Forecasting price movements in betting exchanges using Cartesian Genetic Programming and ANN

Ivars Dzalbs[1], Tatiana Kalganova[1]

[1] Brunel University London, Kingston Lane, Uxbridge, UB8 2PX, UK
Tatiana.Kalganova@brunel.ac.uk

Abstract. Since the introduction of betting exchanges in 2000, there has been increased interest of ways to monetize on the new technology. Betting exchange markets are fairly similar to the financial markets in terms of their operation. Due to the lower market share and newer technology, there are very few tools available for automated trading for betting exchanges. The in-depth analysis of features available in commercial software demonstrates that there is no commercial software that natively supports machine learned strategy development. Furthermore, previously published academic software products are not publicly obtainable. Hence, this work concentrates on developing a full-stack solution from data capture, back-testing to automated Strategy Agent development for betting exchanges. Moreover, work also explores ways to forecast price movements within betting exchange using new machine learned trading strategies based on Artificial Neuron Networks (ANN) and Cartesian Genetic Programming (CGP). Automatically generated strategies can then be deployed on a server and require no human interaction. Data explored in this work were captured from 1st of January 2016 to 17th of May 2016 for all GB WIN Horse Racing markets (total of 204GB of data processing). Best found Strategy agent shows promising 83% Return on Investment (ROI) during simulated historical validation period of one month (15th of April 2016 to 16th of May 2016).

Keywords: Algorithmic trading, Financial series forecasting, Betting exchange

1 Introduction

People have loved to bet on various events since the beginning of written history [1]. In particular, many people are gambling on sporting events. In past few years gambling has grown into multi billion industry. In the UK alone, remote (online) gambling sector generated £4.47bn (April 2015 - March 2016). Out of that, online betting and betting exchanges generated total of £1.72bn Gross Gambling Yield (GGY) [2]. This have led to increased interest in forecasting sporting event outcomes using mathematics and statistics for years.

With the rise of betting exchanges in 2000, punters can not only take bets, but also offer their own, in peer to peer fashion. Following the concept of more familiar stock exchange markets. This have attracted a new sort of customer, the full time, high-volume trader who buy and sell odds just like financial traders buy and sell stock or trade on foreign currency exchange. Although, the value of betting exchanges cannot be compared to financial markets, Betfair – the leading betting exchange - processes more than seven million transactions each day – more than all European stock exchanges combined [3].

This research investigates possible advantages of using machine learning with Feed forward multilayer perceptron (MLP) Artificial Neural Networks (ANN) and Cartesian Genetic Programming (CGP) to predict price movements on pre-race GB horse racing markets. The paper has been structured as follow: Section 2 explores current state of the art approaches, Section 3.1 outlines the machine learning and back testing platform and Section 3.2. describes the machine learning models explored. Furthermore, Section 4 reports on the results achieved.

2 Related work

There have been various attempts to predict the outcome of various sporting events using machine learning –dog racing (greyhound) [4] [5], tennis [6] [7], soccer [8] [9] [10], cricket [11] and horse racing [12] [13] [14].

However, little work has been done on forecasting the price movements inside the betting exchanges. Because betting exchange is very similar to traditional stock or currency exchange markets, there have been couple of attempts using stock market strategies and analysis on betting exchange markets. For example, [15] looked at how human behavior affects the price movements and volumes, trading patterns and strategies.

Detailed introduction to the domain of financial time series prediction using artificial neural networks are described in [16] and [17]. Whereas [18] goes into details of financial time series prediction using Cartesian Genetic Programming and ANN hybrid – Neuro Evolution.

Most of the academic papers in betting exchange forecasts, focus on tennis trading, mainly due to hierarchical nature of the tennis markets. [19] investigates set-by-set analysis based on [20] and Markov chains. In contrast, [21] uses artificial neural networks to predict the price movements on in-play markets with a custom cost function. Furthermore, [22] utilizes machine learning to boost the performance of plain strategies in Horse Racing markets, while [23] explores mathematical and statistical dynamics of Horse Racing markets.

The most relevant work has been compared in Table 1. To the best of the author's knowledge, there has not been any work done in price movement forecasting in Horse Racing markets. In addition, datasets used to predict the outcome of the event is very small compared to datasets needed to predict and back-test price movements. This provides unique niche for research which is conducted in this paper.

Table 1. Current state of the art compared

Work	Forecast type	Market type	Number of markets	Approximate number of selections[1]	Machine learning tools used
J. P. A. Santos, 2014 [22]	Outcome	Horse Racing	1 843[2]	15 062	RapidMiner
A. Bunyan, 2014 [24]	Outcome	Horse Racing	14	112[3]	R & WEKA
M. Sipko, 2015 [7]	Outcome	Tennis	6 315	12 630	Logic Regression and ANN
Ø. Norstein, 2008 [21]	Price movements	Tennis	388	776	ANN[4]
Proposed work	*Price movements*	*Horse racing*	***3 040***	***24 964***	*ANN and CGP*

Furthermore, in order to accommodate the machine learning algorithms, an underlying platform is required. There are various commercial trading tools available that offers users to create transactions at betting exchanges. The most popular ones - Market Feeder Professional, Gruss Software, Bet Angel, BFExplorer, Geeks Toy. However, none of these offer integrated machine learning features and only one – Market Feeder Pro allows back-testing. JBet [22] is a betting exchange tool developed in Java that allows users to access, record and replay markets. It also implements machine learning techniques to assist strategy development. SPORTSBET [25] is a framework that is built on an open-source event-driven platform called URBI. It allows dynamic market re-construction and evaluate strategy performance. Although it implements some stochastic search heuristic in order to improve strategy performance, it does not use any machine learning techniques to develop the strategy itself. Since both JBet and SPORTSBET are not available to general public, they cannot be used as the platform for this paper. However, the concepts of both these works are used as basis in developing machine learning and back-testing platform described in this paper.

