Optimizing the Convergence of ARIMA and LSTM models for Time Series Forecasting

Project Report - April-2022

Indian Institute of Information Technology, Allahabad

Submitted By
Sanghmitra Tamrakar
M.Tech IT (SE)
2nd Semester
MIT2021051

Under the supervision of
Dr. Ranjana Vyas
Indian Institute of Information Technology, Allahabad

Abstract

Time Series forecasting is an important area of machine learning because many forecasting problems have a time component. A Time Series is a collection of observations taken in chronological order over time. There have been numerous approaches used in the past for efficiently forecasting the next lag of time series data. When compared to other statistical models, the ARIMA model has demonstrated superior precision and accuracy in predicting future lags of time series. With recent advancements in computer computational power, the deep learninng alorithms are being used for time series forecasting. Long short Term Memory (LSTM) model is among the most popular deep learning models used today due to their capability of remembering past sequences. The performance of LSTM and ARIMA models depend on the selection of the hyperparameters. In this work the best set of hyperparameter are identified using different hyperparameter tuning methods such as Grid search, Random search and Bayesian search. A comparison of these hyperparameter tuning algorithms is discussed in order to generate the best performing model.

Keywords

Time series, ARIMA, LSTM, Grid search, Bayesian Search

Contents

1 Introduction

Time series analysis is useful for analysing historical datasets and their patterns, as well as understanding and matching the current situation with patterns derived from previous stages. In a time series, time is often the independent variable and the goal is usually to make a forecast for the future. A time series has several components, including trend, seasonality, cyclicity, and irregularity. Earlier, statistical models were used to identify and analyse these patterns. Because of advancements in the machine learning and deep learning fields, various deep learning approaches for forecasting and analysing time series data have emerged.

Deep learning models' performance is influenced by a set of parameters known as hyper-parameters. Hyperparameters are manually selected training variables that are set before the model is trained. Hyperparameters are different from the internal model parameters, such as the neural network's weights, which can be learned from the data during the model training phase. We want to identify a set of hyperparameter values that archive the better performance on the data in a reasonable amount of time before we start the training phase. This is referred to as hyperparameter tuning or optimization. It plays a vital role in the prediction accuracy of machine learning algorithms.

2 Literature Survey

Traditional time series forecasting methods are primarily based on statistical fundamentals, such as simple exponential smoothing (SES) and exponential smoothing (ETS)[1], which are appropriate for predicting time series without a trend. In contrast, Holt and Damped exponential smoothing [2] are commonly used for time series with trends. Furthermore, more accurate classic models, such as the ARIMA [3], combining autoregressive, moving averages, and integrated parts, were developed. The characteristics of the time series to be analysed have a large impact on the quality of the results. It is possible to obtain very small prediction errors in a stationary series with clearly defined patterns and few variations. The Seasonal Autoregressive Integrated Moving Average (SARIMA) is an ARIMA model that is commonly used with seasonal data. The work published in [4] conducted an empirical study that concluded that the SARIMA model, as well as other more complicated prediction methods, produced accurate forecasts.

Deep learning has grown in popularity in recent years, particularly in computer vision and natural language processing. Deep neural networks can learn complicated data representations. For sequence modelling, recurrent neural networks (RNNs) have been developed. Many RNN-based architectures have been developed for time series forecasting applications, owing to the natural interpretation of time series data as sequences of inputs and targets. The first RNN architectures had limitations in learning long-term dependencies in data due to issues with exploding and vanishing gradients due to the infinite lookback window. As a result, LSTM networks were created to address these limitations by improving the network's gradient flow. Li et al. [5] proposed an ARIMA-LSTM ensemble that improved forecasting accuracy for high-frequency financial time series.

