DESIGN A NEURAL NETWORK FOR TIME SERIES FINANCIAL FORECASTING: ACCURACY AND ROBUSTNESS ANALISYS

LEANDRO S. MACIEL, ROSANGELA BALLINI

Instituto de Economia (IE), Universidade Estadual de Campinas (UNICAMP)
Rua Pitágoras, 65 Cidade Universitária Zeferino Vaz
CEP 13083-857 Campinas – São Paulo – Brasil
Emails: leandro_maciell@hotmail.com; ballini@eco.unicamp.br

ABSTRACT
Neural Networks are an artificial intelligence method for modeling complex target functions. For certain types of problems, such as learning to interpret complex real-world sensor data, Artificial Neural Networks (ANNs) are among the most effective learning methods currently know. During the last decade they have been widely applied to the domain of financial time series prediction and their importance in this field is growing. This paper aims to analyze the neural networks for financial time series forecasting. Specifically the ability to predict future trends of North American, European and Brazilian Stock Markets. Accuracy is compared against a traditional forecasting method, generalized autoregressive conditional heteroscedasticity (GARCH). Furthermore, it is examined the best choice of network design for each sample of data. It was concluded that ANNs do have the capability to forecast the stock markets studied and, if properly trained, can improve the robustness according to the network structure.

Key words: Artificial Neural Networks, Finance Forecasting, Economic Forecasting, Stock Markets.

1. INTRODUCTION

There is a long history of research on finance and economic modeling. Time series analysis is one of the most widely used traditional approaches in this field. There are two kinds of models to describe the behavior of time series as follows. The first are the Linear Models: A linear approach to time series analysis is typically effected through one of the following techniques: (a) Box-Jenkins techniques, (b) Kalman filters, (c) Brown's theory of exponential smoothing, (d) piecewise regression. The second are the Nonlinear Models: (a) Taken's Theorem, (b) the Mackey-Glass equation. These techniques attempt to reconstruct the time series based upon a sampling of the data in order to forecast future values. Although these techniques are statistically powerful, they produce low success rates when they are used to forecasting financial markets.

Recent evidence shows that financial markets are nonlinear; however, these linear methods still provide good ways of describing nonlinear systems found in the financial market time series analysis (Fang *et al.*, 1994). Bollerslev (1986) provide an excellent survey of the existence of nonlinearities in the financial data, and developed a model to predict financial time series called Generalized Autoregresssive Conditional Heterocedasticity (GARCH) that combines all the features observed in these series. But, the economy is evolving (rather slowly) over time. This feature cannot easily be captured by fixed specification linear models, however, and manifests itself in the form of an evolving coefficient estimate. Many factors interact in finance and economics including political events, general economic conditions, and trader's expectations. Therefore, predicting finance and economics movements is quite difficult.

Artificial Neural Networks (ANNs) are a very powerful tool in modern quantitative finance and have emerged as a powerful statistical modeling technique. ANNs provide an attractive alternative tool for both researches and practitioners. They can detect the underlying functional relationships within a set of data and perform tasks such as pattern recognition, classification, evaluation, modeling, prediction and control (Anderson and Rosenfeld, 1988; Hecht-Nielsen, 1990; Hertz *et al.*, 1991; Hiemstra and Jones, 1994). Several distinguishing features of ANNs make them valuable and attractive in forecasting. First, ANNs are nonlinear data-driven. They are capable to perform nonlinear modeling without an *a priori* knowledge about the relationships between input and outputs variables. The non-parametric ANN model may be preferred over traditional parametric statistical models in situations where the input data do not meet the assumptions required by the parametric model, or when large outliers are evident in dataset (Lawrence, 1991; Rumelhart and Mcclelland, 1986; Waite and Hardenbergh, 1998; Wasserman, 1993). Second, ANNs are universal functions approximation. It has been shown that a neural network can approximate any continuous function to any desire accuracy (Hornik, 1993; Hornik *et al.,* 1989). Third, ANNs can generalize. After learning the data presented to them, ANNs can often correctly infer the unseen part of a population even if the sample data contain noisy information. Neural Networks are able to capture the underlying pattern or autocorrelation structure within a time series even when the underlying law governing the system is unknown or too complex to describe.

Some articles have reviewed journal articles on how ANNs can be applied to finance and economic. Wong and Selvi (1998) classified the articles by year of publication, application area, journal, various decision characteristics (problem domain, decision process phase, level of management, level of task interdependence), means of development, integration with other technologies, and major contribution. Zang *et al.* (1998) surveyed articles that addressed modeling issues when ANNs are applied to forecasting. They summarized the most frequently cited advantages and disadvantages of the ANN models. Chatterjee *et al.* (2000) provided an overview of the ANN system and its wide-ranging used in the financial markets. Their work has further discussed the superiority of ANN over traditional methodologies. The study concluded with a description of the successful use of ANN by various financial institutions. Edward Gately, in his book, *Neural Networks for Financial Forecasting,* describes the general methodology required to build, train, and test a neural network using commercially available software.

In this paper we aim to analyze and examine the use of neural networks to predict future trends of North American, European and Brazilian Stock Markets Indexes. The indexes are: Dow Jones and S&P 500 (United States), DAX (Germany), CAC40 (France), FTSE (UK), IBEX35 (Spanish), PSI20 (Portugal) and IBOVESPA (Brazil). We hope to give the detailed discussion of the application of a neural networks tool to forecasting stock markets economic indicators. As a comparison, we analyze GARCH model applied to each series to evaluate the accuracy of ANNs. A discussion about how ANNs can incorporate the heteroscedasticity of financial time series was performed to verify the robustness of the model. This paper is organized as follows. Section 2 discusses applications to stock market index prices forecasting with neural networks. Sections 3-4 describe GARCH and Neural Networks models respectively. Section 5 shows the structure of neural network applied. Performance comparison between the methods is described in Sec. 6. Finally, conclusions are given in Sec. 7.

2. APPLICATIONS STOCK MARKET INDEX FORECASTING

The stock market is one of the most popular investments owing to its high-expected profit. However, the higher expected profit, the higher is the risk implied. The stock market, which has been investigated by various researches, is a rather complicated environment.

