Stock Price Forecasting Using Recurrent Neural Networks With Increased Dynamic Elements

Stavros Ieronymakis
2645715

School of Business and Economics
MSc Finance: Duisenberg Honours in Quantitative Risk Management

Submitted: 30.06.2023
Supervisor: (dr.) Normal Seeger

Abstract

In this paper, the aim is to produce accurate stock predictions. Recurrent Neural Networks have made considerable advances in this field but results vary across dataset and applications. A Recurrent Autoregressive Integrated Moving Average (ARIMA) is compared to a Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The predictions generated are fed back into the training sets and new predictions are made giving them a more 'dynamic' element. Results suggest that for scarcity of data such an approach increases the predictive power of those model. Both 'dynamic' versions outperform the Recurrent ARIMA model especially for low amount of available data and short term predictions. NASDAQ 100 index was investigated with data collected ranging from 1 to 5 years. Predictions were made for 5 (week), 21 (month) and 63 (quarter) trading days.

Contents

1 Introduction

Stock price prediction has long been regarded as the 'Holy Grail' in the field of Finance. Extensive literature, consisting of numerous books and papers, has already been dedicated to this topic, and the volume of research in this area continues to grow. The ability to accurately forecast future stock prices holds immense value, as the potential for financial gains is enormous. As researchers explore new methodologies and approaches, their aim is to achieve (near) perfect predictability in stock markets.

Last couple of decades, humanity has witnessed the evolution of Artificial Intelligence (AI). What was considered as science fiction in the mid 19th century, it has found its way to our daily lives. One has to simply look into a mobile phone to find many applications that use AI or even the cars we drive (**Spector 2006**). In the 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence, authors Hendtlass and Ali, explain that a better understanding of the ways systems work and continuous development in computational resources, have increased the applications of Neural Networks (NNs). They are particularly effective at learning patterns in complex datasets and making predictions based on those patterns. Notably, NNs have been increasingly applied in finance from different angles such as forecasting stock prices, portfolio management, credit risk assessment and fraud detection (**Ding 2015** ; **Si 2017** ; **Jiang 2017**; **Shen 2018**).

Especially, stock price forecasting has been a vivid area of research (**Baba 1992**; **Devadoss 2013**; **Vrbka 2017**). In this study Recurrent Neural Networks (RNNs) are considered. The most widely used models proposed by the literature to conduct such analyses, are the Long-Short Term Memory model (LSTM) and Gated Recurrent Unit model (GRU), because of their similar architecture and design elements. LSTM and GRU are specifically designed to capture and model long-term dependencies in sequential data. (**Selvin 2017**; **Sethia 2019**; **Sunny 2020**). Although LSTM and GRU are both types of RNNs, so inherently dynamic, conventional LSTM/GRU models, which will be referred to as 'static' models, operate in a static manner. Meaning that they rely solely on the training data provided during the training phase. To address this limitation and enhance the models adaptability to evolving trends, a more 'dynamic' variation is proposed. Therefore, the first level of innovation, is feeding the predicted output back into the training set, allowing the model to incorporate and learn from its own predictions. These 'dynamic' LSTM/GRU models offer a promising approach to capturing and adapting to complex trends in the data, potentially outperforming the traditional 'static' LSTM/GRU models.

Using *Yahoo! Finance*, historical data was downloaded for the publicly traded NASDAQ 100 index (`^NDX`) in an attempt to capture high-tech market trends. Therefore, the data of interest are a time series of a particular market and the aim is to predict

the 'Close' daily prices of the sequential dataset.The secondary level of innovation, in this approach, lies in the proposal of an alternative perspective for the input features. Two additional features; medium and range, are engineered and fed as input features to the RNN models. All models ('static' and 'dynamic') will be evaluated compared to a benchmark; an Autoregressive Integrated Moving Average (ARIMA) which is a popular conventional time series econometric forecasting model used to analyse and predict future values based on the patterns and trends observed in historical data (**Adebiyi 2014**; **Mondal 2014**). To make the comparison meaningful, a Recursive element is added to the ARIMA model in an attempt to capture long and short term dependencies of the actual values, similar to the RNNs.

The objective of this paper is to investigate whether incorporating additional 'dynamic' design elements into LSTM/GRU models; as to fed the predictions back to the training set; can enhance their predictive performance. Furthermore, two new input features are introduced through feature engineering, aiming to go beyond the conventional input features used in the literature. This research seeks to explore alternative feature sets that have received limited attention so far. Therefore, the hypothesis put forth is that the integration of new features and a more 'dynamic' design approach would lead to improved predictive accuracy of the LSTM/GRU models across three distinct forecasting periods. A week, a month and a quarter of a year (5,21 and 63 trading days, respectively) will serve as the prediction windows and data is collected ranging from 1 to 5 years. Evidence suggest that when limited data is available, such an approach can increase the predictive performance of those models.

In the following section, the literature on ARIMA and 'static' LSTM/GRU models will be reviewed, providing a deeper understanding of their characteristics and performance in various applications. Next, in the methodology section, the newly engineered features and 'dynamic' models will be introduced, as well as the recursive element of the ARIMA. Section 4 presents the results, followed by a discussion section where some remarks are discussed over the results. Finally, in the last section, the performance is compared and some limitations and future research proposals are presented as concluding remarks.

2 Theoretical Background

AI is a relative wide term that incorporates many aspects such as Machine Learning (ML) and Deep Learning (DL). One major component are Neural Networks (NNs). In simple terms, a NN is a type of Machine Learning model that is inspired by the structure and function of the human brain. It consists of a series of connected nodes, or 'neurons', that are organized into layers (**IBM 2023**). Each neural network model has its own strengths and weaknesses, mainly derived from their architectural design and how they

process data. An RNN is inherently dynamic by design. This dynamic nature is what sets RNNs apart from other NNs and makes them great candidates for time series analysis. (**Dase 2010**; **Adebiyi 2012**).

Traditional neural networks assume independence among inputs, making them ineffective for sequential data and varying input/output sizes (**Bao 2017**). In contrast, RNNs are well-suited for sequential data, as they incorporate memory loops and recurrent hidden states (**Rumelhart 1986**). However, RNNs suffer from the vanishing gradient problem, where gradients diminish during back-propagation over long distances, hindering learning ability . This problem is addressed by LSTM/GRU networks, which utilize memory cells to mitigate the vanishing gradient problem and preserve information over multiple time steps (**Hochreiter 1997**).

2.1 Long Short-Term Memory (LSTM)

The LSTM has an architecture specially designed to address the vanishing/exploding gradient problem. It does so by adding a *cell state* into the *hidden state* typically found in an RNN. As already mentioned, LSTM are useful for capturing temporal dependencies and patterns in time series data. They achieve this by enabling the processing of sequences of data instead of individual time steps. This sequence of data is attained by a sliding window approach and is what gives this model its inherently dynamic feature (**Selvin 2017**). Below is a graphical representation of the process for individual time steps.

Going through the process one step at a time, beginning from the model initialization (Figure: [1]). An input x_{t-1} is fed into the model, and the model generates a hidden and cell state for further processing.

$$h_{t-1} \quad \text{(previous hidden state)}$$

$$c_{t-1} \quad \text{(previous cell state)}$$

In more detail, the next input x_t arrives, and the input gate is calculated.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + \sum_{i=t-w}^{t-1} W_{rc}c_i + b_i)$$

where σ is the sigmoid activation function defined as $\sigma(x) = \frac{1}{1+e^{-x}}$; W_{xi}, W_{hi}, W_{ci}, and W_{rc} are the weight matrices associated with the input x_t, previous hidden state h_{t-1}, previous cell state c_{t-1}, and previous cell states c_i within the rolling window, respectively. Lastly, w represents the number of time steps included in our rolling window and b_i is a bias term.

Next, the forget gate is calculated.

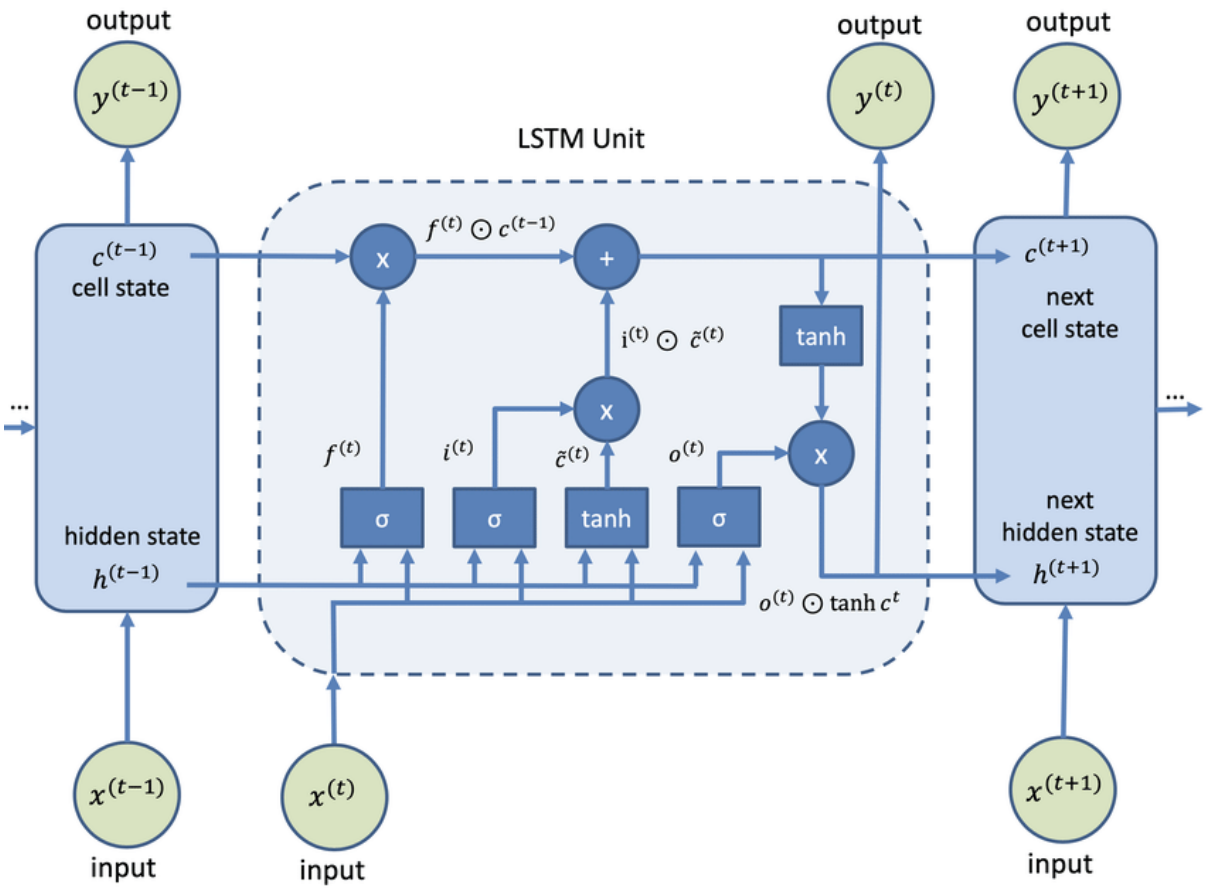


Figure 1: LSTM Architecture.

source: researchgate.net

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + \sum_{i=t-w}^{t-1} W_{rc}c_i + b_f)$$

Again, σ is the sigmoid activation function, W_{xf}, W_{hf}, W_{cf}, and W_{rc} are the weight matrices, w the number of time steps and b_f is a bias term.
Following, the output gate is calculated as

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + \sum_{i=t-w}^{t-1} W_{rc}c_i + b_o)$$

The candidate cell state is now calculated

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

where tanh is a nonlinear activation function commonly used in machine learning algorithms, defined as $\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$.
The cell state is now updated

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t + \sum_{i=t-w}^{t-1} W_{rc} c_i$$

where $\odot$ represents the Hadamard product, f_t is the forget gate output, c_{t-1} is the previous cell state, i_t is the input gate output, $\tilde{c}_t$ is the candidate cell state, and $\sum_{i=t-w}^{t-1} W_{rc} c_i$ captures the influence of previous cell states within the rolling window on the current cell state.

Finally, the hidden state is calculated as

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

where o_t is the output gate output and c_t is the updated cell state.

Therefore, in the sliding window approach, the LSTM model considers the relevant previous time steps within the rolling window to update the cell state, enabling it to capture temporal dependencies and patterns in time series data effectively making it by design 'dynamic'.

2.2 Gated Recurrent Unit (GRU)

GRU is another model designed to address the vanishing/exploding gradient problem in RNNs. Although, by combining the *cell* and *hidden* states into a single hidden state vector, it does so by simplifying the architecture introduced by LSTM, (**Cho2014**).

