ORIGINAL ARTICLE

Journal Section

FYS-STK Project 3 on Machine Learning Fall 2019

Alexander Dahl Aliferis | Marius Stette Jessen

Department of Physics, University of Oslo, Norway

[1]Department of Physics, University of Oslo, Norway

Correspondence
Email: alexadal@fys.uio.no

This report considers a study of machine learning methods conducted on a data set of historical prices for the SP500 index. In addition to financial price and volume data, several common technical indicators are computed and used as features in the data set. The final feature in the data set is search volume data from Google Trends for the key word 'S&P500', i.e the name of the market index of the analysis. The aim of the study is to predict the next days closing price of the index. The methods deployed are Long-Short-Term-Memory (LSTM), Support Vector Machines (SVM) and Feed Forward Neural Network (FFNN). The best Mean Absolute Percentage Error (MAPE) obtained, were 0.57%, 0.68%, 0.72% for SVM, LSTM and FFNN respectively. Best percentage of correct trend obtained (PCT) were 0.56, 0.55 and 0.59 in the same order.
The repository used in the study is located on GitHub: GitHub (https://github.com/alexadal/FYS-STK-4155-Project3) repository. Please refer to the readme.md file in the repository for directions on how the code is structured.

KEYWORDS

Artificial Neural Network, Support vector Machines, LSTM Recurrent Neural Network, Machine Learning

1 | INTRODUCTION

Trading of stocks, bonds and other financial instruments have evolved dramatically in the last decades through the use of electronics. Day trading used to consist of phone calls, blackboards and chalk but have been replaced by direct fiber-optical internet connections and autonomous robots trying to exploit arbitrages. Since the existence of public securities trading, there has always been enormous efforts in predicting the market as it would ensure extreme amounts of wealth. However, contradicting one of the most fundamental theories within finance, namely the efficient market hypothesis and random walk has been hard to prove. The hypothesis itself states that it is impossible to predict stock prices, as the market moves randomly with the propagation of news into media and other information channels.

Lately, with the emerging of artificial intelligence (AI) and machine learning (ML) in combination with internet, there might be better opportunities to contravene the theory. Deep trading with the use of neural networks have been proved to outperform the more traditional financial indicators uniquely based on statistics[4]. Particularly, Gao and Chai[5] have proven that using Long Short Term Memory (LSTM) recurrent neural network, fed with basic stock trading data from the S&P500 index and traditional technical indicators that undergo principal component analysis, to be feasible in predicting next day closing prices. Pyo et al.[9] used ordinary artificial neural network (ANN) and support vector machines (SVM) together with data from google trends in a study of predicting the South-Korean stock index (KOSPI 200). The idea was that google trends, which is a statistical property of keyword volume in google searches, gives a good indication of global interest, news and human perception around the actual keyword. Thus in their study, the keyword used was "KOSPI 200 index", with somewhat mixed result but a conclusion that utilisation of google trends in predictions shorter than a month may be effective.

The purpose of this study, is to further investigate the results of Gao and Chai[5] and Pyo et al.[9], by combining their central findings. The study evaluates the statistical performance of different structures of LSTM, support vector regression (SVR) and ANN on historical data of the S&P500 index together with google trend data on the keyword "S&P500 index". Simply put, the study tries to replicate the input structure of Gao and Chai[5] with the additional input of google trend data. For performance measure, mean average percent error (MAPE), Average Mean Absolute Percentage Error (AMAPE), root means sqaured error (RMSE), mean absolute error (MAE) and percentage of correct trend (PCT) are used.

2 | DATA SET

The data used in this study is concatenated from two sources: Yahoo Finance and Google Trends. From Yahoo Finance, daily historical trading data for the S&P500 Index[2] is fetched. The index measures the stock performance of 500 large US companies traded on different stock markets. Yahoo Finance provides the opening price, highest price, lowest price, closing price and the trading volume for a day. It also gives the closing price adjusted for dividends. Daily google search volumes on 'S&P500' were collected thought the pytrends[1] Python package. The data from both sources are taken for all trading days from January 2nd 2013 to November 29th 2019. Table 1 presents the columns in the data set. By inspection, it was clear that for the period chosen, AdjClose and Close were identical. Thus the AdjClose column was dropped from the analysis. In addition to the above, several financial indicators are calculated from collected data, in order to verify any potential affection of predictions. The calculation of these indicators and additional input features are described by Gao and Chai[5] and summarized in table 2. For a more thorough explanation of the various indicators, please refer to Gao and Chai[5].

In Figure 1 below follows a correlation matrix of the data set. There is a quite high correlation between Google trend data and next day close price.

TABLE 1 Input Data

Label	Description	Source	Symbol
Open	Opening price of the day	Yahoo Finance	OP
High	Highest price of the day	Yahoo Finance	HI
Low	Lowest price of the day	Yahoo Finance	LO
Close	Closing price of the day	Yahoo Finance	CL
AdjClose	Closing price of the day adjusted for dividends.	Yahoo Finance	AD
Volume	Opening price of the day	Yahoo Finance	VO
Trend	Daily search volume in Google (relative to maximum search volume)	Google Trends	-
Next_day	Closing price of the following day	Yahoo Finance	-

