Data Science in Finance and Economics

DSFE, 2(2): 147–162.
DOI: 10.3934/DSFE.2022007
Received: 12 March 2022
Revised: 23 April 2022
Accepted: 28 April 2022
Published: 26 May 2022

http://www.aimspress.com/journal/dsfe

Research article

Multi-level stacking of LSTM recurrent models for predicting stock-market indices

Fatima Tfaily[1] and Mohamad M. Fouad[2,*]

[1] Department of Business and Finance, BoxHill-Kuwait College, Abu-Halifa 54600, Kuwait
[2] Department of Computer Engineering, Mansoura University, Mansoura 35516, Egypt

* **Correspondence:** Email: mfouad@ieee.org.

Abstract: The ability to predict stock-market indices is important to investors and financial decision-makers. However, the uncertainty of available information makes accurate prediction extremely challenging. In this work, we propose and validate a multi-level stacking model of long short-term memory (LSTM) units for the short-term prediction of stock-index closing prices. The proposed machine-learning model is trained using historical data to predict next-day closing prices. The first layer of the multi-level stacked structure contains an ensemble of recurrent LSTM models that receives time-series data of historic opening, closing, high and low prices for current and previous days and outputs predictions about the next day's closing prices. The second and third layers consist of stacked multi-layer perceptron meta-models. We validated the new model on two stock indices, demonstrating its advantages over single-LSTM models. We also compared its performance against several extant statistical and machine-learning models on a subset of Standard & Poor's 500 index data between 2000 and 2016 using correlation and statistical metrics.

Keywords: predicting stock prices; long short-term memory; multi-level stacking; financial prediction; ensemble learning

JEL Codes: G17, C53, D83

1 Yahoo Finance S&P 500 (https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC)
2 Yahoo Finance NASDAQ Composite (https://finance.yahoo.com/quote/%5EIXIC?p=%5EIXIC)

1. Introduction

Forecasting the next day's stock-market index (SMI) prices is crucial to investor success, and the research being applied to this problem is extensive. In financial fields, technical analyses exploit historical price data to make decisions, assuming that previous and current behaviours impact a stock's value evolution (Cervelló-Royo et al., 2015; Vargas et al., 2017). However, obtaining consistent forecasting accuracy is extremely challenging, owing to the non-linearity and uncertainty of future value. Thus, good prediction requires the elaboration of reliable methods to find stochastic relationships between present and future stock values using extant information.

Recently, machine-learning (ML) methods have shown great results in making statistically driven non-linear predictions based on the availability of copious historical data (Vargas et al., 2017; Li et al., 2021; Banerjee et al. 2021; Hassan et al., 2021). Moreover, ML models consistently outperform regression methods, especially with forecasting (Hassan et al., 2021; Wang et al., 2021; Weng et al., 2018). However, as implied, ML requires large amounts of historical data so that the stochastic relationships between changing variables can be "learned" (Jiao et al., 2017; Bahrammirzaee et al., 2010). As we will see, the availability of such dada is rare.

Numerous models have been used to predict SMI closing prices, including long short-term memories (LSTMs), gated recurrent units (GRUs), support vector regressors (SVRs), multi-layer perceptrons (MLPs), *k*-nearest-neighbour (KNN) algorithms (Li et al., 2021; Banerjee et al., 2021; Hassan et al., 2021; Wang et al., 2021; Weng et al., 2018; Bahrammirzaee et al., 2010; Gao et al., 2017; Heaton et al., 2017) and several others. For short-term closing-price prediction (Banerjee et al., 2020), the authors applied MLP models, LSTM, gated recurrent units (GRU), and bidirectional long short-term memory (BLSTM) to predict stock prices over short terms. They also compared the overall performance of their approaches against classic artificial neural networks (ANNs). They have addressed the problem of predicting the equity "close prices" for a short period. The authors reported that the prediction's average error, shown by MLP, LSTM, GRU, BLSTM, and BGRU models, varied between 9 and 10%. In (Weng et al., 2018), a financial expert system was designed, comprising a knowledge base module that captures historical stock prices and technical indicators, calculates sentiment scores of published news articles and mines trend data from Google searches. This information is transferred to an ML module to make informed predictions. The authors of (Jiao et al., 2017) compared the performance of four contemporary classification algorithms (i.e., random forest, gradient boosted trees, ANNs and logistic regression models) in predicting the movements of 463 Standard & Poor's (S&P) 500 stocks. They demonstrated that data quality and accurate feature selection are crucial to the success of SMI prediction and that immediate past information contains the most pertinent signals. In (Bahrammirzaee et al., 2010), ML methods outperformed statistical regression models when dealing with numerous financial problems, including portfolio management, financial prediction and credit evaluation. In (Gao et al., 2017), the authors proposed a recurrent neural network (RNN)-based LSTM model to predict stock closing prices. It applies optimisation techniques to compute the adaptive learning rates for each parameter in the neural models. The authors used as input all the available parameters in each data record of the training dataset, such as opening price (OP), low price (LO), high price (HI), closing price (CL), adjusted price (AD) and volume-of-trading (VO) metrics, showing superior performance over other ML methods. In (Heaton et al., 2017), the authors applied deep-learning techniques to forecast the directional daily movements of the S&P 500 index using technical indicators from financial news. The method was trained to recognise complex patterns and relationships among variables using RNNs and convolutional neural networks (CNNs) from the natural language processing field. The experiments reported in (Heaton et al., 2017) showed that a CNN model outperformed an RNN in detecting text semantics. However, RNNs are still better at recognising contextual information and modelling temporal features of SMI prediction.

