City University

London, MSc in Data Science

Project Report

2020

Impact of Covid-19 on Stock Market Index: An algorithm comparison for forecasting time series

Prerna Prakash Gupta

Supervised by: Arturd' Avila Garcez

30/11/220

Abstract:

In this study we will be using deep learning models in order to forecast the daily closing values of the Dow Jones Industrial Average index (DJIA). This study will take into account the global daily Covid-19 cases and examine the trajectory of the DJIA index with a multivariate time series model. In our study we will explore the optimal model for time series forecasting, in case of noisy data with a shorter time horizon. We will examine the more recent artificial intelligence approaches like Recurrent Neural Networks (RNN) which have tendency to adapt to the features created by the data itself and compare and contrast these with traditional machine learning techniques like Support Vector Regressions, basis different evaluation metrics and parameters.

LSTM, SVM, Time Series, Covid-19, Non-Stationary

Appendix A: Project Proposal for MSc in Data Science

Individual Project Proposal
Prerna Prakash Gupta - MSc Data Science FT1
Email: Prerna.prakash-gupta@city.ac.uk
Phone:0091-9871155595
Supervisor: Artur Garcez

Trajectory of Stock prices in wave of Covid-19 Pandemic

1) Introduction:

1.1) Background

On March 11, the World Health Organization characterized COVID-19 as a pandemic. With widespread commotion in personal lives and uncertainty of the future, whereby half of the world's population was under, Covid-19 has stirred tremendously feverish behaviour by investors worldwide. In such a situation studying asses price changes is particularly interesting as it gives us a unique opportunity to study the behaviour of equity markets and investor expectations. In this study we will be using deep learning models in order to forecast the daily NASDAQ stock exchange rate. *We will examine daily time series data of Covid-19 deaths and examine the future trajectory of NASDAQ stock exchange rate in wave of the current pandemic, with a multivariate time series model.*

1.2) Overview: Time Series

The seminal approach in time series forecasting since the past four decades has been the landmark Box-Jenkins Auto-Regressive-Integrated-Moving-Average ARIMA model. It is available for autoregressive processes (AR), moving average processes (MA) and the combination of the two (ARMA). Further, this model may also have seasonal fluctuations in the SARIMA time series. However, a major limitation of this model is its high error rate as soon as normality and/or linearity in the data is lost (1). Further, ARIMA concentrates on the values of the past series data and is unable to capture the dynamism in the time series process (2). Thus, for the purpose of our model we will explore the more recent artificial intelligence approaches like Recurrent Neural Networks (RNN). The particularity of neural networks is their tendency to adapt to the features created by the data itself and capability to retain long term dependencies in the time series data (3). However, RNN is expected to be plagued by the problem of catastrophic forgetting, whereby with successive time periods, the network loses the tendency to retain the information gathered from the previous time periods as the weights as successively updated with the time periods. We will investigate the same in our hypothesis on the asset price data. Further, we will compare, critique and contrast the aforementioned model with the continual learning algorithm, whereby, a network architecture is constructed that can continually accumulate knowledge over different tasks without the need for retraining from scratch, with methods in particular aiming to alleviate forgetting (4).

Based on the above, our research question is forecasting the future trajectory of NASDAQ composite index, in the wave of Covid-19 pandemic. In this regard, we will evaluate the conventional artificial Neural Network models (elaborated below). We will further examine if these models are plagued by 'catastrophic forgetting' phenomenon and compare and contrast the aforementioned model with continual learning algorithms on our dataset. Going forward, we will use the optimal model to predict the future trajectory of NASDAQ stock prices.

2) Critical Context:

In the field of time series modelling, there has been an extensive range of research work. Oussama Fathi, proposed a hybrid ARIMA and Long Short Term Memory network (LSTM) approach whereby ARIMA model is expected to capture the seasonal peaks in the data and LSTM is expected to capture the stable part of the time series; for the purpose forecasting US dollar/British pound exchange rate data (5). Hua, Zhao, Li, et al; proposed the Random Connectivity LSTM (RCLSTM) model to reduce the considerable computational cost of LSTM on traffic and user mobility in telecommunication networks. As opposed to LSTM, RCLSTM was formed via stochastic connectivity between neurons, which achieved a significant breakthrough in the architecture formation of neural networks (6). In 2019, Lei Ji, Yingchao Zou, et al; proposed a hybrid ARIMA-CNN-LSTM model, where the ARIMA was used to capture the linear features, the Convolutional Neural Network (CNN) was used to capture the hierarchical data structure while the LSTM was used to capture the long-term dependencies in the data to forecast the scale of carbon emissions (2). In order to deal with conventional ANN problems of catastrophic forgetting, Parisi, R. Kemker, Part, Kanan, and Wermter; performed an empirical study on catastrophic forgetting and developed a protocol for setting hyperparameters and method evaluation, on two methods, namely Elastic Weight Consolidation and Incremental Moment Matching (IMM) (7). Further, Lange, Aljundi, Masana et al; explored continual learning algorithms in order to deal with the problem of catastrophic forgetting on task-incremental classification, where tasks arrived in a batch-like fashion, and were delineated by clear boundaries (4).

Meanwhile, the research work in the arena of application of machine learning and deep learning models in the field of financial time series is relatively recent. Baestaens, Van Den Bergh, and Vaudrey (1995) and Refenes, Zapranis, and Francis (1994), did some pioneering work in the field, using simple ANN architectures and displayed their performance relative to the logistic regression (LR) models (8)(9). Further, Ping-Feng Pai, Chih-Sheng Lin in 2003 proposed a hybrid approach with linear ARIMA models and support vector machines to capture the nonlinear patterns in data, in order to predict stock prices (10). Karim et al, investigated a time series sequence classification task using the augmentation of CNN with LSTM and RNN sub-modules, and concluded that they significantly enhance the performance with a nominal increase in model size and required minimal pre-processing of the data set (11). Additionally, Jian-Zhou, Wanga Ju-Jie, Wanga Zhe-George, Zhangbc, Shu-Po Guoa proposed a new approach of forecasting the stock prices via the Wavelet De-noising-based Back Propagation (WDBP) neural network which is particularly vital while examining highly noisy and non-stationary stock prices data (12). Further, Qiu and Song predicted the trend of Japanese stock market index by using an ANN model optimized by using genetic algorithms (GA) (13). Philps, Garcez and Weyde investigated the catastrophic forgetting phenomenon in LSTMs and proposed continual learning algorithm on sliding window base learner, Feed Forward Neural Network (FFNN), and a sequential base learner, LSTM(14).

3) Approaches: Methods & Tools for Design, Analysis & Evaluation

3.1) Data & Tools

- ➢ We will be using high-frequency daily data, to forecast the value of NASDAQ-100 a stock market index made up of 103 equity securities issued by 100 of the largest non-financial companies listed on the NASDAQ stock market. The variable we will forecast will be the average of the daily opening and closing price of the index.
- ➢ In order to study the impact of Covid-19 on the trajectory of the NASDAQ stock index, we will use input data since March 1st, 2020; and include daily values of
    - Covid-19 related deaths in US,
    - NASDAQ opening price,
    - NASDAQ closing price,
    - NASDAQ high,
    - NASDAQ low
    - NASDAQ volume;

  as input variables; for constructing multivariate time series models.
- ➢ We will critique and contrast 2 models on the aforementioned dataset, namely 1) LSTM, 2) CLA.
- ➢ For initial network design we have identified PyTorch, Keras (Python Tensorflow wrapper) as potential tools.

3.2) Model 1: LSTM

RNN's suffer with the drawback of handling historical time series data due to the problem of gradient descent or gradient explosion, to deal with which an improved cyclical version of RNN, called Long short-term memory (LSTM) was introduced. LSTM is suitable for processing and predicting data with gaps and delays in a time series. Below we briefly define the structure of LSTM.

When dealing with long run time series data, the series typically displays some discriminative patterns in different time periods called shifting patterns. For the purpose of our analysis we will be dealing with multi-variate sequential time series data. The shifting patterns of these series contains noise, in addition to discriminative patterns, hence requiring a highly sophisticated analysis. The shifting patterns in sequential data severely degrade the performance of traditional time series methods, for the propose of dealing with which; we will use Jia, Khandelwal, Karpatne and Vipin Kumar's (2019) novel sequence classification method by automatically mining shifting patterns from multivariate sequences (15).

Going forward, we aim to analyse a sliding time window to capture different time periods over the sequence, which is further combined with LSTM to model temporal dependencies in sequential data. By incorporating temporal relationships in sequential data, LSTM assists in better uncovering discriminative patterns within each time window. For each time window that starts from time t, the raw input features are represented within the time window as $x_t = \{z_t, z_{t+1}, ..., z_{t+w-1}\}$. Each time window t contains multi-variate features at several consecutive time steps, in order to extract more representative local patterns from each time window x_t. The hidden representations are denoted by h_t, which is generated via an LSTM cell using both the raw features x_t within the current time window and the information from previous time window. Then the latent output p_t is generated from h_t. The discriminative patterns typically follow specific temporal evolutionary process.

We now briefly introduce the architecture of the LSTM model (Fig1). Each LSTM cell contains a cell state c_t, which serves as a memory and allows the hidden units h_t to reserve information from the past. The cell state c_t is generated by combining c_{t-1} and the information at time, t. Hence, the transition of cell state over time forms a memory flow, which enables the modelling of long-term dependencies. A new candidate cell state C_t' is generated by combining x_t and h_{t-1} into a function, as:

$$C_t' = \tanh(W_h^c h_{t-1} + W_x^c x_t)$$

$$f_t = \delta(W_h^f\, h_{t-1} + W_x^f x_t)$$

$$g_t = \delta(W_h^g h_{t-1} + W_x^g x_t)$$

$\{W_h^f, W_x^f\}$ and $\{W_h^g, W_x^g\}$ denote two sets of weight parameters for generating forget gate layer f_t and input gate layer g_t, respectively. The forget gate layer is used to filter the information inherited from c_{t-1}, and the input gate layer is used to filter the candidate cell state at time t. In this way we obtain the new cell state c_t as:

$$c_t = f_t c_{t-1} + g_t c_{t-1}'$$

Finally, we generate the hidden representation at t by filtering the obtained cell state using an output gate layer o_t, as:

$$o_t = \delta(W_h^o\, h_{t-1} + W_x^o x_t)$$

$$h_t = o_t \tanh(c_t)$$

With the hidden representation h, we produce the latent output of each time window t using a sigmoid function with parameter U, as follows:

$$p_t = \delta p_t U h_t$$

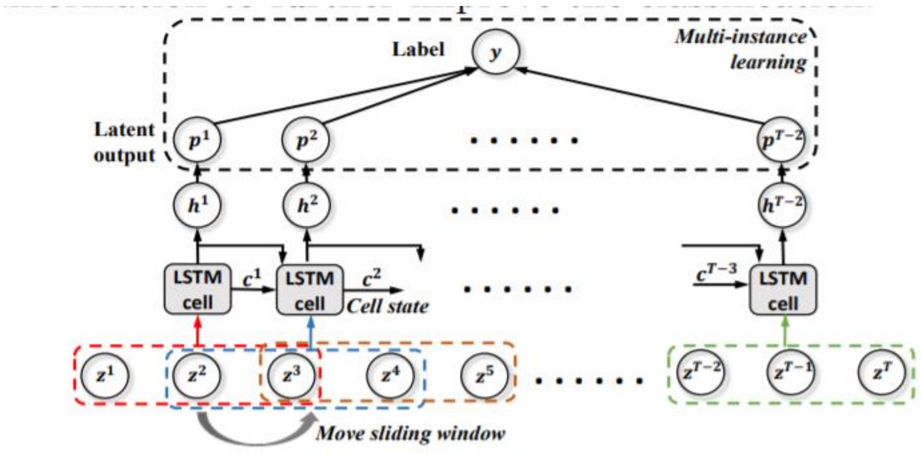


Fig1: The raw features from each sliding window is fed into LSTM structure, and the output structure is aggregated using the multi-instance learning approach.

3.3) Model 1: Setup

We will be using the minimum objective function as the optimization goal, the Adam optimization algorithm will be used to continuously update the weight of the neural network, and the network layer and batch size will be tuned to select the optimal one. The number of layers, length of the sliding window and batch size will be used as parameters of model.

3.4) Catastrophic Forgetting: From LSTM to Continual Learning

It is a tendency for knowledge of previously learnt history of the time series data to be lost abruptly, as new data is incorporated. This phenomenon, termed catastrophic forgetting occurs specifically when the network is trained sequentially on multiple tasks, because the weights in the network that are important for old data are updated to meet the objectives of new data. Machine learning models are static models incapable of adapting or expanding their behaviour over time. As such, the training process needs to be restarted, each time new data becomes available and the model needs an update (16). Research on artificial neural networks has focused mostly on static tasks, where the data is usually shuffled to ensure i.i.d. conditions, and performance largely increases with repeated revisiting of the training data over multiple epochs (19). Without special measures, artificial neural networks, trained with stochastic gradient descent, are subject to catastrophic forgetting, to circumvent which continual learning algorithms are used.

3.5) Model 2: Continual learning augmentation

Next we will examine, Philps, Garcez and Weyde's Continual learning augmentation (CLA) approach on our data, which is expected to augment conventional learner for time-series regression with memory and henceforth target the problem of catastrophic forgetting.

The memory functions of CLA are applied as a sliding window, of the input data of the multivariate time series X over t time periods. The approach is initialized with an empty memory structure, M, and a chosen base learner, φ, parameterized by θ_B. The chosen base learner produces a forecast value y'_{t+1}in each period as time steps forward. A remember-gate, j, appends a new memory, M_t^m , to M, which is input by the change in the base learner's absolute error at time-point t. A recall-gate, g, balances a mixture of current and memory-based forecasts to result in the final outcome, y'_{t+1}. Fig 2 shows the functional steps of remembering, recalling and balancing learner-memories.

Repeating patterns are required in time series of the input data to provide memory cues to remember and recall different past states. Learner parameters, θ_m, trained in a given past state can then be applied if that state approximately reoccurs in the future. Memory parameter of CLA changes in size over time as new memories are remembered and old ones forgotten. Each memory column consists of a copy of a past base learner parameterization, θ_m, and a representation, X_m, of the training data used to learn those parameters. As the sliding window steps into a new time period, CLA recalls one or more learner-memories by comparing the latest input data, X_t, with a representation of the training data stored in each memory column, X_m. Memories with training data that are more similar to the current input series will have a higher weight applied to their output, $y'_{m,t+1}$, and therefore make a greater contribution to the final CLA output, y'_{t+1}

Remembering is triggered by changes in the absolute error series of the base model, as the approach steps forward through time. These changes are assumed to be associated with changes in state which are indicated by a function j. j defines a change and stores a pairing of the parameterization of a base model, θ_B, and a contextual reference, X_t.

Immediately after the remember event has occurred, a new base model is trained on the current input, overwriting θ_B. Theoretically, for a fair model of a state, ε_B would be approximately i.i.d. with a zero valued mean. Therefore, the current base mode would cease to be a fair representation of the current state, when B exceeds a certain confidence interval, in turn implying a change in state. J_{Crit} represents a critical level for B, indicating a change point has occurred in state. Memories are only stored when the observed absolute error series, ε_B, spikes above a critical level, J_{Crit}. J_{Crit} is a hyperparameter, optimized at every time step, to result in a level of sensitivity to remembering that forms an external memory, M, resulting in the lowest empirical forecasting error for the CLA approach over the study term up until time T.