There are three degrees of market efficiency. The strong form of the efficient markets hypothesis state that all information that is knowable is immediately factored into the market price for a security. If this is true, then all of those price predictors are definitely wasting their time, even if they have access to private information. In the semi-strong form of the efficient markets hypothesis, all public information is considered to have been reflected in price immediately as it became known, but possessors of private information can use that information for profit. The weak form holds only that any information gained from examining the security past trading history of reflected in price. Of course, the past trading history is public information implying that the weak form is a specialization of the semi-strong form, which itself is a specialization of the strong form of the efficient markets hypothesis.

Stock market fluctuations are the result of complex phenomena, whose effect translates into a blend of gains and losses that appear in a stock market time series that is usually predicted by extrapolation. The periodic variations follow either seasonal patterns of the business cycle in the economy. Short-term and day-to-day variations appear at random and are difficult to predict, but they are often the source for stock trading gains and losses, especially in the case of day traders.

Numerous investigations gave rise to different decision support systems for the sake of providing the investors with an optional prediction. Many experts in the stock markets have employed the technical analysis for better prediction for a long time. Generally speaking, the technical analysis derives the stock movement from the stock´s own historical value. The historical data can be used directly to form the support level and the resistance or they can be plugged into many technical indicators for further investigation. Conventional researches addressing this research problem have generally employed the time series analysis techniques (i.e. mixed auto regression moving average (ARMA)) as well as multiple regression models (Huang *et al.,* 2005). Considerable evidence exists and shows that stock market price is to some extent predictable (Lo and Mackinlay, 1988).

2.1 DISCUSSION OF INPUT VARIABLES

There are two kinds of theoretical approaches to determine the input variables for the stock market index forecasting with neural networks. The first one introduces the relationship among the stock market index price and other macroeconomic indicators. The second one introduces nonlinearity in the relation among stock prices, dividends and trading volume.

Chen (1991) studied the relation between changes in financial investment opportunities and change in the economy. This paper provided additional evidence that variables such as the default spread, term spread, one-month *T*-bill rate, lagged industrial production growth rate and dividend-price ratio are important determinants of the future stock market index. This study interpreted the ability of these variables to forecasting the future stock market index in terms of their correlations with changes in the macroeconomic environment. Fama and French (1993) identified three common risk factors: the overall market factor, factors related to firm size and book-to-market equity, which seem to explain average returns on stocks and bonds. Ferson and Schadt (1996) showed that the omission of variables like lagged stock index and previous interest rates could lead to misleading results. Sitte and Sitte (2000) discussed the predictive ability of time delay neural networks for the S&P 500 index time series.

The vector autoregression (VAR) method is mainly used to investigate the relationship between variables. Its advantage in that multiple variables can be investigated at the same time and the interdependence can be tested automatically with the sophisticated statistically significance level. Ao (2003a; 2003b) found that: (1) HK depends on its past price, JP, Nasdaq, S&P and DJ; (2) AU depends on the past price, S&P and DJ; (3) depends on its past price, HK, Nasdaq, S&P and DJ; (4) JP depends on its past price, Nasdaq, S&P and DJ; (5) DJ depends on its past price and Nasdaq; (6) S&P depends on its past price and Nasdaq. The results from the VAR modeling suggest that, for the Asian markets, the relevant information is the own historical value as well as the stock movements from the US markets. It is also positive to know the extent and time-dependant nature of the markets dynamic when we draw the correlation diagram of the local market with the US markets. Further investigation can tell us that, at the time of low correlation like the late 90s of the Asian Financial crisis, the Hong Kong market (and similarly other Asian markets) is dominated by the local events like the currency problems. At other periods, the local market is greatly correlated with the US markets.

In summary, the set of potential macroeconomic indicators are as follows: term structure of interest rates (TS), short term interest rate (ST), long term interest rate (LT), consumer price index (CPI), industrial production (IP), government consumption (GC), private consumption (PC), gross national product (GNP), gross domestic product (GDP). These are the most easily available input variables that are observable to a forecaster. Though other macroeconomic variables can be used as inputs, the general consensus in the literature is that the majority of useful information for forecasting is subsumed by interest rates and the lagged predictive variable. The term structure of interest rates, i.e. the spread of long-term bond yields over short-term bond yields, may have some power in forecasting the stock index.

In this paper we utilized as input variables the historical data of each series studied[1], which are showed in Table 1. The goal is to analyze the influence of the ANNs structure in the results of forecasting. After chosen the better structure we compared the performance of the ANNs method and GARCH model.

Table 1 – Stock Market Indexes that form the Sample.

Country	*Index*	*Variable*
United States	Dow Jones	DOW
United States	S&P 500	S&P
Germany	DAX	DAX
France	CAC40	CAC
United Kingston	FTSE	FTSE
Spanish	IBEX35	IBEX
Portugal	PSI20	PSI
Brazil	IBOVESPA	IBOV

[1] The data was obtained in http://finance.yahoo.com/ URL accessed on August 17, 2008

3. GARCH MODELS

The ARIMA models have one severe drawback: they assume that the volatility[2] of the variable being modeled (e. g. stock price) is constant over time. In many cases this is not true. Large differences (of either sign) tend to be followed by large differences. In other words, the volatility of asset returns appears to be serially correlated (Campbell *et al.,* 1997).

ARCH (Autoregressive Conditional Heterocedasticity) model were developed in order to capture this property of financial time series. The ARCH [3] process is defined as

ARCH (q): $y_t = \sigma_t \varepsilon_t$ (1)

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{q} \alpha_i y_{t-i}^2} \tag{2}$$

where σ_t is the conditional standard deviation of y_t given the past values of this process. The ARCH(q) process is uncorrelated and has a constant mean, a constant unconditional variance (α_0), but its conditional variance is nonconstant. This model has a simple intuitive interpretation as a model for volatility clustering: large values of past squared returns (y_{t-i}^2), give rise to a large current volatility (Martin, 1998).

The ARCH(q) model is a special case of the more general GARCH(p,q) model defined as (GARCH means "Generalized ARCH")

GARCH(p,q): $y_t = \sigma_t \varepsilon_t$ (3)

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2} \tag{4}$$

In this model, the volatility today depends upon the volatilities for the previous *q* days and upon the squared returns for the previous *p* days.