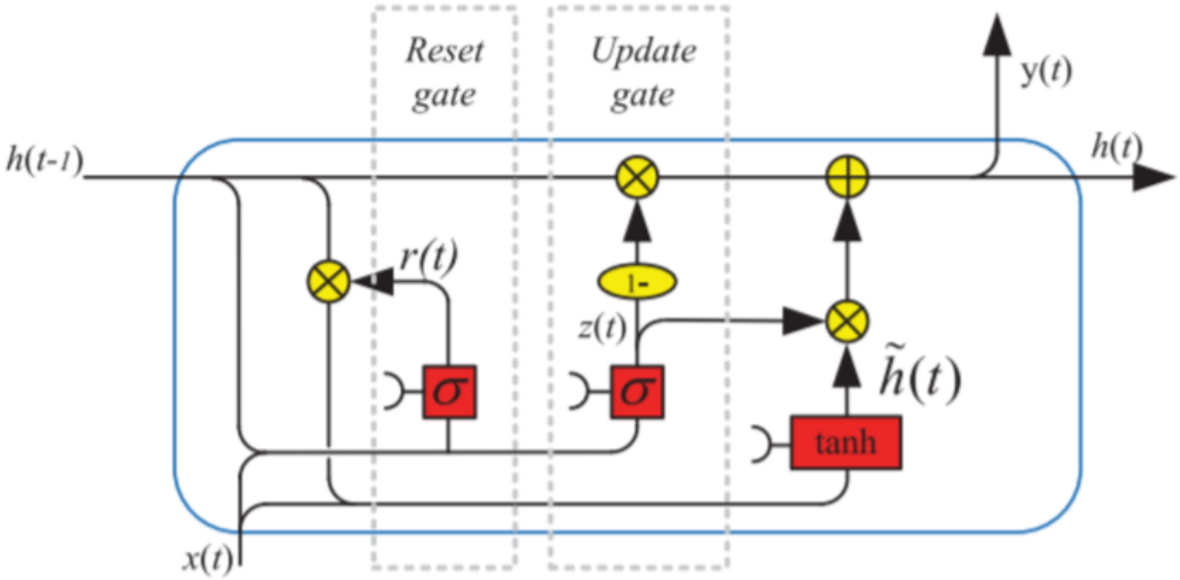


Figure 2: GRU Architecture.

source: researchgate.net

Starting from the model initialization, the GRU process is as follows. An input window in the form of a matrix X_{t-1} is provided to the model, which generates a hidden state matrix for further processing both being of shape (window size,input dimension).

$$H_{t-1} \quad \text{(previous hidden state)}$$

In more detail, when the next input window matrix X_t arrives, the model performs the following calculations.
Firstly, updating the gate

$$Z_t = \sigma(W_z \cdot \begin{bmatrix} X_t \\ H_{t-1} \end{bmatrix})$$

Similarly to the LSTM model, σ being the sigmoid activation function and W_z a weight matrix. By concatenating X_{t-1} and H_{t-1} vertically, we create a matrix $\begin{bmatrix} X_{t-1} \\ H_{t-1} \end{bmatrix}$ with a shape of $(\text{window size} \times 2, \text{input dimension})$ or $(\text{window size} \times 2, \text{hidden dimension})$, depending on the dimensionality of the input and hidden states.
Then the gate is reset

$$R_t = \sigma(W_r \cdot \begin{bmatrix} X_t \\ H_{t-1} \end{bmatrix})$$

In the last step, a Hidden State Candidate is computed

$$\tilde{H}_t = \tanh(W_h \cdot \begin{bmatrix} X_t \\ R_t \odot H_{t-1} \end{bmatrix})$$

With tanh and $\odot$ following similar definitions as in the LSTM model.
Finally, the Hidden State is updated

$$H_t = (1 - Z_t) \odot H_{t-1} + Z_t \odot \tilde{H}_t$$

The output of the GRU model can then be used as input to the next layer or as the final output.

In terms of usage and performance, LSTM and GRU are similar in many ways. Both models excel in handling sequential data with long-term dependencies, and their effectiveness can vary depending on the specific task and dataset. LSTM tends to have more parameters and may be more suitable for complex tasks that require precise memory management. GRU, with its simplified architecture, may offer advantages in terms of computational efficiency and can be a good choice for simpler tasks or when computational resources are limited (**Gao2021**).

2.3 AutoRegressive Integrated Moving Average (ARIMA)

Obviously, a benchmark is needed to compare against the relative performance of our models. Many conventional techniques exist such as Autoregressive (AR), Moving Average (MA), Vector Autoregression (VAR), to name a few (**Bratu 2012**). RNN models claim that they produce more accurate predictions of the next lag in a time series. On the other hand, empirical evidence suggest a mixture of results. In some cases ARIMA outperforming RNN models (**Yamak 2019**) and vice versa (**Siami 2018**). Therefore, an ARIMA analysis is needed to determine the performance of the typical 'static' LSTM/GRU and that of the 'dynamic'.

As for the ARIMA models to produce meaningful results, certain assumptions need to hold (**Ho 1998**).

Stationarity : Statistical properties of the series, such as mean, variance, and autocorrelation, do not change over time. Stationarity is important for ARIMA models because they rely on the assumption that the relationship between past and future values of the series remains constant.

Weak Dependence : Future values of the time series are dependent only on its past values and not on other external factors, for example significant external shocks or influences.

Linearity : Relationship between past and future values of the series can be described by linear equations so it can be modeled as a linear combination of its past values and past errors.

No Outliers : There are no significant outliers or extreme observations in the time series that could affect the model's estimation and forecasting accuracy.

No Seasonality : The time series data does not exhibit any seasonal patterns.

In the Methodology section, some assumptions will be taken as given and for some the appropriate tests will be conducted. But the overall general procedure is as follows.

Step 1: Data Preparation

A time series dataset needs to satisfy the stationarity assumption. If the data is non-stationary, a differencing technique needs to be performed until stationarity is achieved.

$$\begin{aligned} \text{First-order differencing:} \quad & \Delta y_t = y_t - y_{t-1} \\ \text{Second-order differencing:} \quad & \Delta^2 y_t = (\Delta y_t) - (\Delta y_{t-1}) = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ \text{Generalized differencing of order } d: \quad & \Delta^d y_t = \Delta(\Delta^{d-1} y_t) \end{aligned}$$

Step 2: Identification of Model Order

Determine the order of the AR (autoregressive) and MA (moving average) components of the ARIMA model.
- Autocorrelation Function (ACF): Examine the ACF plot to identify the lag at which the autocorrelation drops significantly. This provides an estimate of the order of the AR component (p). Calculated as

$$\rho_k = \frac{\text{Cov}(Y_t, Y_{t-k})}{\sqrt{\text{Var}(Y_t) \cdot \text{Var}(Y_{t-k})}}$$

- Partial Autocorrelation Function (PACF): Analyze the PACF plot to identify the lag at which the partial autocorrelation cuts off. This indicates the order of the MA component (q).

Step 3: Model Fitting

Fit the ARIMA model to the data. The general equation for an ARIMA(p, d, q) model is:

$$(1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)(1 - B)^d X_t = (1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q)\epsilon_t$$

with X_t is the observed time series, B is the backshift operator ($BX_t = X_{t-1}$), ϕ_i are the autoregressive coefficients ,θ_i are the moving average coefficients and d is the differencing order.

Step 4: Model Estimation

Use the chosen model order to estimate the parameters of the ARIMA model.
- Maximum Likelihood Estimation: Estimate the parameters by maximizing the likelihood function. The objective is to minimize the sum of squared errors between the actual values y_t and the predicted values $\hat{y}_t$ for each time step t. The estimation process aims to find the optimal parameter values that minimize this error, represented as:

$$\hat{\theta} = \arg\min_{\theta} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2$$

Step 5: Model Diagnostics

Evaluate the goodness-of-fit of the ARIMA model.
- Residual Analysis: Analyze the residuals to check for any patterns or significant autocorrelation. This can be done by examining the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots.

- Quantile-Quantile plot (QQ): Observations are sorted in an ascending order and the empirical quantiles are calculated as

$$\text{Empirical Quantile} = \frac{i - 0.5}{n}$$

which gives the proportion of data points equal or below a certain value. Here, i is the observation index, n is the total number of observations. Then the theoretical quantiles are determined assuming a certain distribution, which can be calculated as

$$\text{Theoretical Quantile} = F^{-1}(p)$$

where F^{-1} is the inverse cumulative distribution function (quantile function) of the assumed distribution, and p is the desired probability.

Step 6: Model Forecasting

Use the fitted ARIMA model to make future predictions.
- One-Step Ahead Forecast: Generate one-step ahead forecasts by recursively using the observed data and estimated model parameters.

$$\hat{y}_{t+1} = \hat{c} + \sum_{i=1}^{p} \hat{\phi}_i y_{t+1-i} + \sum_{i=1}^{q} \hat{\theta}_i \epsilon_{t+1-i}$$

- Multi-Step Ahead Forecast: Generate forecasts for multiple time steps ahead using the previously predicted values.

$$\hat{y}_{t+h} = \hat{c} + \sum_{i=1}^{p} \hat{\phi}_i y_{t+h-i} + \sum_{i=1}^{q} \hat{\theta}_i \epsilon_{t+h-i}$$

In these equations, $\hat{y}_{t+1}$ represents the one-step ahead forecast, $\hat{y}_{t+h}$ represents the forecast for h time steps ahead, $\hat{c}$ represents the estimated constant term, $\hat{\phi}_i$ is the estimated autoregressive (AR) coefficients, $\hat{\theta}_i$ the estimated moving average (MA) coefficients, y_t the observed value at time t, ϵ_t the residual at time t, and p and q represent the order of the AR and MA components, respectively. The reader should be aware that these equations assume a basic ARIMA model without differencing ($d = 0$). If differencing is involved, additional terms and coefficients would be included in the equations. But as it will be shown later $d = 1$ will be used in the analysis.

Step 7: Model Evaluation

Assess the accuracy and performance of the ARIMA model.

- Mean Absolute Error (MAE): Calculate the average absolute difference between the predicted values and the actual values.

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

- Mean Squared Error (MSE): Compute the square root of the average of the squared differences between the predicted values and the actual values.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$$

- R-squared (Coefficient of Determination): Measure the proportion of the variance in the dependent variable that can be explained by the model.

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2}$$

- Akaike Information Criterion (AIC): A measure of the relative quality of the model, balancing the goodness-of-fit and the complexity of the model.

$$\text{AIC} = -2\log(L) + 2k$$

where L is the likelihood of the model and k is the number of estimated parameters.

Lower values of MAE, RMSE, and AIC indicate better model performance, while higher values of R-squared indicate a better fit of the model to the data. The reader should be aware that more model evaluation metrics exist but these are the metrics that will be used in this analysis.

3 Methodology

This section outlines the methodology employed to develop and evaluate the LSTM and GRU models for forecasting future values. First, the 'static' versions are discussed and in the following section, the differences of the 'dynamic' version are presented.

3.1 Part A: 'static' LSTM / GRU

Data Collection

Historical stock market data for the NASDAQ 100 index (^NDX) was obtained from Yahoo! Finance using the `yfinance` library. The dataset was retrieved for a period ranging from 1 to 5 years. Daily (trading days) data are available starting from 12/06/2018

until 19/06/2023 resulting in various observation length and are displayed in Table: [1] . Six features are available to download directly; Open, High, Low, Close, Adj Close and Volume. 'Open' being the opening price, 'High'/'Low' being the highest/lowest price the stock has achieved in that day, respectively. 'Close' is the closing pricing of the stock for that date. The rest features are explained in finance.yahoo.com site. Since they will be dropped in later steps, they are not included here.

Period	Observations	End Date
5 years	1259	18/06/2023
4 years	1009	17/06/2023
3 years	754	12/06/2023
2 years	502	12/06/2023
1 year	250	19/06/2023

Table 1: Observations by Period

Data Preprocessing

The collected data was processed to engineer additional features to serve as inputs for the models. Specifically, the "Medium" feature was computed as the average of the daily High and Low prices, and the "Range" feature represented the difference between the High and Low prices. 'Medium' feature was engineered to smooth 'High/Low', while 'Range' was designed to capture the volatility aspect of those features. Unnecessary features were removed, and the remaining [Open, High, Low, Medium, Range, Close] were retained for further analysis. Features as 'Adj Close' and 'Volume' were dropped to reduce dimensionality issues (**Koppen 2000**). Therefore, our initial dataset is a (n , 6) matrix, maintaining the 6 dimension input as other similar studies but introducing newly engineered features to increase model performance.

$$\text{Medium} = \frac{\text{High} + \text{Low}}{2} \qquad \text{Range} = \text{High} - \text{Low}$$

The training data and labels are split into training and validation sets using the `train_test_split` function from `scikit-learn`. A test size of 0.2 (20%) is specified, indicating that 20% of the data will be used for validation. The purpose of splitting the training data into training and validation sets is to evaluate the model's performance during training and prevent overfitting (**Koppen 2000**).

Data Scaling

To ensure uniformity in the range of input data and increase computational performance, a `MinMaxScaler` was applied to scale the filtered data between 0 and 1. The

scaler was fitted on the training data and later applied to the validation and testing data.

Data Windowing

The input data was created using a sliding window approach (Figure: [3]). A window size between w = [5, 21, 63] trading days was tested. Meaning that at each time step, the model was trained using the previous w days of data to make predictions for 'Close'. Starting from the beginning of the dataset, the data is iterated over and windows of data are created. Each window includes the past 'window size' observations, which consist of the input features as well as the corresponding target value. By sliding the window by one step at a time, we ensure that each observation is included in multiple windows, allowing the model to learn from different contexts. The specific window sizes were selected in an attempt to capture a shorter period of one week (5), a medium range of a month (21) and a longer period of a quarter (63). Importantly, these windows will also serve as the forecasting periods.

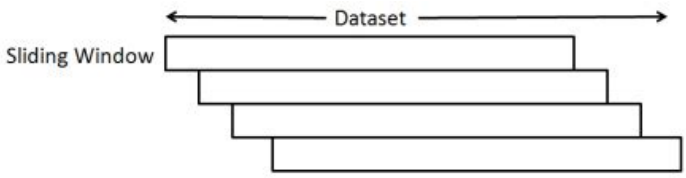


Figure 3: Sliding Window approach.

source: researchgate.net

Model Architecture

The s_LSTM/s_GRU model was implemented using the `Keras` library with a TensorFlow backend. It consisted of a single s_LSTM/s_GRU layer with a user-specified hyperparameters consisting of number of units (neurons), activation function, batch size, epochs and window sizes . A dense output 1 unit layer was added to predict the next day's closing price. The models were compiled using the Adam optimizer and mean squared error loss (MSE). Using trial-and-error approach, different architectures were tested. A single layer outperformed double and triple layered models.

Hyperparameter Tuning

To identify the optimal hyperparameters for the s_LSTM/s_GRU models, a manual search over different combinations of hyperparameter was conducted. This resulted in 243 (3^5) possible combinations . A parameter grid was defined, including the parameters listed in Table: [2] . The mean squared error (MSE) was used as the evaluation metric, and the model with the lowest MSE was selected as the best model.

Parameter	Values
units	[50, 100, 150]
epochs	[30, 50, 100]
batch_size	[32, 64, 128]
activation	['relu', 'tanh', 'sigmoid']
window size	['5', '21', '63']

Table 2: Parameter Grid

Model Evaluation

The model is trained on the training data and evaluated on the validation data at the end of each epoch during training. This allows monitoring of the model's loss on unseen data and helps in making decisions about model training, such as early stopping. The `EarlyStopping` callback from `tensorflow.keras` library is used during the training of the LSTM model. The patience parameter is set to 5, meaning that training will be stopped if there is no improvement in the validation loss for 5 consecutive epochs. This approach helps prevent the model from overfitting the data by automatically terminating the training process when the validation loss ceases to decrease. By employing early stopping, unnecessary computations are avoided, saving time and resources. It enables the model to be trained for an optimal number of epochs, striking a balance between underfitting and overfitting. The best model obtained prior to early stopping is then utilized for evaluation and future predictions.