The recall of memories takes place in the function g, which calculates $y'_{m,t+1}$ a mixture of the predictions from the current base model and from model-memories y'_{t+1}= g(X_t,M_t). The mixture coefficients are based on comparing the similarity of the current time varying context X_t with the contextual references X_m stored with each individual memory. Memories that are more similar to the current context have a greater weight in CLA's final modelling outcome.

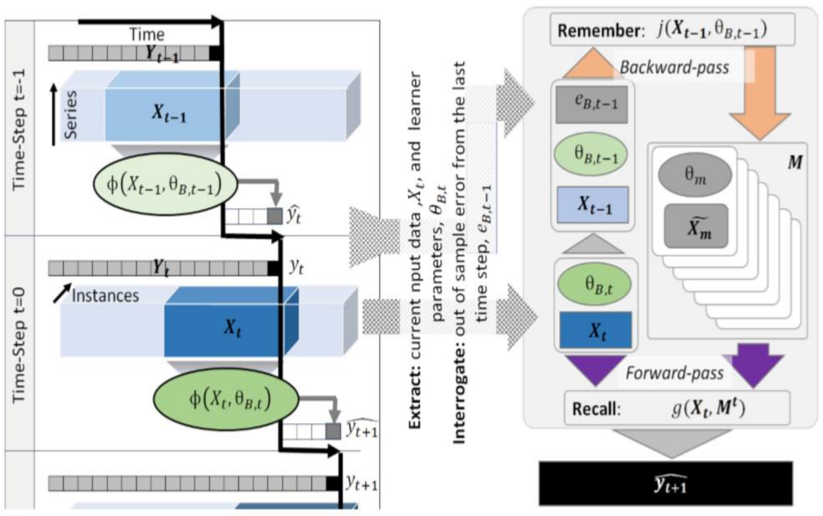


Figure 1: Continual learning augmentation architecture. Backward-pass: When y_t becomes observable, *remember-gate*, j, assesses $|\epsilon_{B,t-1}|$ for change and if change has occurred, extracts $\theta_{B,t-1}$ adding a memory column to M. Forward-pass: output of learner-memories and base learner are balanced by the *recall-gate*, g.

3.6) Model 2: Setup

The CL approach will be tested on sliding window base learner, FFNN, and a sequential base learner, LSTM. Different similarity approaches will be used to drive the memory recall-gate; namely, Euclidean distance (ED), dynamic time warping (DTW), auto-encoders (AE) and a novel hybrid approach, warp-AE (15). Base learners would be batch trained at each time-step, forecasting 3 months ahead daily values for the NASDAQ index.

3.7) Evaluation:

The two models will be compared, contrasted and evaluated. The hyper-parameter tuning of each of the two models will be done by performing k-fold cross validation on the validation data and the selected models will be tested on the hold out data. We will finally choose the better model using smaller test error, for the purpose of forecasting asset price index, in wake of the corona pandemic.

4) Work Plan

Project Timeline

		May	June					July				Aug					Sept			
S. No.	Task/Week Starting	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	7	14	21	28
1	Project Proposal	🟧																		
2	Reference Search		🟩																	
3	Literature Tools			🟩																
4	Literature Review			🟩																
5	Dataset Collection				🟦															
6	Dataset Augmentation					🟦														
7	Replicating Existing Models						🟩	🟩												
8	Creating Alternative Models								🟩	🟩	🟩									
9	Compare and comtrast models								🟩	🟩	🟩									
9	Model Refinement											🟪	🟪	🟪						
10	Evaluate Models														🟪	🟪	🟪			
11	Draft Report	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦			
12	Final Report																🟦	🟦		
13	Submit Final Report																		🟦	🟦

5) Risks:

- The capability of LSTM and CL models to perform well on non-stationary data is dubious, especially in the field of overly noisy and non-stationary time series real world stock market data, and thus, needs to be closely scrutinized.
- Both ANN and CL approaches lead to a loss of certain essential characteristics in classical statistics such as linearity, normality, homoscedasticity or observation independence, which can lead to biased results (5).
- Further, prediction of stock market indices is tricky since they are highly volatile and a lot of factors come into play like human behavioural expectations, which are hard to quantify.
- Additionally, the number of network layers of LSTM affects the learning ability, training time, and test time of the model. In theory, the deeper the LSTM layer, the stronger the learning ability. However, the deeper the model, the higher the complexity of the model, the more difficult it is to converge, and the more difficult and time consuming is the

training (18). Thus, the LSTM layers need to be intricately chosen to reach an effective optimum.

- Both LSTM and CL algorithms require a large dataset for time series tasks. In this regard, the efficacy of the models on three-month daily data (in wake of Covid-19 pandemic), remains a cause of concern.
- Determination of choice of window size for sequence learning is a complicated task. What periods preceding the current period must be chosen in order to explicate the time series needs to be intricately determined. In this regard, while time series Auto-Regressive regression can be a potent tool, it is far from comprehensive.

6) Ethical, Legal and Professional Issues:

This work complies with ethical regulations of Department of Computer Science, Mathematics and Engineering as outlined in the Part A: Ethics Checklist of the Research Ethics Review Form.

7) References:

1) Zhang, G., 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, pp.159-175.
2) Ji, L., Zou, Y., He, K. and Zhu, B., 2019. Carbon futures price forecasting based with ARIMA-CNN-LSTM model. *Procedia Computer Science*, 162, pp.33-38.
3) Singaravel, S., Suykens, J. and Geyer, P., 2018. Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction. *Advanced Engineering Informatics*, 38, pp.81-90.
4) Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, Tinne Tuytelaars, 2019, Continual learning: A comparative study on how to defy forgetting in classification tasks
5) Fathi, 2008, Time series forecasting using a hybrid ARIMA and LSTM model
6) Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, Honggang Zhang, 2017, Traffic Prediction Based on Random Connectivity in Deep Learning with Long Short-Term Memory
7) Parisi, G., Kemker, R., Part, J., Kanan, C. and Wermter, S., 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113, pp.54-71.
8) Baestaens, D., Den Bergh, W. and Vaudrey, H., 1995. Options as a predictor of common stock price changes. *The European Journal of Finance*, 1(4), pp.325-343.
9) Nicholas Refenes, A., Zapranis, A. and Francis, G., 1994. Stock performance modeling using neural networks: A comparative study with regression models. *Neural Networks*, 7(2), pp.375-388.
10) Pai, P. and Lin, C., 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, 33(6), pp.497-505.
11) Karim, F., Majumdar, S., Darabi, H. and Chen, S., 2018. LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access, 6, pp.1662-1669.
12) Wang, J., Wang, J., Zhang, Z. and Guo, S., 2011. Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*.
13) Qiu, M., Song, Y. and Akagi, F., 2016. Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market. *Chaos, Solitons & Fractals*, 85, pp.1-7.Making Good on LSTMs' Unfulfilled Promise
14) Daniel Philps, Arturd' Avila Garcez, Tillman Weyde, 2019, Making Good on LSTMs' Unfulfilled Promise
15) Xiaowei Jia, Ankush Khandelwal, Anuj Karpatne, Vipin Kumar V., 2019, Discovery of Shifting Patterns in Sequence Classification
16) Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. and Hadsell, R., 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), pp.3521-3526.
17) R. M. French, 1999 "Catastrophic forgetting in connectionist networks," Trends in cognitive sciences, vol. 3, no. 4, p, p. 128–135,
18) Zhang, T., Song, S., Li, S., Ma, L., Pan, S. and Han, L., 2019. Research on Gas Concentration Prediction Models Based on LSTM Multidimensional Time Series. Energies, 12(1), p.161.
19) Karim, F., Majumdar, S., Darabi, H. and Chen, S., 2018. LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access, 6, pp.1662-1669.

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part. The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**

2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**
2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**

A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.** **If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.**		*Delete as appropriate*
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**NO**

1. Introduction:

1.1. Background

On March 11, the World Health Organization characterized COVID-19 as a pandemic. With widespread commotion in personal lives and uncertainty of the future. Covid-19 has stirred tremendously feverish behaviour by investors worldwide. In such a situation studying asset price changes is particularly interesting as it gives us a unique opportunity to study the behaviour of equity markets and investor expectations.

In this study we will be using deep learning models in order to forecast the daily closing values of the Dow Jones Industrial Average index. The Dow Jones Industrial Average, is a stock market index that tracks 30 large, publicly-owned blue chip companies trading on the New York Stock Exchange (NYSE) and the NASDAQ. The DJIA is the second oldest U.S. market index and was designed to serve as a proxy for the health of the broader U.S. economy.[i] *We will study the daily global Covid-19 cases and examine the trajectory of Dow Jones Industrial Average in wave of the current pandemic, with a multivariate time series model.*

1.2. Research Questions

Stock market data is characterized by high level of volatility, is typically noisy and unstructured and displays evidence of non-stationarity. An important research question we seek to answer is the accuracy and efficiency of studying traditional deep learning neural network models like Long short-term memory vs learning algorithms characterized by the capacity control of the decision function, like support vector regression. We will particularly examine this in context of a multivariate time series data with relatively shorter time horizons.

In this regard, we seek to critique and evaluate Karmiani; Kazi; Nambisan; Shah; Kamble (2019) (1) and Chniti, Bakir, Zaher (2017) (2), that LSTM outperforms SVR for time series prediction problems.

Next we seek to evaluate Zhang; Wang; Liu; Bao (2018) (3) claim that LSTM is suitable and efficient in extracting the long term dependencies of data in case of a time series algorithm and the overfitting problem in LSTM can be countered by inclusion of the drop-out parameter.

We will further examine and critique Zhang , Song , Li, Li Ma, Pan and Han (2019) (4) hypothesis that the that the loss value of two LSTM layers is significantly lower than that of less that 2 or more than LSTM layers; and in case of time series forecasting, prediction with 2 LSTM layers has a result closest to the true data value and therefore considering the accuracy, fitting effect, and running time of the model, the prediction effect of the 2-layer LSTM model is most optimal. We will further evaluate their claim that with the increase in batch size beyond a certain point, while the time complexity is improved, the prediction accuracy of the model was compromised.

We will further evaluate Sauda and Shakya (2019) (5) smaller look back periods in case of LSTM time series prediction, far outperform larger look back periods.

Going forward, we will delve deeper into Kyoung-jae Kim(2002) (6) study that for financial time series prediction SVR with Gaussian kernel outperforms SVR with polynomial kernel.

Lastly, we will investigate L. J. Cao and Francis E. H. Tay (2003) (7) study for financial time series forecasting that that for SVR with RBF kernel, $\boldsymbol{\sigma}2$ that too small a value of (1–100) causes SVM to overfit the training data while too large a value of (10000–1 000 000) causes SVM to underfit the training data. An appropriate value for would be between 100 and 10 000. Further, they state that the performance of SVM is highly insensitive to the choice of ε.

2. Overview:

2.1. Dataset Description

We will be examining daily time series data from 31st December 2019 to October 29th 2020, for the purpose of our analysis. As per European Centre for Disease Prevention and Control, in the 10-month period, the Covid-19 cases have reached from 27 to 3.3mn, meanwhile the stock market prices of DJIA index have fallen by -4.75%.

Dataset Summary Statistics (Fig1)

	DJIA Daily Closing Vaue	Daily Covid-19 cases
count	189.00	189.00
mean	26165.99	8771984.00
std	2387.00	10026400.00
min	19862.99	27.00
0.25	24118.98	115820.00
0.50	26513.41	4396614.00
0.75	28175.19	15202430.00
max	29371.56	33423470.00

2.2. Dataset Examination

2.2.1 Time Series Analysis

Time-oriented data can sometimes be very noisy, given the fact that it is time-based, and data variations can happen within less than a second. Whether it is minimal or large variations, over an extended period; it can become chaotic and hard to draw conclusions from. Further, with most time series there is hardly a prior hypothesis available and time series it is often ridden with cyclicality, trend and seasonality. To draw effective conclusions and forecasts, accounting for these complications is vital.

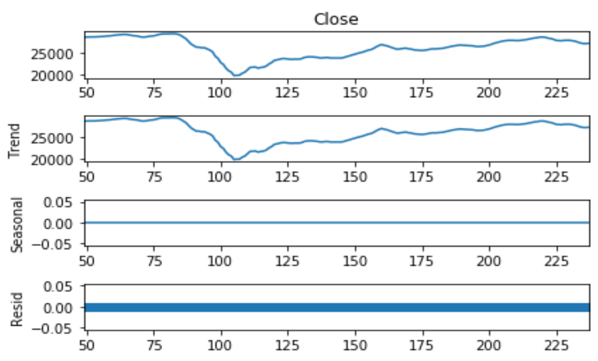
In order to develop a forecasting model, it is imperative that the time series is stationary. A stationary time series process has long term mean and variance independent of time. Non stationary time series are spurious, and need to be transformed into stationary data before being modelled. In this regard, we examined the DJIA index in the ten month period under investigation.

1) We conducted the Augmented Dicky Fuller (ADF) test in order to determine the stationarity of the Dow Jones Industrial average index in the ten-month period under study. The ADF is a unit root test, that determines how strongly a time series is defined by a trend. The ADF test statistic returned a value of -2.5 for the DJIA series, thus we fail to reject the null hypothesis that time series has a unit root (it is non-stationary) at 1% level of significance; however, we can reject the null at 10% level of significance. Thus, the series needs to be investigated further before drawing meaningful conclusions.
2) A time series can be decomposed into base level, trend, seasonality and error. A time series can be modelled as:

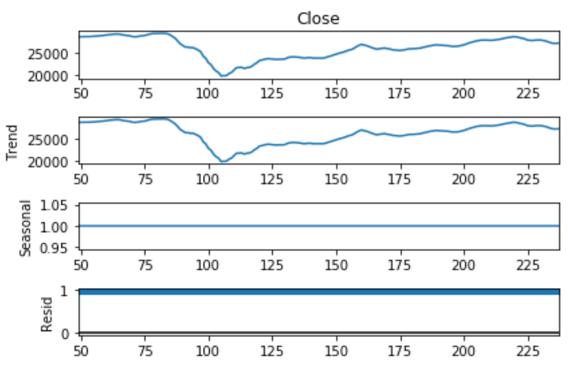
   - **Additive time series:**
     Value = Base Level + Trend + Seasonality + Error
   - **Multiplicative Time Series:**
     Value = Base Level x Trend x Seasonality x Error

3) When we view the plots of additive and multiplicative decomposition (Fig2,) we observe that the DJIA index in the ten-month period was entirely extracted as the trend component and there was no evidence of seasonality. Further, the residuals in additive case were centred around 0, thus the naïve decomposing couldn't separate the error or noise from the series. However, the residuals in multiplicative case were centred around 1, whereby in this case the noise could be extracted from the model.

Additive Decomposition of DJIA index (Fig2)



Multiplicative Decomposition of DJIA index (Fig3)



2.2.2 Time Series Inference

2.2.2.1 Model Architecture

To delve deeper in the properties of the DJIA index, we began with the seminal landmark Box -Jenkins (1980) Auto-Regressive Integrated Moving Average ARIMA approach which is an iterative model building process to select an adequate model for time series data. With classical Ordinary Least Squares (OLS) regression, it is not possible to explain the behaviour of a time series, since OLS merely takes into account the impact of independent variables on the dependent variable. However, time series is likely to be influenced by its own past values.

ARIMA(p,d,q) model is a linear regression model, whereby an observation is a weighted sum of its own lagged values and forecasted errors. The ARIMA forecasting equation for a stationary time series is a linear (i.e. regression-type) equation in which the predictors consist of lags of the dependent variable and/or lags of the forecast errors.

