Expert Systems With Applications 54 (2016) 193–207

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Evaluating machine learning classification for financial trading: An empirical approach

Eduardo A. Gerlein[a,c,*], Martin McGinnity[a,b], Ammar Belatreche[a], Sonya Coleman[a]

[a] *Intelligent Systems Research Centre, University of Ulster, Londonderry, UK*
[b] *School of Science and Technology, Nottingham Trent University, Nottingham, UK*
[c] *Electronics Department, Pontificia Universidad Javeriana, Bogotá, Colombia*

ARTICLE INFO

Keywords:
Trading
Financial forecasting
Computer intelligence
Data mining
Machine learning
FOREX markets

ABSTRACT

Technical and quantitative analysis in financial trading use mathematical and statistical tools to help investors decide on the optimum moment to initiate and close orders. While these traditional approaches have served their purpose to some extent, new techniques arising from the field of computational intelligence such as machine learning and data mining have emerged to analyse financial information. While the main financial engineering research has focused on complex computational models such as Neural Networks and Support Vector Machines, there are also simpler models that have demonstrated their usefulness in applications other than financial trading, and are worth considering to determine their advantages and inherent limitations when used as trading analysis tools. This paper analyses the role of simple machine learning models to achieve profitable trading through a series of trading simulations in the FOREX market. It assesses the performance of the models and how particular setups of the models produce systematic and consistent predictions for profitable trading. Due to the inherent complexities of financial time series the role of attribute selection, periodic retraining and training set size are discussed in order to obtain a combination of those parameters not only capable of generating positive cumulative returns for each one of the machine learning models but also to demonstrate how simple algorithms traditionally precluded from financial forecasting for trading applications presents similar performances as their more complex counterparts. The paper discusses how a combination of attributes in addition to technical indicators that has been used as inputs of the machine learning-based predictors such as price related features, seasonality features and lagged values used in classical time series analysis are used to enhance the classification capabilities that impacts directly into the final profitability.

1. Introduction

Data mining is the process of finding hidden patterns within data using automatic or semi-automatic methods (Witten, Frank, & Hall, 2011). In particular, machine learning (ML) techniques have shown impressive performance in solving real life classification problems in many different areas such as communications (Di, 2007), internet traffic analysis (Nguyen & Armitage, 2008), medical imaging (Wernick, Yang, Brankov, Yourganov, & Strother, 2010), astronomy (Freed & Lee, 2013), document analysis (Khan, Baharudin, Khan, & E-Malik, 2009), biology (Zamani & Kremer, 2011) and time series analysis (Qi & Zhang, 2008). Although complex models such as Neural Networks (NN) and Support Vector Machine (SVM) techniques are studied within the ML field, several other approaches also exist, characterized by a greater degree of simplicity when compared with NN and SVM. Despite this apparent simplicity, some of the ML techniques may be well-suited for quantitative analysis within the financial industry, as their capabilities for finding hidden patterns in large amounts of data may help in financial forecasting for trading.

Financial trading of securities using technical and quantitative analysis has been traditionally modelled by statistical techniques for time series analysis such as the ARMA (Box et al., 1994) and ARIMA models, and the more sophisticated ARCH technique (Engle, 1982). In contrast to these statistical approaches, complex models coming from the ML field have emerged attempting to predict future movements of securities' prices (Yoo, Kim, & Jan, 2007). The extensive literature has shown how some ML techniques specializing in classification and regression tasks have demonstrated

[*] Corresponding author at: Electronics Department, Pontificia Universidad Javeriana, Bogotá, Colombia. Tel. +5713208320 ext 5549.

E-mail addresses: gerlein-e@email.ulster.ac.uk (E.A. Gerlein), tm.mcginnity@ulster.ac.uk (M. McGinnity), a.belatreche@ulster.ac.uk (A. Belatreche), sa.coleman@ulster.ac.uk (S. Coleman).

http://dx.doi.org/10.1016/j.eswa.2016.01.018

that they are well-suited for quantitative analysis within the financial industry, as their capabilities of finding hidden patterns in large amounts of financial data may help in derivatives pricing, risk management and financial forecasting. One of the most published projects that uses such techniques in financial applications is Standard & Poor's Neural Fair Value 25 portfolio (Smicklas, 2008), which selects on a weekly basis 25 stocks using an artificial NN, from a total of 3000 stocks yield relative to that of the S&P 500 index, attempting to outperform the market by calculating a stock's weekly fair value based on fundamental analysis. Particularly for securities trading, the utility of complex models such as NN, SVM and hybrid models (Cai, Hu, & Lin, 2012) have been extensively studied and have led to promising results. Nevertheless, information regarding the incorporation of such methods into trading floor operations tends to remain hidden to the public, for commercial proprietary reasons (Yamazaki & Ozasa, n.d.; Duhigg, 2006; Patterson, 2016).

In terms of financial trading, analysts in the industry (usually referred to as "quants") have developed *technical indicators* that are used to identify the most suitable moments to open and close trades, and are possibly the most popular tools currently used in technical trading. Published research that aims to incorporate Computational Intelligence (CI) in financial prediction shows how those *technical indicators* have been used as inputs to ML models to find the hidden patterns or relationships among them, in order to predict future prices, trends or a percentage of confidence in those predictions. With the possible exception of long term averages, those *technical indicators* are constructed using information of prices over short periods in the past, no more than 20–30 trading periods in order to incorporate the historical behaviour in a single value. The selected trading period is part of the trading strategy and might vary from long frames of 1 day to small frames of 1 minute, or even smaller time windows as in the case of high frequency trading. The construction of such indicators can be seen as a process used in large scale time series data sets called *dimension reduction* (Wang et al., 2005) that attempts to transform the series to another domain seeking a version of the data set that might be much simpler to analyse. In contrast to time series analysis where the data set is seen as a whole entity, ML classification tasks construct independent *instances* that are representative examples of the concept to be learned.

Financial predictions that incorporate ML approaches construct the training, test and off-sample data sets as a collection of *instances* using popular *technical indicators* as reported in a number of papers. Hence, an *instance* is created usually using the value of the current price and the instantaneous values of the mentioned indicators, generating a static picture of the situation of the market for the exact time that the *instance* is constructed. In this scenario, each *instance*, i.e. prices and their correspondent *technical indicators* used as attributes, becomes itself an independent example of the problem, which avoids the time dependence in the series, approaching the problem as simple classification task rather than a time series analysis in the strict meaning of the word. The hypothesis in this case is that once a ML model is trained, it may be able to classify individual *instances* using the technical indicators as attributes, due to the fact that those unseen instances represent in turn the invariant circumstances of the market at certain points in time, and that the result of the classification task can be interpreted as a trend forecast. The main implication of this hypothesis is that the financial forecasting can benefit from the use of simpler ML techniques rather than using complex time series analysis approaches, simplifying the use of computational resources while at the same time avoiding indexing and ordering issues in the data sets.

This paper addresses the question of the usefulness of low-complex ML classifiers in financial trading, and in particular will demonstrate if such low-complexity binary classification approaches are able to generate consistent profitable trading over an extensive period of time. The paper's main contribution resides in the fact that simple machine learning models that traditionally have been precluded from financial applications, as opposed to the more complex NN and SVM, can be used to generate profitable transactions on the long term with the correct combination of periodic retraining, training set size and attribute selection. The work is motivated mainly by the results reported in (Barbosa, 2011), which claims outstanding financial results using simple classifiers. In this case, a simple model is characterized by the low computational requirements for both the training and the classifying process due to the inherent simplicity of the learned model (instance-based classifiers, decision trees and rule-based learners). While the main objective of ML classification is to maximise accuracy, this might not be the best metric to evaluate the performance of such systems when used in the context of financial trading. The most important metric when assessing trading strategies is undoubtedly profitability, reflected in this paper as cumulative return over a specific trading period. In this paper, it is developed an empirical comparison between average accuracy and cumulative return as the main metrics of performance of a set of six machine learning models (OneR, C4.5, JRip, Logistic Model Tree, KStar and Naïve Bayes). The models produce a binary classification used later to predict price movement (up or down in the next trading period) for the USDJPY currency pair using six hour time frames over a trading time-frame of six years. The six hour time frame was selected to be able to validate and compare with the results reported in (Barbosa & Belo, 2008b), although the same approach used in the experiments can be applied to different time frames. A set of experiments was conducted where the results of modifying three variables were studied: training set size, period of retraining and number of attributes for the training and test sets. The results show relative low accuracy, only a few points over 50%, but at the same time, very promising results in terms of profitability. Later, further experiments are conducted on simulated trades over the same period of time using EURGPB and EURUSD currency pairs, and similar results are reported.

The remainder of the paper is organised as follows: Section 2 discusses related work using ML in financial forecasting applications. Section 3 presents the general experimental setup, describing the data sets, and the attribute selection to feed the models and briefly describes the different ML algorithms used in the experiments as well as their particular parameter set up in order to present comprehensive information for future experiment replication. Section 4 discusses the results detailing each one of the simulated trading scenarios. Finally, Section 5 concludes the paper and explores future work.

2. Machine learning in financial forecasting

Within the financial trading chain two main areas can be identified, where the use of ML techniques have reported particularly successful implementations: derivatives pricing, risk management and financial forecasting. Financial forecasting is possibly the most important application within ML for data mining in Capital Markets. ML techniques for forecasting include expert or rule-based systems, decision trees, NNs and genetic computing. Applications within the trading cycle such as Algorithmic Trading Engines[1] and

[1] Algorithmic trading engines in the buy side, are essentially semi-automatic computer aided systems that help retail investors to take the best financial decisions in terms of high returns at lowest possible risks, and by means of programming specific rules the system are capable of transmitting pre- and post-trade data about quotes and trades to other market participants (Hendershott, 2003 Chan, 2008). The literature also reports the use of algorithmic trading engines in the sell

Order Matching Engines[2] (Hendershott, 2003) have the potential to incorporate different levels of CI, and in particular ML techniques. The majority of existing ML-based methods for trading use *technical indicators* as part of the training attributes extracted from the financial series instead of using the raw prices as a training vector. Maggini, Giles, and Horne (1997) had pointed out that there is an inherent difficulty in generating statistically reliable *technical indicators*, due to the fact that the rules inferred to produce accurate predictions are changing continually in financial time series, and that it is even possible to evidence the presence of a high number of contradictory instances in the training sets due to the fact that market data exhibit statistical characteristics found in other types of time series. This situation is reflected in the large volume of papers (Chen & Shih, 2006; Eng, Li, Wang, & Lee, 2008; Kim, 2003; Lee, Park, O, Lee, & Hong, 2007; Li & Kuo, 2008; Tenti, 1996) that have reported accuracies under 60% with ML models which have shown impressive performance in areas other than financial prediction. According to Sewell and Yan (2008), for certain markets such as futures and FOREX, it may be necessary to generate predictions with an accuracy marginally higher than the one obtained by a random classifier to obtain profits due to two main factors: low costs and leverage.

