers which worked directly with the base representation would simply use it directly. This structure is shown in Figure 3.4 with some of the classifiers for clarity.

Accord.NET implementations were used for the LDA, logistic regression and Naive Bayes classifiers. The Accord.NET SVM implementation was found to be too slow; in response to this I found an optimised implementation in the EmguCV computer vision library. I wrote the WKNN implementation.

3.5.2 Grid Search

A common grid search implementation was devised which was used for classifiers who needed to learn parameters. The grid search could be used to start from some predefined coarse bounds and attempt to refine the best coarse point found by refining the grid locally around this point. Although not guaranteed to find a global optimum, this method would at least find a local optimum in the parameter space. The "best" parameters would be chosen according to the precision they achieved. The complexity of this grid search approach is exponential in the number of parameters, for example if we have p parameters and we search on a grid with n points per parameter, the total number of configurations to explore is p^n.

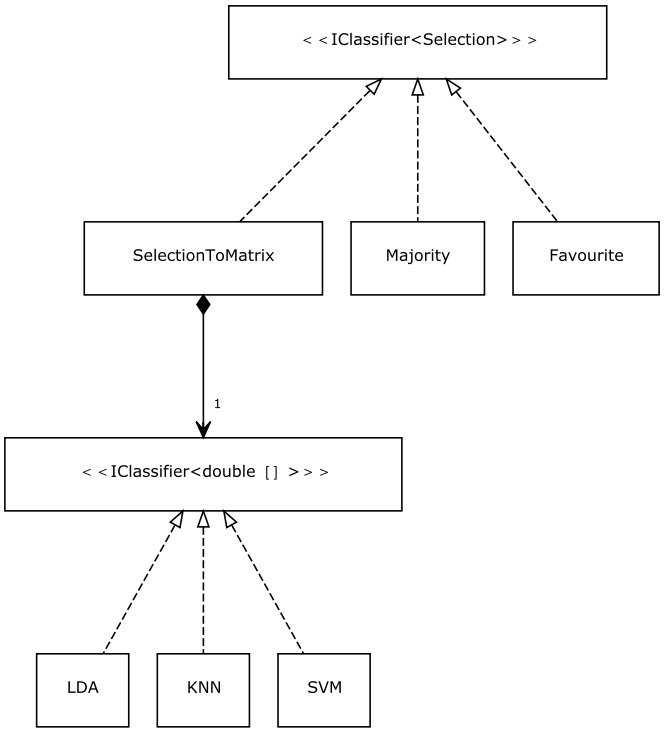

Figure 3.4: Selection To Matrix Classifiers

3.5.3 Cross Validation

I took care when performing cross validation to consider the specific nature of this problem. It does not make sense to have one selection from a match in the training set and another selection from the same match in the testing set! So, cross validation was instead done by match, by first splitting up all matches into folds and then when it came to training the classifiers (which use selections) I would just extract all the selections from all the matches in that fold.

3.5.4 Experiments

Experiments were designed to be repeatable; each simulated data set is uniquely defined by the specification (typically stored in the XML file representation) and the random seed used to generate it. Experiments alter a variable of interest whilst keeping others constant and then write the performance results to file for later analysis.

3.6 Testing

3.6.1 Unit Testing

I made use of NUnit for all unit testing, which was especially important for checking the data representation transformations were valid.

In addition, I found that the NUnit test runner was useful as a simple way to run several of my experiments simultaneously.

3.6.2 Acceptance Testing

I made a point of, whenever I encountered a problem with the code, to create a corresponding test case which described the problem. I found this technique especially important with the TennisInsight scraper, when I found bugs in the HTML code which threw exceptions in my code, I could include the bad page as a test example for future reference.

4

Simulated World Investigation

This chapter investigates the simulated world which was constructed, beginning with a baseline population in Section 4.1 which is used to guide the choice of a data representation and feature selection methods. Section 4.2 goes on to make modifications to the baseline population to investigate the effect of various changes in the behaviour of the tipsters on the predictive performance of the classifiers. Finally Section 4.3 gives a brief summary on what we have learnt over the process of conducting these experiments.

4.1 The Need For a Baseline Population

Given the number of degrees of freedom in the simulated world, we need to have a way of comparing performance to some known state; to this end, a specific population is considered as the baseline and future experiments will introduce small changes to this population as we learn more about how the variables in the simulated world affect the population's wisdom and our ability to detect it.

A sport with just one market, which can have exactly two outcomes is considered. The initial baseline population consists of 200 tipsters, a group of 100 smart tipsters with $r = 0.1$ and a group of 100 random tipsters. For the odds setter, $r = 0.2$ is chosen so that the smart tipsters have a slight edge and the odds setter does not introduce over round into the odds set. All tipsters only tip when they perceive that they have an edge of at least 10%. Finally, all tipsters use a proportional stake sizing strategy, with a maximum stake size of 100.

This population was chosen since it represents an optimistic potential situation, where there is a clear cut distinction between random and smart tipsters, with some tipsters having more knowledge than the odds setter. It is the kind of situation we would hope to be able to perform well on by picking up on the signal from the smart tipsters.

For all of the data sets investigated in this chapter I carried out 10 fold cross validation a minimum of three times and averaged the results in an attempt to smooth out variance introduced by the inherent random nature of the simulations. The only classifiers which exhibited noticeable large variances in performance were the support vector machines, which will be seen to exhibit erratic changes in performance in some cases. Each trial

was simulated from the same population specification but using different random seeds. Learning classifier hyper-parameters was achieved using a coarse to fine grid search where each parameter configuration is evaluated with a 5 fold cross validation on the training set, to obtain optimal precision.

4.1.1 Data Representation

Of the six potential representations discussed previously (stake, odds, stake and odds and the "and not" versions), it is unclear which is best suited to the problem. The baseline population will be used to guide this choice.

Additionally, it is unclear how large a sample size is "enough" for each of the classifiers; we will consider a range of sample sizes to be sure that each classifier has the opportunity to perform optimally.

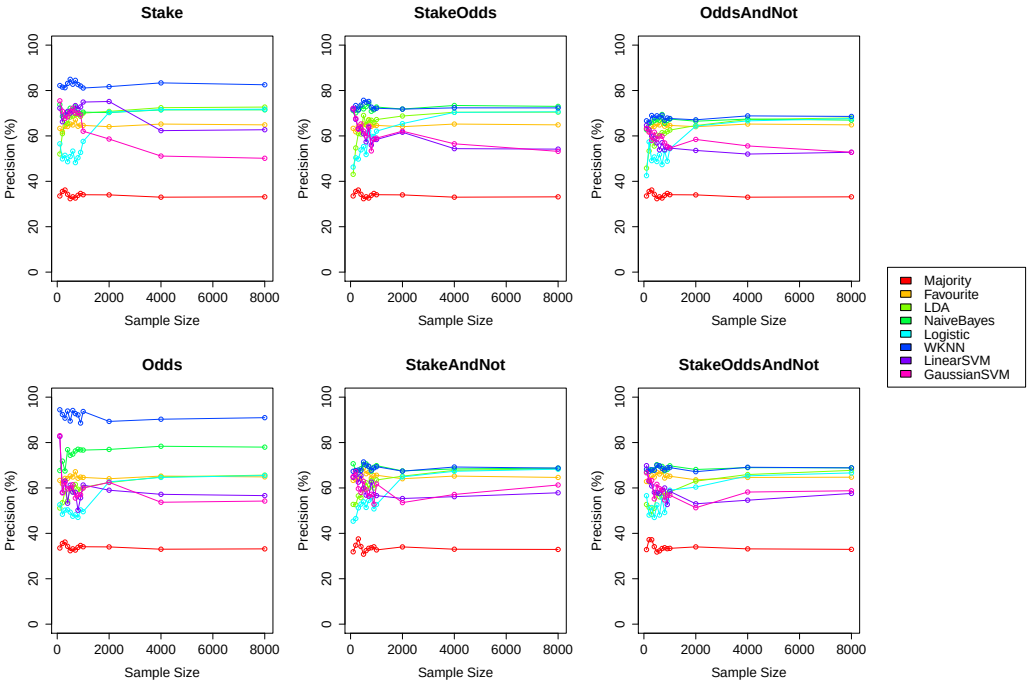

Figure 4.1: Data Representation Precision

Figures 4.1 and 4.2 show a comparison of the recall and precision achieved by each classifier under each data representation for the baseline population for varying sample sizes.

I found that the logistic regression approach would occasionally not reach convergence, even for the largest sample sizes considered. The performance results using the models which did not converge are included for completeness, it seems logistic regression may be ill-suited to the problem. Also, the two support vector machine methods appeared to be the worst performing; for some representations they approached a precision of around 50%, no better than guessing!

From the precision plots it seems clear that the three "and not" representations perform

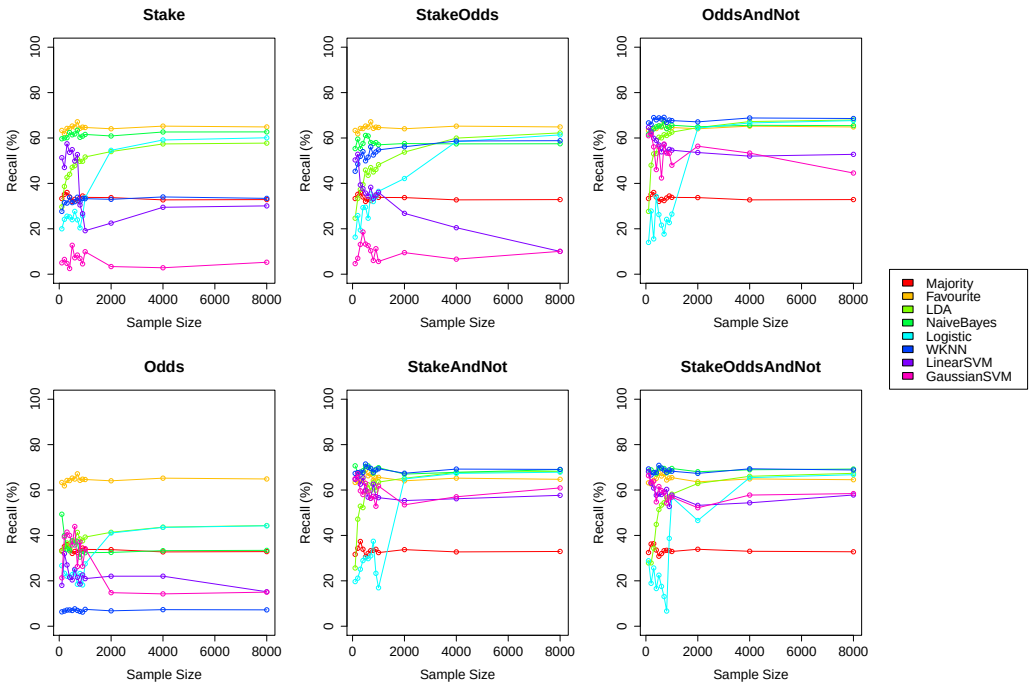

Figure 4.2: Data Representation Recall

worse in all cases when comparing them to their single form counterparts, however the recall of these representations is better in almost all cases. Since we are most interested in the precision of the classifiers, the choice is now between the odds and stake representations, which had the best precision rates.

Although the odds representation appears to be most suited from these, upon further investigation, I found that the high precision rates achieved came at a cost; essentially all the classifiers were doing was filtering the public information, by waiting for selections with very small odds which would be almost certain to win.

Because of this, I consider the "flat return on investment", which can be defined as

$$\text{ROI}_{\text{Flat}} = \text{Precision} \cdot \overline{\text{Odds}_{\text{TP}}} - 1$$

where $\overline{\text{Odds}_{\text{TP}}}$ is the mean odds of all the true positive classifications. This is equivalent to the return on investment which would have been achieved by placing a unit stake on each of the positive classifications.

To illustrate this, the flat ROI for the classifiers under the stake and odds representation for a sample size of 2000 is presented in Table 4.1.

	Majority	Favourite	LDA	NB	LR	WKNN	LSVM	GSVM
Stake	-9.73%	1.70%	4.08%	9.28%	3.87%	6.42%	12.09%	-12.26%
Odds	-9.73%	1.70%	1.21%	4.53%	1.14%	5.53%	10.55%	-22.39%

Table 4.1: Baseline Flat ROI Under Odds/Stake Representations

It seems that the flat ROI is larger or equal under the stake representation than the odds for all of the classifiers. Due to this, we will proceed using the stake representation

only. Also, note that the Gaussian SVM has the largest negative return, whereas its linear counterpart has the largest positive. It is suspected that this may be due to the relatively large number of random tipsters in the baseline population, which result in essentially lots of "mis-labelled" data, which it seems the Gaussian SVM was sensitive to.

The choice of using only the stake representation may seem intuitive since we seem to be ignoring potentially useful information in the odds, but in fact the stake representation is indirectly influenced by the odds information and so we *are* taking the information into account, but just not by directly using it as a feature. Additionally, it seems clear that the stake is more expressive for a tipster; he not only gets to say "these odds are acceptable to me" (as in the odds representation) but, by betting his beliefs (as in the stake representation) he gets to say "this is how much edge I think I have", which will depend on what the odds being offered were.

4.1.2 Filtering

Instead of just directly using all available tipsters, it may be possible to "filter out" tipsters who appear to be behaving randomly as a kind of feature selection step, rather than expecting the classifiers to figure out which tipsters are behaving randomly.

One way to do this is by considering the strike rate of each of the tipsters (the proportion of their tips which won), since for a tipster who makes N tips, if they were behaving randomly we would expect the number winning tips to follow a Binomial$(N, 0.5)$ distribution. A simple binomial test can be carried out for each tipster, taking into account how many of their tips won.

A shortcoming of this method is when tipsters are not behaving randomly, but achieve a strike rate less than the favourite strategy. Such "sub-smart" tipsters would slip through the net.

Another method is to consider the mean profit made by each tipster; if a tipster is behaving randomly we would expect a mean profit of zero. A one sample student t-test can be used to formalise this by testing the null hypothesis that the mean profit is zero.

The two methods could also be combined; it may be possible that a tipster with a seemingly "random" strike rate still manages to make a consistent profit. This might occur if the tipster was good at predicting the true odds of outsiders for example. To combine the two methods, we look for tipsters who were significant in *either* of the two tests.

Note that two-sample tests are appropriate here, since a "losing" strategy, e.g. a tipster who obtains a strike rate of 10% can be turned into a "winning" strategy by simply betting on the alternative outcome.

In practise, I noticed an initially surprising result. The "random" tipsters in the baseline population are in fact not random at all in the sense of the above binomial test. The tipsters are considering the public odds information and essentially waiting for points where their predictions disagree with the odds setters predictions. However, since the tipsters have simply randomly generated these predictions, it turns out that this results in them disagreeing with the odds setter almost all the time and so they end up adopting

a "non-favourite" strategy. Indeed, observing the average strike rate of the tipsters confirmed that the strike rate was approximately one minus the favourite strike rate.