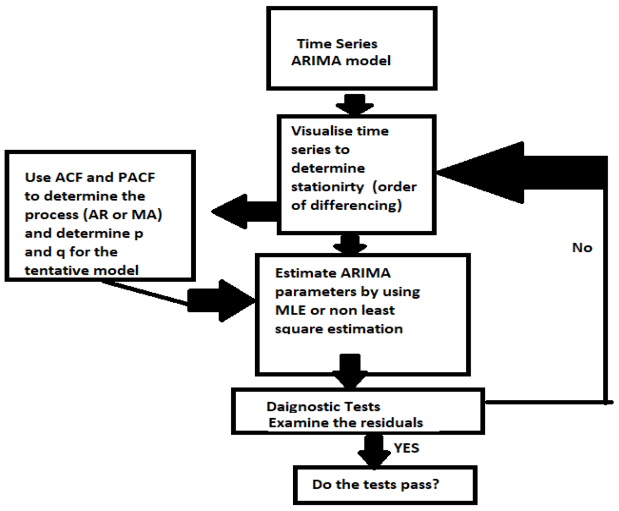
A nonseasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

- p is the number of autoregressive terms,
- d is the number of nonseasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation

Model Selection: In proceeding with ARIMA model, the first step is to examine the visual plot of the series, which indicates the trend, seasonality and cyclicality component. This decides the order of differencing to make the series stationary. This is followed by visualizing the Auto-Correlation and Partial Autocorrelation Functions of the time series data which help us decide *If any, AR and MA process should be used in the model.*

Parameter Estimation: Computationally arriving at coefficients that best fit the ARIMA model (like maximum likelihood estimation or non-linear least-squares estimation)

Model diagnostics*:* Lastly diagnostics tests which are determined by examining the residuals. It's an iterative process, if diagnostic tests fail, we have to return to step one and attempt to build a better model.


Time Series ARIMA model
Visualise time series to determine stationirty (order of differencing)
Use ACF and PACF to determine the process (AR or MA) and determine p and q for the tentative model
Estimate ARIMA parameters by using MLE or non least square estimation
No
Daignostic Tests Examine the residuals
YES
Do the tests pass?

2.2.2.2 Model Selection

Next, we studied the Auto Corelation Function (ACF) of the DJIA index closing values in the ten-month period. The ACF defines how the correlation between any two values of the time series changes as their separation changes. It is a time domain measure of the stochastic process memory. Generally, for an error signal, et, the ACF is defined as[ii]

$$\rho_k = \sqrt{\frac{Cov(e_t e_{t+k})}{var(e_t) var(e_{t+k})}}$$

For a stationary stochastic process of variance σ^2, the previous expression for the ACF reduces to the equation below, since the variance of e_t is independent of time and constant across all time periods.

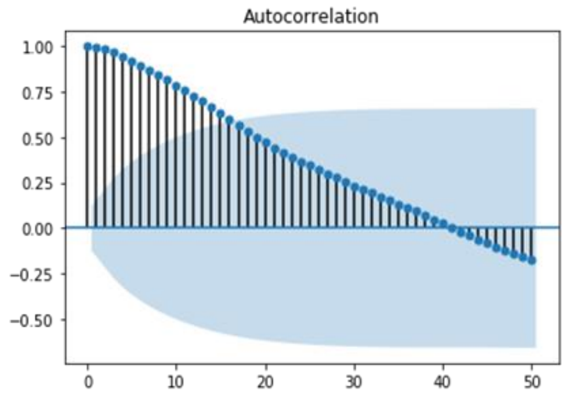
$$\rho_{k=}\frac{Cov(e_t)}{\sigma}$$

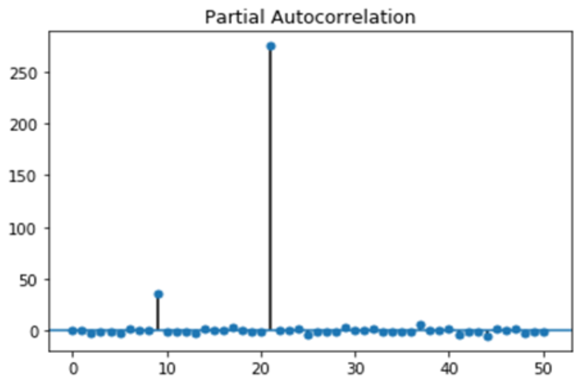
1) *Auto-Correlation Function*

On plotting the autocorrelation function (Fig 4) we see that the function shows correlation with past 14 lagged values (dependence on its own past values to the tune of 14th order) since first 14 values are statistically significant and the magnitude of corelation is decreasing with each successive time period.

ACF of DJIA index shows corelation till the 14th lag (Fig4)

PACF of DJIA index shows at the 9th & 21st lag (Fig5)





2) *Partial Auto-Correlation Function*

The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags, in contrast to ACF which doesn't account for correlation at shorter previous lags. iii For PACF we see the 9th and 21st lag statistically significant, showing direct relationship between observation at period 0 and its 9th and 21st lagged value for the DJIA index in the ten month period. -

2.2.2.3 Parameter Estimation

In our DJIA index time series data we noticed ACF with significance till 14th order, and PACF with correlation at the 9th and 21st lag, respectively. This implied that an observation in the series was correlated with its 14 past values, with declining correlation as the lag values were increasing/ decreasing in time from current period. Further, the magnitude of correlation with values after 9th was very minimal.

Thus, we opted for 9 autoregressive terms in the model. Additionally, when the series was first differenced, the ADF test showed evidence of stationarity at 1% level of significance. Thus, the order of integration was chosen as 1. Hence, we fit the ARIMA (9,1,0) for the DJIA index.

- *The ARIMA model showed statically significant correlation with the first, second, third, sixth and seventh lagged values.*
- *While the 3rd and 6th lagged values showed a negative impact, the 1st, 2nd and 7th lagged values showed a positive impact on the current time period.*

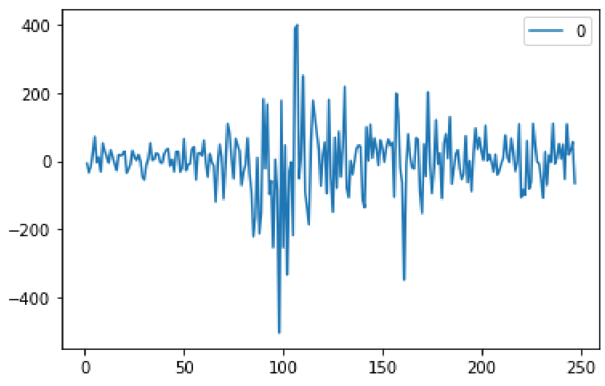
Hence the DJIA index period t is as follows:

$$\boldsymbol{DJIA_t = 0.7 * DJIA_{t-1} + 0.4 * DJIA_{t-2} - 0.3 * DJIA_{t-3} - 0.5 * DJIA_{t-6} + 0.5 * DJIA_{t-6} + \varepsilon_t}$$

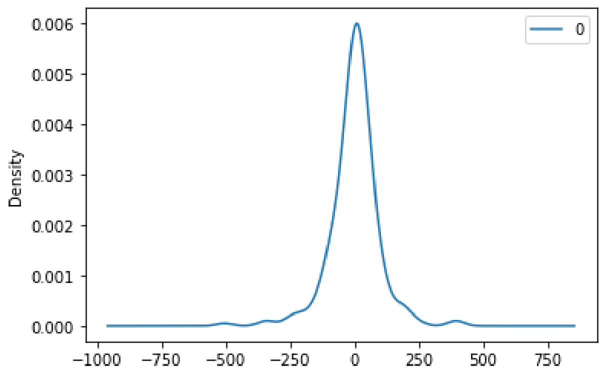
2.2.2.4 Model Diagnostics

- The scatter plot of residuals against the predicted values were randomly scattered around 0, indicating our model's predictions are correct on an average (Fig 6).
- It is imperative for the residuals of the model to show evidence of normality with e~N(0,1); in order to meet the assumptions of the ARIMA model. The kernel density of residuals showed normality centred at 0 (Fig 7).
- Next, we checked residuals for serial correlation in order to determine if there was any leftover trend in the residuals, which would imply that our model was unable to successfully capture the pattern of the time series. In this regard, we plotted the ACF of the residuals, which showed no correlation, and henceforth the model had captured the characteristic of the time series (Fig 8).
- This was validated by the p value of the Ljung-Box test statistic> 0.05, whereby we failed to reject the null, H0, is that our model does not show lack of fit.

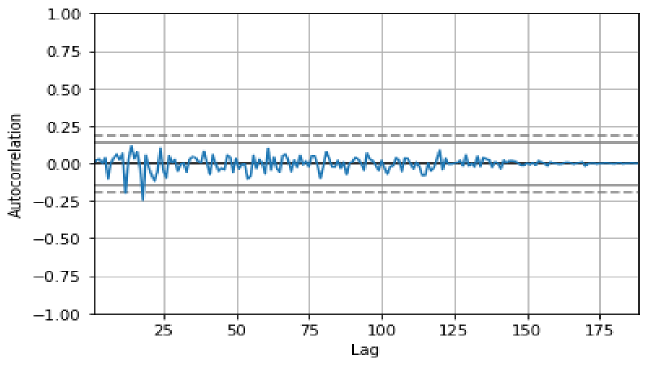
Scatter plot of residuals scattered around 0 randomly (Fig 6)



Residuals showed evidence of normality (Fig 7)



ACF of the residuals, which showed no correlation (Fig 8)



2.2.2.5 Limitations of ARIMA model

However, a major limitation of this model is its high error rate as soon as normality and/or linearity in the data is lost. Further, ARIMA concentrates on the values of the past series data and is unable to capture the dynamism in the time series process. Additionally, we were unable to capture the direct disruption cause by Covid-19 related cases, since ARIMA didn’t offer the necessary tools to fit a multivariate model.

Thus, for the purpose of our model we will explore the more recent artificial intelligence approaches like Recurrent Neural Networks (RNN) and compare them with traditional approaches like SVM. The particularity of neural networks is their tendency to adapt to the features created by the data itself and capability to retain long term dependencies in the time series data (8). Going forward, we will compare and contrast these latest neural network approaches like LSTM with traditional machine learning techniques like Support Vector Regressions. The results would be evaluated in the context of shorter sample time series data, on basis of different metrics.

3. Critical Context

In the field of time series modelling, there has been an extensive range of research work. Oussama Fathi, proposed a hybrid ARIMA and Long Short Term Memory network (LSTM) approach whereby ARIMA model is expected to capture the seasonal peaks in the data and LSTM is expected to capture the stable part of the time series; for the purpose forecasting US dollar/British pound exchange rate data (9). Hua, Zhao, Li, et al; proposed the Random Connectivity LSTM (RCLSTM) model to reduce the considerable computational cost of LSTM on traffic and user mobility in telecommunication networks. As opposed to LSTM, RCLSTM was formed via stochastic connectivity between neurons, which achieved a significant breakthrough in the architecture formation of neural networks (10). In 2019, Lei Ji, Yingchao Zou, et al; proposed a hybrid ARIMA-CNN-LSTM model, where the ARIMA was used to capture the linear features, the Convolutional Neural Network (CNN) was used to capture the hierarchical data structure while the LSTM was used to capture the long-term dependencies in the data to forecast the scale of carbon emissions (11). In order to deal with conventional ANN problems of catastrophic forgetting, Parisi, R. Kemker, Part, Kanan, and Wermter; performed an empirical study on catastrophic forgetting and developed a protocol for setting hyperparameters and method evaluation, on two methods, namely Elastic Weight Consolidation and Incremental Moment Matching (IMM) (12). Further, Lange, Aljundi, Masana et al; explored continual learning algorithms in order to deal with the problem of catastrophic forgetting on task-incremental classification, where tasks arrived in a batch-like fashion, and were delineated by clear boundaries (13). Meanwhile, the research work in the arena of application of machine learning and deep learning models in the field of financial time series is relatively recent. Baestaens, Van Den Bergh, and Vaudrey (1995) and Refenes, Zapranis, and Francis (1994), did some pioneering work in the field, using simple ANN architectures and displayed their performance relative to the logistic regression (LR) models (14)(15). Further, Ping-Feng Pai, Chih-Sheng Lin in 2003 proposed a hybrid approach with linear ARIMA models and support vector machines to capture the nonlinear patterns in data, in order to predict stock prices (16). Karim et al, investigated a time series sequence classification task using the augmentation of CNN with LSTM and RNN submodules, and concluded that they significantly enhance the performance with a nominal increase in model size and required minimal pre-processing of the data set (17). Additionally, Jian-Zhou, Wanga Ju-Jie, Wanga Zhe-George, Zhangbc, Shu-Po Guoa proposed a new approach of forecasting the stock prices via the Wavelet De-noising-based Back Propagation (WDBP) neural network which is particularly vital while examining highly noisy and non-stationary stock prices data (18). Further, Qiu and Song predicted the trend of Japanese stock market index by using an ANN model optimized by using genetic algorithms (GA) (19). Philps, Garcez and Weyde investigated the catastrophic forgetting phenomenon in LSTMs and proposed continual learning algorithm on sliding window base learner, Feed Forward Neural Network (FFNN), and a sequential base learner, LSTM (14).

Support Vector Regression has also been a popular choice of algorithm for time series forecasting, especially before the development of relatively newer deep learning algorithms like LSTM. Huang, Nakomori and Yuang (2004) (20) compared and contrasted Support vector machine (SVM) with Linear Discriminant Analysis, Quadratic Discriminant Analysis and Elman Backpropagation Neural Networks by forecasting the weekly movement direction of

NIKKEI 225 index. Their study displayed that SVM outperformed other methods for a classification problem. we compare its performance. Karmiani, Kazi , Nambis , Shah and Vijaya Kamble (2019) (21)provided a comparative analysis between LSTM and SVR algorithms on the basis of accuracy, variation and time required for different number of epochs. They displayed that if the requirement of the model was high accuracy and low variance, LSTM would be a better choice but it is comparatively slower. Zhixi Li; Vincent Tam (2017)(22) studied the price movements of stocks in China and used different performance measures of the LSTM and SVM on a range of stocks exhibiting different volatilities. Their empirical results displayed that the overall performance of the original SVM excels on all stocks and also stocks of low volatility, whereas the original LSTM consistently achieves the best overall performance of high-volatility stocks in the SSE 50 Index. Cao and Tay (2003) (23) first examined the feasibility of applying SVM in financial forecasting by comparing it with the multilayer back-propagation (BP) neural network and the regularized radial basis function (RBF) neural network. They further investigated the variability in performance of SVM with respect to the free parameters. Their simulation showed that among the three methods, SVM outperformed the BP neural network in financial forecasting. Mellit & A. Massi Pavan & M. Benghanem (2013) (24) examined an application of least squares support vector machine (LS-SVM) for short-term prediction of meteorological time series. A comparison between LS-SVM and different artificial neural network (ANN) architectures (recurrent neural network, multi-layered perceptron, radial basis function and probabilistic neural network) was discussed. They displayed that LS-SVM produced significantly better results than ANN architectures.

