REVISTA MEXICANA DE ECONOMÍA Y FINANZAS
Nueva Época
REMEF
(THE MEXICAN JOURNAL OF ECONOMICS AND FINANCE)

Revista Mexicana de Economía y Finanzas
Nueva Época

Volumen 15 Número 1, Enero - Marzo 2020, pp. 105-122

DOI: https://doi.org/10.21919/remef.v15i1.376

(Recibido: 15/febrero/2019, aceptado: 9/agosto/2019)

imef
fundación de investigación

Financial time series forecasting using Artificial Neural Networks

Roberto Gallardo Del Angel[1]

Universidad Veracruzana, México

Abstract

This paper contains a financial forecast using Artificial Neural Networks. The analysis uses the traditional Backpropagation algorithm, followed by Resilient Backpropagation, to estimate the network weights. The use of Resilient Backpropagation Neural Networks solves the learning rate determination problem. Both algorithms are consistent and offer similar predictions. Six major Stock Exchange Market indices from Asia, Europe, and North America were analyzed to obtain hit ratios that could then be compared among markets. A dependent variable was constructed using daily close prices, which was then used for supervised learning and in a matrix of characteristic variables constructed using technical analysis indicators. The time series dataset ranges from January 2000 to June 2019, a period of large fluctuations due to improvements in information technology and high capital mobility. Instead of prediction itself, the scientific objective was to evaluate the relative importance of characteristic variables that allow prediction. Two contribution measures found in the literature were used to evaluate the relevance of each variable for all six financial markets analyzed. Finding that these measures are not always consistent, a simple contribution measure was constructed, giving each weight a geometric interpretation. Evidence is provided that the Rate-of-Change (ROC) is the most useful prediction tool for four aggregate indices, the exceptions being the Hang Seng and EU50 indices, where fastK is the most prominent tool.

JEL Classification: G11, G15, G17

Keywords: Financial Forecasting, Machine Learning, Neural Networks

Predicción financiera de series de tiempo utilizando Redes Neuronales Artificiales

Resumen

Este documento contiene una predicción financiera utilizando Redes Neuronales Artificiales. Hacemos nuestro análisis utilizando el algoritmo de Backpropagation tradicional y luego Backpropagation Resiliente para estimar los pesos en las redes. El uso del algoritmo de Bacpropagation Resiliente permite resolver el problema de la determinación de la tasa de aprendizaje. Ambos algoritmos son bastante consistentes y arrojan predicciones similares. Analizamos seis índices principales de los mercados bursátiles de Europa, Asia y América del Norte para generar índices de aciertos que puedan compararse entre mercados. Usamos precios de cierre diarios para construir una variable de dependiente para dirigir el aprendizaje (aprendizaje supervisado) y una matriz de variables de características construidas utilizando indicadores de análisis técnico. El rango de datos de la serie de tiempo va desde Enero de 2000 a Junio de 2019, un periodo de grandes fluctuaciones debido a mejoras en

[1]Faculty of Economics, Universidad Veracruzana. Avila Camacho s/n esquina Avenida Xalapa. Xalapa, Veracruz. Mexico. Email: rogallardo@uv.mx. I am very grateful to Prof. Hyejin Ku from York University for her academic advice. The author is the sole responsible for any errors.
*No declared funding source for research development

Resumen

la tecnología de la información y una alta movilidad de capital. En lugar de la predicción en sí misma, el objetivo científico es evaluar la importancia relativa de las variables independientes que permiten la predicción. Utilizamos dos medidas de contribución utilizadas en la literatura para evaluar la relevancia de cada variable para los seis mercados financieros analizados. Descubrimos que estas medidas no siempre son consistentes, por lo que construimos una medida de contribución simple que le da a cada peso una interpretación geométrica. Proporcionamos algunas pruebas de que la tasa de cambio (ROC) es la herramienta de predicción más útil para cuatro índices generales, con las excepciones siendo el índice Hang Sheng y EU50, en donde el fastK es el más destacado.
Clasificación JEL: G11, G15, G17
Palabras clave: Pronóstico Financiero, Aprendizaje Automático, Redes Neuronales

1 Introduction

This work is about Artificial Neural Networks (ANN) and their applications to financial time series forecasting. We use two types of algorithms, backpropagation (BP) and resilient backpropagation (RBP), to produce the weights needed for prediction. The final scientific objective is to use the network weights to estimate some measures of relative importance. One of the main difficulties when using non-parametric methods such as ANN is the interpretation and meaning of the weights (parameters) obtained. Though interpretation is difficult due to the nature and purpose of machine learning methods, we intend to offer some conclusions on the importance of the variables used for prediction. In this respect, ANN analysis is the method for obtaining information about which variables are more relevant for forecasting.

The most common architecture for prediction in times series is the single layer or the multi-layer perceptron feed-forward networks. When deciding on the activation function it is common to decide on a sigmoid type, which is the standard when the prediction is on the range between zero and one. The simplest and most common learning rule for forecasting is the *error-correction* type. But perhaps one important parameter needed to feed our analysis is the *learning rate* which decides on how well the updates in the network performs.

When using the traditional Backpropagation algorithm we must do some previous work in order to choose the learning rate that best fits our model. But this could be a time consuming process and there is not always assurance that the network will work well. One way to go around this problem is to use a different algorithm that may endogenously determine this learning rate. We decided to use the Resilient Backpropagation (RBP) algorithm which offers a simple and heuristic method to find the network weights without first determining the learning rate.

The RBP algorithm is an improvement on traditional Neural Networks using backpropagation algorithm, first proposed by Riedmiller and Braun [14] in 1993. Riedmiller developed a flexible algorithm to tackle the main problems in the traditional Backpropagation, in especial the vanishing gradient problem and the need for cross-validation analysis for estimating the learning rate. The new algorithm allowed for weights backtraking and a heuristic adjustable learning rate that improved prediction.

The Artificial Neural Networks history began perhaps since 1940's when McCulloch and Pitts [12] first proposed the idea of simulating neuronal activity using mathematical logic. But it was until early 1960's that the idea that machines can learn was first explored by Rosenblatt [15] with the creation of the Perceptron Algorithm. For the first time the possibilities of Artificial Intelligence were recognized when this algorithm was tested on an IBM 704 computer. Although there were great expectation about artificial intelligence at that time, the computer technology was not well advanced at that time to make Artificial Neural Networks

to work in their full potential. A big lap forward in ANN research was PJ Werbos's 1974 unpublished PhD Dissertation "Beyond regression: new tools for prediction and analysis in the behavioural sciences", where he first proposed backpropagation to train Neural Networks. A detailed explanation on Backpropagation can be found also in Werbos [17]. Backpropagation algorithm was indeed a break-trough that allowed the effective use of gradient descent method in the training of ANN.

The development of the Neocognitron by Fukushima and Miyake [4] inspired the creation of Convolutional Neural Networks (CNN) which are indeed Deep Neural Networks (DNN) with multi layers but where some hidden layers are called convolutional layers as they perform a convolution that connects to the next hidden layers of neurons. This type of neural networks are often used in pattern recognition problems. In 1982 John Hopfield, with his paper Hopfield [6], invented the Hopfield Network with the purpose of modelling human memory. This later was known as Recurrent Neural Network, in which time lapses between hidden layers and neurons were important to model human learning process and memory. Several other types of DNN with different variations and architectures have been created in recent years. Deep learning is currently a field very dynamic with great possibilities.