Performance Metrics

To assess the performance of the s_LSTM/s_GRU model, several evaluation metrics were calculated. These included the mean squared error (MSE), mean absolute error (MAE), and R-squared (R^2) score. To make the MSE more interpretable and comparable to the original data range, it is necessary to scale it back. This involves reversing the scaling transformation applied to the data during preprocessing. The MSE is scaled back to the original data range by multiplying it with the squared difference between the maximum and minimum values of the 'Close' feature in the original data. This rescales the MSE to match the magnitude of the original 'Close' values.

Visualization

Learning curves are plotted for each window size in the parameter grid. These curves show the training and validation loss of the model over the epochs, providing a visual representation of the model's convergence and overfitting tendencies.

Forecasting

The best model is used to forecast future values based on the dynamic window size. The forecasted values are then compared to the actual values for a specific window size. The forecasted and actual 'Close' prices are plotted to visualize the performance of the model in predicting future trends.

A process schematic has been included in the Appendix section (Figure: [15]), as well as descriptive statistics of the available datasets (Tables: 17, 18, 19, 20, 21).

3.2 Part B: 'dynamic' LSTM / GRU

The process is the same for the 'dynamic' version of the models, apart from the forecasted values. In the 'static' versions, forecasts are compared against actual values and the performance is evaluated. In this version, the forecasted values are fed back into the training sets and the models are then re-trained using the extended training sets. As a result, a new set of forecasts are produced - for each window size - and compared / evaluated against the actual values, again. For the selected window sizes this results in 89 total forecasts which are then fed back to the initial training sets. For different periods this means different percentage increase of observations as shown in Table[3].

Period	Increase (%)
5 years	0.070691025
4 years	0.088206145
3 years	0.118037135
2 years	0.177290837
1 year	0.356

Table 3: Percentage Increase of Observations by Period

Similarly, a flowchart of the process is included in the Appendix together with the extended training sets descriptive statistics.

3.3 Part C: ARIMA(p,d,q)

Data Collection

Historical data is collected using the same method as explained in previous sections. Therefore, it will not be covered again. The only difference is that an end date of 23/06/2023 is used. That results in the subsets of data displayed in Table:[4].

Period	Observations	End Date
5 years	1257	23/06/2023
4 years	1007	23/06/2023
3 years	755	23/06/2023
2 years	503	23/06/2023
1 year	251	23/06/2023

Table 4: Observations by Period

Data Preprocessing

For the ARIMA model, 'Close' prices are used so there is no need to feature engineer additional features, as was done for the RNN models. Therefore all features but Closing prises are dropped. Upon inspection no missing values are detected and there is no need for outlier detection as all data is directly downloaded using yfinance library. Summary statistics tables of the corresponding subsets can be found in the Appendix section (Tables:[22] - [26]).

Stationarity Test

As discussed previously, ARIMA models come with certain assumptions. In this paper, all assumptions will be taken as valid except for the stationarity that will be tested using a Augmented Dickey-Fuller (ADF) test. In this test the null hypothesis is considered that the time series has a unit root, therefore indicating non-stationarity and the alternative hypothesis being that the time series is indeed stationary.

$$\text{ADF Statistic} = \frac{\text{Estimated Coefficient}}{\text{Standard Error}}$$

with the Estimated Coefficient representing the coefficient obtained from the regression of the differenced series on the lagged values of the series and the Standard Error is the standard error of the estimated coefficient.

It measures the strength of the relationship between the current observation and the lagged values of the series. The more negative the ADF statistic, the stronger the evidence against the presence of a unit root and the more likely the series is stationary or equivalently if the p-value is less than 0.05 (**Mushtaq 2011**).

ACF / PACF Analysis

Next step is to examine the ACF/PACF so the correlation between our time series and its lagged values can be calculated. The correlation is calculated using

$$\rho_k = \frac{\sum_{t=k+1}^{N}(x_t - \bar{x})(x_{t-k} - \bar{x}_k)}{\sqrt{\sum_{t=k+1}^{N}(x_t - \bar{x})^2 \sum_{t=k+1}^{N}(x_{t-k} - \bar{x}_k)^2}}$$

with ρ_k being the correlation coefficient at lag k, x_t the value of the time series at time t, $\bar{x}$ the mean of the time series, x_{t-k} the value of the time series at lag k time units earlier, $\bar{x}_k$ the mean of the time series at lag k time units earlier and N the total number of observations in the time series. Outputs are plotted in a graph together with the time series after applying differencing technique.

Model build and forecasting

Now the data is split into training and test sets using the same 80%/20% analogy as in the RNN models. The model is then trained by fitting an ARIMA model with (p,d,q) parameters. The optimal parameters per subset of data is identified from the ACF/PACF analysis earlier.As a result, $\hat{y}$ estimates are produced for each time step and then compared to the actual value of the dataset. The predicted value is then added to the training data for the next iteration, and the process is repeated for subsequent time steps (Recursive Forecasting). A Recursive Forecasting approach was selected over a simple ARIMA model to increase comparability of the models. With every prediction being added back to the model, it is expect from the model to capture the long-term effect while adjusting for the short-term individual actual values, similar to what LSTM/GRU are designed to capture.

Model Evaluation and Visualization

The forecasted output is evaluated using MAE,MSE and R^2 metrics, discussed previously. The results are plotted against the actual values of the data subset. Here two approaches will be used. First approach, predicted and actual closing prices are plotted for the full length of the test set as a single plot after the forecasting loop and the evaluation metrics are calculated. Second approach, predicted and actual closing prices are plotted for the test set separately for different prediction horizons (5,21 and 63 days). Evaluation metrics (MAE, MSE, R^2) for each prediction horizon are calculated individually. That way the model can be evaluated on its performance as a whole, but splitting the forecasts in the forecasting periods, equivalently to the RNN models, should help with comparability.

Model Diagnostics

To help access the goodness of fit of our trained model and potentially indicate any assumptions violations, a diagnostics plot is included in the Appendix for both approaches (Figure: [17]). These diagnostics include a histogram distribution of the residuals. A normal distributions is desired. Also, a Quantile-Quantile (Q-Q) plot is added to compare the distribution of the residuals to the theoretical normal distribution. A diagonal line is optimal.In addition, an Autocorrelation plot is added to help with identifying any remaining patterns or seasonality in the residuals. No lagged values should fall outside the confidence bands. Lastly, the residuals are plotted so their patterns can be observed. Ideally, they should resemble a white noise distribution.

4 Results

In this section the results from training the LSTM/GRU models will be presented. First for the 'static' versions, followed by the 'dynamic', for both LSTM and GRU models.

4.1 Part A: 'static' LSTM / GRU

After downloading the data and pre-processing them as described in the Methodology section, the models are trained. First a summary of the optimal hyperparameters and evaluation metrics will be shown, followed by the learning curves of each model. Next, the forecasting versus actual values are plotted and finally, the evaluation metrics for each forecasting window are presented.

s_LSTM

Now, the LSTM model is ready to be trained and produxe forecasts for the three different forecasting periods of 5, 21 and 63 trading days. In Table:[5] the summary of optimal hyperparameters and evaluation metrics of the s_LSTM model are displayed for each subset of datasets.

Years	Units	Epochs	Batch Size	Activation	Window Size	MAE	MSE	R^2
5	150	30	32	sigmoid	5	0.019682986	71736.1874	0.952630462
4	150	100	32	sigmoid	5	0.026608998	95512.0462	0.935188616
3	150	30	64	sigmoid	5	0.036172923	90384.05624	0.904920143
2	150	50	64	sigmoid	21	0.034537416	61673.98566	0.679165662
1	50	100	64	sigmoid	5	0.079127638	195314.4698	-0.536686613

Table 5: Summary of 'static' LSTM optimal parameters and evaluation metrics

Next, the learning curves of the training subsets (80%) versus the valuation subsets (20%) are displayed in Figure:[4].The blue solid line represents the training loss while

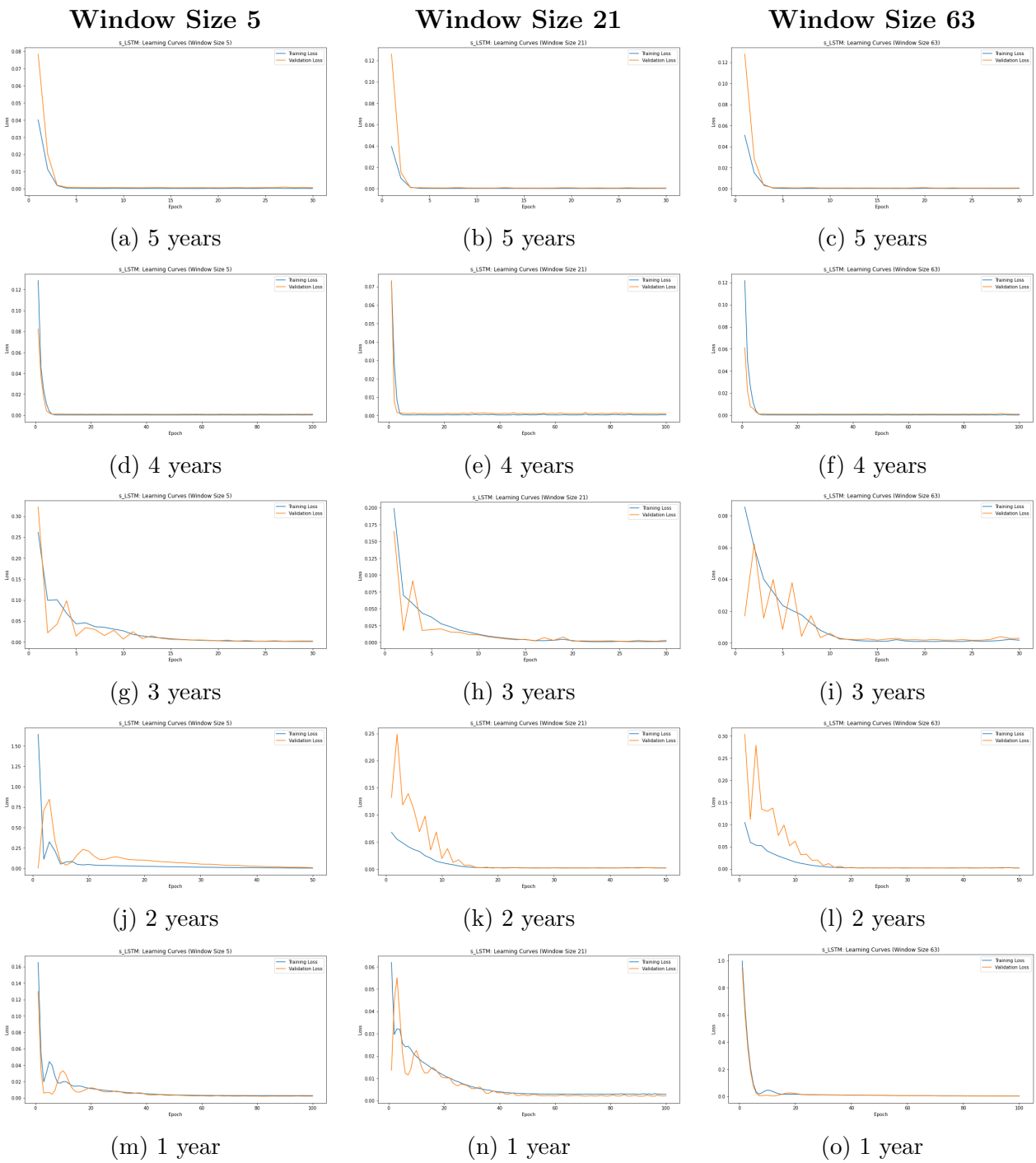
the orange solid line is the validation loss of each model. Each column represents the different window sizes, while each row is for the different subsets of data. Starting from 5 years worth of data on the top row, to 1 year of collected data in the last row.

Following, in Figure:[5] the forecasts values are plotted against the actual values of the dataset. The solid blue line represents the Actual values of the dataset, while the solid orange line are the forecasted values. Again each column represents the different forecasting windows of 5, 21 and 63 trading days, while each row is for the specific subset of years of data.

Lastly, in Table:[6], the evaluation metrics of the s_LSTM model for each forecasting period are displayed. Metrics of MAE, MSE and R^2 for each subset of data are included.

	Forecast 5	**Forecast 21**	**Forecast 63**	**Years**
MAE	0.008976919	0.008978025	0.009788194	
MSE	0.000144337	0.000121571	0.000130074	5
$\mathbf{R^2}$	0.784131641	0.759326712	0.962754493	
MAE	0.01345452	0.012120987	0.011838121	
MSE	0.000226055	0.000184565	0.000199873	4
$\mathbf{R^2}$	-1.543682642	0.898671468	0.923099281	
MAE	0.014539585	0.018111869	0.017059049	
MSE	0.000277167	0.000588671	0.000468962	3
$\mathbf{R^2}$	0.312606936	0.825807248	0.889408764	
MAE	0.027526967	0.020205413	0.016049227	
MSE	0.00097186	0.000524439	0.000379244	2
$\mathbf{R^2}$	0.697550811	0.723127965	0.915800446	
MAE	0.019164111	0.015418136	0.025526915	
MSE	0.000421673	0.000390915	0.001146721	1
$\mathbf{R^2}$	0.114859332	0.570580311	0.949075289	

Table 6: s_LSTM evaluation metrics for different years of data



(a) 5 years (b) 5 years (c) 5 years

(d) 4 years (e) 4 years (f) 4 years

(g) 3 years (h) 3 years (i) 3 years

(j) 2 years (k) 2 years (l) 2 years

(m) 1 year (n) 1 year (o) 1 year

Figure 4: Training Loss — Validation Loss —
s_LSTM Learning Curves across different window sizes and years of data