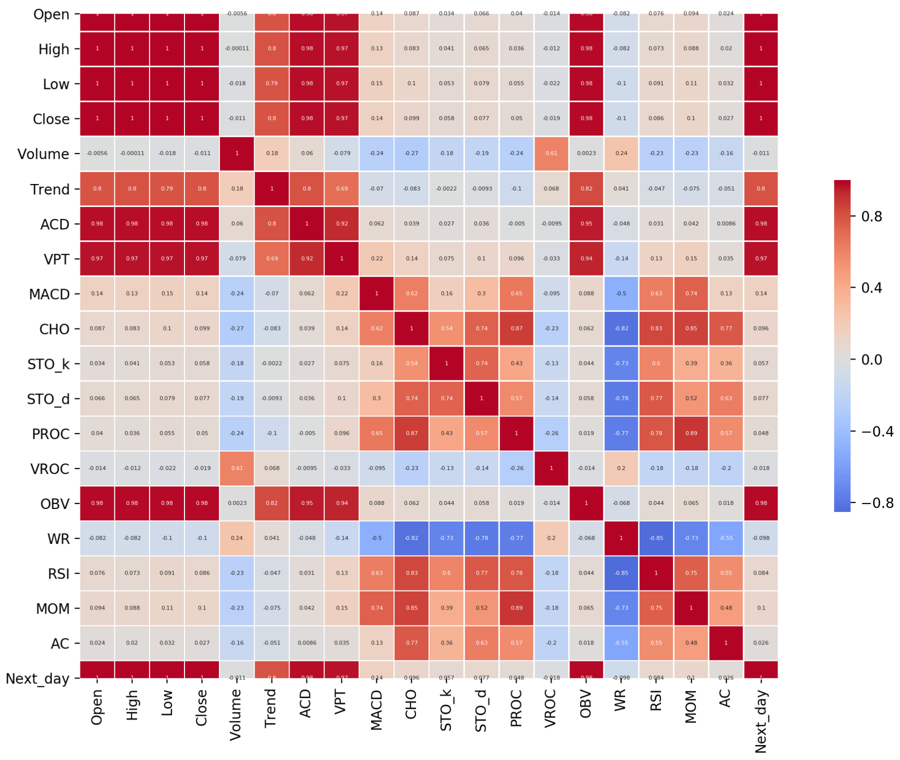

FIGURE 1 Correlation matrix of the input features

TABLE 2 Technical Indicators

Indicator	Formula	Description
ACD	ACD$_{t-1}$ + $VO\times\frac{\text{(CL-LO)-(HI-CL)}}{\text{HI-LO}}$	Accumulation distribution
MACD	EMA(CL,12)-EMA(CL,26)	Moving Average Convergence Divergence
CHO	EMA(AD,3)-EMA(AD,10)	Chaikin oscillator
STO-%K	$\frac{\text{CL-Lowest(5)}}{\text{Highest(5)-Lowest(5)}}$	"Slow" - Stochastic Oscillator
STO-%D	MA(STO-%K,3)	"Fast" - Stochastic Oscillator
VPT	VPT$_{t-1}VO{+}\times\frac{CL-CL_{t-1}}{CL_{t-1}}$	Volume Price Trend
W-%R	$\frac{Highest(14)-CL}{Highest(14)-Lowest(14)}$	Williams Percent Range
RSI	100-$\frac{100}{1+RS}$, $RS=\frac{\text{Average of Upward Price Change}}{\text{Average of Downward Price Change}}$	Relative Strength Index
MOME	CL-CL$_{t-14}$	Momentum

AC	AO-MA(AO,5), where
	AO = MA(MedianPrice,5)-MA(MedianPrice,34),
	MedianPrice = $\frac{HI+LO}{2}$
		Price Acceleration

PROC	$\frac{CL-CL_{t-12}}{CL_{t-12}}$	Price Rate of Change
VROC	$\frac{VO-VO_{t-12}}{VO_{t-12}}$	Volume Rate of Change
OBV	OBV$_{t-1}+VO$; if $CL\geq$ CL$_{t-1}$ $OBV_{t-1}-VO$; if $CL<CL_{t-1}$	On Balance Volume
NOTE	MA(x,t) = $\frac{1}{t}\sum_{i=1}^{t}x_i$ EMA(x,t) = $\alpha\times x+(1-\alpha)\times EMA(x,t-1),\alpha=2/(t+1)$ $\text{Highest}(n)=\max([CL_{t-n},...,CL_{t-1},CL_t)$ $\text{Lowest}(n)=\min([CL_{t-n},...,CL_{t-1},CL_t)$	Moving Average Exponential Moving Average Highest closing price during n previous days Lowest closing price during n previous days

2.1 | Pre-processing of data

The data is split into training, validation and test sets. As the data consist of time series, the data points are not shuffled before splitting. This means that the data in the training set is from the earliest dates, then the validation set followed by the test set. The models are trained using the training set and tuned using the validation set. Finally the test set is used to measure the performance of the models.

2.1.1 | Scaling

First, the data is scaled using Scikit-learn's MinMaxScaler[13] so that all the training data are between 0 and 1. The same transformation is performed on the validation and test set. Secondly, the mean of the training data set is removed from all data points using Scikit-learn's StandardScaler[16]. It should be noted that, as the market index is generally increasing, and the data is normalized based on the training set, there is risk of the model underestimating the real value. To somewhat account for this, the validation and training sets are kept relatively small at 10% each of the total amount of data.

2.1.2 | Principal Component Analysis (PCA)

To reduce the execution time of the algorithms to be applied, Principal Component Analysis(PCA) is performed on the financial indicator features. It is likely that several of the technical indicators are highly correlated,[5] which is why PCA is performed on them. The idea of PCA, is to reduce the number of variables, while retaining as much of the variances as possible (Marsland[7]). While retaining 95% of the variances in the data set, Scikit-learn's PCA algorithm[14] was able to reduce the number of the financial indicators to be used as features from thirteen to seven.