On the side of financial time series analysis, Sapankevych and Sankar [16] made a survey on the SVM (Support Vector Machines) and focused on times series prediction using SVM. Kim [11] is an analysis on financial time series using SVM on the Korean composite stock exchange market and and compares the SVM results with Artificial Neural Network (ANN) models and finds that the SVM outperforms slightly those of ANN models. Huang et al. [7] predicted the Japanese Stock Exchange index using SVM and concluded that SVM has a better hit ratio than neural networks. Kara et al. [10] is a similar analysis on the Istanbul stock index using ANN and SVM. Contrary to the previous works on financial time series, this work concluded that the Neural Networks performed better than SVM with a higher hit ratio. Cao and Tay [3] is an analysis using Futures contract in the Chicago Mercantile Market using SVM, backpropagation NN and Regularized Radial Basis function NN. Their results also show that SVM outperforms backpropagation NN and has similar performance against Radial Basis function NN.

This work shows the basic formulation of Artificial Neural Networks and their practical application to time series. We introduce financial forecasting using the Resilient Backpropagation Algorithm (RBP), which was proposed by Martin Riedmiller und Heinrich Braun in Riedmiller and Braun [13] in 1992. They published their work the next year in Riedmiller and Braun [14]. This algorithm tries to solve the problem of the learning rate especially in noisy data. Igel and Hüsken [9] is also a work in RBP which explains the weight backtracking technique.

As mentioned before, the final scientific objective is to measure which features are important for prediction in whole financial markets. Huang et al. [8] is a paper that includes different measures for assessing the relative importance of each feature in the neural network prediction. The two importance measures found in the literature are the Garson and Yoon contribution measures but we noticed that both are not highly correlated. The interpretation and comparison between both measures is not straight forward and, in average, the correlation between them is about 0.54 in our analysis. Our hypothesis is that these contribution measures are not well suited to describe the relative importance of each feature. We decided to construct a simple measure in order to describe the importance of each feature variable in the best ANN model obtained.

Given the above scientific objective, we do not focus in model selection techniques. Although prediction depends on the network architecture and other technical choices such as the learning rule or the activation function, the main purpose is to observe which features are better for prediction. This is information is already embedded in the data and its complexity. Although the determination of the *best* model is important for prediction (e.g. evolutionary algorithms) we decided to focus on this subject in future research.

In this work we are going to work with ANN for binary classification with the objective of predicting

ups and downs in the stock exchange indexes. In the first part of this work we introduce Neural Networks and the Resilient Backpropagation Algorithm. In the second part, we use data from six stock exchange markets (Hong Kong, Japan, Germany, Europe50, Canada and Mexico indexes) in order to obtain prediction on the ups and downs on the stock indexes. The final part of this work includes an analysis on the relative importance of the feature variables using different contribution measures.

2 Artificial Neural Networks

2.1 The theory

Artificial Neural Networks using the Backpropagation algorithm is a traditional method for classification and forecasting. Though several versions of Deep Neural Networks (DNN) are now popular powerful tools for analysis, still the backbone behind all architectures of Neural Networks continue to be the gradient descent method used in Feedforward and Backpropagation algorithms. Both ANN and DNN have a wide range of important commercial applications. There have been numerous efforts to design artificial neural networks based on Von Neumann's architecture, trying to produce intelligent programs that mimic biological neural network. Neurons are very special cells in the human brain, interconnected with each other and responding to stimuli using chemical and electric reactions with connections called synapses. The idea of ANN is to simulate neurons stimuli process and let this neurons to learn by themselves.

ANN can perform complex classification problems. For a simple binary classification, the idea is to construct a decision function $h(x)$ to approximate the outcome or label y (which is binary, either zero or one). The decision can be interpreted as a simple weighted function, a linear combination of, let's say, two features x_1 and x_2:

$$h(x;\theta,b) = \theta_1 x_1 + \theta_2 x_2 + b \tag{1}$$

Which can also be written in the form:

$$y = h(\mathbf{x};\theta,b) = \theta^T \mathbf{x} + b \tag{2}$$

The main objective is to find the vector of weights θ and the parameter b (bias) that may be used to describe and predict the outcome y. A Neural process is a biological process that describes how a neural cell learns. A neural cell processes information from stimuli it receives and then uses a series of synapses to pass on new information already processed. An ANN tries to reproduce the same process, using inputs nodes to receive information, one hidden layer to process the information using an activation function, and then output nodes where the processed information is received. Given an output y we must train our network to *learn* and obtain these output values. At the end, the weights θ will be obtained and will be used for prediction.

The process of training a ANN will depend on the activation function we want to use as well as the method to find the appropriate weights recursively. Usually, we may initiate to train the ANN with random input values and then apply weights to every data point that will pass on information to a hidden layer where the information will be processed by an activation function. Weights θ^T are recalculated iteratively until convergence is achieved. It is desirable that the values we get from the decision function $h(\mathbf{x};\theta,b)$ being in a small range, for example, between 0 and 1. One way to achieve this is to map the decision function $h(\mathbf{x};\theta,b)$ into a new function what will give an output between 0 and 1:

$$h(\mathbf{x};\theta,b) = g(\theta^T\mathbf{x}+b) \tag{3}$$

To represent g as a new function of $\mathbf{z}(\mathbf{x}) = \theta^T\mathbf{x}+b$, we may start to use a sigmoid function of the type (although there are other types of functions that may be useful to represent g):

$$g(\mathbf{z}) = \frac{1}{1+e^{-z}} \tag{4}$$

This is akin to a logistic regression function with $g(\mathbf{z}) \in [0,1]$. We call this *activation function* because this function will process the information and put forward an output. This activation function will be in every neuron, in every hidden layer of the neural network.

The decision function will approximate the label $h(\mathbf{x}^{(i)};\theta,b) \approx y^{(i)}$ for every data point $i = 1,...,n$. The idea is to use all the past data to learn the values θ, b and to approximate the values of y (supervised learning). We may try to minimize the sum of square errors (learning rule) as our *objective function* or *loss function* as follows:

$$H(\theta,b) = \sum_{i=1}^{n}(h(\mathbf{x}^{(i)};\theta,b) - y^{(i)})^2 \tag{5}$$

The crucial step is to minimize the error function $(h(\mathbf{x}^{(i)};\theta,b) - y^{(i)})^2$. The main task will be to obtain the parameters θ and b in such way that the error is minimized. One optimization technique is to use iteration in order to approach to the optimum values of θ and b. One optimization method is called the *stochastic gradient descent* algorithm:

$$\theta_1 = \theta_1 + \alpha\Delta\theta_1 \tag{6}$$

$$\theta_2 = \theta_2 + \alpha\Delta\theta_2 \tag{7}$$

$$b = b + \alpha\Delta b \tag{8}$$

Where the $\Delta\theta_i$ and Δb are the gradients of the parameters. The important in this method is to find the gradients in $\Delta\theta_i$ and Δb in order to iterate until convergence. The term stochastic comes from the idea of initializing the algorithm using random weights, usually between zero and one. The idea to start with random weights during iteration in order to avoid local minima. Another problem in this method is the learning rate determination of the network α. If α is too small the algorithm is very slow and if too large can bring about large fluctuation. Additional analysis is needed to find α, for example using Cross-Validation analysis or optimization.