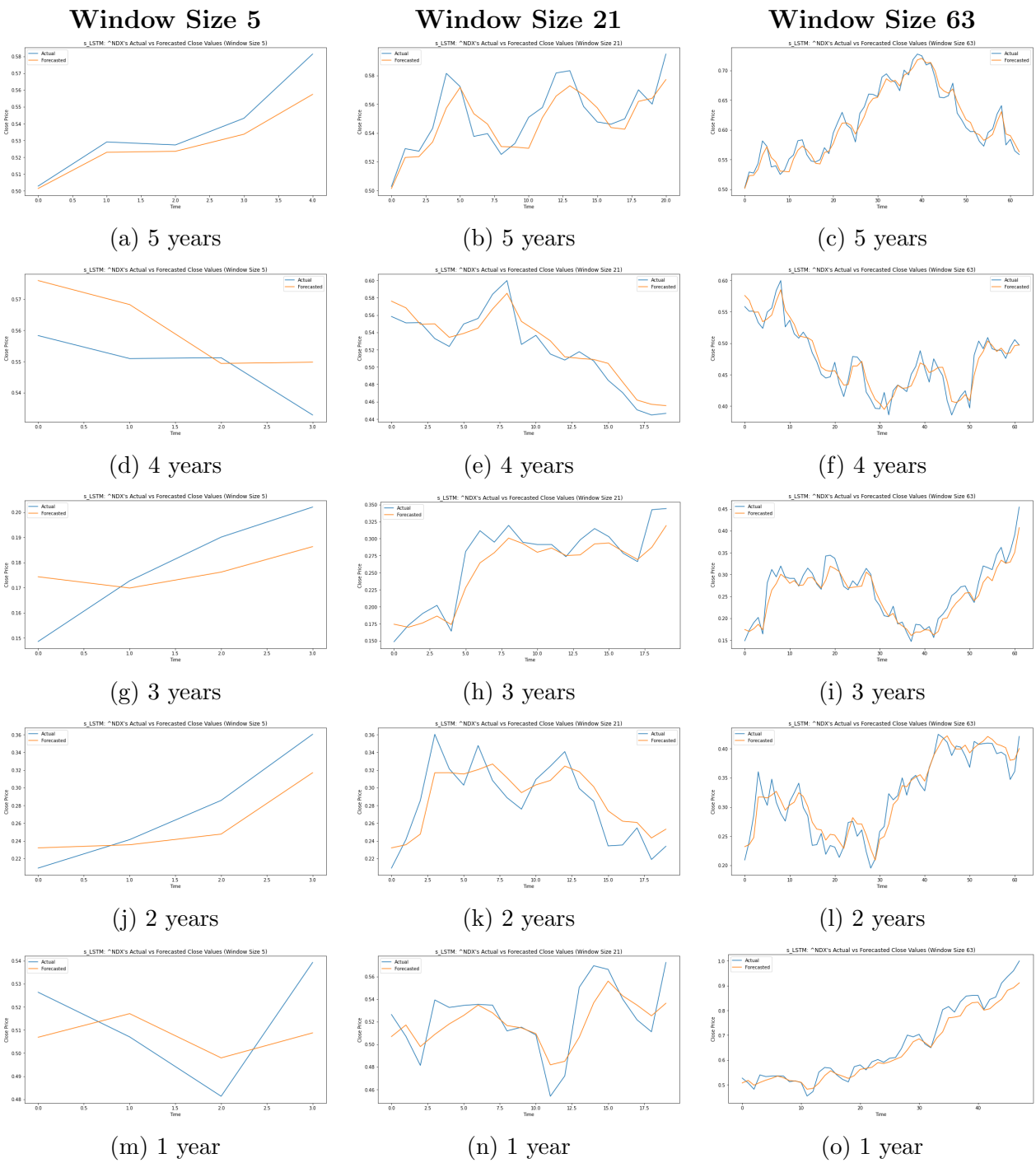


Figure 5: s_LSTM Forecasted vs Actual values across different window sizes and years of data

s_GRU

Similar to the s_LSTM model, the summary of the hyperparameters per subset of data and the corresponding evaluation metrics are displayed in Table:[7].

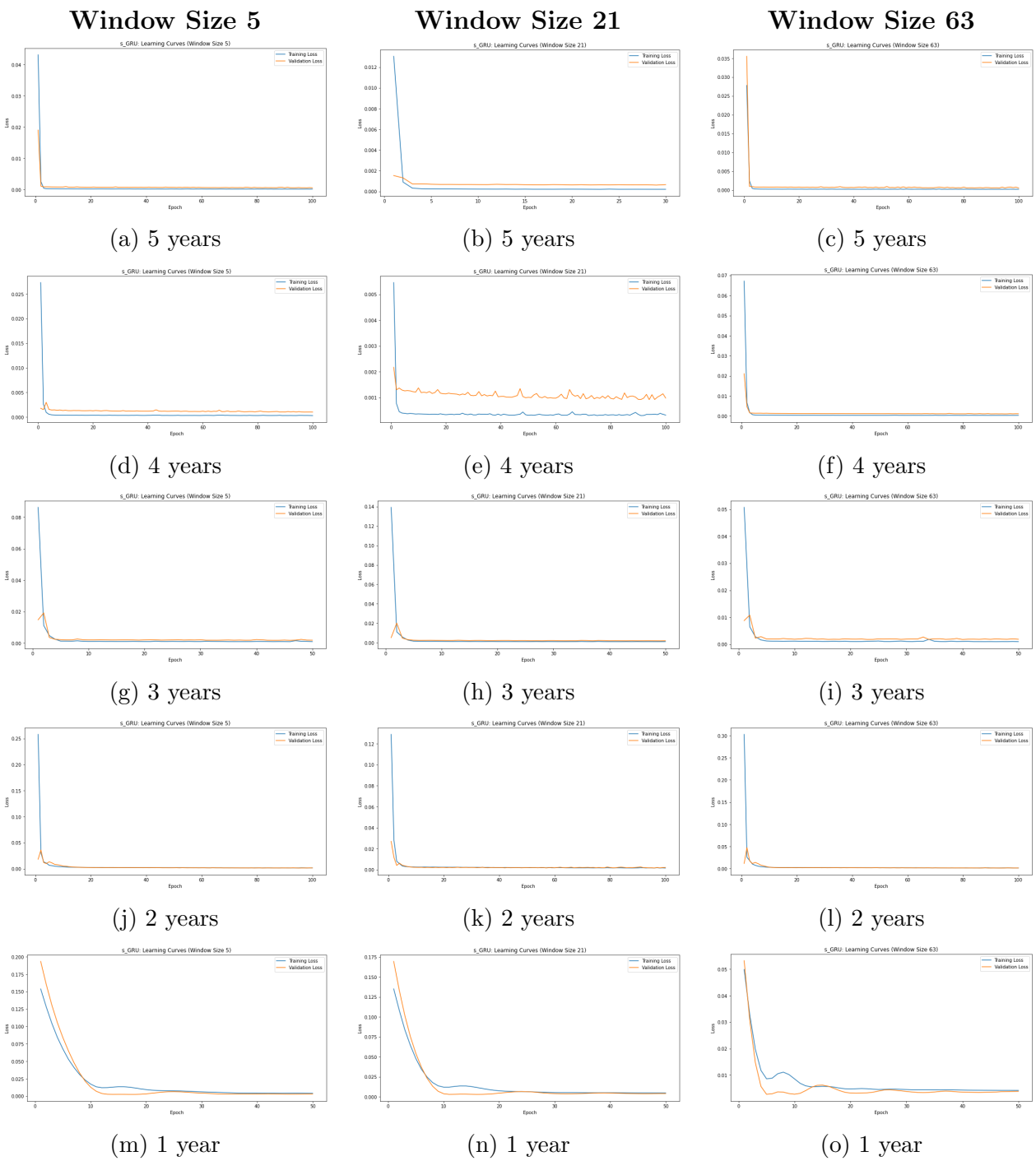
Years	Units	Epochs	Batch Size	Activation	Window Size	MAE	MSE	R^2
5	100	100	32	tanh	5	0.025850792	127876.9782	0.921830889
4	50	100	32	tanh	63	0.029353934	115365.3808	0.920683683
3	50	30	32	relu	5	0.04115505	112968.1036	0.85918331
2	150	100	32	relu	21	0.038382004	74521.39708	0.616819219
1	50	50	128	relu	63	0.323438538	2247885.944	-0.933908985

Table 7: Summary of 'static' GRU optimal parameters and evaluation metrics

Next, the learning curves across the three different window sizes and different years of data are shown in Figure: [6]. Again, the training loss is in the blue solid line while the validation loss is in the orange solid line. Each row represents different dataset per year,starting from the top for 5 years worth of data, to the last row for 1 year of data. Horizontally, each column represents the different forecasting windows, starting on the left for window size of 5, 21 and 63 trading days. In Figure:[7] the forecasted values in the solid orange line are plotted against the actual values represented by the solid blue line. Lastly, the evaluation metrics of the predictions for each forecasting window and years of data are summarized in Table:[8].

	Forecast 5	Forecast 21	Forecast 63	Years
MAE	0.011987684	0.011328923	0.009914025	
MSE	0.000238051	0.000176048	0.000142602	5
R^2	0.313923683	0.748402309	0.955472611	
MAE	0.007946731	0.009644782	0.011682035	
MSE	7.99E-05	0.000138612	0.000199676	4
R^2	-0.191852677	0.897002471	0.944852686	
MAE	0.020015482	0.019269699	0.01720525	
MSE	0.000656105	0.000561565	0.000450722	3
R^2	-4.339520192	0.047432042	0.847352138	
MAE	0.016421254	0.020173969	0.016714414	
MSE	0.000302459	0.000560901	0.000431408	2
R^2	-0.201902036	0.676879734	0.872790136	
MAE	0.051297611	0.03962212	0.059892461	
MSE	0.003303027	0.002040086	0.004643873	1
R^2	-6.475743759	-0.938963675	0.758292454	

Table 8: s_GRU evaluation metrics for different years of data



(a) 5 years (b) 5 years (c) 5 years

(d) 4 years (e) 4 years (f) 4 years

(g) 3 years (h) 3 years (i) 3 years

(j) 2 years (k) 2 years (l) 2 years

(m) 1 year (n) 1 year (o) 1 year

Figure 6: Training Loss — Validation Loss —
s_GRU Learning Curves across different window sizes and years of data

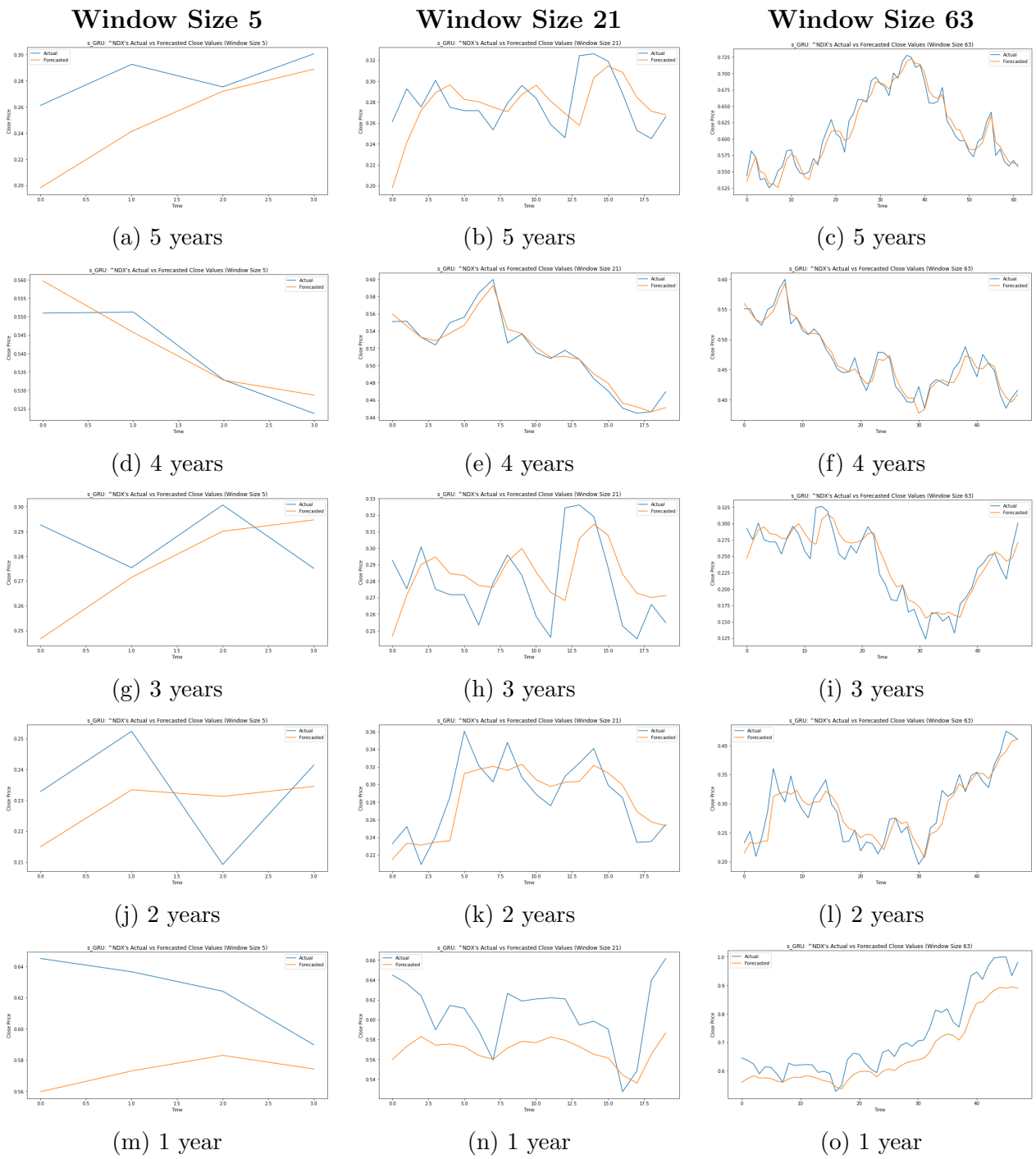


Figure 7: s_GRU Forecasted vs Actual values across different window sizes and years of data

4.2 Part B: 'dynamic' LSTM / GRU

Now the forecasts are fed back into the training set and the models are re-trained. A similar structure, as in the 'static' versions, will be used to exhibit the results. First a summary of the optimal hyperparameters and evaluation metrics will be shown. Subsequently, the learning curves and the forecasted versus actual values for each model will be featured. Finally, the evaluation metrics for each forecasting window will be exhibited.