4. Description of choice of training and evaluation methodology

- 80% data was used as training data and 20% data was used as hold-out validation data.
- Time series data is riddled with temporal dependencies, thus chronological order of the data must be maintained while fitting the model. Therefore, instead of using k-fold cross validation, we have used hold-out cross-validation where the hold-out data (split temporally) was used the evaluate the model.
- The independent variables in our analysis are the period t closing value of the DJIA index and period t Covid-19 cases. Meanwhile, the independent variable is period t+1 closing value of the DJIA index (basis daily data in the ten-month period).
- We used three day moving average of the time series, in order to is to help smooth out the DJIA index over the ten-month period of time, by creating a constantly updated average price.
- The series didn't display any evidence of outliers, possibly since a short time duration was under study (from onset of Covid-19 cases).
- For the purpose of our data, we used the standard min-max scalar form of normalization which used the mean and standard deviation to fit the data series into a [0,1] range lying between the minimum and maximum value of the DJIA index.

4.1. Model 1: LSTM Framework

RNN's suffer with the drawback of handling historical time series data due to the problem of gradient descent or gradient explosion, to deal with which an improved cyclical version of RNN, called Long short-term memory (LSTM) was introduced. LSTM is suitable for processing and predicting data with gaps and delays in a time series. Below we briefly define the structure of LSTM. The LSTM setup most commonly used in literature was originally described by Graves and Schmidhuber.

When dealing with long run time series data, the series typically displays some discriminative patterns in different time periods called shifting patterns. For the purpose of our analysis we will be dealing with multi-variate sequential time series data. The shifting patterns of these series contains noise, in addition to discriminative patterns, hence requiring a highly sophisticated analysis. The shifting patterns in sequential data severely degrade the performance of traditional time series methods, for the purpose of dealing with which; we will use Jia, Khandelwal, Karpatne and Vipin Kumar's (2019) novel sequence classification method by automatically mining shifting patterns from multivariate sequences (25).

Going forward, we aim to analyse a sliding time window to capture different time periods over the sequence, which is further combined with LSTM to model temporal dependencies in sequential data. By incorporating temporal relationships in sequential data, LSTM assists in better uncovering discriminative patterns within each time window. For each time window that starts from time t, the raw input features are represented within the time window as $x_t = \{z_t, z_{t+1}, ..., z_{t+w-1}\}$. Each time window t contains multi-variate features at several consecutive time steps, in order to extract more representative local patterns from each time window x_t. The hidden representations are denoted by h_t, which is generated via an LSTM cell using both the raw features x_t within the current time window and the information from previous time window. Then the latent output p_t is generated from h_t. The discriminative patterns typically follow specific temporal evolutionary process.

We now briefly introduce the architecture of the LSTM model (Fig8). Each LSTM cell contains a cell state c_t, which serves as a memory and allows the hidden units h_t to reserve information from the past. The cell state c_t is generated by combining c_{t-1} and the information at time, t. Hence, the transition of cell state over time forms a memory flow, which enables the modelling of long-term dependencies. A new candidate cell state $C_t'$is generated by combining x_t and h_{t-1} into a function, as:

$$C_t' = \tanh(W_h^c h_{t-1} + W_x^c x_t)$$

$$f_t = \delta(W_h^f\, h_{t-1} + W_x^f x_t)$$

$$g_t = \delta(W_h^g h_{t-1} + W_x^g x_t)$$

$\{W_h^f, W_x^f\}$ and $\{W_h^g, W_x^g\}$ denote two sets of weight parameters for generating forget gate layer f_t and input gate layer g_t, respectively. The forget gate layer is used to filter the information inherited from c_{t-1}, and the input gate layer is used to filter the candidate cell state at time t. In this way we obtain the new cell state c_t as:

$$c_t = f_t c_{t-1} + g_t c_{t-1}'$$

Finally, we generate the hidden representation at t by filtering the obtained cell state using an output gate layer o_t, as:

$$o_t = \delta(W_h^o\, h_{t-1} + W_x^o x_t)$$

$$h_t = o_t \tanh(c_t)$$

With the hidden representation h, we produce the latent output of each time window t using a sigmoid function with parameter U, as follows:

$$p_t = \delta p_t U h_t$$

Fig8: Raw features from each sliding window is fed into LSTM structure, and the output structure is aggregated using the multi-instance learning approach

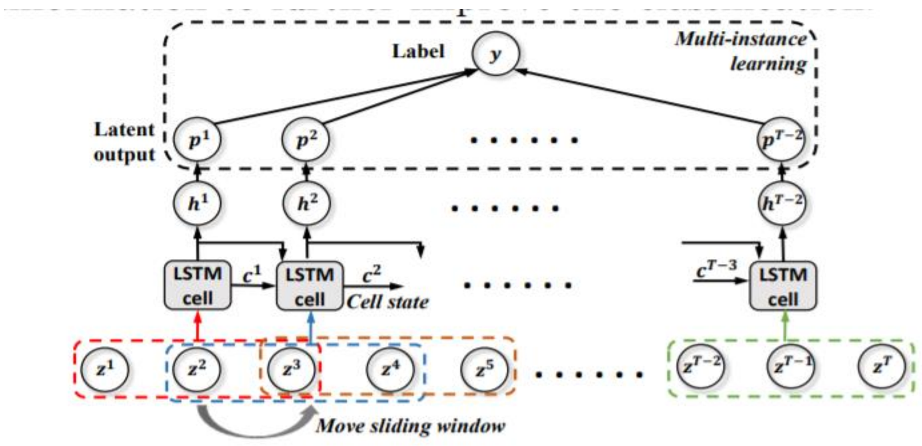
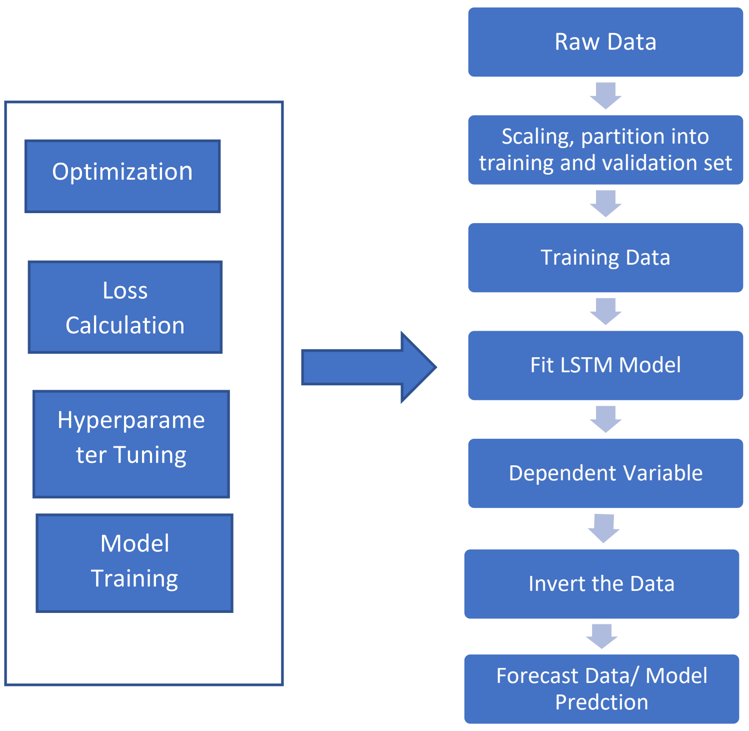


Fig 9: Methodology of LSTM implementation



4.1.1. Advantages of LSTM:

In conventional neural network setting the gradient of the loss function is scaled during back propagation and as a result, the gradient either increases or declines exponentially with time. The gradient has a tendency to (1) blow up or (2) vanish: with successive time intervals. The temporal evolution of the backpropagated error exponentially depends on the size of the weights (Hochreiter 1991). Case (1) may lead to oscillating weights, while in case (2) learning to bridge long time lags takes a prohibitive amount of time, or does not work at all. Thus, the gradient either dominates the next weight adaptation step or effectively gets lost.

To avoid this scaling effect, LSTM corresponds to a scaling factor that is fixed to one.iv The LSTM overcomes the exploding or vanishing gradient problems that plague the conventional neural networks. The LSTM enforces an algorithm whereby constant (thus neither exploding nor vanishing) error flow through internal states of the gates defined above. It does so by keeping the error flow constant through the gates which allows for weights adjustments as well as truncation of the gradient when its information is not necessary.

Further, LSTM is adept to handle non stationary noisy data. Long Short-Term Memory (LSTM) are capable of identifying structure and pattern of data such as non-linearity and complexity in time series (27).

4.1.2. LSTM Architecture on choice of our data

1) Optimiser

Typically for deep learning models, gradient-descent is an efficient method of optimisation if the loss function is differentiable when computing the partial derivates of the loss function with its parameters; since the computation of first-order partial derivatives w.r.t. all the parameters is of the same computational complexity as just evaluating the function. Gradient descent converges to the global minimum for convex error surfaces and typically to local minimum for non-convex surfaces. However, since the gradient of the whole dataset needs to be computed for each epoch batch gradient descent is computationally expensive.

Often, objective functions are stochastic. For example, many objective functions are composed of a sum of subfunctions evaluated at different subsamples of data; in this case optimization can be made more efficient by taking gradient steps w.r.t. individual subfunctions, i.e. stochastic gradient descent (SGD) or ascent.v As discussed above, batch gradient descent is computationally very inefficient due to successive iterations on the entire dataset, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time(27).

However, objectives may also have other sources of noise than data subsampling, such as dropout, noise, non-stationarity (Hinton et al., 2012) and therefore more efficient optimisation algorithms needed to be examined.

For the purpose of our study, we used the *Adaptive Moment Estimation (Adam)* method of stochastic optimization. Adam combines advantages of two novel optimisation algorithms namely, Adaptive Gradient Algorithm (AdaGrad) that incorporates a learning rate for each parameter and thereby is efficient for problems with sparse gradients and Root Mean Square Propagation (RMSProp) which incorporates learning rate of each parameter that learns basis average of recent magnitudes of the gradients for the weight and henceforth is efficient on noisy, nonstationary data and on problems involving dropouts.

Adam optimization only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Adam optimization is highly advantages since the magnitudes of parameter updates are invariant to rescaling of the gradient. Adam does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing (27). Further, Kingman and Ba (2017) highlighted that on stochastic regularization methods, such as dropout which are used to prevent over-fitting, adam showed better convergence than all other stochastic first order methods on multi-layer neural networks trained with dropout noise.

2) Learning Rate

Learning rate is not a hyper parameter to considered in our model. Learning rate decay is typically used as a hypermeter in order to enable the cost function to converge at a global minimum. Adam ensures efficient convergence by choice of two hyperparameters α (step size) and β1, β2 ∈ [0, 1), (exponential decay rates for the moment estimates). Adam improves gradient descent optimization by calculating individual learning rates for each parameter at the current step in descent based on the given learning rate. Adam calculates a set of learning rates that adapt better to the immediate surroundings and optimizes better as a result. Thus, given inbuilt structure of Adam, learning rate was not required as a hyper-parameter (27), therefore learning rate is not prudential as a parameter to be tuned in our model.

Adam optimiser adapts the step sizes on the way of gradient descent without actually changing or decaying the learning rate. Therefore, a reduction in α merely would change the step size of convergence without changing the learning rate. Adam helps learning rate decay by the sense of adapting each step size along the way of convergence. Therefore, Adam indirectly helps learning rate decay (without changing the learning rate) reach a point of convergence by adapting each step along the way.

3) Loss Function

We use the standard Mean Square Error (MSE) for error back propagation of the LSTM. The MSE is calculated as the average of the squared differences between the predicted and actual values. The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0. The squaring means that larger mistakes result in more error than smaller mistakes, meaning that the model is punished for making larger mistakes.

The Gaussian probability density function is determined only by its first- and second-order statistics, and the effect of linear systems on low order statistics is well known. Under the linearity and Gaussianity assumptions, further supported by the central limit theorem, MSE, which solely constrains second-order statistics, would be able to extract all possible information from a time series whose statistics are solely defined by its mean and variance (27). The Dow Jones Industrial average index showed a Gaussian distribution, and hence we chose MSE as the preferred loss function under the inference framework of maximum likelihood. While, MSE is sensitive to outliers our data didn't display evidence of outliers.

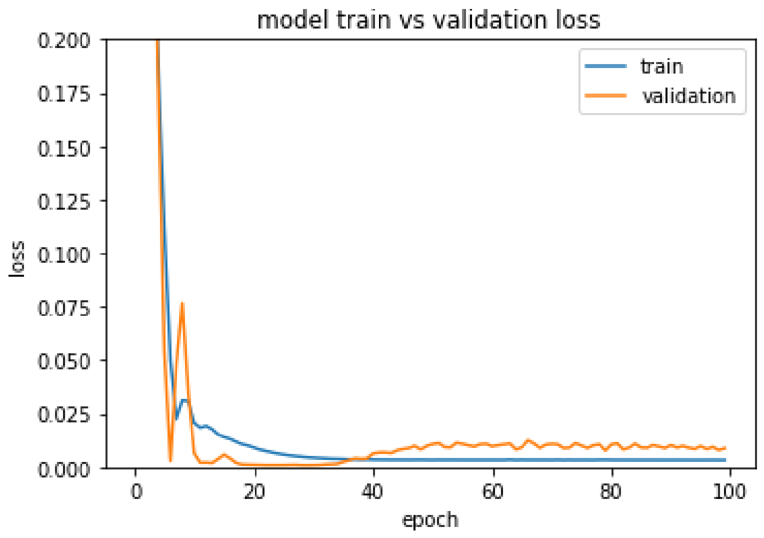
4.1.3. Choice of Hyper-Parameters in our LSTM Model

We used a grid search algorithm for the purpose of determining the most efficient model. The metric of choice we used for testing the optimality of the model was the root mean square error (RMSE) on the hold-out validation data.

- **Look-Back:** Number of time steps in the past, impacting the current period values [t-1, t-2]; [t-1, t-2, t-3]; [t-1, t-2, t-3,t-4]
- **Number of LSTM layers:** [1,2]
- **Number of units in each layer:** [5,10,20,50]
- **Drop out %:** [0, 0.01, 0.001]

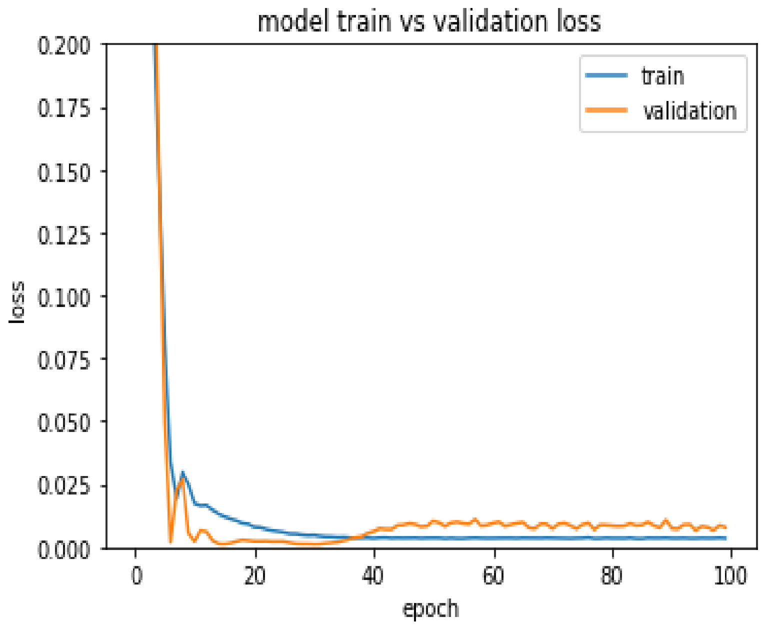
- The results were trained for 100 epochs and evaluated on the 20% holdout data. First grid search was carried out on the given hyper parameters. *The validation error was minimised at root mean square error of 826, with 2 look-back periods, 2 layers of 50 neurons each; and 0 drop-out (Fig 10).* However, as can be seen in the figure, the model displayed evidence of overfitting. We saw after roughly 37 epochs, the validation loss began to rise again, and showed excessive instability with periods of falling below training loss.