By computing partial derivatives of the error function with respect to the parameters, the gradients become:

$$\Delta\theta_1 = 2\left(g(\theta^T\mathbf{x}+b) - y^i\right)\left(1 - g(\theta^T\mathbf{x}+b)\right)g(\theta^T\mathbf{x}+b)x_1^i \tag{9}$$

$$\Delta\theta_2 = 2\left(g(\theta^T\mathbf{x}+b) - y^i\right)\left(1 - g(\theta^T\mathbf{x}+b)\right)g(\theta^T\mathbf{x}+b)x_2^i \tag{10}$$

$$\Delta b = 2\left(g(\theta^T\mathbf{x}+b) - y^i\right)\left(1 - g(\theta^T\mathbf{x}+b)\right)g(\theta^T\mathbf{x}+b) \tag{11}$$

The stochastic gradient descent algorithm allows to learn the decision function $h(\mathbf{x};\theta,b)$ computing the above gradients by iteration. First we must set a random value for θ and b and use the initial values $\mathbf{x}^i$ and

$\mathbf{y}^i$ to compute the derivatives 9, 10 and 11. Once the parameters θ_1, θ_2 and b have been computed, we use these values in 6,7 and 8 and then go back to the input values $\mathbf{x}^i$ and $\mathbf{y}^i$ again. This iteration continues until convergence, when the decision function $h(\mathbf{x}^*; \theta^*, b^*)$ is obtained and used for prediction. To predict $\mathbf{y}$ we use the parameters θ^* multiplied by $\mathbf{x}$ plus the term b^* after been put through the sigmoid function. In this simple network the inputs represent the first layer l and then a single neuron (perceptron) in the next layer $l+1$ where the activation function processes the information and pass it over to the final output which is in the final layer L.

The gradient descent algorithm is a key feature of an ANN. Although more sophisticated algorithms are being developed, still gradient descent algorithm is still the core method in ANN. There is also the disadvantage of vanishing gradient when weights are too small and make the gradient to go to zero. Perhaps the vanishing gradient problems was the main disadvantage of the ANN and also the main motivation to develop more sophisticated networks.

Another idea is to separate the data into smaller problems and to solve for each problem separately. For example, in our binary classification problem, some data with a label equal to zero will be a single cluster between two separated clusters of data labelled one. Now we will need two decision functions with more parameters and we need to construct a neural network with two neurons. The idea is to make a decision function of decision functions so that to predict the label $h((h_1(\mathbf{x}), h_2(\mathbf{x})); \omega, c) \approx y$. We can use the previous gradient descent method as before, but now we have a more complex structure.

What we are building now is a neural network where the hidden layer that store the activation function $g(z)$ has two neurons. Both inputs now go through the whole network and the parameters ω and θ are now the weights for the decision functions while the biases are now b and c. The new decision function is:

$$\begin{aligned} h(\mathbf{x}) &= g(\omega_1 h_1(\mathbf{x}) + \omega_2 h_2(\mathbf{x}) + c) \\ &= g(\omega_1 g(\theta_1 x_1 + \theta_2 x_2 + b_1) + \omega_2 g(\theta_3 x_1 + \theta_4 x_2 + b_2) + c) \end{aligned} \tag{12}$$

In the above decision function all parameters and biases must be found at the same time using gradient descent. We are iterating forward, which means that iteration to update parameters must go back to each input data point in the training sample $\{\mathbf{x}^i, y^i\}$ in layer l until convergence is accomplished. This iteration process is called *feedforward* Neural Networks.

2.2 Backpropagation algorithm

Another way to learn is to use the *backpropagation algorithm* (BP). But before we may consider the possibility of more complex ANN architectures. We may consider adding more neurons but also additional hidden layers to our Neural Networks, on what is commonly known as Multilayer Perceptron Network. BP requires that once the *feedforward* process has been completed and we have arrived to the output layer L, then we go back on the entire network to perform a *backward* pass for all layers $l = L-1, L-2, ..., 2$. In order to perform a backward pass we must redefine our variable z and the decision function. The function z is now:

$$z^{(l)} = \theta^{(l)T} h^{(l-1)} + b^{(l)} \tag{13}$$

And the decision function becomes:

$$h^{(l)} = g(z^{(l)}) = g(\theta^{(l)T} h^{(l-1)} + b^{(l)}) \tag{14}$$

The first decision function is just the layer of inputs $h^{(1)} = \mathbf{x}$, which are used to update the next decision function $h^{(2)} = g(\theta^{(1)T} h^{(1)} + b^{(1)})$ and so forth until we get the output $h(\mathbf{x}) = h^{(L)} = g(\theta^{(L-1)T} h^{(L-1)} + b^{(L-1)})$

which is a scalar vector. Because we are now using multiple neurons, we can understand $h^{(l)}$ as a vector as well as the input vector $\mathbf{x}$ and the parameters θ and b. We must proceed in a similar way we obtained the gradients in 9, 10 and 11. First we must find the derivative of the function $g(\mathbf{z})$ respect to the parameters θ and b. For the vector of weights:

$$\frac{\partial g(z^{(l)})}{\partial \theta^{(l)}} = \frac{\partial g(z^{(l)})}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial \theta^{(l)}} \tag{15}$$

And for the bias:

$$\frac{\partial g(z^{(l)})}{\partial b^{(l)}} = \frac{\partial g(z^{(l)})}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial b^{(l)}} \tag{16}$$

The second part of the above derivatives, $\partial z^{(l)}/\partial \theta^{(l)}$ and $\partial z^{(l)}/\partial b^{(l)}$ comes from 13:

$$\frac{\partial z^{(l)}}{\partial \theta^{(l)}} = h^{(l-1)} \tag{17}$$

$$\frac{\partial z^{(l)}}{\partial b^{(l)}} = I \tag{18}$$

To find the first part of the above derivatives 15 and 16, we define $\delta^{(l)} = \partial g(z^{(l)})/\partial z^{(l)}$. Because $z^{(l+1)} = \theta^{(l+1)T} h^{(l)} + b^{(l+1)}$, then:

$$\frac{\partial z^{(l+1)}}{\partial h^{(l)}} = \theta^{(l+1)T} \tag{19}$$

and since $h^{(l)} = g(z^{(l)})$, then:

$$\frac{\partial h^{(l)}}{\partial z^{(l)}} = \frac{\partial g(z^{(l)})}{\partial z^{(l)}} = g'(z^{(l)}) \tag{20}$$

Now we can obtain the derivative $\delta^{(l)}$:

$$\delta^{(l)} = \left(\frac{\partial h^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l+1)}}{\partial h^{(l)}} \frac{\partial g(z^{(l)})}{\partial z^{(l+1)}} \right) = \left(\theta^{(l+1)T} \delta^{(l+1)} \right) \odot \left(g'(z^{(l)}) \right) \tag{21}$$

Where the $\odot$ is for the Hadamard product. Now we can find the updates for the gradient descent and perform the backpropagation:

$$\Delta \theta^{(l)} = \frac{\partial g(z^{(l)})}{\partial \theta^{(l)}} = \delta^{(l)} h^{(l-1)T} \tag{22}$$

$$\Delta b^{(l)} = \frac{\partial g(z^{(l)})}{\partial b^{(l)}} = \delta^{(l)} \tag{23}$$

2.3 Resilient Backpropagation Neural Networks (RBP)

The backpropagation algorithm allows the network to learn and get the parameters θ and b in a more refined way. The main drawback of both feedforward and backpropagation is that, some gradients may become zeros and then some neurons may deactivate themselves. In rare cases the data and initial values produce the deactivation of neurons weakening the network and decreasing the predictive power, what is know as *vanishing* gradient problem. This is the main motivation to develop more complex architectures called *Deep* Neural Networks or DNN.