Recently, other researchers have forwarded hybridised models consisting of two or more ML models (Chen et al., 2017; Lu et al., 2013). In (Chen et al., 2017), the authors proposed a hybridised framework consisting of a feature-weighted KNN model with a feature-weighted SVM to predict index values. Their method assigns weights to each feature based on its classification importance, computed from the information value gained. It then feeds historical data to the feature-weighted KNN to estimate prices. Their model was validated and tested on two Chinese. SMIs at different temporal horizons and resolutions. In (Lu et al., 2013), the authors suggested a hybrid model consisting of a nonlinear independent component analysis (NLICA) module, an SVR and a particle swarm optimisation (PSO) algorithm. In the first step, NLICA is applied to handle the nonlinearity expected in SMI data records so that the collection of features can be facilitated. Then, these features are used as inputs to train the SVR model, whose parameters are optimised by the PSO algorithm. The authors evaluated the performance of their approach in predicting the closing prices of the Taiwan Capitalisation Weighted Stock Index and the Shanghai Stock Exchange Composite Index. To reduce the dimensionality of the input vectors used for prediction, the authors of (Zhong et al., 2017) applied principal component analysis (PCA) and kernel-based PCA (KPCA) to restructure the original data. The authors then used an ANN model on the pre-processed data records to predict the daily direction of future market returns.

Practically, stability and consistency are two key challenges faced by researchers when building reliable prediction models, especially when dealing with time-series data (Li et al., 2021; Al-Hajj et al., 2018). A particular challenge for ML systems is the limited availability of datasets for training and validation. When lacking sufficient data, weak learners are often substituted, but they rarely produce operationally sound predictions. One method of remedying this weakness is to combine several basic weak-learner models so that their outcomes can be consolidated to produce a higher-resolution decision. For this purpose, stacking learning techniques have been developed. They consist of an ML meta-model that combines the outcomes of an ensemble of individual base learners (Al-Hajj et al., 2019; Sagi et al., 2018). This "ensemble learning" technique aggregates complementary outcomes from a set of weak learners to produce a final decision tuned to the input signal. Recently, several ensemble-learning approaches have been proposed to improve both short- and long-term time-series data predictions (Sagi et al., 2018; Hochreiter et al., 1997; Ren et al., 2015; Briscoe et al., 2011). Notably, not all of these methods use ML stacking for aggregation.

In this paper, we introduce an ensemble method that uses a multiple-level stacking technique of recurrent models to predict SMI closing prices using historical time-series data. The basic stacked models are LSTM variants (Hochreiter et al., 1997) and their predictions are combined using two MLP layers, also stacked. The base predictors are designed to predict closing prices one day ahead, using the index data from n prior days. In this paper, we demonstrate the advantages of dividing index information across several base models and feeding their decisions to a post-prediction stage. Our experiments illustrate the superiority of the suggested strategy against global systems that receive and process all SMI information using a single model. We compare the performance of our suggested approach to several statistical and ML models (e.g., a moving average (MA) predictor, a regression SVM and more). These models were designed and implemented by another research group on a subset of S&P 500[1] index data from 2000 to 2016 (Gao et al., 2017). The suggested method is also validated and tested against the Nasdaq[2] Composite index. For each index, the multi-level stacking approach shows good improvement with respect to the individual predictors of the ensemble.

The remainder of this paper is organised as follows. In Section 2, we introduce the ensemble methods and describe their advantages. In Section 3, we describe the MLP and LSTM methods. In Section 4, we explain our suggested multi-level stacking approach and the components thereof. The benchmark dataset and the preparation of the data records are illustrated in Section 5. In Section 6, we describe our experimental settings and their results and analyses. Finally, we conclude our study and draw future perspectives in Section 7.

2. Ensemble method

An ML ensemble model combines multiple base learners trained to solve specific sub-problems of classification or regression. The outputs of the base models are then aggregated to produce final, consolidated prediction and classification responses. This approach decreases output variance and enhances overall performance (Al-Hajj et al., 2019; Sagi et al., 2018; Ren et al., 2016). Generally, constructing an ensemble technique requires a good understanding of the complementarity of both the aggregated metrics and the base-learner strengths. Figure 1 depicts a parallel aggregation ensemble technique.

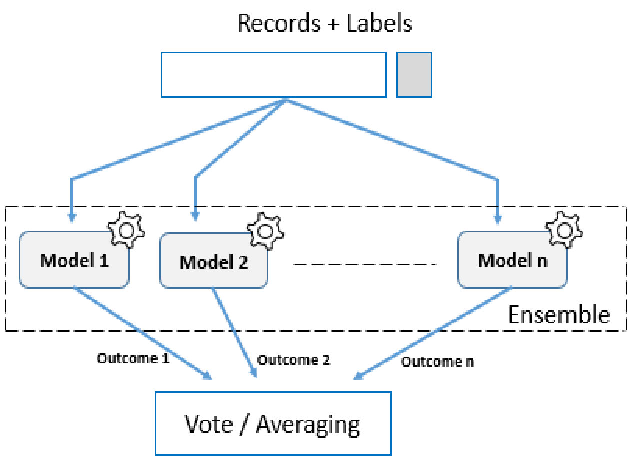


Figure 1. General architecture of a parallel ensemble using *n* base models.

The consolidation of outputs can use an averaging technique for regression or a voting technique for classification (Dietterich et al., 2000; Liu et al., 2021), depending on the problem being addressed. The choice of base models depends on many factors, such as the dimensionality of the problem space, the distribution hypothesis and the availability of training data. The base models may therefore receive similar or dissimilar features from a subset of the training data. Thus, they may incur high biases and variances individually because of their low or high degrees of model freedom. Notably, ensemble techniques have been found to decrease both bias and variance. The ensemble methods show good enhancement when the base models have considerable complementary information (Dietterich et al., 2000; Liu et al., 2021).

A stacking technique trains a machine learning meta-model to smartly generate a final decision based on those of the base models. The base models in a staking ensemble may receive similar or different features from a subset of training data. A stacking algorithm includes the following steps:

- Split the training data into two subsets: one for training the base models and another to validate their outputs and train the meta-model that combines the base models' outputs.
- Use the data of the first subset to train *n* selected individual base models.
- Each base model makes predictions for each observation in the validation subset.
- These predictions are used to train the meta-model for combination.

Figure 2 depicts the general structure of the stacking ensemble technique.

