Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices

Ming-Chien Sung [1]*, Larry Olanrewaju Orimoloye [1], Tiejun Ma[1], Johnnie E.V. Johnson[2]

Abstract

Many existing learning algorithms suffer from limited architectural depth and the locality of estimators, making it difficult to generalize from the test set and providing inefficient and biased estimators. Deep architectures have been shown to appropriately learn correlation structures in time series data. This paper compares the effectiveness of a deep feedforward Neural Network (DNN) and shallow architectures (e.g., Support Vector Machine (SVM) and one-layer NN) when predicting a broad cross-section of stock price indices in both developed and emerging markets. An extensive evaluation is undertaken, using daily, hourly, minute and tick level data related to thirty-four financial indices from 32 countries across six years. Our evaluation results show a considerable advantage from training deep (cf. shallow) architectures, using a rectifier linear (RELU) activation function, across all thirty-four markets when 'minute' data is used. However, the predictive performance of DNN was not significantly better than that of shallower architectures when using tick level data. This result suggests that when training a DNN algorithm, the predictive accuracy peaks, regardless of training size. We also examine which activation function works best for stock price index data. Our results demonstrate that the RELU activation function performs better than TANH across all markets and time horizons when using DNN to predict stock price indices.

Keywords: Financial time series forecasting; deep feedforward Neural Network; market efficiency; machine learning.

[1] Centre for Risk Research, Southampton Business School, University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom, email: {ms9; oo1u16; tiejun.ma} @soton.ac.uk.
*Corresponding author.
[2] Nottingham Business School, Burton St., Nottingham, NG1 4BU, United Kingdom, email: johnnie.johnson@ntu.ac.uk

1 Introduction

Accurately predicting changes in stock price indices is potentially profitable but is a difficult challenge due to the high degree of uncertainty involved. Proponents of the efficient market hypothesis (EMH) argue that it is impossible to generate abnormal returns through 'more informed' investment decisions (Titan, 2015). However, the accelerated use of advanced algorithms has led to the identification of opportunities to trade profitably on model predictions (Chang, et al., 2009; de Oliveira, et al., 2013; Huang, et al., 2008; Schumaker & Chen, 2009). Many techniques have been developed which can aid this prediction task. For example, machine learning (ML) techniques have been employed due to their ability to handle non-linear data (Krauss et al. (2016). The two most widely used ML algorithms for predicting stock price indices are ANN and SVM (See Zhang and Wu., 2009; Chang, et al., 2009; Mittermayer, 2004; Huang, et al., 2005; Hsu et al. (2016).

It has been suggested that learning multiple levels of distributed representations offers considerable advantages over shallow architectures (Bengio et al., 2009) and empirical work suggests that it is difficult to train shallow architectures to be as accurate as DNN. For vision tasks, a study on deep convolutional nets suggested that deeper models are preferred under a parameter budget (Mathieu et al., 2013). Furthermore, Lecun et al., 2013 trained shallow nets on scale invariant feature transform (SIFT) to classify a large-scale ImageNet dataset and found that it was difficult to train large, high-accuracy, shallow architectures. In addition, Seide et al., (2011) show that deeper models are more accurate than shallow models in speech acoustic modelling and Abdel-Hamid et al., (2012) recorded a 5% increase in predictive accuracy in speech recognition using a deep (vs. shallow) architecture using one million labelled points on test data. Ba et. al, (2014) provide some pointers into why deep feedforward NN might perform better than shallow architectures. He noted that a network with a large enough single hidden layer of sigmoid units can approximate any decision boundary.

Consequently, in attempting to predict changes in stock price indices across thirty-four markets using different time horizons, we deploy several powerful methods inspired by the latest trends in machine learning. First, we compared the effectiveness of DNN and shallow architectures (e.g., Support Vector Machies (SVM) and one-layer NN).

In developing a deep feedforward NN one needs to select the most appropriate activation function for the data. Much of the theoretical literature suggests that RELU (Bengio et al., 2007) greatly accelerates the convergence of stochastic gradient descent (SGD) compared with other functions (Krizhevsky et al., 2012). Unfortunately, RELU units can be fragile during training and can 'die' or vanish (He et al., 2016) when using the backpropagation algorithm. Jarett et al. (2009), working with the Caltech-101 dataset (a dataset of digital images available online for public use for a computer vision task) found that the nonlinear TANH activation function worked particularly well with their type of contrast normalization, followed by local average pooling. It is likely that different data will respond particularly well to the use of either the RELU or TANH activation function because RELU zeros out

negative values while TANH does not. Consequently, an important contribution of this paper is that we examine which activation function works best for stock price index data.

With large positive returns and a low correlation with returns in developed financial markets, emerging financial markets provide theoretically an ideal environment for international portfolio diversification. Consequently, investors seeking high returns and good diversification often turn to these markets (Berger, Pukthuanthong and Yang, 2011). A shallow algorithm, such as a one layer-NN and other non-linear algorithms, might not be able to fully account for the complex relationships in modelling price changes in these markets because they have no level of abstraction. It is therefore difficult to capture complexities inherent in the data structure. We examined whether DNN could better capture these dynamics, thereby providing the basis of a more accurate prediction model.

Most literature exploring the degree to which is possible to predict stock price indices have focussed on one specific time horizon across one or two markets (see the survey in Hsu et al., 2016). This prevents firm conclusions being reached concerning the most appropriate methods for predicting a wide variety of stock indices (which may operate under different market conditions) across different time horizons. Consequently, to fill this important research gap, our aim is to compare the accuracy of deep and shallow architectures for predicting thirty-four different stock indices across different time horizons (daily, hourly, minute and tick level). Our aim here, is not to optimize the performance of a DNN model in any given market. Rather, the analysis provides a means of comparing, across a wide range of different market and prediction environments, the performance of deep architectures with shallower architectures which previous literature suggests are particularly effective for predicting stock price indices.

The remainder of this paper is organized as follows:er In section 2, we review related work in the financial and ML literatures. In section 3, we develop three hypotheses, based on the literature review, concerning how different methodological factors affect predictive accuracy. In section 4, we outline our experimental design. In section 5, we present our empirical results and in section 6 we discuss these in the light of existing literature. We conclude in section 7 with a summary of the main findings, the implications the work and suggested areas for further research.

2 Literature review

Increasingly, the DNN approach is seen as an important ML tool. DNN have had important empirical successes in a number of traditional AI applications, such as computer vision and natural language processing (Bengio, 2009; Bengio et al., 2013). As a result, DNN are attracting much attention from the academic and industrial communities. Companies like Google, Microsoft, Apple, IBM and Baidu are investing in DNN, with the first widely distributed products being used by consumers aimed at speech recognition. DNN is also used for object recognition (Google Goggles), image and music information retrieval (Google Image Search, Google Music), as well as computational advertising (Le et al., 2012).

Furthermore, a DNN building block (the restricted Boltzmann machine, or RBM) was used as a crucial part of the winning entry in a million-dollar ML competition (the Netflix competition) (Salakhutdinov et al., 2007; Toscher et al., 2009).

There have been several studies demonstrating the effectiveness of DNN methods in a variety of application domains such as image and speech processing (Huang & LeCun, 2006) but it has rarely been employed for forecasting problems. DNN have achieved very low error rates on various image databases (the MNIST database of handwritten digits, 2014) (Fukushima, 2003) and in this domain it has been reported that the learning process of DNN methods are 'surprisingly fast' (Hinton, Osindero, & Teh, 2006). DNN have also been shown to be more effective than conventional algorithms when employed for electroencephalographic signal processing (Ren & Wu, 2014). Much of this work was based on a paper by Deng et. al, (2013), which reported a major breakthrough in DNN speech recognition. Given the success of DNN in both image and speech processing, it is surprising that DNN has, as outlined below, rarely been used to predict stock price indices and its performance across a range of markets and forecast time horizons has not been explored.

Krauss et al. (2016) were the first to compare DNN, gradient-boosted trees and random forest with several ensemble models in the context of statistical arbitrage. Each model was trained on lagged returns of all stocks in the S&P 500 financial market, after elimination of survivor bias. They used sliding window methods to forecast one-day-ahead trading signals. Using out of sample data, they found that the ensemble algorithm was able to generate a 0.45 percent return on a daily basis. However, when they compared the predictive accuracy of the three algorithms, random forest outperformed DNN and gradient-boosted trees. They noted that careful hyper-parameter tuning of DNN could have improved its performance. Fischer et. al. (2017) applied a subset of Recurrent Neural Networks (i.e. Long Short-term Memory (LSTM) networks) to financial time series predictions using a single stock market index. They found that LSTM networks outperformed memory-free classification methods, i.e., random forest (RAF) and a logistic regression classifier (LOG). In addition, Eunsuk et. al. (2017) constructed a DNN using stocks returns from the KOSPI market (the major stock market in South Korea) to examine the effect of three unsupervised feature extraction methods (principal component analysis, autoencoder, and the restricted Boltzmann machine) on the network's overall ability to predict future market behaviour. They found that DNN could extract additional information from the residuals of the autoregressive model to improve prediction performance.

A review undertaken by Hsu et al. (2016) indicated that shallow architectures, such as SVM and one-layer NN, are currently the main methods employed for predicting movements in stock price indices. As indicated above, only a handful of subsequent studies have employed deep architectures. This may be linked to the fact that DNN in general can better handle different classes of structured and unstructured datasets (used for images, text and time series data: see LeCun et al., 2015). In addition, support vector machines (SVM) and/or one-layer NN may be the most widely used methods for

financial market forecasting, because of their ability to recognize patterns in nonlinear, dynamic time series data (Chang et al., 2009; Lee, 2009; Zbikowski, 2015).

Hsu et al (2016)'s extensive survey covered 28 markets, of which the US accounted for 12 and Taiwan 7. The most popular prediction method in these papers was ANN and only four studies considered both ANN and SVM. One of the interesting findings of the Hsu et al (2016) survey was that very few studies evaluated prediction models in a dynamic fashion i.e. sliding window approach. Rather, the prevailing approach, used in 25 out of 28 previous studies, is to split a financial time series into a training and a hold-out test set. We refer to this approach as 'static' because it uses the same prediction model throughout the whole testing period, without updating.

To increase predictive accuracy using SVM and/or one-layer NN, some authors have suggested the use of more dimensions to augment existing stock price data (Pan et. al., 2005, Kara et al., 2011). Pan et. al., (2005) present a computational approach for predicting the Australian stock market index (AORD), using multi-layer feed-forward neural networks from the AORD time series data and various interrelated markets. They noted that effectively selecting the data that is fed into a NN can improve the predictive accuracy. They showed that by including additional dimensions to augment existing datasets, they could produce an 80% prediction accuracy. Others have proposed a multi-stage approach to financial market forecasting, by first selecting an optimal feature extraction before developing ensemble models (Huang et al., 2008).

Our review of the stock price index prediction literature has highlighted four key gaps. First, that the effectiveness of DNN ML methods for predicting stock price indices has not been compared to the predictive performance of existing methods which employ a shallower architecture across several major financial markets. Second, a holistic performance evaluation of two of the mostly used activation functions for predicting stock price indices (RELU and TANH) has not been undertaken. Third, a comparison of the accuracy of a range of ML methods when predicting stock prices indices across a variety of time horizons (i.e. daily, hourly, minute and tick level) has not been undertaken. This is the case, despite the fact that there is some evidence that in other domains DNN performs best when datasets are big. However, there may be some limits beyond which the predictive performance does not improve. Fourth, no comprehensive comparative study of the predictive accuracy of DNN and alternative ML methods for predicting stock price indices in developed vs. emerging markets, has been conducted. DNN have showed some promising results using tick data in some emerging (cf. developing) markets, but SVM has been shown to produce more accurate predictions in other emerging markets (Alexandre, Pedro & Sabino, Sarah & Albuquerque, Pedro, 2015). We aim to help fill these important gaps in the literature.

3 **Hypothesis**

Previous stock price index prediction studies using ML, have predominantly employed daily time horizon data (Hsu et al., 2016). This could be attributed to fact that these data are free whereas this is not the case for intraday data. For example, Cenesizoglu T. and Timmermann A. (2008) found that stock returns prediction models based on daily or longer data intervals generally underperform models predicting over a shorter time interval.