Another algorithm commonly used in ANN is the heuristic *Resilient Backpropagation* algorithm (RBP). This algorithm is a slightly different version of the Backpropagation but instead of using the magnitude of the gradients $\Delta\theta_1$, $\Delta\theta_2$ and Δb we use their sign. The purpose of this modification, first proposed by Riedmiller and Braun [14] in 1993 and applied by Anastasiadis et al. [2] in 2005, was to adapt the learning rate over the entire neural network. The RBP converges faster than the common BP algorithm but keeps its more general attributes. Günther and Fritsch [5] produced an efficient algorithm to estimate the RBP with weight backtraking.

With the Backpropagation algorithm we have seen that the weights are updated following the general form:

$$\theta_{ij}^{(l+1)} = \theta_{ij}^{l} + \alpha\Delta\theta_{ij}^{(l)}$$

Where $\theta_{ij}^{(l)}$ means the weight from neuron j to neuron i in layer l. In the previous section we found that the updated amount $\Delta\theta_{ij}^{(l)}$ was $\Delta\theta^{(l)} = \delta^{(l)}h^{(l-1)}$ (we removed the transpose upper letter T to facilitate explanation). We know that this is also:

$$\Delta\theta_{ij}^{(l)} = \left(\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}}\right)\delta^{(l)}$$

The RBP algorithm proposes that the update is performed with the sign of the derivative rather than the size of it as follows:

$$\Delta\theta_{ij}^{(l)} = -sign\left(\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}}\right)\delta_{ij}^{(l)} \tag{24}$$

Another importance change is that the update parameter $\delta^{(l)}$ is changed to $\delta_{ij}^{(l)}$ in the following form:

$$\delta_{ij}^{(l)} = \begin{cases} min(\eta^{+}\cdot\delta_{ij}^{(l-1)}, \delta_{max}) & ,if\ \frac{\partial z^{(l-1)}}{\partial\theta_{ij}^{(l-1)}}\cdot\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}} > 0 \\ max(\eta^{-}\cdot\delta_{ij}^{(l-1)}, \delta_{min}) & ,if\ \frac{\partial z^{(l-1)}}{\partial\theta_{ij}^{(l-1)}}\cdot\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}} < 0 \\ \delta_{ij}^{(l)} & otherwise \end{cases}$$

Where $0 < \eta^{-} < 1 < \eta^{+}$ are pre-set values. The value of $\delta_{ij}^{(l)}$ is updated in every step according with the change of sign of the derivative in 24 and the above definition. When the sign changes this means that a minimum has been missed so the network applies the either $\eta^{-}\cdot\delta_{ij}^{(l-1)}$ or δ_{min} which ever larger. The reader may notice that this process also eliminates the need of determining exogenously the learning rate α as in the traditional backpropagation algorithm, because the parameter η is a close adaptative substitute.

The methods of weight backtraking is also based on heuristics, and the idea is to keep using previous weights for updating (some weights only). For example, if:

$$\frac{\partial z^{(l-1)}}{\partial\theta_{ij}^{(l-1)}}\cdot\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}} \geq 0,\ then\ \Delta\theta_{ij}^{(l)} = -sign\left(\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}}\right)\delta_{ij}^{(l)}$$

But if less than zero, we use the previous update:

$$\frac{\partial z^{(l-1)}}{\partial\theta_{ij}^{(l-1)}}\cdot\frac{\partial z^{(l)}}{\partial\theta_{ij}^{(l)}} < 0,\ then\ \Delta\theta_{ij}^{(l)} = \Delta\theta_{ij}^{(l-1)}$$

This implementation trick avoid the updating of the learning rate then avoiding using the *otherwise* option

Table 1: Selected features and formulas

Stochastic % K	$\frac{CP_t - LL_{t-n}}{HH_{t-n}} \times 100$
Stochastic % D (Stochastic moving average of K)	$\frac{\sum_{i=0}^{n-1} \%K_{t-i}}{n}$
Slow %D (Moving average of %D)	$\frac{\sum_{i=0}^{n-1} \%D_{t-i}}{n}$
Momentum	$CP_t - CP_{t-4}$
Rate of Change (ROC)	$\frac{CP_t}{CP_{t-n}} \times 100$
Williams' %R	$\frac{H_n - CP_t}{H_n - L_n} \times 100$
A/D Accumulation/Distribution Oscillator	$\frac{H_t - CP_{t-1}}{H_t - L_t}$
Disparity5	$\frac{C_t}{MA_5} \times 100$
Price Oscillator (OSCP)	$\frac{MA_5 - MA_{10}}{MA_5}$
Commodity Channel Index	$\frac{M_t - SM_t}{0.015 D_t}$, where: $M_t = (H_t + L_t + C_t)/3$ $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$, $D_t = \frac{\sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t \rvert}{n}$
Relative Strength Index	$100 - \frac{100}{1+(\sum_{i=0}^{n-1} UP_{t-i}/n)/(\sum_{i=0}^{n-1} DW_{t-i}/n)}$

above. The advantages of using RBP algorithm is that reduces computation with the advantage of similar, if not better, precision. It is very useful when the data contains noise, which means that it preforms well when applied to financial time series data sets. In the next section we will present some results using RBP neural networks.

3 Time Series Forecasting

3.1 The data

The first task in this work is to forecast a time series using binary classification with ANN methods. A basic classification would be to describe the behaviour of a stock or stock index in order to predict its movement. Predicting stock prices is important as we would want to decide if we need to buy or sell a stock or predict the ups and downs of a price index. In this case we would want to define a label $\mathbf{y}$ with binary values $(0, 1)$ with zero for a drop in the stock prices compared to a day before, and one for an increase in the stock prices. We can use for example, the closing prices of stocks or stock indexes to construct this label. This is a practical way to observe movements in the stock prices, as we need to know if prices will go up or down for very practical decision making. For example, we may decide to sell if a stock price is likely to go down or buy when price is likely to go up. Broadly speaking, trading in the stock market is based in these simple decisions, and usually a good trader uses, among other tools, technical analysis to decide whether is time to sell or buy stock. Although we are not applying this binary classification problem to any specific stock, the underlying principle of classification remains the same.

The next question is defining the features that will be used to predict the movements in stock prices. In other words, we need the matrix of features $\mathbf{x}$ that will help to define the label $\mathbf{y}$. We decided to use some technical analysis concepts as in Kim [11], most of them taken from Achelis [1]. Technical analysis indicators will be our features matrix $\mathbf{x}$. An experienced trader may read the concepts in table 1 and along with additional information then try to predict changes in stock prices. These features are mostly ratios of prices, moving averages or both.