This seems an unlikely thing to happen in reality; we would not expect tipsters to be taking the public information into account and then going against it intentionally! In response to this finding, I now consider a modified version of the baseline population where the "random" group is now a group of 100 truly random tipsters, who simply ignore the public odds and make decisions based entirely on their own estimates, into the baseline population to see if the filtering methods would be effective.

Considering a sample of 2000 matches, the combined filtering strategy is applied to this modified baseline data set. The results are presented in Figure 4.3, with the corresponding non-filtered results for reference. Applying this filtering also made the logistic regression approach a viable option; there were no problems reaching convergence this time. This is likely a direct result of the dimensionality reduction which occurs as part of this feature selection process.

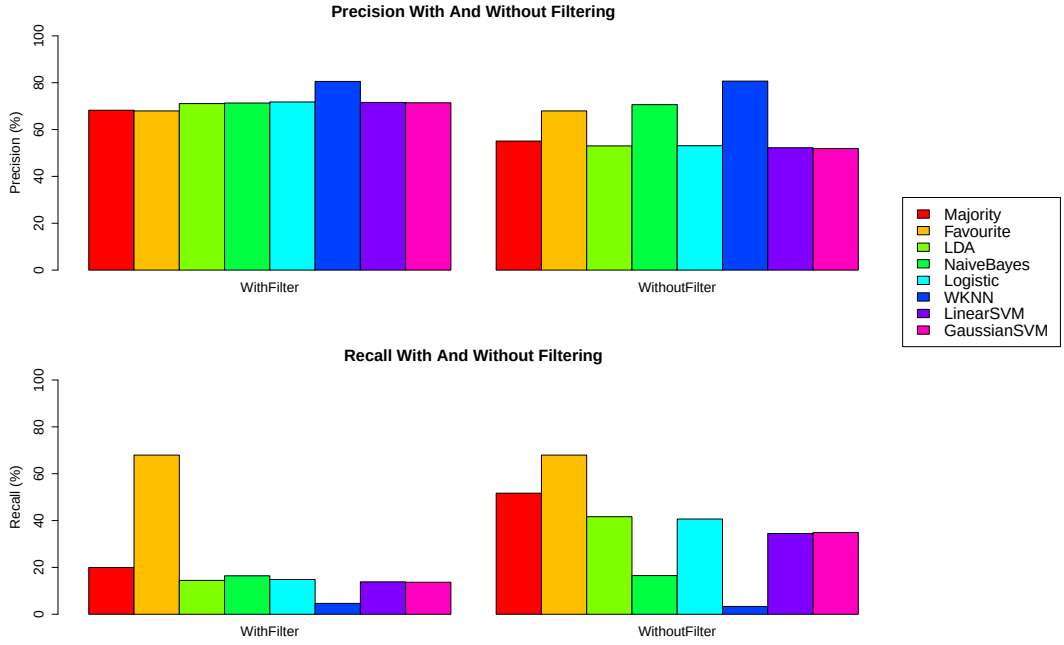

Figure 4.3: Filtered Baseline Data Performance

Since most or even all of the random tipsters have been filtered, even the simply majority vote now becomes a reasonable solution. Note that the Naive Bayes and WKNN methods remain at approximately the same level of both precision and recall; it seems as if they are capable of dealing with the presence of random tipsters better than other classifiers. Since all classifiers improved in terms of precision following the filtering it seems sensible to include this filtering method as a preprocessing step when considering new data.

4.2 Experiments

Considering the previous findings, we now modify the baseline to include a group of sub-smart instead of the random population, this is done since in any case where there is a truly random group of tipsters, we could just use the aforementioned techniques to identify and filter them out and so it makes more sense to include a group of tipsters who would not have been filtered out.

A group of "sub-smart" tipsters with $r = 0.6$ is introduced for this purpose in the following experiments, chosen to have a strike rate less than the favourite strategy. These tipsters will ignore the public odds information in making decisions on whether to place a bet or not, since if they did not, the phenomenon observed with the "random" tipsters following a non-favourite strategy would re-emerge. The sub-smart group aims to model tipsters who either have an independent source of information which is inferior to the public information, or tipsters who misuse the public information.

The sample size is also fixed to 2000 matches for the following experiments. This is done both to keep the computations feasible and also to represent the fact that, in the real world, we may only have access to a fixed number of matches and not have the luxury of generating more data at will.

The performance of the classifiers can be benchmarked by looking at the corresponding performance that would have been achieved using either the favourite strategy (which encompasses the public information) and the simple majority voting strategy (which is a naive way of using the data). Outperforming the favourite strategy in terms of precision may indicate that the classifiers have extracted knowledge from the smart tipsters in the population.

4.2.1 Population Size

For this experiment, we will consider keeping the proportions of smart to sub-smart tipsters being fixed in a 1:1 ratio, as in the baseline model, but now increasing the total population size.

Increasing the population size corresponds to increasing the dimensionality of the problem. For small enough dimensions, this is not a problem, but there comes a point where the computations become infeasible. Figure 4.4 shows the performance of the classifiers for a range of population sizes. Past a population size of 1000 I found the computation would become intractable.

To consider even bigger population sizes whilst remaining computationally feasible, I decided some form of data compression or feature selection may be appropriate. I considered two approaches, using PCA to compress the data set and a novel approach involving considering the magnitude of the factor loading arising from an LDA analysis as a means for feature selection.

In practise, I found that the PCA method turned out to be unsuitable; the transformation appeared to destroy the sparseness of the data which made it a bad choice in this case since this degraded the performance to almost guesswork.

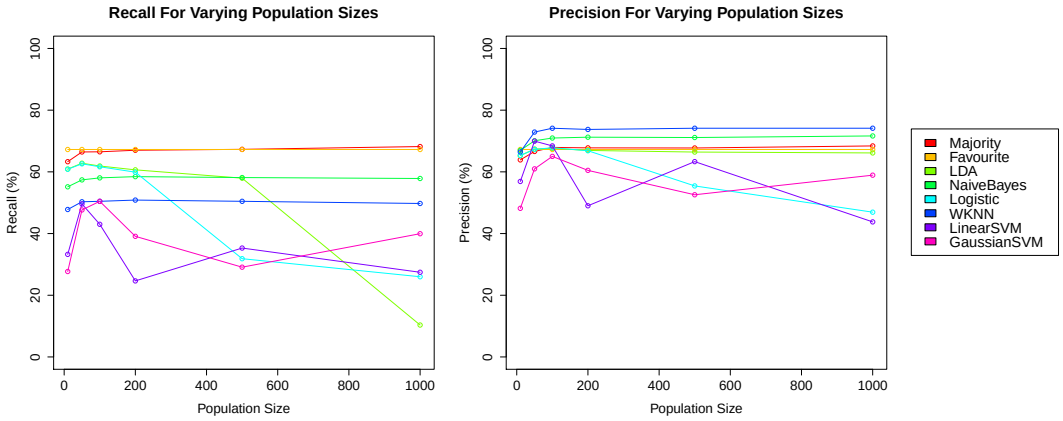

Figure 4.4: Varying Population Sizes (Direct Approach)

For the LDA method, we simply perform LDA on the data in question, then identify the top L factor loadings with highest magnitude, where L can be chosen to directly control the dimensionality of the problem so that it remains feasible. The idea here is that all we are doing is selecting the L most "important" features in some sense and so the original data structure is preserved.

Figure 4.5 shows the performance obtained using this method with $L = 100$. Unfortunately, I still found that there was a limit to this; beyond a population of 1000, the LDA also began to take prohibitively long to run and so now the feature selection process was a bottleneck in itself. In addition to this, although the selection method appears to have been effective in terms of precision, the recall rates are severely negatively affected.

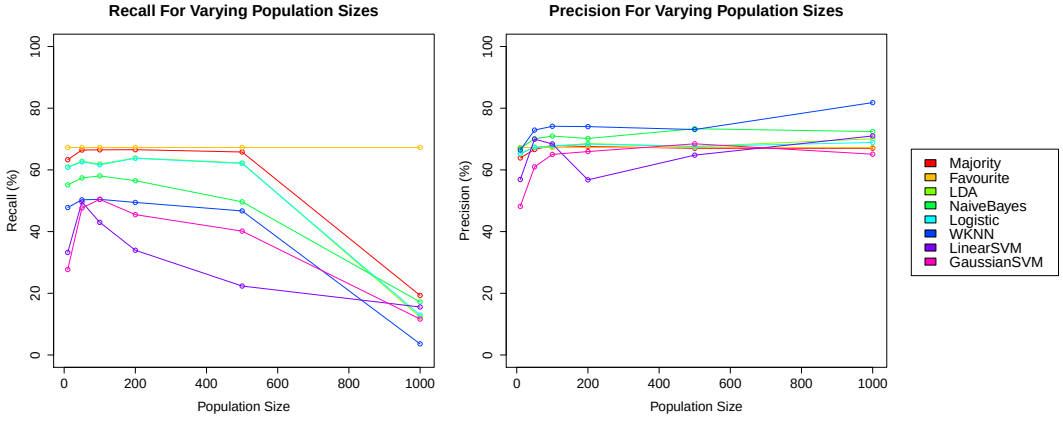

Figure 4.5: Varying Population Sizes (LDA Compressed)

One final way to reduce the dimensionality by feature selection is simply to take the "top N significant" tipsters in terms of either their strike rate or return on investment. In other words, we perform hypothesis tests as in Section 4.1.2 and record the p values achieved. Then, we simply take the N most significant observations (i.e. those with smallest p values).

In the case where all tipsters make exactly the same number of tips, this would corre-

spond to just taking the tipsters with the top N strike rates or return on investments, however in cases where the tipsters make different numbers of tips the p value method is important. This is because, for example, a tipster with a strike rate of 90% achieved by getting 9 out of 10 tips correct is perhaps not as impressive as a tipster who also achieved 90% but by getting 900 out of 1000 tips correct.

I found this method scaled much better than the LDA method and I was able to look at population sizes in excess of 1000. Figure 4.6 shows the results for compressing using the top 100 most significant strike rates.

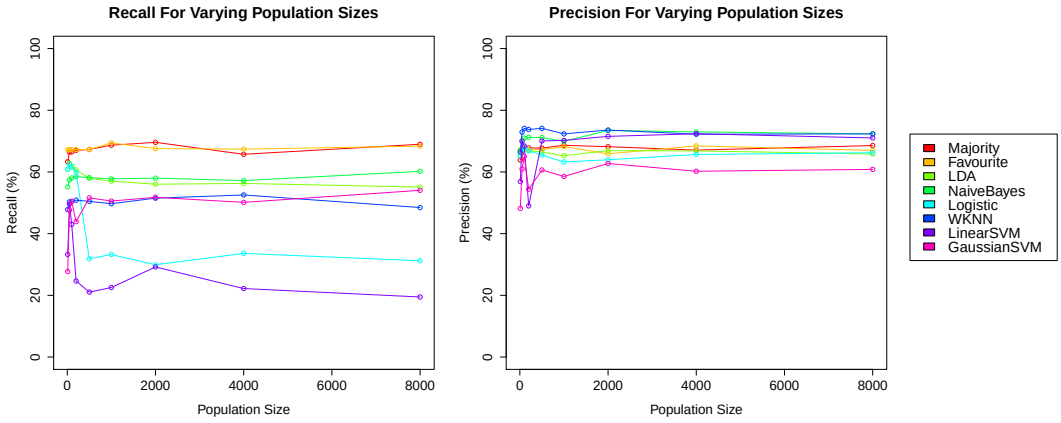

Figure 4.6: Varying Population Sizes (Significant Strike Compressed)

The performance appeared to remain constant, so long as the number of dimensions you are willing to keep remains constant. This makes sense since you are essentially keeping the same smart corner of the population which is most significant.

4.2.2 Proportion of Smart Tipsters

Here, we consider a "needle in the haystack" problem where the proportion of smart tipsters is lowered so that the majority of tipsters in the population are now sub-smart.

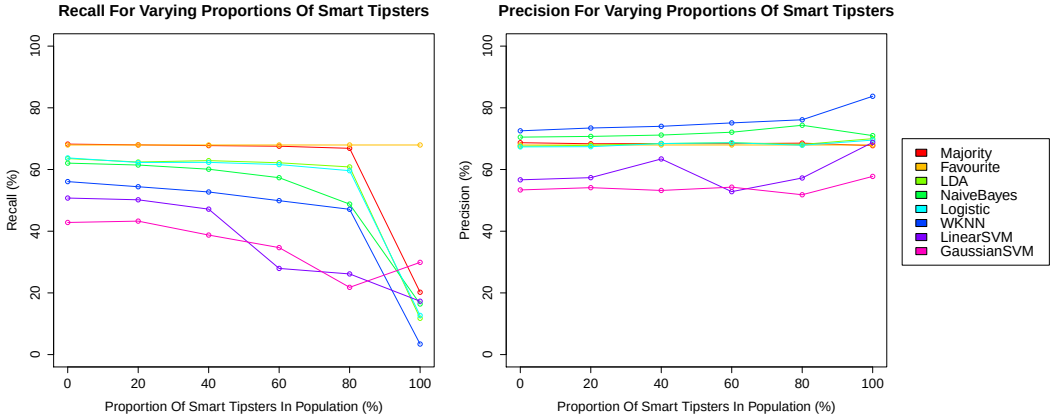

Figure 4.7: Varying Proportions of Smart Tipsters

Figure 4.7 shows how, even when a very small proportion of the population is smart, the WKNN and Naive Bayes classifiers can still achieve precision rates greater than the favourite strategy precision. The other classifiers manage to match the precision of the favourite strategy, but seem to struggle to exceed it by more than a fraction of a percentage point, even in the case that all of the population is smart.

The recall of all the classifiers remains reasonably high, staying above 50% in most cases, apart from the case of all the population being smart, upon which the recall rates plummet. A possible explanation for this is that, when all the population is smart, there will be less occasions that the tipsters perceive they have an edge and so less tips are made in total.

4.2.3 Edge of Smart Tipsters

Varying the edge that the smart tipsters have over the odds setter corresponds to the varying amounts of private information the smart tipsters may be privy to. Figure 4.8 shows how, as the r value of the tipsters decreases, the performance decreases accordingly. In the worst case scenario, with $r = 0.5$ so that the smart tipsters are no longer smarter than the odds setter, the precision performance appears to tend to that of the favourite strategy.