2.1.3 | Sliding window

Similar to Gao and Chai[5], a sliding window is applied in order to create sequenced input data. That way each input contains data for the current day and the x previous days. The length, x, i.e the number of days, of the window is varied in this study to find the optimal length. Figure 2 is borrowed, but slightly generalized, from Gao and Chai[5] and illustrates the mechanics of how the input data is sequenced using the sliding window. The study conducted by Gao and Chai[5] only considered a window size of 20 days.

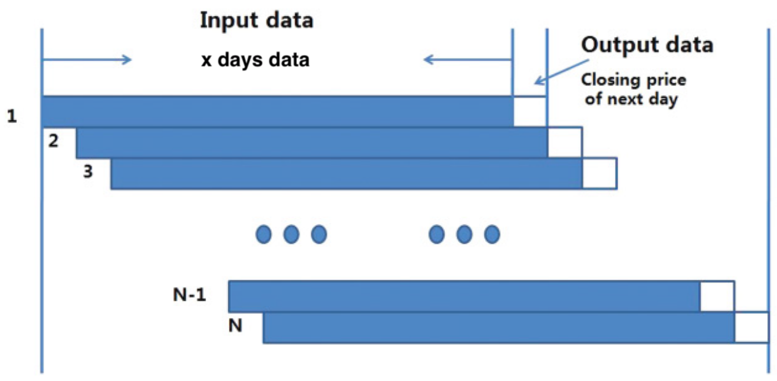

FIGURE 2 Schematic of the sliding window. Taken from Gao and Chai[5], where the length of the windows are generalized to x days.

3 | THEORY AND METHODOLOGY

3.1 | SVM

SVR is a subclass of support vector machines, which is a method that tries to maximize the distance or margin of a separating hyperplane between two or more classes of data. The equation of a hyperplane in p dimensions is[6]

$$b + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p = 0 \tag{1}$$

or in vector notation

$$f(x) = \boldsymbol{w}^T \boldsymbol{x} + b = 0, \tag{2}$$

where the vector $\boldsymbol{w}^T = (w_1, \ldots, w_P)$, is called the normal vector, wich points in a direction orthogonal to the hyperplane surface.

In a proposed classification problem, the function above separates two hyper planes, M_1 and M_2. Thus, two data points x_1 and x_2, found on opposite sides of $f(x)$ in equation 2 will result in the following:

$$\boldsymbol{w}^T \boldsymbol{x}_1 + b = M_1 \tag{3}$$

and

$$\boldsymbol{w}^T \boldsymbol{x}_2 + b = M_2, \tag{4}$$

The distance between the classes can be found by subtracting equation 3 from equation 4:

$$M_2 - M_1 = \boldsymbol{w}^T(\boldsymbol{x}_2 - \boldsymbol{x}_1) = M \tag{5}$$

Further normalizing equation 5 by $||\boldsymbol{w}|| = \sqrt{\sum_{i=1}^{p}(w_i^2)}$, gives the expression

$$\frac{\boldsymbol{w}^T(\boldsymbol{x}_2 - \boldsymbol{x}_1)}{||w||} = \frac{M}{||w||} \tag{6}$$

The distance or margin given by the left hand side can thus be maximized by finding the minimum of $||\boldsymbol{w}||$, subject to the constraint in equation 7 below, that each observation is classified correctly

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1. \tag{7}$$

For practical reasons the the minimization of $||\boldsymbol{w}||$ resulting from equation 6, is replaced by $\frac{1}{2}||\boldsymbol{w}||^2$ (Raschka [10]).

3.1.1 | SVR

In the case of regression, the algorithms used are somewhat more complicated, but the main logical concept of individualizing the hyperplane which maximizes the margin is preserved . The idea is to construct a band of $[-\epsilon, +\epsilon]$ about the hyper plane that embraces the largest possible amount of the data points close to the hyper-plane. Also, with the introduction of slack, which is the allowance of points at a total distance ξ outside the band boundary, the following steps constitutes the SVR method [11]

- minimize $\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N}(\xi + \xi^*)$. $\xi_i and \xi_i^*$ are on opposite sides of the boundary, C is a constant.
- Under the constraints
  - $y_i - wx_i - b \le \epsilon + \xi_i$
  - $wx_i + b - y_i \le \epsilon + \xi_i^*$
  - $\xi, \xi^* \ge 0$

As the stock prices are not of a linear form, kernel functions to transform the data into higher feature dimensions have been applied in this study. The transformation is expressed as $x \longrightarrow z = \phi(x)$ and yields the "dual problem" with solution [11]

$$f(x) = \sum_{i=1}^{n_{sv}} (\alpha_i - \alpha_i) K(\boldsymbol{x_i}, \boldsymbol{x}) \tag{8}$$

$$0 \le \alpha_i * \le C \tag{9}$$

$$0 \le \alpha_i \le C$$

,

where n_{sv} is the number of support vectors i.e points positioned on $[+\epsilon, -\epsilon]$, α_i dual variable for each data point and $K(\boldsymbol{x_i}, \boldsymbol{x})$ a kernel function. The kernel function used in this paper was the gaussian radial basis function (rbf) kernel presented by [11]

$$K(x_i, x_j) = exp\left(-\frac{||x_i, x_j||}{2\sigma^2}\right). \tag{10}$$

3.2 | Deep Learning

3.2.1 | ANN

As in a previous study by the authors, a feed forward artificial neural network (FFNN) is applied. The FFNN consist of a number of neurons or nodes in different layers, that "communicate" in mathematical form. Between each neuron in neighboring layers, there is a connection represented by a weight variable, and before information is propagated forwards, the accumulated signal at a node has to go through an activation threshold, $f(z)$, to yield an output. This activation is often the same as the binary conditional probability depicted below [6]

$$p(y_i = 1|x_i, \hat{\beta}) = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)},$$

also known as the sigmoid function, $\sigma(\beta_i, x_i)$. Layers that are neither input nor output layers in neural networks are called hidden layers.