Artificial NNs are probably the most common method utilized in financial forecasting. Early works such as that of Tenti (Tenti, 1996) compared the performance of three recurrent neural networks based on their returns in the simulated forecasts on currency futures. The inputs to the networks include *technical indicators* (average directional movement index, trend movement index and the rate of change). Tenti also takes into account trading costs, and reports positive returns in the trading simulation, demonstrating that NN techniques can indeed be used as forecasting tools. Lu and Wu (2009) show another example of stock market forecasting with artificial neural networks. The paper compared a NN model's performance against the ARIMA model, predicting the direction of future values of the S&P 500 Index. The experiments showed that the NN-based system outperformed the ARIMA model only in stable market conditions, since the system only exhibits a modest 23% level of accuracy against the ARIMA's 42% in more volatile scenarios. Kamruzzaman and Sarker (2003) compared the performance of the ARIMA model with several NN models when forecasting exchange rates of currency pairs in the FOREX market. The NNs were trained with back-propagation, scaled conjugate gradient and back-propagation with Bayesian regularization, using exchange rates in the previous period and moving averages as inputs. The accuracy in the prediction as well as the normalized mean square error and the mean absolute error were used to measure global performance, showing that all NN models outperformed the ARIMA model with an accuracy of 80%. Taking into account that only the best results obtained were reported and that, in general ML techniques do not present high levels of accuracy with unseen data that differ significantly from the training sets, these impressive results must be considered with care. (McDonald, Coleman, McGinnity, Li, & Belatreche, 2014), investigate the effectiveness of a number of machine learning algorithms and their combinations at generating one-step ahead forecasts of a number of financial time series. The authors found that hybrid models, consisting of a linear statistical model and a non-linear machine learning algorithm, are effective at forecasting future values of the series, particularly in terms of the future direction of the series.

While artificial NNs are considered the most popular technique in financial forecasting, other reports also show promising results with different data mining models. Using SVM, Kim (2003) attempted to forecast daily price directions of the KOSPI stock index. The model used technical analysis indicators (momentum, Williams %R and commodity channel index) as inputs and the best accuracy obtained after training several models with different parameters was 57.83%. The work also presented a comparison with the back propagation NN (with 54.76% of accuracy) and nearest-neighbour model (51.98% of accuracy). This middle-range level of accuracy is expected due to the high volatility of financial time series, but to achieve it several models needed to be trained. This study concludes that no single model is perfectly suited in all market conditions, and even more importantly, the models must be retrained frequently to maintain the forecasts accurate. In another study, Tay and Cao (2001) also compared SVM with back propagation NN to forecast prices of five types of futures contracts. On average, the SVM approach obtained better accuracy than the back propagation NN, but also in middle-range levels: 47.7% by SVM against 45.0% obtained by back propagation NN. SVM also outperformed a back propagation NN in the work presented by Chen and Shih (2006), where these techniques were used to predict the value of six Asian indices, obtaining 57.2% level of accuracy with SVM and 56.7% with NN models.

Apart from NN and SVM, in the study presented in (Maggini et al., 1997) the authors proposed a heuristic method to select different inputs for a non-linear machine learning algorithm, discarding the option of time series prediction and limiting the problem to classification to determine the class of price variation, although there is no specification in the paper if the problem is restricted to a binary (up/down) or multi-class classification (up/down/stable) problem. The selected method is the *K-nearest neighbours* with a sliding window dataset used to retrain the model at every time step. The metric selected to evaluate the performance was mean square error. The paper concludes that it is impossible to predict price variation with enough accuracy, discouraging the use of this approach and attributing the poor results to the weakness of the model and a poor selection of inputs that might affect the price movement. Nevertheless, the authors seem to be focused on accuracy and do not provide any financial results in the trading period used in the simulation. J. Li, Tsang, and Park (1999) predicted expected return in the Dow Jones Industrial Average using Financial Genetic Programming, and compared it with random decisions and C4.5 decision tree classification. They used some simple technical indicators such as short and long term moving averages and long and short term price filters. The authors did not focus only on accuracy of predictions but also on annualized returns and positive returns of a simulated set of investments following the predictions. They reported over 60% in positive returns and over 40% in annualized returns over a trading period of four years for the Genetic Programming model and over 40% for the C4.5 decision tree. In both cases, the results represent an outstanding financial return even without taking into account trading costs, which suggest that technical indicators might generate profitable rule-based models to predict complex financial time series.

Barbosa and Belo presented several reports using single agents to execute algorithmic trading in the FOREX Market (Barbosa & Belo, 2008b), a micro-society managing a hedge fund (Barbosa & Belo, 2010) and a multi-agent system for multiple markets trading (Barbosa & Belo, 2008a), focusing on profitability and maximum drawdown as performance metrics. The proposed architecture is divided into three modules in charge of (a) predicting the immediate next trend by means of an ensemble of binary classifiers, (b) a risk management module to decide how much to invest

side, brokers and investment banks, to manage the vast amount of daily orders from the clients usually arrived before the market opening hours, deciding how and when to execute those orders taking into account existing regulations and at the same time, minimizing the effect on prices [53].

[2] An order matching engine is a trading system that facilitates the exchange of financial instruments between multiple parties by means of a transaction algorithm that translates orders into trades pairing buyers with sellers in terms of transaction prices and quantities (Hendershott, 2003).

in each trade by a case-based engine that analyses past trades, and (c) a rule-based system, where a set of rules resulting from human experience can be incorporated to enhance the trading decision and add limit and stop-loss orders, trading and closing policies. The system performs learning by means of an update to the weighted ensemble according to the results of the individual simple ML models such as OneR, C4.5, JRip, Logistic Model Tree, KStar, NN, SVM and Naïve Bayes, and also retrains the classifiers at fixed periods to adapt to new market regimes. What is interesting in this development is that even though the classifier module only produced a not very impressive 52.74% accuracy, the complete system produced a success rate of 66.67% in profitability over the tested period, performing fewer but more profitable trades and avoiding trades that were expected to be unprofitable due to the combination of the different modules, with a high level of automation.

In all these works, there is a general agreement in favour of the use of ML models for performing financial forecasting. Most articles report positive results and may be seen as empirical evidence against the efficient market hypothesis[3], to demonstrate that there is some predictability of market prices based on historical data (Li et al., 1999). With the possible exception of the work presented by Barbosa and Belo, in general all the reviewed reports focused on the use of sophisticated data models such as NN, SVM and genetic programming approaches, using *technical indicators* as inputs. These approaches are shown to be well-suited for financial data modelling and forecasting, but still there is a need to study in depth the capabilities and limitations of these techniques for one of the most competitive industries in the world. The lack of reports that explore data mining approaches with an inherent simplicity of the learned model such as instance-based classifiers, decision trees and rule learners to generate consistent profitable trading suggest the further exploration and discussion in this area, and is the main motivation for this paper. Additionally, it is important to establish a relationship between accuracy and profitably in ML-based trading, due to the fact that financial trading incurs considerable costs, which must be included in the assessment of any new technique.

3. Experimental setup

In order to validate the assumption suggested by Barbosa and Belo (2008a) where low complexity ML models can be used to trade in a consistent profitable fashion, a multiagent system is implemented to carry out a series of trading simulation experiments over a two year period. A six hour time frame gives the opportunity to open and close trades four times per day, expecting greater price movements during the interval, and consequently opening the possibility to obtain greater profit per trade on average compared to low range time frames such as 1 hour or less. At the same time, as suggested in (Barbosa & Belo, 2008a), the convenience of selecting six-hour time frames, starting at midnight, will guarantee that the trades do not coincide with the traditional times where major reports are released such as the *Nonfarm Payrolls Employment* or the *interest rates*, avoiding the high volatility associated with those events that impact directly in slippage[4].

The multiagent system is built using a JAVA-based multiagent framework called BESA (González, Avila, & Bustacara, 2003). The Organizational Approach for Agent Oriented Programming methodology proposed by González and Torres (2006) was followed. The machine learning models used in these experiments to predict price trends were integrated in the agents' forecasting modules using the WEKA toolbox (Witten et al., 2011). Given the "open-source" nature of the tool, WEKA-based classifiers are imported and instantiated as JAVA objects, providing a seamless integration with a custom application, great flexibility and simplified implementation, focusing the efforts on the feature extraction task.

Fig. 1 depicts the general architecture of the system, where two agents are shown: a *Market Agent* and a *Trading Agent*. The role of the *Market Agent* is to encode financial information using Japanese Candlestick charts (open, low, high, close prices at the pre-defined 6 hour time frames) and a 32-bit UNIX timestamp. Additionally, the *Market Agent* sends the instrument information to the *Trader Agent,* and maintains an order book to keep track of the orders as well as the profits of the client (*Trader Agent*).

The *Trader Agent* receives information about the state of the market as raw price data in Japanese Candlestick format, and outputs market orders to open and close trades at every trading period. An internal pre-processing module is in charge of calculating different technical indicators as part of the feature extraction task, which serve as an input feed to one of the six classifiers selected. The classifier module in turn is in charge of generating a price trend forecast for the next trading period: "*the price will increase in the next trading period*" or "*the price will decrease in the next trading period*". Finally, using the result of the classifiers, the *Trader Agent* generates opening and closing orders and sends them to the *Market Agent*. The ML models inside the Trading Agent are set up to produce a binary classification output predicting the direction of the price of the financial instrument of interest for the next trading period. The classification results are used to decide if the instrument should be bought (long trade) or sold (short trade). The first set of experiments included six different models trained with historical exchange data of the USDJPY currency pair. Details on the particular set up (attributes and training parameters) are discussed in the following sections.

3.1. Data set and attribute selection

From the raw prices data, nine attributes are constructed off-line to build the attribute vectors that comprise the initial training set. The particular attribute selection is based on Barbosa and Belo (2008a), which suggests that the integration of a diverse types of features such as seasonality features, lagged values and technical indicators such as moving averages, RSI and WR, are used in the construction of the models to enhance the classification capabilities in both the training and prediction processes. The characteristics of the data and the selected attributes are shown in Table 1:

The nine attributes are calculated off-line for the training set derived directly from the historical price information. The trading simulation is performed using a test set comprised of off-sample instances covering a two year period as described in Table 1. For

[3] The financial industry still debates the idea of accepting that there is no possibility to "beat the market" and in consequence, that instrument prices are not predictable. In other words, that is impossible to be able to obtain a consistently return higher than an index growth with a simple buy-and-hold strategy (Becket & Essen, 2010). The efficient market hypothesis claims at any given point in time, an instrument's price always fully reflects all the information available. The random walk hypothesis says that stock prices follow a random walk model, i.e. the variations in price from one time step to the next one are completely independent. The martingale hypothesis suggest that forecasting based on historical prices is ineffective (Becket & Essen, 2010). However several famous investors (Ellis, 2001), have been successful for decades making profitable financial forecasts, something that should not be possible if asset prices were completely random.

[4] Slippage is defined as the difference between the expected price for a trade and the effective price at which it is executed. In FOREX, slippage often occurs at high volatility periods where the prices exhibit unexpected movements, generally during news events releases, which makes very difficult to execute an order at a specific price. Slippage in the trading of stocks is related to the spread between the ask and bid prices, usually at execution of market orders at the time of a spread movement generally caused by the presence of large orders executed when there is not enough buyers/sellers to fill the desired price level to maintain the expected price of trade.