Table 1 shows twelve well known technical indicators for trading. These are constructed with simple market data such as closing price (CP), lowest low price (LL), highest high price (HH), high (H) and low

(L) prices during the trading day or period and it is very common to use Moving Averages (MA) for their construction. The stochastic oscillators such as stochastic$\%K$ and $\%D$ are used by traders to know if an stock is overbought or oversold. The Momentum is used to see any change on the price trend while the Rate of Change (ROC) measures the speed of the ascent/descent of every new trend. The Williams' Accumulation/Distribution indicator measures the market pressure, which advices to sell when Williams' A/D is low and stock prices are high and buy when Williams' A/D is high and market prices are low. The Disparity5 is just an ratio of the closing price with respect to the 5 days Moving Average and the OSCP or Price Oscillator is the growth of the moving average. The Commodity Channel Index (CCI) is used as a leading indicator to observe the strength of a stock, whether a stock is overbought or oversold. The Relative Strength Index (RSI) is used as a leading indicator to observe the historical strength of a stock using the ups and downs in historical closing prices. The WilliamsR is a momentum indicator that reflects how the close price compare with the highest price.

All features in table 1 are associated with the prices of the stocks and are used to interpret the trends of stock market prices. The entire data set for a given stock market index will be the label $\mathbf{y}$ and the feature matrix $\mathbf{x}$ that describes the label. All technical analysis indicators will be used to classify our label in both directions, ups and downs for the entire stock market index. There are dozens more technical analysis indicators that can be used, but we are trying to use some the most popular and also applied in other similar research.

This section contains an empirical analysis using RBP algorithm in order to predict time series, particularly changes in stock price indexes. We chose to predict changes in six major European, Asian and North American stock market indexes. We used six stock indexes: The European STOXX50 that contains blue chip stock from the 50 best performing companies in leading sectors in Europe; the DAX which is also an index that contains 30 blue chip German companies; the Nikkei stock exchange index, the Hang Seng index which is the stock exchange index from Hong Kong, the Canadian Toronto Stock Exchange index and the Mexican Stock Exchange index IPC.

We decided to use daily data for each stock exchange from January 2000 to June 2019, less than five thousands daily observation in each market. Compared with the same data from 1980's and 1990's, the period of analysis is high frequency data and contains sharp financial crashes, perhaps due to the new trading methods using electronic platforms and the availability of information online. Financial markers are now more competitive as communication technology has improved along with capital mobility. Table 5 in the appendix contains the summary statistics for the six markets on closing, high, low and open market prices.

3.2 Estimation

With the information on the average prices in each market, we first constructed our matrix of features $\mathbf{x}$ that were used to describe the label $\mathbf{y}$, where $\mathbf{y}$ are the changes in close price. The idea is to approach to the daily changes in close prices in each stocks market. In this case, the classification problem will be to assign a label of 1 if the close price index in time CP_t is higher than the previous day CP_{t-1}, and 0 otherwise so that $\mathbf{y} \rightarrow [0, 1]$. Our data points $\mathbf{x}$ will be the features that help us to find the weights $\mathbf{w}$ and the bias b.

Because ANN is a supervised machine learning method, we are going to demand to the network to find the best way to predict $\mathbf{y}$ using the twelve technical analysis features constructed using indicators in table 1 (matrix $\mathbf{x}$). This matrix will serve as the input data and the main objective of the ANN algorithm is to find patterns in $\mathbf{x}$ so that to approach the vector $\mathbf{y}$. Because the weights produced by the ANN are sensitive to the scale of each feature we must normalize the features matrix $\mathbf{x}$ and transform them into a matrix with values in the range of $(0, 1)$ to ensure large values do not overcome features with small values.

When the whole data set with $\mathbf{y}$ and $\mathbf{x}$ have been constructed, the next step is to divide the data set

into a training and test sets. The training set will be used by each algorithm to learn and the second set for testing if predicted values match with the data points. In our analysis, we used the first 70% data points as training set and the remaining as test data set. In the appendix the reader will find the summary statistics for each feature $\mathbf{x}$ and the label $\mathbf{y}$ by stock market.

The only thing left for clarification will be the estimation of the Hit ratio for each prediction. After running each of the ANN models, we will get the predicted values using the test data into a new data set with predicted values for the label $\hat{\mathbf{y}}$. One way to evaluate the performance of the ANN predictions, is to construct a hit ratio. Because the predicted values are real numbers and our test data set has a binary label, we must transform the predicted values into binary outcomes. We set up a threshold of 0.5 which mean that any predicted values higher than 0.5 will be set up to 1 (close price was higher than in previous day) if not equal to zero (close price was lower than in previous day). The prediction results are matched in the form:

$$R_i = \begin{cases} 1 & if\ Predicted\, value = test\, data \\ 0 & otherwise \end{cases} \tag{25}$$

The prediction performance is measured using a hit ratio, defined by:

$$HR = \frac{1}{m}\sum_{i=1}^{m} R_i \tag{26}$$

This hit ratio is the percentage of correct matches where $\mathbf{y} = \hat{\mathbf{y}}$. This is a simple coefficient that can be used to compare performance (prediction) for each neural network model.

The main part of the empirical analysis requires to use ANN to predict time series. We trained different single layer networks using traditional Backpropagation and Resilient Backpropagation Neural Network algorithm. At first, single layer neural networks were constructed with 6, 12, 18 and 24 neurons each using standard logistic and error functions. Later we trained multi-layer networks with 6 and 12 neurons in three hidden layers. The results are shown in table 2.

Table 2: Financial Forecasting using Artificial Neural Networks

Backpropagation (learning rate=0.1)						
INDEX \ Neurons	6	12	18	24	6-6-6	12-12-12
TSE (Canada)	0.4522	0.4522	0.4522	0.5478	0.4522	0.4522
IPC (Mexico)	0.4905	0.5095	0.4905	0.5095	0.4905	0.4905
Nikkei (Japan)	0.4600	0.5400	0.4600	0.5400	0.4600	0.4600
Europe50	0.4763	0.4763	0.4763	0.5237	0.4763	0.4763
Han Seng (Hong Kong)	0.4757	0.5243	0.4757	0.5243	0.4757	0.4757
DAX (Germany)	0.4612	0.5388	0.4612	0.5388	0.4612	0.4612
Resilient Backpropagation (weight bactracking)						
INDEX \ Neurons	6	12	18	24	6-6-6	12-12-12
TSE (Canada)	0.5478	0.5959	0.6678	0.5458	0.5478	0.5478
IPC (Mexico)	0.5095	0.5194	0.5095	0.5095	0.5095	0.5095
Nikkei (Japan)	0.4892	0.4929	0.5743	0.4899	0.5131	0.5049
Europe50	0.4564	0.4515	0.4522	0.4536	0.4557	0.4529
Han Seng (Hong Kong)	0.5028	0.5125	0.4944	0.4993	0.5271	0.5049
DAX (Germany)	0.5287	0.5273	0.5745	0.5300	0.5388	0.5179

One disadvantage of the ANN is the cost in training increases when the architecture becomes more complex. As the number of neurons and hidden layers increase, the longer the training time is required. On the other hand, ANN with backpropagation may obtain better performance due to a more flexible updating. Table 2 contains the hit ratios for different ANN models with different architectures. The upper part contains the hit ratios using traditional backpropagation with a learning rate of 0.1 while the lower part contains the hit ratios using resilient backpropagation with weight backtracking.

With the only exception being the Europe50 index, we find larger hit ratios using resilient backpropagation. This does not mean that we cannot achieve better results in traditional backpropagation, but for that we need to find the best learning rate and architecture. And this will require additional statistical analysis in order to decide the correct learning rate, as each model is different.