One of the major drawbacks of ARIMA and deep learning models is their lengthy training period. Various optimization algorithms are developed to determine the correct set of hyperparameters. Although the ARIMA model has fewer hyperparameters, selecting the best model from a given set of hyperparameters is a time-consuming process. Grid search is the most general hyperparameter tuning algorithm, which tries every possible combination of parameters and selects the best model from it. Random search is a variant of the Grid search algorithm that samples the search space at random rather than discretizing it with a Cartesian grid. Random search has proven to be especially effective, especially when the search space is not cubic, i.e. when some hyperparameters have a much greater range of variation than others [6]. Another method for selecting hyperparameters was discussed in Automatic ARIMA modelling, a stepwise search [7] method developed by Hyndman, R. J., and Khandakar. Stepwise search first considers some specified parameters and then selects the next set of parameters based on their model performance. This process was repeated 13 times to find the best set of parameters for ARIMA. ARIMA introduced an evolutionary particle swarm optimization (PSO) algorithm to improve convergence and model accuracy [8]. Bayesian optimization based on Gaussian processes [9] has been proven a better optimisation approach for machine learning and deep learning models.

3 Methodologies

3.1 ARIMA (Auto-Regressive Integrated Moving Average) model

The ARIMA model is fundamentally a linear regression model accommodated to track linear tendencies in stationary time series data. The model is expressed as ARIMA(p,d,q). Parameters p, d, and q are integer values that decide the structure of the time series model; parameter p, q each is the order of the AR model and the MA model, and parameter d is the level of differencing applied to the data. The mathematical representation of the ARIMA model of order (p, d, q) is as follows-

$$Y_t = \sum_{k=1}^{p} \alpha_k Y_{t-k} + \sum_{l=1}^{q} \beta_l \theta_{t-l} + c \tag{1}$$

Term c a constant; α_k and β_l are coefficient values of AR model variable model variable Y_{t-k} and MA model variable θ_{t-l} is an error term. It is assumed that θ_{t-l} has zero mean with constant variance, and satisfies the given condition. To apply the ARIMA model the time series must be stationary. For making a time series stationary the values with itself past lags are differenced and ensuring that the mean, variance and covariance must be constant. The number of times time series is differenced is accounted by parameter 'd'.

The parameters p, d and q refer-
ARIMA(p,d,q) , here p = no. of autoregressive lag
d = no. of times difference taken
q = no. of moving average terms

3.2 LSTM (Long Short-Term Memory) model

LSTM is a variant of the Recurrent Neural network(RNN). Compared to traditional RNNs, its hidden layer module structure has been greatly redesigned. Although RNNs can effectively handle nonlinear time series, there are still two problems: (1) Due to gradient vanishing and gradient explosion, RNNs cannot handle time series with excessive delays; (2) training the RNN model requires a predetermined delay window length, but it is difficult to automatically obtain the optimal value of this parameter. Thus, the LSTM model emerged. The LSTM model replaces the hidden layer of RNN cells with LSTM cells to achieve long-term memory. The special design of the LSTM model makes it able to learn long-term dependencies and is particularly suitable for time series analysis.
An LSTM has three gates: an input gate that selects whether or not to accept incoming input, a forget gate that deletes irrelevant data, and an output gate that determines what data to output. These three gates are analog gates that work in the 0 to 1 range and are based on the sigmoid

function. The three sigmoid gates are depicted in Figure 1 The cell state is represented by a horizontal line that runs through the cell.

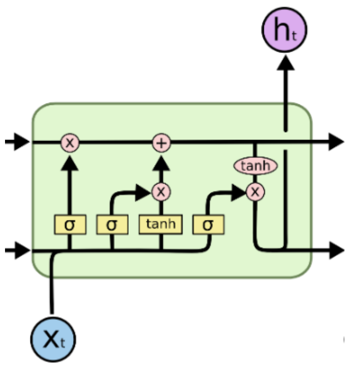

Figure 1: An LSTM cell

3.3 Hyperparameter tuning methods

3.3.1 Stepwise Search for ARIMA

The stepwise approach follows the strategy laid out by Hyndman and Khandakar in their 2008 paper [7]. In this search strategy initially four models are trained and their AIC (Akaike Information Criterion) are computed.