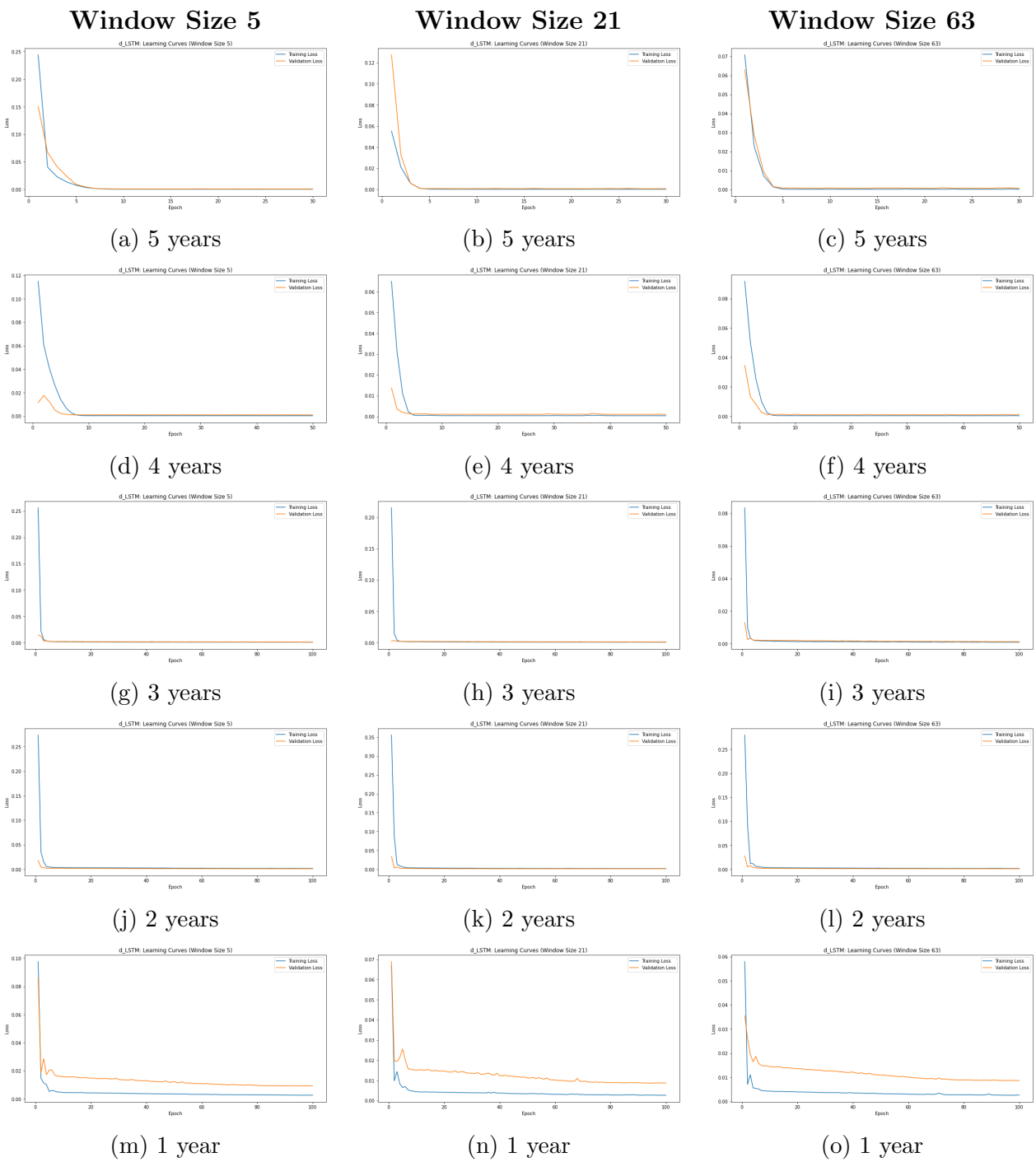
d_LSTM

Now, the d_LSTM model is re-trained using the extended training set and produce forecasts for the three different forecasting periods. In Table:[9] the summary of optimal hyperparameters and evaluation metrics of the d_LSTM model are displayed for each subset of datasets across different years.

Years	Units	Epochs	Batch Size	Activation	Window Size	MAE	MSE	R^2
5	150	30	32	sigmoid	63	0.024399912	106112.8411	0.956268491
4	50	50	32	sigmoid	5	0.029435885	115814.9032	0.951698902
3	100	100	32	relu	63	0.15198944	1296456.594	-1.543632711
2	50	100	32	relu	21	0.132876128	920901.242	-0.934404561
1	50	100	32	tanh	5	0.230183627	1334937.292	0.049447972

Table 9: Summary of 'dynamic' LSTM optimal parameters and evaluation metrics

Then, in Figure:[8] the learning curves for the newly trained model are showcased. The same percentage for the new training (80%) and new validation (20%) set is used. In Figure:[9] the actual and forecasted values are plotted for different forecasting windows.



(a) 5 years (b) 5 years (c) 5 years

(d) 4 years (e) 4 years (f) 4 years

(g) 3 years (h) 3 years (i) 3 years

(j) 2 years (k) 2 years (l) 2 years

(m) 1 year (n) 1 year (o) 1 year

Figure 8: Training Loss — Validation Loss —
d_LSTM Learning Curves across different window sizes and years of data

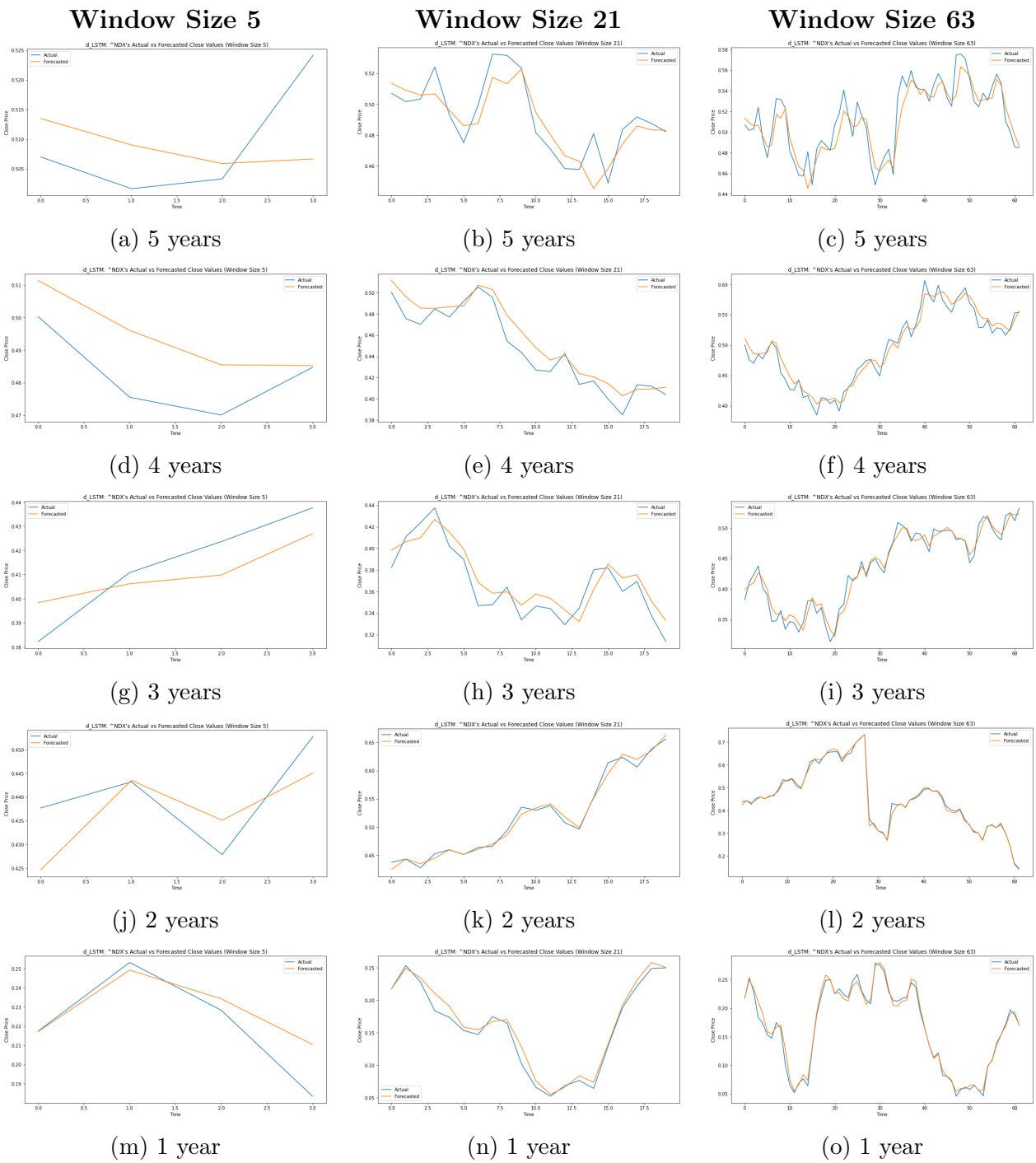


Figure 9: d_LSTM Forecasted vs Actual values across different window sizes and years of data

	Forecast 5	Forecast 21	Forecast 63	Years
MAE	0.008473419	0.009726391	0.010145734	
MSE	0.000101895	0.000153987	0.00017639	5
R^2	-0.27874388	0.72023383	0.837830333	
MAE	0.011889557	0.010455076	0.010167351	
MSE	0.000195482	0.00016163	0.000140723	4
R^2	-0.492066223	0.882967926	0.960502937	
MAE	0.011368268	0.010698482	0.008083391	
MSE	0.000175551	0.000155137	9.41E-05	3
R^2	0.454984233	0.807342679	0.981160482	
MAE	0.007074037	0.006273923	0.006599141	
MSE	7.05E-05	6.50E-05	0.000103211	2
R^2	0.131914994	0.98735358	0.994173312	
MAE	0.009278979	0.008155589	0.005885019	
MSE	0.000194588	0.000118767	6.08E-05	1
R^2	0.68998256	0.972846837	0.987841684	

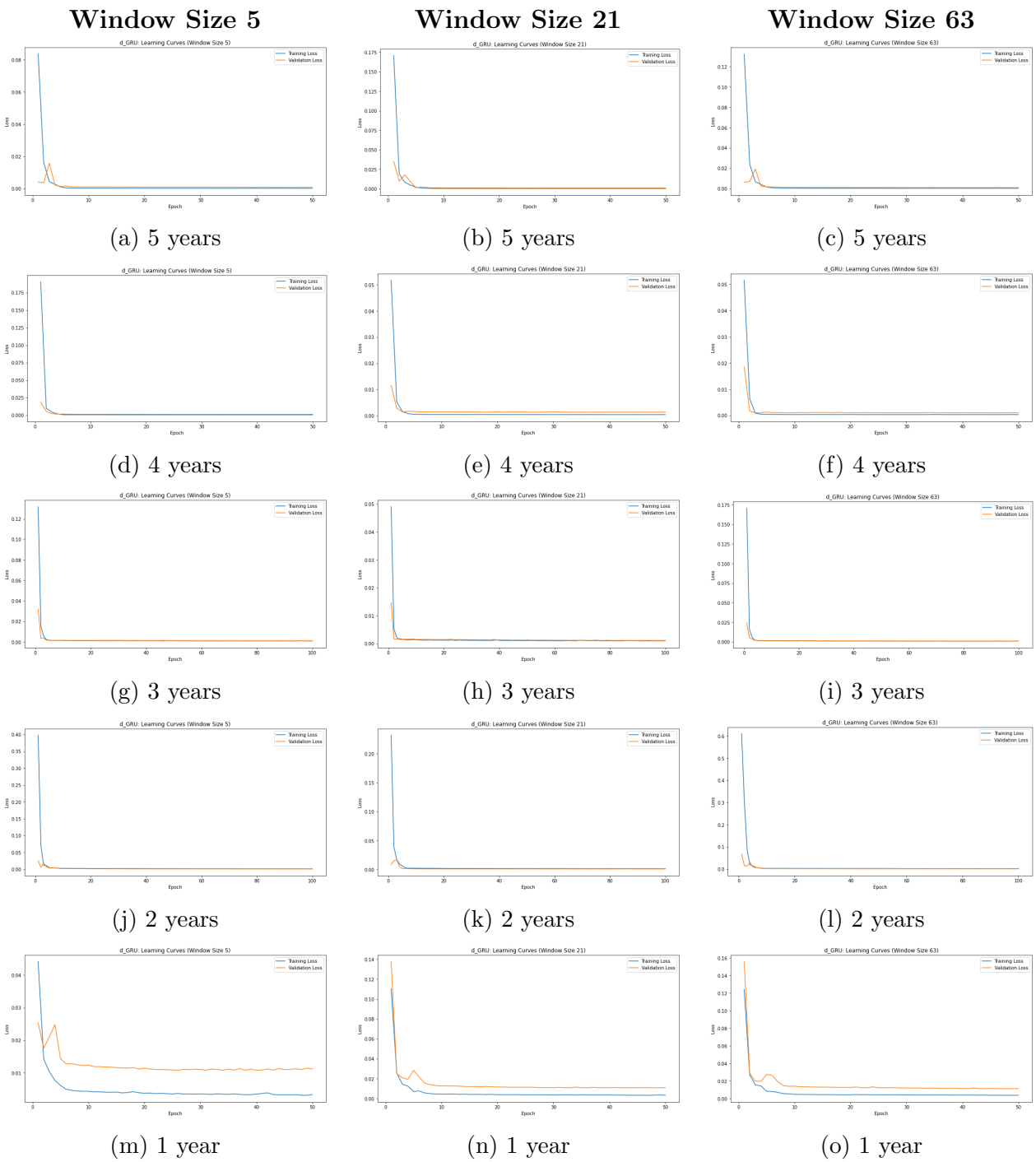
Table 10: d_LSTM evaluation metrics for different years of data

d_GRU

Similarly, the predictions of the s_GRU are fed back into the training dataset and the model is re-trained. In Table[11] the new optimal hyperparameters and the corresponding evaluation metrics are displayed. In Figures 10 and 11 the learning curves and forecasts are plotted in the familiar method. Finally, in Table 12 the evaluation metrics for each forecasting period across all subsets of data are presented.