A long and vigorous line of research followed the basic contributions of Engle and Bollerslev (developers of ARCH and GARCH model respectively), leading a number of variants of the GARCH(p,q) model. These include power GARCH (PGARCH) models, exponential GARCH (EGARCH) models, threshold GARCH (TGARCH) model and other models that incorporate so-called *Leverage effects.* Leverage terms allow a more realistic modeling of the observed asymmetric behavior of returns according to which a "good-news" price increase yields lower subsequent volatility, while "bad-news" decrease in price yields a subsequence increase in volatility. It is also worth mentioning two-component GARCH models which reflect differing short term and long term volatility dynamics, and GARCH-in-the-mean (GARCH-M) models which allow the mean value of returns to depend upon volatility (MARTIN, 1998)[4].

[2] Volatility is the synonym of standard deviation.
[3] This section is based upon Ruppert (2001).
[4] In this work was utilized GARCH(1,1) model as a result of correlation and autocorrelation analysis.

4. NEURAL NETWORKS

Neural Network learning methods provide a robust approach to approximating real-valued, discrete-valued and vector-value target functions. For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks (ANNs) are among the most effective learning methods currently known (Mitchell, 1997).

The study of ANNs has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. In rough analogy, ANNs are built out of a densely interconnected set of sample units, where each unit takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output, which may become input to other units (Mitchell, 1997).

One motivation for ANN systems is to capture this kind of highly parallel computation based on distributed representations. Most ANN software runs on sequential machines emulating distributed processes, although faster versions of the algorithms have also been implemented on highly parallel machines and on specialized hardware designed specifically for ANN applications.

4.1 BASIC DEFINITIONS

The structure of an artificial network of most commonly used type is the multi-layer perceptrons. It consists of several layers of processing units (also termed neurons or nodes). The input values (input data) are fed to the neurons in the so-called *input layer*. The input values are processed within the individual neurons of the input layer and then the output values of these neurons are forwarded to the neurons in the *hidden layer*.

Each connection has an associated parameter indicating the strength of this connection, the so-called *weight*. By changing the weights in a specific manner, the network can "learn" to map patterns presented at the input layer to target values on the output layer. This description of the procedure, by means of which this weight adaptation is performed, is called *learning* or *training algorithm.*

Usually, the data available for training the network is divided in (at least) two non-overlapping parts: the so-called *training* and *testing sets*. The commonly large training set is used to "teach" the network to desire target function. Then the network is applied to data in the test set in order to available its *generalization ability*, i. e. the ability to derive correct conclusions about the population properties of the data from the sample properties of the training set (e. g. if a network has to learn a sine function, it should produce correct results for all real numbers and not only for those in the training set). If the network is not able to generalize, but instead learns the individual properties of the training patterns without recognizing the general features of the data (i. e. produce correct results for training patterns, but has a high error rate in the test set), it is said to be *overfitted* or to be subject to *overfitting.*

4.2 PROPERTIES OF NEURAL NETWORKS

ANN learning is well suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones. It is also applicable to problems for which more symbolic representations are often used, such as the decision tree learning tasks. In this case ANN and decision tree learning produce result of comparable accuracy (Haykin, 2001).

The backpropagation algorithm is the most commonly used ANN learning technique. It is appropriate for problems with the following characteristics (Mitchell, 1997):

- *Instances are represented by many value pairs.* The target function to be learned is defined over instances that can be described by a vector of predefined features, such as the pixel values. These input attributes may be highly correlated or independent of one another. Input values can be any real values.
- *The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.*
- *The training examples my contain errors.* ANN learning methods are quite robust to noise in the training data.
- *Long training times are acceptable.* Network training algorithms typically require longer training times than, say, decision tree learning algorithms. Training times can range from a few seconds to many hours, depending on factors such as the number of weights in the network, the number of training examples considered, and the settings of various learning algorithm parameters.
- *Fast evaluation of the learning target function may be required.* Although ANN learning times are relatively long, evaluating the learning network, in order to apply it to a subsequent instance, is typically very fast.
- *The ability of humans to understand the learning target function is not important.* The weights learned by neural networks are often difficult for humans to interpret. Learned neural networks are less easily communicated to humans than learned rules.

4.3 MULTI-LAYER PERCEPTRONS

The network consists of a set of nodes that constitute the input layer, one or more hidden layers of nodes and an output layer of nodes. The input propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are referred to as multilayer perceptrons (MLPs).

In the 1960s there was a great euphoria in the scientific community about ANN based systems that were promised to deliver breakthroughs in many fields. Single-layer neural networks such as ADALINE[5] were used widely, e. g. in the domain of signal processing. This euphoria was given an end by the publication of Minsky and Papert (1969), who showed that the ANNs used at that time were not capable of approximating target functions with certain properties (target functions that are not linearly separable such as the “exclusive or” (XOR) function). In the 1970s only a small amount of research was devoted to ANNs. In the mid-1980s, the ANNs were “revived” by employment of the *error back-propagation* (EBP) learning algorithm in combination with multi-layer networks (Rumelhart and McClelland, 1986).

Basically, the error back-propagation process consists of two phases through the different layers of the network: a forward pass and a backward pass. In the forward pass, an input vector is applied to the nodes of the network, and its effect propagates through the network, layer by layer. Finally, a set of outputs is produce as the actual response of

[5] ADALINE = ADAptive LInear NEuron.

the network. During this phase the weights are all fixed. During the backward pass the weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired response to produce an error signal. This error is propagated backward through the network, against the direction of synaptic connections – hence the name error back-propagation. The synaptic weights are adjusted so as to make the actual response of the network move closer to desired response (Haykin, 2001).

MLP network consists of at least three layers: input layer, one or more hidden layers and output layer. The nodes are connected by links associated to real number named *weights*. Each node takes multiple values as input, processes them, and produces an output, which can be "forwarded" to other nodes. Given a node *j,* its output is equal to

$$o_j = transfer\left(\sum \left(x_{ji} w_{ji}\right)\right) \tag{5}$$

where o_j is the output of node *j*, x_{ji} is the *i*th input to unit *j*, w_{ji} the weight associated with *i*th input to *j* and *transfer* is the non-linear transfer function responsible for transferring the weighted sum of inputs to some value that is given to the next node[6]. A neuron may have an arbitrary number of inputs, but only one output. By changing the weights of the links connecting nodes, the ANN can be adjusted for approximating a certain function.