- ARIMA(2,d,2) if m = 1 and ARIMA(2,d,2)(1,D,1) if m > 1
- ARIMA(0,d,0) if m = 1 and ARIMA(0,d,0)(0,D,0) if m > 1
- ARIMA(1,d,0) if m = 1 and ARIMA(1,d,0)(1,D,0) if m > 1
- ARIMA(0,d,1) if m = 1 and ARIMA(0,d,1)(0,D,1) if m > 1

The model with smallest AIC value is selected and considered as the "current best" model. In the next steps a other models are considered as follows-

- Where one of p, q, P and Q is allowed to vary by ±1 from the current best model
- Where p and q both vary by ±1 from the current best model
- Where P and Q both vary by ±1 from the current best model

When a model with a lower information criterion is identified, it becomes the new current best model, and the operation is repeated until it cannot find a model with a lower information criterion that is near to the current best model, or the process exceeds one of the defined execution thresholds.

3.3.2 Grid Search

Grid search is the most basic hyperparameter tuning approach. In a nutshell, the hyperparameters' domain is divided into a discrete grid. Then, using cross-validation, we try every possible combination of values in this grid, calculating various performance measures. The ideal combination of values for the hyperparameters is the point on the grid that maximises the average value in cross-validation.
Grid search is an exhaustive technique that considers all possible combinations in order to locate the best point in the domain. The main disadvantage is that it is quite slow.

3.3.3 Random Search

In random search, we define distributions for each hyperparameter which can be defined *uniformly* or with a sampling method. The main difference between random search and grid search is that in random search, not all values are tested, and the values tested are chosen at random. We need to define the number of iterations to be tested for performing search. Random search is comparatively faster than grid search.

3.3.4 Bayesian Search

Bayesian Search approach uses the past evaluation results to sample new candidates that are most likely to give better results. It finds the posterior set of parameters based on the current settings of parameters and prior knowledge. Bayesian optimization is based on the Gaussian Process Models. It aims to gather observations revealing as much information as possible about this function and, in particular, the location of the optimum. It tries to balance exploration (hyperparameters for which the outcome is most uncertain) and exploitation (hyperparameters expected close to the optimum).
Algorithm- Given:-

- A BlackBox Function $f(x)$ which we want to approximate.
- Some data F is called search space.

Steps:-

- Start with a small number of sample points randomly chosen. (Call it Bag1)
- Use these points in Bag1 to compute a **Surrogate Function**.
- Iterate in a loop:
  - Add additional point to Bag1 using an **Acquisition Function**.
  - Re-evaluate the surrogate function.
  - Continue loop unless following occurs:
    Surrogate function max does not change Or variance below threshold Or search space is exhausted.

An acquisition function is a function of the Gaussian process, which will enable the selection of the next point to sample in the hyper-parameter space. There are many ways to define such a function. Expected improvement function (EI) as an acquisition function. It is defined as-

$$EI(x) = E[max(0, f(x) - f(x*))] \tag{2}$$

where f is the Gaussian process fitting response surface, x the hyper-parameter and x* the best hyper-parameter so far.

3.4 Evaluation metrics

3.4.1 Akaike Information Criterion (AIC)

The Akaike information criterion (AIC) is a mathematical method for determining how well a model fits the data from which it was generated. AIC is used in statistics to compare various possible models and determine which one is the best fit for the data. AIC is calculated as follows:

- The number of independent variables used in the model's construction.
- The model's maximum likelihood estimate (how well the model reproduces the data).

According to AIC, the best-fit model is the one that explains the most variation with the fewest number of independent variables. If we have several possible models, we can compare them using AIC. AIC scores with lower values are better, and AIC penalises models with more parameters. So, if two models explain the same amount of variation, the model with fewer parameters has a lower AIC score and is the better-fit model.

$$AIC = 2K - 2ln(L) \tag{3}$$

K is the number of independent variables used and L is the log-likelihood estimate (a.k.a. the likelihood that the model could have produced your observed y-values).