Years	Units	Epochs	Batch Size	Activation	Window Size	MAE	MSE	R^2
5	50	50	64	tanh	21	0.027505624	131706.2395	0.946976527
4	50	50	32	relu	21	0.032328362	141937.9446	0.940285267
3	100	100	64	relu	63	0.129712514	978596.6835	-0.919995
2	50	100	32	relu	21	0.036429609	75228.47331	0.838064917
1	100	50	32	relu	5	0.063854401	90779.13842	0.133032372

Table 11: Summary of 'dynamic' GRU optimal parameters and evaluation metrics



(a) 5 years (b) 5 years (c) 5 years

(d) 4 years (e) 4 years (f) 4 years

(g) 3 years (h) 3 years (i) 3 years

(j) 2 years (k) 2 years (l) 2 years

(m) 1 year (n) 1 year (o) 1 year

Figure 10: Training Loss — Validation Loss —
d_GRU Learning Curves across different window sizes and years of data

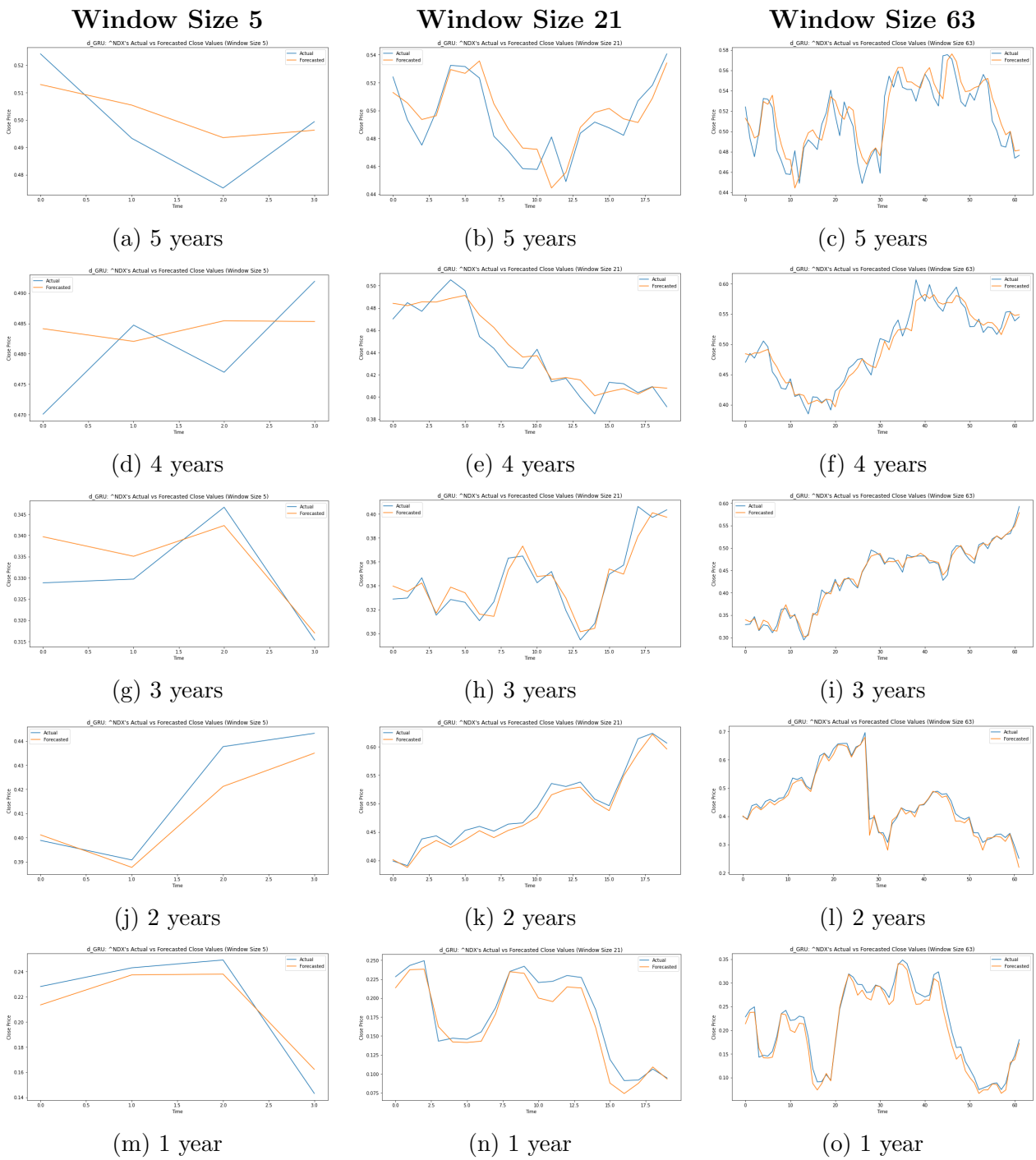


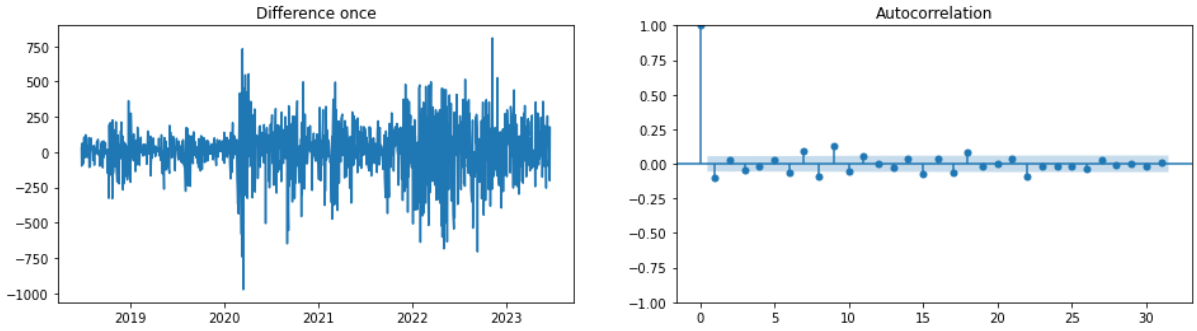
Figure 11: d_GRU Forecasted vs Actual values across different window sizes and years of data

	Forecast 5	Forecast 21	Forecast 63	Years
MAE	0.011201081	0.012240458	0.012363079	
MSE	0.000154928	0.000208807	0.000225634	5
R^2	0.49535959	0.695510928	0.800814771	
MAE	0.004828084	0.006851695	0.009343121	
MSE	3.29E-05	7.22E-05	0.000145339	4
R^2	0.765665025	0.961229865	0.949876772	
MAE	0.005551777	0.007655396	0.006270436	
MSE	4.21E-05	8.24E-05	5.79E-05	3
R^2	0.658814624	0.912705008	0.989169209	
MAE	0.007516857	0.009759131	0.010731343	
MSE	8.85E-05	0.000135722	0.000205648	2
R^2	0.833955093	0.96938868	0.982596617	
MAE	0.012656286	0.012345256	0.012769385	
MSE	0.000184581	0.000225338	0.000246593	1
R^2	0.898879555	0.927903663	0.964923977	

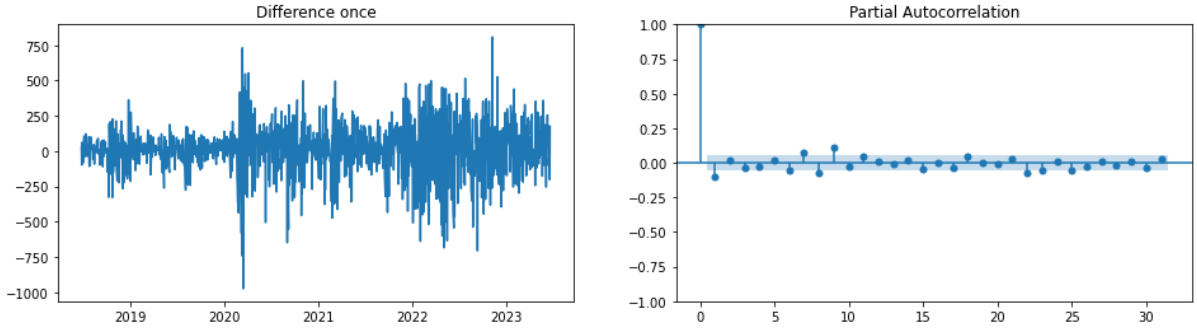
Table 12: d_GRU Evaluation metrics for different years of forecast

4.3 Part C: ARIMA(p,d,q)

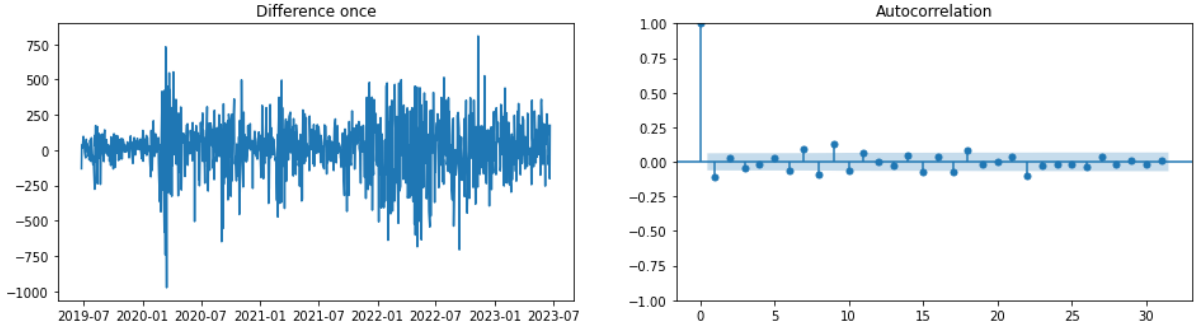
In the final part the results of an ARIMA (p,d,q) will be displayed for the different subsets of data collected and the forecasting periods of 5, 21 and 63 trading days. The actual plots of the closing prices for the index across 5 to 1 years are included in the Appendix (Figure:[16]). First, an ADF test is conducted to identify whether out subsets are stationary or not. The results are summarized in Table: [13] with the ADF and p-values before and after differencing. Secondly, the ACF and PACF for each subset are displayed together with the data after applying differencing technique to ensure stationarity (Figure: [12]). Next, the model is trained using the optimal lag factors found from the ACF/PACF analysis. The corresponding AIC values are also included in Table: [14]. Then the two approaches of evaluation and visuilization are presented. Initially, the ARIMA model is trained across all available observations for each data subset and produces forecasts across the whole length of the dataset. Figure:[17] showcases the predictions of the corresponding ARIMA model. Then, the forecasting periods of 5,21 and 63 trading days are used. In Figure: [14] the forecasted values are plotted against the actual closing values for each prediction period and across different subsets of data. Following, the tables with the corresponding evaluation metrics are showcased; Table: [15] for full length of predictions and Table: [16] for each corresponding forecasting window size.