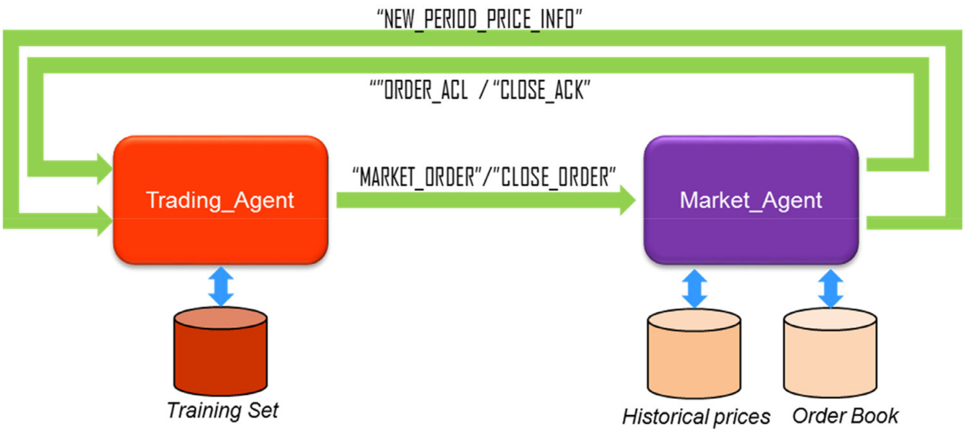

Fig. 1. Trading engine – A multi-agent system composed of two agents. A trading agent in charge of predicting price trends and generating market orders. A market agent is in charge of keeping track of an order book and sends market information using a historical data base.

Table 1
Description of the training and test sets – no-retrain experiments.

Currency:	*USDJPY*	
	Number of instances	*Period*
Training set	5191	Wednesday, 02 January 2002 00:00:00–Friday, 29 December 2006 18:00:00.
Test set (off-sample data):	2510	Thursday, 18 January 2007 12:00:00–Monday, 22 Jun 2009 00:00:00.
	Attributes	
Nine attributes	Hour, day of the week, closing price, percentage of price change, lagged percentage of price change, lagged percentage of price change moving average (10 periods), relative strength index, Williams %R, class.	

the test set (off-sample data), the *trader agent* is in charge of calculating the corresponding attributes every new trading period. The particular details of the selected attributes are:

- HOUR: time of the day in when the instance is captured, e.g., trading fixed 6 hour time frame. The possible values are: 0, 6, 12 and 18.
- DAY_OF_WEEK: nominal attribute represented as a list of nominal labels, i.e., {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}. The last two attributes are selected taking into account seasonality factors in the price change (Nawaz & Mirza, 2012; Maggini et al., 1997).
- CLOSING_PRICE: numeric attribute that represents the price of the instrument at the end of the trading period.
- PERCENTAGE_PRICE_CHANGE: this numeric attribute describes the relative change in the price during the current trading period given by Eq. (1).

$$ppc_{i+1} = ((closePrice_{i+1} - closePrice_i)/closePrice_i) \times 100 \quad (1)$$

- LAGGED_PERCENTAGE_PRICE_CHANGE: This numeric value is used in time series analysis such as ARIMA to integrate to values of the past values of the series. In this case, the lagged percentage price change will be described by the preceding period percentage price change as:

$$lagged\_ppc_i = ppc_{i-1} = ((closePrice_{i-1} - closePrice_{i-2})/closePrice_{i-2}) \times 100 \quad (2)$$

- LAGGED_PERCENTAGE_PRICE_CHANGE_MOVING_AVERAGE: this numeric attribute is calculated constructing the average of prices changes in the last n periods, given by:

$$lagged\_ppc_{MA(n)i} = \left(ppc_i + \sum_{k=1}^{n-1} lagged\_ppc_{i-k} \right)/n \quad (3)$$

- WILLIAMS_%R: this numeric attribute represents a technical indicator whose value oscillates between 0 and −100. When Williams %R oscillator is below −80 or over −20, it means that the instrument is oversold or overbought respectively. The Williams %R compares the closing price in the current period with the lowest and highest prices in the last n periods and it is calculated as:

$$williams\_R(n)_i = ((closePrice_{i-1} - high_n)/(high_n - low_n)) \times 100 \quad (4)$$

- RELATIVE_STRENGTH_INDEX: this attribute is also a technical indicator which compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an instrument. The relative strength index oscillates from 0 to 100 and indicates an overvalued asset when its value reaches 70 and therefore a price drop is expected, or otherwise if the index reaches 30 it indicates that the asset may be undervalued and a price rise will occur. The relative strength index is calculated:

$$relative\_strength\_index(n)_i = 100 - 100 \times (1/(1 - RS(n)_i) \quad (5)$$

- *RS(n)*$_i$ indicates a ratio between the average of past n periods where the price increased and the average of n periods where the price decreased, given by Eqs. (6)–(8):

$$RS(n)_i = UP\_avg(n)_i / DOWN\_avg(n)_i \quad (6)$$

$$UP\_avg(n)_i = \left(\sum_k (closePrice_k - openPrice_k) \right)/n \quad (7)$$

- where k is a period characterized by the close being higher than the previous close, and

$$DOWN\_avg(n)_i = \left(\sum_j (openPrice_j - closePrice_j) \right)/n \quad (8)$$

- where j is a period characterized by the open being higher than the previous opening price.
- CLASS: the class will be defined as a nominal feature that is attempted to be classified by the ML models. It shows the direction that the price will take in the very next trading period. The labels for the class are UP and DOWN, describing if the closing price is expected to rise or decrease compared to the opening price in the very next trading period respectively.

3.2. Performance metrics

Many different techniques have been proposed to evaluate the performance of classifiers, such as repeated cross-validation which is probably the one of the most popular choice especially in a situation with limited-data. In theory, a cross-validation procedure delivers a more confident metric on the future performance of a particular model over unseen new data while at the same time provides a consistent procedure to compare the classification capabilities for different models applied to the same problem. Initially, a 10-fold cross-validation was used to assess classification capabilities measuring the accuracy (not the profitability) of the models in the retraining experiments. Every retraining procedure returns a rate of success from a 10-fold cross-validation routine using the current sliding window as a training set. Finally, all the individual values from each training are averaged to present an expected performance metric of the models. The average 10-fold cross-validation value for each experiment set-up is finally presented as an evaluation metric in the general results.

The financial prediction problem must be assessed also from a different perspective due to problems that can invalidate the prediction results such as *bias, variance* and *data snooping*. One of the main issues in ML is to find a trade-off between *bias* and *variance* (Witten et al., 2011). *Bias* is understood as the error rate, i.e. the proportion of wrongly classified instances over a whole data set. *Variance* is the error that results from perfectly fitting an overcomplex model to a finite data set, which might not reflect the whole set of patterns present in the universes of instances for a particular application. In these experiments the use of reduced training sets during retraining does not reflect patterns in a global wider historical data set, but in turn will locally minimise the *bias* deliberately avoiding a *variance* minimisation mechanism during predictions made in between retraining periods. Therefore, periodic retraining allows the adjustment of the *bias* of the models in the short term, while at the same time addressing v*ariance* in the long term, avoiding the need of using a larger sample of the time-series to build the models (Maggini et al., 1997). This situation is especially desirable in financial forecasting, where the time-series are influenced by many factors than can possibly be captured by a comprehensive set of technical indicators (Maggini et al., 1997).

The approach followed in this paper uses the same data set at each iteration of parameters, situation usually associated with *data snooping*. According to White (2000), this problem is virtually impossible to avoid in time series analysis, and especially in financial analysis due to the existence of an unique set of historical data. The approach used in this paper is entirely empirical in the sense that the variation of parameters in each experiment attempts to observe the impact in the selected metrics rather than being an attempt to prove the model validity or performance.

To minimize the effects given by data snooping a modified procedure of the leave-one-out cross-validation is used. In the traditional approach of the leave-one-out cross-validation, each instance in the dataset is left out once and the training is performed on all the remaining instances. The results of all classifications, one for each instance in the dataset, are averaged to obtain the error estimate. In this experiments a retraining procedure is effectuated at predefined periods of time. During each retraining, the first instances in the training set are replaced with the latest and more up-to-date instances and the classifiers are discarded and reset using the new training set. This can be seen as a leave-*n*-out over a sliding window where *n* is the retraining period. This procedure ensures unbiased results because the training data is kept separated from the test data and the results of classification over the test data results are never used for optimization as in other ML applications where the optimization is desirable as new data is being processed. In this sense, the models receive unseen data to classify which in theory do not exist because they represent points in the future, approaching the evaluation of the classification capabilities to the usage that the models would experience in real life trading while at the same time avoiding the effect given by data snooping, i.e. a forward-testing mechanism as opposed a back-testing mechanism. Using the leave-*n*-out over a sliding window, allows to calculate the cumulative accuracy along the trading period, which is used in this study to validate the different models, which in addition incorporates the temporal component to the assessment as it evaluates performance over time.

It is important to note that for trading applications, higher accuracy in predictions does not always imply higher profits. If the returns obtained in a series of successive trades are not high enough to overcome the associated trading costs (among others: commissions, spreads and slippage) any trading strategy will eventually lose money, even though the strategy seemed to be profitable on paper. In this sense, the cumulative return over an extensive period of time represents a more adequate metric for this type of study. Cumulative return takes into account the underlying accuracy of the predictions that triggered the series of trades. In addition, it also represents a weighted average of the successful rate as the positive predicted trades are associated with the individual profit obtained. Financial trading strategies are to be evaluated by observing their historical trading track records, so the returns are mandatory, which in turn represents a clear and simple performance of investment indicator of interest for financial traders. The results of the trading simulations are assessed using the following metrics:

- Accuracy (%): percentage of accurate predictions for the entire trading period.
- UP_Accuracy (%): percentage of accurate UP predictions for the next trading period.
- DOWN_Accuracy (%): percentage of accurate DOWN predictions for the next trading period.
- Cumulative Return (%): percentage of return accumulated at the end of the trading cycle. The cumulative return for the *i*th period is calculated as:

$$CumulativeReturn_i = (1 + CumulativeReturn_{i-1}) \times (1 + Return_i) - 1 \tag{9}$$

- Maximum drawdown: the maximum drawdown measures the historical maximum peak-to-valley decline of an equity value or in a trading strategy using the cumulative returns to keep track of the movements. In other words, the maximum drawdown indicates the maximum accumulated loss the trading agent experienced while trading.
- Average return per trade: indicates the average return obtained in all the trades at the end of the trading cycle.
- Long_Accuracy (%): long accuracy is tied to the UP_Accuracy since an UP prediction will produce a buy order (long trade). The long accuracy will measure the percentage of those trades that actually presented positive profit without taking into account trade costs.
- Short_Accuracy (%): the short accuracy is tied to the DOWN_Accuracy since a DOWN prediction will produce a sell order (short trade). The short accuracy will measure the

percentage of those trades that actually presented positive profit without taking into account trade costs.