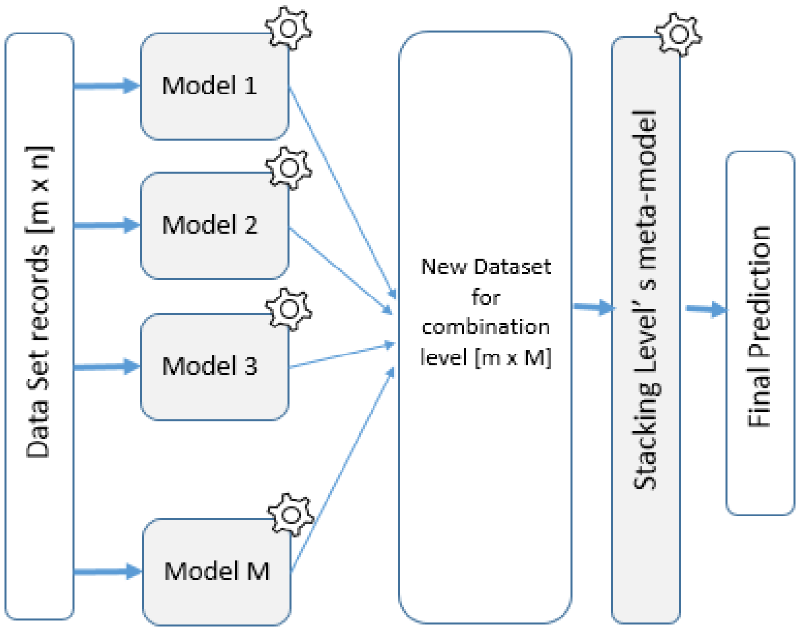

Data Set records [m x n]
Model 1
Model 2
Model 3
Model M
New Dataset for combination level [m x M]
Stacking Level's meta-model
Final Prediction

Figure 2. Multi-level stacking: The outputs of individual models are combined with a meta-model to produce final predictions.

A multi-level stacking model involves multiple subsequent layers of machine learning meta-models of combination. The individual base models return their responses to their respective input data vectors. The first stacking layer produces a new dataset containing the individual base-model outputs, and the second layer takes those outputs as input to a meta-model that outputs the final prediction(s).

3. MLP and LSTM

MLPs consist of sequentially interconnected layers of neurons (Haykin, 2011), which are individual units of computation (i.e., perceptrons). The output of a perceptron is determined by its activation function and the value of the input vector. A basic MLP uses three consecutive layers of perceptrons. The input layer collects input signals, the hidden layer receives the output of the first layer, and the output layer produces the final response. A feed-forward MLP consists of an arbitrary number of hidden layers that support complex calculations and improve the approximation of continuous functions. Generally, a set of labelled input-target pairs are used for training, as parameters are adjusted using back-propagation gradient descent optimisation (Haykin, 2011).

LSTMs are gated models designed to handle the temporal contexts of time-series data (Hochreiter et al., 1997). Their main components are depicted in Figure 3. The LSTM's input gate receives a combination of input signals and a hidden state. It applies a purposeful arrangement of internal gates (e.g., forget, output and arithmetic) that maintains (remembers) only the most important (discriminant) information to prevent the vanishing and exploding gradient problems that arise when back-propagation is used with gradient descent optimisation (Hochreiter et al., 1997; Haykin, 2011).

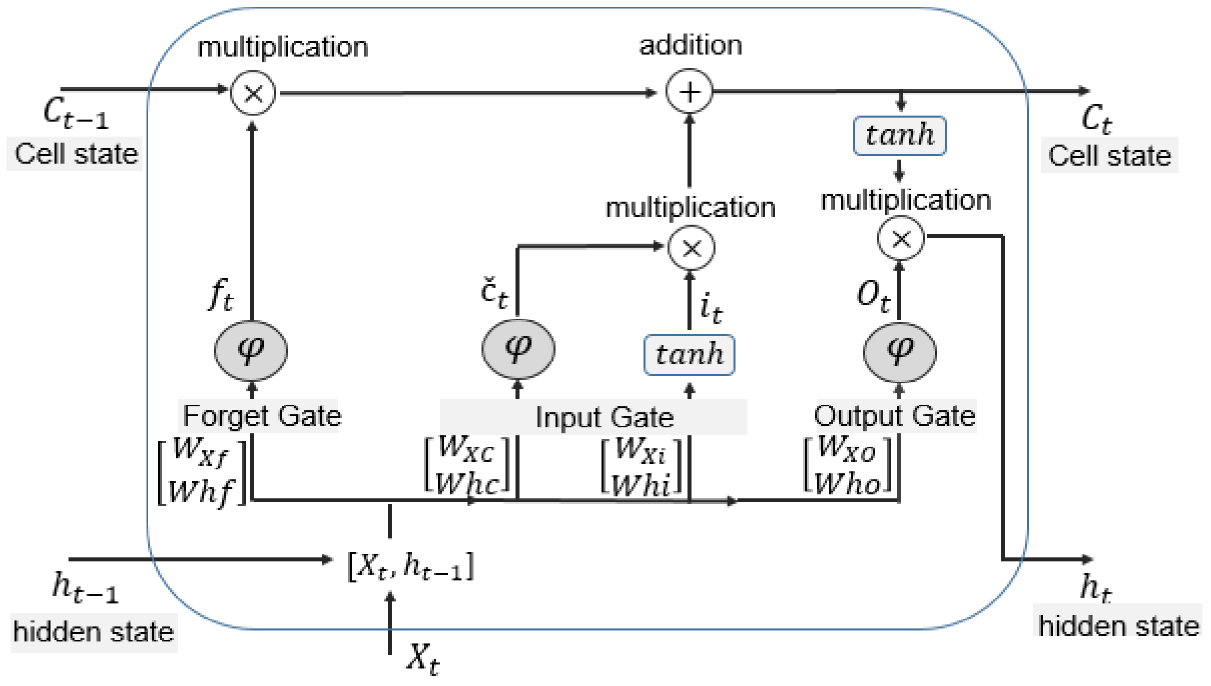


Figure 3. General long short-term memory structure with input vector X_t and output value h_t.