[1] Each event has number of selections, for example Tennis match has only 2 possible outcomes, Soccer (matched odds) – only 3 outcomes (Team 1 win, Team 2 win or Draw). Horse racing has on average 8 possible outcomes.

[2] Source does not specify total number of markets in the dataset, total number of markets was retrieved from 2014/01/01 to 2014/04/01 for all GB WIN horse racing events.

[3] Assuming each horse market contains 8 selections on average

[4] Feed-forward multilayer perceptron (MLP)

3 Methods and implementation

3.1. Framework

Fig 1 demonstrates the overall framework diagram. At first, technical market data, such as price and volume, is extracted from Betfair API. Data is then transferred to a local server on a weekly basis where it is formatted in open-source Protobuf-net for fast deserialization speeds and processing. Additionally, fundamental data such as horse form and running history is imported into the MySQL database. On the local server, each Strategy gathers the necessary data (if any) and develops a model based on either machine learning (ANN or CGP), statistics or plain strategies. Each model then can be used by Strategy agents to either put bets "live" with Betfair API, or for back-testing purposes within Simulator. Once bets are placed, Analyser displays each Strategy agent's performance based on various metrics such as profit, yield, number of bets and similar.

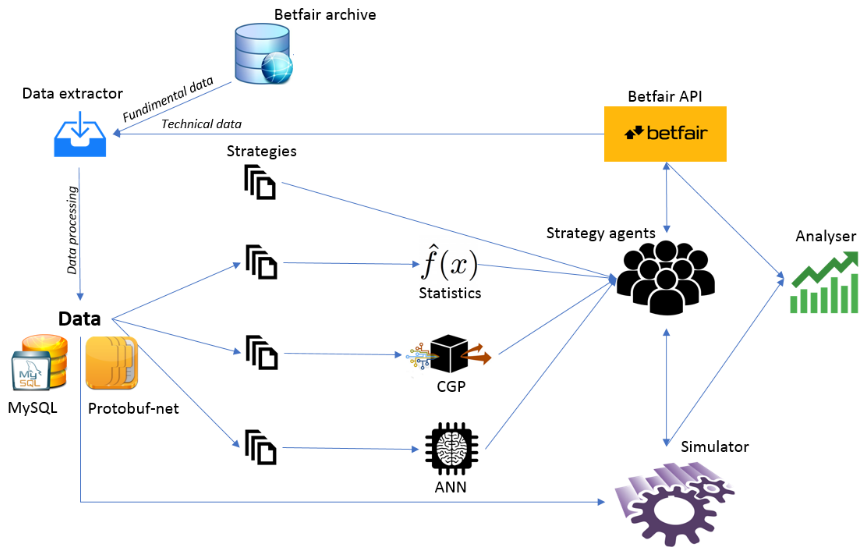


Fig 1. Proposed framework diagram

3.1.1. Data extraction and processing

Most betting exchanges offer different Application Programming Interfaces (API's) for their customers and Betfair is no exception. API products allow to build custom applications with direct access to the Betfair servers. The interface is simple - the client sends a JSON/JSON-RPC request to Betfair server and receives a response. For instance, the client could be requesting the current odd status or making an order request (a bet). Betfair API provided C# example code[5] was used as basis of developing the data collection agent.

Time-stamped price data[6] were collected for around 3000 markets from 1st of January 2016 to 17th of May 2016 using custom built C# application. Data recording for each market begins 3 hours before the scheduled start time. At early stages of the market, odds are saved only every 5 minutes. As the market matures, it is being saved more often. 10 minutes before the race, the market is saved every second and the refresh rate decreases to 0.2 seconds as the market goes in-play. It was found that Protobuf-net[7] serialization format allows the fastest deserialization time, compared to all other C# serializers while maintaining reasonable file size per market. During the recorded period, all GB WIN horse race market consumed 204GB of storage in Protobuf-net format.

3.1.2. Machine learning

Artificial Neural Networks (ANN) have been mainly used for pattern recognition, however, they can also be used for forecasting financial time series. The general idea of forecasting financial and economical time series is to extract features of price and volume information and use that as the input for machine learning tools to predict either overall price direction (classification), specific price at given future time (regression) or a custom loss function such as profit generated. [26] goes into great detail explaining the design of neural network for forecasting financial and economic time series. Feature extraction approaches and output definitions for all models are described in 3.2.1 and 3.2.2.

There are many open source machine learning frameworks for ANN and Encog8 is one of them. This work utilizes Encog C# library, as it has a simple API and is C# based, allowing simple integration into already developed framework. All ANN results in this paper are based on feed-forward multilayer perceptron (MLP) (BasicNetwork class in Encog) with one hidden layer of 50 neurons and ActivationElliott error function (except for Profit, where a custom loss function is used). Furthermore, implementation uses QuickPropagation training with learning rate of 2.

Authors have previously developed a Cartesian Genetic Programming (CGP) algorithm, described in [27], where it was successfully used for time series multi-step ahead forecasting for various industrial time series. Thus, building on top of that knowledge, in this paper we apply CGP for financial time series forecast.
CGP is based on $(1+\lambda)$ evolutionary strategy [28], where during each generation, the number of parents (P) are selected to produce a number of children (C) using mutation operator only. Mutation rate (M) is expressed as percentage of the number of genes that are being mutated. The quality of the chromosome is evaluated using a fitness function. The best chromosomes in the population are defined as fittest members. Furthermore, the fittest member(s) of the population now becomes the

[5] Open source code. Available at: https://github.com/betfair/API-NG-sample-code
[6] Betfair Time-stamped data scheme - https://historicdata.betfair.com/Betfair-Historical-Data-Feed-Specification.pdf
[7] Open source framework. Available at: https://github.com/mgravell/protobuf-net
[8] Encog Machine Learning Framework - http://www.heatonresearch.com/encog/

parent(s) and the process repeats till some end condition is met such as number of generations, fitness value or wall-clock time.