Fig 10: Validation Loss showed evidence of overfitting



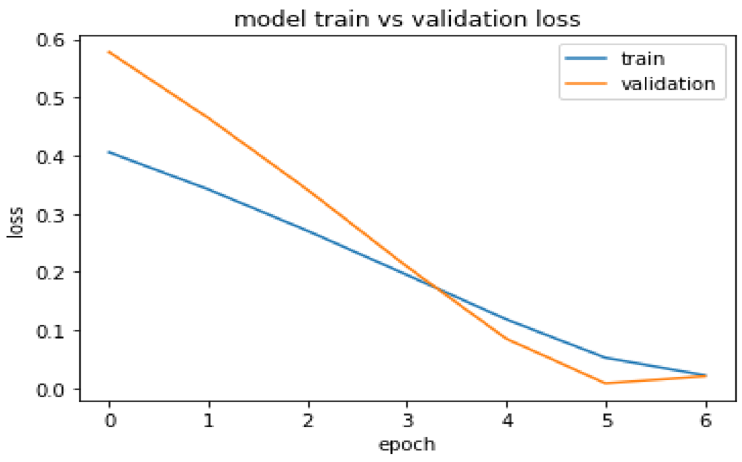
- From our grid search, we then went on to investigate the next model, involving low validation error which displayed drop out of 0.1, and henceforth, we believed it might help us alleviate the overfitting issue. With the inclusion of drop-out parameter, both training and validation error declined, and hence **we agree with Zhang; Wang; Liu; Bao (2018) claim drop-out parameter helped reduce overfitting, and is not responsible for loss of vital information in the LSTM model**. However, on investigating the graph the trajectory of training and validation loss displays we still need to further examine the model and drop-out was not completely eliminated.

Fig 11: Introducing drop out led to a loss in error, but still showed overfitting



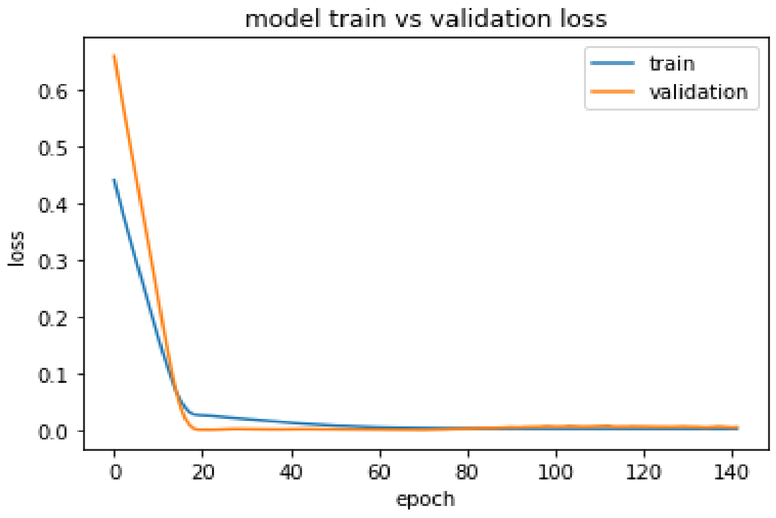
- In the last model we saw, that at the 35th epoch validation error was at the minimum, however, soon after the error began to rise. Henceforth, the aforementioned model was re-trained using an early stopping criterion, whereby as validation loss starts rising we stopped training the model Fig 12. Reviewing the graph below, we saw that training was stopped at the point when validation loss began to plateau for the first time. However, the training stopped at a local minimum of 6 epochs and if we continued training till 35th epoch, the validation loss could have been further minimised.

Fig 12: Early stopping led to underfitting



- Thus the hyper parameter grid search was recomputed (results below) with training at 200 epochs, with early stopping criterion implemented with the patience argument at 75 epochs, which implied that we would wait 75 additional epochs for model to continue training after validation loss began to plateau, giving the training additional time to achieve further minim of validation loss and seek additional improvement

Fig 13: Early Stooping with patience argument led to additional improvement



4.1.4. Grid Search Hyperparameter LSTM Results

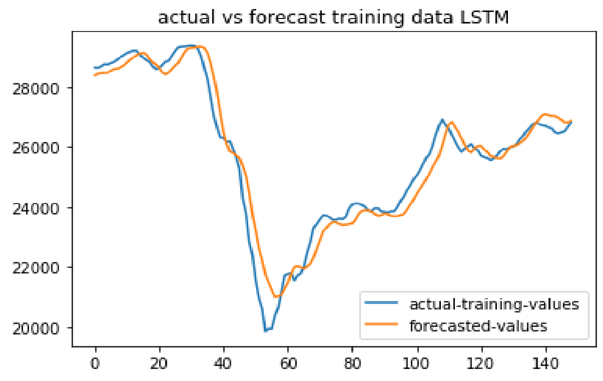
Look Back	Layer 1- units	Layer 2- units	Drop-out	Train RMSE	Validation RMSE	Look Back	Layer 1-units	Layer 2-units	Drop-out	Train RMSE	Validation RMSE
2	50	0	0.01	503.2	789.6	2	50	5	0	480.6	1290.9
2	50	0	0.001	496.9	920.2	2	10	5	0.001	507.3	1316.8
2	20	0	0.01	495.0	938.8	2	50	5	0.001	486.6	1317.2
2	50	0	0	493.6	948.5	3	20	5	0.01	554.9	1337.8
3	20	0	0.01	565.6	1055.7	3	20	5	0.001	565.8	1350.7
2	10	0	0.001	492.9	1058.0	3	5	5	0.001	555.1	1428.1
3	20	0	0	566.4	1059.6	3	20	5	0	558.6	1458.5
2	20	0	0.001	501.6	1071.4	2	10	5	0	528.4	1489.7
3	20	0	0.001	548.3	1203.1	3	5	5	0	641.9	1508.0
2	20	0	0	544.7	1234.8	3	50	5	0.01	542.0	1562.0
3	5	0	0.01	550.8	1337.6	3	50	5	0.001	551.5	1596.5
3	10	0	0	533.2	1349.6	3	50	5	0	555.6	1627.9
3	50	0	0.01	538.3	1379.3	4	5	5	0.001	688.3	1691.6
3	50	0	0	535.5	1380.8	3	10	5	0	546.9	1723.9
2	5	0	0.01	468.1	1393.9	3	5	5	0.01	589.5	1728.5
3	50	0	0.001	516.7	1458.6	3	10	5	0.01	569.7	1776.3
4	20	0	0	604.2	1476.5	4	5	5	0.01	646.4	1805.8
3	10	0	0.001	556.4	1559.5	4	10	5	0.01	616.8	1815.0
4	50	0	0.01	547.9	1589.2	4	10	5	0.001	602.9	1847.3
4	20	0	0.001	525.1	1604.3	4	20	5	0.001	608.4	1865.0
3	10	0	0.01	531.2	1618.1	4	50	5	0	596.3	1866.0
4	20	0	0.01	565.1	1666.4	4	50	5	0.001	569.6	1870.2
4	10	0	0	602.3	1735.6	4	20	5	0.01	600.9	1891.8
4	10	0	0.001	548.1	1743.7	3	10	5	0.001	574.4	1901.0
4	50	0	0	516.7	1779.1	2	5	5	0.001	525.8	1929.9
4	10	0	0.01	598.3	1798.0	4	5	5	0	644.7	1995.9
4	50	0	0.001	544.9	1823.8	4	50	5	0.01	595.0	1996.8
3	5	0	0.001	539.9	1859.8	4	10	5	0	611.4	2050.3
4	5	0	0.01	667.5	1986.8	4	20	5	0	612.6	2391.3
2	10	0	0.01	587.3	2005.1	2	10	10	0.001	550.5	751.0
2	10	0	0	580.6	2150.7	2	20	10	0.01	516.7	872.4
4	5	0	0	611.8	2204.9	2	20	10	0.001	500.1	908.8
4	5	0	0.001	674.2	2271.3	2	10	10	0.01	521.2	936.6
2	5	0	0	618.3	2405.8	2	50	10	0.01	493.5	950.3
2	5	0	0.001	683.7	2509.2	2	10	10	0	501.5	961.1
3	5	0	0	1074.4	4883.9	2	20	10	0	493.8	1032.4
2	20	5	0	532.1	758.9	2	5	10	0	492.0	1068.6
2	10	5	0.01	525.1	804.1	2	50	10	0	477.7	1080.6
2	5	5	0.01	562.7	896.2	2	50	10	0.001	485.9	1127.9
2	20	5	0.001	495.4	900.6	2	5	10	0.001	491.3	1212.5
2	50	5	0.01	495.1	1041.6	2	5	10	0.01	499.9	1234.4
2	5	5	0	506.8	1211.5	3	20	10	0.001	555.9	1331.6
2	20	5	0.01	522.6	1228.3	3	5	10	0	582.3	1363.2
2	50	5	0	480.6	1290.9	3	10	10	0.01	570.8	1370.8
2	10	5	0.001	507.3	1316.8	3	10	10	0	568.0	1380.8
2	50	5	0.001	486.6	1317.2	3	5	10	0.01	597.9	1419.9

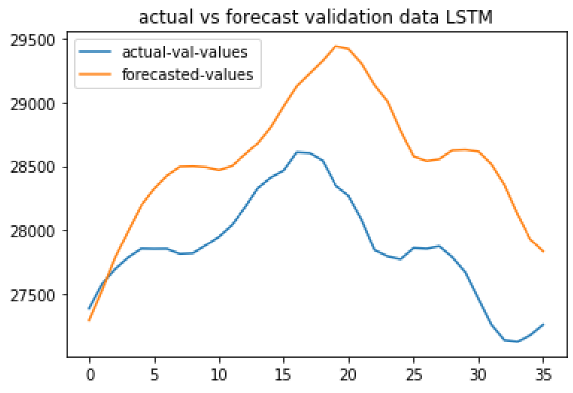
Look Back	Layer 1-units	Layer 2-units	Drop-out	Train RMSE	Validation RMSE	Look Back	Layer 1-units	Layer 2-units	Drop-out	Train RMSE	Validation RMSE
3	5	10	0.01	597.9	1419.9	4	20	20	0.01	599.9	1757.8
3	20	10	0	558.7	1474.8	4	50	20	0.001	564.1	1760.5
3	50	10	0.01	535.7	1524.2	4	10	20	0.01	619.3	1802.4
3	50	10	0	538.5	1558.8	4	50	20	0.01	577.9	1950.5
3	10	10	0.001	582.1	1573.1	4	20	20	0.001	578.2	1955.4
3	50	10	0.001	545.4	1675.9	4	5	20	0.001	619.2	1962.5
3	20	10	0.01	550.9	1677.1	4	5	20	0.01	614.5	1987.7
4	10	10	0.001	614.9	1757.5	4	20	20	0	590.1	2003.4
4	20	10	0.01	595.8	1759.9	4	10	20	0	639.8	2082.0
4	10	10	0	609.4	1802.7	4	5	20	0	624.3	2216.8
4	50	10	0	581.5	1873.3	2	20	50	0.001	498.0	944.8
4	20	10	0.001	594.6	1878.3	2	20	50	0.01	478.9	1124.3
4	10	10	0.01	610.7	1892.7	2	10	50	0.001	480.1	1183.5
4	50	10	0.001	577.8	1910.2	2	5	50	0.01	489.2	1200.1
4	20	10	0	591.6	1925.7	2	10	50	0.01	479.3	1211.6
4	50	10	0.01	579.5	1941.4	2	50	50	0.001	472.4	1273.2
4	5	10	0.001	617.8	2018.1	2	50	50	0	469.5	1301.4
4	5	10	0.01	667.1	2026.6	2	50	50	0.01	475.0	1305.2
3	5	10	0.001	567.0	2126.4	2	10	50	0	480.9	1309.7
4	5	10	0	627.2	2134.4	2	5	50	0.001	489.7	1312.3
2	20	20	0.01	499.5	822.3	2	5	50	0	499.7	1324.7
2	10	20	0.001	493.3	884.5	2	20	50	0	477.7	1324.9
2	10	20	0	492.9	947.7	3	10	50	0.001	577.9	1336.8
2	50	20	0	497.1	980.7	4	50	50	0.01	507.5	1436.3
2	20	20	0.001	484.2	986.5	3	20	50	0	545.8	1450.0
2	10	20	0.01	492.6	1036.5	3	10	50	0.01	555.0	1476.1
2	5	20	0.01	490.5	1097.5	3	10	50	0	544.8	1503.5
2	5	20	0	494.2	1132.2	3	20	50	0.001	545.3	1536.3
2	50	20	0.01	479.4	1145.5	3	5	50	0.001	556.9	1617.9
2	20	20	0	475.9	1148.8	3	50	50	0.01	526.9	1624.4
2	5	20	0.001	503.0	1213.3	3	50	50	0	528.0	1654.5
3	5	20	0.001	603.1	1239.6	3	20	50	0.01	552.4	1681.6
2	50	20	0.001	473.6	1257.1	3	50	50	0.001	527.4	1691.5
3	10	20	0.01	570.4	1309.9	4	50	50	0	556.1	1738.9
3	10	20	0	595.4	1325.5	3	5	50	0	569.4	1750.6
3	10	20	0.001	562.6	1400.6	4	50	50	0.001	551.5	1756.3
3	20	20	0.01	552.2	1409.0	4	20	50	0.01	574.9	1769.7
3	5	20	0	570.5	1414.5	4	5	50	0	602.6	1845.2
3	20	20	0	555.1	1471.3	3	5	50	0.01	557.0	1869.8
3	50	20	0.001	541.8	1500.0	4	10	50	0.01	618.1	1887.9
4	10	20	0.001	620.8	1527.1	4	20	50	0.001	578.9	1894.1
3	20	20	0.001	545.0	1560.3	4	5	50	0.001	602.1	1896.5
3	5	20	0.01	595.0	1561.5	4	10	50	0	576.0	1912.9
3	50	20	0.01	547.8	1618.6	4	5	50	0.01	611.6	1957.4
4	50	20	0	566.1	1691.4	4	10	50	0.001	602.3	1990.5
3	50	20	0	519.3	1697.5	4	20	50	0	574.1	2013.8

4.1.5. Experimental Results: LSTM

Our optimal model was found at 2 look back periods, with 2 layers and 10 units in each layer, with drop out of 0.001. In this model, the validation data showed RMSE of 751 and training RMSE of 550. On reviewing the plot of validation and training loss against epochs, we see now the overfitting problem had been successfully resolved (Fig 13).

Thus we are in consonance with Zhang , Song , Li, Li Ma, Pan and Han (2019) claim that in case of time series forecasting, prediction with 2 LSTM layers has a result closest to the true data value considering the accuracy and fitting effect of the model.





4.1.6. 2 Layer Model

- As can be seen in the graphs below, for training error we can see that the increase in units of layer 1, training error is declining, meanwhile with increase in units of layer2, training error first declines then rises. Additionally, with a rise in time steps , training error rises (Fig 14) . The colour dimension displays the training error.