An LSTM cell receives a signal vector X_t at instant t. The hidden state, h_{t-1}, represents the cell's output at instant $t-1$, and the cell state, c_{t-1}, output at $t-1$. The current hidden state, h_t, represents the LSTM's output at t, which becomes the next input alongside the cell state, c_t. The LSTM unit includes five sets of weights, defined as follows.

$$\text{Weights of the forget gate: } \begin{bmatrix} W_{xf} \\ W_{hf} \end{bmatrix} \in R^{(D+H)*H}, b_f \in R^H \tag{1}$$

$$\text{Weights of the input gate: } \begin{bmatrix} W_{xi} \\ W_{hi} \end{bmatrix} \in R^{(D+H)*H}, b_i \in R^H \tag{2}$$

$$\text{Weights of the output gate: } \begin{bmatrix} W_{xo} \\ W_{ho} \end{bmatrix} \in R^{(D+H)*H}, b_o \in R^H \tag{3}$$

$$\text{Weights of the cell: } \begin{bmatrix} W_{xc} \\ W_{hc} \end{bmatrix} \in R^{(D+H)*H}, b_c \in R^H \tag{4}$$

$$\text{Weights of the output layer: } W_{hk} \in R^{H*K}, b_k \in R^K \tag{5}$$

Variable ***D*** represents the dimensionality of the input vector, ***K*** represents the number of classes defined for a classification problem, and ***H*** represents the number of LSTM units in a layer. ***W(·)*** is the weight matrix, and ***b(·)*** denotes a bias vector that impacts the transformation of a particular gate's state. For a given input vector, its forward calculation is defined as follows.

The output of the forget gate is computed with Equation (6), the output of the input gate is computed with Equation (7), and the cell state is determined using Equations (8) and (9) in turn. Lastly, the output of the LSTM cell is found using Equations (10) and (11).

$$f_t = \varphi(\begin{bmatrix} W_{xf} \\ W_{hf} \end{bmatrix}^T [X_t, h_{t-1}] + b_f) \tag{6}$$

$$i_t = \varphi(\begin{bmatrix} W_{xi} \\ W_{hi} \end{bmatrix}^T [X_t, h_{t-1}] + b_i) \tag{7}$$

$$\check{c}_t = \tanh(\begin{bmatrix} W_{xc} \\ W_{hc} \end{bmatrix}^T [X_t, h_{t-1}] + b_c) \tag{8}$$

$$c_t = f_t \circ c_{t-1} + t \circ \check{c}_t \tag{9}$$

$$o_t = \varphi(\begin{bmatrix} W_{xo} \\ W_{ho} \end{bmatrix}^T [X_t, h_{t-1}] + b_o) \tag{10}$$

$$h_t = o_t \circ \tanh(c_t) \tag{11}$$

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

The outputs of the forget, input and output gates and cell memory are denoted with $\boldsymbol{f_t}$, $\boldsymbol{i_t}$, $\boldsymbol{o_t}$ and, $\check{c}_t$, respectively. The sigmoid function, $\boldsymbol{\varphi}$, is defined by Equation (12) as the activation function, which denotes the element-wise product of two vectors to determine the cell response. The product of the LSTM unit includes its output value, h_t(), and its state memory, c_t, which are applied iteratively to the LSTM unit at the next time step. More details about LSTMs and other gated models can be found in (Hochreiter et al., 1997).

4. Multi-level stacking model

The core multi-level stacking model uses three consecutive layers. The first consists of ***M*** LSTM units, each receiving successive values related to SMI closing prices of ***N*** previous days, including day ***d*** (current), to predict tomorrow's (d+1) SMI closing. The second layer contains ***S*** MLP meta-models, each collecting the outputs of the LSTM base models. The third layer contains a single MLP meta-model that combines the outputs of the meta-models. Note the clear distinction of our applied model in Figure 4 from the one previously described, in that ours applies an additional meta-model layer of multiple MLPs.

In our case study, we validate our approach on an S&P 500 data subset. Independent variables are analysed to determine the best inputs to the base prediction models and their LSTM configurations. All LSTMs use the same activation function with the same number of hidden layers. The back-propagation-through-time (BPTT) algorithm is applied for deep learning (Hochreiter et al., 1997), and all MLPs in the sub-meta-model layer use the same number of internal hidden layers, but with different numbers of perceptrons and activation functions. The final meta-model layer consists of one hidden layer and one perceptron to output the final prediction.

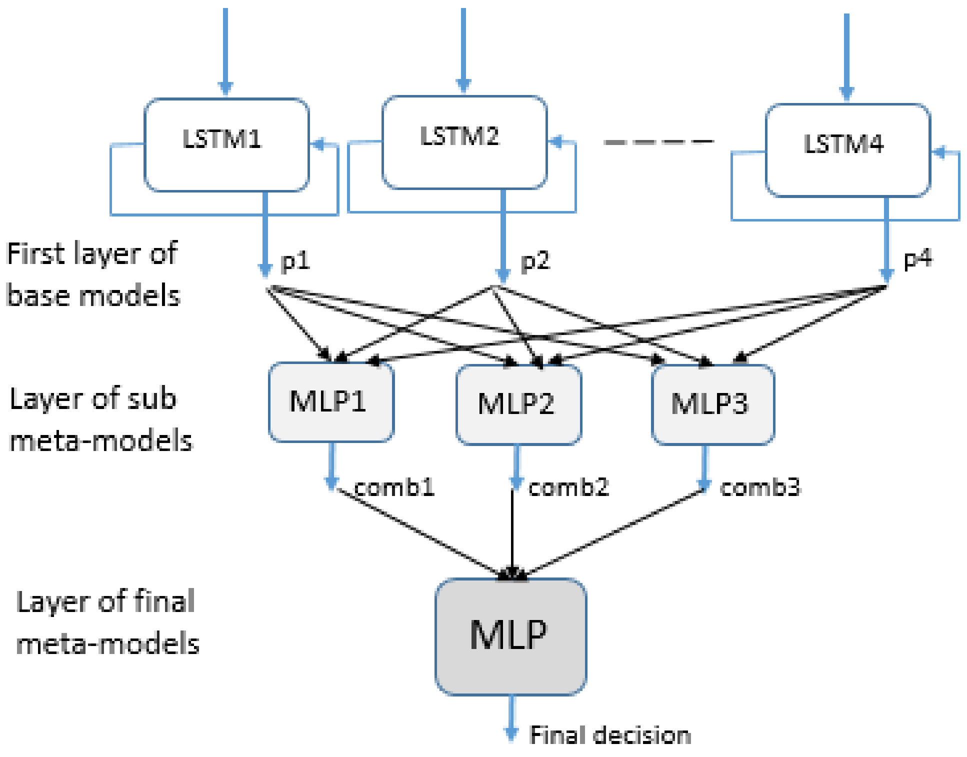


Figure 4. Each model in the first stacking layer combines the responses of the individual LSTM models. The combinations are forwarded to the next stacking layer.