Bearing this in mind, DNN with its superior representation space compared to shallow models, should out-perform other algorithms when predicting stock indices using intraday data. This is likely to arise because DNN may effectively employ the significantly larger and more complex dataset to create greater accuracy (Abdel-Hamid et al., 2012). Cho et al. (2016) conducted experimental studies on deep learning systems to explore how much data is needed to train a medical image to achieve high accuracy with a low standard deviation. They found that an increase in the training data from 5 to 50 training images significantly improved accuracy but this did not improve significantly from 100 to 200. In particular, they found that beyond a certain training set size the accuracy hardly improved. In addition, Angelidis et al., (2008) compared the performance of various models using intra-day vs. inter-day. They investigated their forecasting performance for three European equity indices and found that intra-day models clearly produced the most accurate variance forecasts.

To explore the view that DNN are more accurate than shallow models when employing larger datasets, we test the following hypothesis:

H1: Predictive accuracy is higher using DNN for intraday (i.e. minute/tick) time horizons than SVM and one-layer NN.

DNN requires activation functions to map the output of a node given an input or set of inputs. The use of Rectified Linear Unit (RELU) has become very popular in the last few years. It computes the function $f(x)= max(0, x)$. In other words, the activation has a threshold of zero (i.e. takes only positive numbers). Krizhevsky et al. (2012) found that employing RELU greatly accelerated the convergence of stochastic gradient descent (cf. when employing TANH functions). They argued that this is due to its linear, non-saturating form. Compared to TANH neurons that involve expensive operations (exponentials etc.), the RELU can be implemented by simply thresholding a matrix of activations at zero. Nair et al. 2010 used CIFAR- 10 data (consisting of 60000 32x32 colour images in 10 classes, with 6000 images per class) to build a four-layer convolutional network and concluded that RELU was several times faster and produced more accurate predictions than TANH. However, it is likely that different datasets will possess different features that will lead to either RELU or TANH producing more accurate predictions. To our best knowledge, a comparison of the predictive accuracy of the two most widely used activation functions (RELU or TANH) for stock price indices, has not been undertaken. Consequently, based on the findings from the other domains discussed above, we test our second hypothesis:

H2: Using the RELU activation function produces more accurate stock price index predictions than using the TANH activation function across all time horizons.

The financial economics literature suggests that informational efficiency differs between established and emerging financial markets (Griffin et al., 2010; Karemera & Ojah, 1999). In particular, there is some evidence that the EMH, which posits that asset prices reflect all relevant information (Fama, 1970 & 1991), may not hold in emerging markets because of extraneous factors (financial, economic, political, social and cultural environment). These factors can lead emerging markets to suffer greater volatility and to experience more sudden stock market declines than developed markets.

It has been found that using a Gaussian distribution to model price changes or stock market returns in emerging markets is not appropriate because the prices and returns are fat-tailed (i.e. large price changes can be expected relatively often) (see, for example, Harvey, 1995; Bekaert and Harvey, 1997). A shallow algorithm, such as a one layer-NN and other non-linear algorithms, might not be able to fully account for the complex relationships in modelling price changes in these markets because they have no level of abstraction and it is therefore difficult to capture complexities inherent in the data structure. Taken together, these factors motivate the following hypothesis:

H3: The accuracy of stock price index predictions using a DNN is greater in emerging markets than in developed markets and is greater than that arising from employing a model with a shallower architecture.

The proposed set of hypotheses enable us to examine the extent to which various factors, such as the forecasting horizon, activation functions, selected models (i.e. DNN vs. SVM vs one-layer NN), and data from emerging (vs. developed) markets influence the accuracy of financial time series forecasting. The testing of these hypotheses should be helpful in determining if DNN have a place in stock market predictions.

[Table 1 about here]

4 Experimental design

Table 1 summarizes the main factors we considered (forecast time horizon, activation functions and market classification) which could influence the predictability of stock price indices. Thirty-four financial time series from a wide cross section of the world's major markets were used to test our hypotheses. Our design is similar to that of Gerlein, et al.'s (2016) study in which we performed repeated experiments to clarify the influence of several factors on the prediction performance of simple ML classifiers. In addition, we conducted robustness checks of our results by testing different levels of parameters settings and obtained the best results for each ML algorithm. This provides confidence that reliance can be placed on the conclusions regarding the comparative effectiveness of the shallower architecture ML and deeper architecture DNN methods.

In the following subsections, we discuss the factors we examined and motivate our choices of individual factor levels.

4.1.1 Data, variables, and forecasting time horizon

The dataset employed in this study was obtained from TickWrite Data Inc., including time series of financial indices from around the world. We have focused on predicting national stock indices because these were used in the majority of previous ML studies that predict direction of price changes (e.g. Bodyanskiy & Popov, 2006; de Oliveira et al., 2013; Huang et al., 2008; Huang et al., 2005; Pan, Tilakaratne, & Yearwood, 2005; Qian & Rasheed, 2007). We included as many markets as possible to cover both developed and emerging markets, since one of the aims of the study is to compare the performance of DNN in emerging *vs*. developed markets. In addition, to ensure comparability, we wanted to use the same period for each market. This restricted the sample period because the availability of intraday data is limited and, for many markets, is only available from 2008 onwards. This required us to choose the data period where intraday data was available for most markets between 1 Feb 2008 - 19 Feb 2014 (6-year period with 1500 trading days) with an exception of Brazilian market for which we only have the data for 4 years between 1 Feb 2010 – 19 Feb 14. The raw data contains prices for each market at tick level. We were then able to transform/expand the tick-level data into minute, hourly and daily level data across multiple markets, to explore the sensitivity of DNN algorithms to various time horizons and to different degrees of financial market complexity.

4.1.2 Market classification

The term emerging markets was first used in the 1980s by Antoine van Agtmael, a World Bank economist. The classification of countries as emerging markets is carried out and reviewed on a regular basis by a range of international financial institutions. They all use different categories, methodologies and degrees of granularity. It is, therefore, not surprising that there is no agreed consensus on what constitutes an emerging market (Colm Keaney (2012)). The Financial Times Stock Exchange (FTSE), for example, categorizes emerging markets into 9 ‘advanced’ and 13 ‘secondary’ emerging markets, whereas Bloomberg's Morgan Stanley Capital International (MSCI) emerging market index comprises 26 countries, split into three regions. For this paper, we only count a market as emerging if it is classified by the FTSE and MSCI as emerging. In table 2 we list the markets which the data restrictions outlined in section 4.1.1 enabled us to include and we identify which of these are categorized as ‘emerging’ by the MSCI and the FTSE.

[Table 2 about here]

4.1.3. Forecasting methods

The most widely used ML methods discussed in the literature for financial time series forecasting are SVM and NN (Ballings, et al., 2015). One of the explanations for their popularity (cf. other advanced

techniques, such as bagged or boosted decision trees), is that they are better suited to handling continuous covariates (Hsu et al., 2016). These are the type of covariates which occur frequently in financial time series.

SVMs belong to the general category of kernel methods (Shawe -Taylor et. al., 2004; Scholkopf et al., 2002), that depend on the data only through dot-products. When this is the case, the dot product can be replaced by a kernel function, *n* which computes a dot product in some possibly high-dimensional feature space. This has two advantages: First, the ability to generate nonlinear decision boundaries using methods designed for linear classifiers. Second, the use of kernel functions allows the user to apply a classifier to data that has no obvious fixed-dimensional vector space representation (Ben-Hur et al., 2010). Modelling using SVM involves the construction of a hyperplane as the decision surface such that the margin of separation between positive and negative examples is maximized (Xu, Zhou, & Wang, 2009). For a training set of samples, with input vectors $x_i \epsilon R^d$ and corresponding labels $y_i \epsilon$ {+1 , -1}, SVM learns how to classify objects into two classes.

We face a two-class problem, with label +1(close stock price index increases) and -1(close stock price index decreases). We define x as a vector with components x_i (in our case openindex (opening price of index), closingindex (closing price of index), highestindex (highest value the index reaches in the period) and lowestindex (lowest value the index reaches in the period)) and x_i denote the i*th* vector in a dataset composed of *n* labelled examples (x_i, y_i) where y_i is the label associated with x_i. The objects x_i are called inputs. To define a linear classifier, we use the dot product between the vectors, also referred to as the inner product or scalar product. A linear classifier is based on a linear discriminant function of the form:

$$f(x) = w^T x + b \quad (1)$$

The vector w is known as the weight vector, and b is called the bias. Consider the case b=0. The set of points x_i such that $w^T=0$ are all points that are perpendicular to w and go through the origin - a line in two dimensions, a plane in three dimensions and more generally, a hyperplane. A hyperplane divides the space into two according to the sign of the discriminant function $f(x)$. The boundary region, classified as positive and negative, is called the decision boundary of the classifier. A classifier with linear decision boundary is called a linear classifier and if the decision boundary depends on the data in a non-linear way, the classifier is said to be non-linear (Ben-Hur et al., 2010).

The parameters of SVM include the type of kernel function, the degree of kernel function (d: for a polynomial kernel; γ for a radial basis kernel) and a regularization constant c. Tables 3a, 3b and 3c show the numbers of levels tested in different parameter settings for all the algorithms compared in this paper. To determine the parameters efficiently, we followed the approach used by Patel et al. (2015). Namely, we test five levels on d, ten levels of γ and 4 to 5 levels of c in the parameter setting experiments. These parameters and the levels which are tested, are summarized in Table 3a. For one stock index, these settings of parameters yield a total of 20 and 40 treatments for the SVMs employing

polynomial and radial basis kernel functions, respectively. We selected the best polynomial kernel SVM model and the best radial basis SVM model (using parameter combinations that resulted in the best average of the training and holdout prediction performances) for comparison with NN and DNN.

[Table 3a about here]

NN is a nonlinear prediction method and is one of the most popular algorithms for stock index prediction (Kara et. al., 2011). NNs are often referred to as feedforward NNs or multilayer perceptron (MLP). Inspired by the functioning of biological NNs, NNs are a dense network of inter-connected neurons which get activated based on inputs. A one-layer feed-forward NN is employed in our study. Inputs for the network are four simple variables (Openindex/Closingindex/Highestindex/Lowestindex) which are represented by four neurons in the input layer. We did not to use technical indicators since previous studies question their usefulness (Lesmond et al., 2004) and an extensive simulation by Hsu et al. (2016) found that technical indicators offer little advantage over simple covariates in terms of accuracy or ROI. We employ an output layer with a single neuron, with log sigmoid as the transfer function, resulting in a continuous value output between 0 and 1. A threshold of 0.5 is used to determine whether we predict an increase (> 0.5) or decrease (<0.5) in the stock price index. We conducted robustness checks on the use of 0.5 as a threshold, but found that using 0.5 produced the most accurate predictions. In addition, this approach is in line with that employed in the majority of previous studies (e.g. Bodyanskiy & Popov, 2006; de Oliveira et al., 2013; Huang et al., 2008; Huang et al., 2005; Pan, Tilakaratne, & Year- wood, 2005; Qian & Rasheed, 2007; Hsu et al.,2016). We used tan sigmoid as the transfer function for the hidden layer since it was the setting that produced the most accurate predictions. Gradient descent with momentum was used to adjust the weights; at each epoch, weights were adjusted so that a global minimum could be reached.

The NN model parameters include the number of hidden layer neurons (*n*), the value of the learning rate (*lr*), the momentum constant (*mc*) and the number of epochs (*ep*). To determine appropriate levels in an efficient manner, ten levels of *n*, nine levels of *m*c and ten levels of *ep* were tested in the parameter setting experiments. Initially, the value of *lr* was fixed to 0.1. The parameters and the levels which were tested are summarized in Table 3b. We selected the three best NN models for comparison with DNN and SVM by selecting the three parameter combinations that resulted in the highest average prediction accuracy in the training and holdout data sets. For these top performing models, the learning rate *lr* was varied in the interval of [0.1, 0.2].

[Table 3b about here]

A DNN consists of an input layer, two or more hidden layers, and an output layer. DNNs are made up of many different functions (Dixon et al., 2016). For example, we might have four functions $f^{(1)}$,$f^{(2)}$,$f^{(3)}$ and $f^{(4)}$ all connected in a chain, to form $f(x) \approx f^{(4)}$,$f^{(3)}$,$f^{(2)}$ $f^{(1)}(x))$. In this case, $f^{(1)}$ is called

the first layer, $f^{(2)}$ the second layer, and so on; the final layer of a feedforward network being referred to as the output layer. The output layer is either a classification or regression layer, to match the output space. The overall length of the network of the chain gives the depth of the model and deep learning arises from this terminology.