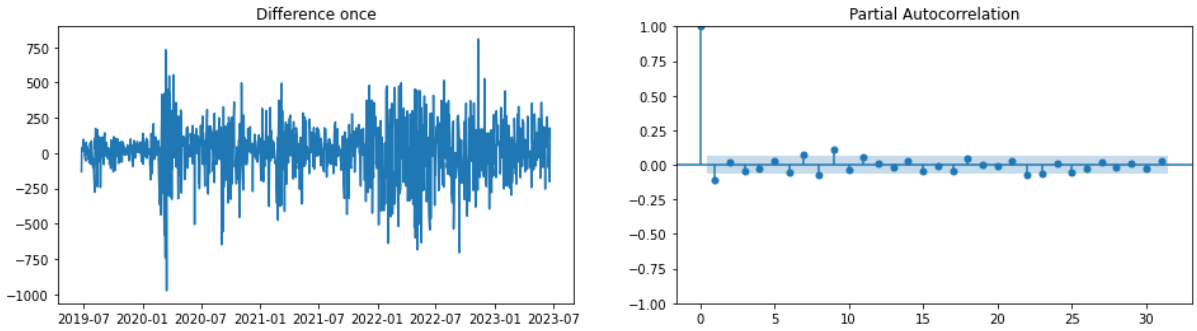
(a) ACF - 5 years



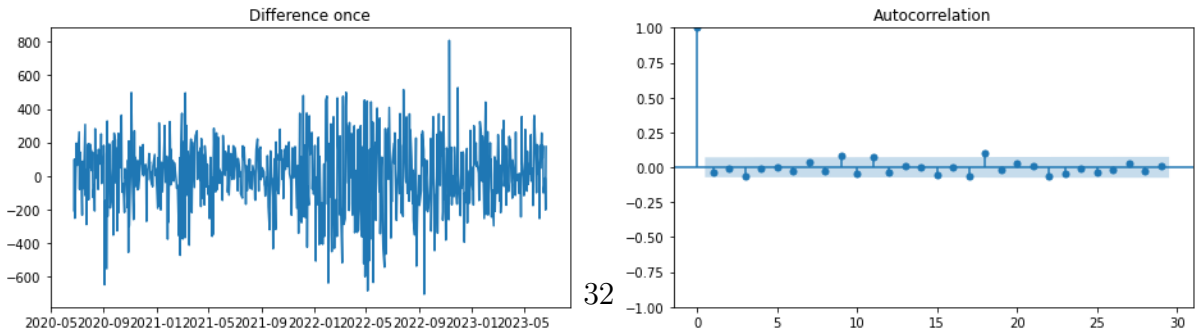
(b) PACF - 5 years



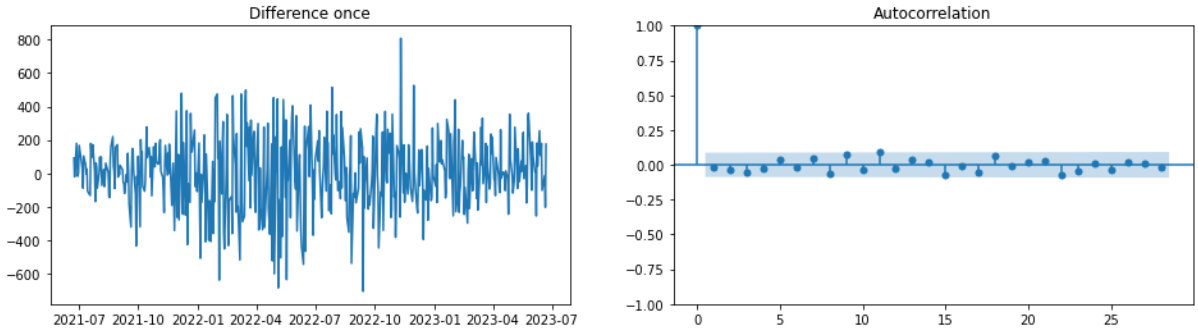
(c) ACF - 4 years



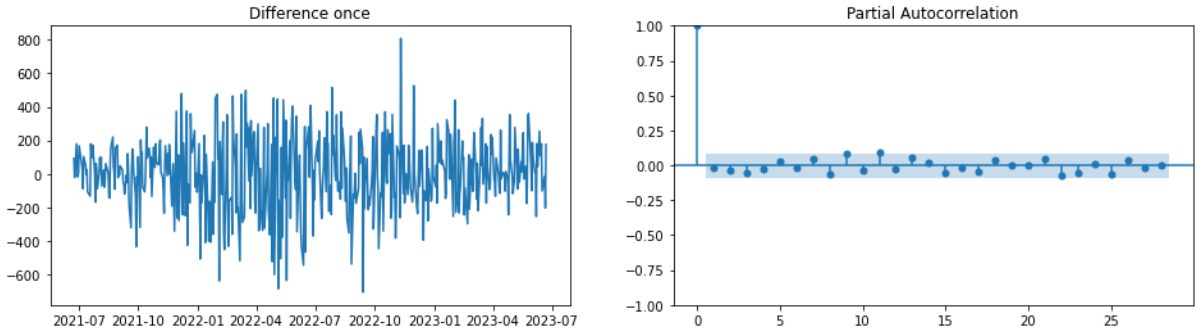
(d) PACF - 4 years



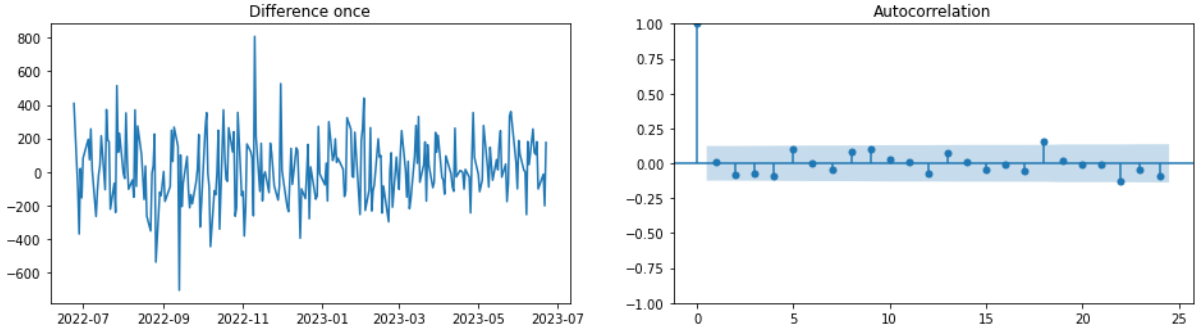
(e) ACF - 3 years



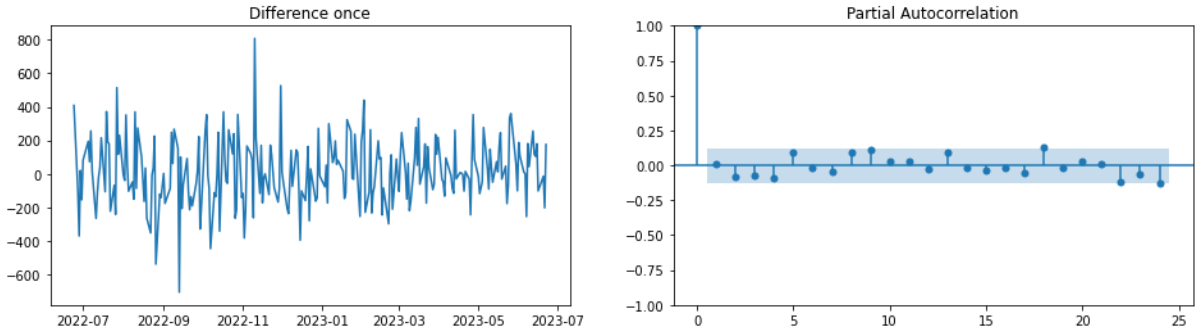
(g) ACF - 2 years



(h) PACF - 2 years

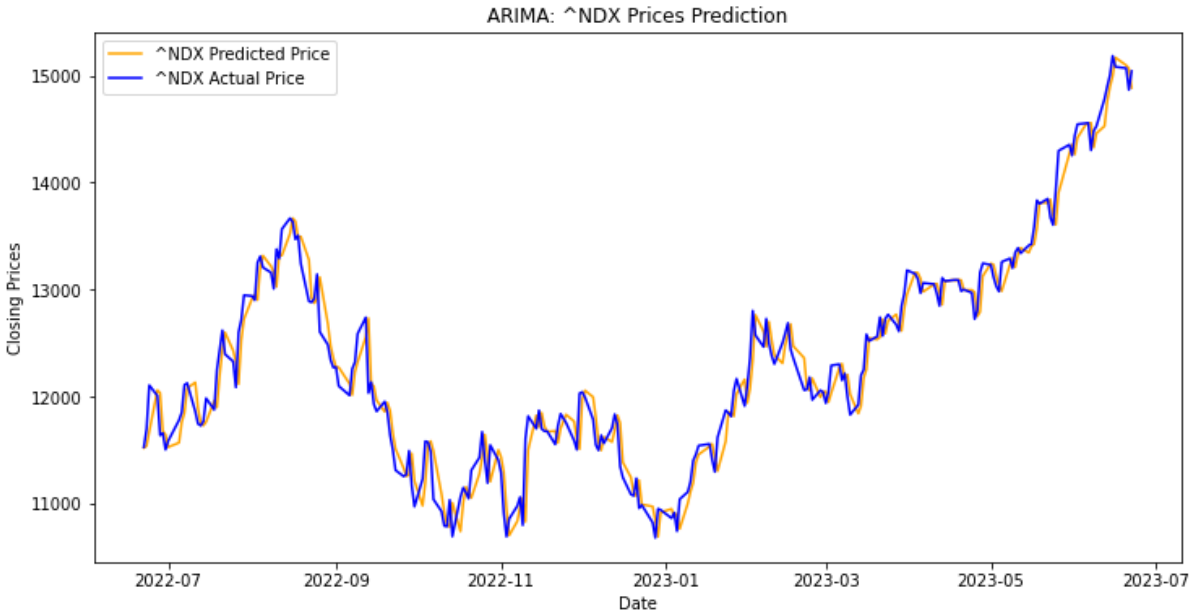


(i) ACF - 1 year

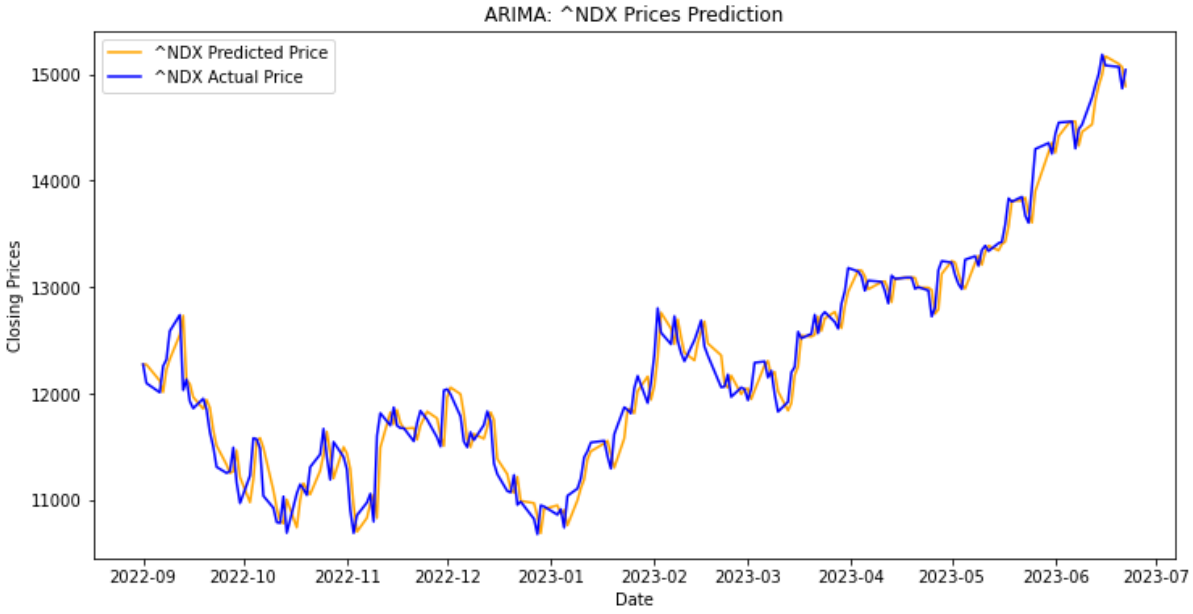


(j) PACF - 1 year

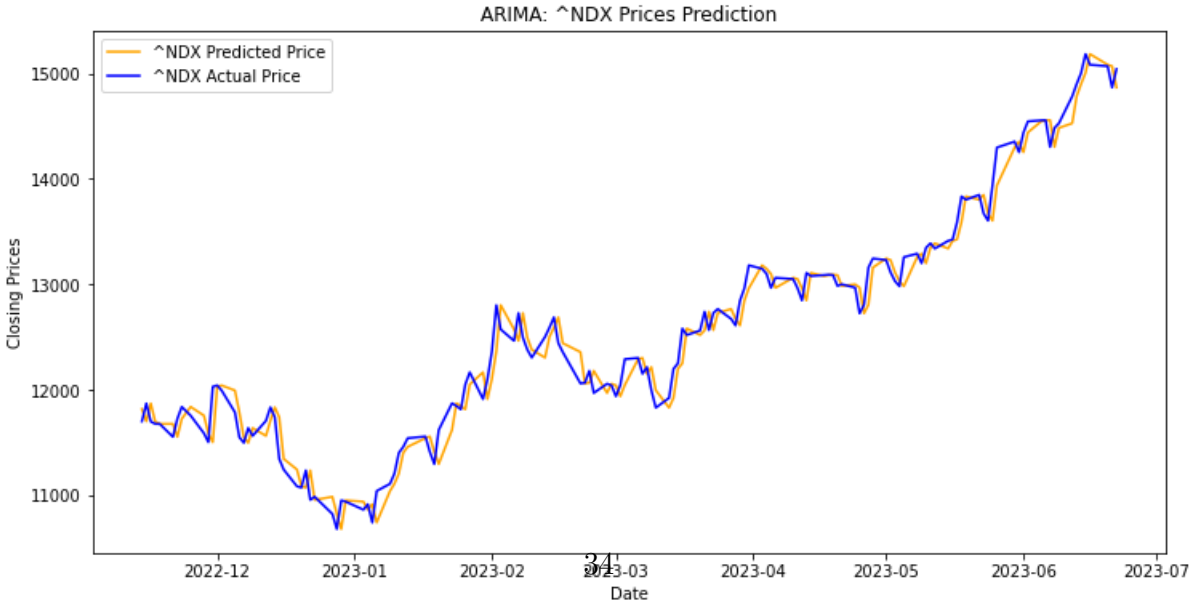
Figure 12: ACF and PACF for different years (continued)