In this work, we implement and validate our stacked base-learner LSTM and MLP model using the Scikit-Learn toolkit (Pedregosa et al., 2011).

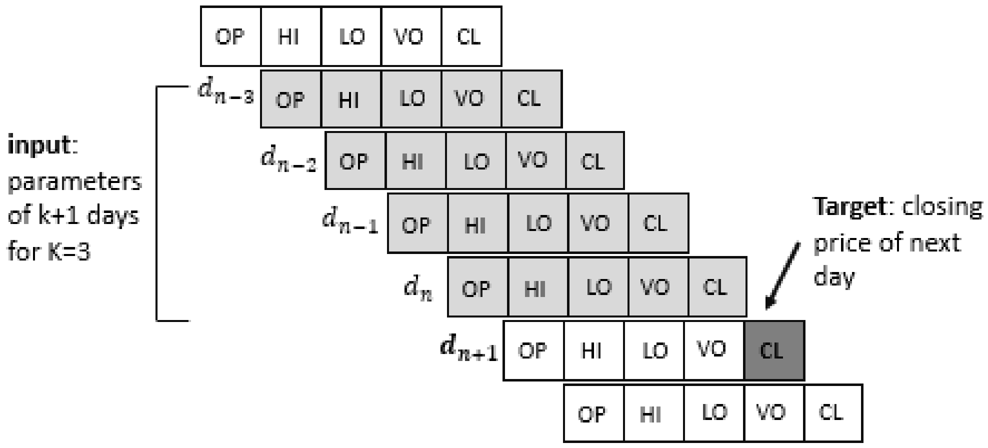
5. Benchmark datasets

5.1. SMIs

Training and validation experiments were conducted on a subset of the S&P 500 index extracted from Yahoo Finance, containing 505 common stocks maintained by 500 publicly traded companies covering about 80% of the U.S. equity market per capitalisation (Jiao et al., 2017). Daily OP, LO, HI, CL, AD and VO values between 3 January 2000 and 10 November 2016 (4,243 trading days), were analysed. We validated the advantages of our approach on a subset of the Nasdaq Composite index covering the period between 1 January 2004 and 31 December 2019 using the same horizon of one day ahead.

5.2. *Input sequence preparation*

A sliding-window method (Figure 5) was used to pre-process the sequential input vectors for the LSTM base models. Data of k days prior to day d_n were used to predict SMI closing prices on day d_{n+1}. Consequently, the input sequence held records representing days d_n, d_{n-1}, d_{n-2},…, d_{n-k}. The output target was the closing price on day d_{n+1}.



input: parameters of k+1 days for K=3
d_{n-3}
d_{n-2}
d_{n-1}
d_n
d_{n+1}
OP
HI
LO
VO
CL
Target: closing price of next day

Figure 5. Sliding-window pre-processing of the time-series data.

Each LSTM in the first layer received one or a combination of two of the six parameter values listed above. For instance, one LSTM tooks the closing and opening prices of the previous k days, while another took closing and high prices. In this work, we inspected the impact of each combination on LSTM performance, and the results are provided in Section 6.2.

6. Experiments and results

6.1. *Statistical evaluation metrics*

To evaluate our model performance, we applied root mean-square error (RMSE), mean bias error (MBE), mean absolute error (MAE), mean absolute percentage error (MAPE) and average MAPE (AMAPE) (Heaton et al., 2017; Al-Hajj et al., 2019). RMSE is commonly used to assess regression performance and is sensitive to large approximation errors. Therefore, it is useful when large estimation faults lead to higher costs than minor faults. The RMSE was computed using the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\mathrm{M_{p,i}}-M_i)^2}{n}}. \tag{13}$$

where $\mathrm{M_{p,I}}$ denotes the estimated value, and M_i is the exact value.

MAE is the average of prediction errors, not counting their directions, and is evaluated as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\mathrm{M}_{\mathrm{p,i}} - M_i|. \tag{14}$$

MBE computes the average bias of predictions, tallying under- and overestimations and allowing them to cancel one another. It is defined as follows:

$$MBE = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{M}_{\mathrm{p,i}} - M_i). \tag{15}$$

MAPE evaluates the relative error between the predicted and measured values, as depicted in Equation (16):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{\mathrm{M}_{\mathrm{p,i}} - M_i}{M_i} \right|. \tag{16}$$

AMAPE calculates the average MAPE between the estimated and measured values. It is computed by the following equation:

$$AMAPE = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{\mathrm{M}_{\mathrm{p,i}} - M_i}{\frac{1}{n}\sum_{i=1}^{n} M_i} \right|. \tag{17}$$

6.2. Experiments

In our experiments, we divided the examined dataset into three parts, as follows:

1. One training subset for the base predictors,
2. One validation subset to train the array of meta-models in the second layer and the one in the output layer,
3. One testing subset to evaluate the LSTMs and MLPs. This latest subset was not seen by any of the individual or meta-models during the training and validation phases.

The ensemble of base STMs had similar structures and used the same activation function. Furthermore, the same BPTT algorithm and its parameters were used to train the LSTM-based predictors (Hochreiter et al., 1997) . Each predictor received a sequence of SMI closing prices with a probably of one of the other available parameters' values of the stock for a day *d* and the *k* preceding days (*d-1, d-2, ..., d-k*). The resultant LSTM models took the following inputs for the current day *d* and the *k* preceding days.

- LSTM-1: takes only the stock's closing prices (CL) for the current day *d* and the *k* preceding days.
- LSTM-2: takes the stock's closing prices coupled with the opening prices (CL, OP) of the same stock for the current day *d* and the *k* preceding days.
- LSTM-3: takes the stock's closing prices coupled with the highest prices (CL, HI) of the same stock for the current day *d* and the *k* preceding days.
- LSTM-4: takes the stock's closing prices coupled with the lowest prices (CL, LO) of the same stock for the current day *d* and the *k* preceding days.