Moreover, for a given layer, the input can be represented as z_i^l (as opposed to x_i for the input layer), and the output as $y_i = a_i^l = f(z_i^l)$. z_i^l is again a function of weights, w, and biases, b[6]

$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l, \tag{11}$$

where the superscript $l - 1$ indicate outputs of the prior layer. Consequently, the cost function in the final layer, that is to be minimized, is[6]

$$C(\hat{W}) = -\sum_{i=1}^{n} \Big(t_i \log a_i^L + (1 - t_i) \log (1 - a_i^L) \Big) \tag{12}$$

where the targets are defined as t_i. To minimize the cost function, back-propagation with "Adam" (defined later) is used. Back-propagation is, simply put, utilization of the chain rule wrt. to all the activations, weights and biases from the final to the first layer. A short, but more detailed explanation follow the steps below, where values are presented by matrix-notation. Please note that $\boldsymbol{w}^T$ has the shape $(n_{\text{prev.layer}}, n_{\text{next.layer}})$ and $\circ$ denotes as the Hadamard product[6].

1. Do a feed-forward. That is, send the input signal throughout the network and evaluate the cost.
2. Divide the computation of derivatives into two pieces, one for the final outer layer and the other for the inner hidden layers
   - Outer Layer
     - calculate $\delta_{out} = \frac{\partial C}{\partial a_{out}} \circ \frac{\partial a_{out}}{\partial z_{out}} = \boldsymbol{a_{out}} - \boldsymbol{y}$ (for the cross-entropy cost)
     - $\frac{\partial z_{out}}{\partial w_{out}} = \boldsymbol{a_{out-1}}$
     - $\frac{\partial C}{\partial w_{out}} = \delta out^T \circ \frac{\partial z_{out}}{\partial w_{out}}$ and $\frac{\partial C}{\partial b_{out}} = \sum_{n_{inputs}} \delta_{out}$

   - Hidden layers
     - For layer h, ranging from the second last to the first layer, calculate
     - $\delta_{h+1} = \frac{\partial C}{\partial a_{h+1}} \circ \frac{\partial a_{h+1}}{\partial z_{h+1}}$, i.e delta from the previous layer in the iteration, that goes backwards from the outer layer
     - $\frac{\partial C}{\partial a_h} = \frac{\partial C}{\partial a_{h+1}} \circ \frac{\partial a_{h+1}}{\partial z_{h+1}} \circ \frac{\partial z_{h+1}}{\partial a_h} = \delta_{h+1} \circ \boldsymbol{w_{h+1}}$
     - $\frac{\partial z_h}{\partial a_h} = f_h'(z_h)$
     - $\delta_h = \frac{\partial C}{\partial a_h} \circ \frac{\partial z_h}{\partial a_h} = \delta_{h+1} \circ \boldsymbol{w_{h+1}} \cdot f_h'(z_h)$
     - $\frac{\partial z_h}{\partial w_h} = \boldsymbol{a_{h-1}}$

- To finally give, $\frac{\partial C}{\partial w_h} = \delta_h^T \circ \frac{\partial z_h}{\partial w_h}$ and $\frac{\partial C}{\partial b_h} = \sum_{n_{inputs}} \delta_h$

As the study of predicting stock prices is not a classification problem, the cost function in equation 12 is replaced by MSE as shown in equation 13

$$MSE = C(\hat{W}) = \frac{1}{2N}\sum_{i}^{N}(y_i - t_i)^2. \tag{13}$$

Further, the activation function in the final layer of the model (also for LSTM) is set to linear i.e

$$f(x) = x$$
$$f'(x) = 1$$

or ReLU:

$$f(x) = max(0, x)$$
$$f'(x) = 0 \text{ or}$$
$$f'(x) = 1$$

In terms of training the model, Adam optimizer were applied to solve the minimization problem of finding optimal weights and biases. The Adam optimizer takes a running average of both the first and second moment of the gradients to update the weights and biases[6]. The following expressions summarize the steps involved[5]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla Q(w), \tag{14}$$
$$v_t = \beta_1 v_{t-1} + (1 - \beta_2)(\nabla Q(w))^2, \tag{15}$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{16}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{17}$$
$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{18}$$

where $Q(w)$ is the error function, $\nabla Q(w)$ is the gradient, m_t is the first moment estimate at time t, and v_t the second moment estimate of t. Equation 18 is the actual update of weights, and a similar expression yields for the biases.

3.2.2 | LSTM

LSTM goes under the group of recurrent neural networks (RNNs) which are very much like FFNNs, except that they also have connections pointing backward[6]. Thus, RNNs as seen in Figure 3, form connections of loops where output of the previous information, propagated through the network is put back into the model together with new information.

LSTM neural networks are capable of learning with large sequences of information, hence remembering for long periods[8].