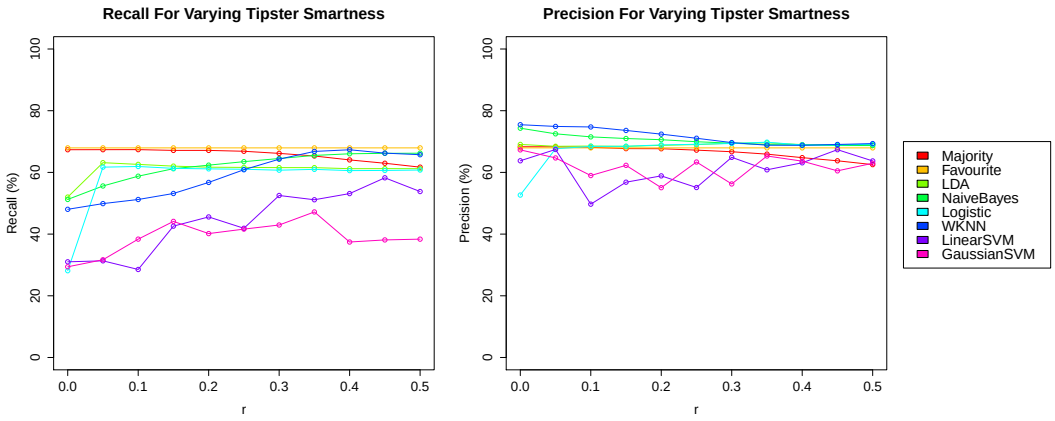

Figure 4.8: Varying Edge of Smart Tipsters

The performance of the SVM classifiers was the most erratic of all and it is beginning to seem they may be unsuited to the problem because of this. On the flip side, the Naive Bayes and WKNN methods have performed consistently well so far and are looking to be the most likely candidates for the real world investigation.

Another thing to note is that the logistic regression approach actually performs *worst* when the tipsters are able to predict perfectly (with $r = 0$)! A possible explanation is that, when the smart tipsters are able to predict perfectly, they make less tips overall because they are waiting for opportunities where the odds setter has "got it wrong" so they have an edge, which makes the pattern less clear to the regression, since it has lots of missing data in these cases.

4.2.4 Copycat Tipsters

Another possible behaviour of tipsters is that they simply copy a particular "pack leader". For example, if the pack leader happened to be a random tipster, this would mean that the same bad opinion is represented multiple times, whereas if the pack leader were a smart tipster, a good opinion would essentially be repeated.

This can also be thought of as the number of "original opinions" within a group, where one original opinion would correspond to everyone copying just one tipster and on the other end of the scale, every tipster may have their own independent opinion.

The number of original opinions per group (random, smart, sub-smart) is varied to see what effect the number of original opinions had on performance.

Note that there is a chance that this time the majority of the random group are not be filtered, for example if by chance one does very well and all follow this.

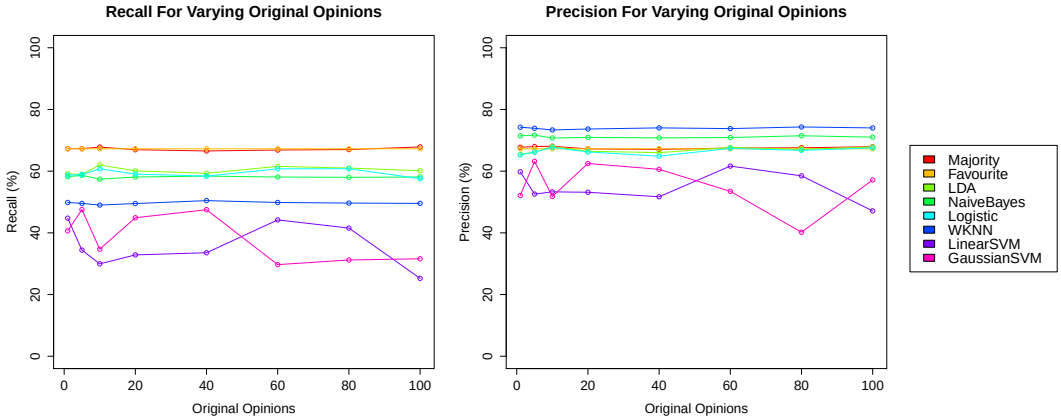

Figure 4.9: Varying Number Of Original Opinions

Figure 4.9 shows how, interestingly, the number of original opinions per group appears to have no significant effect on the performance of the classifiers. In other words, it seems that what matters is that there is at least *some* reliable opinion and not necessarily the number of original opinions. As in previous experiments, the SVM classifiers show no clear pattern and their suitability is again put into question, whilst the WKNN and Naive Bayes classifiers perform the best.

4.2.5 Tipping Frequency

Something else to consider is that, in reality, tipsters will not consider *every* match but instead occasionally decide not to even bother looking at a particular match. This is modelled by having tipsters who consider some percentage of all matches and the rest of the time abstain. The choice of which matches to abstain on is assumed to be independent for each tipster.

Figure 4.10 shows how the recall is negatively affected in all cases for a low tipping frequency. The precision of most of the classifiers is largely unaffected, with the exception

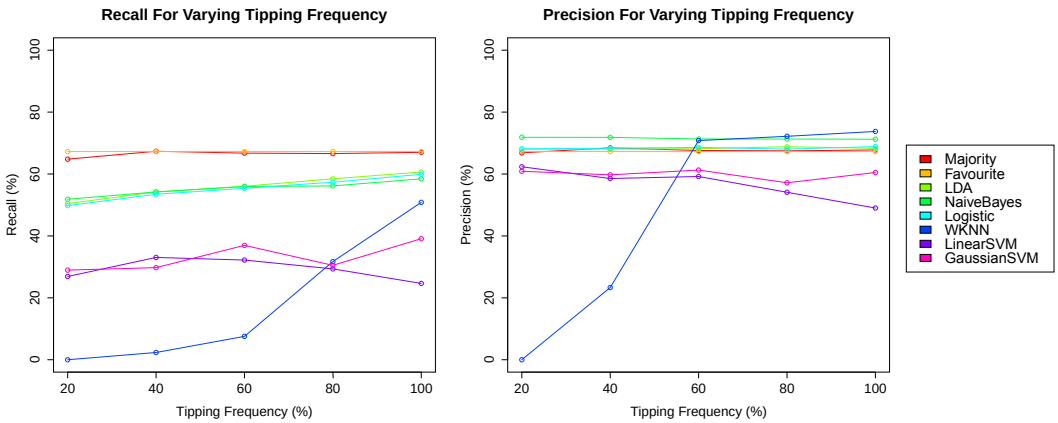

Figure 4.10: Varying Tipping Frequency

of WKNN which performs extremely poorly for a low tipping frequency. Yet again the SVM classifiers show unclear changes in performance.

4.2.6 Discrete Bets

Another issue in reality is that the increments tipsters can bet in may be limited, for example only multiples of 10. I consider the varying the level of discreteness, from tipsters being allowed to bet any continuous amount from 0 to 100, to a point where they are only allowed to bet either 0 or 100.

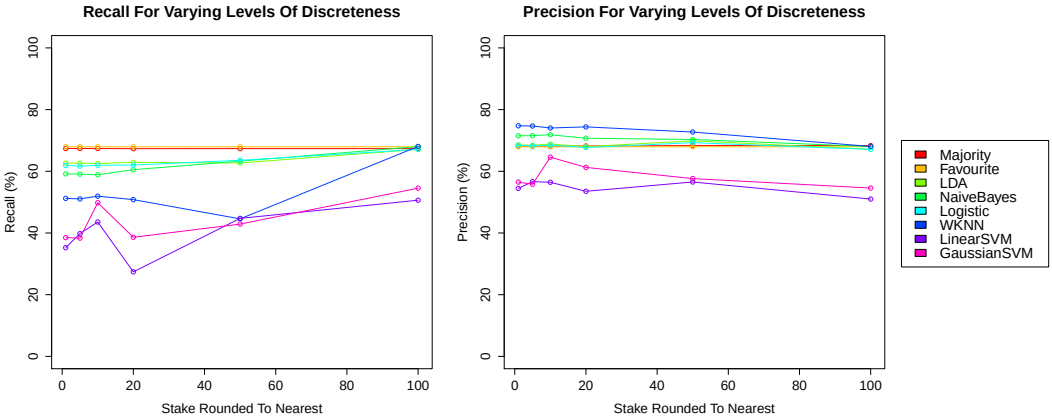

Figure 4.11: Varying Discreteness

As you may expect, Figure 4.11 shows how increased discreteness degrades performance as the tipsters are allowed to be less expressive of their opinions.

4.2.7 Specialist Tipsters

This experiment is designed to model the idea that tipsters may have a certain type of match they are an expert on, so that for these types of matches the tipsters are smart

but for all other matches they have no more knowledge than random tipsters.

Two variations are considered. In the first, the tipster behaves randomly in the case that they encounter a match they are not an expert on, which may happen if the tipster is not aware that they only have a good strategy for certain matches; in their eyes they may believe they are globally smart. In the second, the tipster is aware of their special subject and simply abstains from making a tip for all matches which they are not an expert on.

The effect of the number of groups of tipsters is varied from having all smart tipsters globally smart on all matches, to 10 groups of tipsters who are partially smart on 10 corresponding sets of disjoint matches.

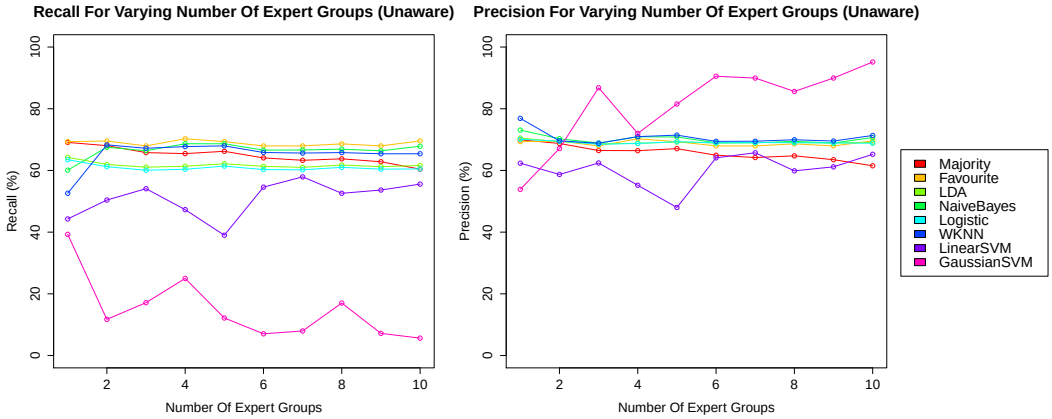

Figure 4.12: Varying Number Of Expert Groups (Unaware)

Figure 4.12 shows the result in the unaware case. It seems that the jump from tipsters being globally smart to there being two sets of expert groups results in a drop in precision, however increasing the number of expert groups has a much smaller effect, apart from in the case of the majority vote.

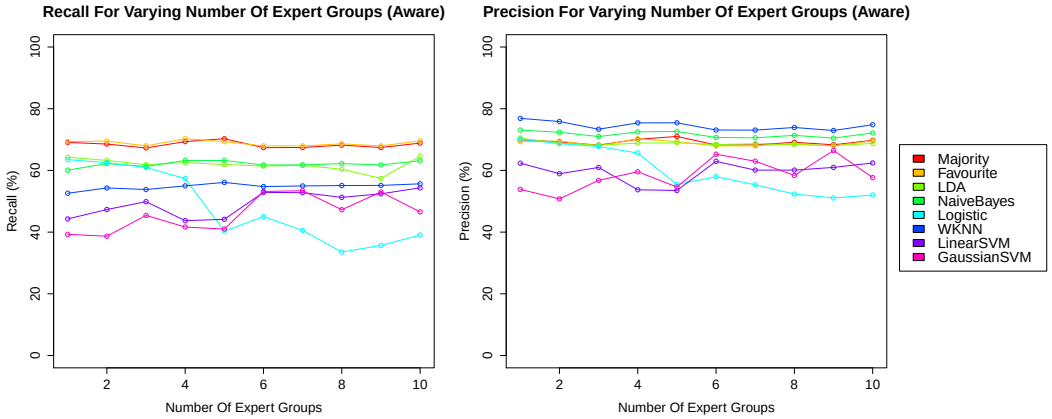

Figure 4.13: Varying Number Of Expert Groups (Aware)

Figure 4.13 shows the result in the aware case. Here these is a slight decrease in precision for most classifiers and notably logistic regression suffers the most. The overall precision

rates achieved are larger than the corresponding rates in the unaware case.

In both cases, we again see erratic results for the SVM solutions.

4.2.8 Too Smart?

Something else to consider is that, throughout these experiments, I have been assuming that tipsters are aiming to make money; they only bet when they think that they have an edge. The problem with this from the perspective of this analysis is that we cannot distinguish between situations where the tipster abstains due to perceiving the bet as being "bad value" and situations where the tipster abstains due to external reasons, as we looked at previously.

But what if they disregarded this and instead directly told me their predictions? To model this, tipsters simply ignore the market and bet an amount which corresponds to the confidence they have in the selection winning; literally "betting their beliefs".

To test this, we will reconsider the modified baseline population, with the smart and sub-smart tipsters, however now having all tipsters ignoring the market and directly betting their beliefs.

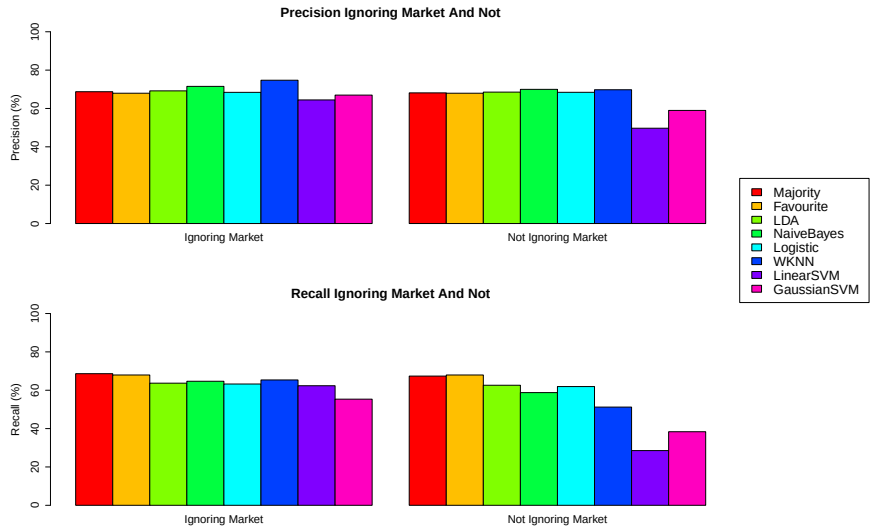

Figure 4.14: Ignoring The Market

Figure 4.14 shows how having tipsters directly bet their beliefs does appear to improve the performance in this case. This makes sense, since in all the cases where a tipster would previously have abstained due to there being "no value" he now tells me his prediction and so overall there is more useful information available to pick up on.

Perhaps then, in reality, the optimal data to collect would be these direct "how likely do you think this selection is to win?" variables. This could be done in several ways, for instance asking tipsters the minimum odds they would take a bet at, or even directly asking for percentage predictions. A proposal for a new kind of tipping website which

would aim to collect this kind of information is presented in Section 6.2.6.

4.3 Summary

By experimenting with different populations in the simulated world, we have found that it is possible to spot the wisdom in the crowds in a variety of scenarios.

We developed methods to "filter out" tipsters using hypothesis tests which consider the strike rate and stream of profits achieved. These were shown to be effective ways of dealing with tipsters who behave randomly and pollute the data with noise, as well as an effective way to reduce dimensionality.