This combination of parameters was found to be the best at providing good overall prediction performance. Moreover, we trained another LSTM (i.e., LSTM-5) to receive the sequence of all parameters to simulate a global approach. It was trained using the same BPTT algorithm. The objective was to compare the ensemble approach of LSTMs 1–4 with the global approach of LSTM-5. Table 1 provides an assessment of the ensemble approach vs. the global approach.

Table 1. Prediction performances of individual and meta-models for 200 days of S&P 500 data records.

Model	*RMSE*	*MAE*	*MBE*	*MAPE*	*AMAPE*	*R*
LSTM-1	18.2596	13.9685	8.5051	0.6714	0.6661	0.9808
LSTM-2	21.5456	18.1596	−14.9989	0.8892	0.8945	0.9804
LSTM-3	16.6898	12.1862	3.6156	0.5737	0.5667	0.9801
LSTM-4	19.1559	14.9226	−9.9370	0.7252	0.7179	0.9800
MLP-1	16.6037	12.0364	−4.1061	0.5847	0.5774	0.9807
MLP-2	16.6846	12.1971	−4.5574	0.5920	0.5853	0.9807
MLP-3	16.5341	11.8976	−3.0420	0.5781	0.5705	0.9804
LSTM-5	21.7336	17.7976	−14.2938	0.8658	0.8580	0.9802
MLP-4	**16.0964**	**11.4979**	**−1.0135**	**0.5300**	**0.5227**	**0.9808**
Multi-level stacking						

The second layer of the global stacking model contained three MLP models for stacking. Each included four inputs in its first layer and one hidden layer. The MLPs were varied by one or more of the following parameters: (1) the number of hidden neurons in the hidden layer, (2) the activation functions in the hidden and output layers, and (3) the learning rate type (i.e., constant or adaptive) (Pedregosa et al., 2011).

In some of the MLP models in this layer, we used the adaptive moment estimation optimizer (Adam*)*. The resultant configuration was as follows,

- MLP-1: Nine hidden neurons, identity activation and constant learning rate with Adam optimisation (Pedregosa et al., 2011)
- MLP-2: 10 hidden neurons, identity activation and constant learning rate with Adam optimisation
- MLP-3: 10 hidden neurons, identity activation and adaptive learning with stochastic gradient descent optimisation (Pedregosa et al., 2011).

The meta-model in the final layer consisted of one MLP with three input units in the first layer. Its hidden layer included six hidden perceptrons using the identity activation function and a constant learning rate with Adam optimisation. The output layer contained a single output to represent the final prediction. The parameters were selected using five-fold cross-validation (Bergstra et al., 2012).

The overall results showed that the stacking-model results were superior to any single LSTM or MLP from internal phases. The MLP with the highest performance was MLP-3. Table 1 shows that MLP-3 decreased the average RMSEs of the base predictors (total 18.9127) by around 12.57%, according to Equation (17):

$$Dec\ in\ RMSE\ = 12.57\% = \frac{18.9127-16.5341}{18.9127} * 100. \tag{18}$$

The third and fourth columns of Table 1 show good MAE and MBE scores for MLP-3 over the best-performing LSTM-3. The LSTM outputs provided complementary information, as their over- and underestimations cancelled out. This phenomenon is reflected by the alternating signs of the four main LSTM MBEs. The meta-model in the final layer outperformed all other MLPs and LSTMs. The scores in Table 1 show that MLP-4 decreased the average RMSE scores related to individual models (total 18.9127) by around 14.89%, as shown in Equation (18):

$$Dec\ in\ RMSE\ = 14.89\% = \frac{18.9127 - 16.0964}{18.9127} * 100. \tag{19}$$

The overall multi-stacking model provided significantly better MBE scores than those of the constituent parts. Negative MBEs in the LSTMs reflect underestimated SMI closing prices. However, the relatively small MBEs in the MLPs indicate that they have good short-term performance. The results in Table 1 also illustrate the advantages of our stacked approach against LSTM-5, which received all SMI parameters. The stacked version outperformed LSTM-5 in nearly all evaluation metrics.

Figure 6 illustrates the LSTM daily predictions of SMI closing prices of the next day's S&P 500 values. The best was LSTM-3. For clarity, we show predictions for two consecutive months of September and October 2016.

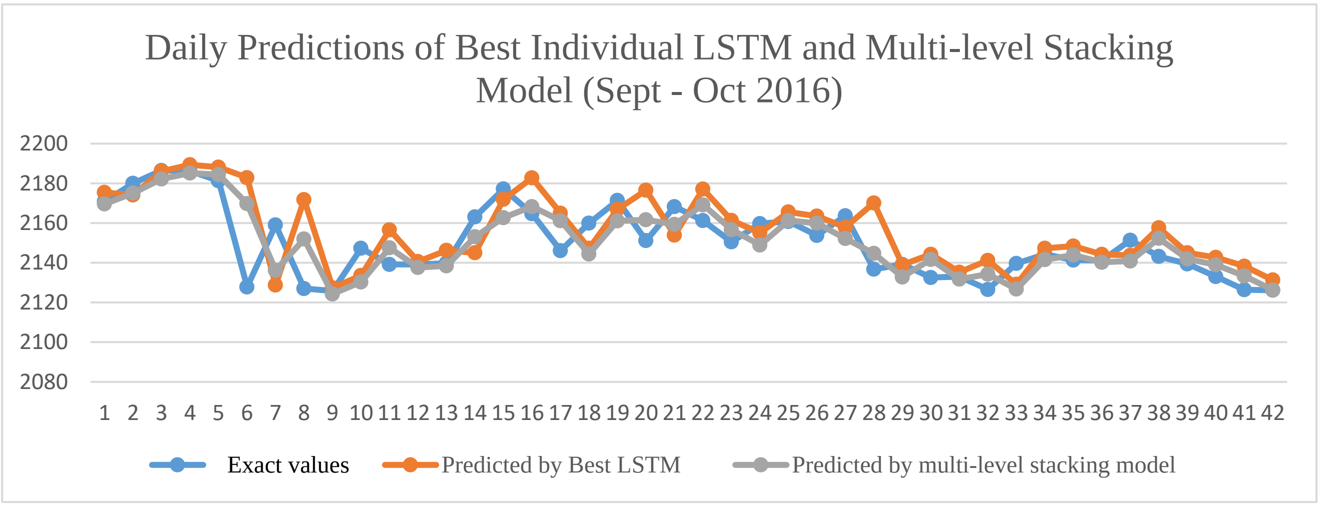


Figure 6. Daily predictions of next day's S&P 500 closing prices provided by the best LSTM and MLP for September-October 2016.