3.4.2 Root Mean Square Error (RMSE)

The root-mean-square error (RMSE) is a commonly used measure of the differences between predicted values (sample or population values) and observed values. The RMSE is defined as the square root of the second sample moment of differences between predicted and observed values, or the quadratic mean of these differences. When calculated over the data sample used for estimation, these deviations are called residuals, and when computed out-of-sample, they are called errors (or prediction errors). The RMSE aggregates the magnitudes of prediction errors for various data points into a single measure of predictive power. Because it is scale-dependent, RMSE is a measure of accuracy used to compare forecasting errors of different models for a specific dataset rather than between datasets.

$$rmse = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{4}$$

where y_i is ith actual value and x_i is ith actual value.

$$normalisedRMSE = rmse/max(Y) - min(Y) \tag{5}$$

4 Implementation and Results

Implementation - https://github.com/sanghmitr/Hyperparameter-optimization-ARIMA-LSTM-Time-series

4.1 Dataset used

The time series data used for this study is stock prices. The dataset is extracted from Yahoo finance which contains daily stock prices. For experimentation *Amazon* stock is chosen. The daily stock prices are considered from January 2015 to April 2022. The daily stock data includes date, low, high, open, close, volume columns. For the model training and performance evaluation daily closing prices of the stock is considered. The training set contains 80 percent of complete dataset and 20 percent is used for validation.

4.2 ARIMA Hyperparameters

Seasonal ARIMA model is considered for the training and forecasting purpose. S SARIMA requires selecting hyperparameters for both the trend and seasonal elements of the series. There are three trend elements that require configuration. They are the same as the ARIMA model; specifically:

- p: Trend autoregression order.
- d: Trend difference order.
- q: Trend moving average order.

There are four seasonal elements that are not part of ARIMA that must be configured; they are:

- P: Seasonal autoregressive order.
- D: Seasonal difference order.
- Q: Seasonal moving average order.
- s: The number of time steps for a single seasonal period.

4.3 LSTM Hyperparameters

- **Number of nodes and hidden layers -** The layers between the input and output layers are called hidden layers. The number of nodes in each layer can be configured. Many nodes within a layer can increase accuracy, fewer number of nodes may cause under-fitting. Number of units considered for search space- [16, 32, 64]
- **Dropout Rate -** Every LSTM layer should be accompanied by a dropout layer. Such a layer helps avoid overfitting in training by bypassing randomly selected neurons, thereby reducing the sensitivity to specific weights of the individual neurons. While dropout layers can be used with input layers, they shouldn't be used with output layers as that may mess

up the output from the model and the calculation of error.
Dropout rate is considered for search space - [0.1, 0.2, 0.3]

- **Optimizer -** While training the deep learning model, we need to modify each epoch's weights and minimize the loss function. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy.
  Optimizers considered for search space - ['RMSprop', 'Adam']

- **Number of Epochs -** This hyperparameters sets how many complete iterations of the dataset is to be run. While theoretically, this number can be set to an integer value between one and infinity.
  Number of epochs considered for search space - [10, 20, 30]

- **Batch Size -** This hyperparameter defines the number of samples to work on before the internal parameters of the model are updated. Large sizes make large gradient steps compared to smaller ones for the same number of samples "seen".
  Batch sizes is considered for search space - [50, 100]

4.4 ARIMA Results

Below mentioned plots are drawn using the best models suggested from stepwise search, grid search and bayesian search. The plots are drawn between actual vs predicted values on test dataset.