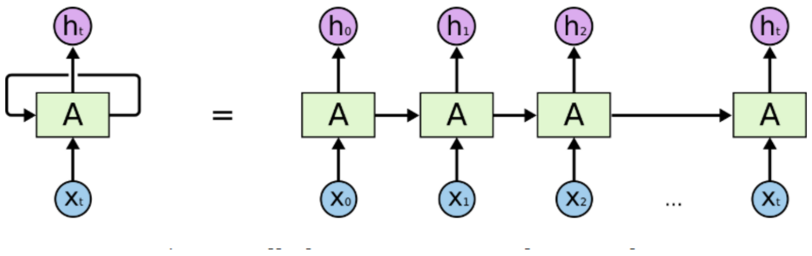

FIGURE 3 Illustration of RNN. Source: Olah[8]

The chain-like structure of LSTM consist of a repeating module, or memory cell, with four interacting network layers as depicted in Figure 4. These layers form an input gate, a forget gate, an output gate and a neuron unit[5]. The cell itself has a cell state which runs down the entire chain as a conveyor belt[8]. The mentioned gates decides whether information should be added, read and forgotten from the cell state. The following equations and Figure 5 describes the process in detail[8]

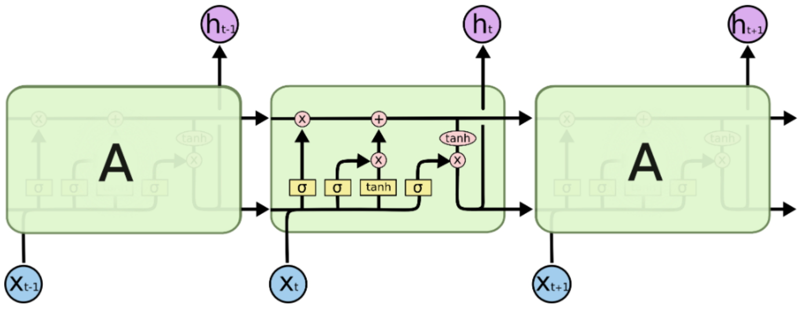

FIGURE 4 Illustration of LSTM model. Source: Olah[8]

$$f_t = \sigma\left(W_f\,[C_{t-1}, h_{t-1}, x_t] + b_f\right) \tag{19}$$

$$i_t = \sigma\left(W_i\,[C_{t-1}, h_{t-1}, x_t] + b_i\right) \tag{20}$$

$$o_t = \sigma\left(W_o\,[C_{t-1}, h_{t-1}, x_t] + b_o\right) \tag{21}$$

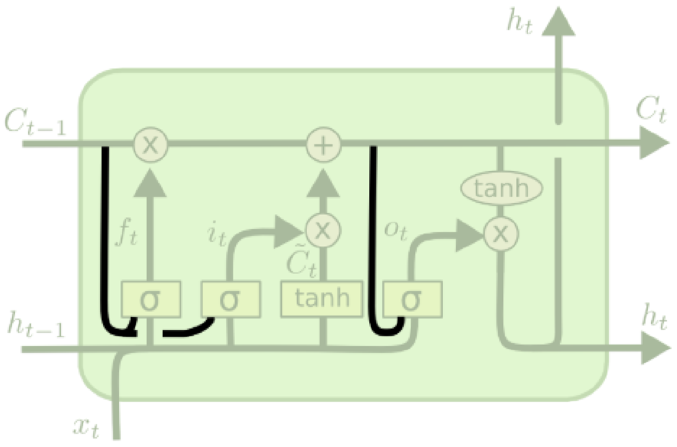

FIGURE 5 Illustration of a LSTM cell. Source: Olah[8]

$$\tilde{C}_t = \tanh\left(W_C\left[C_{t-1}, h_{t-1}, x_t\right]\right) + b_C \tag{22}$$

$$C_t = f_t \times C_{t-1} + (1 - f_t) \times \tilde{C}_t \tag{23}$$

$$h_t = o_t \times \tanh C_t \tag{24}$$

where x_t is the input vector at time t, h_t is the output vector, C_t is the cell state, i_t the input gate vector, f_t the forget gate vector, o_t is the output gate vector, W_i, W_f, W_o, W_c are weight matrices, b_f, b_i, b_o, b_C are bias vectors and, σ, the sigmoid activation function[5]. The expressions in brackets represents concatenation of past information and new inputs, in order to decide what to forget. To determine optimal weight and biases, the Adam optimizer was also used in the case of LSTM.

3.3 | Model architectures

As described in the subsequent sections, parameter tuning was deployed to optimize the input parameters of the models. Additionally, all the mentioned models were optimized for different window sizes.

3.3.1 | SVM

To find the optimal SVR model for each window size, Scikit-Learn's GridSearchCV[12] was used. As explained, cross validation was not used, but Scikit-Learn's PredefinedSplit[15] was deployed to ensure the correct data were used as training and validation during the grid search. The input parameters used in the grid search are summarized in table 3. The values for C, Gamma and Epsilon were spaced out on a logarithmic scale in the intervals given in the table.

TABLE 3 Input parameters for tuning of the Support Vector Machine

Parameter	Values
Kernel	'rbf'
C	[10e-8, 10e3]
Gamma	[10e-8, 10e3]
Epsilon	[10e-8, 10e-1]

3.3.2 | Deep Model architectures

The neural networks were built in python using Tensorflow 2.0 and Keras. In order to obtain optimal models, a randomized grid-searches by the use of Talos [17] were performed. Parameters that were used during the tuning were; window size, number of neurons, learning rate, epoch size, batch size and dropout. The latter, dropout, is a form of regularization with random dropout of nodes. This simulates having a large number of network architectures with the use of a single model only, and is effective of reducing over-fitting [3]. The default output activation function of the models was set to linear.