3.3. Machine learning classifiers

From the extensive spectrum of data mining algorithms, six models are used in the study presented in this paper to generate the financial prediction capabilities. In addition to their inherent simplicity, the selection of the algorithms was made in a way that ensured a representative algorithm coming from several approaches traditionally used in data mining was present in the experiments: instance-based classifiers (also called *lazy models*), decision trees and rule learners. The agents' decision modules are built using the WEKA library[5] (Witten et al., 2011). Before discussing the experiments related to financial prediction, it is important to mention the basic characteristics and set-ups used for the classifiers in this work.

The naïve Bayes is an instance-based classifier which generalization rules are derived from the Bayes' theorem of conditional probability. For nominal or discrete attributes, the probability of an attribute to belong to a particular class is calculated by determining their relative frequency in the training set. On the other hand, numeric attributes can be discretized and treated similar to nominal attributes, or otherwise, it is assumed that the values of the attributes follow a particular probability distribution function. In this paper, the normal distribution was assumed to model the numeric attributes.

The K∗ model (Cleary & Trigg, 1995) is also an instance-based classifier that tests a new instance by determining those instances in the training set that are closer to it using a similarity measurement, and assigning the predominant class to the new instance. The distance between two instances in K∗ is calculated by measuring the "*complexity*" of transforming instance *a* into instance *b* using a sequence of predefined operations and calculating the probability of the occurrence of the sequence if the operations are chosen randomly. An important parameter is used in the calculation of probabilities called "*blending parameter*". Selecting the same "*blending parameter*" for all the attributes gives equal weighting to each one of them, and is usually the approach used when applying the K∗ algorithm. WEKA gives the option of setting the blend parameter for the experiments, and in this work, it was set at 20%. The experiments were conducted with the Blend setting modes parameter set to "*spherical*".

The C4.5 (Quinlan, 1993) is an internally structured tree, in which each leaf represents a classification, while the branches that connect the leaf to the root node equate to conjunctions of conditions that lead up to that classification. The C4.5 algorithm makes use of entropy to grow the tree iteratively from the training instances separating the training instances into different branches, according to the values of a specific attribute until all the instances in the branch belong to the same class or when none of the potential splits results in an information gain. For the WEKA usage of the C4.5 classifier, the confidence threshold for the pruning parameter was set to 0.25, and the minimum number of instances accepted per leaf was set at 2.

Logistic Model Tree, LMT (Landwehr, Hall, & Frank, 2005), is a classifier that implements two models for classification: linear logistic regression and tree induction. The implementation of WEKA uses the LogitBoost algorithm, as a numeric optimization tool to estimate the parameters needed in the logistic regression procedure. To find a root node, the LogitBoost algorithm is run over the training set to build a logistic regression model in a five-fold cross-validation iterative process. The data are then split using the C4.5 criterion discussed previously. The building of the model and the splitting of the data are continued as long as there are at least the minimum number of instances (set as parameter) present at a node. For the experiments conducted in this paper, the parameters required by the WEKA API were: the number of iterations for LogitBoost set to 1, the minimum number of instances at which a node can be split set to 15, the weight trimming for LogitBoost was set to 0 for no weight trimming.

The Repeated Incremental Pruning To Produce Error Reduction – RIPPER – rule learner (Cohen, 1995), takes one class at a time, and attempt to create a rule that covers as many instances of that class as possible, by iteratively adding conditions to it that apply only to the class being targeted. The method for picking the best attribute is based on maximization of the accuracy for the class under scrutiny. Rules are created greedily adding antecedents to a rule until it becomes 100% accurate. In each iteration, the condition to be added is selected by testing every possible value for each attribute, and picking the condition with the highest information gain. The parameters used in the WEKA implementation were: number of folds set to 3, the minimal weights of instances within a split set to 2, the number of runs of optimizations set to 2 and the seed for randomization set to 1.

4. Experimental results: trading simulations

A preliminary set of experiments was conducted where the Trading Agent trained a specific ML model and posteriorly simulates a total of 2510 trades over the whole test set period shown in Table 1. The results in this scenario were poor in terms of both accuracy of the predictions and cumulative returns. A subsequent set of experiments was conducted where periodic retraining was used while investigating the performance effect of varying three variables: retraining set size, period of retraining and number of attributes. The inclusion of periodic retraining in the models has shown that profitable trading can be achieved by selecting a trade-off in these variables, which might be different for each ML model. To validate these experiments, the same procedure was used to simulate trades using GPBUSD and EURUSD currency pairs.

4.1. Single training experiments

The first set of experiments consists of an initial training of the ML models using the entire training set described in Table 1 with a total of 5191 instances. Once the models are built, the *Market Agent* feeds the *Trader Agent* with the price information from the off-sample data. The system simulates the trading activity for each one of the selected ML models and generates as output the final order book with the respective trades at the end of the off-sample period.

Table 2 presents the trading results for the six machine learning models and an additional random trader. As can be seen, the accuracy of the models is not higher than 51.5%. Intuitively it is possible to think that an accuracy of approximately 50% in trading is not better than a random guess, as suggested by the results obtained in the cumulative returns which are similar to the negative profits obtained by the random trader. Surprisingly, OneR being the simplest classifier, obtained positive cumulative return at the end of the trading cycle as opposed to the results shown by the other classifiers using the same setup. OneR constructed its model based on the Williams %R oscillator, generating a complex set of more than 200 conditions to test the generated prediction rule creating fine grain set intervals while the value of the indicator might fall when applied to unseen instances. The question that arises at this point is whether the OneR classifier is indeed able to identify the most meaningful attribute to predict the behaviour of the price for the next trading period, or if in this particular case, are the good results a are consequence of mere luck.

[5] WEKA information, API and source code is available at: http://www.cs.waikato.ac.nz/ml/weka/.

Table 2
Simulation results of the USD/JPY trading agents, Single training at inception.

Ticker					USDJPY				
Training set size					5242				
Attributes	Attributes : <hour>,<day>,<closing_price>,<ppc>,<lppc>,<lppcma>,<RSI>,<Williams%R>,<class> (9 attributes)								
	Accuracy	DOWN accuracy	UP accuracy	SHORT accuracy	Long accuracy	MAXDD	Average ret/trade	Cummulative return	Trades
Model	(%)	(%)	(%)	(%)	(%)	(%)	(%)	(%)	
K∗	**49.84**	50.72	49.14	49.02	49.14	29.68	−0.0068	**−13.59**	2510
C4.5	**51.08**	53.06	50.20	51.89	50.20	23.32	0.0034	**1.20**	2510
Jrip	**49.76**	50.59	49.02	49.41	49.02	33.10	−0.0090	**−22.08**	2510
NB	**50.60**	51.65	49.83	50.61	49.83	34.30	−0.0132	**−26.45**	2510
LMT	**50.92**	52.96	50.06	52.15	50.06	26.71	−0.0054	**−10.46**	2510
OneR	**51.27**	50.06	52.96	50.98	50.43	13.13	0.0119	**31.96**	2510
Rand	**48.49**	47.58	49.34	48.26	47.58	39.59	−0.0142	**−31.47**	2510

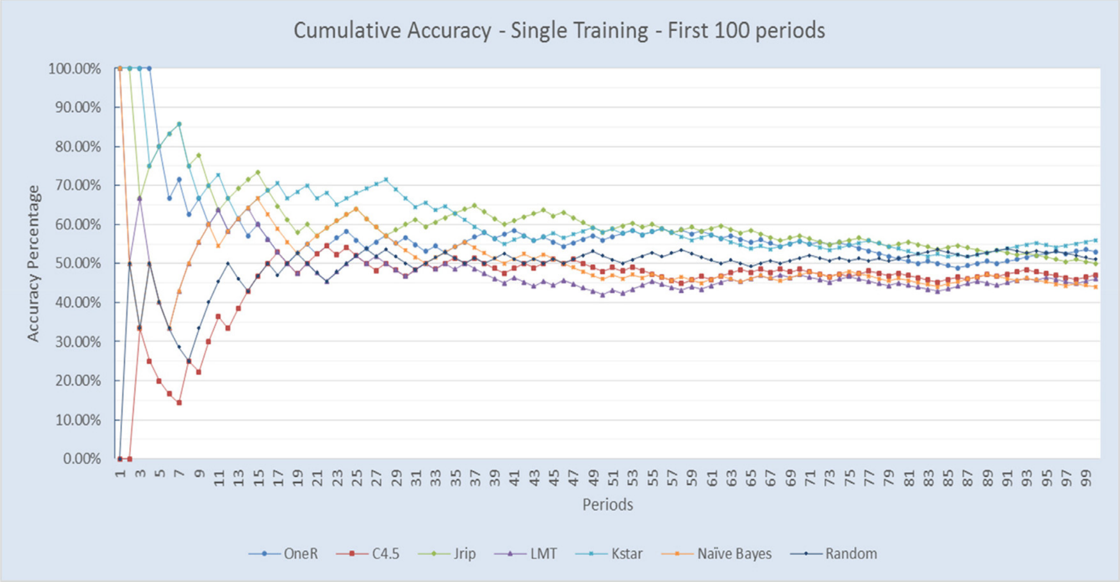

Fig. 2. Cumulative accuracy single training experiment, first 100 trading periods.

It is worth to note form Table 2 is how comparable accuracies, e.g. between K∗ with 49.84% and JRip with 49.76%, leads to considerably different cumulative returns of −13.59% and −22.08%, respectively. The same case is observed for the C4.5 and OneR which have comparable accuracy, 51.08% and 51.27% respectively, yet considerably different cumulative returns of 1.2% and 31.96%. Even though the final accuracy seems to be comparable, the return per trade is showing that the K∗ was less prone to lose money than the JRip in the first case. While it is true that OneR's accuracy is not significantly higher than the other models, what captures attention is its positive average return per trade of 0.0119%, significantly higher than the rest of the classifiers, and which made the model profitable at the end of the trading cycle. Data not present in Table 2 showed that the average return per trade is 0.06537% for the long trades and 0.2691% for short trades, which means that the OneR classifier was extremely profitable in downwards periods for this particular experiment. For the other classifiers, while prediction accuracies may be similar to those observed in the literature (Chen & Shih, 2006; Eng et al., 2008; Kim, 2003; Lee et al., 2007; Li & Kuo, 2008; Tenti, 1996), the average return per trade does not allow the models to have a positive return at the end of the trading period. These initial results suggest that it is possible to generate positive return with modest middle range accuracy. The premise in (Barbosa & Belo, 2008b) points out that a low value of accuracy, such as the ones observed in Table 2, is not necessarily a bad situation if the return per trade resulted from the correctly predicted trades are profitable enough on the long term to overcome the loses of those trades incorrectly predicted. This situation is precisely what an investor would expect in real life trading, where an individual cannot win all the time, just expect to be profitable when trades go well. In these experiments the cumulative return is calculated as a relative metric that is updated every period including the profit of the last trade as shown in Eq. (9).

Fig. 2 presents a detailed cumulative accuracy during the first 100 periods. As one can expect according to the results of Table 2, all the models tend to reach steady values of approximately 50%, but also an interesting feature can be noted: the first 50 trading periods after the initial training showed accuracies over 60% for most of the models, specifically for Naïve Bayes, OneR, K∗ and JRipper. Consequently, it could be presumed that if a periodic retraining process is executed after *n* periods (with *n* approximately less than 50), the cumulative accuracy may perform similar to the initial period and thus sustain a higher average than the 50% obtained for the single training case.