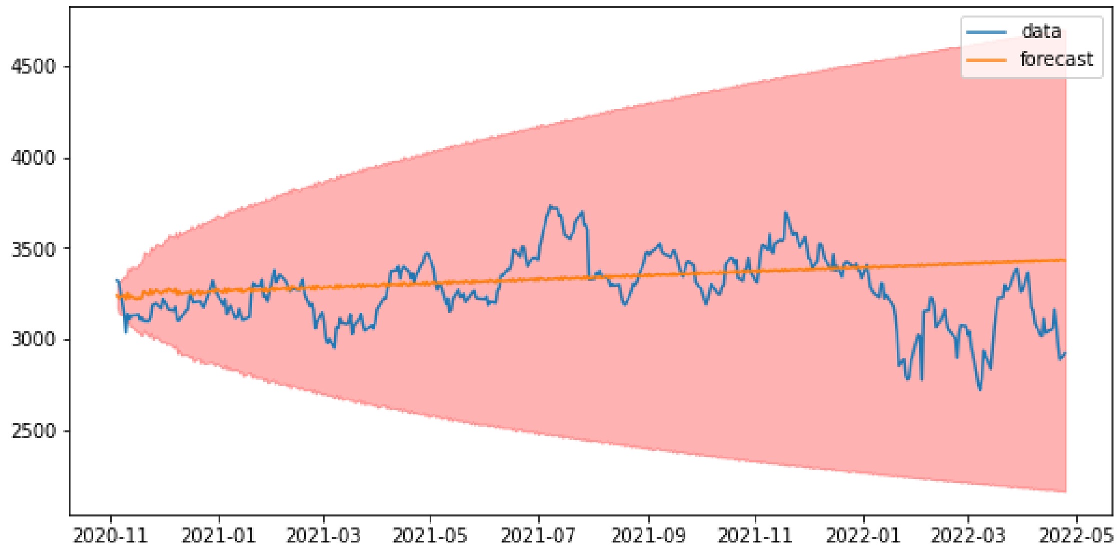

Figure 2: Prediction results of best model received from ARIMA stepwise search

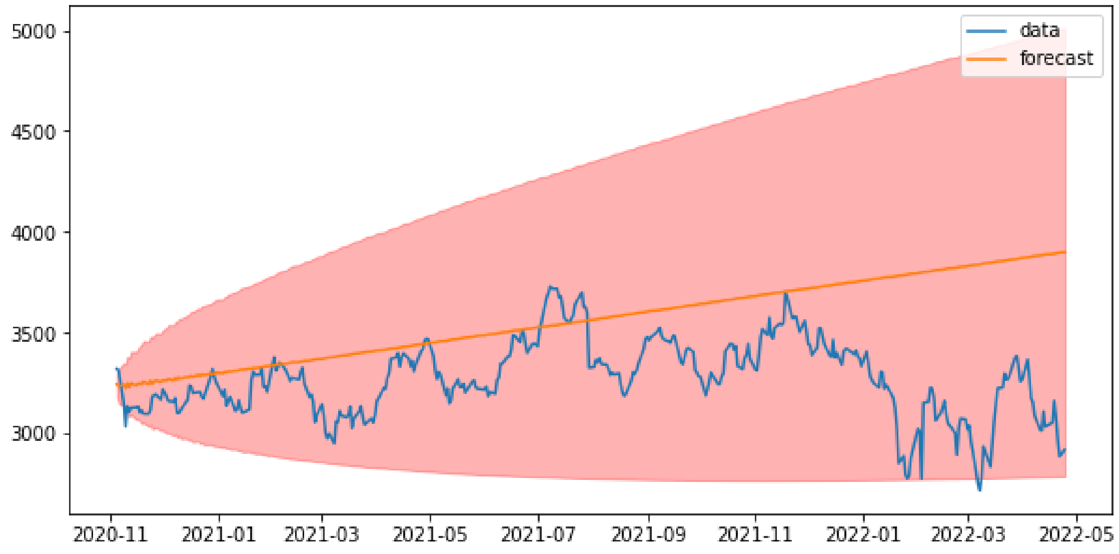

Figure 3: Prediction results of best model received from ARIMA Grid search

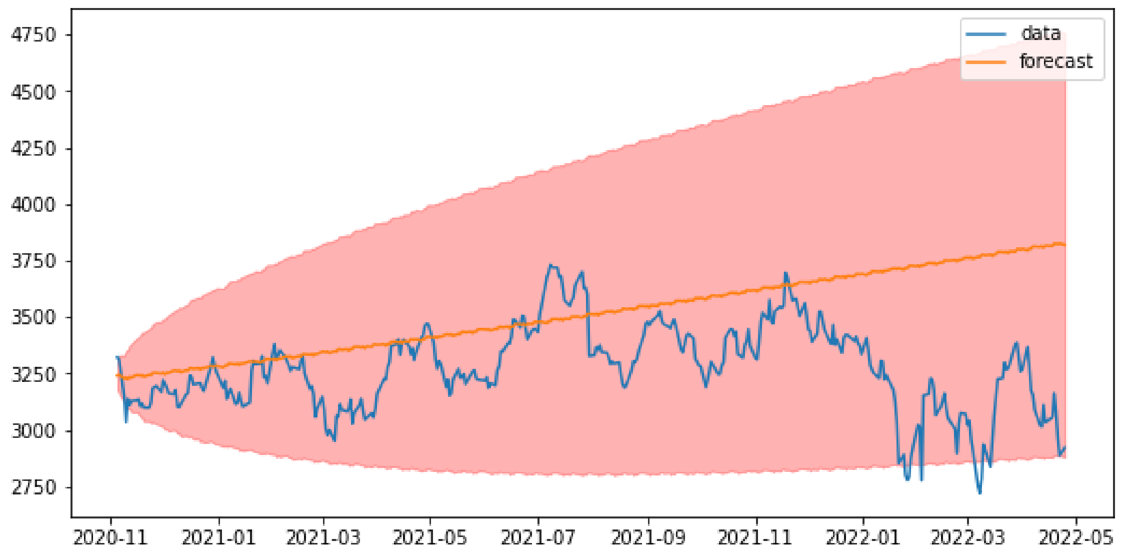

Figure 4: Prediction results of best model received from ARIMA Bayesian search

Search Method	**Best Hyperparameters received (p, d, q)(P, D, Q, s)**	**Time Taken to search (in seconds)**	**Minimum AIC value**	**Normalised RMSE**
Stepwise Search	(2, 1, 2)(1, 0, 1, 12)	126.6	14351.93	0.205
Grid Search	(2, 1, 2)(1, 0, 0, 3)	293.97	14361.53	0.395
Bayesian Search	(4, 0, 0)(0, 1, 1, 7)	78.77	14346.65	0.35