TABLE 4 Input parameters for tuning of the Support Vector Machine

Parameter	FFNN	LSTM
Learning Rate	[1e-5, 1], with steps of 20	[1e-3, 1e-1], with steps of 15
First Neuron	20, 50, 128	10, 50, 128, 256
Second Neuron	20, 50, 128	10, 50, 128, 256
Batch Size	10, 20, 50, 100, 300	10, 20, 50, 100
Epochs	10, 30, 50, 100	20, 30, 50, 90
Dropout	0, 0.2	0, 0.2, 0.4

As the total number permutations from the above table became very large, the grid-search was down-sampled to only map 1% of these. Also, as the optimization process for LSTM with Google trend data turned out to be extremely time-consuming, the number of permutations were further reduced in the case of excluding trends data. Hence, the first round of optimization formed a new basis of parameters used in subsequent rounds of tuning i.e for input features excluding trends data.

In addition to the Talos optimization, a try and search "hand tuning" were also performed by the authors.

3.4 | Performance measures

In this section the metrics used to measure the performance of the models, are presented. Equation 25 represent the Mean squared error, MSE. Here z is the true value, $\tilde{z}$ is the predicted value and N is the total number of data points.

$$MSE = \frac{1}{N}\sum_{i}^{N}(z_i - \tilde{z}_i)^2 \tag{25}$$

MSE is simply the average value of the prediction errors, giving the most optimal score of zero. MAPE is the mean absolute percentage error, depicted in equation 26. It measures the deviation between the predicted and true values as a percentage.

$$MAPE = 100 \times \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\tilde{z}_i - z_i}{z_i}\right| \tag{26}$$

The Mean Absolute Error (MAE), is depicted in equation 27. It represents the average of the absolute error.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\tilde{z}_i - z_i| \tag{27}$$

Equation 28 is the Root Mean Square Error (RMSE), which is the standard deviation of the residuals.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{z}_i - z_i)^2} \tag{28}$$

$$AMAPE = 100 \times \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\tilde{z}_i - z_i}{(1/N)\sum_{i=1}^{N} z_i}\right| \tag{29}$$

Equations 29 and 30 represent the Average Mean Absolute Percentage Error (AMAPE) and the Percentage of Correct Trend (PCT) respectively. PCT is a measure that is used to measure the model's ability to predict the correct trend.

$$PCT = \frac{1}{N}\sum_{i=1}^{N}\mu_i, \text{ where } \mu_i = \begin{cases} 1, & \text{if } (\tilde{z}_{i+1} - z_i)(z_{i+1} - z_i) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

4 | RESULTS

Results of running the optimized models for different window-sizes including Google trend, are showcased in tables 5-7. From the tables, it is evident that the best performance, regardless of window-size, is achieved by SVR. Further evaluation of the tables, give an intuition that a 20 day window-size give the overall best performance with respect to MAPE and PCT. The model-outputs initiating these performance measures are plotted and compared to the actual index value in Figure 7. The figure is a good indicator of how MAPE and PCT represent different levels of performance. Low MAPE give a good fit to the true curve while high PCT, although perhaps not as noticeable, indicates how well the trend is predicted. For all models, there is no obvious indication of better performance with gradually increased window size.

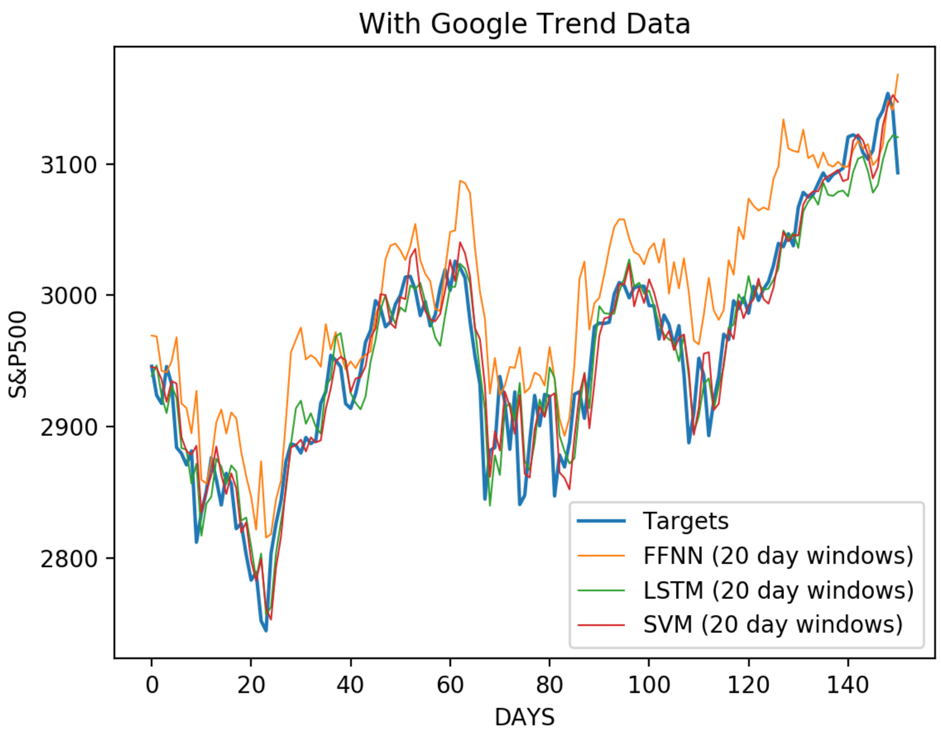

FIGURE 6 Prediction of the models created with data from Google trends, plotted against the true target values.