- In case of validation error plot we see that for both layers an increase in units after a point leads to a fall in error, implying that increasing units of layer 2 after a point is leading to overfitting. Further, increase in time steps leads to a decline in error after a point like training error graph (Fig 15). The colour dimension displays the validation error.

Fig 14: LSTM: 2 layer model hyperparameter : Training Error explored in the colour dimension

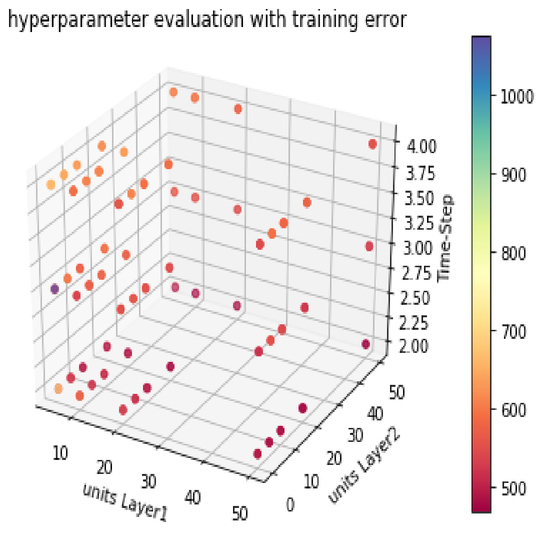
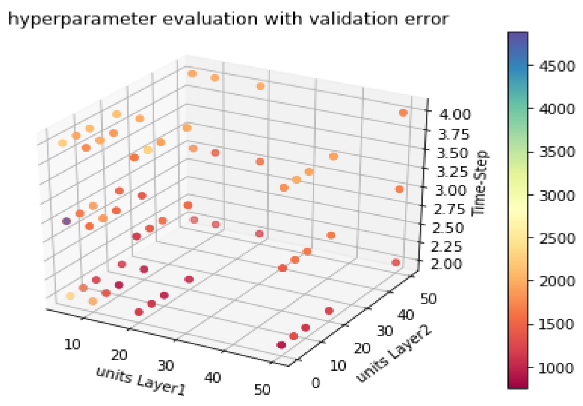


Fig 15: LSTM: 2 layer model hyperparameter : Validation Error explored in the colour dimension



4.1.7. Single Layer Model : LSTM

- Next, we look at models with a single layer and evaluate their performance on training vs validation error. We see that with the increase in the size of the layer the training error falls, meanwhile validation error first falls then rises, indicating overfitting.

- With an increase in time steps both training error and validation error rise, which means that larger time steps aren't optimal for single layer model. **Hence again, we are in consonance with Sauda and Shakyatjhat (2019) that smaller look back periods in case of LSTM time series prediction, far outperform larger look back periods.**

Fig 16: LSTM: 1 layer model hyperparameter : Training Error explained in the colour dimension

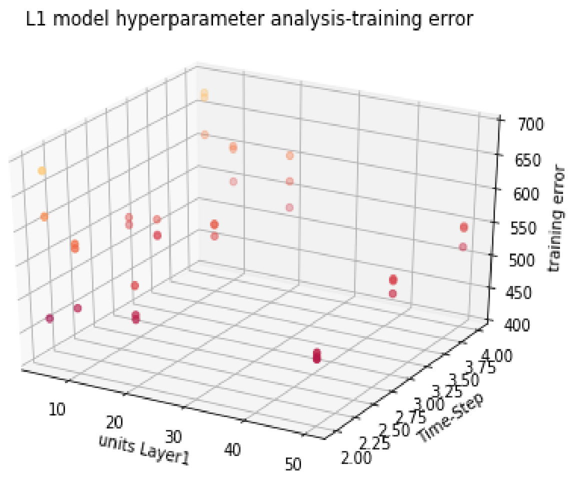
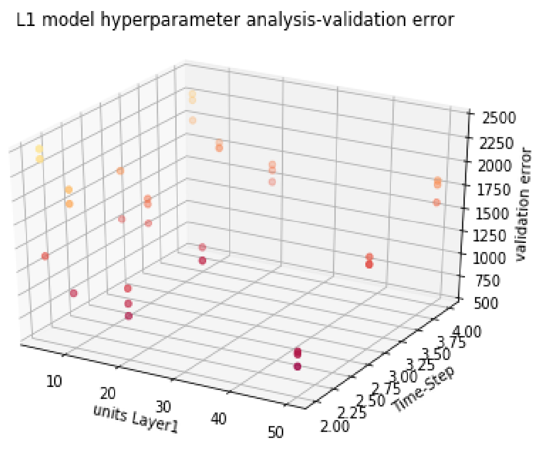


Fig 16: LSTM: 1 layer model hyperparameter: Validation Error explained in the colour dimension



4.2. Model 2: Support Vector Regression Framework

4.2.1 SVR Model Architecture

Support Vector Machines formulated by Vapnik, 1995 is an algorithm that attempts to find a separating hyperplane in the original input space to demarcate or separate the training data. In regression estimation, the data points that realize the maximal margin are called support vectors. If the training set is not linearly separable, then a nonlinear boundary has to be constructed. In order to achieve the nonlinear boundary, the original input space is mapped into a higher dimensional space called feature space. The feature space is then searched for by a hyperplane that can separate the instances in the same feature space. The mapping from the input space to the feature space is defined by a kernel function. The technique also allows for misclassification by introducing a penalty factor 'C' in the optimization model and the total penalty is found by summing up penalties on each misclassification. Therefore, the technique finds a hyperplane that minimizes the sum of the reciprocal of the margin and the total penalty (29).

The SVM model can be extended to a regression setting. The support Vector Regression (SVR) solves a nonlinear regression in a linear way by mapping nonlinear input data from original dimensional feature space to a higher dimensional feature space. For training data $\{(x_1,y_1), (x_2,y_2)\ldots..(x_n, y_n)\}$, with y_i, as the the target values and n data points , the SVR function takes the form as:

$$f(x) = \langle x,w\rangle + b \text{ with } w \in X , b \in R,$$

where <.,.> denotes dot product in X, i.e.) denotes a nonlinear space transformation, w is the weight vector and b is the bias. In SVR we aim to find parameters such that the deviation from the target value doesn't exceed a constant ε and at the same time is as flat as possible. We aim to attain the smallest value of w so as to prevent overfitting (flatness in this means that one seeks a small w). One way to ensure this is to minimize $\|w\|^2$ i.e. <w,w> . This can be stated as a convex optimization problem(29).

minimize $\frac{1}{2} \|w\|^2$

subject to $y_i - \langle x_i, w\rangle - b \leq \varepsilon$

$\langle x_i, w\rangle + b - y_i \leq \varepsilon$

The convex optimisation problem may not be feasible at times, therefore slack variables (ξi, ξi∗) were introduced to cope with the infeasibility of the problem. In this regard, we arrive at Vapnik (1995) formulation

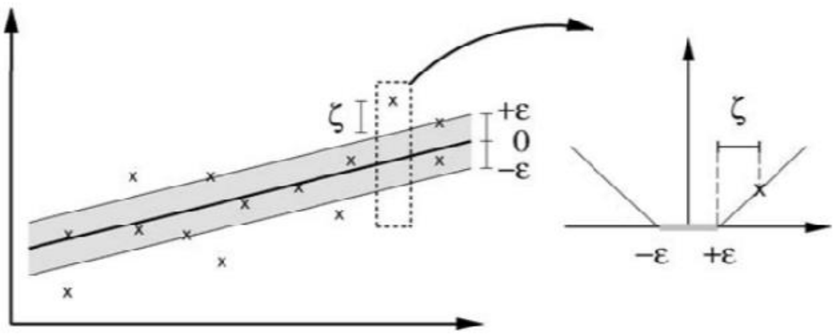
minimize $\frac{1}{2} \|w\|^2 + C\Sigma\ (\xi i + \xi_i^*)$

Subject to $y_i - \langle x_i, w_i\rangle - b \leq \varepsilon + \xi i$

$\langle x_i, w\rangle + b - y_i \leq \varepsilon + \xi_i^*$

$\xi i, \xi_i^* \geq 0$

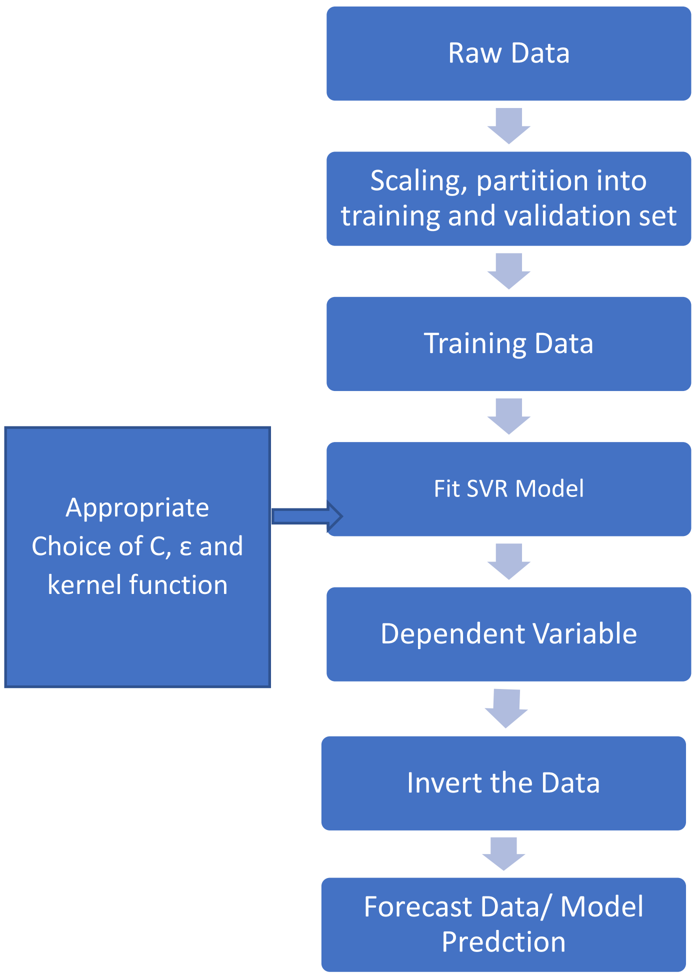
The constant C is a regularization parameter that controls the trade-off between the achieving a low training error and a low testing error, that is the ability to generalize your classifier to unseen data (bias-variance trade off) i.e. the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. According to the above equation, all data points whose y-values differ from f (x) by more than ε, are penalized. The slack variables, and (ξi, ξ_i^*) correspond to the size of this excess deviation for upper and lower deviations, respectively, as represented graphically in the figure below (30).



As can be seen only the points outside the shaded region determine the loss function and in this case the deviations are penalized in a linear fashion. When the dual of this problem is created, SVR can be extended to nonlinear functions (30).

The dual of the optimization problem is a quadratic programming problem with linear constraints, which is easier to solve than the primal and ensures a unique global optimum. The input vector only appears inside the dot product, which ensures that the dimensionality of the input space can be hidden from the remaining computations. That is, even though the input space is transformed into a high dimensional space, the computation does not take place in that space but in the linear space (Gunn, 1998). Additionally, the dual form does allow the replacement of the dot product of input vectors with a non-linear transformation of the input vector.

Fig 17: Methodology of SVR implementation


Raw Data
Scaling, partition into training and validation set
Training Data
Appropriate Choice of C, ε and kernel function
Fit SVR Model
Dependent Variable
Invert the Data
Forecast Data/ Model Predction

4.2.2 LSTM Architecture on choice of our data

1) Optimization Algorithm:

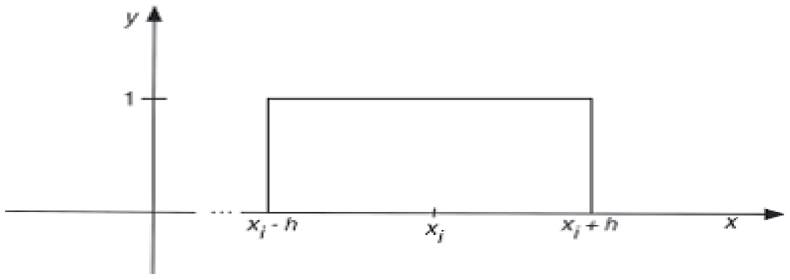
SVR's cost function is convex as well and we will be using Sequential Minimal Optimization which solves a large quadratic programming(QP) problem by breaking them into a series of small QP problems that improve computational efficiency(27).

2) Kernel Function:

As described above, different SVM algorithms use different types of kernel functions. The kernel functions in SVR allow the non-linear decision surface can be transformed to a linear equation in a higher number of dimension spaces. We define the kernel or a window function as follows:

K(x)=1, if ‖w‖ ≤ 1, and 0 otherwise.

The kernel functions return the inner product between two points in a suitable feature space. A kernel function must be positive definite in order to guarantee a unique optimal solution to the quadratic optimization problem. The value of the kernel function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise . For a fixed xi, the function is K(z-xi)/h) = 1 inside the closed ball of radius h centred at xi, and 0 otherwise as shown in the figure below:



- **Polynomial kernel**

  *K(xi,xj)= (xi*xj +1)* ^{d,} *where d is the degree of the polynomial*

  In case of polynomial kernel, the degree of the polynomial need to be carefully selected. A higher value of d, might maximize the training accuracy, however can be indicative of overfitting and lead to a low validation accuracy.

- **Gaussian radial basis function (RBF)**

  K(xi,xj)= exp(-γ‖xi-xj‖2) for γ>0 and var *γ=1/2σ2*

  In case of RBF kernel, the parameter gamma which is the inverse of the standard deviation of the RBF kernel (Gaussian function) needs to be optimized. When σ^2 *is very large, the output of kernel function f is close 1, as* σ^2 *getting smaller, f moves towards to 0. In a big* σ^2 *has higher bias and lower variance(underfitting), while a small* σ^2 *regards it 'further' which has lower bias and higher variance (overfitting)*

4.2.3 Choice of Hyper-Parameters in our SVM Model

We used a grid search algorithm for the purpose of determining the most efficient model. The metric of choice we used for testing the optimality of the model was the root mean square error (RMSE) on the hold-out validation data.