4.2. Retraining experiments

The hypothesis that periodic retraining may increase the average accuracy, motivated the next set of experiments where the retraining period was varied as long as two different sets of attributes were used to create the instances, as shown in Table 3. While the previous experiment followed the set up proposed in (Barbosa & Belo, 2008b), at the time of conducting the experiments

Table 3
Description of the training and test sets – retraining experiments. Two sets of attributes were used.

Currency:	*USDJPY*	
	Number of instances	*Period*
Training set	5242	Wednesday, 02 Jan 2002 00:00 - Thursday, 18 Jan 2007 06:00
Test set (off-sample data):	6442	Thursday, 18 Jan 2007 12:00 - Wednesday, 17 Apr 2013 00:00
	Attributes	
Five attributes	Hour, day of the week, lagged percentage of price change, percentage of price change moving average (10 periods), class.	
Nine attributes	Hour, day of the week, closing price, percentage of price change, lagged percentage of price change, lagged percentage of price change moving average (10 periods), relative strength index, Williams %R, class.	

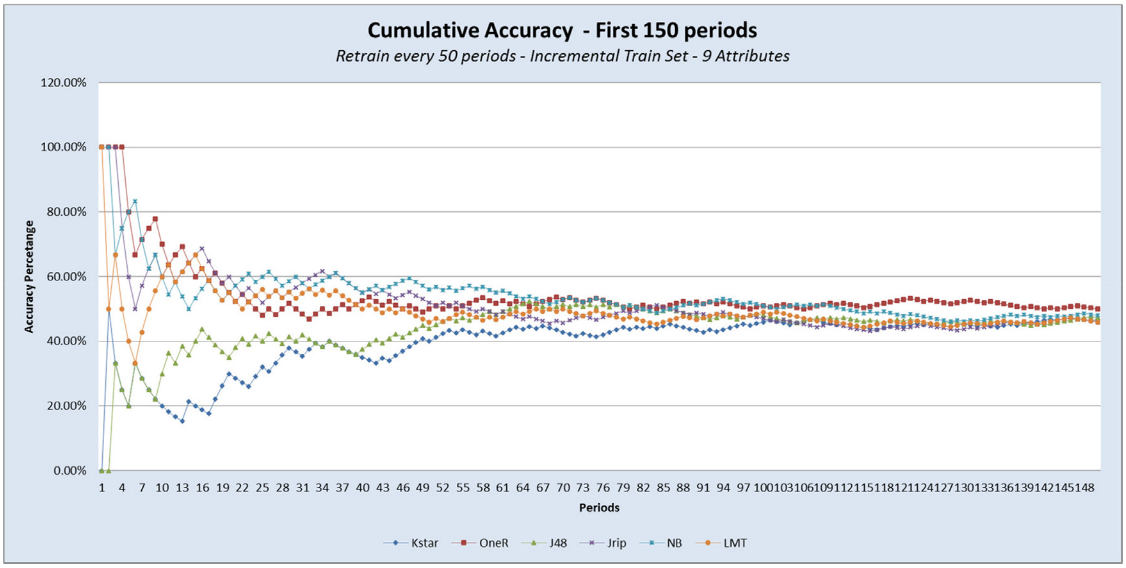

Fig. 3. Cumulative Accuracy for financial prediction of USDJPY discriminated by the machine learning models in the first 150 trading periods, for periodic retraining every 50 periods and incremental training set size. Accuracies for the different classifiers tend to flatten at the final average value from early times in the trading periods which consisted in a total of 6442 points in time.

for this paper, several more years of data were available. Information of prices of four additional years were included in the test set.

4.2.1. Periodic retraining every 50 periods, incremental training set size, 9 attributes

A first periodic retraining experiment was executed, setting the retraining period to 50 trades. The training set size was incrementally grown incorporating the information of the past 50 trading periods as new training instances. As expected, the time consumed in retraining the models was also incremented every new iteration. Contrary to what was expected, the cumulative accuracy did not show the expected recovering behaviour at the retraining points. The tendency to flatten around the 50% for the average accuracy has persisted. Fig. 3 shows the cumulative accuracy for the retraining period $n = 50$ and incremental training set size since inception for the first 150 trading periods.

Table 4 presents the results obtained for retraining the models every 50 trading periods with an incremental training set size. An increment of two points in the total average accuracy was reached for all the models with the exception of the OneR classification which decremented its average accuracy. Additionally, C4.5, JRipper and LMT obtained positive returns, although not significant for a trading period of almost six years. Given the results of Table 4, it is clear that the incremental augmentation of the size of the training set did not show a major effect on the average accuracy. On the other hand, it had a significant effect on the cumulative return reflected on the average returns per trade. This general increment in the profits may be caused by the fact that the periodic retraining allowed the models to learn new unseen patterns in the price trend. This characteristic seems to be critical when trading during 2008 and 2009, years where the financial system suffered erratic behaviours and high volatility.

The training set is incrementing its size every retraining procedure; thus it is also possible to infer that a big set of instances used for retraining might be generating noisy information in the construction of the models, due to the fact that the patterns in the distant past might be poorly related to the current situation of the market and in some cases opposite relationships might be encountered in the training set. The simple models used in these tests might not be able to generalize rules from such a large amount of data. Cumulative results obtained by Naïve Bayes, OneR and K∗ may be evidence of this assumption.

4.2.2. Experiments with variable retraining set size (sliding window), retraining periods and number of attributes

Periodic retraining improved the average accuracy and cumulative profits, but the incremental size of the training set size may be affecting the classifiers' performances in a negative way. While it is important to give the opportunity to the models to learn unseen patterns in the price trends, at the same time, it will be desirable that they are able to predict using a more "*up to date*" training set that represents a fresher situation of the market without taking into account particular conditions located too far in the past. Thus in further tests, four sizes in the training set were selected using the last n instances using a sliding window approach, trying to cover a range of training set sizes. The selected values where $n = [500, 1000, 2000, 4000]$. The retraining period is being also varied every 5, 10, 15 and 20 trades, and the instances are

Table 4
Prediction results for USDJPY. Retraining period = 50, Retraining test size = incremental since inception, 9 attributes.

Ticker	USDJPY										
Retrain set size	Incremental										
Retrain periods	50										
Attributes	Attributes : <hour>,<day>,<closing_price>,<ppc>,<lppc>,<lppcma>,<RSI>,<Williams%R>,<class> (9 attributes)										
	Accuracy	DOWN accuracy	UP accuracy	SHORT accuracy	Long accuracy	MAXDD	Average ret/trade	Cummulative return	Long average ret/trade	Short average ret/trade	Trades
Model	(%)	(%)	(%)	(%)	(%)	(%)		(%)	(%)	(%)	(%)
OneR	**49.86**	51.19	48.59	48.61	48.71	31.59	−0.0005	**−7.22**	−0.0032	0.1560	6441
C4.5	**52.25**	54.57	50.78	52.39	50.74	20.28	0.0104	**87.50**	0.0065	0.3000	6441
Jrip	**50.81**	52.17	49.54	51.34	51.05	23.13	0.0042	**25.81**	0.0015	−0.0239	6441
LMT	**52.84**	54.58	51.41	51.41	52.34	22.42	0.0048	**30.98**	0.0019	0.0213	6441
Kstar	**50.73**	52.05	49.45	50.33	49.88	37.21	−0.0002	**−5.17**	−0.0028	−0.1077	6441
NaiveBayes	**52.59**	53.42	51.51	50.98	51.26	41.07	−0.0034	**−23.13**	−0.0070	−0.1716	6441

constructed using both sets of attributes shown in Table 3. It is important to clarify that the objective of varying the parameters was not to optimize, but an empirical evaluation of the effects on models prediction performance at varying the retaining period and the number of attributes. A heuristic approach was used to vary the parameters. In total 12 different setups were used to cover different combinations of the variables in order to visualize the effects of these parameters in the model predictions and the subsequent simulated trades, i.e. small retraining set size with frequent retraining, medium training set size with higher period and the use of both five and nine attributes, and large retraining set size with less frequent retaining. Table 5 presents a consolidated summary of the 12 different configurations used for the experiments. In all the cases the average accuracy was not higher than 55% for the USDJPY data set, but on the other hand, the cumulative profit was greatly improved.

A group of experiments was conducted with a retraining set of size 500, setups 1–3 in Table 5. Retraining every five trading periods and using five attributes, presented similar effects compared to the results obtained with a retraining period of 10. Average accuracies experienced an increment in one or two points compared to the results observed in Table 4 that use an incremental training set size, reaching values of approximately 52–53% with the exception of OneR that maintained an accuracy between 49% and 50% while at the same time decreasing drastically its cumulative return. What is notable is the substantial increment in the cumulative return for the rest of the models. When retraining occurs every 15 periods using nine attributes, the average accuracies experienced a slight decay, reflected as well in the cumulative profits at the end of the trading season. Eventhough OneR experienced an increment of one point in its accuracy and a substantial increase in the cumulative return, it is still negative. K∗ presents a reversed behaviour observed in 49.7567% of accuracy while maintaining a high positive cumulative return.

The next group of experiments were conducted using a retraining set of size 1000 instances, the results are presented in setups 4–9 in the Table 5. Setup 4 with a retraining period of 10 and using five attributes, exhibited that the OneR classifier benefited from the increased size of the training set compared with the previous setup presenting a slight upturn in the accuracy but on the downsize, it experienced a reduction its cumulative return, which might suggest that the rule learned from a smaller set of attributes is not adequate to generate profits. OneR – One Rule, selects minimum-error attribute for prediction. For this classifier, a greater training set might include higher error values as the number of training instances increased or at the same time, if the number of attributes is incremented, because just one of the attributes may not exhibit enough correlation with the prediction of the class. K∗ did not seem to perceive substantial improvements over the previous setup but again presented a reverse effect when compared with the case of using nine attributes while maintaining the retraining set size and the retraining periods, although it continued presenting positive returns for all the setups using 1000 instances for the training set. For the rest of the models, the increment in the training set, compared with size 500, as well as the reduction of the number of attributes generated beneficial effects in the average accuracy and cumulative profit. What is starting to appear is the fact that higher accuracy may not imply higher return. This is the case for K∗, which showed an accuracy of 49.76% and a cumulative return of 62.47% in setup 3, but exhibited 49.30% accuracy with an impressive 130.74% cumulative return for the setup 6. The opposite effect is observed with JRipper, which obtained 51.92% accuracy with 41.86% cumulative return in setup 3, and 52.32% accuracy with a smaller 12.46% cumulative return in setup 7.

A more frequent retraining seems to have a positive effect on the general classifier performance, but at the same time, having a bigger retraining set compared to the previous set up may produce positive results as well. Naïve Bayes uses normal distribution to estimate the weights of the numeric attributes and a conditional probability derived from the frequency of occurrences for the nominal attributes, thus the bigger training set size will imply wider normal distribution shapes since the longer the training periods, the more prone they will be to exhibit different financial time series regimes. A smaller training set may produce values more closely related to the current trend in the prices, and thus, the model can exhibit higher accuracy. A longer retraining period of 15 using nine attributes per instance used in setup 6, also exhibits a slight increase in most of the accuracies. K∗ presented an impressive cumulative return of 174.64%, higher than the one obtained in other setups which may suggest that this model is better at generating more profitable trades when using the set of nine attributes. The less frequent retraining points seems to affect the JRipper model which was more capable of generating profitable trades retraining every 10 periods instead of 15. LMT seems not to be greatly affected by the change in the number of attributes. Naïve Bayes and C4.5 tend to be more accurate and more profitable with a reduced set of five attributes in all the experiments so far.