Each chromosome represents one approximation equation. The equation is assembled using arithmetic operators defined by primary arithmetic equations, also called gates. Each gate can have a maximum of number of inputs, J. Each gate type defines one primary arithmetic operation, see Fig 2. Variations of primary arithmetic equations used in evolutionary process are defined in the function library in advance. Therefore, each gate is represented using collection of the following genes:

- The number of inputs to the gate
- Gate type
- Collection of inputs for a specific gate
- Gate constant

Each chromosome is defined as following:

- Collection of gates, G
- Collection of chromosome outputs

The number of chromosome outputs is defined by the number of outputs to be forecasted. The evolutionary process is driven by mutation that is focused on:

- Changing the number of inputs in gate
- Changing the gate function out of the function library
- Changing the gate inputs
- Changing the gate constant

Furthermore, mutation of chromosome outputs is also allowed and the initial population is randomly generated.

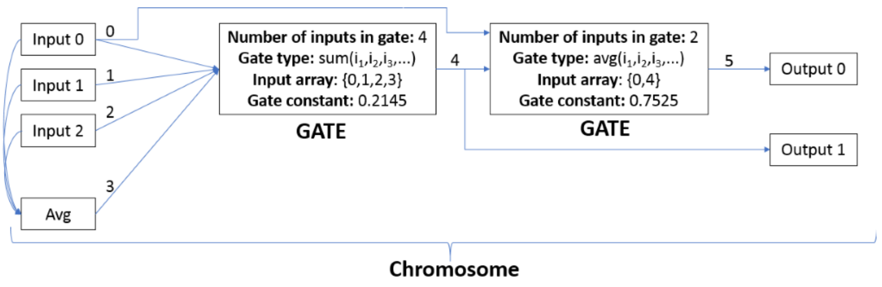


Fig 2. Graphical representation of gate and chromosome. In this example, 3 inputs are used to generate 2 outputs. Only 2 gates are used and one additional input (avg). Output 1 uses first gate (output label 4), whereas Output 0 uses output of the second gate (output label 5).

From previous work in [27], it was concluded that CGP offers competitive results compared to ANN and Support Vector Machines (SVM). The best configuration of CGP found in [27] is used in this work, see Table 2.

Table 2. Cartesian Genetic Programming parameters used for all test cases (except for Profit where custom fitness function is used). More detailed implementation of the architecture, please refer to previous work in **[27]**.

The number of gates (G) in chromosome	50
Mutation rate (M)	5%
The number of children (C)	2
Population size (P)	8
Termination criteria	24 hours elapsed
The maximum number of inputs in a gate (J)	20
Fitness function	Mean Square Error

Furthermore, Function library primitive arithmetic functions used are as follows:

```
sum(i1,i2,i3,...);
multiply(i1,i2,i3,...);
const*sum(i1,i2,i3,...);
subtract(i1,i2,i3,...);
division(i1,i2);
multiply(i1,i2,const);
const*i1^2;
const*i1^3;
i1^const
```

Where *i* corresponds to a double value input in the *Gate* and *const* corresponds to the Gate constant as described in Fig 2. If the output of the gate is not a number or has overflowed, first input value of the gate is passed as the output.

Moreover, Fig 3 shows an overview of machine learning strategy development. Initially, a form of strategy is defined with its corresponding feature set – properties that defines individual entry. These inputs can be either technical data, such as previous price history, volume matched, or fundamental data, such as horse form, previous race history or combination of both. Furthermore, inputs will be used as feature sets for the supervised machine learning stage by either CGP or ANN.

The dataset is then split into three groups – Training set, Testing set and Validation set, 50%, 25% and 25% respectively. The machine learning algorithm trains the model based on the training dataset. In order to estimate model forecasting performance, the trained model is then presented with the test dataset. The prediction performance on test dataset is then evaluated based on overall profit. If performance is not adequate, strategies/model parameters, such as number of gates (CGP) or number of hidden neurons (ANN), are adjusted and the model trained again. If the model is accepted, it is implemented as a Strategy agent. Newly created Strategy agent then goes through the simulation stage on completely unseen data (validation set). If the agent is producing profit, it is being accepted and deployed for live trading. However, if the agent is making a loss, it is discarded and the whole process restarts.