NNs rely on linear functions and these have limitations when training large and complex data (Bengio et al., 2007). The appeal of DNNs (vs. NNs), therefore, stems from their ability to learn non-linear relationships.

We consider the classic feedforward architecture as shown in (2), where a neuron in the previous layer l is fully connected with all neurons in the subsequent layer l+1 via directed edges, each representing a certain weight. Also, each non-output layer of the net has a bias unit, serving as an activation threshold for the neurons in the subsequent layer. As such, each neuron receives a weighted combination α of the n_l outputs of the neurons in the previous layer l as an input, as follows:

$$\alpha = \sum_{i=1}^{n_l} w_i x_i + b, \tag{2}$$

with w_i denoting the weight of the output x_i and b the bias. The weighted combination α is transformed via an activation function f, so that the output signal $f(\alpha)$ is relayed to the neurons in the subsequent layer. Following Krizhevsky et al. (2012), we used both rectified linear units (which computes the function $\alpha = \sum_{i=1}^{n_l} w_i x_i + b$) and TANH, which constricts a real-value into the range [-1, 1]).

For the entire network, let W be the collection $\cup_{l=1}^{L-1} W_l$, with W_l denoting the weight matrix that connects layers l and l+1 for a network of L layers. Analogously, let B be the collection $B = \cup_{l=1}^{L-1} b_l$, with c_l denoting the column vector of biases of layer l. The collections of W and B fully determine the output of the entire DNN as shown in (3). Learning takes place by adapting these weights in order to minimize the error using the training data. In particular, the objective is to minimize some loss function $L(W,B|j)$ for each training set j. We are trying to solve a classification problem (i.e. the stock price index falls or rises), which means we can use the following cross-entropy loss function:

$$L(W,B|j) = -\sum_{y \epsilon o} \left(\ln \left(o_y^{(j)} \right) t_y^{(j)} + \ln (1 - o_y^{(j)})(1 - t_y^{(j)}) \right) \tag{3}$$

where y represents the output units, o the output layers, $o_y^{(j)}$ the prediction and $t_y^{(j)}$ the actual output for example j. This loss function is minimized by stochastic gradient descent. The gradient of the loss function $L(W, B|j)$ is calculated via backpropagation. We implemented this using SAS Viya in-memory architecture, taking advantage of the cloud analytics service (CAS) for efficient and fast processing.

Since the input space is very simple (i.e. four variables), we used a straightforward architecture. The design of an NN is more of an art than a science and tuning parameters is often undertaken via computationally intensive hyper-parameter optimization routines and cross-validation. We tried multiple hidden layers (from 2 to 10). A popular rule is to set the number of neurons in the first hidden layer of a feedforward network so as to use as many neurons as there are inputs (Krizhevsky et al., 2012). We followed this approach. Our network is relatively small (4 inputs and 160 weight parameters) but deep learning allows for large-scale models with thousands of features and millions of parameters, offering significant potential for further studies with larger numbers of inputs and weight parameters. The number of levels tested in different parameter settings for DNN are summarised in table 3c.

[Table 3c about here]

4.2 Performance measurement

4.2.1 Predictive accuracy

We used accuracy to evaluate the performance of the proposed models. Computation of these evaluation measures requires us to estimate Precision and Recall. These are evaluated from True Positive (TP), False Positive (FP), True Negative (TN) and False. We define these parameters, as follows:

$$Precision_{positive} = \frac{TP}{TP+FP} \tag{4}$$

$$Precision_{negative} = \frac{TP}{TP+FP} \tag{5}$$

$$Recall_{positive} = \frac{TP}{TP+FN} \tag{6}$$

$$Recall_{negative} = \frac{TN}{TN+FP} \tag{7}$$

Precision is the weighted average of precision positive and negative, while Recall is the weighted average of recall positive and negative. Accuracy is estimated using the following:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{8}$$

4.2.2 Model evaluation

Lo (1991) and Sadique et al. (2001) conducted extensive experiments using various time horizons in different markets and concluded that stock market returns exhibit long-term memory. This suggests that a ‘static’ approach for predicting stock price indices (i.e. the same prediction model is used throughout

the whole testing period, without updating) is the one most likely to yield higher predictive accuracy. This was confirmed by Hsu et al.'s (2016) study, which compared the static approach and a dynamic approach involving a sliding-window cross-validation performed in model training and evaluated multiple times using smaller chunks of sequentially ordered data (suggested by Lessmann et al., 2011). Hsu et al. (2016) demonstrated that the static outperformed the dynamic approach. Consequently, the static approach is employed in this study. To test the accuracy of the forecasting models, we used the standard static approach, whereby we split a financial time series chronologically into three non-overlapping sets: 50, 25 and 25 percent for training, validation and testing, respectively. The training data is used to estimate the parameters of the forecasting model and the validation set is used to tune meta-parameters (e.g., the regularization parameter, miniBatchsize, learning rate etc.) by means of empirical experimentation. For example, using daily data, of the 1500 trading days for each of the 34 financial time series in our data, we use the first 50% (750) of trading days to train the forecasting models with alternative meta-parameter settings (see Tables 3a,b,c) and assess their accuracy on the following 375 trading days (our validation sample). The validation sample predictions reveal the best meta-parameter setting for a given time-series. The fully-specified models with fixed optimal meta-parameters are trained and validated on the first 75% (1125) of trading days in the dataset. These models are then used to generate predictions in the test set (the remaining 375 trading days). To measure forecast accuracy, the predictions are compared to the actual values of the target variable in the test set. This allows us to compare models (e.g., SVM vs. NN vs DNN) in terms of their predictive accuracy on hold-out data. We call this approach a static evaluation, because the same model is used to forecast all observations in the test set (de Oliveira et al., 2013; Kara et al., 2011).

4.2.3 ANOVA (Analysis of variance) for experimental design

We used ANOVA to test for differences among the mean levels of model accuracy across four time horizons. To qualify these differences, we used Tukey's Standardized Range (HSD) test. HSD is a single-step multiple comparison procedure and statistical test. It can be used on raw data or in conjunction with an ANOVA (post-hoc analysis) to find means that are significantly different from each other. If there are two treatment levels, this analysis is equivalent to a t-test comparing two group means. The assumptions of analysis of variance (Steel and Torrie 1980) are that treatment effects are additive and experimental errors are independently random with a normal distribution that has mean zero and constant variance.

To test if our data is consistent with the null hypothesis, we calculate the F-test statistic (using the ANOVA), f_0, using the ratio of the variability between groups (S_B^2) to the variability within groups (S_W^2). The F-test statistic has an F-distribution with $df_1 = k-1$ and $df_2 = n_{tot}-k$, where k is the number of groups and n_{tot} is the total number of observations.

5. Results

5.1. Factors influencing the accuracy of forecasts of stock price indices.

The experimental design includes three main factors: forecast time horizon, activation functions and market classification. We obtained 544 individual prediction results: across 34 financial markets x 4 forecast time horizons x 4 model specifications (SVM, NN, DNN (with both the RELU and TANH activation functions; indicated, hereafter, as DNN (RELU) and DNN (TANH), respectively). The results enable us to compare the forecasting accuracy across different time intervals.

The null hypothesis is that the means of all groups are identical. Figure 1 displays the distribution of model accuracy across all time horizons among different model specifications. The F-test for the daily data for example suggests that there are differences among the accuracy of model performance i.e. the variation of model accuracy among the four models (numerator) is much larger than the variation accuracy within the four models ($F = 19.04$, $p<0.0001$). However, this does not show if they are significantly different from each other. To examine this, we perform a Tukey's Standardized Range (HSD) test. The HSD table is usually denoted by letters to represent if the mean value are significantly different or not. If the same letter is shown, this denotes that the mean is not significantly different while a different letter shows that there is a significant difference in some of the mean values.

A number of conclusions emerge from the results presented in Table 4. For daily data of 1,500 observations, SVM performs significantly better than all the other algorithms (HSD grouping is 'A'). The second best model is NN, which is not significantly better than DNN (RELU) since both HSD groupings are 'B', but is significantly better than DNN (TANH) with an HSD grouping of 'C'. Figure 2 shows a box plot of prediction accuracy for all four model specifications. These results are not surprising, because the daily data sample size is relatively small and it has generally been found that DNN only performs better than shallow architectures when using large datasets (Abdel-Hamid et al., 2012).

We increased the sample size by using hourly data (15,000 observations). Figure 3 shows a box plot of prediction accuracy for all four model specifications. Using this data, the F-test produces an insignificant result ($F = 2.49$, $p>0.0635$), suggesting that there are no differences in the predictive accuracy of the different model specifications. Table 5 shows HSD groupings of all the same letter 'A,' which confirms our findings of no differences in accuracy between the model specifications. Once again, these results are not surprising, since most studies only point to DNN producing superior results when far larger datasets are employed (Abdel-Hamid et al., 2012; Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov 2014). However, the fact that with hourly data, DNN now produces accuracy figures more similar to the shallower architecture models, suggests that the increased data size improves DNN's relative accuracy.

[Figure 1 about here]

[Table 4 ~ 7 about here]

[Figure 2 & 3 about here]

When employing the data at minute level, the number of observations increased substantially (to 800,000). An F-test then suggested that there were significant differences in the predictive accuracy of the four models ($F = 9.36$ $p < 0.0001$). Table 6 shows that the predictions from the two DNNs (using RELU and TANH activation functions) are significantly more accurate than those employing the SVM and NN models (see Figure 4). The two DNNs also have an HSD grouping of 'A' while SVM and NN have an HSD grouping of 'B'. SVM is slightly more accurate than NN, but not significantly so.

Finally, we explored the prediction accuracy of the four model specifications using tick level data (10 million observations for the training and validation). Table 7 shows that there was no significant difference between the performance of all four models ($F = 0.75$, $p > 0.5234$). The HSD groupings are all same letter – 'A'. Figure 5 shows a box plot of prediction accuracy for all four model specifications. It might have been expected that DNN with either activation function would perform better than both SVM and NN, given DNN's more complex architecture. However, our results are in line with those of Cho et al. (2016), who found that increasing the size of training data leads to an increase in predictive accuracy of DNN up to a point, but beyond that the accuracy does not change much regardless of training size. He attributed this to the fact that the training sets did not have a balanced class to reach the desired accuracy. This shows that the performance of DNN is not only influenced by sample size but by other combinations of factors (Ba et al., 2014). In particular, DNN can be very powerful because it can employ more parameters, it can learn more complex functions given the same number of parameters and it has better inductive bias (and, thus, learns more interesting/useful functions). However, it has also been shown in some cases that shallow neural networks can learn these deep functions using the same number of parameters as the original deep models. For example, on the TIMIT phoneme recognition and CIFAR-10 image recognition tasks, shallow networks were trained to perform similarly to complex, well-engineered, deeper convolutional architectures (Ba et al., 2014). In summary, the result presented does not support the hypothesis H1, that predictive accuracy is higher using DNN for intraday (i.e. minute/tick) time horizons than SVM and one-layer NN.

[Figure 4 ~ 6 about here]

The algorithms we have employed so far in our analysis have no memory. Each input is processed independently, with no states kept between inputs. With such networks, to process a sequence or a temporal series of data points, one must show the entire sequence of the network at once: to turn it into a single data point. A recurrent neural network (RNN) on the other hand adopts the same principle, albeit in a simplified manner: it processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. In effect, an RNN is a type of neural network that has an internal loop and there are two types of RNN: Long Short-Term Memory

(LSTM) and Recurrent Unit (GRU). To explore the robustness of our findings, we analyse results related to two recurrent neural networks (RNN models: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

The results of this analysis, presented in Tables 9-12 and Figures 8-11, lead us to similar conclusions to those we reached based on a DNN. For example, the F-test based on the analysis of daily data suggests that there are differences between the accuracy of the models i.e. the variation of model accuracy among the six models (numerator) is much larger than the variation accuracy within the six models ($F = 16.65$, $p<0.0001$). Several conclusions emerge from the results presented in Table 9. For daily data (1,500 observations), the accuracy of both SVM and RNN (GRU) are significantly better than all the other algorithms (HSD grouping is 'A'). The third best model is RNN (LSTM), which is not significantly better than NN and DNN (RELU), since all have an HSD grouping 'B'. However, these are significantly better than DNN (TANH) with HSD grouping 'C'. Figure 8 shows a box plot of prediction accuracy for all six model specifications. With the addition of the two RNN models, our results still show that SVM performs better than all DNN methods but not significantly better than RNN (GRU) using daily data.