Table 1: Comparison of results received from stepwise, grid and bayesian search

From the above data table we can observe that the bayesian search is comparatively converge faster towards best set of hyperparameters than stepwise and grid search. It also requires less number of iterations to find optimal set of hyperparameters. Model accuracy is comparatively better in stepwise search.

4.5 LSTM Results

Below mentioned plots are drawn using the best models suggested from grid search, random search and bayesian search. The plots are drawn between actual vs predicted values on test dataset.

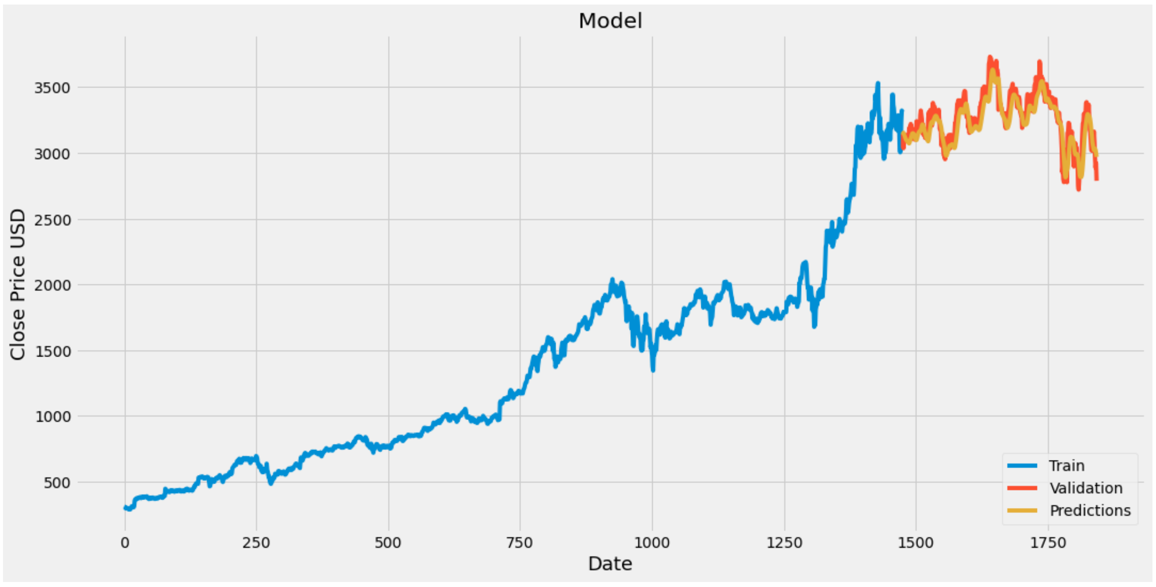

Figure 5: Prediction results of best model received from grid search applied on LSTM model