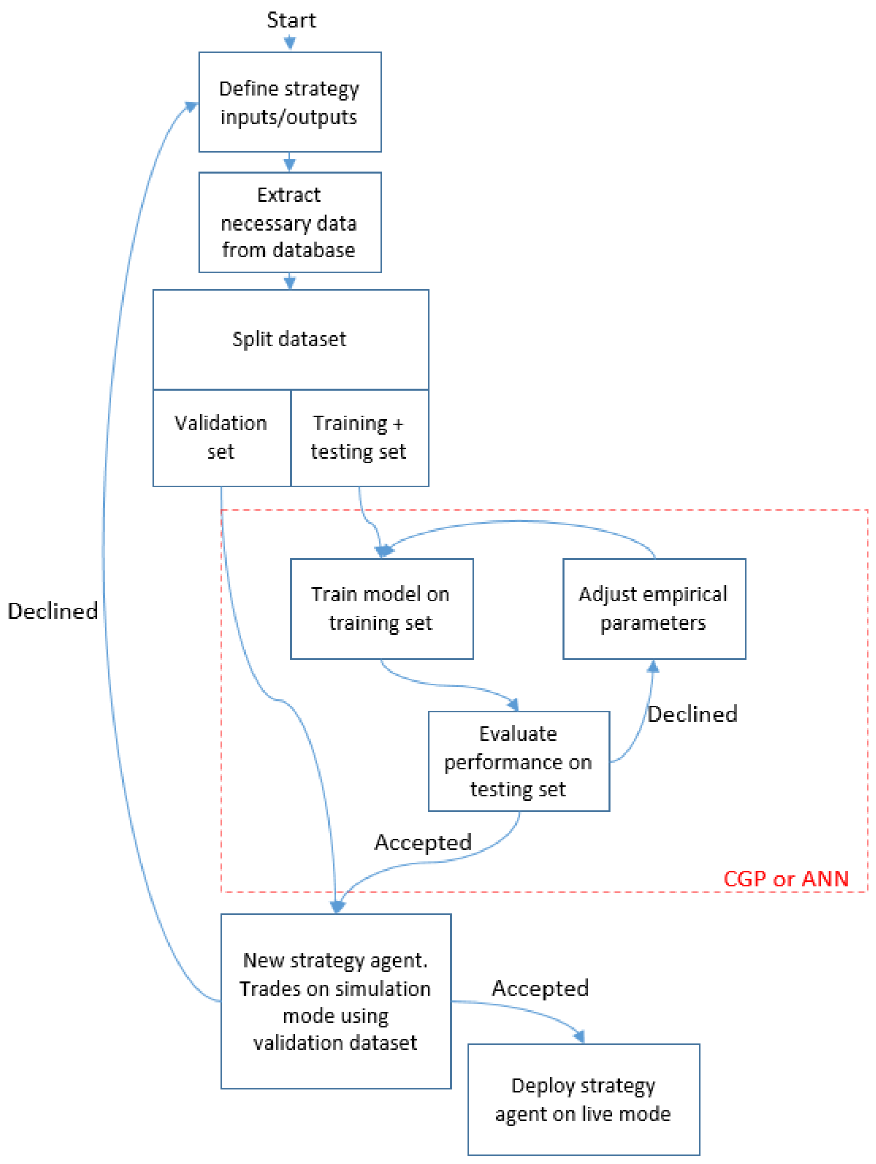
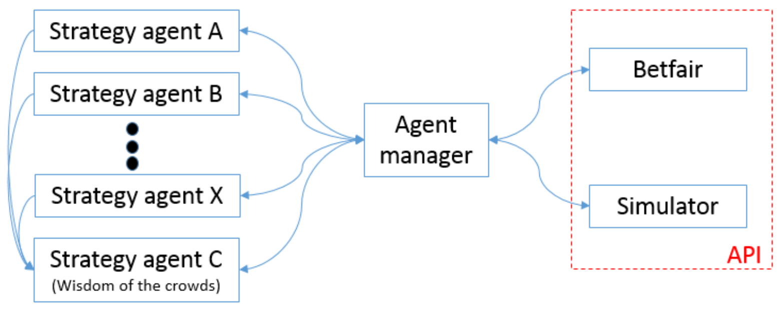

Start
Define strategy inputs/outputs
Extract necessary data from database
Split dataset
Validation set
Training + testing set
Train model on training set
Adjust empirical parameters
Evaluate performance on testing set
Declined
Accepted
CGP or ANN
Declined
New strategy agent. Trades on simulation mode using validation dataset
Accepted
Deploy strategy agent on live mode

Fig 3. Flowchart of machine learning strategy development

3.1.3. Strategy agents and Agent manager

Strategy agent – an automated algorithm that trades on the betting exchange, obeying predefined rules. Each machine learning strategy that passes verification discussed in Section 3.1.2, gets implemented as a Strategy agent. Statistical or empirical (plain) strategies can be implemented directly as a Strategy agent without the lengthy machine learning process.

Agent manager, as name suggests, manages all strategy agents and place bets accordingly. For every price update from Betting exchange/Simulator (referred as API in this section), *Agent manager* provides the new data to each *Strategy agent* and receives betting instructions in return. If received instructions specify that a bet must be placed, *Agent manager* places that bet in the buffer. Once all *Strategy agents* are notified and bet instructions received, *Agent manager* places all bets with API. API then responds with either bet success or failure. Manager notifies each *Strategy agent* with the corresponding response, if bet was not accepted, *Strategy agent* can try placing the bet again on the next price update. Furthermore, if the bet was only partially accepted, *Strategy agent* can try and place the remaining stake at a later stage. Placing orders (bets) via *Agent manger* and buffering them, allows consolidation of orders on the same selection and therefore minimizes the overall number of API requests. API requests take considerable time (depending on the network speed and physical location), in the range of 10-200ms per JSON-RPC request. Overall bet placement flow is shown in Fig 4.



1. Agent manager requests prices from API
2. API responds with live prices
3. Agent manager supplies the prices for each Strategy agent
4. Strategy manager responds with order (bet) instructions
5. Agent manager creates a bet order with API
6. API accepts/declines the order
7. Agent manager notifies Strategy agents of API's response
8. Repeat step 1-7 until market has ended.

Fig 4. Bet placement flow using Agent manager where Simulator is the Betfair API betting exchange simulator. Arrows represent the communication/information sharing paths.

"*Wisdom of the crowds refers* to the ability of statistical aggregates based on multiple opinions to outperform individuals, including experts, in various prediction and estimation tasks" [29]. Wisdom of the crowds can also be applied for betting exchange domain, where each *Strategy agent* votes for a specific outcome, such as price movement and a higher-level *Strategy agent* decides the final outcome (represented in Fig 4 as *Strategy Agent C*). The final decision can be based on average prediction, how contradictory the individual Agent's predictions are or combination of both.

3.1.4. Simulator

In order to correctly calculate a strategy's performance, it needs to be emulated correctly on real-life data. This is done using *Simulator*, see Fig 5. *SimInterface* class uses the *IClient* interface to replicate Betfair API calls, therefore, shifting from Simulator (development) to Betfair API (live) is done simply by changing the interface object.

At the start, *Simulator* requests a list of all market catalogues of interest, for example, all WIN horse racing markets for given day. *SimInterface* API then gathers the necessary information from MySQL database and responds to the *Simulator* with corresponding market/event IDs. Furthermore, API loads all requested market technical data in RAM. *Simulator* then creates a thread for each market. Moreover, each market has its own agent manager, which requests prices from the API. When an order is placed, API saves corresponding bet, with its associated bet ID, onto a file for further analysis. Once an event has finished, *Simulator* destroys corresponding market objects.