A small group of experiments was conducted with a retraining set size of 2000 instances shown in setups 10 and 11 in Table 5. When retraining every 15 periods and using nine attributes, C4.5, JRipper and K∗ increased their accuracy with the increased training set. Nevertheless, again K∗ presented a reverse effect, increasing its accuracy but reducing the capability of generated profitable trades as can be seen in the reduction of the cumulative profit from 174.63% (setup 6) to 137.52% (setup 11), although still

Table 5
Consolidated results for the experiments with variations in retraining set size, retrain period, and number of attributes.

Currency pair:					USDJPY					
Experiment setup	*Retrain set size*	*Retrain periods*	*# of attributes*	Metrics	Machine learning models					
					OneR	C4.5	Jrip	LMT	Kstar	NaiveBayes
Setup 1	*500*	5	*5*	**Accuracy**	49.29	53.38	51.79	53.17	51.79	52.99
				DOWN accuracy	48.35	52.82	51.01	52.86	51.19	52.25
				UP accuracy	50.16	53.82	52.43	53.39	52.81	53.65
				10-fold cross-val.	53.71	51.90	53.51	53.31	48.90	50.90
				Cumulative return	**–55.79**	**116.88**	**18.25**	**106.17**	**69.94**	**142.89**
Setup 2	*500*	10	*5*	**Accuracy**	49.56	52.97	51.41	53.00	51.48	52.74
				DOWN accuracy	48.61	52.35	50.59	52.64	51.16	51.98
				UP accuracy	50.41	53.48	52.05	53.25	52.77	53.42
				10-fold cross-val.	51.70	52.10	52.71	52.30	52.71	51.30
				Cumulative return	**–53.02**	**116.04**	**6.97**	**107.96**	**64.02**	**123.55**
Setup 3	*500*	15	*9*	**Accuracy**	50.49	52.00	51.92	52.76	49.77	52.20
				DOWN accuracy	49.61	51.29	51.29	52.33	50.44	51.50
				UP accuracy	51.36	52.53	52.34	53.04	52.06	52.75
				10-fold cross-val.	51.30	49.10	51.90	51.10	51.90	53.31
				Cumulative return	**–14.69**	**18.61**	**41.86**	**41.48**	**62.47**	**54.76**
Setup 4	*1000*	5	*5*	**Accuracy**	50.01	53.89	52.37	53.72	51.01	53.58
				DOWN accuracy	49.11	53.18	51.53	53.57	51.29	52.86
				UP accuracy	50.87	54.53	53.16	53.82	52.94	54.23
				10-fold cross-val.	50.15	50.85	51.25	50.05	50.85	52.95
				Cumulative return	**–15.35**	**146.06**	**84.50**	**145.19**	**70.22**	**136.94**
Setup 5	*1000*	10	*5*	**Accuracy**	49.98	53.69	52.96	53.70	51.17	53.56
				DOWN accuracy	49.09	52.96	52.25	53.59	51.37	52.84
				UP accuracy	50.84	54.34	53.56	53.78	53.02	54.21
				10-fold cross-val.	51.35	51.35	50.15	49.65	49.75	51.75
				Cumulative return	**–30.87**	**140.40**	**92.24**	**156.82**	**81.05**	**127.91**
Setup 6	*1000*	15	*9*	**Accuracy**	51.13	52.88	52.46	53.45	49.30	52.20
				DOWN accuracy	50.25	52.24	51.82	53.19	50.92	51.43
				UP accuracy	51.98	53.39	52.96	53.64	52.48	52.85
				10-fold cross-val.	50.35	52.45	50.55	52.05	49.85	53.35
				Cumulative return	**–0.67**	**65.21**	**45.69**	**169.88**	**174.64**	**30.91**
Setup 7	*1000*	5	*9*	**Accuracy**	51.53	52.68	52.32	53.17	49.30	52.29
				DOWN accuracy	50.66	52.04	51.63	52.83	50.51	51.52
				UP accuracy	52.37	53.18	52.88	53.41	52.12	52.96
				10-fold cross-val.	51.85	52.15	48.35	51.05	50.55	54.25
				Cumulative return	**4.49**	**50.30**	**12.46**	**105.99**	**130.75**	**30.41**
Setup 8	*1000*	10	*9*	**Accuracy**	51.02	52.94	52.49	53.84	49.30	52.04
				DOWN accuracy	50.14	52.32	51.74	53.69	50.86	51.25
				UP accuracy	51.85	53.44	53.13	53.95	52.41	52.72
				10-fold cross-val.	50.05	52.25	51.05	51.95	51.45	52.75
				Cumulative return	**–10.14**	**74.82**	**74.00**	**156.45**	**145.05**	**26.91**
Setup 9	*1000*	15	*5*	**Accuracy**	49.65	54.03	53.02	53.78	50.86	53.45
				DOWN accuracy	48.75	53.34	52.30	53.71	51.35	52.72
				UP accuracy	50.52	54.64	53.65	53.82	53.00	54.12
				10-fold cross-val.	50.55	51.15	51.45	51.35	51.15	51.55
				Cumulative return	**–31.07**	**147.86**	**59.15**	**153.56**	**74.61**	**132.82**
Setup 10	*2000*	15	*5*	**Accuracy**	49.90	53.86	54.01	53.11	50.70	53.41
				DOWN accuracy	48.99	53.38	53.48	52.99	51.91	52.67
				UP accuracy	50.75	54.23	54.45	53.19	53.45	54.07
				10-fold cross-val.	50.08	53.38	52.93	53.08	50.58	52.28
				Cumulative return	**–3.33**	**181.29**	**137.54**	**28.30**	**74.27**	**72.54**
Setup 11	*2000*	15	*9*	**Accuracy**	49.84	52.99	53.08	53.27	52.10	52.77
				DOWN accuracy	48.93	52.51	52.38	52.91	51.38	52.05
				UP accuracy	50.70	53.34	53.69	53.52	52.89	53.39
				10-fold cross-val.	49.67	52.88	52.48	53.48	51.53	53.63
				Cumulative return	**–12.67**	**98.07**	**68.40**	**25.04**	**137.52**	**1.72**
Setup 12	*4000*	20	*9*	**Accuracy**	50.21	53.28	52.54	52.65	51.32	52.86
				DOWN accuracy	51.89	55.65	53.69	54.18	52.63	53.82
				UP accuracy	49.28	51.33	50.99	51.48	50.00	51.67
				10-fold cross-val.	48.53	53.32	53.42	53.94	51.77	53.34
				Cumulative return	**–3.86**	**104.05**	**53.54**	**14.04**	**31.49**	**17.74**

attractive from an investor point of view. OneR decreased its accuracy and cumulative return as the training set increased. Naïve Bayes also reduced its accuracy and cumulative profit, a situation that offers strong evidence to the premise that the increase of the training set can generate negative effects on the performance for these two models. Although the LMT model decreased its performance, the results are still outstanding, obtaining 137.54% cumulative return in setup 10. For LMT, higher values in its accuracy and cumulative return were obtained with a smaller training set of 1000 when using nine attributes.

For a larger retraining set size of 4000 instances, a longer retraining period of 20 was set, using 9 attributes, as shown in setup 12 in Table 5. It should be noted that as the size of the training set is incremented, the time consumed in building the models is also

Table 6
Combination of parameters for (a) maximum cumulative return and (b) maximum average accuracy, by model. As seen, in some cases, both maximums do not reflex a direct correlation.

(a)	OneR	C4.5	Jrip	LMT	Kstar	NaiveBayes
Accuracy (%)	51.53	53.86	54.01	53.45	49.30	52.99
Max Cumulative return (%)	4.49	181.29	137.54	169.88	174.64	142.89
Retrain set size	*1000*	*2000*	*2000*	*1000*	1000	*500*
Retrain periods	*5*	*15*	*15*	*15*	15	*5*
# of attributes	*9*	*5*	*5*	*9*	9	*5*
(b)						
Max accuracy (%)	51.53	54.03	54.01	53.84	52.10	53.58
Cumulative return (%)	4.49	147.86	137.54	156.45	137.52	136.94
Retrain set size	*1000*	*1000*	*2000*	*1000*	2000	*1000*
Retrain periods	*5*	*15*	*15*	*10*	15	*5*
# of attributes	*9*	*5*	*5*	*9*	9	*5*

greatly increased. In general, the models obtained positive cumulative returns but not as high as the ones obtained with smaller training sets. 4000 instances represent 76% of the initial training set, thus this situation may be replicating the negative effects in the prediction performance observed for the first two setups when no retraining was implemented or an incremental training set was used.

4.3. Results analysis

It is possible to note from Table 5, that the results obtained by the average 10-fold cross-validation does not seem to exhibit a correlation with the cumulative accuracy and the cumulative returns. Taking for example, the Setup 1 and the Setup 2 for the Naïve Bayes model, it can be noted that both the accuracy and the cumulative return decreased simultaneously, from 52.99% to 52.74% for the accuracy and 142.89–123.55% for the cumulative profit, whilst the assessment obtained by the 10-fold cross-validation increased from 50.90% to 51.30%. Following the Setup 3 also for the Naïve Bayes classifier, the 10-fold cross-validation rises even more 53.31% but the accuracy and cumulative return diminished to 52.20% and 54.76, respectively. For the Setups 4 and 5 for the Naïve Bayes model, the 10-fold cross-validation follows decrementing behaviour observed by in the accuracy and cumulative return. Similar contradictory cases can be found along Table 5, suggesting that 10-fold cross-validation might not be the more appropriate method for this type of study. The traditional fold cross-validation divides the data set series into sets used to train and generalize, usually using a random approach. As discussed, the models does not generalize well using a large training set, due to the fact that similar market conditions represented by the particular values of the attributes in the instances can present opposite classes. In those cases, classifiers as the ones used in these experiments are unable to generalize rules. The rationale behind using smaller training sets relies in the fact that points in time closer to the trading period that need to be predicted are more likely to exhibit similar conditions related to the market and therefore the training procedure will use them. A 10-fold cross-validation cannot ensure that the more recent points in time series are kept in the training set during a particular fold, and the obtained performance over the training set might not reflect the future performance of the classifier in the incoming unseen instances.

The models used in the experiments present high sensitivity to the size of the training set and the number of attributes as can be seen in the consolidated results presented in Table 5. Table 6 presents the maximum values obtained by each model under two criteria: (a) maximum cumulative return and (b) maximum average accuracy. OneR, JRipper and K∗ presented a direct correlation between accuracy and cumulative profit. C4.5, LMT and Naïve Bayes share the number of attributes used to construct the models in both cases, but with vary retraining periods and retraining set size.