We also compared the predictive accuracy of all six methods using hourly data (15,000 observations) and Figure 10 shows a box plot of these results. Using this data, the F-test produces an insignificant result ($F = 4.12$, $p>0.0014$), suggesting that there are no differences in the predictive accuracy of the different model specifications. Table 9 shows HSD groupings of all the same letter 'A,' which confirms our findings of no significant differences in accuracy between the model specifications. However, RNN (GRU) produced marginally better results than the rest; a result in line with Chung et al., 2014, who showed that RNN (GRU) is better at generalizing than RNN (LSTM) when parameters are updated and it can also solve complex long time lags.

[Table 9 - 12 about here]

[Figure 8 & 9 about here]

We also undertook the same comparisons of predictive accuracy between the models using data at the minute level, (observations increased substantially, to 800,000). An F-test then suggested that there *were* significant differences in the predictive accuracy of the six models ($F = 7.34$ $p <0.0001$). The results presented in Table 11 show that the predictions from the two DNNs (using RELU and TANH activation functions) are significantly more accurate than those from the RNN (GRU), RNN (LSTM), SVM and NN models (see Figure 10). The two DNNs also have an HSD grouping of 'A' while RNN (GRU), RNN (LSTM), SVM and NN all have an HSD grouping of 'B'.

Finally, we explored the predictive accuracy of the six model specifications using tick level data (10 million observations for the training and validation). The results presented in Table 12 show that there were no significant differences between the predictive performance of all six models ($F =$

0.48, p>0.7917). The HSD groupings for all six models are the same, i.e. 'A'. Figure 11 presents a box plot of prediction accuracy for all six model specifications. Despite introducing RNN models, which are considered start-of-the-art , we do not see any evidence to support H1, that predictive accuracy is higher using DNN for intraday (i.e. minute/tick) time horizons than SVM and one-layer NN.

[Figures 10 - 11 about here]

To test the second hypothesis, we explored whether RELU is better than TANH for stock price index predictions. Many practitioners have suggested that when employing DNN, one should start by using RELU, as this is regarded as a good approximator (Nair et al., 2010). Figure 6 shows boxplots of the predictive accuracy of DNN models using both RELU and TANH activation functions across all four-time horizons. Our results show that RELU is a better activation function for predicting stock price movement. This was expected, as many previous studies have found RELU to be far better than TANH (e.g., Krizhevsky et al., 2012). However, our results suggest that there are significant differences between the predictive accuracy which can be achieved when employing the RELU and TANH activation functions in DNN functions across the four-time horizons ($F = 16.87$, $p<0.0001$) (see Figure 6). Whilst the RELU activation function leads to better predictions across all time horizons, it was far better than TANH when employing intraday data, especially minute data, across all markets (see Table 6). Our findings support our second hypothesis that using the RELU activation function produces more accurate stock price index predictions than using the TANH activation function across all time horizons.

To test our third hypothesis, we examined whether the predictive performance of DNN was better when applied to emerging markets and whether it outperforms both SVM and NN in these markets. We believed this may be the case because of its ability to approximate complex functions (e.g., using intraday and daily stock price data, DNN can learn time shifted correlations among stock prices: Ibikule et al. (2017) and Krauss et al. (2016)) and the behaviour of emerging markets is considered to be more complex (cf. developed markets) (Berger, et al., 2011).

Figure 7, compares the accuracy of the DNN model in predicting prices in emerging vs developed markets across all four time frames (daily, hourly, minute, tick). The results suggest that there was no difference in the predictive accuracy achieved between emerging and developed markets using DNN for stock price index prediction when using daily, hourly or minute level data. However, when employing tick level data the predictive accuracy of a DNN model applied to emerging markets appears to be greater than that achieved for developed markets (i.e. superior accuracy of 10%). Emerging markets for which DNN produced particularly high accuracy (i.e. >70%) using tick level data include Thailand, Malaysia and Czech Republic.

[Figure 7 about here]

To formally test whether DNN produced more accurate predictions for emerging markets than SVM or NN, we examined the statistical significance of observed mean differences across factor levels when

using regression analysis (9). In particular, we estimate the following models to explain predictive accuracy:

$$Accuracy = \alpha + \beta_{DR1} DR1 + \beta_{DR2} DR2\, \varepsilon \ , \tag{9}$$

where DR1 and DR2 are dummy variables taking value 1 (for DNN (RELU) and 0 for SVM. DR2 and taking value 1 (for DNN (RELU) and 0 for NN, respectively. Fitting various algorithms as a factor variable in the model allows the response to vary in a nonlinear way and requires more parameters than a linear term. Factor variables are typically fitted using 'treatment contrasts', which means that one level of the factor variable forms the baseline and the remaining levels of the factor have corresponding coefficients.

The results of the regression analyses comparing the accuracy of DNN (RELU), SVM and NN are displayed in Table 8. The F-statistics and p-values confirm the statistical significance of the regressions. The adjusted R^2 values suggest that the independent variables explain about 19% of the observed predictive accuracy. This value may appear rather low. However, it is important to remember that the development of prices in financial markets is driven by a multitude of factors, many of which are not considered in this study.

In interpreting the result, it is important to remember the reference model that forms the basis of the comparison (i.e. the DNN (RELU) model). Considering the regression coefficient of the intercept in the accuracy regression model, such a model produces a directional accuracy regression model of 65%. The coefficients of SVM and NN are significantly different from DNN (see Table 8) and the negative signs shows a reduction in accuracy compared to DNN. We reject H3, since the accuracy of DNN in both emerging markets and developed markets were similar across all time horizons. When comparing DNN with SVM and NN, a few markets appear to be better, which might suggest why the coefficient of our regression for DNN is higher.

[Table 8 about here]

6 Discussion

Our results show that when employing daily and hourly data, shallow architectures (SVM and one-layer NN) produce more accurate stock index predictions than DNN. These results are in line with earlier studies which show shallow architectures produce better predictions when the dataset is small. It has been suggested that this arises because the data does not have a complex structure (Bengio et al., 2007).

As expected, we find that the accuracy of stock index price predictions using a DNN model are significantly better than that achievable using other commonly employed ML techniques (SVM and one-layer NN) when the data size increases significantly (i.e. when using a minute level data). However, we cannot accept hypothesis H1, that the predictive accuracy is higher using DNN (cf. SVM and one-layer NN) for all intraday time horizons, because when using tick data (over 10 million cf. 800,000

observations for minute level data)), the predictive accuracy of DNN models using both types of activation function were no better than SVM. This may appear surprising, as previous studies have suggested that the performance of DNN increases as data size increases. However, Cho et al. (2016) found that an increase in the size of training data for DNN led to improvements in predictive accuracy up to a certain point (an increase from 5 to 50 training images), after which accuracy did not change substantially, regardless of the training size. This might explain the behaviour we experience using this data. On the other hand, Ba et al. (2014) suggested that in some cases shallow neural networks are able to learn deep functions using the same number of parameters as the original deep learning. He used TIMIT phoneme recognition and CIFAR-10 dataset to demonstrate this. In addition, when he trained shallow neural networks directly on the original labelled training data with the same number of parameters as DNN, he noted that shallow neural networks learned to mimic a DNN with high fidelity. Both shallow neural networks and DNN are universal, that is they can approximate arbitrarily well any continuous function of variables in a compact domain. The assumption that DNN performs better than a shallow architecture using more data applies mostly to deep convolutional networks where the locality of the functions are approximated at each stage, such as in computer vision, text and speech (Mhaskar et al. 2016). This is clearly not the case in our situation. The nature of our problem is not compositional i.e. our data does not have a compositional structure $f(x_1, \ldots\ldots, x_d) = h_1(h_2 \ldots\ldots (h_j(h_{i1}(x_1, x_2), h_{12}(x_3, x_4)), \ldots\ldots)$, hence we do not expect any accuracy advantage that can be derived using deep convolutional networks. This suggests that the function learned by the DNN we employ here does not really have to be very deep.

Our results offer support for H2, namely, that the RELU activation function produces more accurate stock price index predictions than TANH across all time horizon. This further explains why Nair et al. (2010) used CIFAR- 10 data to build a four-layer convolutional network and concluded that RELU was several times faster and better than TANH. One of the limitations of TANH activation functions is that they saturate. This means that large values snap to 1.0 and small values snap to -1.0. The function is also not sensitive to changes around its mid-point, such as 0.0. According to Krizhevsky et al., (2012), the biggest advantage of RELU is non-saturation of its gradient, which greatly accelerates the convergence of stochastic gradient descent compared to the TANH function. Another nice property is that compared to TANH neurons that involve expensive operations, the RELU can be implemented by simply thresholding a matrix of activations at zero. Glorot et al., (2011) carried out various experiments comparing the performance of RELU vs TANH using four image classification datasets of different scales. They noted that RELU can find local minima of greater or equal quality than those obtained with its smooth counterpart, TANH. They also noted that RELU is computationally efficient. RELU truly proves itself when used to solve non-zero-centred problems, such as in our case. In addition, Nair et al. (2010) explored various rectified nonlinearities and found them to improve performance most

especially for classification problems.Consequently, it is not surprising that most papers that achieve state-of-the-art results have used RELU, rather than TANH (e.g., Krizhevsky et al., 2012).

Our results do not fully support H3, that the predictive accuracy of DNN is generally greater in emerging markets than in developed markets. In fact, there was no significant difference in predictive accuracy in emerging markets when using DNN and SVM for daily, hourly and minute level data. However, we did observe higher predictive accuracy when using DNN (cf. SVM) to forecast whether a stock price index rises or falls using tick level data for three specific emerging markets, Thailand, Malaysia and Czech Republic.

Sutsarun et. al (2014) studied the impact of *coup d'etats* in Thailand in 2006 and examined the effect on both short-run and long-run returns, volatility, liquidity, and liquidity risk of returns on the Stock Exchange of Thailand index over the period 1 January 1996 to 31 December 2011. They used tick data in their analysis and found that the immediate reaction to the coup was more evident in the stock market with a reduction in stock returns. This period overlaps with the data period used for our analysis and could explain why DNN was able to capture the complexity better than a shallow architecture in this market using tick data. As noted by Mhaskar et al. (2017), a shallow architecture in such cases is unable to account for the hierarchical network matching structure; that is shallow architectures can only exploit much smaller associated complexity, while DNN is "better" at approximating it.

Similarly, it has been found that Central and Eastern European countries (CEEC) such as Czech Republic suffer from spill over effects transmitted from major stock market turmoil in the USA and China (Deltuvaitė Vilma, 2016). The collapse of Lehman Brothers bank in United States in 2008 was the most significant shock transmitted to stock markets. Their empirical results also suggest that the transmission of other systemic shocks (e.g. the Middle East financial markets crash (May 2006), Greek debt crisis (April 23, 2010), Portugal's debt crisis (May 16, 2011)) was also observed in some of the CEECs countries. This in turn requires that the network learns a compact representation, and this is only fully possible with a deep architecture. This could explain why shallow architectures were not able to match the performance of DNN in this market. Leow and Evelite (2015) examined the presence of a political cycle in Malaysian stock market returns and volatilities in the period of February 1982 to April 2012. They claimed that the Malaysian stock market tends to overreact to unexpected political events such as removal of a Deputy Prime Minister and the resignation of the Prime Minister. Their research confirmed previous studies, which showed that the presence of a political cycle in Malaysia stock market volatilities is statistically significant. Our results suggest that using DNN in such markets with a RELU activation function can help capture the sudden change in market dynamics. A shallow architecture on the other hand does not have a sufficiently complex network to guarantee a given accuracy of an unknown target function f, necessary in this market.

7 Conclusions

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years, such as in computer vision and natural language processing. It has turned out to be very successful at discovering intricate structures in high-dimensional data and is, therefore, applicable to many domains of science, business and government. In addition, DNNs have been predicted to have many more successes in the near future, largely because they require very little engineering by hand. Consequently, DNNs can easily take advantage of increases in the amount of available computational ability and data. With so many successful applications of DNN, it is perhaps surprising that few studies have employed DNN to forecast financial time series (Krauss et al., 2016). Our paper is the first, to the best of our knowledge, to use DNN in the context of predicting the direction of stock price movements across multiple markets in order to understand to what extent this novel algorithm is sensitive to sample sizes. We set out to clarify if this is the case by comparing the predictive accuracy of widely used shallow methods with DNN using different forecast time horizons (daily/hourly/minute/tick). The main aim of our paper is not to optimize the forecasting performance of a DNN model in any given market. Rather, the aim is to compare (across a wide range of markets) the forecasting performance of DNN with those ML methods which have dominated previous research studies, to identify under what conditions DNN outperforms these models and which activation function in DNN (RELU or TANH) produces the best results.