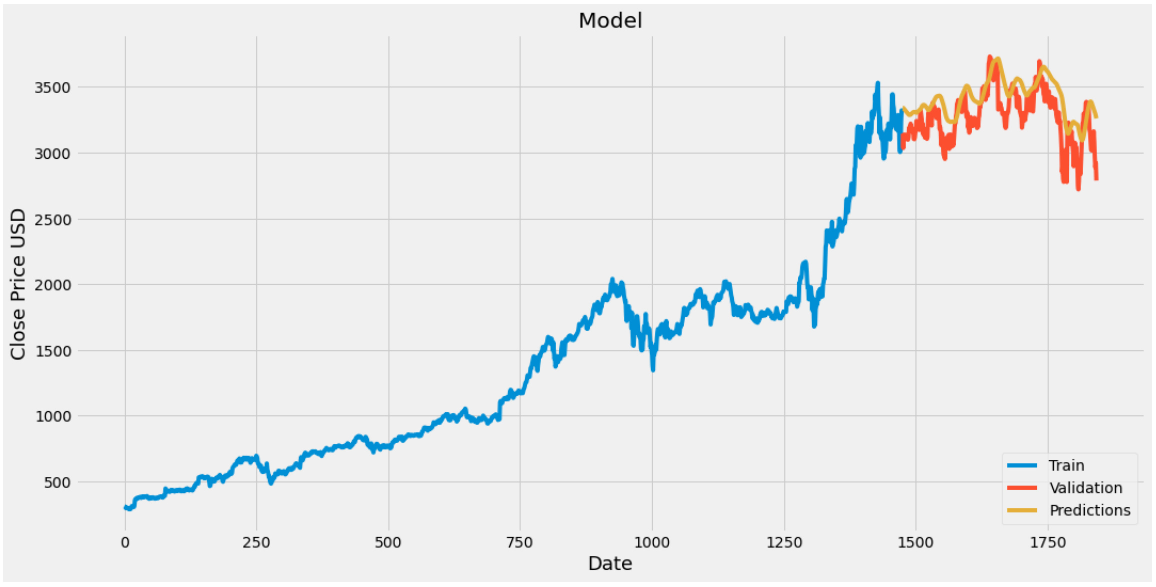

Figure 6: Prediction results of best model received from random search applied on LSTM model

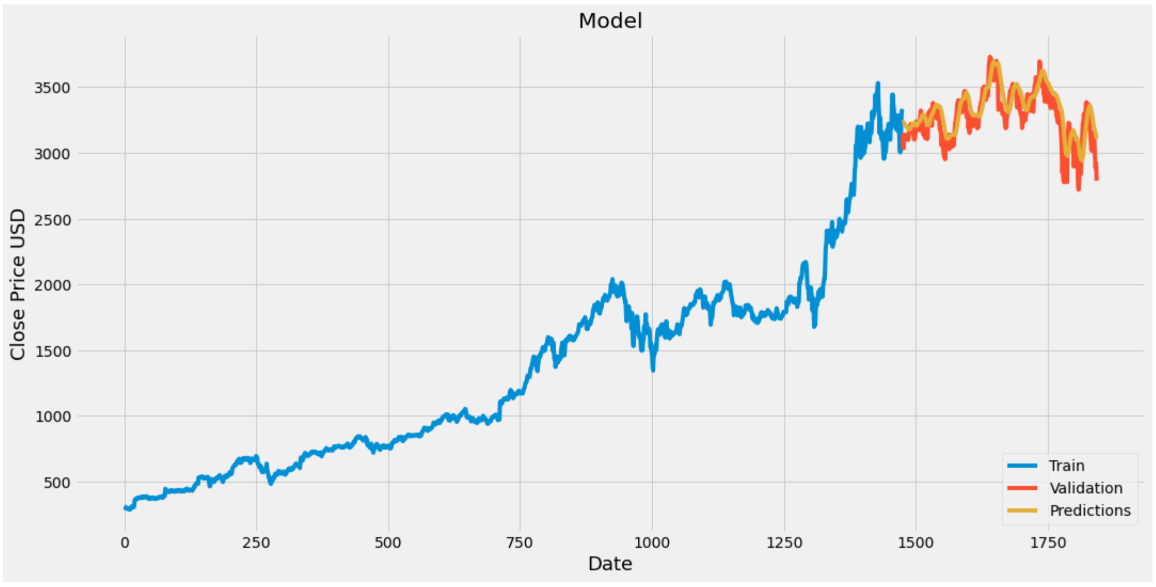

Figure 7: Prediction results of best model received from bayesian search applied on LSTM model