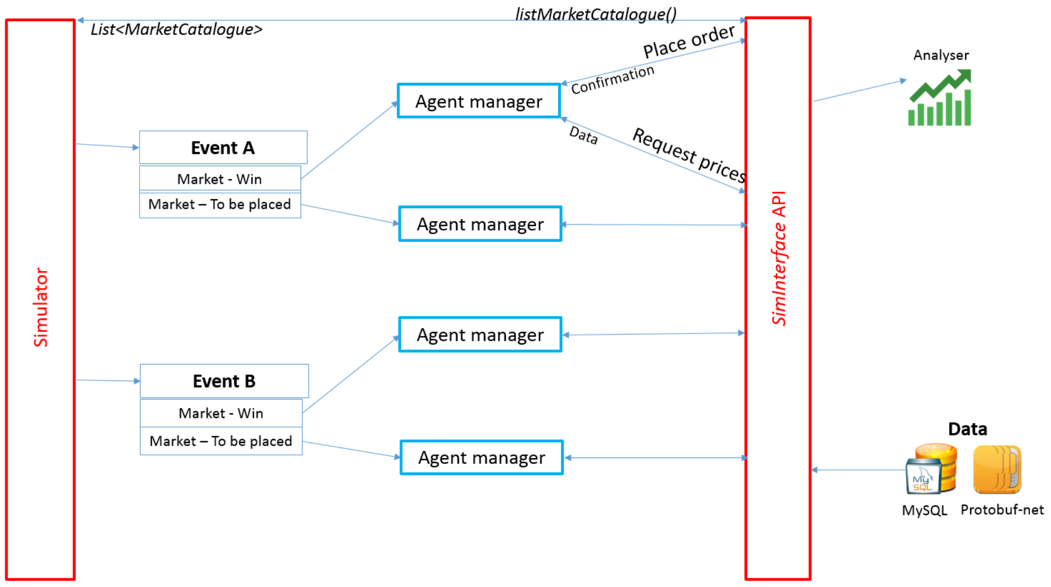


Fig 5. Overview of Simulator implementation

Requesting prices

It is essential that every single Agent manager's requested price information is a correct representation of the price at that time of the market. This is achieved by using system wide time. Time is increased in 10ms intervals and can be varied in speed by a speed factor. For instance, a speed factor of 1 would allow simulation of strategies in real time. However, that would take long time and be impractical for a month-long simulation. A speed factor of 100, would mean that every 100 seconds of historical data is processed in 1 second on simulation.

Every request to the API is time-stamped based on the system-wide time and can be used to match the corresponding price entry in the historical data. If such entry does not exist, the closest previous entry is returned. This approach allows to correctly emulate sequential time series on multiple markets and multiple events. Therefore, for example, in a football tournament, one team winning a match just before another match has started, could impact the price of correlated event dramatically.

Although, sequential market processing would allow the most accurate results, it takes a long time. A month worth of simulation at speed factor of 100, would take 7.2 hours, excluding any input/output overheads. Hence a parallel approach, where each market is processed individually, has also been implemented.

Placing orders

When a strategy is being back-tested, there can be instances where the bet stake exceeds the available volume on the specified price. For example, placing a bet of £100 when only £10 is available, would end up as £10 matched and £90 unmatched. However, when the next set of data comes in, there is a high chance that the £10 that was previously matched in *Simulator* would be still available. This is not correct, as in real market that volume would have been already matched. Therefore, there is a need to keep track of the placed bets and the matched volume for the specific price. For above example, if £90 is left unmatched, *Simulator* should be able to distinguish any new volume on that price and keep matching the unmatched bet, or keep it unmatched, if price moves in other direction. Back-testing emulator based on [25] has been implemented.

3.1.5. Analyzer

When simulation is finished, it is necessary to evaluate the performance of a given strategy [25]. This can be done via various back testing metrics, such as plain profit, profit after commission, number of winning/losing bets, number of consecutive wins/losses, Max Drawdown (MDD), Max Run-up (MRU), Average win/loss, Return on Investment (ROI), Risk Adjusted Rate of Return (RAR), Pessimistic return on margin (PROM), Perfect Profit (PP), Strategy Efficiency (SE) and T-test, all documented in [25].

T-test is statistical test that examines how likely results have occurred by chance alone. A T-test below 1.6 suggest that results are more likely occurred by pure chance, above 1.6 - has the potential for a long term sustained results. The higher the score, the more likely strategy will be able to perform long term [30].

These metrics are evaluated for each strategy agent and an Excel spreadsheet created automatically to display all statistics in a user-friendly format.

3.2. Machine learning models

Price movements in horse racing markets are volatile and on average, the most volume (money) matched is in the last 10 minutes before the race start time. From recorded data analysis it was concluded, that the largest drifts of pre-race are observed starting at around 6 minutes before the race start time. Due to more trading opportunities, strategies explored in this paper concentrates on predicting the Starting Price (price at time 0) 6 minutes before the race start time. The machine learning inputs are described in 3.2.2 while the output handling and bet execution are described in 3.2.1.

3.2.1. Objectives

All objectives in this section, except for Profit are evaluated based on Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - e_i)^2 \qquad \textbf{(1)}$$

Where y is the forecast output and e is the expected output for a given dataset entry i and n is the is the total number of dataset entries.

Regression

This objective tries to predict the exact exit price at time 0. For example, if entry price (6 minutes before the race start time) is 2.04 – 2.06, where 2.04 is the available Back price and 2.06 is the available Lay price, and exit price (at time 0) is 2.22 and 2.24, the expected output would be 2.22. Thus, a profitable trade would be a Lay bet at 2.06 and a Back bet at 2.22. The implied probability odds format is used instead of decimal format due to the normalized range of 0 to 1. Bet execution criteria – forecasted price is higher or lower than actual price.