As expected, our results show that data size has an effect on the relative predictive performance of DNN. SVM and one-layer NN outperformed DNN when daily and hourly data was used. However, the predictive accuracy of DNN was significantly better than that of SVM and one-layer NN using minute level data. Previous research has suggested that DNNs perform particularly well when they can learn the underlying structure using large datasets. It is likely that this explains our results, since there is a significant increase in data size from the 1,500 observations for daily prediction to the 800,000 observations for minute predictions (Bengio et al., 2007).

Interestingly, the accuracy of stock index price predictions at the tick level based using DNN was not significantly better than predictions based on shallower architectures. Since many studies have advocated the use of DNN when confronted with complex, big data, one would have expected DNN to have a better predictive performance than other methods at the tick price level. Some studies have suggested that this is not always the case (e.g., Cho et al., 2016) and further research is needed, employing data other than that related to stock markets, to see if similar findings are obtained. To test the robustness of our approach we introduced two widely used advanced RNN methods (i.e. LSTM and GRU) since they are considered to have memory. Our results do not show any significant improvement from using DNN.

Taken together, our results suggest that practitioners who are looking to include DNN as one of the algorithms they use when predicting stock price movement, should first consider the complexity

of the decision space, the size of the dataset and how balanced is the target class. A suitable future research avenue might be to develop an ensemble model which combines the power of SVM and DNN (Krauss et al., 2016).

A second important contribution of the research is that we have demonstrated that the RELU activation function outperforms the TANH activation function when employing DNN across all forecast time horizons (daily, hourly, minute and tick level). Carefully tuning the hyper parameter optimization may still yield advantageous results for both activation function. This is subject to further research using more stock market data. In case a practitioner is oblivious to which activation function to apply and if cheap computing power is available, RELU should be explored extensively to gain better predictive accuracy.

A third contribution of the paper is the discovery that DNN outperforms shallower architectures at the tick level in some emerging markets. In particular, we found that predictions based on DNN at the tick level were significantly more accurate than those based on shallower architectures for the Thai, Czech and Malaysian markets. We believe DNN was able to capture the recent upward trend in their economic growth which was also reflected in the data. The high levels of predictive accuracy achieved for these markets (79%, 84% and 79%, respectively) suggest significant exploitable inefficacies in these markets. Since stock markets are rapidly changing in emerging markets, practitioners should always compare both shallow architectures and DNN before taking any decision. Further research should explore datasets from sub-markets within emerging markets using DNN.

Overall, this paper has implications for financial economics and professional finance practitioners. In particular, we provide empirical evidence that the most widely celebrated machine learning technique (DNN) does not necessarily outperform SVM and NN in all cases using large data from many major financial markets. We show that RELU should be leveraged widely by practitioners as this was better in comparison to TANH when analysing stock market data. Lastly, none of the algorithms we tested were superior for predicting stock price indices in emerging *vs* developed markets, even though all the methods offer the prospect of identifying inefficiency in the pricing in these markets.

References

Angelidis, T., & Degiannakis, S. (2008). Volatility forecasting: Intra-day versus inter-day models. *Journal of International Financial Markets, Institutions and Money*. 18: 449-465. Doi:10.1016/j.intfin.2007.07.001.

Abdel-Hamid, O., Mohamed, A-r.,Jiang, H., & Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing*. 4277-4280.

Ackley, D.H., Hinton, G.E., & Sejnowski,T.J (1985). A learning algorithm for boltzmann machines. *Cognitive Science*, 9 (1): 147-169.

Alexandre, P., Sabino, & S.,Albuquerque, P. (2015). Portfolio Selection with Support Vector Machines in Low Economic: Perspectives in Emerging Markets. *Economic computation and economic cybernetics studies and research / Academy of Economic Studies.* 49(4):261.

Ba, L.J., & Caruana, R. (2014). Do deep nets really need to be deep? Proceedings of the 27th International Conference on Neural Information Processing Systems, 2: 2654-2662.

Ballings, M., Van den Poel, D., Hespeels, N. , & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42 (20): 7046–7056.

Bekaert, G., & Harvey, C. R. (1997). Emerging equity market volatility. *Journal of Financial Economics*, 43 (1): 29-77.

Bengio, Y. (2013). Deep Learning of Representations: Looking Forward. *Statistical Language and Speech Processing. SLSP. Lecture Notes in Computer Science,* 7978. https://doi.org/10.1007/978-3-642-39593-2_1.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Journal Foundations and Trends in Machine Learning archive*, 2(1): 1-127.

Bengio, Y., & Lecun, Y. (2007). Scaling learning algorithms towards AI. *In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), Large-scale kernel machines MIT Press.*

Ben-Hur, A., & Weston, J. (2010). A user's guide to support vector machines. *Methods in Molecular Biology series*. 2010; 609:223-39. https://doi.org/10.1007/978-1-60327-241-4_13.

Berger, D., Pukthuanthong, K. J., & Yang, J. (2011). International Diversification with Frontier Markets, *Journal of Financial Economics*, 101(1), 227-242.

Bodyanskiy, Y., & Popov, S. (2006). Neural network approach to forecasting of quasiperiodic financial time series. *European Journal of Operational Research*, 175 (3), 1357–1366.

Bordes, A., Chopra, S., & Weston, J. (2014). Question answering with subgraph embeddings. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. http://arxiv.org/abs/1406.3676v3.

Boser, B.E., Guyon, I. M., & Vapnik,V.N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*: 144-152.

Cenesizoglu, T., & Timmermann, A. (2008). Is the Distribution of Stock Returns Predictable? http://dx.doi.org/10.2139/ssrn.1107185.

Chang, P.-C., Liu, C.-H., Lin, J.-L., Fan, C.-Y. , & Ng, C. S. (2009). A neural network with a case based dynamic window for stock trading prediction. *Expert Systems with Applications,* 36: 6889–6898.

Cho, J., Lee, K., Shin, E., Choy, G., & Do, S. (2016). How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? https://arxiv.org/abs/1511.06348.

Collobert, R. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12: 2493–2537.

Cortes, C. & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20: 273-297. https://doi.org/10.1023/A:1022627411411.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS Workshop on Deep Learning.*

de Oliveira, F. A., Nobre, C. N., & Zárate, L. E. (2013). Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index-case study of PETR4, *Petrobras, Brazil. Expert Systems with Applications*, 40 (18): 7596–7606.

Deltuvaitė, V. (2016). Transmission of Shocks through Stock Markets Channel: The Case of the CEECs. *Procedia Economics and Finance*. 39: 292-297.

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., & Williams, J. (2013). Recent advances in deep learning for speech research at Microsoft in Acoustics, *Speech and Signal Processing (ICASSP), IEEE International Conference*, 8604–8608.

Eunsuk, C., Chulwoo, H., & Parka, F.C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications* 83: 187–205.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25 (2): 383–417.

Fama, E. F. (1991). Efficient capital markets: II. *The Journal of Finance*, 46 (5): 1575–1617.

Farabet, C., Couprie, C., Najman, L., & Lecun, Y. (2013). Learning hierarchical features for scene labeling, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8):1915-29. doi: 10.1109/TPAMI.2012.231.

Fischer, T., & Krauss, C. (2017). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2) 10.1016/j.ejor.2017.11.054.

Fukushima, K. (2003). Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180.

Gerlein, E. A., McGinnity, M., Belatreche, A. , & Coleman, S. (2016). Evaluating machine learning classification for financial trading: An empirical approach. *Expert Systems with Applications*, 54, 193–207.

Glorot, X., Bordes, A., Bengio, Y. (2010). Deep Sparse Rectifier Neural Networks. *Journal of Machine Learning Research*, 15, 513-520.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., & Bengio, Y. (2013). Maxout Networks, *International Conference on Machine Learning* – 28: III,1319-III,1327.

Griffin, J. M., Kelly, P., & Nardari, F. (2010). Do Market Efficiency Measures Yield Correct Inferences? A Comparison of Developed and Emerging. *Markets Review of Financial Studies*, 2 (8): 3225-3277.

Harvey, C. R. (1995). Predictable Risk and Returns in Emerging Markets. *Review of Financial Studies*, 8(3): 773–816.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV*, 770-778. doi: 10.1109/CVPR.2016.90.

Hinton, G.E., Osindero, S., & Teh, Y. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18: 1527–1554.

Hsu, M-W., Lessmann, S., Sung, M-C., Ma, T., & Johnson, J.E.V. (2016). Bridging the divide in financial market forecasting: machine learners vs. financial economists. *Expert Systems with Applications*, 61: 215-234.

Huang, C.-J., Yang, D.-X., & Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, 34, 2870–2878.

Huang, F. J., & LeCun, Y. (2006). Large-scale learning with SVM and convolutional nets for generic object categorization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR (Vol. 1, pp. 284-291). [1640771] DOI: 10.1109/CVPR.2006.164.

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

Huanhuan, Y., Rongda, C., & Zhang, G. (2014). A SVM stock selection model within PCA. *Procedia Computer Science*, 10.1016/j.procs.2014.05.284.

Ibikunle, G., Möws, B. (2017). How random are intraday stock prices? Evidence from deep learning. Edinburgh Business School, Working paper. Available at: https://www.research.ed.ac.uk/portal/files/32816714/Ibikunle_and_M_ws_2017.pdf (Accessed: 11 November 2018).

Jaakkola, T., & Haussler, D. (1998). Exploiting Generative Models in Discriminative Classifiers, *In Advances in Neural Information Processing Systems*, II, 487-493.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *IEEE 12th International Conference on Computer Vision, Kyoto*: 2146-2153, doi: 10.1109/ICCV.2009.5459469.

Kara, Y., Acar Boyacioglu, M., & Baykan, Ö. 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: 5311–5319.

Karemera, D., Ojah, K., Cole, J.A. (1999). Random Walks and Market Efficiency Tests: Evidence from Emerging Equity Markets. *Review of Quantitative Finance and Accounting* (1999) 13: 171. https://doi.org/10.1023/A:1008399910942.

Kearney, C. (2012). Emerging Markets Research: Trends, Issues and Future Directions. *Emerging Markets Review*, 13 (2): 159-183.

Krauss, C., Anh Do, X., & Huck, N. (2016). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2), 10.1016/j.ejor.2016.10.031.

Krizhevsky, A., Ilya, S., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *International Conference on Neural Information Processing Systems*, 1: 1097-1105.

Le, Q.V., Ranzato, M.A., Monga, R., Devin, M., Chen,K., Corrado, G.S., Dean, J., & Ng, A.Y.(2012). Building high-level features using large scale unsupervised learning. *International Conference on Machine Learning. Proceedings of the 29th*: 507-514.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature,* 521 (7553): 436–444. http://dx.doi.org/10.1038/nature14539 10.1038/nature14539.

Lecun, Y., Farabet, C., Couprie C., & Najman, L. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 35(8):1915-29 doi: 10.1109/TPAMI.2012.231.

Leow, S., Celis, E. (2015). Political cycle and stock market - the case of Malaysia. *Journal of Emerging Issues Econ Finance Bank*. 4(1),1461–1512.

Lesmond, D. A., Schill, M. J., & Zhou, C. (2004). The illusory nature of momentum profits. *Journal of Financial Economics*, 71(2): 349–380.

Lessmann, S., Sung, M., & Johnson, J. E. (2011). Towards a methodology for measuring the true degree of efficiency in a speculative market. *Journal of the Operational Research Society*, 62: 2120–2132.

Leung, M. K., Xiong, H. Y., Lee, L. J., & Frey, B. J. (2014). Deep learning of the tissue regulated splicing code. *Bioinformatics* 30: i121–i129.

Lo, A. W. (1991). Long-Term Memory in Stock Market Prices. *Econometrica*, 59 (5), 1279–1313.

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., & Svetnik, V. (2015). Deep neural nets as a method for quantitative structure-activity relationships. *Journal of Chemical Information and Modeling,* 55: 263–274.

Mathieu, M., LeCun, Y., Fergus, R., Eigen, D., Sermanet, P., & Zhang, X. (2013). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *International Conference on Learning Representations (ICLR2014)*, http://arxiv.org/abs/1312.6229.

Mhaskar, H., Tomaso, P. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*. 14 doi:10.1142/S0219530516400042.