On the other hand, resilient backpropagation has a flexible and heuristic way to choose the learning rate and update the gradient for a better descent. The reader may notice that there is little room for improvement in each model using backpropagation as we use a single learning rate for every model. However, resilient backpropagation has room for improvement as the learning is controlled during convergence. Estimation in table 2 will vary as long as we choose different activation functions, learning rates and gradient methods for updating, but we decided to leave model selection for future research.

4 Contribution measures

This work focuses not only on financial forecasting using ANN but also offers a descriptive analysis on the overall performance of the features used for prediction. This is an important issue because we need information on the relative relevance of each feature in the learning process. We know that each feature was normalized when constructing the matrix $\mathbf{x}$, so we may be able to apply some indicators on similar data and obtain some comparable results.

Table 3: Correlation coefficients: Garson, Yoon and Trapezoid Contribution Measures

Index	Garson\Yoon	Garson\Trapezoid	Yoon\Trapezoid
DAX	0.625	0.816	0.808
NIKKEI	0.789	0.851	0.931
IPC	0.550	0.722	0.908
HS	0.050	0.307	0.881
EU50	0.619	0.621	0.512
TSE	0.511	0.698	0.864

In order to find the relative importance of each feature we must apply a measure using the weights from the ANN analysis. The magnitude of each weight in every network tell us about the relative importance of each feature. This section provides with some measures on the relative contribution of each feature on the final output in a Neural Network. We estimated each contribution measure on the best single hidden layer ANN neural network. For example, if the input layer has $i = 1, 2, ...I$ nodes, and the hidden layer has $j = 1, 2, ..., J$ neurons, the final output weights will come from equal number of nodes $k = 1, 2, ..., J$. At first, we introduce two different measures called Garson and Yoon measures, similar to Huang et al. [8]. The first contribution measure is the Garson measure:

$$GC = \frac{\sum_{j=1}^{J} \frac{|w_{ji}||v_{jk}|}{\sum_{i=1}^{I}}}{\sum_{i=1}^{I}\sum_{j=1}^{J} \frac{|w_{ji}||v_{jk}|}{\sum_{i=1}^{I}|w_{ji}|}}$$

And the second is the Yoon measure:

$$YC = \frac{\sum_{j=1}^{J} w_{ji} v_{ji}}{\sum_{i=1}^{I} |\sum_{j=1}^{J} w_{ji} v_{ji}|}$$

The Garson measure can be interpreted as percentages of contribution on the final output. Yoon contribution index is more complicated to interpret, though we may interpret a high absolute value of Yoon measure as high relevance. Both measures are designed for a single layer Neural Network, then the best single layer results for each model. The results of the estimation are shown in table 4 for each market (the number in the parenthesis shows the number of neurons in the hidden layer). For four markets the ROC seems to be the feature with the highest contribution to the financial forecasting, except for the Hang Sheng index and the Euro50 index. These were the only two indexes where we used the best single layer hit ratio using backpropagation.

Table 4: Contribution Measures for each Feature by Stock Market

Features	TSE (18n)			Nikkei (18n)			DAX (18n)		
	Garson	Yoon	Trapezoid	Garson	Yoon	Trapezoid	Garson	Yoon	Trapezoid
A/D	1.67%	-0.019	2.8%	5.52%	-0.018	2.03%	5.58%	0.027	2.66%
CCI	2.43%	-0.005	3.6%	6.55%	0.015	3.45%	6.56%	-0.006	3.27%
Disp10	7.26%	-0.058	3.8%	8.21%	-0.017	4.12%	7.07%	-0.041	3.01%
Disp5	9.92%	-0.027	3.7%	9.07%	0.013	4.35%	7.16%	-0.006	4.84%
fastD	12.15%	-0.023	10.0%	8.38%	-0.093	10.67%	9.20%	-0.040	7.44%
fastK	8.08%	0.010	3.8%	6.73%	0.085	4.93%	8.37%	-0.040	5.00%
Moment	9.56%	0.035	4.9%	8.42%	0.005	4.66%	10.49%	-0.020	5.85%
OSCP	9.65%	0.015	3.9%	9.19%	0.012	3.54%	8.44%	-0.009	4.05%
ROC	15.73%	0.637	38.5%	13.84%	0.722	42.38%	13.42%	0.558	39.58%
RSI	7.94%	0.010	5.4%	8.70%	-0.001	3.61%	8.82%	0.042	5.75%
slowD	4.08%	-0.020	5.3%	6.49%	0.005	4.39%	6.04%	0.011	4.18%
WilliamsR	11.53%	-0.143	14.3%	8.89%	0.015	11.87%	8.84%	-0.201	14.37%

Features	IPC (12n)			HSI (12n)			EU50 (24n)		
	Garson	Yoon	Trapezoid	Garson	Yoon	Trapezoid	Garson	Yoon	Trapezoid
A/D	3.35%	0.001	1.33%	7.54%	0.034	5.71%	7.87%	-0.076	6.38%
CCI	6.84%	0.001	2.95%	8.24%	0.118	8.44%	9.88%	0.162	9.77%
Disp10	7.49%	0.018	2.99%	7.98%	0.107	9.58%	7.86%	0.033	8.41%
Disp5	7.13%	-0.003	2.86%	8.39%	0.124	9.48%	7.86%	0.087	8.05%
fastD	10.21%	-0.021	8.45%	7.05%	0.003	7.17%	7.71%	-0.052	7.09%
fastK	8.79%	-0.098	7.90%	9.59%	0.238	15.42%	9.22%	0.110	13.09%
Moment	7.51%	-0.003	1.42%	7.31%	0.081	6.61%	7.07%	0.027	7.27%
OSCP	10.09%	-0.022	3.23%	8.05%	0.056	7.65%	8.43%	0.078	8.23%
ROC	13.29%	0.717	49.24%	8.37%	0.065	7.44%	9.10%	0.139	9.73%
RSI	9.15%	0.003	2.27%	8.95%	0.130	9.35%	7.75%	0.099	7.36%
slowD	6.41%	0.004	4.47%	6.21%	0.017	6.23%	8.43%	-0.046	8.37%
WilliamsR	9.75%	-0.109	12.90%	12.33%	-0.028	6.91%	8.82%	0.090	6.25%

One of the problems of the above measures is consistency. Both measures are positively correlated but just. For example, table 3 shows the correlation coefficient between the Garson Measure and the Yoon Measure. Both measures are highly correlated when analysing the Nikkei Index, but they are completely different in the Hang Seng index with a correlation of just 0.05. Another drawback of the Garson and Yoon measures is that they become difficult to calculate in more complex network architectures. Under such considerations, a different measure is needed to evaluate the contribution of each feature in a ANN model.

We decided to give a geometric interpretation to the weights in order to establish their relevance. For example, in a one-hidden layer neural network, we interpret the weights w_{ji} and v_{jk} as the lengths of the

opposite sides of a triangle. Multiplying the network weights in this form we can interpret the entire measure as the area of several triangles that make up to a irregular trapezoid:

$$TC = \sum_{j=1}^{J} \frac{|w_{ji}||v_{jk}|}{2}$$

An appealing feature of this Trapezoid Contribution measure is that can be applied to any number of hidden layers and neurons in the network and is quite easy to calculate and interpret if we make percentages with the whole area and its parts. Table 4 show the relative importance of each feature from the ANN analysis using Garson, Yoon and the Trapezoid measures. For Japan, Canada, Mexico and German indexes the ROC is the most influential feature to predict the stock market index while the fastK is the most important in the Hong Kong and European50 indexes.