4.4 LEARNING ALGORITHMS

Usually, the weights of the ANN must be adjusted using some learning algorithm in order for the ANN to be able to approximate the target function with a sufficient precision. In this section is presented stochastic gradient descent back-propagation learning algorithm as follows[7].

The term "neural network" refers to a MLP trained with this learning algoritm, often called "back-propagation" or "error back-propagation" (EBP). Assume an ANN uses the following error function

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} \left(t_{kd} - o_{kd}\right)^2 \tag{6}$$

where o_{kd} is the output value produced by output neuron *k*, t_{kd} the desire (correct) value this neuron should produce and *D* denotes the set of all training patterns, i. e. $E(\vec{w})$ is the sum of prediction error for all training examples. Prediction errors of individual training examples are in turn equal to the sum of the differences between output values produced by the ANN and the desire (correct) values, where $\vec{w}$ is the vector containing the weights of the ANN.

The goal of a learning algorithm is to minimize $E(\vec{w})$ for a particular set of training examples. There are several ways to achieve this, one of them being the so-called *gradient descent* method. Basically, it works in the following way (Schraudolph and Cummins, 2002):

[6] There are a several types of transfer functions and they can be seen in Haykin (2001).
[7] See more learning algorithms in Haykin (2001).

1. Choose some (random) initial values for the model parameters.
2. Calculate the gradient *G* of the error function with respect to each model parameter.
3. Change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error, i. e., in the direction of –*G*.
4. Repeat steps 2 an 3 until *G* gets close to zero.

Let $G = \nabla f(x)$ the gradient of function *f* is the vector of first partial derivatives

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right) \tag{7}$$

In our case, $G = \nabla E(\vec{w})$ (i. e. the derivative of the error function *E* with respect to the weight vector $\vec{w}$).

Having this in mind, we will now explore the *gradient descent back-propagation* (error back-propagation) learning algorithm. First, a neural network is created and the parameters are initialized (the weights are set to small random numbers). Then, until the termination condition (e.g. the mean squared error of ANN is less than a certain error threshold) is met, all training examples are "taught" the ANN. Inputs of each training example are fed to the ANN, and processed from the input layer, over the hidden layer(s) to output layer. In this way, vector *o* of output values produced by the ANN is obtained.

In the next step, the weights of the ANN must be adjusted. Basically, this happens by "moving" the weight in the direction of steepest descent of the error function. This happens by adding to each individual weight the value

$$\Delta w = \eta \delta_j x_{ji} \tag{8}$$

where η is the *learning rate* that determines the size of the step that we use for "moving" towards the minimum of *E,* and δ_j represents the error term of neuron *j*[8]. The learning rate can be through of as the length of the arrows[9].

There are many improvements of this algorithms such as momentum term, weight decay etc described in the appropriated literature (Bishop, 1996). Nevertheless, MLP in combination with stochastic gradient descent learning algorithm is the most popular ANN used in practice[10]. Another important feature of this learning algorithm is that it assumes a quadratic error function, hence it assumes there is only one minimum. In practice, the error function can have – apart from the global minimum – multiple local minima. There is a danger for the algorithm to land in one of the local minima and thus not be able to reduce the error to highest extent possible by reaching a global minimum. Next section shows an ANN design to our data, and step by step a comparison with GARCH model.

[8] Stochastic gradient descent backpropagation learning algorithm derivative can be see in Haykin (2001).

[9] Usually, $\eta \in \Re, 0 < \eta \leq 0.9$. Note that too large η leads to oscillation around the minimum, while to small η can lead to a slow convergence of the ANN.

[10] This structure was utilized in the present work.

5. DESIGN OF ANN IN STOCK MARKET FORECASTING

The methodology described in this section is based upon Kaastra and Boyd (1996). The design of a neural network successfully predicting a financial time series is a complex task. The individual steps of this process are listed bellow:

1. Variable Selection
2. Data Collection
3. Data Preprocessing
4. Data Partitioning
5. Neural Network Design
6. Training ANN

Detailed description of each step is presented below.

5.1 VARIABLE SELECTION

Success in designing a neural network depends on a clear understanding of the problem (Nelson and Illingworth, 1991). Knowing which input variables are important in the market being forecasted is critical. This is easier said than done because the very reason for relaying on a neural is for its powerful ability to detect complex nonlinear relationships among a number of different variables. However, economic theory can help in choosing variables which are likely important predictors. At this point in design process, the concern is about the raw data from which a variety of indicators will be developed. These indicators will from the actual inputs to the neural networks (Kaastra and Boyd, 1996).

The financial researcher interested in forecasting market prices must decide whether to use both technical and fundamental economic inputs from one or more markets. Technical inputs are defined as lagged[11] values of dependent variable[12] or indicators calculed from the lagged values.

The model applied in this paper uses lagged values of the dependent variables as a result of correlation and autocorrelation analysis[13]. Table 2 shows input structures performed for each data utilized.

[11] "Lagged" means an element of the time series in the past. For example, at time *t*, the values $y_{t-1}, y_{t-2}, y_{t-p}$ are said to be lagged values of the time series *y*.
[12] Dependent variable is the variable whose behavior should be modeled or predicted (Doughrty, 1992).
[13] Such models have outperformed traditional ARIMA-based models in price forecasting, although not in all studies (Sharda and Patil, 1994; Tang *et al.,* 1990).