The most consistent performing classifiers were found to be the WKNN and Naive Bayes approaches, managing to achieve precision rates in excess of the favourite strategy precision in all the situations considered. The SVM performance was seen to exhibit a lot of variance, occasionally achieving good performance but on the whole performing the worst. The logistic regression and LDA classifiers typically achieved very similar recall and precision rates, which perhaps suggests that they are picking up on the same kind of information, although LDA coped with smaller sample sizes better.

We found that, as you might expect, increasing the proportion of and "smartness" of the smart tipsters resulted in higher precision rates, whilst restricting tipsters bet sizes to discrete values decreased precision. Surprisingly, the number of original opinions in the population was found to have no observable effect on performance. The tipping frequency also had much less of an effect than expected, with only WKNN suffering a large decrease in precision for low tipping frequencies, although all classifiers suffered decreased recall rates. Groups of aware experts were found to outperform groups of unaware experts in terms of the precision rates achievable on the populations. Finally, we found that tipsters who ignore the market are much more expressive than tipsters who do not, leading to increased recall and precision for all of the classifiers.

5

Real World Investigation

In this chapter we investigate the real world TennisInsight data set. We begin with an overview of the data in Section 5.1 which considers the overall behaviour of the population. Section 5.2 goes on to analyse the performance of the classifiers which were considered in Chapter 4, leading on to a final analysis of the potential return on investments that would have been achieved. Finally Section 5.3 considers an alternative approach to classification and the performance of this approach in comparison to the other classifiers that were previously considered.

5.1 Overview of Data

Before beginning to investigate the TennisInsight data set we first explore the typical behaviour of the population as a whole. This is an important first step since we do not know how far from the simulated populations the real world data will vary.

5.1.1 Monthly Site Overview

An initial surprising finding was that, of the 13118 members on the site at the end of 2011, only 2181 (approximately 17%) of these had *ever* made a single tip. This can be interpreted in several ways. Perhaps many members sign up to follow the advice of the minority who do actively use the site. Alternatively, perhaps many of the members are fake and used by the site to either stress test or perhaps for marketing purposes - "come join us, we have 10,000 members!". Either way, this kind of non-tipping member will not be considered in future analysis.

Now the data collected from the launch of TennisInsight in 2008 till the end of 2011 is considered on a month by month basis. This is done to get an idea of how the number of tips placed and number of active members varies over the course of each year, as shown in Figure 5.1.

Note that the frequency proportion scale in the Figure has been normalised per data series, so that 1.0 represents the highest observed frequency for that series. For each month, new members are members who happened to place their first tip in that month

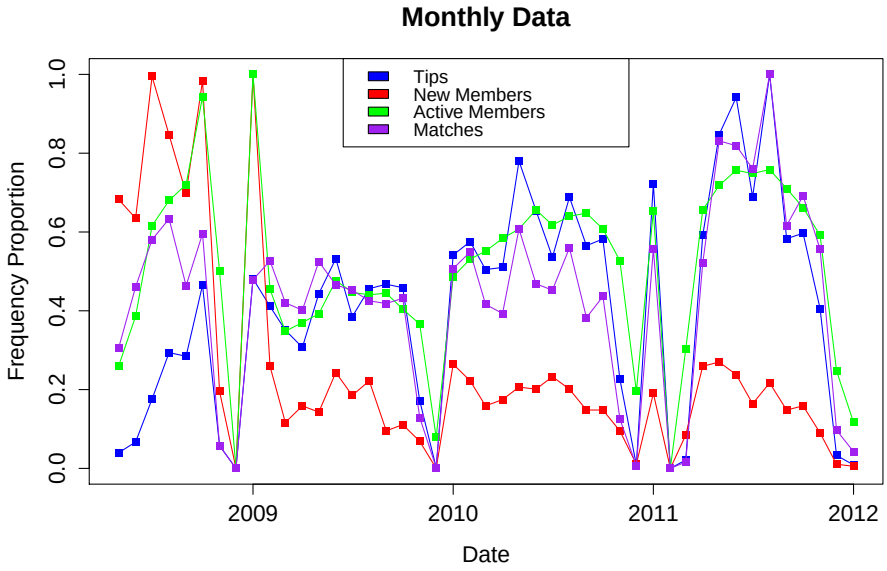

Figure 5.1: TennisInsight Monthly Overview

and active members are members who placed at least one tip in that month.

The dips in activity during the winter months are due to fact that tennis is a sport which is mainly played during the summer months; this is highlighted by the peaks around July each year.

Note that 2008 follows a different pattern than subsequent years, having the highest influx of new members, but a relatively small number of tips compared to subsequent years. This is despite the fact that the number of matches is similar to that of other years and the number of active members is also the highest recorded.

One possible explanation for this is that, following the site's launch, many new members decide to try out the site, but do not go on to become regular users. Due to this, the 2008 data will be excluded when we come to the final analysis stage, to be considered as a kind of "burn-in" period for the site.

From 2009 onwards, the number of new members remains similar year by year, but the number of active members is increasing year by year. This is a good sign for the purposes of this investigation; it indicates that members from previous years have remained active in subsequent years and so there will be a wealth of historical data available for this kind of member.

5.1.2 Tipster Behaviour

By considering the number of days a tipster has been active (i.e. the number of days between their first tip and most recent tip) this can give us an idea of how long the population tends to remain active on the site from when they made their first tip.

Something else to consider is the total number of tips made by each member; members who tip infrequently and make a very small number of tips are not useful for the investigation since it will be hard to tell if their behaviour was due to chance or not with such

a small sample size. As a rule of thumb, tipsters who have made less than 30 tips will be excluded in future analysis.

Figure 5.2 shows the distribution of number of days active and also the number of tips made by each member on a log scale. As you might expect, there is some correspondence here: the longer a tipster remains active, typically the larger the number of tips they will have made.

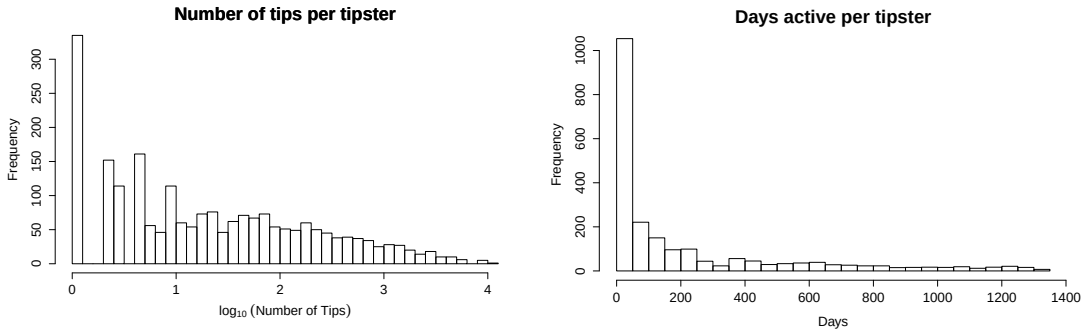

Figure 5.2: TennisInsight Population Activity

Approximately 16% of all active tipsters were only active for a single day and 42% were active for more than 100 days. Additionally, approximately 59% of all active tipsters made less than 30 tips and only 11% made more than 500 tips.

5.1.3 Random Behaviour?

It is possible that the performance of the tipsters was due to chance; we proceed to investigate whether this appears to be the case.

For a tipster predicting the outcome of a single match, the null hypothesis that the tipster selects at random corresponds to a probability of 0.5 for correctly guessing the outcome.

For M such independent predictions, the expected number of correct predictions is distributed as $Binomial(M, 0.5)$.

If we extend this to T tipsters, where tipster i has made M_i predictions, then the total number of correct tips across all tipsters is distributed as $Binomial\left(\sum_{i=1}^{T} M_i, 0.5\right)$.

A simple Binomial test can then be carried out to test whether the observed number of correct tips differs significantly from what we would expect if all tipsters were behaving randomly.

The observed number of winning tips was 332242 out of 521988 trials, resulting in an observed probability of success of approximately 0.636, with corresponding 95% confidence interval of $[0.635, 0.638]$ and $p < 2.2 \times 10^{-16}$, so we conclude that the probability of success is certainly not equal to 0.5 and in fact appears to be significantly greater than 0.5.

5.1.4 Profitable Behaviour?

Another test is to see whether the mean profit (where return is the amount of money lost or won for each bet) over all the returns obtained by all tipsters appears to be significantly different from zero, to determine whether the tipsters appear to be profitable or not on average. As such, a one sample Student's t-test was conducted using all returns, which resulted in a sample mean return of -0.239 units with corresponding 95% confidence interval of $[-0.476, -0.003]$ and $p = 0.047$, so we conclude that the mean return does appear to be different from zero at the 5% level, however since the confidence interval is negative, it appears that the true mean is either less than zero or "practically zero".

Figure 5.3 shows the distribution of sample returns, with a red line at the mean return. Note that there are several spikes for return less than zero. This is because tipsters are only allowed to place stakes in discrete increments determined by TennisInsight.

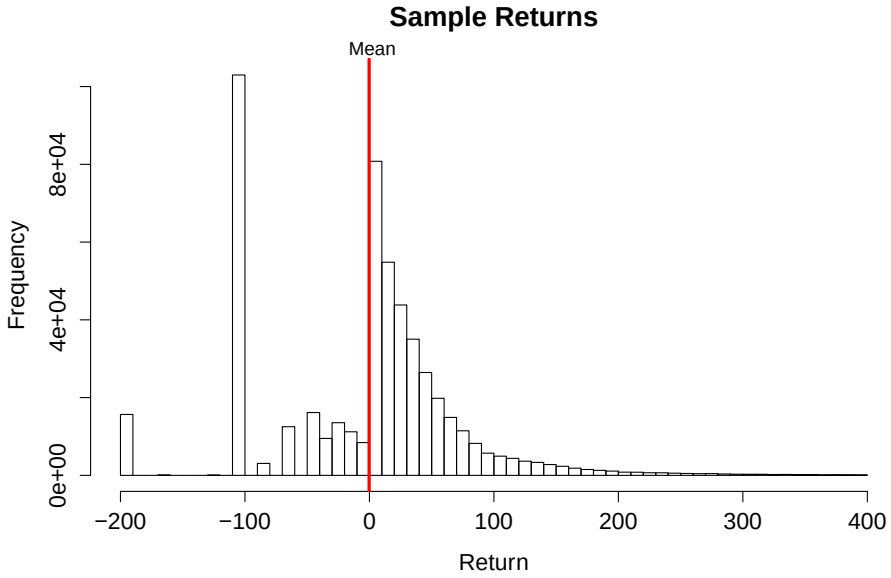

Figure 5.3: TennisInsight Sample Returns

5.1.5 Population Significance

The previous two tests can also be applied to each tipster's tip history individually. To visualise the results of these tests, Figure 5.4 shows a scatter plot, with a point for each tipster corresponding to their ROI and strike rate and coloured according to which tests were significant at the 5% level. Note also the big purple point representing the favourite strategy; at first sight it appears that the majority of significant points are clustered close to the favourite strategy point.

Approximately 70% of the population was found to have a significant strike rate but only 4% have a significant ROI. All tipsters with a significant ROI had a positive ROI, which suggests that all tipsters are attempting to make as much money as possible. All but one of the tipsters with a significant ROI also had a significant strike rate, which seems to say that having a significant ROI is a much stronger condition than having a

significant strike rate.

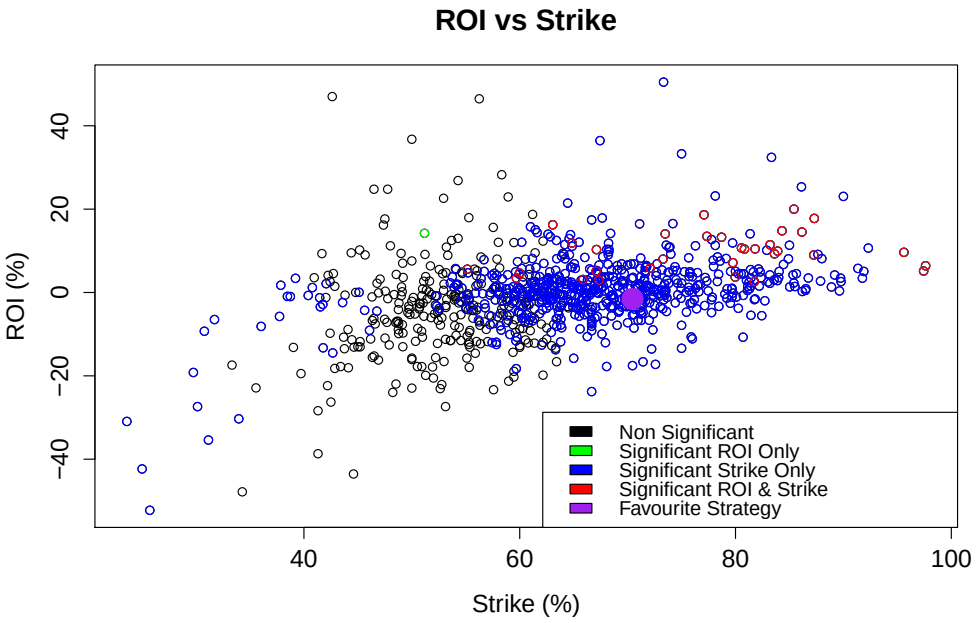

Figure 5.4: TennisInsight ROI vs Strike Scatter

5.1.6 Better Than Favourite Selection?

Now we consider whether the performance of the tipsters in terms of their strike rates and return on investments appear to be better than the favourite selection strategy.

Figure 5.5 shows the distribution of strike rates and ROIs achieved by the tipsters. The red lines indicates the strike rate and ROI which would have been achieved if, for every tip made by all of the tipsters, a tip of equal stake were instead placed on the favourite (referred to as the "favourite strategy"). The favourite strategy ROI turned out to be approximately -1.5% with a strike rate of approximately 70%.

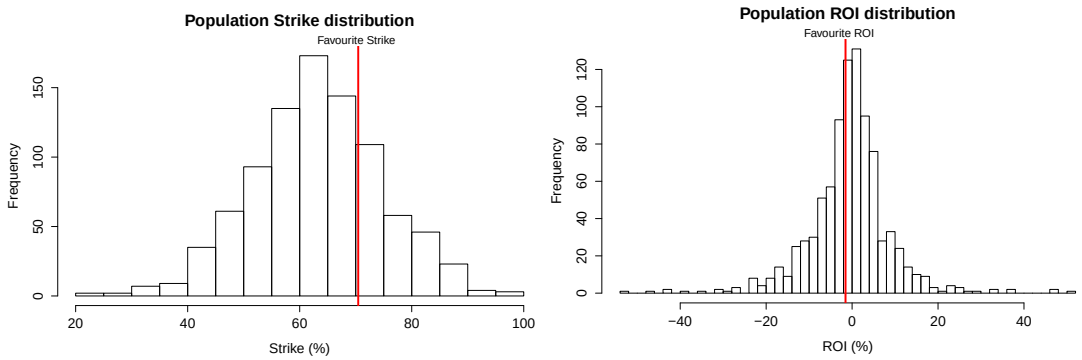

Figure 5.5: TennisInsight Population Performance

From the two distributions in Figure 5.5, it would seem that the population appears to

have an average ROI greater than that which would have been achieved if all tipsters used the favourite strategy but an average strike rate less than the favourite strategy.