Figure 7 presents the monthly average prediction errors of the next-day' SMI closing prices, as provided by the best LSTM model (LSTM-3), the best MLPs meta-model in the first stacking layer, namely the MLP-3, and the meta-model in the third layer. For clarity, we show the monthly average prediction errors for April-September 2016, inclusive.

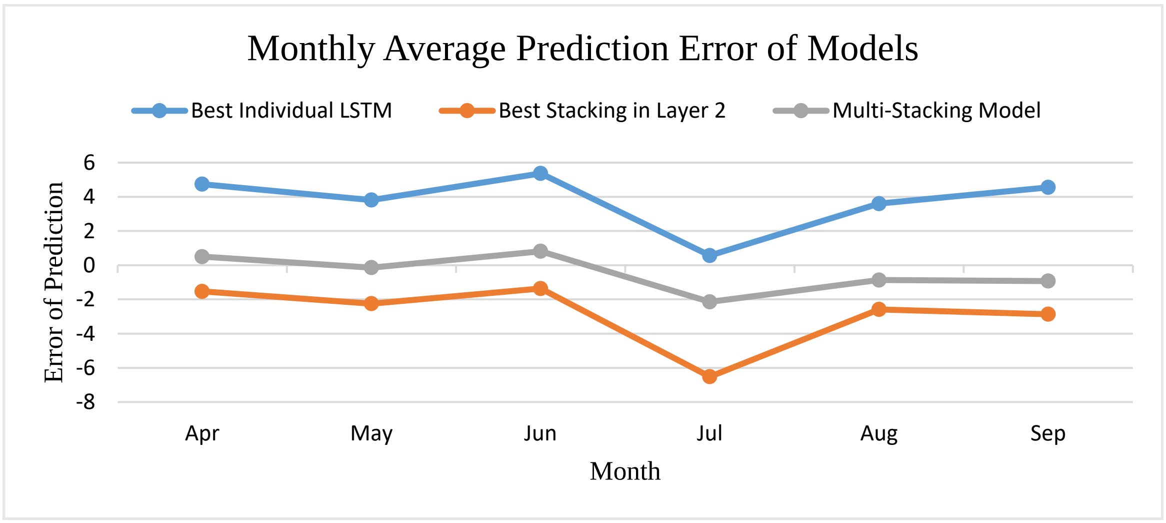


Figure 7. Monthly average prediction errors of SMI closing prices provided by the best LSTM and MLPs for April-September, 2016.

Figure 7 shows that the proposed multi-level stacking model outperformed the best internal models in terms of the least prediction error deviation, which varied between −2 and 2. The related chart shows more stability than the other two charts.

Table 2 illustrates the advantages of our approach when predicting Nasdaq Composite SMI closing prices. The multi-level stacking model outperformed the best internal models on nearly all evaluation metrics.

Table 2. Prediction performance of individual and meta-models for 200 days of Nasdaq records between 01/2004 and 12/2019.

Model	*RMSE*	*MAE*	*MBE*	*MAPE*	*AMAPE*	*R*
LSTM-1	103.5955	89.2724	−70.4078	1.1130	1.1102	0.9745
LSTM-2	75.8403	57.6078	−4.2171	0.7178	0.7105	0.9744
LSTM-3	78.2722	58.7863	10.1621	0.7308	0.7238	0.9731
LSTM-4	76.5654	58.5826	−10.4901	0.7322	0.7231	0.9744
MLP-1	76.3491	58.4035	−13.8733	0.7292	0.7212	0.9748
MLP-2	76.3317	58.4016	−14.8544	0.7299	0.7213	0.9750
MLP-3	77.8307	60.5109	−21.0190	0.7538	0.7479	0.9750
MLP-4	**74.4014**	**55.7698**	**−0.6689**	**0.6950**	**0.6876**	**0.9753**
Multi-level stacking						

Finally, to validate the utility of our multi-stacking approach, we compared its scores to those obtained by other statistical and ML models applied to the same S&P 500 dataset, covering the same period (Gao et al., 2017). These models included SVM, MA, exponential MA and stand-alone LSTM predictors. Table 3 compares the MAE, RMSE, MAPE and AMAPE scores of the comparison models (Gao et al., 2017) to those of LSTM-3 and our proposed multi-level stacking model.

Table 3. Comparing the performance of the proposed approach against other prediction models introduced in (Gao et al., 2017) on 200 days of S&P 500 data records.

Model	*MAE*	*RMSE*	*MAPE*	*AMAPE*
Moving average	25.6319	32.3547	1.2415	1.2273
Exponential moving Average	14.7985	19.4146	0.7145	0.7086
Support vector machine	12.8311	17.5105	0.6229	0.6152
Long short-term memory (Weng et al., 2018)	12.2613	16.7404	0.5940	0.5871
LSTM-3	12.1862	16.6898	0.5737	0.5667
LSTM multi-level stacking	**11.4979**	**16.0964**	**0.5300**	**0.5227**

In summary, for the considered prediction problem, we found that it was most efficient to divide the feature values from the SMI dataset into discriminant subsets and to input them in a complementary fashion into individual base models ensembled to provide output to consolidating meta-model layers. Doing so well minimizes hyperparameter adjustments during training and renders available datasets sufficient for training, validating, and testing the overall recurrent model.

The scores in the last row of Table 3 show that the proposed approach has well exploited the complementary outputs of base and sub meta-models to outperform all the described models in (Gao et al., 2017) for almost all the reported metrics.