Ticks

This objective tries to predict the number of tick changes and direction. The expected output is normalized by dividing the number of ticks by 100. Furthermore, the expected output is either positive or negative with an offset of 0.5. For example, an entry price of 2.04 – 2.06 and exit price of 2.22 and 2.24 have a tick gap of 8 (between 2.06 and 2.22) and as a profitable trade would be a Lay bet, 8 is assumed to be negative. Therefore, the expected output would be 0.5+(-8/100) = 0.42. If the entry bet is a Back bet, the tick gap is assumed to be positive. Bet execution criteria – a tick threshold of 3 ticks is applied for both Back and Lay bets, so a forecast output of 1 tick would not lead to a bet execution.

Classification

This objective tries to classify the betting options into three classes – Back bet, No bet and Lay bet, where Back bet is an expected output of 1, No bet is 0.5 and Lay bet is 0. For example, for the entry price of 2.04 – 2.06, and exit price of 2.06 and 2.08, the expected output would be 0.5, as no profit could be made. Furthermore, for an entry price of 2.04 – 2.06 and exit price of 2.00 - 2.02, the expected output would be 1 – a back bet. Bet execution criteria - a threshold of 0.1 is applied for both Back and Lay bets, so an output of 0.88 means no bet, since it is below the threshold (1 - 0.1 = 0.9).

Profit

Instead of focusing on forecasting error, this objective calculates the possible trade profit based on Equation 2:

$$ClosingStake = \frac{OpeningPrice}{ClosingPrice} * OpeningStake \quad \textbf{(2)}$$
$$Overall\ profit = \begin{cases} ClosingStake - OpeningStake & If\ first\ bet\ was\ Back \\ OpeningStake - ClosingStake & If\ first\ bet\ was\ Lay \end{cases}$$

For instance, using classification example above, if forecast output is 0.05 (a lay bet) and the entry price is 2.06 – 2.08 and exit price (price at time 0) is 2.22 – 2.24, the cost function would be evaluated based on profit made on a Lay-Back trade. Therefore, the overall profit with a stake of 10 units would be 10-((2.08/2.22) *10) = 0.63, after 5% commission, the profit is 0.6 units. As both CGP and ANN try to minimize the error, the trade profit is multiplied by -1. Consequently, a trade leading to a negative profit would have a larger error compared to a profitable trade. If the model does not meet the bet execution criteria, a profit of 0 is returned.

3.2.2. Machine learning inputs

This section derives 6 different trading models by altering how the raw data is processed before used as inputs for ANN and CGP.

6min_AP - Average prices

This model uses average minute prices from 30 to 6 minutes before the race to predict a Back or Lay bet and exit the trade at time 0. The average price for a one minute period is calculated by adding all Back-Lay price pairs together and dividing by total records during that interval – n (Equation 3).

$$P_{avg} = \frac{1}{n}\sum^{n}\left(\frac{P_{back}+P_{lay}}{2}\right) \quad \textbf{(3)}$$

This approach allows consolidation of input prices and also standardizes the input vector for the machine learning algorithm, thus eliminating any inconsistencies in record sample rates. The input vector is then normalized by using the inverse of the values.

6min_APC - Average price changes

Instead of using raw recorded prices, this model pre-processes the input prices further by calculating the price change between intervals *t* (Equation 4). The model uses price intervals from 30 to 6 minutes to predict a profitable trade at 6 minutes before the race with an exit bet at time 0. The implied probability price changes are then used as machine learning inputs.

$$P_{\Delta} = P_{avg(t)} - P_{avg(t-1)} \tag{4}$$

6min_AP_WOM - Average prices with custom Weight of Money

This model explores the relationship between average Weight of Money (WOM) and average interval price. For given back or lay price P at tick offset i the available volume V is weighted against the distance for given price (Equation 5 and Equation 6 for back and lay volume respectively). This would penalize available volume that is far from the current best price, while reward available volume closer to the best price. Furthermore, the overall Weight of Money is then calculated as a ratio between weighted volume available to back against both back and lay WOM (Equation 7).

$$W_{back} = \sum_{i=1}^{5} \frac{V_{P_{back}(i)}}{i} \tag{5}$$

$$W_{lay} = \sum_{i=1}^{5} \frac{V_{P_{lay}(i)}}{i} \tag{6}$$

$$WOM = \frac{W_{back}}{W_{back}+W_{lay}} \tag{7}$$

The WOM is calculated for every entry within the minute interval and averaging value used as the input vector for machine learning, similarly to the average interval price.

6min_AP_2OUT - Average prices with stop profit.

This model uses the same inputs as in 6min_AP, with the additional output of the exit price. 6min_AP only tries to predict the entry bet direction and exits the trade at time 0, whereas this model tries to forecast the entry bet direction 6 minutes before the race and also proposes an exit price. If during the 6-minute period price reaches the proposed exit price, an exit trade is made. If the proposed exit price is not reached, exit trade is made at time 0 the same way as for 6min_AP. If proposed exit price is lower than entry price, the proposed exit price is ignored and exit trade is made at time 0.

6min_AP_3OUT – Average prices with stop profit and dynamic staking

This model is an extension of 6min_AP_2OUT, where it tries to propose an additional output – stake size. If the stake size exceeds the maximum liability, it is capped at maximum. Furthermore, if the proposed value is negative, an absolute value is used instead.

top5_AP_WOM – Top 5 favorite relationships

This model tries to explore the relationship between top 5 selections (horses) within a given market. Only markets with at least 5 selections are used in this model. The top 5 selections are determined 30 minutes before the race scheduled start time. Inputs for all 5 selections are pre-processed the same way as in 6min_AP_WOM and all inputs combined. Furthermore, this model expects to have 5 entry bet predictions at 6 minutes before the race and 5 exit bets at time 0.

4 Results

This section describes results for multiple strategy models. All models were trained on training set for 24 hours using Intel Core i7 3930K @ 4.2GHz CPU (100% of CPU utilisation with parallel processing), then evaluated based on overall profit during training set (Table 4) and for unseen data - testing set (Table 5). Furthermore, all models used a limited liability staking plan of 100 units. Which means that with any single bet maximum potential loss is 100 units. Default Back/Lay bet stake is 10 units (used for all models except 6min_AP_3OUT). However, if the lay bet exceeds the maximum liability, it is capped at the maximum. For instance, if the proposed bet from the model is a lay bet at odds of 21, the stake is 100/(21-1) = 5 units.