To investigate this, we consider using a paired Student's t-test to test the hypothesis that the difference in means is significantly different from zero. For every tipster, we compute the actual strike rate and ROI they achieved and then compute the strike rate and ROI which *would* have been achieved had the tipster followed the favourite strategy, which is then subtracted from the sample strike rate and ROI.

For the ROI data, the mean of the differences is 0.250% with a corresponding 95% confidence interval of $[-0.458, 0.957]$ and $p = 0.489$, so we cannot reject the null hypothesis that the difference in means is equal to zero.

For the strike rate data, the mean of the differences is -9.11% with a corresponding 95% confidence interval of $[-9.66, -8.56]$ and $p < 2.2 \times 10^{-16}$, so we conclude that the difference in means is certainly not equal and in fact the favourite strategy strike rate appears to be significantly higher than the strike rate achieved by the tipsters.

Another way to test whether the sample and favourite strategy distributions differ is by considering the empirical cumulative distribution functions (CDFs) and conducting a Kolmogorov-Smirnov (KS) test which tests whether the maximal difference between two empirical CDFs is significant.

For the ROI data, the KS test resulted in a maximal distance of $D = 0.077$ with corresponding $p = 0.009$, so we conclude that the distributions appear to differ significantly.

For the Strike data, the KS test resulted in a maximal distance of $D = 0.444$ with corresponding $p < 2.2 \times 10^{-16}$, so we conclude that the distributions appear to differ significantly.

Figure 5.6 shows the distribution functions used to perform the tests.

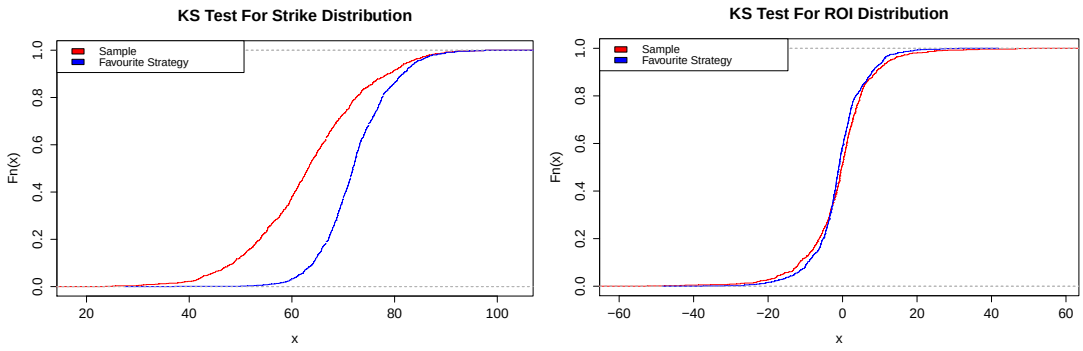

Figure 5.6: TennisInsight KS Tests

Considering both the t-tests and the KS tests, it seems as though the population is *not* behaving how you would expect it to if everyone followed the favourite strategy, however the mean ROI may be similar.

One final way to compare the favourite strategy to that of the tipsters is to consider another paired student t-test and corresponding KS test, this time looking at the *entire* stream of returns which were obtained by *all* of the tipsters (as we did in Section 5.1.4) as well as the stream of returns which would have been obtained had they all followed

the favourite strategy.

The mean of the differences is 1.034 units with a corresponding 95% confidence interval of $[0.759, 1.309]$ and $p = 1.7 \times 10^{-13}$, so we conclude that the mean return does differ significantly and it seems that, on average, the returns obtained by the tipsters are greater than the corresponding favourite strategy returns.

Lastly, the KS test, testing the null hypothesis that the two streams of returns were sampled from different distributions (see Figure 5.7), resulted in a maximal distance of $D = 0.088$ with corresponding $p < 2.2 \times 10^{-16}$, so we conclude that the distributions appear to differ significantly, which agrees with the t-test result.

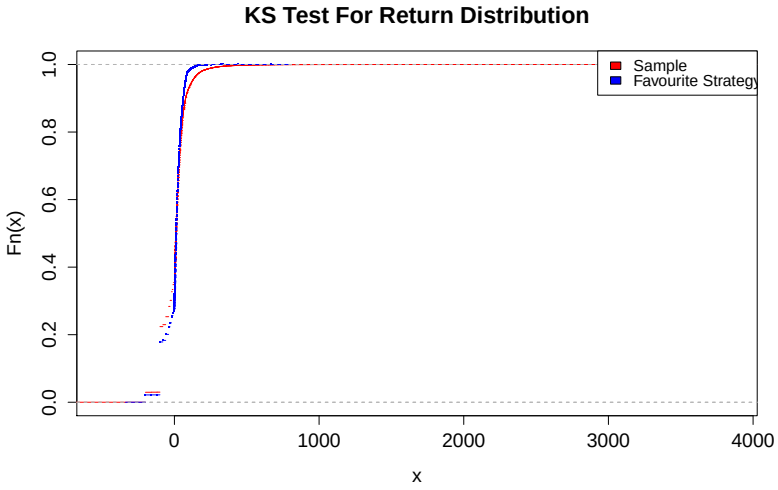

Figure 5.7: TennisInsight KS Test Returns

5.1.7 Summary

Considering the TennisInsight data set, we found that only a small percentage of the members on the site actively participated on a regular basis. Amongst these, the majority achieved a strike rate which was significantly non-random but only a small minority achieved a significant positive return on investment at the 5% level.

The mean return (for a single tip) obtained by the population was found to be significantly different from zero and in fact negative. When considering the performance of the tipsters against favourite selection, we found that there was strong evidence that, on average, the strike rate of the favourite strategy was greater than the strike rate achieved by the tipsters, but we did not find sufficient evidence to conclude that the ROI was different on average. We did however find that the distribution of strike rates and ROIs were significantly different.

Finally, we found that the mean return (for a single tip) obtained by the tipsters was significantly greater than the mean return that would have been achieved under the favourite strategy and additionally the distribution of returns was significantly different.

Given these results, it seems that the population is neither behaving randomly nor simply following the favourite strategy. Although the population as a whole does not achieve a

positive mean return, it does achieve a mean return greater than the favourite strategy which seems to say that they do have a better strategy for winning more (or losing less!) money. Interestingly, this does not coincide with the population having a better strike rate than the favourite strategy on average, which suggests that the strike rate is perhaps not the most important factor in earning money and rather knowing how much to bet on any given match is.

5.2 Analysis

We now go on to evaluate the performance of the meta-tipster, using the filtering methods and classifiers considered in Chapter 4.

5.2.1 Cleaning The Data

As discussed in the previous section, we will not be considering tipsters who made less than 30 tips. This is done for two reasons. Firstly, tipsters who make very few tips were found to be unlikely to remain active for very long and so there is no point considering tipsters who are unlikely to be providing new data in the future. Secondly, a small sample size may give misleading results in the hypothesis tests used to perform a preliminary feature selection considering the tipster strike rates and return on investments.

5.2.2 Look Back Periods

Something which may be different with the real world data that was not the case in the simulated world is the possibility that tipsters may change their strategy at any time. Perhaps a tipster who was once a very good performer now changes to an inferior strategy for example.

Taking this into consideration, I decided to investigate whether this was the case for the TennisInsight data. To do this, I consider various lengths of "look back" periods. A look back period is simply the number of matches that are used to train a classifier; if we are to train a classifier today, how far do we look back at historical data to train the classifier with?

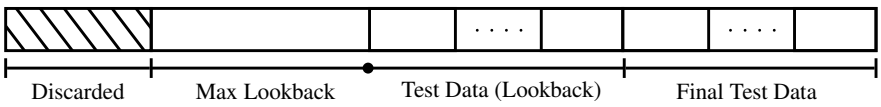

Figure 5.8: TennisInsight Data Split

To do this, I decided upon a maximum look back period to consider of 4000 matches and then split my data set up so that I had 10 blocks of test data for investigating the look back and 10 more blocks of test data for the final testing phase, as shown in Figure 5.8. This 10 block segmentation method means that we only train on data in the "past" relative to our current block of test data. The results were averaged over the 10 blocks.

Note the discarded portion of 2008 data which was considered as a "burn-in" phase for the site.

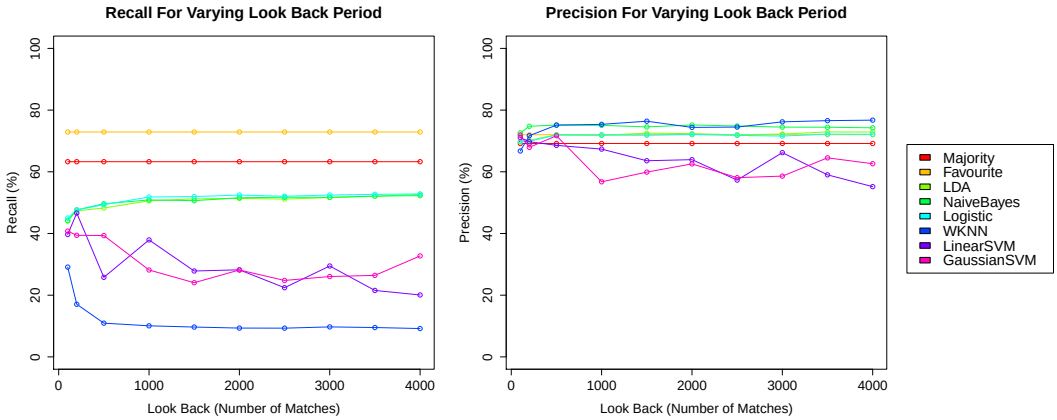

Figure 5.9: TennisInsight Look Back Periods

Surprisingly, I found that there did not seem to be an optimal look back length and instead a simpler "more data is better" pattern emerged, as shown in Figure 5.9. As a result of this, I decided to use the maximum look back period considered in the final testing phase. Also, similar to what happened in the simulated world, the SVM classifiers performed poorly for all the look back lengths considered. The Naive Bayes and WKNN classifiers performed the best, although the recall of the WKNN was exceptionally small.

5.2.3 Final Test Data

Now we move on to the final 10 unseen blocks of test data which span a period covering half of 2010 and all of 2011.

Each block of training data (which looks back 4000 matches from the current test block) was cleaned to ensure only tipsters with at least 30 tips in the training period were considered. As before, we then apply a filtering phase where tipsters who have a significant ROI or strike rate at the 5% level are kept whilst non-significant tipsters are filtered out.

The results are presented in percentage form in Table 5.1.

	Majority	Favourite	LDA	NB	LR	WKNN	LSVM	GSVM
Precision	67.2%	70.1%	71.6%	73.3%	71.2%	67.5%	62.4%	52.7%
Recall	66.6%	73.1%	46.2%	40.1%	47.0%	13.4%	24.9%	24.1%

Table 5.1: TennisInsight Final Recall and Precision (Stake Representation)

Only three of the classifiers (LDA, NB, LR) managed to achieve a precision rate higher than the favourite strategy, the most successful being the Naive Bayes approach which was 3.2% more precise. The poorest performers, as in the simulated world, are the SVM classifiers, with the GSVM barely achieving a precision above guesswork. The recall of all the classifiers is inferior to the favourite strategy, with the best recall achieved by the simple majority vote and the worst by the WKNN classifier at just 13.4%.

I also decided to return to the odds representation, which was found to yield higher precision rates than the stake representation in the simulated world, to see whether this would also be the case with the real world data. The results for the odds representation are presented in Table 5.2

	Majority	Favourite	LDA	NB	LR	WKNN	LSVM	GSVM
Precision	67.2%	70.1%	72.4%	79.4%	71.9%	75.6%	60.3%	54.9%
Recall	66.6%	73.1%	47.9%	37.4%	50.6%	9.5%	20.7%	22.1%

Table 5.2: TennisInsight Final Recall and Precision (Odds Representation)

As was the case with the simulated world, we see an increase in precision for almost all of the classifiers (with the exception of the LSVM). Recall decreases for NB, WKNN, LSVM and GSVM and increases for LDA and LR. The question now is whether this increase in precision will come at the cost of deceased potential earnings as we observed in the simulated world.

5.2.4 Bahamas Or Bust!

Even though some of the precision rates achieved were less than or only marginally better than the favourite strategy, it is still possible that they may be more profitable than the favourite strategy. To test this, we now consider the return on investment that would have been achieved had the advice of the classifiers been followed.

Two stake strategies are considered, a flat staking strategy and a fractional Kelly strategy. The odds considered in calculating profit streams are the "closing odds" provided by TennisInsight, which represent the best odds available across a range of bookmakers at the time the market closed.

The mean over-round (see Appendix A.2) for the closing odds for the final test data matches was found to be 2.69%.

The distribution of over-rounds is shown in Figure 5.10. The two peaks might represent the typical "low risk" and "high risk" over-rounds set by the bookmakers. For example, when two tennis players are evenly matched, the bookmakers may set a higher over-round since they are less certain of who will win, whereas if one player is expected to dominate the other, the bookmakers can afford to set a lower over-round figure.

Flat Stake

Here we consider the simple stake strategy where a unit stake is placed for each win classification. This can be applied to all of the classifiers.

The results for the stake representation are shown in Figure 5.11.

The only classifier to achieve a positive return on investment was LDA, with a ROI of 0.15%. After conducting a Student's t-test on the profit stream however, I found that I could not reject the null hypothesis that the mean profit was equal to zero at the 5% level. The mean profit was 0.0015 units with a corresponding 95% CI of $[-0.0167, 0.0196]$ and $p = 0.8736$.