Mittermayer, M.-A. (2004). Forecasting Intraday stock price trends with text mining techniques. *Proceedings of the 37th annual Hawaii international conference*: 10.

Nair, V., & Hinton, G.E. (2010). Rectified linear units improve restricted boltzmann machines. *International Conference on International Conference on Machine Learning*: 807-814.

Pan, H., Chandima, T., & Yearwood, J. (2005). Predicting Australian Stock Market Index Using Neural Networks Exploiting Dynamical Swings and Intermarket Influences. *Journal of Research and Practice in Information Technology*, 37(1) 10.1007/978-3-540-24581-0_28.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, 42(1), 259-268.

Qian, B., & Rasheed, K. (2007). Stock market prediction with multiple classifiers. *Applied Intelligence*, 26: 25–33 http://doi.org/ . doi: 10.1007/s10489- 006- 0001- 7.

Ren, Y., & Wu, Y. (2014). Convolutional deep belief networks for feature extraction of EEG signal. *International Joint Conference on Neural Networks (IJCNN), Beijing*, 2850-2853. doi: 10.1109/IJCNN.2014.6889383.

Rumelhart, D. E., Hinton, G. E., &Williams, R. J. (1986). Learning internal representations by error-propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Volume 1, Issue 6088: 318-362.

Sadique, S., & Silvapulle, P. (2001). Long-Term Memory in Stock Market Returns: International Evidence. *International Journal of Finance & Economics*. (6) 59-67. 10.1002/ijfe.143.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine learning.*791-798.

Scholkopf, B., & Cristianini, N. (2002). Support vector machines and kernel methods: The new generation of learning machines. *Ai Magazine,* Volume 23 Issue 3: 31-41.

Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems* (TOIS), 27(12).

Seide, F., Li, G., Xie, C., & Dong, Y. (2011). Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription. *IEEE Workshop on Automatic Speech Recognition & Understanding*. 10.1109/ASRU.2011.6163899.

Shawe-Taylor, J., & Cristianini, N. (2004). Kernel Methods for Pattern Analysis. *Cambridge university press.*

Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., & Salakhutdinov R (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research Volume* 15 Issue 1: 1929-1958.

Steel, R.G.D., Torrie, J.H. (1980). *Principles and Procedures of Statistics, Second Edition, New York: McGraw-Hill Book Co.*

Sutsarun, L., Sirimon, T., Wee, M., & Brooks, R (2014). Thai Financial Markets and Political Change. *Journal of Financial Management, Markets and Institutions*, Issue 1: 5-26.

Timmermann, A., & Granger, C.W.J. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting* Volume 20 Issue 1:15–27.

Titan, A. G. (2015). The Efficient Market Hypothesis: review of specialized literature and empirical research. *Procedia Economics and Finance* 32(2015) 442 – 449.

Tompson, J., Stein, M., LeCun, Y., & Perlin, K., (2014). Real-time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACMTrans. Graph*. doi:10.1145/2629500.

Töscher, A., Jahrer, M., & Bell, R. M. (2009).The Big Chaos solution to the Netflix grand prize. http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf.

Williams, C. K. I., & Rasmussen, C.E. (1996). Gaussian Processes for Regression. *In Advances in Neural Information Processing Systems,* 8. eds. D. S. Touretzky, M. C. Mozer, M. E. Hasselmo, MIT Press.

Xu, Q., Zhou, H., Wang, Y., & Huang, J. (2009). Fuzzy support vector machine for classification of EEG signals using wavelet-based features. *Journal of Medical Engineering & Physics*, 31(7):858-65. doi: 10.1016/j.medengphy.

Żbikowski, K. (2015). Using Volume Weighted Support Vector Machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Systems with Applications*, 42(4): 1797-1805.

Zhang, B. G., Patuwo, E, & Hu,Y. M (1998). Forecasting with artificial neural networks: *The state of the art. International Journal of Forecasting*, 14 (1): 35-62.

Zhang, Y., & Wu, L. (2009). Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network. *Expert Systems with Applications*, 36(5): 8849-8854.

Table 1: Summary of experimental settings

Hypotheses	Experimental Factor	Factor Levels
H1	Forecast horizon	Intraday *vs.* Daily
H2	Activation functions	TANH *vs.* RELU
H3	Market classification	Emerging *vs*. developed market

Table 2 Classification of the financial markets employed in this study as ‘emerging’ and ‘developed’

Emerging market classification		Economy	Index
FTSE	MSCI		
		Spain	IBEX 35
		Switzerland	Swiss Market Index
		Finland	OMXH25
Y	Y	Turkey	ISE-100
		France	CAC 40
		Sweden	OMX ALL - SHARE Stockholm Index
		Norway	OSE All Share Index
Y	Y	Brazil	Brazilian Bovespa Futures
		Hong - Kong	Hang Seng index
Y	Y	Czech	Prague Stock Exchange Index
		US	Dow Jones Industrial Average
		Denmark	OMX Copenhagen Index
		Latvia	Riga Index
		Austria	ATX
		US	S&P 500
Y	Y	South Africa	FTSE/JSE Africa Top40
Y	Y	Hungary	BUX
		UK	FTSE 100
Y	Y	Thailand	Thai Stock exchange MAI securities index
		Germany	DAX
		Korea	KOSPI 200 Index
		Singapore	Straits times index
		Netherland	AEX
		Belgium	BEL20
		Japan	Nikkei 225
		Canada	SP TSX composite index
		US	NASDAQ-100
Y	Y	China	ShangHai SE composite index
Y	Y	Indonesia	Jakarta composite index
		Italy	FTSE MIB Index
		Portugal	PSI-20
		Estonia	OMX Tallinn index
Y	Y	Malaysia	FTSE Bursa Malaysia KLCI index
		Lithuania	OMX Vilnius index

Table 3a: The numbers of levels tested in different parameter settings for SVM

Parameters	Levels (polynomial)
Degree of kernel function (*d*)	1,2,…..5
Gamma in Kernel function (γ)	0,0.1,0.2,….,5.0
Regularization parameter (*c*)	1,10,100

Table 3b: The numbers of levels tested in different parameter settings for NN

Parameters	Levels(s)
Number of neurons (*n*)	10,20,……,100
Epochs (*ep*)	1,2,…..,10
Momentum constant (*mc*)	0.1,0.2,…..,0.9
Learning rate (*lr*)	0.1,0.2

Table 3c: The numbers of levels tested in different parameter settings for DNN

Parameters	Levels(s)
Number of neurons (*n*)	10,20,……,100
Epochs (*ep*)	1,2,…..,10
Momentum constant (*mc*)	0.1,0.2,…..,0.9
Learning rate (*lr*)	0.1,0.2
Hidden layers	2,3,……,10
Activation functions	RELU/TANH
MiniBatchSize	5,6,7 ……,15

Table 4: Tukey's Standardized Range (HSD) test: Comparing the Accuracy of Models using Daily Data

Alpha	0.05
Error of degree of freedom	132
Error mean square	0.005
Critical value of Studentized Range	3.679
Minimum Significant Difference	0.045

Tukey Grouping	Mean	Methods
A	0.578	SVM
B	0.522	NN
B	0.489	DNN(RELU)
C	0.452	DNN(TANH)

Table 5: Tukey's Standardized Range (HSD) test for Accuracy of Models using Hourly Data

Alpha	0.05
Error of degree of freedom	132
Error mean square	0.005
Critical value of Studentized Range	3.69
Minimum Significant Difference	0.045

Tukey Grouping	Mean	Methods
A	0.557	SVM
A	0.549	DNN(RELU)
A	0.538	DNN(TANH)
A	0.514	NN

Table 6: Tukey's Standardized Range (HSD) test for Accuracy using Minute Data

Alpha	0.05
Error of degree of freedom	132
Error mean square	0.008
Critical value of Studentized Range	3.68
Minimum Significant Difference	0.057

Tukey Grouping	Mean	Methods
A	0.645	DNN(RELU)
A	0.617	DNN(TANH)
B	0.558	SVM
B	0.547	NN

Table 7: Tukey's Standardized Range (HSD) test for Accuracy of Models using Tick Data

Alpha	0.05
Error of degree of freedom	132
Error mean square	0.009
Critical value of Studentized Range	3.68
Minimum Significant Difference	0.062

Tukey Grouping	Mean	Methods
A	0.607	DNN(RELU)
A	0.585	SVM
A	0.582	DNN(TANH)
A	0.573	NN

Table 8: Regression analysis of Predictive accuracy – DNN (RELU) vs SVM vs NN

Predictive accuracy	Estimated Coefficient	Std. Error	t value	Pr (>|t|)
(Intercept)	0.64559	0.01489	43.358	< 2e-16***
as.factor(NN)	-0.09824	0.02106	-4.665	9.67e-06***
as.factor(SVM)	-0.08765	0.02106	-4.162	6.73e-05***
Residual standard error	0.08682	df	99	
		Adjusted R -		
R-Squared	0.2094	Squared	0.1935	
F- Statistic	13.11	(on 2 and 99 DF)		
p-value	8.87E-06			

Table 9: Tukey's Standardized Range (HSD) test: Comparing the Accuracy of Models using Daily Data

Alpha	0.05
Error of degree of freedom	198
Error mean square	0.005
Critical value of Studentized Range	4.1
Minimum Significant Difference	0.045

Tukey Grouping	Mean	Methods
A	0.578	SVM
A	0.555	RNN(GRU)
B	0.524	RNN(LSTM)
B	0.522	NN
B	0.489	DNN(RELU)
C	0.452	DNN(TANH)

Table 10: Tukey's Standardized Range (HSD) test for Accuracy of Models using Hourly Data

Alpha	0.05
Error of degree of freedom	198
Error mean square	0.005
Critical value of Studentized Range	4.07
Minimum Significant Difference	0.047

Tukey Grouping	Mean	Methods
A	0.585	RNN(GRU)
A	0.557	SVM
A	0.557	RNN(LSTM)
A	0.549	DNN(RELU)
A	0.538	DNN(TANH)
A	0.514	NN

Table 11: Tukey's Standardized Range (HSD) test for Accuracy using Minute Data

Alpha	0.05
Error of degree of freedom	198
Error mean square	0.008
Critical value of Studentized Range	4.01
Minimum Significant Difference	0.061

Tukey Grouping	Mean	Methods
A	0.645	DNN(RELU)

A	0.617	DNN(TANH)
B	0.566	RNN(LSTM)
B	0.558	SVM
B	0.552	RNN(GRU)
B	0.547	NN

Table 12: Tukey's Standardized Range (HSD) test for Accuracy of Models using Tick Data

Alpha	0.05
Error of degree of freedom	198
Error mean square	0.009
Critical value of Studentized Range	4.07
Minimum Significant Difference	0.068

Tukey Grouping	Mean	Methods
A	0.607	DNN(RELU)
A	0.591	RNN(GRU)
A	0.591	RNN(LSTM)
A	0.585	SVM
A	0.582	DNN(TANH)
A	0.573	NN

Figure 1: Boxplot, comparing the forecasting accuracy of all four model specifications (SVM, NN, DNN (RELU), DNN (TANH)), across all four time horizons (daily, hourly, minute, tick).