Table 2 – Variables Selection

Variables	*Input Past Closing Values*
DOW	$DOW_{t-1}, DOW_{t-2}, DOW_{t-3}$
S&P	$S\,\&\,P_{t-1}, S\,\&\,P_{t-2}$
DAX	DAX_{t-1}, DAX_{t-2}
CAC	$CAC_{t-1}, CAC_{t-2}, CAC_{t-3}, CAC_{t-4}$
FTSE	$FTSE_{t-1}, FTSE_{t-2}, FTSE_{t-3}$
IBEX	$IBEX_{t-1}, IBEX_{t-2}$
PSI	$PSI_{t-1}, PSI_{t-2}, PSI_{t-3}, PSI_{t-4}$
IBOV	$IBOV_{t-1}, IBOV_{t-2}, IBOV_{t-3}$

The frequency of the data depends on the objectives of the researcher. A typical of-floor trader in the stock or commodity futures markets would likely use daily data if design a neural network as a component of an overall trading system. An investor with a longer term horizon may use weekly or monthly data as inputs to the neural network to formulate the best asset mix rather than using a passive buy and hold strategy (Kaastra and Boyd, 1996).

5.2 DATA COLLECTION

The research must consider cost and availability when collecting data for the variables chosen in the previous step. Technical data is readily available from many vendors at a reasonable cost whereas fundamental information is more difficult to obtain. Time spend collecting data cannot be used for preprocessing, training and evaluating network performance. The vendor should have a reputation of providing high quality data; however, all data should still be checked for errors by examine day to day changes, ranges, logical consistency and missing observations (Kaastra and Boyd, 1996).

Missing observations which often exist can be handled in a number of ways. All missing observations can be dropped or a second option is to assume that the missing observations remain the same by interpolating or averaging from nearby values. In this work, we assume that there are not missing observations in the sample and, some values that can be seen as *outliers* are presents in the data, because we aim to modeling stock markets mainly in turbulence scenes, characterized by low losses.[14]

5.3 DATA PROCESSING

As in most other neural networks applications, data processing is crucial for achieving a good prediction performance when applying neural networks for finance time series prediction. The input and output variables for which the data was collected are rarely fed into thee network in raw form. As the very least, the raw data must be scaled between the upper and lower bonds of the transfer functions (usually between zero and one minus one and one).

Two of the most common data transformations in both traditional and neural network forecasting are first differencing and taking logarithm of a variable. First

[14] The sample beginning on January 12, 2000 and finished on July 27, 2008.

differencing, or using changes in a variable, can be use to remove a linear trend of data. Logarithmic transformation is useful for data which can take on both small and large values. Logarithmic transformations also convert multiplicative or ratio relationships to additive which is believed to simplify and improve the network training (Masters, 1993)[15]. In this work we used the logarithmic transformation of return

$$R_t = \ln\left(\frac{Index_t - Index_{t-1}}{Index_{t-1}}\right) \tag{9}$$

where R_t represents the normal logarithmic of returns. This approach is especially useful in financial time series analysis and produce good results according to the literature (see Fama, 1965; Granger and Morgenstern, 1970). Also, the returns behavior is more approximated to a Normal probability distribution, but, as will be show in this work, it is a very hardly hypothesis.

5.4 DATA PARTIONING

Common practice is to divide the time series into three distinct sets called the training, testing and validation[16] (out-of-sample) sets. The training set is the largest set and is used by neural network to learn the patterns present in data. The testing set, ranging in size from 10% to 30% of the training set, is used to evaluate the generalization ability of a supposedly trained network. A final check on the validation set chosen must strike a balance between obtaining a sufficient sample size to evaluate a trained network and having enough remaining observations for both training and testing. The validation set should consist of the most recent contiguous observations. In this work the approach in evaluation neural networks used as fallows:

1. Training Set: 80%
2. Testing Set: 15%
3. Validation Set: 5%

5.5 NEURAL NETWORK DESIGN

There are an infinite number of ways to construct a neural network. *Neurodynamics* and *architecture* are two terms used to describe the way in which a neural network is organized. The number of input neurons is one of the easiest parameters to select once the independent variables have been reprocessed because each independent variable is represented by its own input neuron[17]. The tasks of selection of the number of hidden layers, the number of the neurons in the hidden layers, the number of input neurons as well as the transfer functions are much more difficult.

[15] Another popular data transformation is to use ratios of input variables. See Tomek and Querin (1984).
[16] In some of studies, the term *testing set* is used as a name for the validation set.
[17] Each data has its specific input variables as described in Table 1.

5.5.1 NUMBER OF HIDDEN LAYERS

The hidden layer(s) provide the network with its ability to generalize. In practice, neural networks with one and occasionally two hidden layers are widely used and have performed very well. Increasing the number of hidden layers also increases computation time and the danger of overfitting which leads to poor out-of-sample forecasting performance. In the case of neural networks, the number of weights, which is inexorably linked to the number of hidden layers and neurons, and the size of the training set (number of observations), determine the likelihood of overfitting (Baum and Haussler, 1989). It was applied neural networks structure with one and two hidden layers to a comparison.

5.5.2 NUMBER OF HIDDEN NEURONS

Despite its importance, there is no “magic” formula for selecting the optimum number of hidden neurons. Therefore researches fall back on experimentations. However, some rules of thumb have been advanced. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters (1993). For a three-layer network with n input neurons and m output neurons, the hidden layer would have $\sqrt{n \cdot m}$ neurons. Baily and Thompson (1990) suggest thet the number of hidden layer neurons is a three-layer neural network should be 75% of the number of input neurons. Katz (1992) indicates that the optimal number of hidden neurons will generally be found between one-half to three times the number of input neurons. Ersoy (1990) proposes doubling the number of hidden neurons until the network´s performance on the testing set deteriorates. Klimasauskas (1993) suggests that there should be at least five times as many training facts as weights, which sets an upper limit on the number of input and neurons. Because of these features, this work applied different structures to all data, chosen randomly, with 22, 34, 40, 52 and 60 neurons in the hidden layer as to describe the best structure according to the index.

5.5.3 NUMBER OF OUTPUT NEURONS

Deciding on the number of neurons is somewhat more straightforward since there are compelling reasons to always use only one output neuron. Neural network with multiple outputs, especially if these outputs are widely spaced, will produce inferior results as compared to a network with a single output (Masters, 1993). The modeling applied in this work aims to one day past closing value in the future forecasting, and as cited above, was utilized one output layer structure.

5.5.4 TRANSFER FUNCTION

The majority of current neural network models use the sigmoid transfer function, but others such as the tangens hyperbolicus, arcus tangens and linear transfer functions have also been proposed (Haykin, 2001).