TABLE 5 SVM - With Google Trend

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
1	23.4907	0.5734	0.53	16.7867	0.5688
3	24.6429	0.6388	0.50	18.7676	0.6357
5	23.2911	0.5880	0.50	17.2171	0.5831
7	23.2747	0.5936	0.50	17.3970	0.5890
10	24.2358	0.6134	0.47	17.9591	0.6078
15	23.8645	0.6110	0.54	17.8876	0.6050
20	24.5533	0.6176	0.56	18.0961	0.6119

TABLE 6 FFNN - With Google Trend

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
1	32.9325	0.8872	0.59	26.0993	0.8844
3	42.1581	1.2618	0.44	37.3545	1.2653
5	32.7677	0.8320	0.49	24.6036	0.8332
7	36.5846	0.9532	0.53	27.9698	0.9469
10	30.4403	0.7925	0.51	23.2461	0.7867
15	75.1494	2.1448	0.57	63.0788	2.1336
20	52.1441	1.4512	0.56	42.5509	1.4389

TABLE 7 LSTM - With Google Trend

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
3	42.9214	1.2176	0.44	36.2142	1.2267
5	65.2849	1.9181	0.44	57.2721	1.9395
7	34.9352	0.9690	0.45	28.7171	0.9722
10	30.7255	0.8282	0.48	24.5271	0.8301
15	34.8700	0.9181	0.47	27.2007	0.9201
20	31.5564	0.8203	0.54	24.1654	0.8172
Hand-tuned	1 input	1 hidden layer	Relu activation		
20	26.0709	0.6843	0.55	20.1071	0.6802

Table 8-10 show performance of running the models without Google trend as an input feature. Again, SVR outputs the overall best results also supported by Figure 7. There is little or no clear improvement of performance in the case of SVR and FFNN, but for LSTM, MAPE results are better with the addition of trend data. This difference is proposedly caused by optimization bias as only a subset of the total permutations were used in tuning the model without Google trend data. However, running the models with other parameters of identical numbers and values while also seeding the randomized search, did not change this finding (See Appendix for reference).

Compared to Gao and Chai [5], the results obtained are in a somewhat satisfactory range. There are however some differences to assess. The current study achieves the best performance for SVR, in contrast to LSTM in Gao and Chai [5]'s study. Overall, as LSTM introduces memory to the analysis, it is expected to outperform the other models. The LSTM models used in this study are due to time constraints presumably not tuned to their fullest. Another source of differences are the data set itself. Stock markets are constantly changing with time and can change dramatically in both trend and value. High gradients in the index price evolution yields large changes in index prices and will presumably give overall larger MAPE-values. Normalization without information leak is also another difficult aspect, as the index is drifting upwards over the given time-horizon. Hence, not all min and max values in the test and valid set that are normalized with values from the train set, are scaled between 0 and 1. Several attempts on different normalization techniques were applied without any improvement. How sub-optimal normalization directly affects the different models are difficult to determine. As opposed to Gao and Chai[5], this study obtained SVR as the optimal model. This result, indicates a lesser importance of memory in stock predictions.

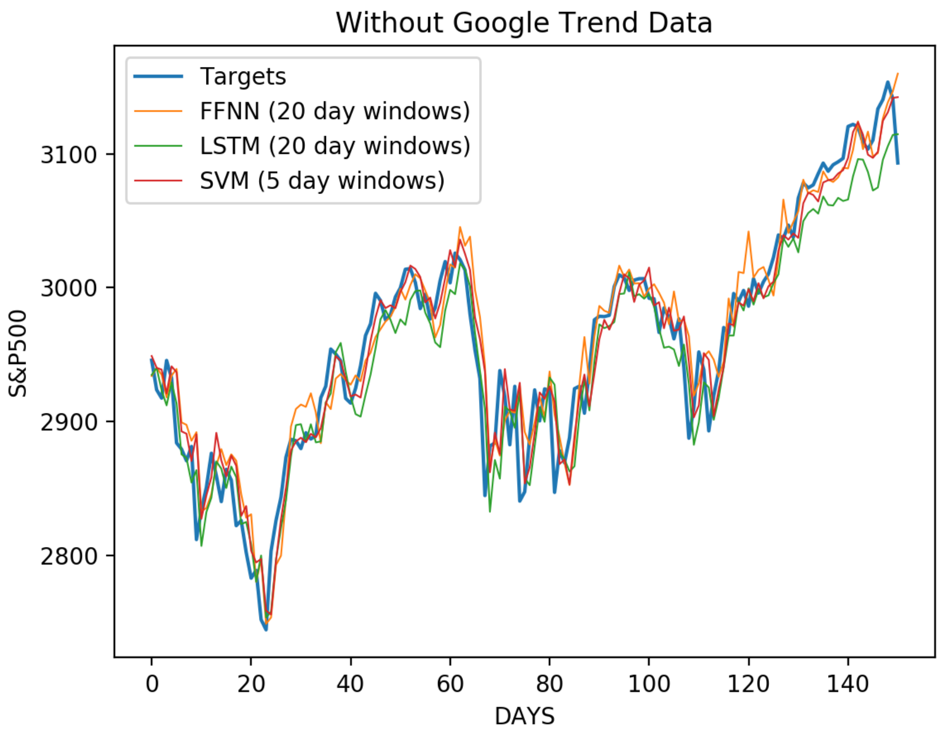

FIGURE 7 Prediction of the models created without data from Google trends, plotted against the true target values.