- **Kernel Function** : [RBF, Polynomial]
- **Degree for a polynomial kernel** [3,4,5]
- **Gamma of radial kernel** [1, 0.1, 0.01, 0.001]
- **C** [0.1, 1, 10, 100]
- **ε** [0.1, 0.01, 0.001]
- **Look-back** [2,3,4]

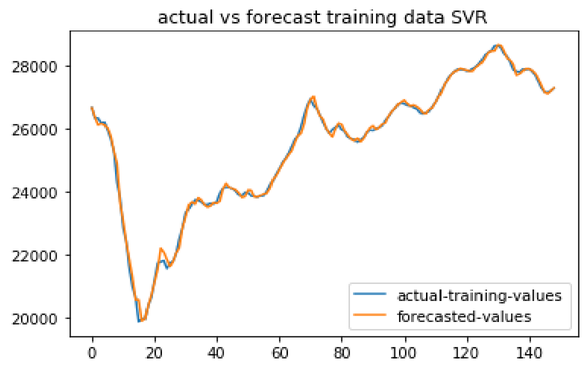
Hyperparameter Grid Search Results: SVR

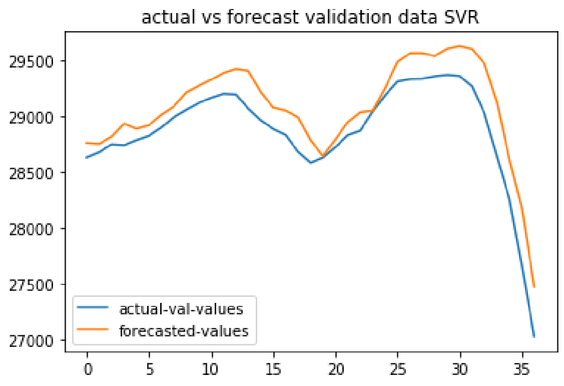
Look-back	C	epsilon	gamma	Train RMSE	Val RMSE	Look-back	C	epsilon	gamma	Train RMSE	Val RMSE
4	100	0.01	0.001	275.53	498.30	4	0.1	0.1	0.01	1718.6141	3111.48
3	100	0.001	0.001	289.49	529.21	3	0.1	0.1	0.01	1816.7988	3235.066
3	100	0.01	0.001	293.72	532.83	5	100	0.001	0.1	132.90	184.47
4	100	0.001	0.001	268.60	536.61	3	100	0.001	0.1	151.32	194.64
5	100	0.01	0.001	253.15	537.43	4	100	0.001	0.1	139.37	200.73
5	100	0.001	0.001	241.07	544.79	5	100	0.01	0.1	132.47	275.61
5	10	0.001	0.001	532.26	1154.37	4	100	0.01	0.1	136.30	282.50
5	10	0.01	0.001	522.48	1238.16	3	100	0.01	0.1	154.59	328.46
4	10	0.001	0.001	496.12	1245.42	5	10	0.01	0.1	146.76	367.95
4	10	0.01	0.001	482.15	1258.47	3	10	0.001	0.1	171.64	378.38
3	10	0.001	0.001	484.21	1406.72	4	10	0.01	0.1	147.09	378.82
	100	0.1	0.001	519.10	1420.38	4	10	0.001	0.1	145.86	421.70
3	10	0.01	0.001	484.71	1421.08	5	10	0.001	0.1	145.52	435.56
4	100	0.1	0.001	524.05	1492.15	3	10	0.01	0.1	172.01	470.54
3	100	0.1	0.001	522.62	1555.77	3	1	0.01	0.1	277.98	740.82
5	10	0.1	0.001	752.04	2061.16	5	1	0.001	0.1	240.73	754.20
4	10	0.1	0.001	783.18	2192.90	5	1	0.01	0.1	251.41	754.55
3	10	0.1	0.001	857.29	2439.30	4	1	0.01	0.1	274.91	769.11
5	1	0.01	0.001	1634.89	2482.45	4	1	0.001	0.1	254.79	802.17
5	1	0.001	0.001	1621.27	2517.29	3	1	0.001	0.1	269.13	820.67
4	1	0.01	0.001	1742.09	2637.50	5	0.1	0.01	0.1	519.81	1356.42
4	1	0.001	0.001	1732.53	2671.95	4	0.1	0.01	0.1	491.67	1380.44
3	1	0.01	0.001	1856.071353	2798.358	4	10	0.1	0.1	436.31	1389.54
3	1	0.001	0.001	1850.669641	2820.97	4	100	0.1	0.1	436.31	1389.54
5	0.1	0.01	0.001	2157.490453	2911.278	3	10	0.1	0.1	424.32	1400.82
4	0.1	0.01	0.001	2168.574489	2928.027	3	100	0.1	0.1	424.32	1400.82
5	0.1	0.001	0.001	2150.149052	2942.767	5	0.1	0.001	0.1	534.64	1411.40
3	0.1	0.01	0.001	2179.972283	2945.182	4	0.1	0.001	0.1	497.97	1421.73
4	0.1	0.001	0.001	2161.737307	2957.234	5	10	0.1	0.1	454.95	1534.21
3	0.1	0.001	0.001	2173.69544	2971.83	5	100	0.1	0.1	454.95	1534.21
5	1	0.1	0.001	1644.129169	2995.089	3	0.1	0.01	0.1	497.52	1565.13
4	1	0.1	0.001	1714.942584	3109.163	3	1	0.1	0.1	532.57	1566.65
3	1	0.1	0.001	1814.610005	3234.633	3	0.1	0.001	0.1	502.36	1585.63
5	0.1	0.1	0.001	2106.801496	3244.6	4	1	0.1	0.1	587.86	1643.84
4	0.1	0.1	0.001	2117.876093	3250.09	5	1	0.1	0.1	577.62	1651.32
3	0.1	0.1	0.001	2128.674867	3263.914	5	0.1	0.1	0.1	770.04	2166.03
5	100	0.01	0.01	150.60	252.32	4	0.1	0.1	0.1	806.41	2265.50
4	100	0.01	0.01	151.37	271.05	3	0.1	0.1	0.1	911.00	2526.50
5	100	0.001	0.01	146.15	292.46	5	100	0.001	1	106.18	185.76
3	100	0.001	0.01	162.24	303.51	5	100	0.01	1	108.66	337.49
4	100	0.001	0.01	146.10	308.53	3	100	0.01	1	139.86	418.96
3	100	0.01	0.01	174.95	378.64	5	10	0.01	1	123.44	438.41
3	10	0.01	0.01	284.87	555.74	4	100	0.001	1	119.43	439.08
5	10	0.01	0.01	246.51	564.30	4	100	0.01	1	121.17	441.21
3	10	0.001	0.01	282.91	566.55	4	10	0.01	1	127.32	473.64
5	10	0.001	0.01	231.00	570.84	3	10	0.01	1	146.73	515.59
4	10	0.001	0.01	262.58	587.81	5	10	0.001	1	121.98	563.18
4	10	0.01	0.01	267.84	592.17	3	100	0.001	1	141.20	576.20
5	100	0.1	0.01	427.26	1029.90	4	1	0.01	1	139.83	646.00
4	100	0.1	0.01	429.22	1037.12	3	10	0.001	1	146.52	656.59
5	1	0.001	0.01	532.39	1172.87	4	10	0.001	1	127.33	680.24
3	100	0.1	0.01	413.20	1208.83	3	1	0.001	1	165.22	713.16
5	1	0.01	0.01	518.68	1241.63	5	1	0.01	1	144.09	713.56
4	1	0.001	0.01	494.79	1266.15	4	1	0.001	1	142.73	740.61
4	1	0.01	0.01	479.55	1266.53	5	1	0.001	1	140.68	781.02
3	1	0.001	0.01	484.53	1420.39	3	1	0.01	1	176.84	803.06
3	1	0.01	0.01	488.72	1444.42	3	0.1	0.01	1	341.51	1433.59
5	10	0.1	0.01	523.40	1462.01	5	0.1	0.01	1	339.97	1438.31
4	10	0.1	0.01	532.22	1529.79	5	0.1	0.001	1	323.01	1467.82
3	10	0.1	0.01	522.85	1556.78	3	0.1	0.001	1	329.32	1498.29
5	1	0.1	0.01	753.55	2064.77	4	0.1	0.001	1	340.71	1502.05
4	1	0.1	0.01	780.14	2190.87	4	0.1	0.01	1	352.38	1536.99
3	1	0.1	0.01	857.37	2446.54	4	10	0.1	1	415.54	1639.93
5	0.1	0.01	0.01	1640.60	2488.57	4	100	0.1	1	415.54	1639.93
5	0.1	0.001	0.01	1627.03	2523.37	3	1	0.1	1	410.76	1641.15
4	0.1	0.01	0.01	1745.94	2640.56	3	10	0.1	1	410.76	1641.15
4	0.1	0.001	0.01	1736.357007	2675.102	3	100	0.1	1	410.76	1641.15
3	0.1	0.01	0.01	1858.291245	2799.442	4	1	0.1	1	425.44	1643.70
3	0.1	0.001	0.01	1852.909385	2821.99	5	10	0.1	1	438.90	1656.67
5	0.1	0.1	0.01	1645.122933	3004.44	5	100	0.1	1	438.90	1656.67
4	0.1	0.1	0.01	1718.614108	3111.48	5	1	0.1	1	455.13	1661.91
3	0.1	0.1	0.01	1816.798829	3235.066	3	0.1	0.1	1	549.40	2089.69
5	100	0.001	0.1	132.90	184.47	5	0.1	0.1	1	570.31	2155.02
3	100	0.001	0.1	151.32	194.64	4	0.1	0.1	1	555.22	2189.25

Look-back	C	epsilon	degree	Train RMSE	Val RMSE	Look-back	C	epsilon	degree	Train RMSE	Val RMSE
3	0.1	0.1	3	894.70	885.46	3	0.1	0.001	4	1105.47	3138.28
5	0.1	0.1	3	885.51	1127.67	4	10	0.1	4	904.14	3144.92
4	0.1	0.1	3	895.61	1164.41	4	0.1	0.001	4	1071.25	3152.91
3	1	0.1	3	868.39	1179.50	4	0.1	0.01	4	1078.62	3158.71
3	0.1	0.01	3	911.60	1418.11	3	1	0.001	4	995.49	3174.96
3	0.1	0.001	3	909.46	1421.56	3	10	0.1	4	922.20	3239.21
4	0.1	0.01	3	905.52	1515.15	4	100	0.001	4	915.12	3380.79
4	0.1	0.001	3	901.71	1539.69	3	10	0.01	4	936.68	3431.42
4	1	0.1	3	852.47	1756.18	5	1	0.1	4	945.29	3493.14
5	0.1	0.01	3	880.95	1834.21	3	100	0.1	4	905.75	3553.56
3	100	0.1	3	767.38	1872.03	3	10	0.001	4	933.50	3553.86
5	100	0.1	3	740.16	1912.53	4	10	0.01	4	932.95	4082.63
5	0.1	0.001	3	881.12	1923.98	5	0.1	0.01	4	1051.16	4259.06
3	10	0.1	3	809.40	1979.44	5	1	0.01	4	934.24	4616.86
3	1	0.01	3	852.80	2017.62	4	10	0.001	4	929.06	4617.34
4	100	0.1	3	770.01	2176.30	5	0.1	0.001	4	1046.36	4622.92
5	100	0.01	3	716.48	2242.87	5	1	0.001	4	936.59	4733.86
5	1	0.1	3	830.32	2259.64	3	100	0.01	4	935.33	4786.27
3	1	0.001	3	851.14	2312.71	4	1	0.01	4	988.83	4820.31
5	10	0.1	3	798.19	2367.76	3	100	0.001	4	935.10	4916.06
5	100	0.001	3	709.37	2375.92	4	1	0.001	4	991.78	4948.50
4	10	0.1	3	809.58	2381.34	4	100	0.001	5	1022.92	578.23
3	10	0.01	3	779.98	2601.17	4	100	0.01	5	1028.15	587.94
3	100	0.001	3	750.81	2731.91	3	0.1	0.1	5	1228.84	1481.85
3	100	0.01	3	751.33	2753.18	4	0.1	0.1	5	1175.61	2287.47
4	1	0.01	3	829.00	2821.45	3	1	0.1	5	1117.20	2331.70
3	10	0.001	3	777.51	2838.21	5	100	0.01	5	973.25	2450.36
4	10	0.01	3	759.50	2862.65	3	1	0.01	5	1096.87	2532.35
4	1	0.001	3	831.80	2864.32	5	0.1	0.1	5	1137.44	2652.23
4	100	0.001	3	740.76	2889.06	4	10	0.1	5	1019.14	2806.72
4	100	0.01	3	741.78	2919.45	5	100	0.001	5	982.54	2940.50
5	10	0.01	3	740.06	2991.42	3	1	0.001	5	1095.20	3021.33
5	1	0.01	3	804.92	3022.74	4	100	0.1	5	1004.31	3183.00
5	10	0.001	3	734.87	3044.57	4	10	0.001	5	1048.91	3204.68
5	1	0.001	3	797.72	3212.03	5	1	0.01	5	1027.92	3315.99
4	10	0.001	3	754.48	3367.64	3	100	0.1	5	1024.90	3331.90
3	0.1	0.1	4	1081.19	1172.90	3	100	0.001	5	1050.16	3457.49
4	0.1	0.1	4	1060.63	1523.79	3	0.1	0.01	5	1229.53	3465.10
3	1	0.1	4	1007.60	1789.19	5	1	0.001	5	1030.25	3550.79
5	100	0.001	4	881.51	1953.61	5	1	0.1	5	1003.33	3616.52
5	0.1	0.1	4	1039.03	2057.08	5	10	0.01	5	1003.57	3698.51
5	100	0.01	4	877.81	2083.74	5	10	0.001	5	1007.82	3704.10
4	100	0.1	4	894.58	2411.45	3	10	0.1	5	1043.03	3732.24
5	10	0.001	4	899.61	2472.59	4	10	0.01	5	1050.29	3795.54
3	0.1	0.01	4	1097.80	2612.12	4	1	0.1	5	1054.55	3823.82
5	100	0.1	4	844.76	2793.06	3	0.1	0.001	5	1236.71	3827.64
5	10	0.1	4	863.24	2884.95	4	0.1	0.001	5	1177.16	3832.39
5	10	0.01	4	900.03	2962.23	4	0.1	0.01	5	1172.39	3961.34
4	1	0.1	4	973.20	2981.30	3	10	0.001	5	1066.82	4072.68
3	1	0.01	4	1002.46	3124.31	3	100	0.01	5	1048.99	4108.03
4	100	0.01	4	918.90	3124.70	3	10	0.01	5	1061.62	4309.87
3	0.1	0.001	4	1105.47	3138.28	5	100	0.1	5	964.32	4623.48
4	10	0.1	4	904.14	3144.92	5	10	0.1	5	978.02	4725.93
4	0.1	0.001	4	1071.25	3152.91	4	1	0.01	5	1083.86	4799.79
4	0.1	0.01	4	1078.62	3158.71	4	1	0.001	5	1077.66	4884.41
3	1	0.001	4	995.49	3174.96	5	0.1	0.01	5	1116.44	4950.70
3	10	0.1	4	922.20	3239.21	5	0.1	0.001	5	1116.33	5077.94

4.2.4 Experimental Results: SVR

Above are the results of grid search. We notice that our optimal model minimising validation loss a model with RBF kernel with gamma value 0.1, C value of 100, look back period 4 and epsilon of 0.001. Our optimal model the validation data showed RMSE of 184 and training RMSE of 182.



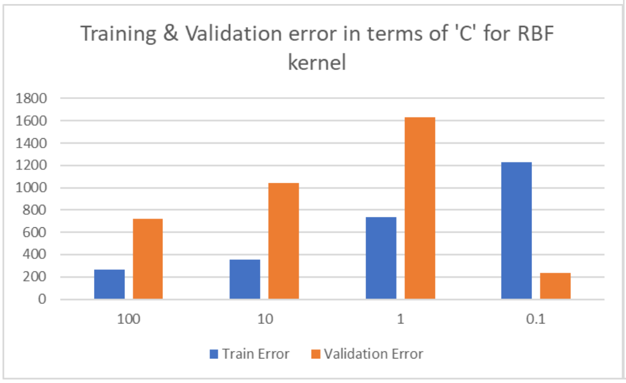


Thus, we agree with Kyoung-jae Kim(2002) study that for financial time series prediction SVR with Gaussian kernel outperforms SVR with polynomial kernel. However, we contrast L. J. Cao and Francis E. H. Tay (2003) study for financial time series forecasting that with RBF kernel, σ^2 optimal value should be between 100 and 10 000, since in our case optimal value of σ^2 =1/gamma is 10.