Linear transfer functions are not useful for nonlinear mapping and classification. Levich and Thomas (1993) and Kao and Ma (1992) found that financial markets are nonlinear and have memory suggesting that nonlinear transfer functions are more appropriate. Transfer functions such as the sigmoid are commonly used for time series data because they are nonlinear and continuously differentiable which are desirable properties for network learning. In this study, sigmoid transfer function was applied in the network proposed[18].

5.6 TRAINING THE ANN

Training a neural network to learn patterns in the data involves iteratively presenting it with examples to the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of the error function. Unless the model is overfitted, this set of weights should provide good generalization. The backpropagation network, applied in this work, uses the gradient descent training algorithm which adjusts the weights to move down the steepest slope of the error surface. Finding the global minimum is not guaranteed since the error surface can include many local minima in which the algorithm can become "struck". This section will discuss when to stop training a neural network and the selection of learning rate and momentum values.

5.6.1 NUMBER OF TRAINING ITERATIONS

Many studies that mention the number of training iterations report convergence from 85 to 5000 iterations (Deboeck, 1994; Klaussen and Uhrig, 1994). However, the range is very wide as 50000 and 191400 iterations (Klimasauskas, 1993; Odom and Sharda, 1992) and training times of 60 hours have also been reported. Training is affected by many parameters such as the choice of learning rate and momentum values, proprietary improvements to the backpropagation algorithm, among others, which differ between studies and so it is difficult to determine a general value for the maximum number of runs.

Also, the numerical precision of the neural network software can affect training because the slope of the error derivative can become very small causing some neural networks programs to move in the wrong direction due to round off errors which can quickly build up in the highly iterative training algorithm. It is recommended that researches determine for their particular problem and test as many randomly starting weights as computational constraints allow (Kaastra and Boyd, 1996). We utilized 500, 1000, 2500, 5000, 8000 and 12000 iterations randomly to choose the best perform to each index.

5.6.2 LEARNING RATE

During training, a learning rate that is too high is revealed when the error function is changing wildly without showing a continued improvement. A very small learning rate also requires more training time. In either case, the research must adjust the learning rate during training or "brainwash" the network by randomizing all weights and changing the learning rate for the new run through the training set.

Initial learning rates used in this work vary widely from 0.1 to 0.9. Most neural network software programs provide default values for learning rate that typically work

[18] Sigmoid transfer function is default in MATLAB® neural network toolbox.

well. Common practice is to start training with a higher learning rate such as 0.7 and decrease as training proceeds. Many network programs will automatically decrease the learning rate as convergence is reached (Haykin, 2001).

6. COMPARISON ANALYSIS

In this section we will go to present the neural network structure implemented for the data that resulted in a minimum error. Also, the results of ANNs and GARCH model are described to a comparison.

Table 3 shows the best neural network structure performed for each index studied.

Table 3 – Neural Network Design

Index	*Inputs*	*Hidden Layer(s)*	*Hidden Neurons*	*Iterations*	*Learning Rate*
DOW	3	2	40	8000	0.4
S&P	2	2	60	5000	0.6
DAX	2	1	22	8000	0.4
CAC	4	2	34	12000	0.5
FTSE	3	1	22	5000	0.7
IBEX	2	1	52	8000	0.5
PSI	4	2	22	2500	0.6
IBOV	3	2	34	80000	0.5

The results show that the choice of structure is different according to the data. Then, do not have any "magic" formula to describe a structure that minimize the error and result in a best result. The best choice have must be search by the randomly alternatives according to the data.

The experimental results revealed that the proposed algorithm provide a promising alternative to stock market prediction resulting in low errors (see Table 4). Table 4 compares the ranked Coefficients of Multiple Determination for each model. The R square value represents the proportion of variation in the dependent variable that is explained by the independence variables. The better the model explains variation in the dependent variable, the higher the R squared value. Without further comparison, the Neural Network best explains variation in the dependent variable, followed by the Regression Model. The ranked error statistics are provided for comparison. These statistics are all based on returns errors between a desire and a neural network output value.

Table 4 – Error Comparison

Index	*R Squared*		*Percentage Mean Error*		*Mean Square Root Error*	
	ANN	*GARCH*	*ANN*	*GARCH*	*ANN*	*GARCH*
DOW	0.97326	0.86327	3.89323	7.73428	0.62152	2.83222
S&P	0.95432	0.73429	2.73273	6.89329	0.87323	3.83282
DAX	0.98732	0.87364	4.98321	8.78383	0.63263	2.71327
CAC	0.94327	0.83272	3.03933	7.32653	0.93289	4.02391
FTSE	0.95342	0.79322	4.32187	6.63733	0.73732	3.93811
IBEX	0.89763	0.86342	3.09323	7.63723	0.83221	2.83917
PSI	0.93721	0.78873	2.67327	6.98430	1.83283	5.63261
IBOV	0.96390	0.80323	2.03115	9.83921	0.63282	3.63783

In Table 4 is relatively easy to visually verify that the neural network model perform better than the regression model. This differs from the model ranking due to R squared values. Neural network model predict the closing value relatively accurately.

In a tentative to evaluate the robustness of the ANN model applied, we analyze the error dimension in sets performed (training, test and validation). The results are measured by Maximum Percent Error (MPE) and Mean Squared Root Error (MSRE) denoted

$$MPE = \max\left\{\frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}\right\} \tag{10}$$

$$MSRE = \sqrt{\frac{1}{n}(y_i - \hat{y}_i)^2} \tag{11}$$

where y_i denote the desire value *i* and $\hat{y}_i$ the neural network output.

The comparison among the network sets is showed in Table 5.

Table 5 – Network sets comparison

Index	*Sets*					
	Training		*Test*		*Validation*	
	MPE	*MSRE*	*MPE*	*MSRE*	*MPE*	*MSRE*
DOW	4.32712	2.12521	4.87126	2.87521	4.91274	3.13134
S&P	7.53272	3.42513	7.92177	3.87532	8.08643	4.05235
DAX	3.34741	2.36282	3.72362	2.76512	4.29347	3.32712
CAC	6.83212	3.78236	7.53132	4.13263	7.35124	4.73512
FTSE	5.97272	3.08221	6.02183	4.02138	6.68724	4.53289
IBEX	6.74162	2.21342	7.21633	3.42156	7.53281	4.02571
PSI	6.26324	4.76532	6.83635	5.73512	7.01963	6.00012
IBOV	4.53127	3.09217	5.02746	4.03821	5.21456	4.63218

We can see in Table 5 that a neural network structure applied to all indexes had the prediction capability and, how has been seen in the low rate of validation errors, a neural network learn with the data and can process good results to forecasting. Finally, the results in Test and Validation sets confirm the generalization capability of a neural network.