Table 3. Overall profit (after 5% commission) for Training set for 36 models (1st of January 2016 to 10th of March 2016, GB WIN horse markets). One result represents one run.

Training set profit (5% commission included), in units

Objective / Approach

		Regression		Ticks		Classification		Profit	
		CGP	ANN	CGP	ANN	CGP	ANN	CGP	ANN
Inputs	6min_AP	1.7	187.56	147.74	231.15	23.47	102.74	589.61	634.32
	6min_APC	10.56	174	185.91	142	175.98	127.55	311.87	444.47
	6min_AP_WOM	12.5	202.53	95.61	144.92	105.3	173.08	325.55	660.17
	6min_AP_2OUT	x	x	x	x	x	x	470.74	441.02
	6min_AP_3OUT	x	x	x	x	x	x	3874.68	6234.09
	top5_AP_WOM	-120.77	673.25	13.08	770.5	15.64	775.92	-286.16	982.01

Table 4. Overall profit (after 5% commission) for Test set for 36 models (11th of March 2016 to 14th of April 2016, GB WIN horse markets). One result represents one run.

Test set profit (5% commission included), in units

Objective / Approach

		Regression		Ticks		Classification		Profit	
		CGP	ANN	CGP	ANN	CGP	ANN	CGP	ANN
Inputs	6min_AP	0.15	-52.3	-65.1	-83.6	4.41	-25.36	-213.32	-287.63
	6min_APC	6.2	-15.39	-88.07	-27.38	9.65	-24.7	2.69	33.14
	6min_AP_WOM	-4.69	-63.82	-15.69	-20.44	-5.32	-17.45	20.01	157.74
	6min_AP_2OUT	x	x	x	x	x	x	-110.25	-135.88
	6min_AP_3OUT	x	x	x	x	x	x	-1481.15	-3003.59
	top5_AP_WOM	-58.8	-482.91	-33.7	-563.99	-0.4	-753.73	-249.51	-657.61

Through result analysis, it was concluded that CGP prefers to "play safe" and not enter into market as much as ANN and hence, the strategy efficiency for ANN on average is higher. Using Profit as objective produce higher profit on average on the training set.

As shown in Table 3, almost all models are capable of forecasting profitable trades on the training set. However, only few models are able to generalise the underlying relationship such that it can be also applied to unseen data – test set. Out of 36 models 8 were able to make profit on the test set, from which only two (6min_APC – Profit – ANN and 6min_AP_WOM – Profit – ANN) have potential to be financially beneficial long term. Both of these models are explored in more detailed – models implemented as Strategy agents and run on the betting exchange Simulator on validation set (15th of April 2016 to 16th of May 2016) and result metrics (discussed in Section 3.1.5) recorded.

4.1. 6min_APC-Profit with ANN

This Strategy Agent uses Profit as an objective and is trained with ANN as described in Section 3.2.2. This model was run on Simulator and the overall trades analysed. During the validation set (15th of April 2016 to 16th of May 2016), 1295 bets were placed and overall profit (after 5% commission) was £41.35 (see Table 5). Although overall profit is positive, one should note that the total number of losing bets are by margin larger than the number of winning bets, but since average win is higher than average loss, it is still possible to make a profit. Furthermore, Risk-adjusted rate of return (RAR) of 33.11% shows a promising return. However, a T-test of 1.25 would suggest that the results have occurred by chance alone and therefore should not be proceeded with.

Table 5. 6min_APC-Profit with ANN Analyser output

Metrics	
Number of trades	1295
Profit after commission	**41.35**
Number of losing trades	706
Number of winning trades	589
Maximum sequential losses	12
Maximum sequential winnings	10
Average loss	-1.17
Average win	1.48
Max drawdown	-12.45
Max run up	19.98
Return on Investment (ROI)	41.35%
Risk-adjusted rate of return (RAR)	33.11%
The pessimistic return on margin (PROM)	36.69%
T-test	1.25136
Strategy efficiency (SE)	1.59%

4.2. 6min_AP_WOM-Profit with ANN

6min_AP_WOM-Profit with ANN on Test dataset (1 month) produced a profit of 157.74 units. Hence, this model was also implemented as a Strategy Agent and evaluated on validation set (15th of April 2016 to 16th of May 2016). Total of 1015 trades were executed, producing a profit of £83.03 (see Table 6). Risk-adjusted rate of return (RAR) of 54.63% shows very promising results for this Strategy Agent. Furthermore, average win and average loss are the same, however, because the total number of winning bets exceeds the total number of losing bets, the overall profit is positive. As there are only marginally more winning trades than losing trades, it is worth noting that it would only take a shift of 4.2% of trades in order to make a profitable Strategy Agent a losing one. Moreover, a T-test of 1.6 would suggest that it is more likely to be a profitable Strategy Agent long term compared to 6min_APC-Profit with ANN.

Table 6. 6min_AP_WOM-Profit with ANN Analyser output

Metrics	
Number of trades	1015
Profit after commission	**83.03**
Number of losing trades	486
Number of winning trades	529
Maximum sequential losses	9
Maximum sequential winnings	8
Average loss	-1.96
Average win	1.96
Max drawdown	-26.00
Max run up	12.89
Return on Investment (ROI)	83.03%
Risk-adjusted rate of return (RAR)	54.63%
The pessimistic return on margin (PROM)	81.22%
T-test	1.60232
Strategy efficiency (SE)	2.95%

5 Conclusion

This research has addressed multiple challenges designing and implementing a back-testing and machine learning framework for betting exchange markets. Furthermore, developed entirely new trading strategies using machine learning algorithms.