One of the best set ups for the experiments consisted of a retraining set of size 1000 instances, retrained every 10 periods, and five attributes, in which five out of six classifiers presented positive and high cumulative returns. Fig. 4 presents a comparison of the evolution of the cumulative return over the entire off sample trading period for the classifiers. What is interesting to note in Fig. 4 is that the period of trading between 2007 and 2009 does not present good performance, possibly as a result of very volatile behaviour in the economy for that particular period of time. On the contrary, the plot shows a consistent increase in the cumulative return after 2010 for most of the classifiers with the exception of OneR. This may suggest that the selection of simple ML models for trading may be useful in times of “normal” behaviour of the market, but that such models will not respond well to extreme events such as the economy crisis during 2008 and the subsequent period of recovery. Each model seems to respond to a particular semi-optimal combination of retraining set sizes and number of attributes, but the retraining period does not affect the final accuracies significantly.

Similar experiments were conducted with other currency pairs. Table 7 shows the experimental set ups that produced maximum cumulative returns and maximum accuracies for EURGPB and EURUSD currency pairs. While cumulative returns obtained by the classifiers with these new currencies were not as impressive as the ones obtained in the USDJPY, with an appropriate balance between the retraining set size, retraining period and number of attributes, the models still manage to obtain positive returns. The lower return for EURGPB and EURUSD currency pairs might reflect the fact that those pairs are less volatile than the USDJPY, i.e. the exchange rate moves only a few pips from one trading period to the next limiting the possibility of obtaining high returns per trade.

From a managerial point of view, the trading strategy followed in this paper must be refined. As seen in Fig. 4, the models are subject to individual loses that were especially critical during 2008 and 2009, years where the financial system suffered erratic behaviours and high volatility. Once a more stable situation in the markets appeared, the models tend to recover and follow a gaining trend in their cumulative returns. The year 2008 kept all the models except K∗ on negative profits. The trading simulation does not take into account trading cost and leveraged trading. If the return obtained in each trade is not high enough to cover for the associated costs, the strategy will lose money even though the prediction is accurate. In this sense the average profit per trade can be an important metric to evaluate how much the associated costs affect the strategy performance. Additionally, when trading under

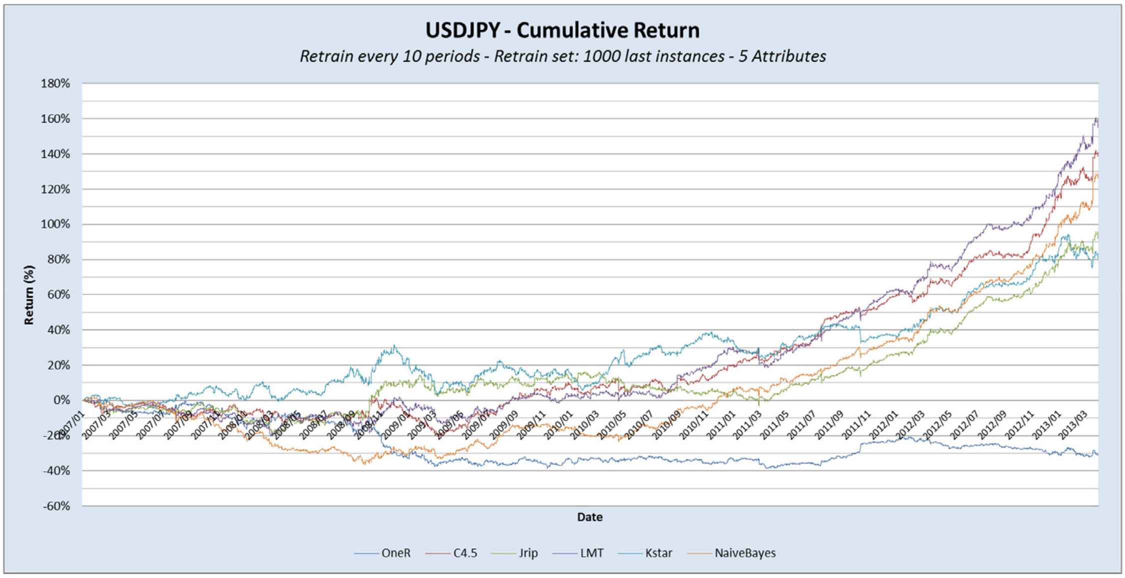

Fig. 4. Cumulative return for the experiment setup: Retraining set Size = 1000; Retraining period = 10; Number of attributes = 5.

Table 7
Experiment set ups for (a) maximum cumulative returns for EURGPB, (b) maximum accuracies for EURGPB, (c) maximum cumulative returns for EURUSD and (d) maximum accuracies for EURUSD.

(a) EURGPB maximum cumulative return	OneR	C4.5	Jrip	LMT	Kstar	NaiveBayes
Accuracy (%)	58.9003	64.0882	63.7465	64.3678	56.8810	63.6844
Max cumulative return (%)	32.0090	57.1697	95.9409	39.4583	73.1667	22.6258
Retrain set size	*500*	*1000*	*1000*	*1000*	500	*1000*
Retrain periods	*15*	*10*	*5*	*5*	15	*15*
# of attributes	*9*	*5*	*5*	*9*	9	*5*
(b) EURGPB maximum accuracy						
Accuracy (%)	58.9003	64.2280	63.7465	64.5076	61.4477	63.6844
Cumulative return (%)	32.0090	27.2639	95.9409	43.4592	21.8619	22.6258
Retrain set size	*500*	*2000*	*1000*	*1000*	2000	*1000*
Retrain periods	*15*	*15*	*5*	*15*	15	*15*
# of attributes	*9*	*5*	*5*	*9*	5	*5*
(c) EURUSD maximum cumulative return						
Accuracy (%)	52.5074	56.4974	56.8079	56.7614	56.9207	53.7184
Max cumulative return (%)	14.5224	33.3007	37.0147	10.3977	92.0795	−9.5586
Retrain set size	*500*	*2000*	*4000*	*500*	500	*500*
Retrain periods	*10*	*15*	*20*	*10*	5	*15*
# of attributes	*5*	*9*	*9*	*5*	5	*9*
(d) EURUSD maximum accuracy						
MaX accuracy (%)	53.3613	57.3513	56.8079	57.7861	58.7869	57.2116
Cumulative return (%)	14.1643	4.5894	37.0147	4.7742	2.8077	−11.8703
Retrain set size	*500*	*1000*	*4000*	*2000*	2000	*4000*
Retrain periods	*15*	*10*	*20*	*15*	15	*20*
# of attributes	*9*	*5*	*9*	*9*	5	*9*

leverage, the profits can be multiplied extensively but also are the losses, where a single trade can generate a margin call. In addition, each of the machine learning models act as an individual trader. A more efficient strategy can be implemented combining the predictions of the different models in a classification ensemble that generates trades according to a weighted voting mechanism based on the past accuracy of the individual models. A weighted decision in addition to risk management strategies such as stop loss and take profit limits must be considered if ML approaches are to be used in a real life automated trading strategy.

4.3.1. Accuracy comparison

Even though the rationale behind this study is to evaluate the trading capabilities of ML models that exhibit a greater degree of simplicity when compared to the traditional NNs and SVM, at this point in the discussion is worth to examine the current results in light of other applications that have focused in those complex models. The following analysis must be taken carefully since the financial instruments that have been assessed are generally not the same, neither are the selection of attributes and training strategies. The main objective of this section is to compare and illustrate the prediction capabilities of machine learning in trading applications.

In (Tenti, 1996) the comparative the performance of three recurrent neural networks is presented based on their returns in the simulated forecasts on currency futures, reporting accuracies between 43% and 48.5%. Wu and Lu (2009) compared a NN model's performance against the ARIMA model, predicting the direction of future values of the S&P 500 Index. The experiments reported a modest 23% level of accuracy against the ARIMA's 42% in more volatile scenarios. In turn, in (Kamruzzaman & Sarker, 2003) the authors compared the performance of the ARIMA model with several NN models when forecasting exchange rates of currency pairs in the FOREX market, showing that all NN models outperformed the ARIMA model with an accuracy of 80%. Taking into account that only the best results obtained were reported and that, in general ML techniques do not present high levels of accuracy with unseen data that differ significantly from the training sets, these impressive results must be considered with care. Using SVM, the work presented in (Kim, 2003) attempted to forecast daily price directions of the KOSPI stock index, using technical indicators (momentum, Williams %R and commodity channel index) as inputs and the best accuracy obtained after training several models with different parameters was 57.83%. The work also presented a comparison with the back propagation NN (with 54.76% of accuracy) and nearest-neighbour model (51.98% of accuracy). In another study (Tay & Cao, 2001) a comparison between SVM and back propagation NN is presented to forecast prices of five types of futures contracts. On average, the SVM approach obtained better accuracy than the back propagation NN, but also in middle-range levels: 47.7% by SVM against 45.0% obtained by back propagation NN. SVM also outperformed a back propagation NN in the work presented by Chen and Shih (2006), where these techniques were used to predict the value of six Asian indices, obtaining 57.2% level of accuracy with SVM and 56.7% with NN models. In addition to simple models, Barbosa and Belo (2008b) also analysed the performance of SVM in the Forex market reporting cumulative accuracy 53.4% with a positive cumulative return of 56.0% trading a currency pair over a two year period.

As seen, the attempts to predict financial time series tend to exhibit modest accuracies, not far from the experiments shown in this paper. Machine Learning approaches are still far from being optimal solutions for financial forecasting due to the complex nature of the financial time series and market erratic behaviours and the possible presence of extreme events that can undermine any generalization or pattern found on them as shown in the capabilities and limitations of the models observed in this paper and in other studies.

5. Conclusions and future work

This paper discusses the usefulness of simple ML models applied to trading scenarios, to verify if it is possible to obtain consistently profitable returns while taking advantage of the computational simplicity of binary classification models. An extensive set of trading simulations in the FOREX market was conducted in which six ML models were used to classify a set of instances constructed from historical price data for USDJP, EURUSD and EURGPB currency pairs. The result of the classification was interpreted as a prediction and used to simulate a series of orders over a trading period of six years. While the paper does not attempt to incorporate new theoretical contribution in the model construction, it provides a valuable insight in the use of machine learning classification to elaborate predictions in financial time series. To the best knowledge of the authors, other related papers focus the construction of sophisticated models that are evaluated by accuracy and/or RMSE. A theoretical contribution in the model construction exceeds the scope of this paper, but opens the door to explore in the construction of machine learning models that are tuned for the particular conditions given by a set of non-discriminatory attributes such as technical indicators and time series related indexes in financial applications.

The selected models present a low computational cost in both the training phase and the classification procedure which benefit their implementation for more frequent trading time frames. The use of simple machine learning allow a seamless upgrade if new attributes are selected. The results have shown that while it is possible to obtain profitability using simple classifiers, each model needs a particular setup taking into account variables such as the retraining period, the retraining set size and the number and type of attributes selected to construct the model. The complexities of the market require a particular combination of the parameters that might change in different market conditions and seasons for the same instrument. The models need to learn new patterns in order to cope with the dynamics of the market, but at the same time, must only use a training comprised of recent values of the time series using a sliding window approach, to avoid noisy patterns that might not be related to the current market situation. A heuristic approach was used to find pseudo-optimal combination of the variables mentioned, but a more systematic approach would be needed to find an optimal trade-off that maximizes both accuracies and cumulative profits, taking into account the particularities of different financial instruments. Most of the models exhibited positive returns for certain experimental setups and combination of parameters, nevertheless their middle range accuracy might be seen as the main weakness of the approach used in this paper.