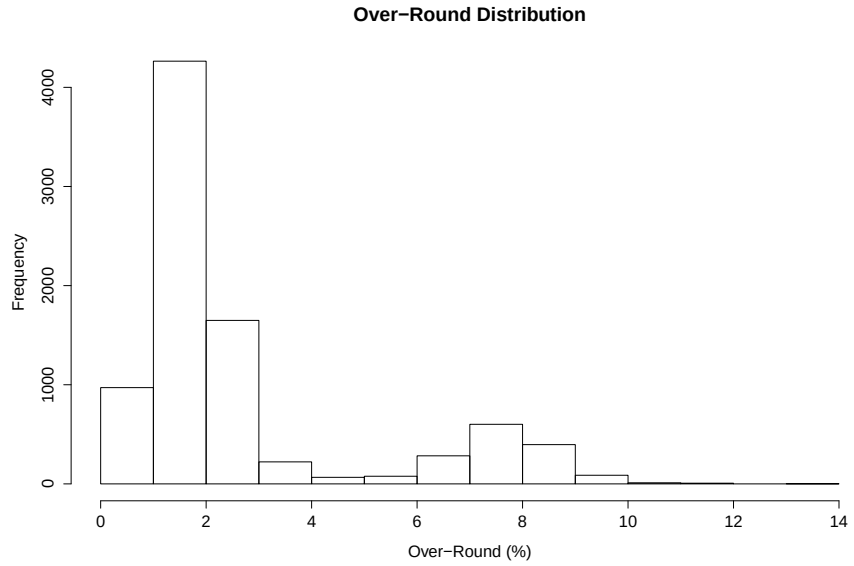

Figure 5.10: TennisInsight Over-Round Distribution

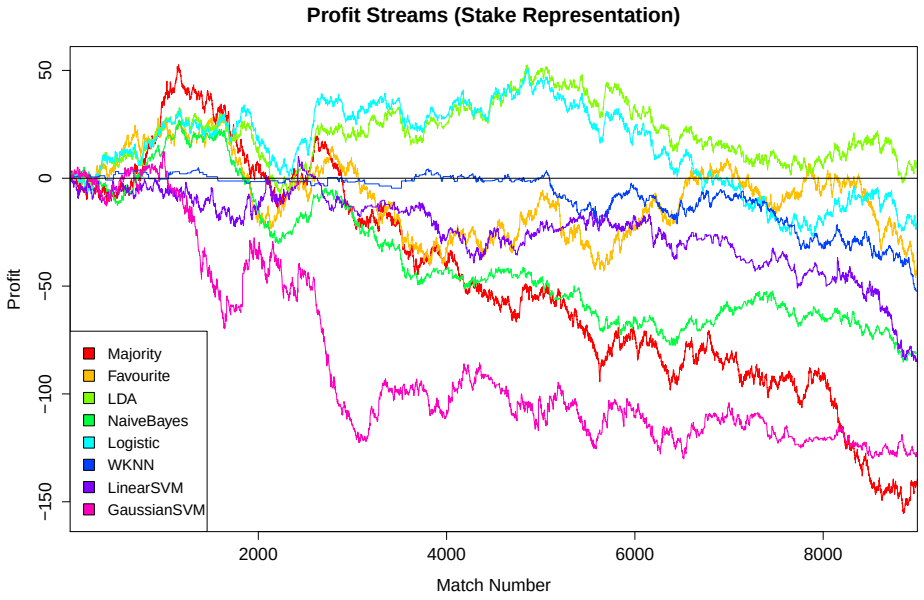

Figure 5.11: TennisInsight Flat Profit Streams (Stake Representation)

After conducting a similar t-test on all of the profit streams, I found that the only significant test was for the simple majority voting. The mean profit was -0.0163 units with a corresponding 95% CI of $[-0.0323, -0.0003]$ and $p = 0.0462$.

Now we consider the odds representation, with results shown in Figure 5.12

The average return on investment has now improved, with all the classifiers hovering closer to the zero profit line. The largest ROI this time is with the Naive Bayes approach, with a ROI of only 0.02%. However, conducting a Student's t-test, I found again that I

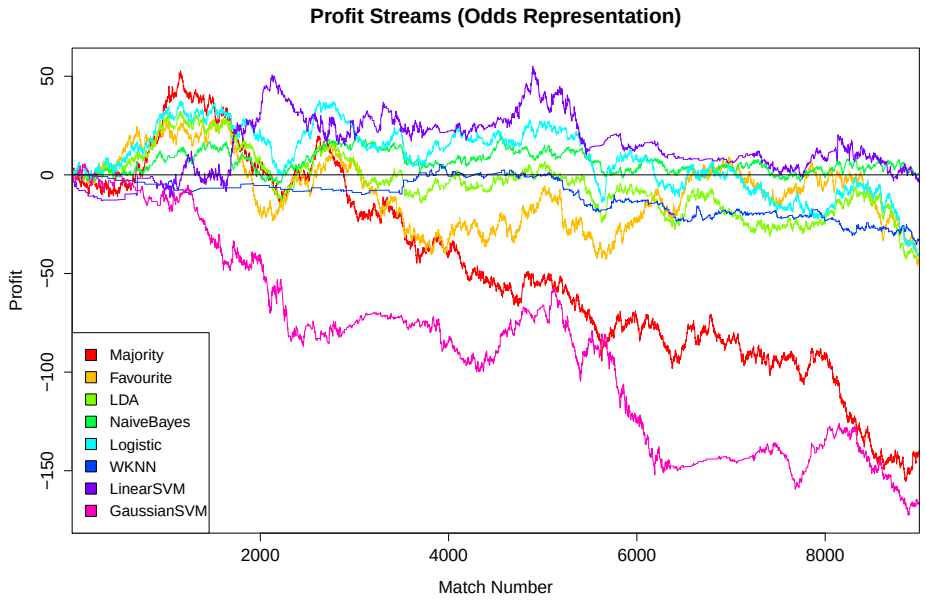

Figure 5.12: TennisInsight Flat Profit Streams (Odds Representation)

could not reject the null hypothesis that the mean profit was equal to zero. The mean profit was 0.0002 units with a corresponding 95% CI of $[-0.0160, 0.0164]$ and $p = 0.9828$.

In addition to the majority voting test being significant, I also found that the GSVM now had a negative mean profit which was significant at the 5% level. The mean profit was -0.0478 units with a corresponding 95% CI of $[-0.0838, -0.0118]$ and $p = 0.0094$.

Considering these tests, we cannot draw any strong conclusions about which representation yielded a better profit; in both cases the only significant tests were for streams with a negative mean profit. For the streams with a positive mean profit, we are still unsure if this could be just due to chance and in fact the mean profit is zero or perhaps negative.

To put things in perspective, let us consider the expected return of a "random strategy" where for every match we have a 50% chance of correctly predicting the outcome.

The expected mean profit of such a strategy is given by:

$$\frac{1}{2m}\left\{\sum_{i=1}^{m} o_i\right\} - 1$$

where m is the number of matches and o_i is the winning decimal odds for match i. See Appendix B.3 for a derivation of this quantity.

For the final test data, this amounted to an expected mean profit of -0.0811 units, corresponding to an expected ROI of -8.11% (since the random strategy placed a bet on *every* match) and an expected total profit of -700.15 units.

It turned out that *all* of the strategies achieved a mean profit significantly greater than -0.0811 units at the 5% level. This suggests that the classifiers are unlikely to have achieved their profit streams "at random".

Kelly Stake

The second stake sizing strategy considered was a fractional Kelly strategy (see Section 2.9.1). This was only applicable to the logistic regression and Naive Bayes approaches, which provide a probability prediction as part of the classification. Unfortunately, even for fractions as low as a tenth, I found that this strategy did not perform well either.

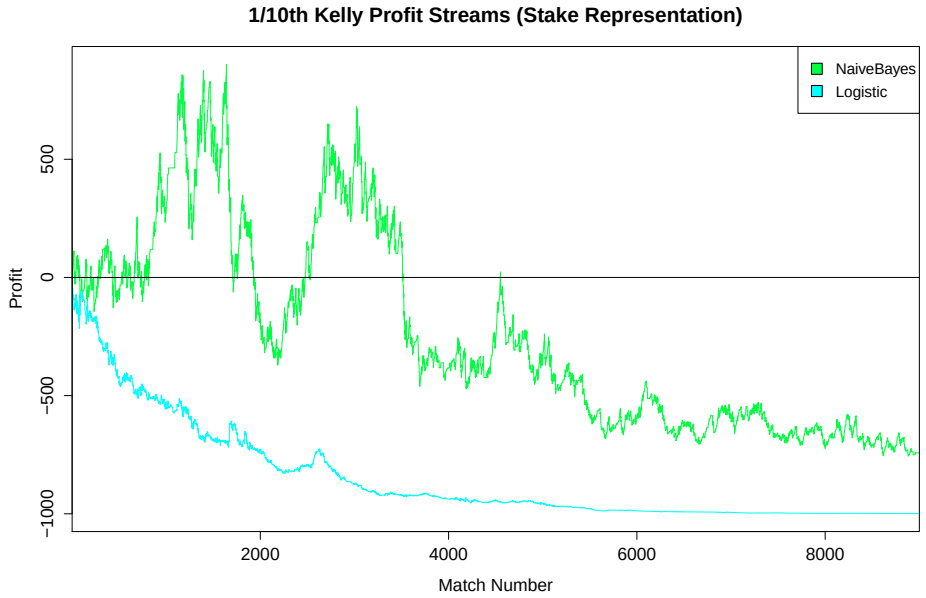

Figure 5.13: TennisInsight Kelly Profit Streams (Stake Representation)

Figure 5.13 shows the profit streams achieved for the stake representation, with a starting bankroll of 1000 units. Although the Naive Bayes approach shows promise early on, almost doubling the original bankroll around the 1200 match mark, ultimately both classifiers lose out in the end. The logistic regression classifier actually loses almost the entire bankroll by around match 7000.

For both classifiers, a t-test on the stream of profits to test whether the mean profit was significantly different from zero at the 5% level was inconclusive in both cases. The Naive Bayes classifier had a mean profit of -0.2255 units with corresponding 95% CI of $[-1.6430, 1.1920]$ and $p = 0.7552$. The logistic regression achieved a mean profit of -0.1447 units with corresponding 95% CI of $[-0.3207, 0.0314]$ and $p = 0.1072$.

A potential reason for the strategy not working as well as you might hope is that the probability predictions provided by the classifiers appear to be far from the true probabilities. To classify correctly we only need to be on the "right side" of the 50% mark, but an accurate probability prediction is clearly much harder to obtain.

Figure 5.14 shows the profit streams achieved using the odds representation. The logistic regression classifier again loses a large proportion of the bankroll in this case, however the Naive Bayes classifier manages to stay hovering around the zero profit mark.

This time, a t-test on the stream of profits to test whether the mean profit was significantly different from zero at the 5% level was inconclusive for the Naive Bayes clas-

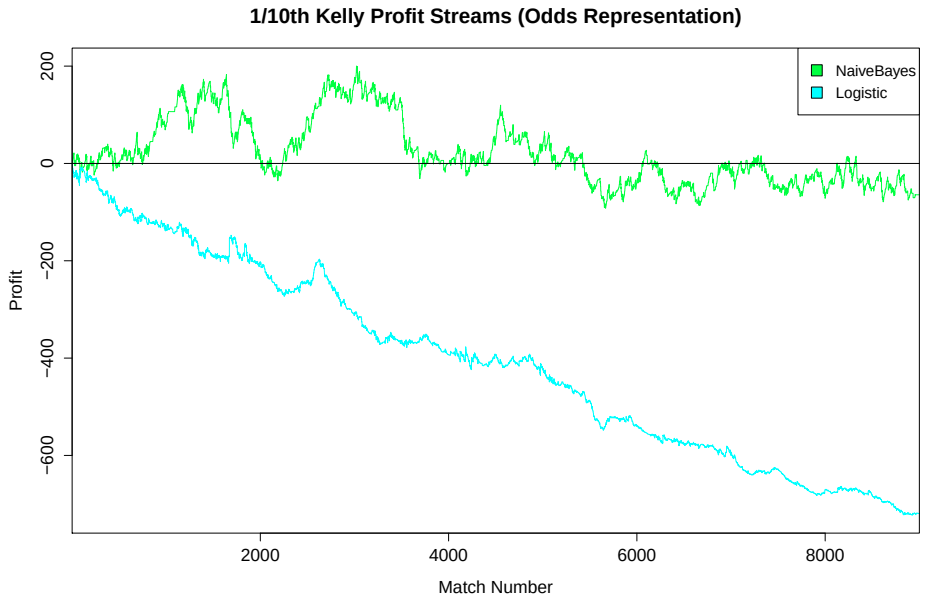

Figure 5.14: TennisInsight Kelly Profit Streams (Odds Representation)

sifier, which achieved a mean profit of -0.0164 units with corresponding 95% CI of $[-0.1791, 0.1462]$ and $p = 0.8430$. However for the logistic regression classifier, a mean profit of -0.0981 units was achieved with corresponding 95% CI of $[-0.1433, -0.0528]$ and $p = 2.2 \times 10^{-5}$, so the mean is certainly not zero in this case and is negative, as we would expect from the profit stream graph.

5.3 One Last Try

Unsatisfied with the returns obtained by the classifiers thus far, I decided to have one last try at finding a classifier that would perform better. In the simulated world, the WKNN classifier performed the best across the board, however the performance on the real world data was not as impressive. I thought it might be worth looking into further and this led me to a modified k-nearest neighbour approach.

5.3.1 Modified Weighted K Nearest Neighbour

I decided to investigate a modified weighted k-nearest neighbour approach (MKNN) which uses the entire case base but has a distance weighting function given by:

$$w(d) = \frac{1}{(d+\alpha)^{\beta}}$$

where $\alpha \in (0, \infty)$ can be used to increase the weighting of close neighbours and $\beta \in (0, \infty)$ can be used to decrease the weighting of neighbours which are further away.

Points with $d = 0$ have weight $\alpha^{-\beta}$ and as $d \to \infty$, $w(d) \to 0$. Figure 5.15 shows the effect of α and β on the distance weighting.

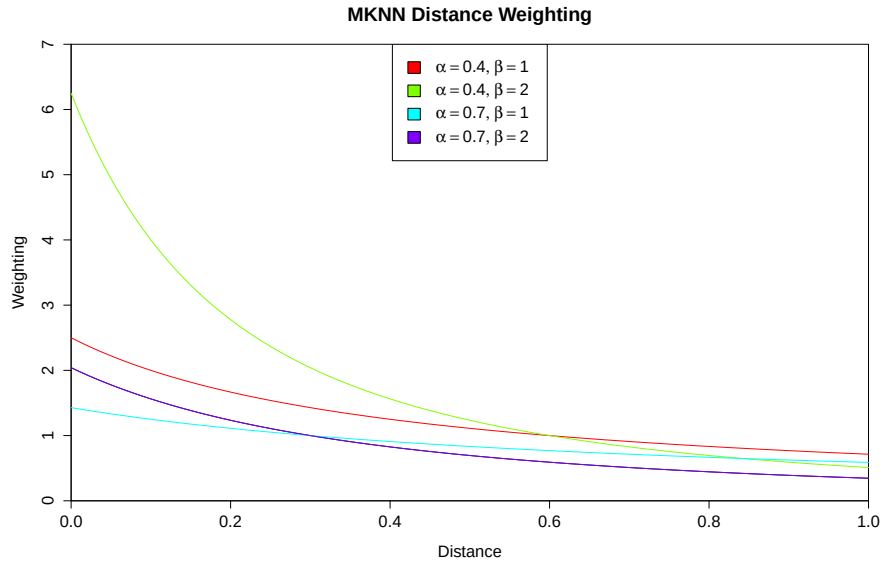

Figure 5.15: MKNN Distance Weighting

The parameters (α, β) are learned by cross validation on the training set over a coarse to fine grid.

5.3.2 Performance

After experimenting with the various representations considered earlier in the project using the look back test data, I ultimately found the most suited for the MKNN approach to be the odds representation, which achieved both the best precision and ROI.

As before, I then considered the final test data and found that the MKNN approach obtained a precision rate of 88.6% and a recall rate of only 9.1%. This was well in excess of the best precision previously achieved by the WKNN approach under the odds representation (which achieved precision of 75.6% and recall of 9.5%).