The proposed framework is generic enough that it can be adopted to various financial markets, however, the only difference is that betting markets are time constrained, i.e., market is finalized as the winner is determined, while most other financial markets are continuous in time. However, all the machine learning approaches explored can also be adapted to any other betting market and is not limited to Horse Racing.

From these findings, it can be concluded that using trading strategies, generated using machine learning algorithms, have the potential for high-risk/high potential return investment. This form of investment would classify as Speculative and therefore could lead to the loss of large amount of funds if money management and risk control where incorrectly applied. Furthermore, additional risk is added by the ever-changing betting exchange fees and regulations. As Betfair is market leader by a large margin, it has the power to apply additional charges to their users. For instance, additional fees such as Premium Charge[9] can be introduced without a notice and therefore decrease the overall Return on Investment. Moreover, additional barriers to entry – a fee for live API key that was introduced in May 2016 - can be enforced on users. It is uncertain what other charges might be applied in the future and hence increased risk on this sort of investment. Additionally, profitable strategy can lose its “edge” and become un-profitable once markets adjust to the new change, according to the efficient market hypothesis [30].

Further work includes live-testing and dynamic re-training of the best performing models. However, this would require much faster model training times, therefore, further investigation in hardware accelerators such as General Processing Units (GPUs) would be useful.

[9] http://www.betfair.com/aboutUs/Betfair.Charges/#

Works Cited

[1] R. Munting, An Economic and Social History of Gambling in Britain and the USA., Manchester University Press, 1996.

[2] G. Commision. [Online]. Available: http://www.gamblingcommission.gov.uk/PDF/survey-data/Gambling-industry-statistics.pdf. [Accessed 11 11 2017].

[3] N. O'Connor, "A Short History of the Betting Exchange Industry.," [Online]. Available: http://www.bettingmarket.com/refraichir010388.htm.

[4] H. Chen, P. B. Rinde, L. She, S. Sutjahjo, C. Sommer, D. Neely, "Expert Prediction, Symbolic Learning, and Neural Networks. An Experiment on Greyhound Racing," *IEEE Expert,* vol. 9, no. 6, pp. 21-27, 1994.

[5] R. P. Schumaker, O. K. Solieman, H. Chen, "Greyhound Racing Using Support Vector Machines: A Case Study," in *Sports Data Mining*, Springer US, 2010, pp. 101-108.

[6] S. Easton, K. Uylangco, "Forecasting outcomes in tennis matches using within-match betting markets," *International Journal of Forecasting,* vol. 26, pp. 564-575, 2010.

[7] M. Sipko, "Machine Learning for the Prediction of Professional Tennis Matches," Imperial College London, 2015.

[8] B. Ulmer, M. Fernandez, "Predicting Soccer Match Results in the English Premier League," Stanford University, 2014.

[9] A. S. Timmaraju, A. Palnitkar, V. Khanna, "Game ON! Predicting English Premier League Match Outcomes," Stanford University, 2013.

[10] D. Buursma, "Predicting sports events from past results. Towards effective betting on football matches," University of Twente, 2010.

[11] S. Kampakis, W. Thomas, "Using Machine Learning to Predict the Outcome of English County twenty over Cricket Matches," University College London, 2015.

[12] C. Kempston, "How to Win at the Track," Stanford University, 2007.

[13] S. Pudaruth, N. Medard, Z. B. Dookhun, "Horse Racing Prediction at the Champ De Mars using a Weighted Probabilistic Approach," *International Journal of Computer Applications,* vol. 72, no. 5, pp. 39-42, 2013.

[14] J. Williams, Y. Li, "A case study using neural networks algorithms: horse racing predictions in Jamaica," in *International Conference on Artificial Intelligence*, Las Vegas, NV. United States, 2008.

[15] F. G. Haahr, "Market efficiency: An analysis of the Internet betting exchange market," Copenhagen Business School, 2011.

[16] D. Pissarenko, "Neural Networks For Financial Time Series Prediction:Overview Over Recent Research," 2001-2002.

[17] I. Kaastra, M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing,* vol. 10, pp. 215-236, 1996.

[18] A.J. Turner, J.F. Miller, "Recurrent Cartesian Genetic Programming of Artificial Neural Networks," in *Genetic Programming and Evolvable Machines*, 2016, pp. 1-28.

[19] A. M. Madurska, "A set-by-set analysis method for predicting the outcome of professional singles tennis matches," Imperial College London, 2012.

[20] F. J.G.M. Klaassen, J. R. Magnus, "orecasting the winner of a tennis match," *European Journal of Operational Research,* vol. 148, pp. 257-267, 2003.

[21] Ø. Øvregård, "Trading "in-play" betting Exchange Markets with Artificial Neural Networks," 2008.

[22] J. P. A. Santos, "A Trading Agent Framework Using Plain Strategies & Machine Learning," UNIVERSIDADE DO PORTO, 2014.

[23] P. A. Bebbington, "Studies in informational price formation, prediction markets, and trading," University College London, London, 2017.

[24] A. Bunyan, "Time Series Analysis and Forecasting of In-Play Odds on a Betting Exchange," School of Computing, National College of Ireland, Dublin, 2014.

[25] P. Tsirimpas, "Specification and Performance Optimisation of Real-time Trading Strategies for Betting Exchange Platforms," Imperial College London Department of Computing, London, 2015.

[26] I. Kaastraa, M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing,* vol. 10, no. 3, p. 215.236, 1996.

[27] I.Dzalbs, T. Kalganova, "Multi-step Ahead Forecasting Using Cartesian Genetic Programming," in *Inspired by Nature*, Springer International Publishing AG, 2018.

[28] J. F. Miller, Cartesian Genetic Programming, Springer Berlin Heidelberg, 2011.

[29] D. V. Budescu, E. Chen, "Identifying expertise and using it to extract the Wisdom of the," *Managment Science,* vol. 61, no. 2, pp. 267-280, 2014.

[30] J. Butler, Betfair Trading Techniques, BPT Publications, 2016.