TABLE 8 SVM - Without Google Trends

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
1	23.3322	0.5865	0.51	17.1775	0.5821
3	25.9337	0.6997	0.44	20.6170	0.6984
5	23.5074	0.5835	0.55	17.0875	0.5787
7	23.7815	0.6099	0.44	17.8811	0.6054
10	23.9296	0.6030	0.49	17.6425	0.5971
15	24.0834	0.6115	0.53	17.8935	0.6052
20	24.1714	0.6336	0.48	18.5895	0.6286

TABLE 9 FFNN - Without Google Trend

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
1	33.4123	0.8826	0.46	26.0890	0.8840
3	35.1334	0.9498	0.55	27.9140	0.9456
5	28.6389	0.7481	0.45	22.0211	0.7457
7	36.1416	0.9334	0.58	27.2955	0.9241
10	29.6547	0.7470	0.54	21.8971	0.7410
15	39.3848	1.1274	0.48	33.5241	1.1339
20	28.1737	0.7206	0.51	21.1008	0.7135

TABLE 10 LSTM - Without Google Trend

Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
15	36.0058	0.9883	0.52	29.2640	0.9898
20	32.3683	0.8978	0.53	26.5070	0.8964
Hand-tuned	1 input	1 hidden layer			
20	28.0486	0.7660	0.44	22.6314	0.7681

5 | CONCLUSION

This study evaluates the predictive performance of three different machine learning models on the S&P500 index. The study further assessed how human perception about the stock market could be included in the predictive analysis through utilization of Google trends. Compared to other literature the values obtained are acceptable.

In contrast to Gao and Chai [5], SVR showed the overall best performance, indicating that memory is of less importance in stock prediction than initially anticipated. However, the authors point out that the LSTM models are not believed to perform optimally due to time-restraint and sub-optimal normalization. There are also a large variety of reasons to why our results otherwise differentiate, as for example the length of the training data and index/stock volatility.

Google trends showed to be highly correlated with next days closing price, but only LSTM was seemingly positively affected by the addition of trend data. This might be a result of biased hyper-parameter tuning, but could be an enhancing factor to Gao and Chai [5]'s study.

In a future analysis, to get an even better distinction of model performance with correspondingly more rigid results, time-series cross validation should be implemented. Further measures of improvement includes longer history of training data, longer/better model hyper-parameter tuning and other sources of human perception. Additionally it would be interesting to use a dynamic LSTM approach, where the model is continuously retrained as the sliding windows and hence time passes along. A proposed alternative to mitigate the normalization problem could also be predicting returns or log-returns instead of stock prices.

references

1. (Year Not specified) pytrends 4.7.2. https://pypi.org/project/pytrends/. Accessed: 2019-12-18.

2. (Year Not specified) Yahoo finance: Sp500. https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC. Accessed: 2019-12-18.

3. Brownlee, J. (2018) Machine learning mastery blog. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. Accessed: 2019-17-10.

4. Eurekahedge () Eurekahedge article. ://www.eurekahedge.com/Research/News/1614/Artificial-Intelligence-AI-Hedge-Fund-Index-Strategy-Profile.

5. Gao, T. and Chai, Y. (2018) Improving stock closing price prediction using recurrent neural network and technical indicators. *Neural Computation*, **30**, 1–22.

6. Hjorth-Jensen, M. (2019) Lecture notes fys-stk4155.

7. Marsland, S. (2014) *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman & Hall/CRC, 2nd edn.

8. Olah, C. (2015) Colah blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2019-12-10.

9. Pyo, S., Lee, J., Cha, M. and Jang, H. (2017) Predictability of machine learning techniques to forecast the trends of market index prices: Hypothesis testing for the korean stock markets. *PLOS ONE*, **12**, e0188107.

10. Raschka, S. (2015) *Python Machine Learning*. Packt Publishing.

11. Sayad, D. S. (Unknown) Sayad blog. http://www.saedsayad.com/support_vector_machine_reg.htm?source=post_page-----8eb3acf6d0ff----------------------. Accessed: 2019-17-10.

12. Scikit-Learn (Year Not specified) Scikit-learn: Gridsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed: 2019-12-18.

13. — (Year Not specified) Scikit-learn: Minmaxscaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. Accessed: 2019-12-18.

14. — (Year Not specified) Scikit-learn: Pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. Accessed: 2019-12-18.

15. — (Year Not specified) Scikit-learn: Predefinedsplit. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.PredefinedSplit.html. Accessed: 2019-12-18.

16. — (Year Not specified) Scikit-learn: Standardscaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed: 2019-12-18.

17. Talos, A. (2018) Talos. http://github.com/autonomio/talos. Accessed: 2019-12-10.

6 | APPENDIX

TABLE 11 LSTM - Best Results Test runs 1

With Trends					
Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
15	32.271383	0.896027	0.483871	26.552055	0.898114
20	47.232142	1.353372	0.460000	40.370654	1.365181
30	37.192466	1.051483	0.435714	31.352288	1.058744
Without Trends					
15	33.678738	0.929194	0.470968	27.502504	0.930263
20	51.709744	1.497265	0.466667	44.757085	1.513513
30	59.285573	1.746564	0.442857	52.262008	1.764850

TABLE 12 LSTM - Best Results Test runs 2

With Trends					
Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
20	33.064375	0.879597	0.500000	26.064688	0.881408
30	32.194757	0.897259	0.457143	26.576684	0.897475
Without Trends					
Window[days]	RMSE	MAPE[%]	PCT	MAE	AMAPE[%]
20	38.470966	1.072913	0.433333	31.837949	1.076637
30	36.959303	1.013679	0.471429	30.156462	1.018362