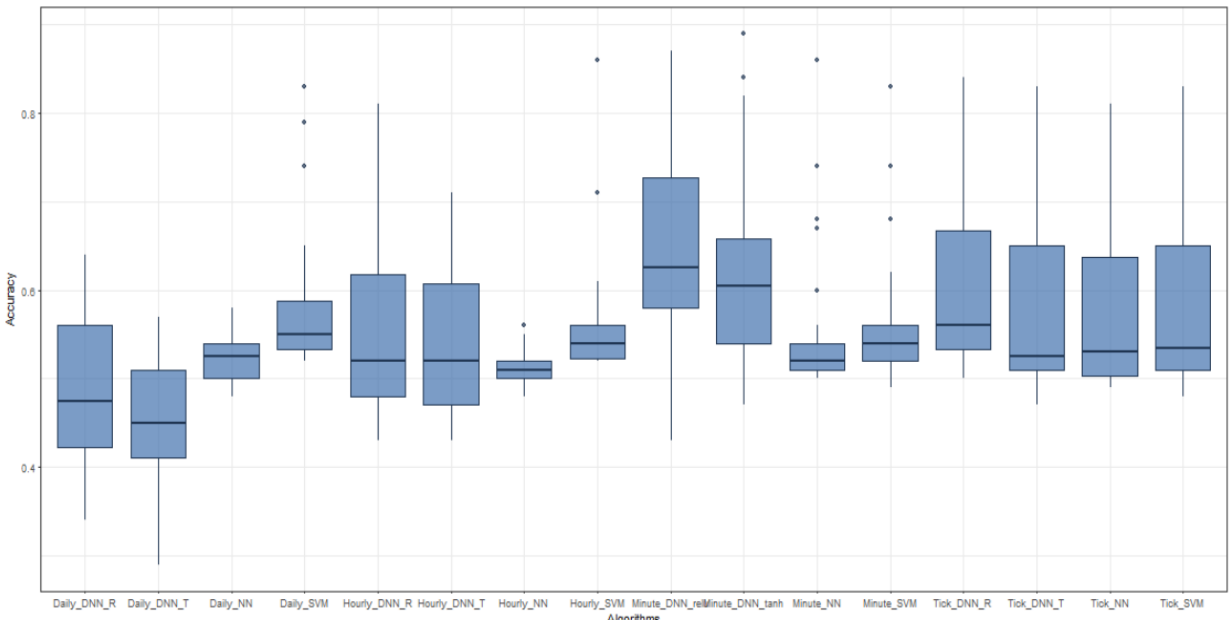

Figure 2: Box plot of prediction accuracy for all four model specifications (SVM, NN, DNN (RELU), DNN (TANH)), using daily data

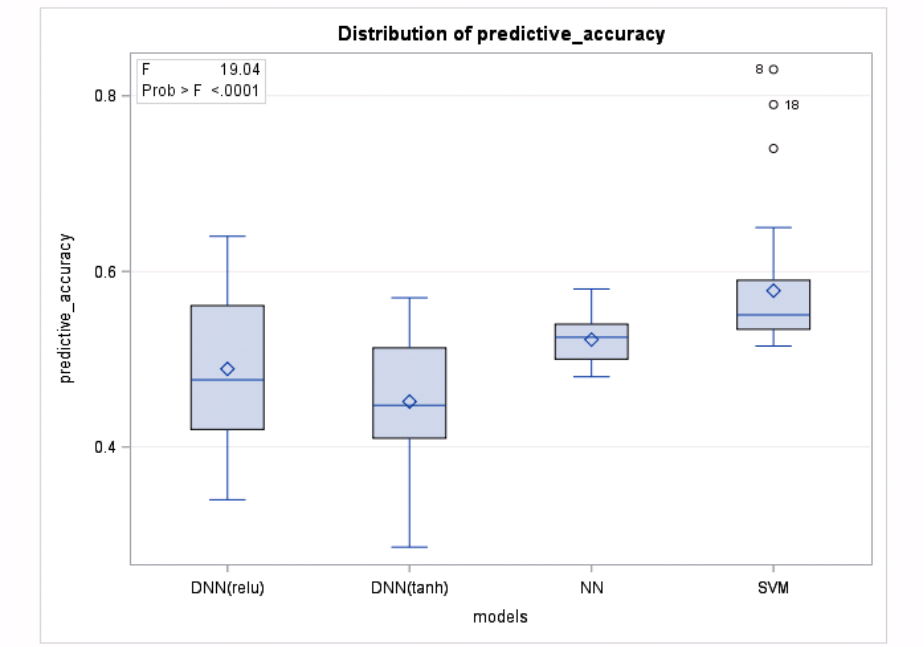

Figure 3: Box plot of accuracy for each all four model specifications (SVM, NN, DNN (RELU), DNN (TANH)), using hourly data

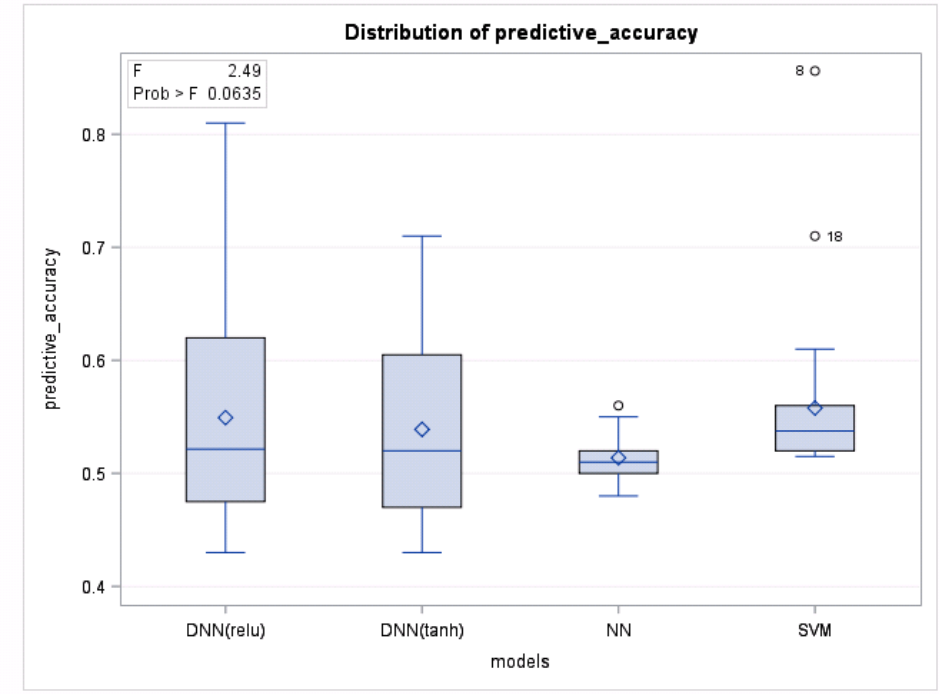

Figure 4: Box plot of accuracy for each of the four model specifications (SVM, NN, DNN using minute data

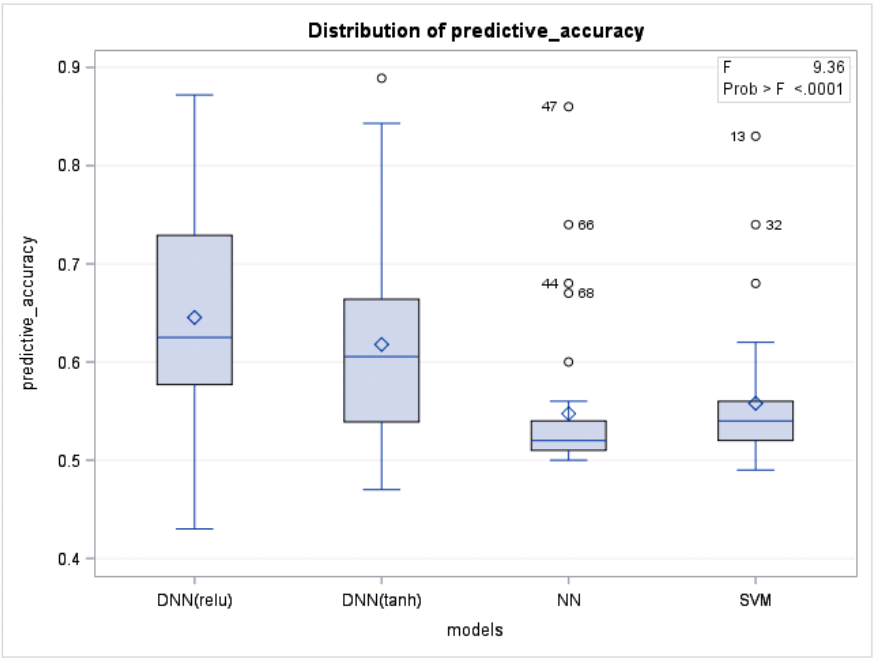

Figure 5: Box plot of accuracy all four model specifications (SVM, NN, DNN (RELU), DNN (TANH)), using tick data

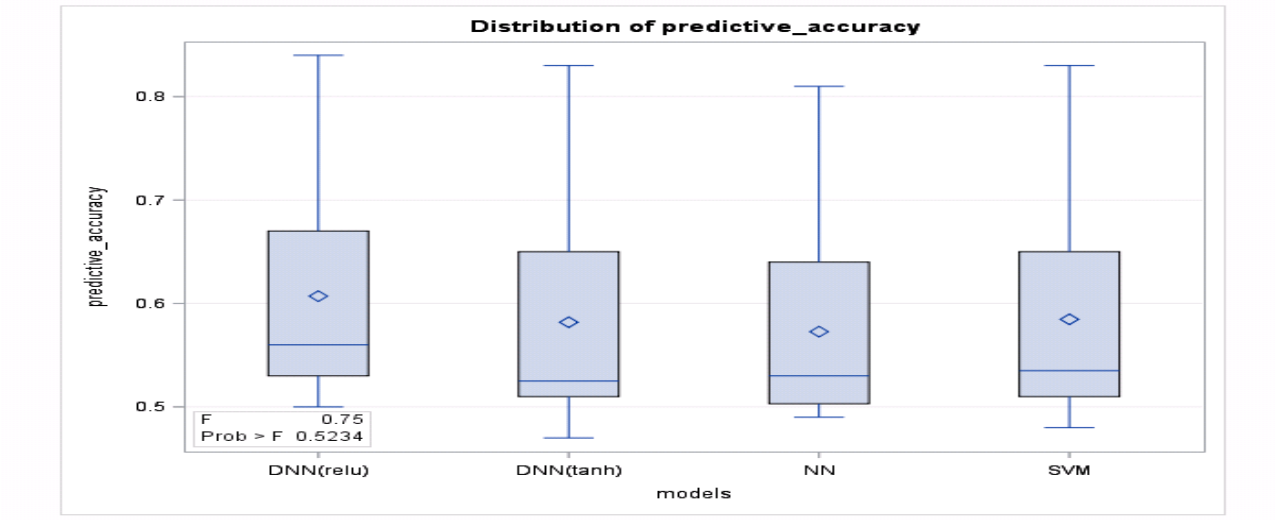

Figure 6: Box plot of accuracy for DNN using RELU/TANH activation functions across all four time horizons (daily, hourly, minute, tick).