The profit stream under a unit stake strategy is shown in Figure 5.16, with the favourite and majority strategy streams for comparison. The MKNN stream exhibits a lot less variance than the other streams we have seen and appears to have an upwards trend, achieving a return on investment of 1.44%. Performing a t-test as we have done before to test whether the mean profit is significantly different from zero was unfortunately inconclusive. The mean profit was 0.0144 units with a corresponding 95% CI of $[-0.0122, 0.0411]$ and $p = 0.2879$. It is worth noting, however, that this is the smallest p value we have achieved alongside a positive mean profit.

It turned out that *all* of the win classifications made by MKNN were also favourites, with the maximum odds taken being 1.74. This made me question whether all MKNN was doing was simply waiting for very short favourites. To investigate this I considered the distribution of all favourite odds less than 1.74 in comparison with the distribution of odds taken by MKNN, shown in Figure 5.17.

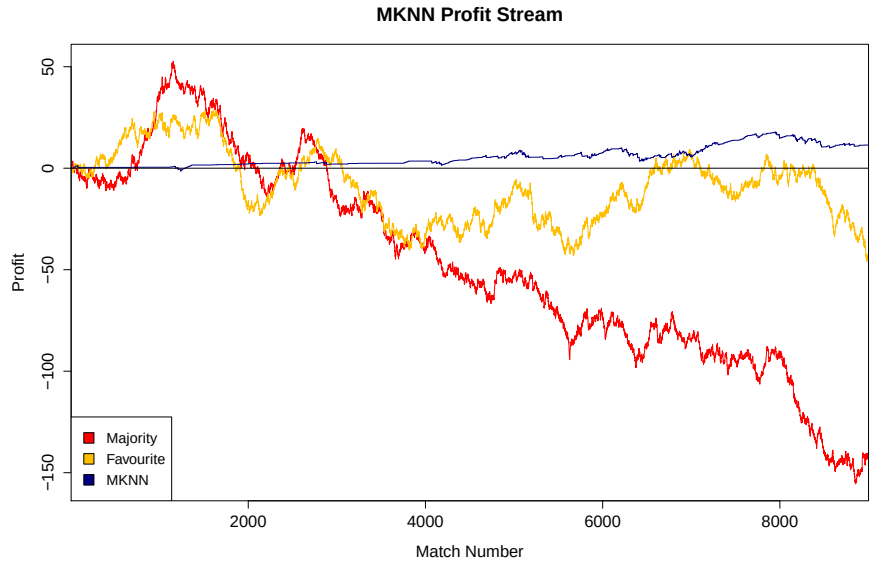

Figure 5.16: TennisInsight MKNN Profit Stream

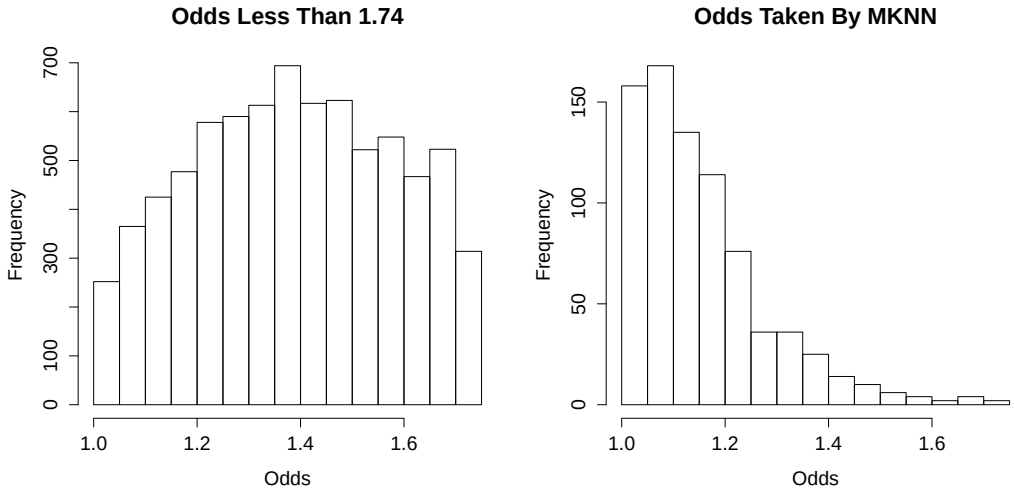

Figure 5.17: Odds Taken By MKNN

It seems that MKNN is not just classifying *all* cases where the favourite odds are less than some threshold. The majority of odds taken are very short odds. This is not a bad thing; bookmakers will tend to give better odds for favourites and worse for long-shots, the so-called "long-shot bias".

5.3.3 Crowd Sourcing A Staking Strategy

As we found out in Section 5.1.6 that the tipsters achieved a significantly worse strike rate yet a significantly better mean profit than the favourite strategy, this may suggest that the tipsters have some knowledge on "how much to bet".

To investigate this, I considered the MKNN approach and rather than placing a unit stake per win classification, I instead staked a measure of the average stake placed by all the tipsters that agreed with this classification. I considered both the mean and median as measures of average, since the mean may be skewed by outliers.

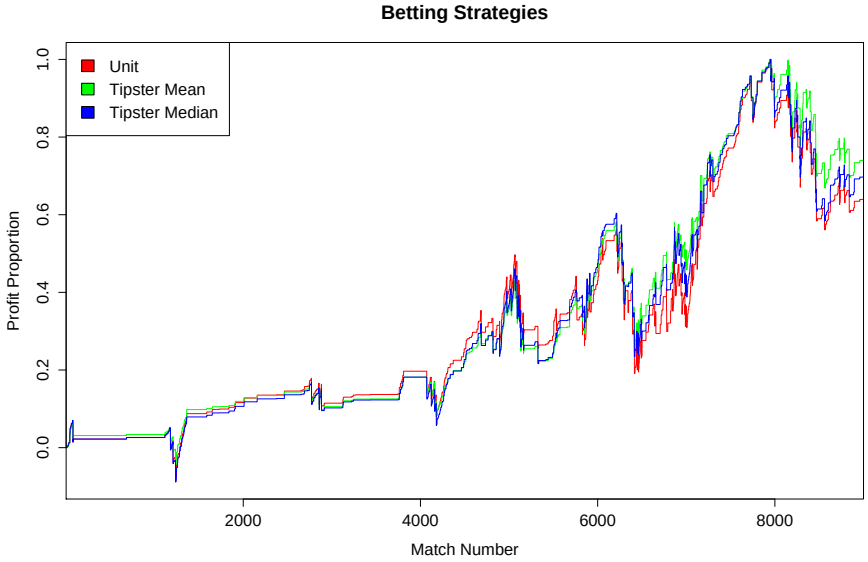

Figure 5.18: Crowd Sourcing A Stake Strategy

Figure 5.18 shows how following the average stake strategy of the tipsters does yield a better return on investment in this case. Note that the scales of the streams are normalised so that 1.0 corresponds to the maximum profit achieved under that stake strategy.

The flat strategy ROI was 1.44% whilst the median strategy resulted in a ROI of 1.70% and the mean strategy a ROI of 2.16%.

Performing once again the now familiar t-test to see if the mean profit was different from zero was unfortunately inconclusive at the 5% level in both cases. The median approach achieved a mean profit of 1.5654 units with a corresponding 95% CI of $[-0.9798, 4.1006]$ and $p = 0.2258$. The mean approach achieved a mean profit of 1.6444 units with a corresponding 95% CI of $[-0.6105, 3.8993]$ and $p = 0.1526$.

5.4 What About The Tipsters?

One final thing to consider is: how did the tipsters fare over the same period of final test data? Figure 5.19 shows the profit streams of the tipsters who achieved a mean profit significantly different from zero using the t-test at the 5% level.

There were only five such tipsters and, surprisingly, they achieved very high return on investments, with the best being 16.26% by "SecondWalz" which had $p = 0.0274$ in the associated t-test. The worst return on investment obtained was a still respectable 5.15% by "uncjrod" which had $p = 0.0188$ in the associated t-test.

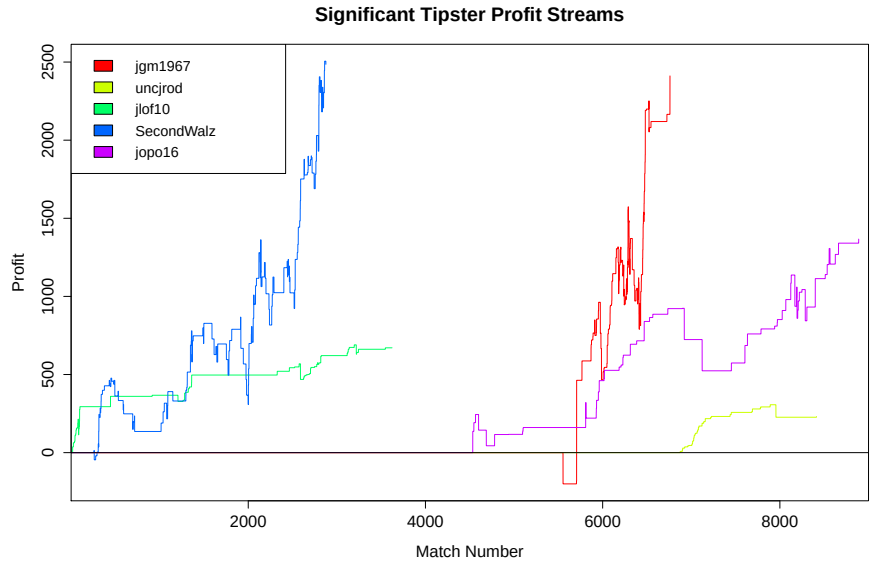

Figure 5.19: Significant Tipster Profit Streams

None of the five significant tipsters remained active over the entire test period, which is interesting because there were tipsters who did remain active over the entire period who did not show up as being significant. In hindsight, it would seem obvious to have just copied these significant tipsters and made a fortune, however in reality we do not have the luxury of seeing the bigger picture at earlier time steps. For example, although these five tipsters were significant over the entire test period, over shorter intervals they may not have been significant and in fact other tipsters who were not significant over the entire test period may have shown up as being significant over these shorter periods.

This raises another important question. I decided to break up the test data into 10 blocks, akin to 10 fold cross validation, which meant that the classifiers were trained at the beginning of each block and used to classify test examples until the end of that block. But what if something game-changing happened *within* the block? Ideally you might retrain the classifier over much shorter intervals, the most extreme case being retraining after each new data point is encountered. This is a very time consuming process. Most notably, each retraining requires relearning the best hyper-parameters. This is discussed further in Section 6.2.2.

6

Discussion

In this final chapter, we summarise the achievements and findings of the project in Section 6.1 and provide several suggestions for future work in Section 6.2.

6.1 Summary Of Findings

We set out to see if we could spot the wisdom in the crowds and extract reliable information from it.

By conducting experiments in the simulated world, we found that it was possible to identify tipsters who were behaving non-randomly through the use of hypothesis tests performed using historical data on the strike rates and profit streams of the tipsters.

The Naive Bayes and weighted K nearest neighbour classifiers consistently performed the best out of all the classifiers considered in each of the experiments, managing to achieve precision rates in excess of those achieved by the favourite strategy. We found the least successful approaches be the linear and Gaussian support vector machines, sometimes behaving little better than guesswork.

Considering the TennisInsight population, we found that the 17% minority of active tipsters on the site appeared to neither be acting randomly nor simply following the favourite strategy. Obtaining a significant positive return on investment was found to be a much stronger condition than having a significantly non-random strike rate, with only 4% of active TennisInsight tipsters achieving a significant ROI whilst 70% had a significant strike rate.

The best performance obtained by the classifiers on the TennisInsight test data set was that of the modified weighted K nearest neighbour approach, with a precision rate of 88.6% and recall rate of 9.1%. This approach also achieved the highest observed ROI of 2.16% using a crowd sourced stake strategy, in the face of a mean over-round of 2.69%. Unfortunately, this positive return was not found to be statistically significant and so it remains to be seen whether the approach would yield a profitable long-term strategy.

6.2 Future Work

Due to both time constraints and the intended scope of the project, some of the ideas that emerged throughout the project were not looked into. The following is a summary of such ideas as a suggestion for possible future work.

6.2.1 Alternative Approaches

There are a wealth of alternative approaches to binary classification which were not explored fully. Given the success of the Naive Bayes and WKNN style approaches, it may make sense to look into methods which build on these. For example, distance metric learning for the k-nearest neighbour classifier[39] and a weighted one order dependency extension to the Naive Bayes classifier[19].

Other possibilities include ensemble classifiers such as random forests[5] and other general ensemble techniques[31] such as AdaBoost, however these may be unsuitable due to the noisy nature of the data.

6.2.2 Shorter Retraining Intervals

As noted in Section 5.4, something else we did not investigate is the effect shorter retraining intervals might have on classifier performance. To clarify, the retraining interval is the length of time we use a trained classifier for. In the final test phase, we split the data into 10 equally sized blocks, but why not 100 or even 1000 blocks? The only real limitation here is the time taken to retrain classifiers. If it took t units of time to train for 10 blocks, it will take $10t$ units of time to train for 100 blocks, for example.

In reality, where the meta-tipster is being used to continuously predict the outcomes of events, a practical approach might be to retrain overnight using the previous day's additional data which would keep the computation feasible.

6.2.3 Markets With Several Outcomes

Throughout the project, we focussed on the case of a two outcome market, as is the case with the TennisInsight data set, yet much of the project is applicable to markets with more than two outcomes. Both the simulated world and the data representations considered are capable of dealing with markets with a variable number of outcomes.

The difference now is that the data set will be unbalanced. For a market with m outcomes, for each instance of that market, we will have $(m-1)$ "lose" data points and only 1 "win" data point. This also means that we go from having equal prior probabilities of $\frac{1}{2}$ in the two outcome case to prior probabilities of $\frac{1}{m}$ for a win and $\left(1-\frac{1}{m}\right)$ for a lose in the m outcome case.

The classifiers would have to take this imbalance into account, for example in the standard K nearest neighbour algorithm it no longer makes sense to give all data points equal weighting and we may instead weight win data points more than lose data points.

6.2.4 We Need More Data!

The main problem we faced when considering the profit streams of the classifiers is that the majority of the hypothesis tests conducted were inconclusive at the 5% level. The only real solution to this is to conduct the tests using more data. Since the original TennisInsight data collection was performed, at the time of writing there are now a further 6 months worth of data (from January to June 2012) which has not been considered.

As well as the TennisInsight data set, it is certainly worth considering other sources of data. OLBG is a possibility, however the 45 day tip history and sensitive flood detection pose problems. TennisInsight has launched another (beta) site SportInsight[1], covering professional hockey, baseball, rugby and football which happens to use the same site template as TennisInsight, which would make data collection an easier task given the current scrapers.

In addition to these kinds of tipping sites, there are hundreds of independent tipsters who run their own "blogs" providing sporting tips. Unfortunately, sourcing such sites is difficult and often the posting style is inconsistent which would make data collection itself a challenge. Although, the site Blogabet[2] verifies and indexes a number of blogs run independently by tipsters with a standard post template for tips, which might make it worth investigating further. Blogabet covers a wide range of sports including tennis, football, rugby and boxing.