Attribute selection represents a critical aspect in the construction of the ML models, since a good set of attributes derived from the financial time series will ease the process of classification. In this paper a set of attributes was selected from diverse fields that include price related features such as the price itself and the percentage of price change, seasonality features such as the day of the week and the hour of the day, lagged values used in classical time series analysis such as lagged percentage of price change and its moving average, and of course technical indicators, which in this case are represented by the Williams %R oscillator and the RSI which relates recent gains and losses over the past trading periods. The attributes must be used in conjunction due to the fact that those values by themselves do not produce enough information regarding the future trend of the price that allow to separate effectively the classes in both the training set and the off-sample data, due to the complexity of financial markets. For instance, it is possible to observe is that technical indicators that present an oscillatory behaviour will repeat their values with time as well as the seasonality features. The prices also move typically around certain ranges during short periods of time and the percentage of price change is also comparable from one instance to another when reduced trading periods are selected, i.e. minutes and hours. These attributes have shown not to be highly discriminative observed in instances with similar attributes that might present contradictory classes for different market conditions. Thus the use of attributes coming from technical analysis and classic time series analysis provides means to represent the state of the market in a particular point in time and as it has been demonstrated in this paper, allows to produce profitable trading despite the fact that the obtained accuracy is not impressive. Future work will include the use of an extended set of attributes that explores some from the extensive offer of technical indicators available in the trading literature for the construction of the classifiers. The main risk of a large group of characteristics is the presence of attributes that represent redundant information. To overcome this drawback, a feature selector can also be implemented such as the one presented in (Cai, Hu, & Lin, 2012) which implements a Restricted Boltzmann Machine to extract features from technical indicators which in turn are the inputs for a SVM-based regression model. Future research will also include new ML models, and other markets besides FOREX. Dynamic retraining windows and variable training set sizes can be investigated for particular ML models in order to determine the precise moment in time when a model needs to be replaced with a better setup of parameters.

Acknowledgements

Eduardo Gerlein is supported by a Vice-Chancellor Research Scholarship (VCRS) from the University of Ulster, as part of the Capital Markets Engineering project.

References

Barbosa, R. P. (2011). *Agents in the market place: An exploratory study on using intelligent agents to trade financial instruments* p. 307 (*Ph.D. Thesis*). Universidade do Minho, Escola de Engenharia. Braga, Portugal.

Barbosa, R. P., & Belo, O. (2008). Algorithmic trading using intelligent agents. In H. R. Arabnia, & Y. Mun (Eds.), *Proceedings of the 2008 International Conference on Artificial Intelligence, ICAI 2008* (pp. 136–142). CSREA Press.

Barbosa, R. P., & Belo, O. (2008). Autonomous forex trading agents. In *Proceedings of the 8th industrial conference on Advances in Data Mining: Medical Applications, E-Commerce, Marketing, and Theoretical Aspects. ICDM'08* (pp. 389–403). Springer-Verlag. doi:10.1007/978-3-540-70720-2_30.

Barbosa, R. P., & Belo, O. (2010). The agent-based hedge fund. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 449–452). IEEE. doi:10.1109/WI-IAT.2010.149.

Becket, M., & Essen, Y. (2010). *How the stock market works* (3rd ed., p. 209). London, U.K.: Kogan Page.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). Time series analysis: Forecasting and control p. 598. Wiley.

Cai, X., Hu, S., & Lin, X. (2012). Feature extraction using restricted Boltzmann machine for stock price prediction. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)* (pp. 80–83). IEEE. doi:10.1109/CSAE.2012.6272913.

Chan, E. P. (2008). *Quantitative trading* p. 208. John Wiley & Sons.

Chen, W., & Shih, J. (2006). Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets. *International Journal of Electronic Finance, 1*(1), 49–67. doi:10.1504/IJEF.2006.008837.

Cleary, J. G., & Trigg, L. E. (1995). K∗: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning: Vol. 5* (pp. 1–14).

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning* (pp. 115–123). Morgan Kaufmann.

Di, M. (2007). A survey of machine learning in wireless sensor networks from networking and application perspectives. In *2007 6th International Conference on Information, Communications & Signal Processing* (pp. 1–5). IEEE. doi:10.1109/ICICS.2007.4449882.

Duhigg, C. (2006). *Artificial intelligence applied heavily to picking stocks*. The New York Times. Retrieved September 26, 2014, from http://www.nytimes.com/2006/11/23/business/worldbusiness/23iht-trading.3647885.html?pagewanted=all&_r=0

Ellis, C. D. (2001). *Wall street people: True stories of today's masters and moguls* p. 360. John Wiley & Sons.

Eng, M. H., Li, Y., Wang, Q.-G., & Lee, T. H. (2008). Forecast forex with ANN using fundamental data. In *International Conference on Information Management, Innovation Management and Industrial Engineering* (pp. 279–282). IEEE. doi:10.1109/ICIII.2008.302.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica, 50*(4), 987–1007.

Freed, M., & Lee, J. (2013). Application of support vector machines to the classification of galaxy morphologies. In *2013 International Conference on Computational and Information Sciences* (pp. 322–325). IEEE. doi:10.1109/ICCIS.2013.92.

González, E., Avila, J., & Bustacara, C. (2003). BESA: Behavior-oriented, event-driven and social-based agent framework. In *Parallel and Distributed Processing Techniques and Applications - PDPTA'03*. CSREA Press.

González, E., & Torres, M. (2006). Organizational approach for agent oriented programming. In *8th International Conference on Enterprise Information Systems - ICEIS* (pp. 75–80).

Hendershott, T. (2003). Electronic trading in financial markets. *IT Professional, 5*(4), 10–14. doi:10.1109/MITP.2003.1216227.

Kamruzzaman, J., & Sarker, R. A. (2003). Comparing ANN based models with ARIMA for prediction of forex rates. *ASOR Bulletin, 22*(2), 1–11.

Khan, K., Baharudin, B. B., Khan, A., & E-Malik, F. (2009). Mining opinion from text documents: A survey. In *3rd IEEE International Conference on Digital Ecosystems and Technologies* (pp. 217–222). IEEE. doi:10.1109/DEST.2009.5276756.

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*(1-2), 307–319. doi:10.1016/S0925-2312(03)00372-2.

Landwehr, N., Hall, M., & Frank, E. (2005). Logistic model trees. *Machine Learning, 59*(1-2), 161–205. doi:10.1007/s10994-005-0466-3.

Lee, J. W., Park, J., O, J., Lee, J., & Hong, E. (2007). A Multiagent approach to Q-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 37*(6), 864–877. doi:10.1109/TSMCA.2007.904825.

Li, J., Tsang, E. P. K., & Park, W. (1999). Investment decision making using FGP: A case study. In *Congress on Evolutionary Computation (CEC'99): Vol. 5* (pp. 6–9). doi:10.1.1.170.2021.

Li, S., & Kuo, S. (2008). Knowledge discovery in financial investment for forecasting and trading strategy through wavelet-based SOM networks. *Expert Systems with Applications, 34*(2), 935–951. doi:10.1016/j.eswa.2006.10.039.

Lu, C.-C., & Wu, C.-H. (2009). Support vector machine combined with GARCH models for call option price prediction. In *International Conference on Artificial Intelligence and Computational Intelligence* (pp. 35–40). IEEE. doi:10.1109/AICI.2009.464.

Maggini, M., Giles, C. L., & Horne, B. (1997). Financial time series forecasting using K -nearest neighbors classification. In *Proceedings of the 1st Nonlinear Financial Forecasting Conf. (INFFC'97)* (pp. 169–181). doi:10.1.1.9.291.

McDonald, S., Coleman, S., McGinnity, T. M., Li, Y., & Belatreche, A. (2014). A comparison of forecasting approaches for capital markets. In *Proceedings of 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)* (pp. 32–39). IEEE. doi:10.1109/CIFEr.2014.6924051.

Nawaz, S., & Mirza, N. (2012). Calendar anomalies and stock returns: A literature survey. *Journal of Basic and Applied Scientific Research, 2*(12), 12321–12329.

Nguyen, T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials, 10*(4), 56–76. doi:10.1109/SURV.2008.080406.

Patterson, S., Letting the machines decide, *The Wall Street Journal [on line]- Europe Edition*, Retrieved on February 10, 2016, http://online.wsj.com/article/SB10001424052748703834604575365310813948080.html.

Qi, M., & Zhang, G. P. (2008). Trend time-series modeling and forecasting with neural networks. *IEEE Transactions on Neural Networks, 19*(5), 808–816. doi:10.1109/TNN.2007.912308.

Quinlan, R. (1993). *C4.5: Programs for machine learning* p. 302. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Sewell, M. V., & Yan, W. (2008). Ultra high frequency financial data. In *Proceedings Conference companion on Genetic and evolutionary computation - GECCO '08* (p. 1847). ACM Press. doi:10.1145/1388969.1388988.

Retrieved September 24, 2014, from Smicklas, T. . (2008). http://seekingalpha.com/article/105288-s-and-p-neural-fair-value-25-portfolio-good-source-for-investment-ideas.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega, 29*(4), 309–317. doi:10.1016/S0305-0483(01)00026-3.

Tenti, P. (1996). Forecasting foreign exchange rates using recurrent neural networks. *Applied Artificial Intelligence, 10*(6), 567–582.

Wang, X., Smith, K. A., & Hyndman, R. J. (2005). Dimension reduction for clustering time series using global characteristics. In *Proceedings of the 5th International Conference on Computational Science (ICCS'05) - Part III: Vol. 3516* (pp. 792–795). doi:10.1007/11428862_108.

Wernick, M., Yang, Y., Brankov, J., Yourganov, G., & Strother, S. (2010). Machine learning in medical imaging. *IEEE Signal Processing Magazine, 27*(4), 25–38. doi:10.1109/MSP.2010.936730.

White, H. (2000). A reality check for data snooping. *Econometrica, 68*(5), 1097–1126.

Witten, I. H., Frank, E., & Hall, M. A. (2011). Data mining: Practical machine learning tools and techniques: Practical machine learning tools and techniques (3rd ed, p. 664). Elsevier.

Yamazaki, T., & Ozasa, S. (n.d.). Ex-Goldman Sachs Banker Starts Hedge Fund Analyzing Japanese Blog Traffic. Bloomberg. Retrieved November 08, 2012, from http://www.bloomberg.com/news/2011-04-21/ex-goldman-banker-starts-hedge-fund-analyzing-japanese-blogs.html

Yoo, P. D., Kim, M. H., & Jan, T. (2007). Machine learning techniques and use of event information for stock market prediction: A survey and evaluation. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06): Vol. 2* (pp. 835–841). IEEE. doi:10.1109/CIMCA.2005.1631572.

Zamani, M., & Kremer, S. C. (2011). Amino acid encoding schemes for machine learning methods. In *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)* (pp. 327–333). IEEE. doi:10.1109/BIBMW.2011.6112394.