- As can be seen, both for RBF and polynomial kernel we plotted training and validation root mean square error (average error for different values of epsilon, gamma and polynomial degree) were studied for different values of C. As described above, 'C' is the penalty hyperparameter of the error term in the training process. As can be seen with high values of C, there is immense stress on reduction in training error, by formation of complex hyperplanes (for both RBF and Polynomial kernels). The C parameter tells the SVM optimization the cost of misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane, Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.
- As 'C' falls in line with expectations training error rises. However, with a decline in 'c' beyond 1 for RBF kernel and beyond 10 for polynomial kernel, validation error rises, indicative of underfitting the model. This implies that the hyperplane is too simple,it is unable to capture the intricacies of data, leading to both high training and validation error(Fig 17,18).

Fig 18: With a decline in 'c' beyond 1 for RBF kernel validation error rises

Fig 19: With a decline in 'c' beyond 10 for RBF kernel validation error rises



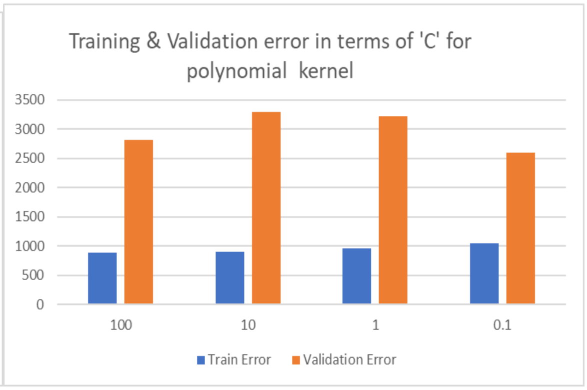
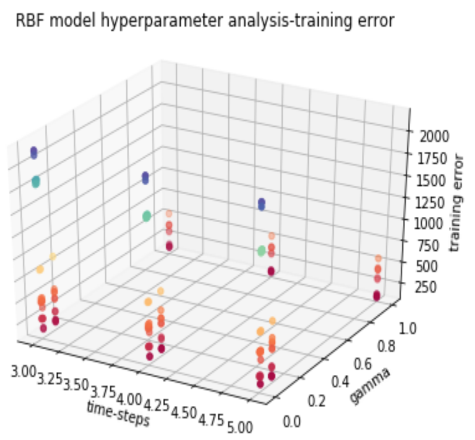
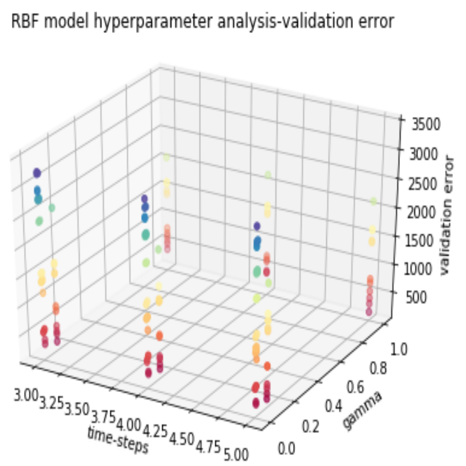


Fig 18: With a rise in 'gamma' training error falls validation error first falls, then rises

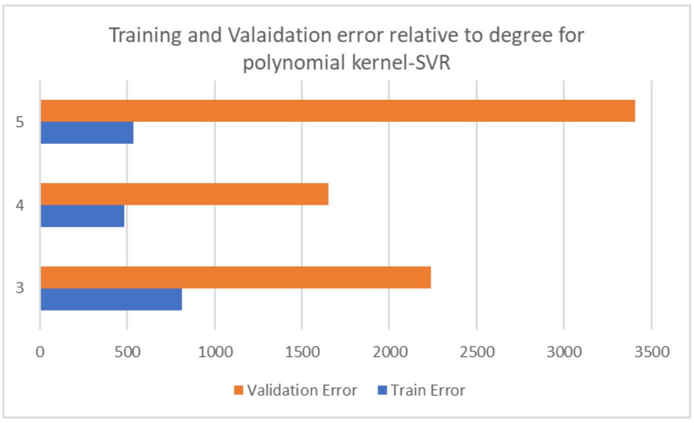
Fig 19: With a rise in 'gamma'





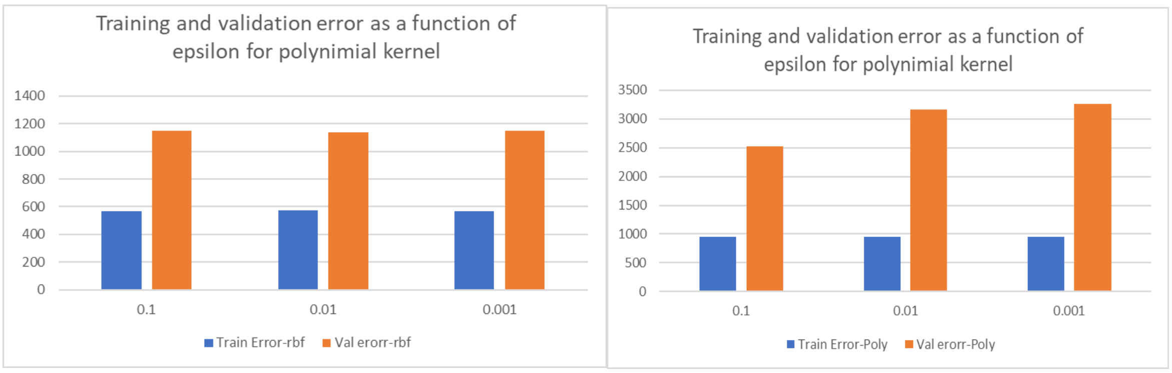
- As gamma rises, we can see a fall in trajectory of training error, since it implies that variance of rbf kernel is falling. Initially, with a rise in gamma, validation error declines, however, with a rise in gamma beyond a point, validation error begins to rise, indicative of overfitting.
- We fail to determine a cohesive trajectory of the time steps with validation and training error.
- As can be seen in the figure below (fig 20), in case of polynomial kernel, while with a rise in degree of kernel beyond 4, both training and validation error decline, implying at 4 degrees, the polynomial function is an optimal fit.

Fig 20: rise in degree of kernel beyond 4, both train & val error decline



- As can be seen by graphs below (fig 21), ε shows no clear trajectory with training or validation error, for both polynomial and rbf kernel. **Thus, consistent with L. J. Cao and Francis E. H. Tay (2003) study that the performance of SVM is insensitive to the choice of ε, our model displayed a similar picture.**

Fig 21: Both RBF and polynomial kernel insensitive to choice of ε



5. Conclusion

5.1 Algorithm Comparsion

The algorithms were evaluated on the basis of the performance of the validation data basis Mean Absolute Percentage Error (MAPE), root mean squared error (RMSE), mean absolute error (MAE), and Symmetric mean absolute percentage error (SMAPE or sMAPE). These are the measures of the deviation between the actual and forecasted values. The smaller given errors, the closer are the predicted time series values to the actual values

- **Our optimal model in case of LSTM was found at 2 look back periods, with 2 layers and 10 units in each layer, with drop out of 0.001.**
- **Our optimal model with SVR was found with RBF kernel with gamma value 0.1, C value of 100, look back period 4 and epsilon of 0.001.**

Evaluation Metrics	Validation error: LSTM	Validation error: SVR
mape	2.5	0.71
smape	2.4	0.71
mae	698.3	206
rmse	780.74	241

We see that for all performance metrics we see that SVR far outperformed LSTM. As the table above lists, in our study SVR showed more a more robust model, in terms of various evaluation metrics presented in the table above.

Our results contradict popular studies like Lakshminarayanan and J McCrae (2003), Karmiani; Kazi; Nambisan; Shah; Kamble (2019) and Chniti, Bakir, Zaher (2017) that suggested that newer deep learning models like LSTM outperform SVR for time series forecasting problems . However, our models consisted of a very small training data of 188 observations (in conjunction with the onset of the Covid-19 pandemic) and thus cannot be generalised.

SVR is typically the most optimal model, where the dimensionality of the input space and the order of the approximation creates a dimensionality of a feature space representation much larger than that of the number of examples. This was the case for our problem under study, with various look-back features and a small time series sample size. Further, interestingly when two algorithms were compared in case of SVR the optimal model consisted of 4 time steps. On other hand for LSTYM 2 time steps optimal and adding a greater time dimension, gave rise to overfitting (30).

SVM requires the solving of a Quadratic Programming (QP) problem over a solution space known to be convex. Therefore, every local optima will also be a global solution. Henceforth, in case of SVM we succeed in finding a unique global solution (Burges and Crisp, 2000). However, while LSTM succeeds in converging, it might often result in convergence at a local optimum which might not be efficient. Further, the training time of LSTM models far exceeds SVR and is computationally very expensive.

We conducted 10 runs of optimal chosen model for both SVM and LSTM. LSTM results displayed much higher variance than SVR for the successive runs. Further, SVR depends on three training parameters: the type of kernel used, and the respective values of C and ε. Meanwhile, LSTM deep network which has large number of parameters to learn and is computationally very expensive and time consuming. Further, the variance of validation error calculated during grid search is much higher for the LSTM model, compared to the SVR model, and therefore is highly sensitive to the choice of the hyperparameters. However, when trained on a very large number of epochs, LSTM errors display lower variance (though larger epochs maybe prone to overfitting). However, for larger time series datasets if the requirement is high accuracy and lower variance, LSTM would be an optimal model of choice (21). Further, the choice of the model depends on the complexity of the data. If the training data is very noisy, a high capacity model like neural networks are more likely to avoid overfitting and lower capacity models might incorporate the noise as part of the model .

5.2 Way Forward

The above mentioned approaches are expected to be plagued by the problem of catastrophic forgetting, whereby with successive time periods, the network loses the tendency to retain the information gathered from the previous time periods as the weights as successively updated with the time periods. These models have a tendency for knowledge of previously learnt history of the time series data to be lost abruptly, as new data is incorporated. This phenomenon, termed catastrophic forgetting occurs specifically when the network is trained sequentially on multiple tasks, because the weights in the network that are important for old data are updated to meet the objectives of new data. The typical models evaluated above, are static models incapable of adapting or expanding their behaviour over time. As such, the training process needs to be restarted, each time new data becomes available and the model needs an update (16). Thus, to circumvent catastrophic forgetting, going forward, continual learning algorithms can be investigated.

Further, an interesting choice of evaluation in the above case of a short time series horizon data and greater dimension can be Sai Li, ; Huajing Fang; Bing Shi's (2018) experiment. They investigated the use of long short-term memory (LSTM) network with error correction based on support vector regression (SVR) to realize multi-step-ahead time series prediction. As the parameters of trained LSTM model cannot be updated in the forecasting stage, the SVR model is used to predict the evolution of the residual error in LSTM to update the model parameters. The proposed LSTM-SVR model is validated and compared with LSTM and SVR, displayed much greater efficiency when compared in terms of accuracy measures.

Sang-Hoon Oh (1997) suggested that during the learning process, the ANN goes through stages in which the reduction of the error can be extremely slow. These periods of stagnation can influence learning times. In order to resolve this problem, if the conventional MSE loss function is replaced with the entropy error function a significant improvement can be seen. Thus, use of entropy loss function can be further examined as a hyperparameter during model training.

6. References

1) Karmiani; Kazi; Nambisan; Shah; Kamble (2019), Comparison of Predictive Algorithms: Backpropagation, SVM, LSTM and Kalman Filter for Stock Market
2) Chniti, Bakir, Zaher (2017), E-commerce Time Series Forecasting using LSTM Neural Network and Support Vector Regression
3) Wang, Meiqin Liu, Zhejing Bao and Senlin Zhang (2018) Short-Term Load Forecasting with Multi-Source Data Using Gated Recurrent Unit Neural Networks
4) Zhang , Song , Li, Li Ma, Pan and Han (2019), Research on Gas Concentration Prediction Models Based on LSTM Multidimensional Time Series
5) Arjun Singh Sauda , Subarna Shakya (2019), Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE
6) Kyoung-jae Kim , 2003, Financial time series forecasting using support vector machines
7) L. J. Cao and Francis E. H. Tay (2003), Support vector machine with adaptive parameters in financial time series forecasting
8) Singaravel, S., Suykens, J. and Geyer, P., 2018. Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction. Advanced Engineering Informatics, 38, pp.81-90.
9) Fathi, 2008, Time series forecasting using a hybrid ARIMA and LSTM model
10) Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, Honggang Zhang, 2017, Traffic Prediction Based on Random Connectivity in Deep Learning with Long Short-Term Memory
11) Ji, L., Zou, Y., He, K. and Zhu, B., 2019. Carbon futures price forecasting based with ARIMA-CNN-LSTM model. Procedia Computer Science, 162, pp.33-38.
12) Parisi, G., Kemker, R., Part, J., Kanan, C. and Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Networks, 113, pp.54-71.
13) Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, Tinne Tuytelaars, 2019, Continual learning: A comparative study on how to defy forgetting in classification tasks
14) Baestaens, D., Den Bergh, W. and Vaudrey, H., 1995. Options as a predictor of common stock price changes. The European Journal of Finance, 1(4), pp.325-343.
15) Nicholas Refenes, A., Zapranis, A. and Francis, G., 1994. Stock performance modeling using neural networks: A comparative study with regression models. Neural Networks, 7(2), pp.375-388.
16) Pai, P. and Lin, C., 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. Omega, 33(6), pp.497-505.
17) Karim, F., Majumdar, S., Darabi, H. and Chen, S., 2018. LSTM Fully Convolutional Networks for Time Series Classification. IEEE Access, 6, pp.1662-1669.
18) Qiu, M., Song, Y. and Akagi, F., 2016. Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market. Chaos, Solitons & Fractals, 85, pp.1-7.Making Good on LSTMs' Unfulfilled Promise
19) Daniel Philps, Arturd' Avila Garcez, Tillman Weyde, 2019, Making Good on LSTMs' Unfulfilled Promise
20) Huang, Nakomori and Yuang (2004) Forecasting stock market movement direction with support vector machine
21) Karmiani , Kazi , Nambis , Shah and Vijaya Kamble (2019), Comparison of Predictive Algorithms: Backpropagation, SVM, LSTM and Kalman Filter for Stock Market

22) Li, Tam (2017) A comparative study of a recurrent neural network and support vector machine for predicting price movements of stocks of different volatilites
23) Cao and Tay (2003) Support vector machine with adaptive parameters in financial time series forecasting
24) A. Mellit, A. Massi Pavan & M. Benghanem (2013),Least squares support vector machine for short-term prediction of meteorological time series
25) Xiaowei Jia, Ankush Khandelwal, Anuj Karpatne, Vipin Kumar V., 2019, Discovery of Shifting Patterns in Sequence Classification
26) Hochreiter, Schmidhuber (1997) Long Shirt Term Memory
27) SIMA SIAMI NAMIN1, AKBAR SIAMI NAMIN (2018) FORECASTING ECONOMIC AND FINANCIAL TIME SERIES: ARIMA VS. LSTM
28) Kingma, Ba(2017) ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION
29) Bukola Titilayo Ojemakinde , (2006), Support Vector Regression for Non-Stationary Time Series y Time Series
30) SMOLA and SCHOLKOPF, (2003), A tutorial on support vector regression
31) Sang-Hoon Oh(1997), Improving the error backpropagation algorithm with a modified error function

i https://www.investopedia.com/terms/d/djia.asp
ii Mohamed N. Nounou, Bhavik R. Bakshi, in Data Handling in Science and Technology, 2000