Collecting data from more than one source will introduce new challenges involving identification of data. For example, what if the name of a tennis player is spelt incorrectly in one source but not in another? Such small discrepancies could be overcome by having one "trusted" source for data and then mapping any discrepancies in other sources to the trusted source's identifier.

6.2.5 Bookmakers As Tipsters

Something else which may be worth investigating is considering the bookmakers themselves as tipsters. Although the majority of the time the odds set by all bookmakers for a market will agree and not vary a great deal, perhaps when there is variation this has some hidden meaning. Maybe Ladbrokes[3] always has inflated opening odds for the outsiders whilst William Hill[4] has inflated opening odds for favourites.

Alternatively, a time series analysis of how the odds set by various bookmakers tend to change from the market opening time to closing time for a specific kind of event has the potential to be revealing. If a particular combination of odds drift rates had a meaning attached to it and this meaning could be detected early on, this information could be exploited.

[1] http://www.sport-insight.com
[2] http://www.blogabet.com/directory
[3] http://www.ladbrokes.com/
[4] http://www.williamhill.com/

6.2.6 A New Tipping Website

We found through experimentation with the simulated world that a more efficient way for us to extract knowledge from the tipsters arises if the tipsters always tell us their predictions, even if they deem a selection as having "no value". To my knowledge, at the time of writing, no existing tipping site implements such a tipping style.

As such, another possible extension is to set up a tipping website which does exactly that. I suggest three possible ways of achieving this:

Direct Percentages
: Tipsters provide percentage predictions directly by adjusting a series of sliders corresponding to the probability they think each runner has of winning.

Lowest Odds Acceptable
: Tipsters provide the lowest odds they would consider for placing a bet on the runner in question.

Be The Bookmaker
: In a virtual world, each tipster gets to lay a book on real world events of their choice and simultaneously place bets with other user controlled bookmakers.

Perhaps the most potential for entertainment for the tipsters lies with the "be the bookmaker" option, which operates almost like a non-anonymous virtual betting exchange. Since it is unrealistic for one tipster to lay a book for *all* events every day, perhaps tipsters could form teams and operate under a single name and compete against other teams.

All of these options would be community driven. The incentive for tipsters is both "just for fun" and also there is the potential for (monetary or otherwise) prizes to be awarded to the best performing tipsters.

A

Odds

This appendix provides details regarding odds and bookmakers. Section A.1 gives details on the different ways odds are typically represented and how to convert between them. Section A.2 discusses the concept of the over-round which is introduced by bookmakers into the odds so they can make a profit.

A.1 Representation

A.1.1 Fractional

Typically odds are represented in the fractional form a/b (read "a to b"), which means that a winning stake of b will result in a profit of a.

For example, odds of 4/1 would imply that the punter stands to make £400 profit on a £100 stake. Note that the original stake is always received in addition to the profit, so the total return in this case is £500.

A.1.2 Decimal

Decimal odds are represented as a decimal d, which means that a winning unit stake will result in a *total* return of d.

For example, odds of 3.0 would imply that the punter stands to receive £300 return for a stake of £100 (i.e. the £100 stake plus a profit of £200).

A.1.3 Conversion

Fractional and decimal odds are related by the simple rule that x in fractional odds corresponds to $x + 1$ in decimal odds.

For example, fractional odds of 4/1 correspond to decimal odds of 3.0.

A.1.4 Implied Probability

The implied probability p is one over the decimal odds d.

For example, the implied probability for odds of 5.0 is $1/5 = 0.2$.

A.2 Bookmakers

A.2.1 Over-round

This concept is best explained by example. For the win market of a three horse race, a bookmaker may have set odds of 2.0 for each runner. This means that the implied probability of each runner is $1/2 = 0.5$. Note that the sum of these is 1.5.

If the bookmaker were to take an equal amount of money on each runner, he would receive 3 units and pay out only 2 units. The profitable extra unit amounts to 50%; the book is said to be 50% "over-round".

In the case of no over-round, the sum of the implied probabilities will be 1.

A.2.2 Normalised Implied Probabilities

The sum of implied probabilities can be used to calculate normalised probabilities which will always sum to 1:

$$p_i^* = \frac{p_i}{\sum_{i=1}^{n} p_i}$$

where p_i^* are the normalised implied probabilities and p_i are the non-normalised implied probabilities.

This allows us to convert odds to probabilities in the presence of over-round.

B

Details

This appendix gives further details which were omitted from the main text. The Naive Bayes classifier is discussed in Section B.1. Details on how LDA is used for classification and feature selection are in Section B.2. Finally a derivation of the expected mean return of a random strategy with flat staking can be found in Section B.3.

B.1 Naive Bayes Classifier

We consider a model with n independent variables $a_1, \ldots, a_n$ which contribute to the probability that a data point belongs to a certain class c.

Bayes' Theorem states that:

$$\Pr(C = c \mid A_1 = a_1, \ldots, A_n = a_n) = \frac{\Pr(C = c) \cdot \Pr(A_1 = a_1, \ldots, A_n = a_n \mid C = c)}{\Pr(A_1 = a_1, \ldots, A_n = a_n)},$$

and we make the naive assumption that

$$\Pr(A_1 = a_1, \ldots, A_n = a_n \mid C = c) = \prod_{i=1}^{n} \Pr(A_i = a_i \mid C = c).$$

Naive Bayes classification of a new data point $(a_1, \ldots, a_n)$ is then given by:

$$\text{classify}(a_1, ..., a_n) = \arg\max_c \left\{ \Pr(C = c) \cdot \prod_{i=1}^{n} \Pr(A_i = a_i \mid C = c) \right\}$$

where c ranges over all the possible classes.

The prior probabilities $\Pr(C = c)$ are either taken as the relative prevalence of each class in the training set or may be specified.

The feature probabilities are taken as:

$$\Pr(A_i = a \mid C = c) = \frac{\#(c, i, a) + sp}{\#c + s}$$

where $\#(c, i, a)$ denotes the number of training examples where attribute i is a with class c and $\#c$ denotes the number of training samples where the class is c.

The s and p terms are used to cope with cases where $\#(c, i, a) = 0$, so s is used as a regularisation parameter used to weight how confident we are in the prior p for $\Pr(A_i = a \mid C = c)$. Taking $s = 0$ reduces to the maximum-likelihood estimates.

B.2 Linear Discriminant Analysis

B.2.1 Details On Method

The first step in LDA is to find two scatter (un-normalised covariance) matrices, referred to as the "between class" and "within class".

Suppose we have C different classes, with N_c data points in class c, each with a sample class mean denoted $\bar{\mathbf{x}}_c$ and sample covariance matrix denoted Σ_c. Additionally we have a grand sample mean across all classes, denoted $\bar{\mathbf{x}}$.

Then the between class scatter matrix is defined as:

$$S_b = \sum_{c=1}^{C} N_c(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^T$$

And the within class scatter matrix is defined as:

$$S_w = \sum_{c=1}^{C} (N_c - 1)\Sigma_c$$

The objective of LDA is then to find a projection matrix Φ_{LDA} which maximises the ratio of the determinant of S_b to the determinant of S_w, which can be written as:

$$\Phi_{\text{LDA}} = \arg\max_{\Phi} \frac{|\Phi^T S_b \Phi|}{|\Phi^T S_w \Phi|}$$

and this ratio is known as "Fisher's criterion".

The optimal projection Φ_{LDA} is the solution to the eigenvector problem:

$$S_b\Phi - S_w\Phi\Lambda = 0$$

so that Φ_{LDA} is the matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues.

Note that there will be at most $C - 1$ eigenvectors with non-zero real corresponding eigenvalues, so in the case of $C = 2$, the projection becomes one dimensional.

B.2.2 Classification

To use LDA as a classifier, we simply compute the projection matrix Φ_{LDA} and use this to find the projected class means, denoted $\hat{\mathbf{x}}_c$ for each class c.

Now, to classify a new data point $\mathbf{x}$, we find its projection $\mathbf{x} \mapsto \hat{\mathbf{x}}$ and then find the class which has projected class mean closest to the projected point. So we classify x as the class $\hat{c}$ such that:

$$\hat{c} = \arg\min_{c} \|\hat{\mathbf{x}}_c - \hat{\mathbf{x}}\|$$

The distance function is typically taken as being the Euclidean distance. This is what is used in the library the project uses to perform LDA classification.

B.2.3 Feature Selection

To use LDA as a feature selection method, we consider the feature loadings. Since we are dealing with a two class problem, there will be only one non-zero eigenvector, with dimension equal to the number of features which we denote as d. This means that the dimension of projected points will also be one.

Each entry e_i of this eigenvector corresponds to how much one unit of feature i contributes to the projection since the projection of a point $\mathbf{x}$ to the one dimensional LDA space is given by:

$$\sum_{i=1}^{d} x_i \cdot e_i$$

By considering the magnitudes $|e_i|$, this gives us a measure of "how important" each feature i is. Since each feature is normalised, this comparison is fair to make.

We then simply take the L loadings with the biggest magnitude factor loadings.

B.3 Expected Mean Profit Of The Random Strategy

We consider a set of m matches where each match was won by a runner with decimal odds o_i. For each match, the random strategy has a 50% chance of picking the winner and so making a profit of $(o_i - 1)$ units and a 50% chance of picking the loser and so making a profit of -1 unit, all under a unit staking strategy.

Let X be the random variable for the mean profit of the classifier after m matches and X_i the random variable for the profit for match i. Then we have that:

$$\begin{aligned}
\mathbb{E}[X] &= \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}[X_i] \\
&= \frac{1}{m} \sum_{i=1}^{m} \left\{ \frac{1}{2} (o_i - 1) + \frac{1}{2}(-1) \right\} \\
&= \frac{1}{2m} \left\{ \sum_{i=1}^{m} o_i \right\} - 1
\end{aligned}$$

Bibliography

[1] Herv Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews Computational Statistics*, 2:433 – 459, 2010.

[2] Patric Andersson, Jan Edman, and Mattias Ekman. Predicting the world cup 2002 in soccer: Performance and confidence of experts and non-experts. *International Journal of Forecasting*, 21(3):565 – 576, 2005.

[3] Patric Andersson, Daniel Memmert, and Eva Popowicz. Forecasting outcomes of the world cup 2006 in football: Performance and confidence of bettors and laypeople. *Psychology of Sport and Exercise*, 10(1):116 – 123, 2009.

[4] Daniel Bernoulli. Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23–36, 1954.

[5] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. 10.1023/A:1010933404324.

[6] Andrew Cadman. Can Punters win? Are UK betting markets on sporting events efficiently aggregating information? `http://www.essex.ac.uk/economics/EESJ/SP07/cadman831.pdf`, 2006.

[7] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. Paper available at `http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf`.

[8] Stephen R Clarke and John M Norman. Home ground advantage of individual clubs in english soccer. *The Statistician*, 44(4):509–521, 1995.

[9] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[10] Pdraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. `http://www.csi.ucd.ie/files/UCD-CSI-2007-4.pdf`, 2007.

[11] Jason Dentler. *NHibernate 3.0 Cookbook*. Packt Publishing, 2010.

[12] Bruno Deschamps and Olivier Gergaud. Efficiency in horse race betting markets: The role of professional tipsters. In Donald B. Hausch and William T. Ziemba, editors, *Handbook of Sports and Lottery Markets*, pages 341 – 354. Elsevier, San Diego, 2008.

[13] R A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[14] David Forrest, John Goddard, and Robert Simmons. Odds-setters as forecasters: The case of english football. *International Journal of Forecasting*, 21(3):551 – 564, 2005.

[15] David Forrest and Robert Simmons. Forecasting sport: the behaviour and performance of football tipsters. *International Journal of Forecasting*, 16(3):317 – 331, 2000.

[16] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition (Computer Science and Scientific Computing Series)*. Academic Press, 2 edition, October 1990.

[17] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. Wiley, 2nd edition, 2000.

[18] Chih-wei Hsu, Chih-chung Chang, and Chih-jen Lin. A practical guide to support vector classification. *Bioinformatics*, 1(1):1–16, 2010.

[19] Liangxiao Jiang and Harry Zhang. Weightily averaged one-dependence estimators. In *Proceedings of the 9th Pacific Rim international conference on Artificial intelligence*, PRICAI'06, pages 970–974, Berlin, Heidelberg, 2006. Springer-Verlag.

[20] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[21] Jr. Kelly, J. A new interpretation of information rate. *Information Theory, IRE Transactions on*, 2(3):185 –189, Sep 1956.

[22] MADlib. Naive Bayes Classifier Description. `http://doc.madlib.net/v0.3/group__grp__bayes.html`.

[23] Makridakis, Wheelwright, and Hyndman. *Forecasting: Methods and Applications*, page 492. Wiley, 3rd edition, 1998.

[24] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41 –48, Aug 1999.

[25] Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 536–542, Cambridge, MA, USA, 1999. MIT Press.

[26] David L. Olsen and Dursun Delen. *Advanced Data Mining Techniques*, page 138. Springer, 2008.

[27] Thorsten Pachur and Guido Biele. Forecasting from ignorance: The use and usefulness of recognition in lay predictions of sports events. *Acta Psychologica*, 125(1):99 – 116, 2007.

[28] K Pearson. Pearson, k. 1901. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[29] P Peduzzi, J Concato, E Kemper, T R Holford, and A R Feinstein. A simulation study of the number of events per variable in logistic regression analysis. *Journal of Clinical Epidemiology*, 49(12):1373–1379, 1996.

[30] Maja Pohar, Mateja Blas, and Sandra Turk. Comparison of logistic regression and linear discriminant analysis : A simulation study. *Metodolski Zvezki*, 1(1):143–161, 2004.

[31] R. Polikar. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21 –45, quarter 2006.

[32] Benjamin Scheibehenne and Arndt Brder. Predicting wimbledon 2005 tennis results by mere player name recognition. *International Journal of Forecasting*, 23(3):415 – 426, 2007.

[33] Martin Sewell. Kernel methods. `http://www.svms.org/kernels/kernel-methods.pdf`, Apr 2007.

[34] Jonathon Shlens. A tutorial on principal component analysis. `http://www.snl.salk.edu/~shlens/pca.pdf`, Apr 2009.

[35] Edward O Thorp. Understanding the kelly criterion. In Thorp et al. [36], chapter 36.

[36] Edward O Thorp, Leonard C MacLean, and William T Ziemba, editors. *The Kelly Capital Growth Investment Criterion: Theory and Practice*. Scientific Press, Singapore, 2010.

[37] Edward O Thorp, Leonard C MacLean, William T Ziemba, and Yonggan Zhao. Medium term simulations of the full kelly and fractional kelly investment stategies. In Thorp et al. [36], chapter 38.

[38] Richard Webby and Marcus O'Connor. Judgemental and statistical time series forecasting: a review of the literature. *International Journal of Forecasting*, 12(1):91 – 118, 1996. Probability Judgmental Forecasting.

[39] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. MIT Press, 2006.

[40] Leighton Vaughan Williams. *Longshot bias: insights from the betting market on men's professional tennis*, pages 215–230. Cambridge University Press, 2005.

[41] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(13):37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.