Search Method	**Best Hyperparameters received**	**Iterations**	**Time (in sec)**	**RMSE**	**Normalised RMSE**	**Parallel execution**
Grid Search	number of units : 64 number of epochs : 20 dropout rate : 0.2 batch size : 50 optimizer : RMSprop	108	960	101.86	0.10	Yes
Random Search	number of units : 16 number of epochs : 10 dropout rate : 0.1 batch size : 50 optimizer : RMSprop	20	165	192.14	0.19	Yes
Bayesian Search	number of units : 54 number of epochs : 5 dropout rate : 0.26 batch size : 25 optimizer : RMSprop	10	280	127.84	0.12	No

Table 2: Comparison of results received from Grid, Random and Bayesian search

All these methods are executed in multiprocessor environment. The hardware configuration used was AMD Ryzen 5 - 3500u quad core processor with 8 logical cores and 8GB RAM. Grid search and Random search were utilising all 8 logical core whereas Bayesian search was performed sequentially.

From the above table 2 We can observe that Grid search is an exhaustive search that performs

108 combinations of hyperparameters and requires comparatively more time than other methods. Because it tries out all possible combinations, its accuracy is also great. Since random search was performed with 20 iterations and on multiple processors, it was the fastest of these methods. Although Bayesian search is executed in a sequential manner, it converges very fast to find the optimal set of hyperparameters. Also, the prediction accuracy received from the best model suggested by bayesian search is better than random search and very close to grid search.

5 Conclusions

This work compares the capabilities of statistical and deep-learning models for time series forecasting. It is observed that the statistical model ARIMA struggles with capturing the non-linear patterns in the data, whereas the LSTM model overcomes this issue. The ARIMA model is comparatively faster than the LSTM, but accuracy is a major concern. Apart from model comparison, the hyperparameter tuning of both approaches is also performed. For ARIMA hyperparameter tuning, the results produced by stepwise search and Bayesian search are comparable. For LSTM hyperparameter tuning, the bayesian search turns out to be superior to others; it converges quickly, as well as the hyperparameter combination received from it has better prediction accuracy. In both ARIMA and LSTM, the grid search is the slowest one. Random search has the advantage over grid search is that it can select random points in hyperparameter search space. In random search, the selected combinations are sampled uniformly, so it has a higher probability that it can have an optimal set of hyperparameters in sampled data. Also, random search can be executed in parallel, which indicates it is better to perform random search than grid search. In comparison with random search, the experimental results show that the Bayesian optimization algorithm based on Gaussian process can achieve great accuracy in just a few samples.

6 Future Work

In future the bayesian search can also be performed in parallel. Different kind of recurrent neural networks such gated-recurrent unit (GRU) can also be tested. There is also scope to use hybrid models for time-series data.

References

[1] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207001001108

[2] A. M. D. Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011. [Online]. Available: https://doi.org/10.1198/jasa.2011.tm09771

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[4] L. Zhang, J. Lin, R. Qiu, X. Hu, H. Zhang, Q. Chen, H. Tan, D. Lin, and J. Wang, "Trend analysis and forecast of pm2.5 in fuzhou, china using the arima model," *Ecological Indicators*, vol. 95, pp. 702–710, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1470160X18306319

[5] Z. Li, J. Han, and Y. Song, "On the forecasting of high-frequency financial time series based on arima model improved by deep learning," *Journal of Forecasting*, vol. 39, no. 7, pp. 1081–1097, 2020.

[6] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[7] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for r," *Journal of statistical software*, vol. 27, pp. 1–22, 2008.

[8] S. Asadi, A. Tavakoli, and S. R. Hejazi, "A new hybrid for improvement of auto-regressive integrated moving average models applying particle swarm optimization," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5332–5337, 2012.

[9] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.