We may notice that the new Trapezoid contribution measure is highly correlated with the Yoon measure but also moderately correlated with the Garson measure. Most importantly, it is easy to calculate and can be applied to more complex network architectures.

5 Final Conclusions

This work contains a financial forecasting using both traditional backpropagation and Resilient Backpropagation Neural Networks and also an analysis on the relative importance of features used for forecasting. We use standard single layer and multi-layer feed forward architectures to evaluate the performance of both algorithms, along with sigmoid activation function and error-correction learning rule, which are common for time series forecasting. The use of the RBP algorithm provides a practical solution to the determination of the learning rate and is especially helpful for data sets with noise such as financial stock indexes. The Resilient backpropagation with weight backtracking is a very flexible algorithm that can adjust to changes in model complexity. Some times it can find a better solution when the model specification changes.

This work provides a simple contribution measure in order to evaluate the importance of features in financial times series forecasting. The main reason comes from the lack of consistency in two available indexes: the Garson and the Yoon contribution measures. A simple measure using the concept of an area of a trapezoid captures de idea of contribution to the prediction using the ANN weights. This Trapezoid contribution measure uses the ANN weights from the best model (highest hit ratio from a single layer ANN) to calculate an area of an irregular trapezoid for every feature variable. Although this concept is simple it reflects the magnitude and influence of each weight in the network and can be interpreted as contribution to the forecasting.

We used the trapezoid contribution measure along with the Garson and the Yoon measures to analyse the relevance of each feature in the best ANN model for each of the six stock exchange indexes. We concluded that the ROC is perhaps a very relevant feature at least for four of the stock exchange indexes used: IPC, TSE, DAX and Nikkei. The European50 index and the Hang Seng index seem to respond more to the FastK indicator despite the Garson and Yoon contribution measures are not consistently showing this. In this respect, the trapezoid contribution measure offers additional relevant information that can be used to evaluate the contribution of each feature in the network.

References

[1] Achelis, S. B. (2001). *Technical Analysis from A to Z*. McGraw Hill New York.

[2] Anastasiadis, A. D., Magoulas, G. D., and Vrahatis, M. N. (2005). New globally convergent training scheme based on the resilient propagation algorithm. *Neurocomputing*, 64:253–270.

[3] Cao, L.-J. and Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks*, 14(6):1506–1518.

[4] Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

[5] Günther, F. and Fritsch, S. (2010). neuralnet: Training of neural networks. *The R journal*, 2(1):30–38.

[6] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[7] Huang, W., Nakamori, Y., and Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers and Operations Research*, 32(10):2513–2522.

[8] Huang, Z., Chen, H., Hsu, C.-J., Chen, W.-H., and Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision support systems*, 37(4):543–558.

[9] Igel, C. and Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123.

[10] Kara, Y., Boyacioglu, M. A., and Baykan, O. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319.

[11] Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319.

[12] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[13] Riedmiller, M. and Braun, H. (1992). Rprop-a fast adaptive learning algorithm. In *Proc. of ISCIS VII), Universitat*. Citeseer.

[14] Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

[15] Rosenblatt, F. (1962). Principles of neurodynamics.

[16] Sapankevych, N. I. and Sankar, R. (2009). Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, 4(2).

[17] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Table 5: Standard Statistics by Stock Market

High price									
Index	N	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
DAX	4,942	7,346.74	2,795.28	6,801.95	2,319.65	13,596.89	11,277.24	0.49	-0.78
Nikkei	4,776	14,120.67	4,285.77	13,636.81	7,100.77	24,448.07	17,347.30	0.41	-0.98
IPC	4,876	28,553.79	15,434.73	31,543.24	5,109.40	51,772.37	46,662.97	-0.24	-1.45
Hang Seng	4,800	19,577.61	5,696.33	20,623.56	8,430.62	33,484.08	25,053.46	-0.06	-0.86
EU50	4,850	3,241.86	717.30	3,111.17	1,809.98	5,464.43	3,654.45	0.86	0.52
TSE	4,917	11,867.28	2,843.10	12,294.60	5,812.90	16,672.70	10,859.80	-0.31	-1.04
Low Price									
Index	N	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
DAX	4,942	7,233.77	2,780.47	6,691.01	2,188.75	13,517.81	11,329.06	0.49	-0.77
Nikkei	4,776	13,935.29	4,257.91	13,403.06	6,994.90	24,217.26	17,222.36	0.42	-0.96
IPC	4,876	28,168.76	15,280.96	31,087.91	4,950.71	51,524.23	46,573.52	-0.23	-1.46
Hang Seng	4,800	19,316.24	5,639.54	20,386.76	8,331.87	32,897.04	24,565.17	-0.06	-0.88
EU50	4,850	3,241.86	717.30	3,111.17	1,809.98	5,464.43	3,654.45	0.86	0.52
TSE	4,917	11,738.87	2,834.99	12,151.10	5,678.30	16,589.80	10,911.50	-0.29	-1.05
Open Price									
Index	N	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
DAX	4,942	7,293.06	2,788.12	6,746.28	2,203.97	13,577.14	11,373.17	0.49	-0.77
Nikkei	4,776	14,032.76	4,273.57	13,553.15	7,059.77	24,376.17	17,316.40	0.42	-0.97
IPC	4,876	28,362.18	15,362.55	31,307.40	5,077.39	51,590.48	46,513.09	-0.23	-1.46
Hang Seng	4,800	19,460.25	5,672.78	20,518.17	8,351.59	33,335.48	24,983.89	-0.06	-0.87
EU50	4,850	3,241.86	717.30	3,111.17	1,809.98	5,464.43	3,654.45	0.86	0.52
TSE	4,917	11,807.88	2,839.82	12,219.80	5,689.40	16,642.10	10,952.70	-0.30	-1.04
Close Price									
Index	N	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
DAX	4,942	7,292.11	2,787.72	6,748.30	2,202.96	13,559.60	11,356.64	0.49	-0.77
Nikkei	4,776	14,027.96	4,273.60	13,541.62	7,054.98	24,270.62	17,215.64	0.42	-0.97
IPC	4,876	28,368.40	15,360.10	31,321.52	5,081.92	51,713.38	46,631.46	-0.23	-1.46
Hang Seng	4,800	19,450.48	5,665.96	20,511.59	8,409.01	33,154.12	24,745.11	-0.06	-0.87
EU50	4,850	3,241.86	717.30	3,111.17	1,809.98	5,464.43	3,654.45	0.86	0.52
TSE	4,917	11,805.41	2,838.86	12,220.20	5,695.30	16,669.40	10,974.10	-0.30	-1.04