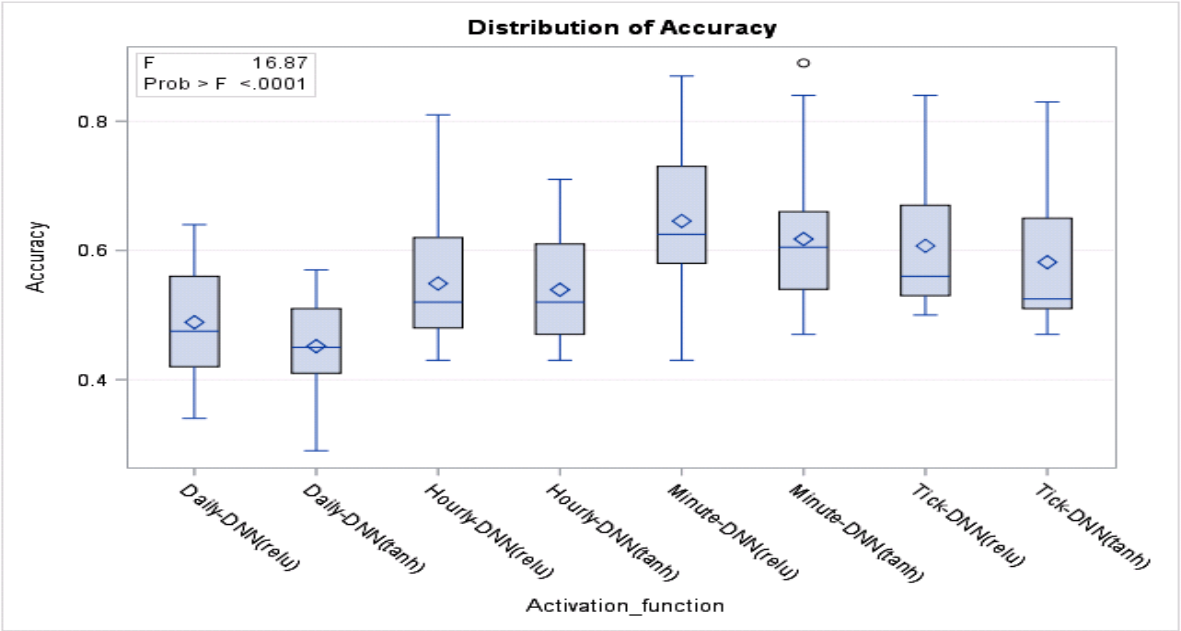

Figure 7: Predictive accuracy of emerging markets VS developed markets using DNN

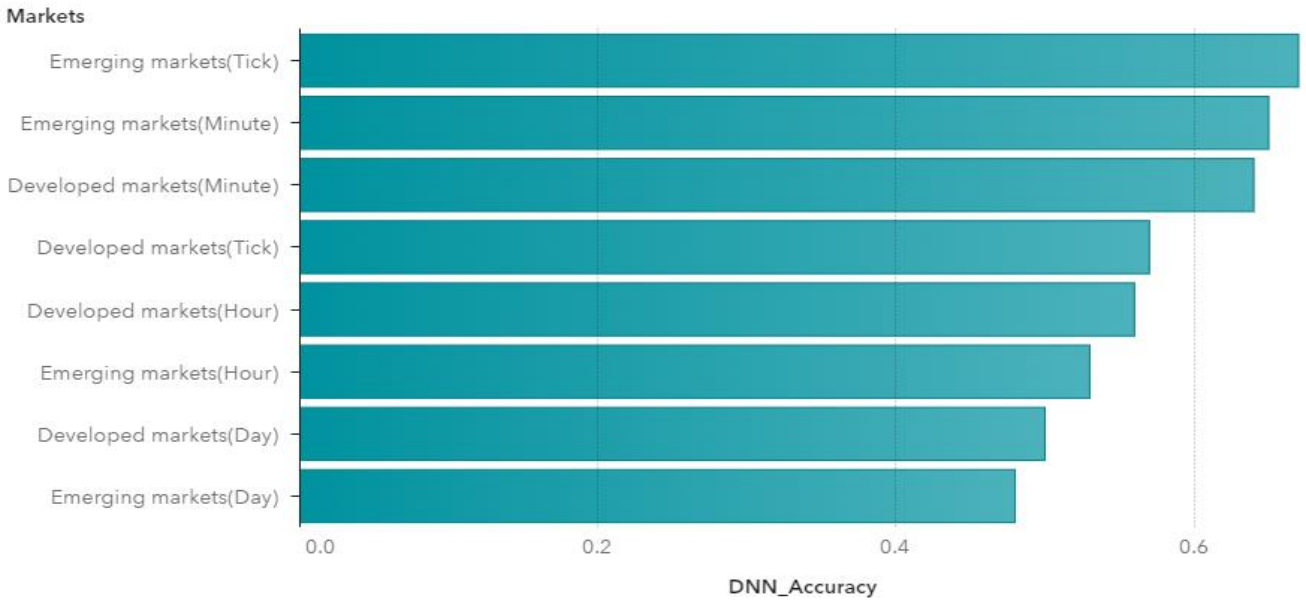

Figure 8: Box plot of prediction accuracy for all six model specifications (SVM, NN, DNN (RELU), DNN (TANH), RNN (LSTM), RNN (GRU)), using daily data

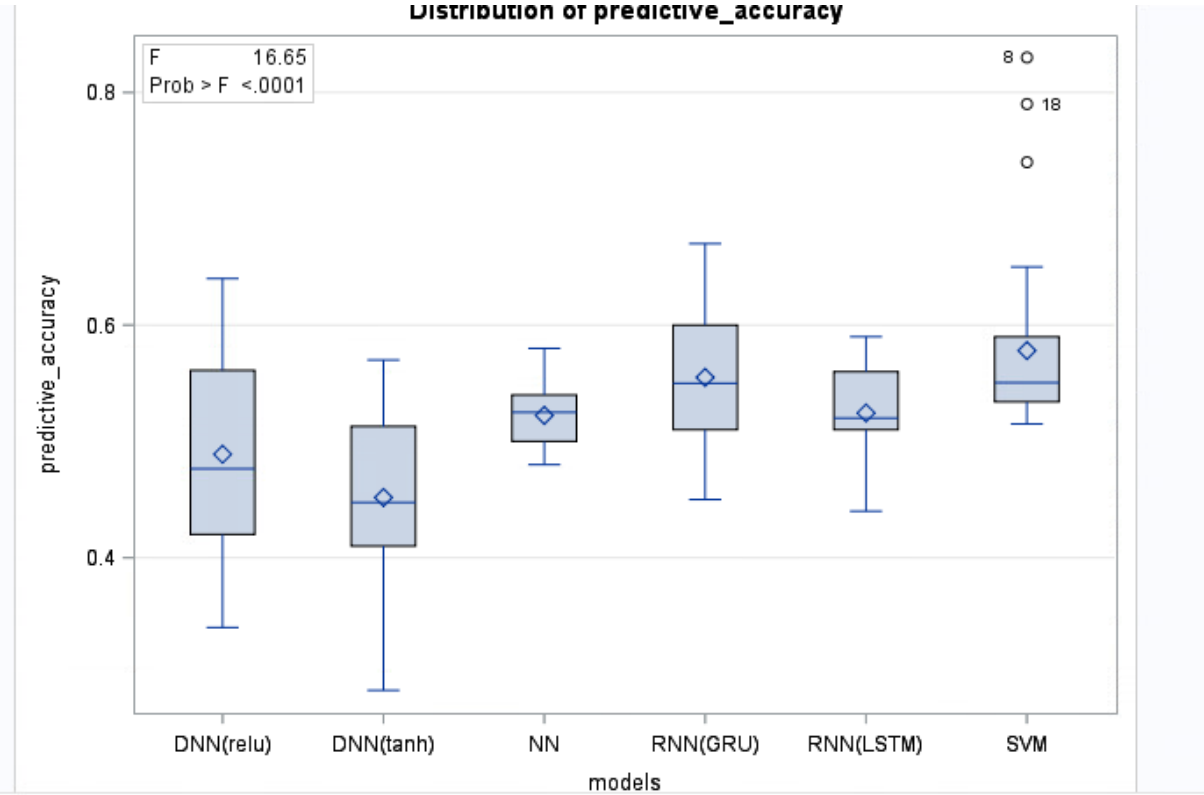

Figure 9: Box plot of accuracy for each all six model specifications (SVM, NN, DNN (RELU), DNN (TANH), RNN (LSTM), RNN (GRU)), using hourly data

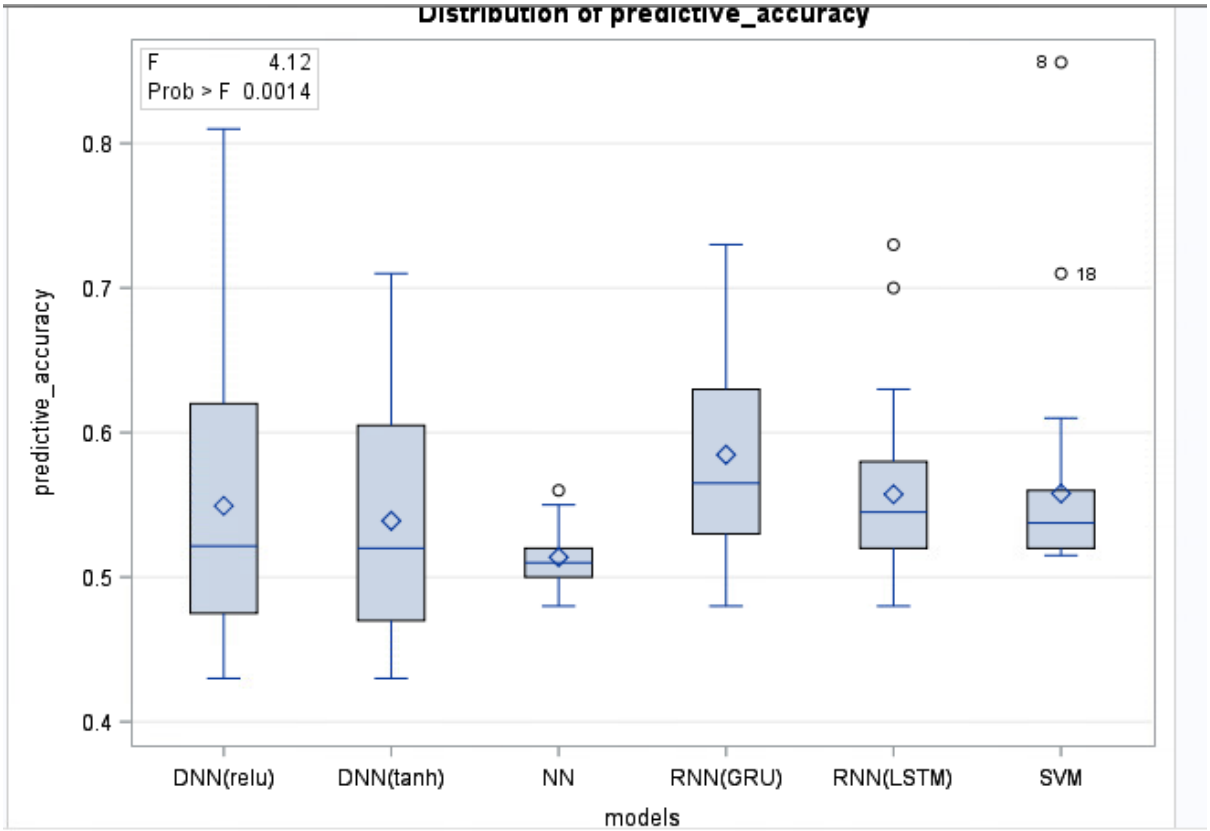

Figure 10: Box plot of accuracy for each of the six model specifications (SVM, NN, DNN (RELU), DNN (TANH), RNN (LSTM), RNN (GRU)) using minute data

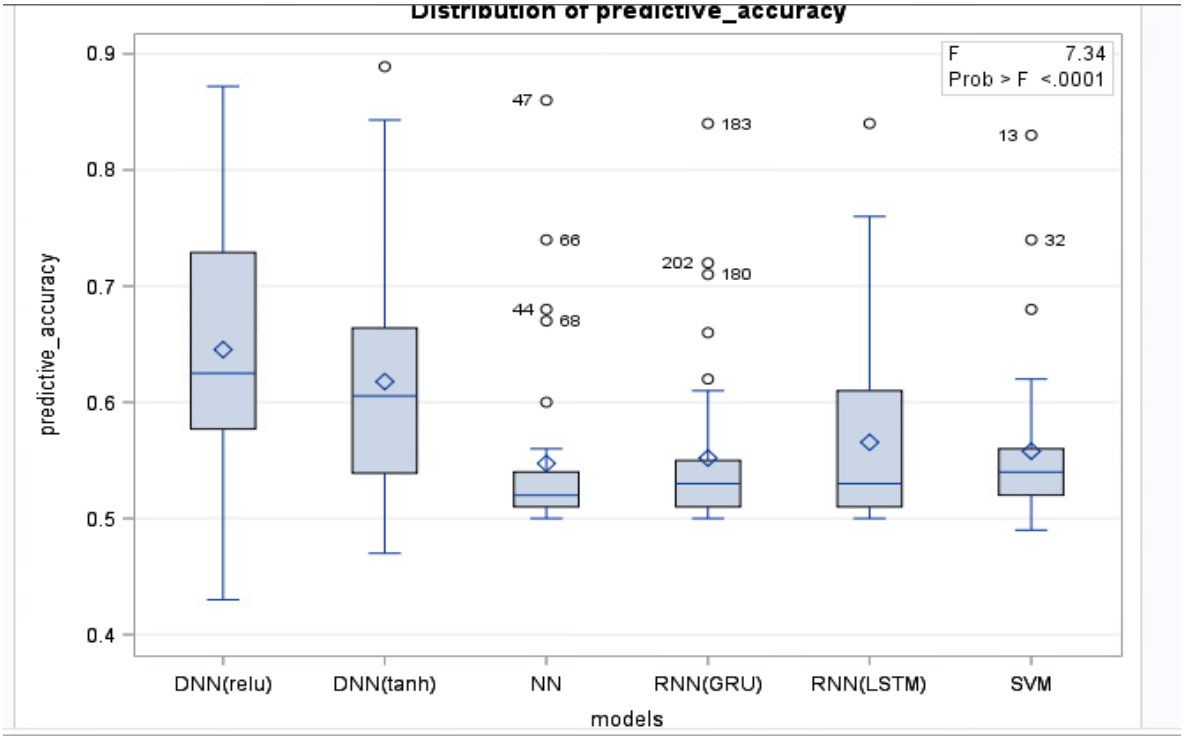

Figure 11: Box plot of accuracy all six model (SVM, NN, DNN (RELU), DNN (TANH), RNN (LSTM), RNN (GRU)), using tick data