One question that we proposed in this work is: Can Neural Network incorporates heteroscedasticity phenomena?

Table 6 provides the residual analysis of each series studied in this work. It includes mean test, test of Normality (Jarque-Bera), test of correlation present in the residuals (Ljung-Box-Pierce Q-Test) and verify the heteroscedasticity in the residuals (Engle´s ARCH Test)[19][20].

Table 6 –Residuals Analysis

Index	*Tests*							
	Mean		*Jarque-Bera*		*Ljung-Box-Pierce*		*Engle´s ARCH*	
	Value	*Statistically zero?*	*Value*	*Normal?*	*Value*	*Correlation?*	*Value*	*Homocedasticity?*
DOW	0.002	Ok	18.972	Ok	28.921	No	12.872	Ok
S&P	0.008	Ok	16.982	Ok	29.990	No	9.8721	Ok
DAX	0.092	Ok	57.923	Ok	30.912	No	14.765	Ok
CAC	0.061	Ok	61.982	Ok	25.821	No	8.8216	Ok
FTSE	0.076	Ok	25.897	Ok	26.732	No	12.872	Ok
IBEX	0.072	Ok	56.932	Ok	33.812	No	15.876	Ok
PSI	0.086	Ok	22.372	Ok	27.978	No	9.991	Ok
IBOV	0.053	Ok	54.862	Ok	31.982	No	13.721	Ok

Analyzing Table 6 we can see that the neural networks residuals for all indexes studied have mean statistically equal to zero, have Normal distribution, there is not correlation between the residuals and, finally, the residuals are homocedasticity. The results show that a neural network structure proposed was capable to series modeling and forecasting, capturing the heteroscedasticity phenomena and confirm the robustness of the method.

7. CONCLUSIONS

This research examined and analyzed the use of neural networks as a forecasting tool. Specifically a neural network's ability to predict future trends of Stock Market Indexes was tested. North American, European and Brazilian Stock Markets Indexes were studied. Accuracy was compared against a traditional forecasting method (GARCH).

While only briefly discussing neural network theory, this research determined the feasibility and practicality of using neural networks as a forecasting tool for the individual investor.

It was concluded that neural networks do have a powerful capacity to forecast all stock market indexes studied and, if properly trained, the individual investor could benefit from the use of this forecasting tool against current techniques for the following reasons:

- When using multiple linear regression, the governing regression assumptions must be true. The linearity assumption itself and normal distribution my not hold

[19] For this tests see Brockwell (1991).
[20] Chebyschev Inequality Test was applied to confirm the results about residuals probability distribution. For all index was confirmed a Normal distribution.

in mostly financial time series. Neural Networks can model nonlinear systems and do not have any assumption about input probability distribution.

- ANNs are universal functions approximation. It has been shown that a neural network can approximate any continuous function to any desire accuracy.
- ANNs can generalized. After learning the data presented to them, ANNs can often correctly infer the unseen part of a population even if the sample data contain noisy information.
- Compared with GARCH model, neural networks are significantly more accurate.
- Heterocedasticity phenomena can be captured by ANNs.

The next step in future works is to integrate neural networks and other techniques such as genetic techniques, wavelet analysis, fuzzy inference, pattern recognition and, traditional time series models, for finance and economic forecasting. The advantages of genetic techniques include adaptiveness and robustness, which avoid neural networks to get stuck at a local optimum. Once the network was trained, tested and identified as being "good", a genetic algorithm was applied to it in order to optimize its performance. The process of genetic evolution worked on the neuron connection of a trained network by applying two procedures: mutation and crossover. The application of hybrid systems seemed to be well suited for the forecasting of financial data. On the other hand, the discussion about input variables can be taken according to each data studied.

ACKNOWLEDGMENT

This work was supported by the Brazilian National Research Council (CNPq) grants 302407/2008-1.

REFERENCES

Ao, S. I. (2003a), "Analysis of the interaction of Asian Pacific indices and forecasting opening prices by hybrid VAR and neural network procedures". In: Proc. Int. Conf. on Computational Intelligence for Modelling, Control and Automation 2003, Vienna, Austria.

Ao, S. I. (2003b), "Incorporating correlated markets' prices into stock modeling with neural network". In: Proc. IASTED Int. Conf. on Modelling and Simulation 2003, Palm Springs, USA, pp. 353-358.

Anderson, J. A. and Rosenfeld, E. (1988), "Neurocomputing: Fundations of research". MIT Press, Cambridge, MA.

Baum, E. B. and Haussler, D. (1989), "What size net gives valid generalization?". **Neural Computation**, 6, pp. 151-160.

Baily, D. and Thompson, D. M. (1990), "Developing neural network applications". **AI Expert,** 12, pp. 33-41.

Bishop, C. (1996), "Neural Networks for Speech and Sequence Recognation". Thompson, London.

Bollerslev, T. R. (1986), "Generalized Autoregressive Conditional Heteroskedasticity". **Journal of Econometrics**, 51, pp. 307-327.

Brockwell, P. J and Davis, R. A. (1991), “Time Series: Theory and Methods”. Second Editon, New York: Springer.

Campbell, J. Y.; Lo, A. W. and Maclinkay A. C. (1997), “The Econometrics of Financial Markets”. Princeton University Press, United Kingston.

Chatterjee, A.; Ayadi, O. F. and Boone, B. E. (2000), “Artificial neural network and the financial markets: A survey”. **Managerial Finance**, 26, pp. 32-45.
Chen, N. (1991), “Financial investment opportunities and the macroeconomy”. **Journal of Finance**, 46, pp. 529-554.

Deboeck, G. J. (1994), “trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets”. Wiley, New York.