Table 6: Summary Statistics Features

TSE (Canada)	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.54	0.50	1.00	0.00	1.00	1.00	-0.14	-1.98
AD	13175.28	4966.81	12835.92	5126.09	21788.43	16662.33	0.22	-1.33
CCI	19.59	109.43	37.32	-330.81	327.16	657.97	-0.43	-0.51
FastK	58.82	30.98	64.24	0.00	100.00	100.00	-0.35	-1.19
FastD	58.82	28.77	63.90	0.25	100.00	99.75	-0.34	-1.23
SlowD	58.82	27.88	63.72	1.14	98.42	97.28	-0.34	-1.22
Williams R	41.18	30.98	35.76	0.00	100.00	100.00	0.35	-1.19
Disp10	100.07	1.71	100.26	85.94	108.36	22.42	-1.15	6.55
Disp5	100.03	1.15	100.14	90.03	106.43	16.40	-0.82	6.09
Moment	6.49	229.82	23.60	-1884.90	1225.50	3110.40	-0.83	4.29
OSCP	0.00	0.01	0.00	-0.08	0.04	0.12	-1.23	6.44
ROC	0.01	1.07	0.06	-9.79	9.37	19.16	-0.66	10.07
RSI	53.14	12.05	53.83	12.78	84.00	71.23	-0.27	-0.31
IPC (Mexico)	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.53	0.50	1.00	0.00	1.00	1.00	-0.12	-1.99
AD	28733.10	19597.32	31712.29	973.64	59315.71	58342.07	-0.13	-1.54
CCI	20.33	110.74	42.03	-357.18	371.29	728.47	-0.38	-0.51
FastK	57.88	30.91	63.36	0.00	100.00	100.00	-0.34	-1.19
FastD	57.90	28.86	63.21	0.51	99.88	99.37	-0.34	-1.25
SlowD	57.92	27.99	63.04	1.91	99.57	97.66	-0.34	-1.25
Williams R	42.12	30.91	36.64	0.00	100.00	100.00	0.34	-1.19
Disp10	100.18	2.20	100.29	84.56	112.72	28.17	-0.57	4.27
Disp5	100.08	1.47	100.14	89.78	109.98	20.20	-0.33	4.76
Moment	29.31	665.07	51.18	-4496.07	3554.29	8050.36	-0.39	3.65
OSCP	0.00	0.01	0.00	-0.08	0.07	0.15	-0.72	4.45
ROC	0.04	1.29	0.07	-8.27	10.44	18.71	0.00	5.38
RSI	53.53	12.56	54.50	11.49	86.44	74.94	-0.20	-0.51
Nikkei (Japan)	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.51	0.50	1.00	0.00	1.00	1.00	-0.05	-2.00
AD	-33210.80	4452.53	-34423.82	-40305.15	-20749.27	19555.87	0.66	-0.51
CCI	8.83	109.64	19.54	-430.68	321.73	752.40	-0.24	-0.67
FastK	55.24	32.47	58.55	0.00	100.00	100.00	-0.20	-1.37
FastD	55.25	30.28	58.34	0.22	100.00	99.78	-0.19	-1.41
SlowD	55.26	29.40	58.24	0.66	98.49	97.83	-0.18	-1.40
Williams R	44.76	32.47	41.45	0.00	100.00	100.00	0.20	-1.37
Disp10	100.03	2.40	100.21	79.79	113.32	33.53	-0.72	4.09
Disp5	100.01	1.61	100.15	86.81	113.79	26.98	-0.65	5.95
Moment	2.43	381.10	25.95	-2415.93	1671.34	4087.27	-0.59	2.75
OSCP	0.00	0.01	0.00	-0.10	0.07	0.16	-0.76	3.69
ROC	0.00	1.52	0.03	-12.11	13.23	25.35	-0.40	6.28
RSI	51.83	12.18	51.70	13.54	92.94	79.41	0.10	-0.30

Table 7: Summary Statistics Features (continue)

Europe50	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.51	0.50	1.00	0.00	1.00	1.00	-0.03	-2.00
AD	2140.52	717.30	2009.83	708.64	4363.09	3654.45	0.86	0.52
CCI	8.84	108.23	23.32	-366.03	367.35	733.38	-0.31	-0.60
FastK	55.99	36.96	61.44	0.00	100.00	100.00	-0.25	-1.44
FastD	56.00	34.09	61.01	0.00	100.00	100.00	-0.24	-1.43
SlowD	56.02	32.95	60.76	0.00	100.00	100.00	-0.23	-1.43
Williams R	44.01	36.96	38.56	0.00	100.00	100.00	0.25	-1.44
Disp10	99.98	2.23	100.23	84.30	110.57	26.27	-0.76	3.08
Disp5	99.99	1.53	100.10	89.74	107.94	18.20	-0.44	2.97
Moment	-1.23	85.01	5.83	-487.77	435.31	923.08	-0.46	2.49
OSCP	0.00	0.01	0.00	-0.07	0.05	0.12	-0.75	2.97
ROC	-0.01	1.46	0.02	-9.01	10.44	19.45	-0.06	4.71
RSI	51.48	11.04	52.21	14.14	77.81	63.67	-0.22	-0.54
Han Seng (HK)	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.52	0.50	1.00	0.00	1.00	1.00	-0.06	-2.00
AD	21143.14	6147.85	23233.24	8928.13	34672.11	25743.98	-0.41	-1.10
CCI	10.54	109.03	19.31	-338.04	300.85	638.89	-0.17	-0.86
FastK	54.97	32.60	58.64	0.00	100.00	100.00	-0.19	-1.42
FastD	54.97	30.54	58.01	1.33	99.59	98.26	-0.17	-1.45
SlowD	54.97	29.66	57.92	2.98	98.79	95.81	-0.17	-1.44
Williams R	45.03	32.60	41.36	0.00	100.00	100.00	0.19	-1.42
Disp10	100.06	2.39	100.23	76.16	110.29	34.13	-0.60	4.57
Disp5	100.03	1.59	100.10	82.57	113.26	30.68	-0.47	7.04
Moment	9.65	553.11	33.43	-4025.33	2952.83	6978.16	-0.35	3.34
OSCP	0.00	0.01	0.00	-0.09	0.06	0.16	-0.54	3.23
ROC	0.01	1.47	0.05	-13.58	13.41	26.99	-0.10	8.05
RSI	52.11	12.58	52.33	15.05	89.41	74.36	-0.04	-0.51
Dax (Germany)	Mean	SD	Median	Min	Max	Range	Skew	Kurtosis
Label	0.53	0.50	1.00	0.00	1.00	1.00	-0.12	-1.99
AD	6225.18	4595.94	5056.53	-825.28	15989.25	16814.53	0.66	-0.79
CCI	15.34	109.28	34.90	-303.85	346.90	650.75	-0.34	-0.69
FastK	57.98	31.51	63.09	0.00	100.00	100.00	-0.31	-1.29
FastD	57.98	29.34	62.85	0.45	99.65	99.21	-0.29	-1.33
SlowD	57.98	28.45	62.83	2.20	99.40	97.20	-0.28	-1.32
Williams R	42.02	31.51	36.91	0.00	100.00	100.00	0.31	-1.29
Disp10	100.07	2.34	100.35	84.06	111.92	27.86	-0.79	3.52
Disp5	100.03	1.57	100.15	90.31	108.14	17.82	-0.50	3.18
Moment	4.40	187.76	16.42	-1267.49	807.60	2075.09	-0.49	2.14
OSCP	0.00	0.01	0.00	-0.08	0.06	0.14	-0.87	3.57
ROC	0.01	1.47	0.08	-8.87	10.80	19.67	-0.06	4.59
RSI	52.78	11.90	53.36	11.24	84.64	73.40	-0.17	-0.41