7. Conclusions

In this work, we addressed the problem of short-term predicting of SMI closing prices for market investment decisions. We introduced a novel framework that combines the capabilities of LSTM models in predicting time-series data results using MLP models to stack their outputs. The base predictors were designed to predict SMI closing prices using their information from the previous n days, and the final decisions of the individual LSTM models were combined through two MLP stacking layers to produce a single prediction output.

We demonstrated the advantages of stacking on prediction accuracy, as the overall model outperformed all internal predictors by combining their complementary decisions. Furthermore, the suggested strategy showed good utility across multiple SMIs and outperformed other global systems specifically designed to ingest the same data to make the same predictions.

Applying the suggested method to new financial SMI datasets with different time horizons and resolutions is a problem for future study. Another opportunity would be to apply the same stacking technique using new gated models. Furthermore, the incorporation of sentiment analysis with financial SMI prediction is a major aspect of our future efforts.

References

Al-Hajj R, Assi A, Fouad MM (2018) Forecasting Solar Radiation Strength Using Machine Learning Ensemble. In: 7th International Conference on Renewable Energy Research and Applications (ICRERA), Paris - France, 10: 184–188. IEEE. https://doi.org/10.1109/ICRERA.2018.8567020

Al-Hajj R, Assi A, Fouad MM (2019) Stacking-Based Ensemble of Support Vector Regressors for One-Day Ahead Solar Irradiance Prediction. In: 8th International Conference on Renewable

Energy Research and Applications (ICRERA), Brasov-Romania, 9: 428–433. IEEE. https://doi.org/10.1109/ICRERA47325.2019.8996629

Bahrammirzaee A (2010) A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Comput Appl* 19: 1165–1195. https://doi.org/10.1007/s00521-010-0362-z

Banerjee S, Mukherjee D (2020) Short Term Stock Price Prediction in Indian Market: A Neural Network Perspective. *Stud Microeconomics* 10: 23–49. https://doi.org/10.1177/2321022220980537

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *J Mach Learni Res* 13: 281–305.

Briscoe E, Feldman J (2011) Conceptual complexity and the bias/variance tradeoff. *Cognition* 118: 2–16. https://doi.org/10.1016/j.cognition.2010.10.004

Cervelló-Royo R, Guijarro F, Michniuk K (2015) Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Syst Appl* 42: 5963–5975. https://doi.org/10.1016/j.eswa.2015.03.017

Chen Y, Hao Y (2017) A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Syst Appl* 80: 340–355. https://doi.org/10.1016/j.eswa.2017.02.044

Dietterich TG (2000) Ensemble methods in machine learning. In: Proc. of International workshop on multiple classifier systems-Springer, Cagliari-Italy, 6: 1–15. https://doi.org/10.1007/3-540-45014-9_1

Gao T, Chai Y, Liu Y (2017) Applying long short term momory neural networks for predicting stock closing price. In: 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing - China, 12: 575–578. IEEE. https://doi.org/10.1109/ICSESS.2017.8342981

Hassan EE, Zhang D (2021) The usage of logistic regression and artificial neural networks for evaluation and predicting property-liability insurers' solvency in Egypt. *Data Sci Financ Econ* 1: 215–234. https://doi.org/10.3934/DSFE.2021012

Haykin S (2011) Neural Networks and Learning Machines: a comprehensive foundation. Pearson, 2011.

Heaton JB, Polson NG, Witte JH (2017) Deep learning for finance: Deep portfolios. *Appl Stochastic Models Bus Ind* 33: 3–12. https://doi.org/10.1002/asmb.2209

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9: 1735–1780.

Jiao Y, Jakubowicz J (2017) Predicting stock movement direction with machine learning: An extensive study on S&P 500 stocks. In IEEE International Conference on Big Data (Big Data). 4705–4713. IEEE. https://doi.org/10.1109/BigData.2017.8258518

Li Y, Yi J, Chen H, et al. (2021) Theory and application of artificial intelligence in financial industry. *Data Sci Financ Econ* 1: 96–116. https://doi.org/10.3934/DSFE.2021006

Liu H, Huang S, Wang P, et al. (2021). A review of data mining methods in financial markets. *Data Sci Financ Econ* 1: 362–392. https://doi.org/10.3934/DSFE.2021020

Lu CJ (2013) Hybridizing nonlinear independent component analysis and support vector regression with particle swarm optimization for stock index forecasting. *Neural Comput Appl* 23: 2417–2427. https://doi.org/10.1007/s00521-012-1198-5

Pedregosa F, Varoquaux G, Gramfort A, et al. (2011) Scikit-learn: Machine learning in Python. *J Mach Learn Res* 12: 2825–2830.

Ren Y, Suganthan PN, Srikanth N (2015) Ensemble methods for wind and solar power forecasting—A state-of-the-art review. *Renew Sust Energ Rev* 50: 82–91. https://doi.org/10.1016/j.rser.2015.04.081

Ren Y, Zhang L, Suganthan PN (2016) Ensemble classification and regression-recent developments, applications and future directions. *IEEE Comput Intell Mag* 11: 41–53. https://doi.org/10.1109/MCI.2015.2471235

Sagi O, Rokach L (2018) Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8: 1249.

Vargas M, De Lima B, Evsukoff A (2017) Deep learning for stock market prediction from financial news articles. In: IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Annecy-France: 60–65. https://doi.org/10.1109/CIVEMSA.2017.7995302

Wang Y, Yan G (2021) Survey on the application of deep learning in algorithmic trading. *Data Sci Financ Econ* 1: 345–61. https://doi.org/10.3934/DSFE.2021019

Weng B, Lu L, Wang X, et al. (2018) Predicting short-term stock prices using ensemble methods and online data sources. *Expert Syst Appl* 112: 258–273. https://doi.org/10.1016/j.eswa.2018.06.016

Zhong X, Enke D (2017) Forecasting daily stock market return using dimensionality reduction. *Expert Syst Appl* 67:126–139. https://doi.org/10.1016/j.eswa.2016.09.027

Zhang C, Ma Y (2012) *Ensemble machine learning: methods and applications*. Boston, MA: Springer Science & Business Media.