Dougherty, C. (1992), “Introduction to Econometrics”. Oxford University Press, New York.

Ersoy, O. (1990), “Tutorial at Hawaii International Conference on Systems Sciences”. January 1990, Hawaii.

Fang, H; Lai, S. and Lai, M. (1994), “Fractal structure in currency futures price dynamics”. **Journal of Futures Markets,** 14, pp. 169-181.

Fama, E. F. (1965), “The behavior of stock markets prices”. **Journal of Business**, 14, pp. 34-105.

Fama, E. F. and French, K. (1993), “Common risk factors in the returns on stocks and bonds”. **Journal of Financial Economics**, 33, pp. 3-56.

Ferson, W. and Schadt, R. (1996). “Measuring fund strategy and performance in changing economic conditions”. **Journal of Finance**, 51, pp. 425-461.

Gately, Edward J. (1996), “Neural Networks for Financial Forecasting”. John Wiley & Sons, New York.

Granger, C. e Morgenstern, O. (1970), “Predictability of stock market prices”. Health Lexington, Massachusetts, USA.

Haykin, S. (2001). “Neural Networks – A Comprehensive Foundation”. IEEE Press, New York.

Hecht-Nielsen, R. (1990), “Neurocomputing”. Addison-Wesley, Reading, MA.

Hertz, J.; Krogh, A. and Palmer, R. G. (1991), “Introduction to the Theory of Neurocomputation”. Addison-Wesley, Reeading, MA.

Hiemstra, C. and Jones, J. D. (1994), “Testing for linear and nonlinear Granger causality in the stock price–volume relation”. **Journal of Finance**, 49, May, pp. 1639–1664.

Hornik, K. (1993), “Some new results on neural network approximation”. **Neural Networks**, 6, pp. 1069-1072.

Hornik, K.; Stinchcomber, M. and White, H. (1989) “Multilayer feedforward networks are universal approximations”. **Neural Networks**, 2, pp. 359-366.

Huang, W.; Nakamori Y. and Wang, S. Y. (2005), "Forecasting stock market movement direction with support vector machine". **Computers & Operations Research**, 32, pp. 2513-2522.

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

Kao, G. W. and Ma, C. K. (1992), "Memories, heteroscedasticity and price limit in currency future markets". **Journal of Future Markets**, 12, pp. 672-692.

Kartz, J. O. (1992), "Developing neural network forecasters for trading". **Technical Analysis of Stocks and Commodities**, 8, pp. 58-70.

Klaussen, K. L. and Uhrig, J. W. (1994), "Cash soybean price prediction with neural networks". In "Conference on Applied Commodity Analysis, Price, Forecasting and Market Risk Management Proceedings", pp. 56-65, Chicago.

Klimasauskas, C. C. (1993), "Applying Neural Networks". In: R. R. Trippi and E. Turban, editors, "Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance", pp. 64-65. Probus, Chicago.

Lawrence, J. (1991), "Introduction to Neural Networks". California Scientific Sortware: Grass Valley, CA.

Levich, R. M. and Thomas, L. R. (1993), "The significance of technical trading rule profits in the foreign exchange market: A bootstrap approach. In "Strategic Currency Investing – Trading and Hedge in the Foreign Exchange Market", pp. 336-365, Probus, Chicago.

Lo, A. W. and Mackinlay, A. C. (1988), "Stock market prices do not follow random walks: Evidence from a simple specification test". **Review of Financial Studies**, 1, pp. 41-66.

Masters, T. (1993), "Practical Neural Network Recipes in C++". Academic Press, New York.

Martin, R. D. (1998), "Garch modeling of time-varying volatilities and correlations". URL: http://fenews.com/1998/Issue4/059802.htm. (URL accessed on July 5, 2008).

Minsky, M. and Papert, S. (1969), "Perceptrons". MIT Press, Cambridge, MA.

Mitchel, T. M. (1997), "Machine Learning". McGraw-Hill.

Nelson, M. M. and Illingworth. (1991), "A Practical Guide to Neural Nets". Addison Wesley, Reading, MA.

Odom, M. D. and Sharda, R. (1992), "A neural network for bankruptcy prediction". In "Proc. IEEE Int. Conf. on Neural Networks", pp. II163-II168, San Diego.

Rumelhart, D. E. and Mcclelland, J. L. (1986), "Parallel Distributed Processing, Explorations in the Microstructure of Cognition". MIT Press: Cambridge, MA.

Ruppert, D. (2001), "GARCH models". URL: http://www.orie.cornell.edu/~davidr/or473/LectNotes/notes/node139.html. (URL accessed on April 25, 2008).

Sharda, R. and Patil, R. B. (1994), “A connectionist approach to time series prediction: An empirical test”. In: Deboeck, G. J., editor, “Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets”, pp. 451-464, Wiley, New York.

Schraudolph, N. and Cummins, F. (2002), “Introductions to Neural Networks”. URL: https://www.icos.ethz.ch./teaching/NNcourse/backprop.html#top. (URL accessed on September 13, 2008).

Sitte, R. and Sitte, J. (2000), “Analysis of the predictive ability of time delay neural networks applied to the S&P 500 time series”. **IEEE Transaction on Systems, Man and Cybernetics**, 30, November, pp. 568-572.

Tang, Z; Almeida, C. and Fishwick, P. A. (1990), “Time series forecasting using neural networks vs. Box-Jenkins Methodology. In: International Workshop on Neural Networks, Auburn, AL.

Topek, W. G. and Querin, S. F. (1984), “Random process in prices and technical analysis”. **Journal of Future Markets**, 4, pp. 15-23.

Waite, T. and Hardenbergh, H. (1989), “Neural nets”. **Programmer´s Journal**, 7, pp. 10-22.

Wasserman, P. D. (1993), “Advanced Methods in Neural Computing”. Van Nostrand Reinhold, New York.

Wong, B. K. and Selvi, Y. (1998), “Neural network applications in business: A review and analysis of the literature”. **Information & Management**, 34, pp. 129-139.

Zang, G.; Patuwo, B. E. and Hu, M. Y. (1998), “Forecasting with artificial neural networks: The state of the art”. **International Journal of Forecasting**, 14**,** pp. 35-62.