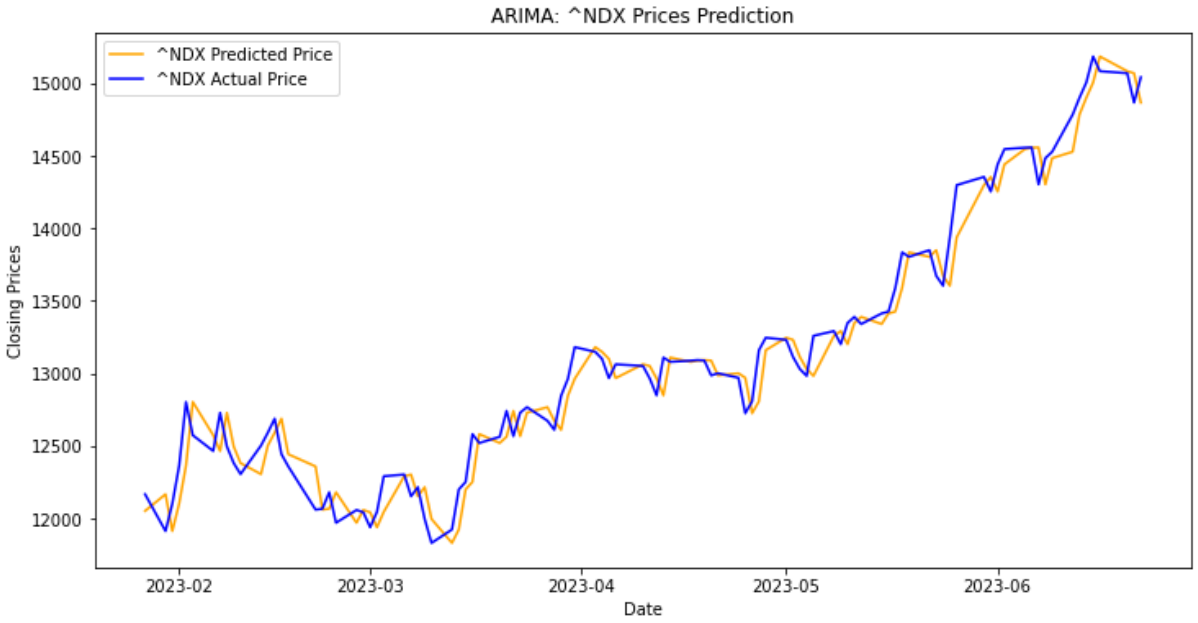
(a) 5 years



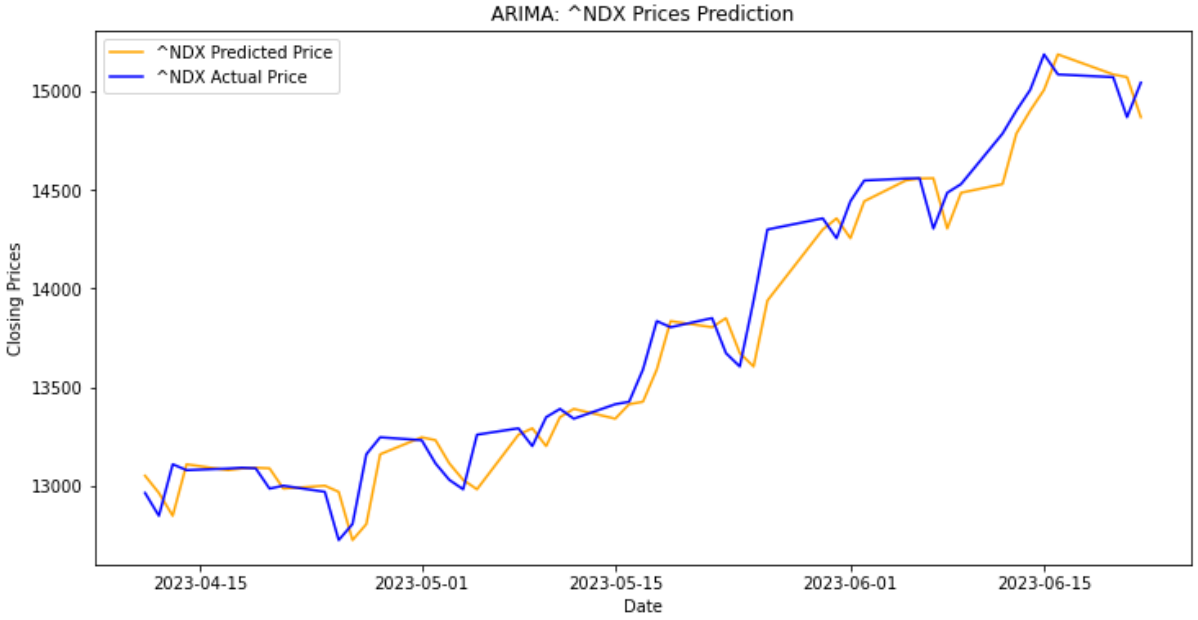
(b) 4 years



(c) 3 years

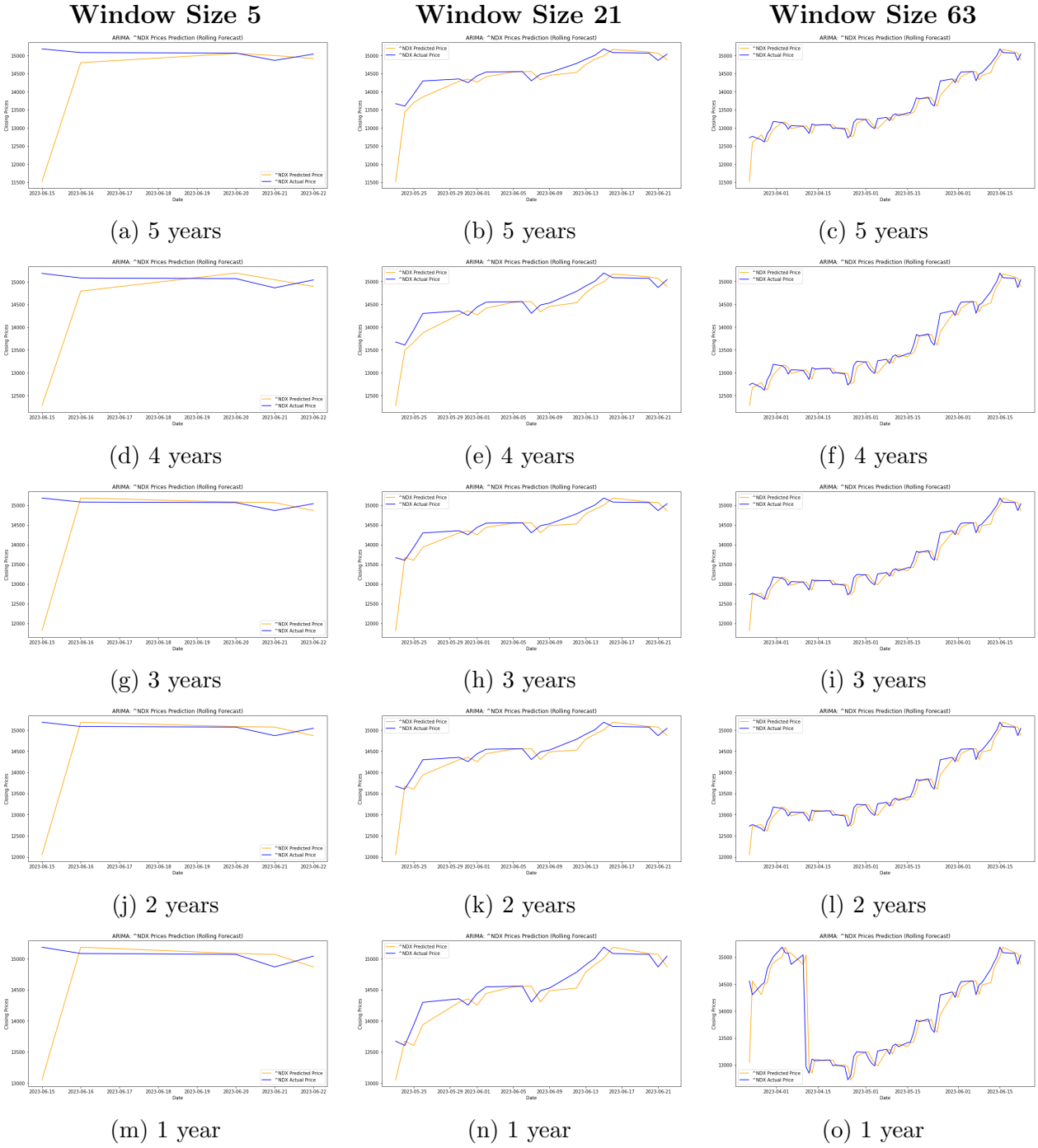


(d) 2 years



(e) 1 year

(continued) ARIMA forecast across subsets for full length of predictions.



(a) 5 years (b) 5 years (c) 5 years

(d) 4 years (e) 4 years (f) 4 years

(g) 3 years (h) 3 years (i) 3 years

(j) 2 years (k) 2 years (l) 2 years

(m) 1 year (n) 1 year (o) 1 year

Actual Price — Forecasted Price —

Figure 14: ARIMA Forecasted vs Actual values across different window sizes and years of data

Year	ADF Statistic	p-value	ADF Statistic (After)	p-value (After)
5	-0.974828075	0.762334125	-9.935879179	2.74E-17
4	-1.460382881	0.553010483	-9.908986475	3.21E-17
3	-2.063830042	0.259305143	-28.51162894	0
2	-1.36769629	0.597661248	-22.62423986	0
1	-0.426328706	0.905609148	-15.69997338	1.40E-28

Table 13: Stationarity test output before and after differencing

Years	p	d	q	AIC	Obs.
5	1	1	1	13218.762	1257
4	1	1	1	10768.679	1007
3	0	1	0	8174.813	755
2	0	1	0	5492.439	503
1	0	1	0	2688.386	251

Table 14: ARIMA Model Selection Results

Years	MAE	MSE	R^2
5	157.168377208	39376.099738430	0.961817068
4	153.732857519	37672.860752932	0.967809542
3	133.479321334	27769.558196223	0.976235354
2	130.495958385	26433.097971151	0.965957463
1	116.216643689	22983.152134129	0.958831766

Table 15: ARIMA performance metrics for full length of observations.

In general, LSTM/GRU models are expected to perform better for longer forecasting periods rather than shorter ones. This is because LSTM/GRU models are designed to capture and model long-term dependencies in sequential data. Therefore, for longer forecasting periods, where long-term dependencies play a significant role, LSTM/GRU models have the potential to outperform simpler models. For shorter forecasting periods, the patterns and dependencies in the data are typically simpler and can be captured effectively by simpler models, such as ARIMA. However, the performance of LSTM/GRU models can also depend on various factors, such as the amount and quality of training data, the complexity of the underlying data patterns, and the specific implementation and hyperparameter settings of the model. The point of interest this paper is exploring is whether these predictions can be incorporated back into the model, providing it with more opportunities to train and thus allowing it to increase its accuracy when compared to the typical-'static' version of the model. In the next section the results are discussed in more detail.

	Forecast 5	Forecast 21	Forecast 63	Years
MAE	842.3331581	243.8539363	135.8870695	
MSE	2708733.846	250861.0744	45899.48845	5
$\mathbf{R^2}$	-253.127055	-0.319567998	0.918905877	
MAE	729.0654584	207.7232669	122.7896001	
MSE	1725746.743	123153.1621	25719.94234	4
$\mathbf{R^2}$	-160.9055111	0.352195346	0.954558618	
MAE	772.2742188	224.2328869	125.7978671	
MSE	2285753.19	192983.7467	34373.55507	3
$\mathbf{R^2}$	-213.4440021	-0.015124315	0.939269621	
MAE	725.3800781	213.0676153	122.0761099	
MSE	1980825.414	154166.677	28456.07017	2
$\mathbf{R^2}$	-184.8363936	0.189059467	0.949724493	
MAE	525.4300781	165.4604725	175.4744234	
MSE	927438.8268	47413.88894	127908.6338	1
$\mathbf{R^2}$	-86.01013506	0.750595621	0.799745679	

Table 16: ARIMA forecast performance metrics for each prediction window.

5 Discussion

In this section, the results will be reviewed and the outputs are further discussed. The first models to visit are the 'static' s_LSTM / s_GRU, followed by the 'dynamic' d_LSTM / d_GRU and finally, the ARIMA model.

5.1 Part A: 'static' LSTM / GRU

s_LSTM

Starting from Table:[5] we notice that MAE and MSE decrease as years of data and therefore number of observations increase.For 1 year of data MAE/MSE values drop from 0.079/195314.57 to 0.019/71736.19 respectively for 5 years or 1259 observations. On the contrary, R^2 increases from -0.5367 to 0.9526. These results are expected as the model has more observations to train and therefore more opportunities to learn the true relations amongst the features. Mainly because of dimensionality issues. This suggests that the model benefits from having a larger historical dataset for training, as it can capture more patterns and improve accuracy. When the evaluation metrics were first introduced, it was mentioned that lower MAE/MSE and higher R^2 was favourable. This is exactly what is observed here.

Moving to Table: [6] it is noticeable that model's performance varies for different forecast horizons and subsets of years. In general, MAE/MSE has a tendency to decrease as the forecasting horizon increases. This suggests that the model's performance improves when predicting further into the future. This aligns with the notion that longer forecasting periods provide more opportunities for the model to capture underlying patterns and trends, resulting in improved accuracy. R^2 show a similar pattern to the MAE and MSE. The R^2 values generally increase as the forecasting horizon increases. This means that the model's ability to explain the variance in the data improves when forecasting further into the future. It is worth noting that some exceptions or anomalies do appear in the given results. Specifically, for subsets 5 and 3 years, MAE/MSE seems to follow a different pattern. This contradicts the general expectation that a longer forecasting period would result in lower forecasting errors. In summary,however, the results suggest that the LSTM model performs better for longer forecasting periods compared to shorter ones, consistently with the expectations.

s_GRU

Advancing to s_GRU and Table:[7], a similar trend, for decreasing MAE/MSE and increasing R^2, is present as the number of observations increase. For one year of data the model produces MAE/MSE of 0.3234/2247885.94 that drops to 0.02585/127876.98 for 5 years of data. On the other hand, R^2 increases from -0.9339 to 0.9218. As already

discussed , these models are designed very similarly and therefore, the same reasoning applies to the s_GRU model regarding performance expectations.

Looking at Table:[8] some variations are noticeable. Overall, MAE/MSE tend to decrease and R^2 increase, as forecasting period increases. R^2 consistently increases as periods increase across all subsets of data. For MAE/MSE some inconsistencies are present, for example year 4 and year 1. For year 4 there is an indication that our model is overfitting, Figure: [6], while year 1 simply because the available data are so low could lead to these inconsistencies. In general, as the forecasting period increases, the model tends to exhibit improved accuracy (lower MAE), precision (lower MSE), and fit (higher R^2), indicating better performance in predicting further into the future.

To summarise, both models seem to perform better for longer forecasting periods, apart from some inconsistencies which will be discussed shortly. Therefore, according to the purpose these models were designed for, their performance is to be expected. On the other hand, one might expect to see lower errors when making short term predictions. That is because forecasting errors can accumulate over time, especially if the model's predictions deviate from the actual values. In longer forecasting periods, errors may have more time to compound, potentially leading to larger discrepancies between the predicted and actual values. From MAE/MSE formula it is apparent that the sum across all observations is taken, thus for larger number of observations the sum should increase. This can result in higher MAE and MSE values for longer horizons compared to shorter horizons, making the performance appear poorer for longer forecasting periods. This is not what is observed here and a number of reasons could potentially explain why is the case.

One would be the data availability. How many historical data are available plays a crucial role in these models accuracy. If the model has access to more historical data for training, it can leverage that information to make better predictions for longer periods. In contrast, for shorter forecasting periods, the model has less historical context to rely on, which can limit its ability to accurately capture the underlying patterns and make precise forecasts. A second reason could be the complexity of patterns in the data. Longer forecasting periods may allow the model to capture and learn complex patterns and relationships in the data more effectively. With more time steps available for training, the model has a better opportunity to recognize and incorporate long-term trends, seasonality, and other patterns that contribute to improved predictions. In contrast, shorter forecasting periods provide limited data for the model to learn these complex patterns, resulting in poorer performance. Thirdly, short-term forecasting tends to be more challenging compared to long-term forecasting. This is because short-term fluctuations and noise in the data can have a greater impact on predictions within a shorter time frame. The inherent volatility and unpredictability of short-term trends can make it more difficult for the model to accurately capture and forecast them, leading to poorer

evaluation metrics. This last reason brings us to the inconsistencies observed. Other reasons such as specific characteristics of the dataset that could have been influenced by external factors may have contributed to the fluctuation from the general pattern. For example, a regulatory change or change in tax rules may have disturbed the market for a specific time period but enough to cause noise.

5.2 Part B: 'dynamic' LSTM / GRU

Looking at the new proposed model, first the results for each model will be discussed followed by a comparison with the 'static' versions.

d_LSTM

Table:[9] summarizes the performance metrics of the 'dynamic' d_LSTM. In general, MAE tends to decrease when available data increase. MSE follows the same pattern apart for 3 years of available data, where it increases before decreasing again. Something similar is observed for R^2. For 1 year of data, poor performance is observed with 0.0495, as expected because of the low amount of observations for the model to train on and the over-fitting indication observed in Figure: [8]. Unexpectedly, the performance decreases for 2 and 3 years worth of data, before rising to high levels of performance, 0.9517 and 0.9663 for 4 and 5 years data respectively. This means that, similar to the s_LSTM model, higher number of observations help increase the performance of our model.

When forecasts are made,the performance of the d_LSTM model improves as the number of years of data increases. Looking at Table:[10], it is evident that MAE/MSE tend to decrease while R^2 is increasing. This suggests that the model becomes more accurate and capable of capturing the underlying patterns and trends in the data as more historical information is available.

d_GRU

From Table: [11] is, again, evident that the model increases in accuracy as available data increase. Apart from year 3 that a spike in MAE/MSE and a dip in R^2 is observed, values of MAE decrease from 0.0649 for year 1 to 0.0275 in year 5. R^2 keeps increasing from 0.133 in year 1 to 0.947 in year 5. On the other hand, MSE values seem to follow an inconsistent pattern for years 3,4 and 5. .

Turning to Table: [12] R^2 scores show a positive trend, with higher values for longer forecasting periods. On the other hand,both the MAE and MSE tend to increase, indicating higher errors in predictions for longer forecasting periods which is not following the familiar pattern. This could be explained by the increase of MSE and the spike in MAE of the trained model, indicating deteriorating prediction accuracy. Similar to d_LSTM an over-fitting indication is present here also (Figure: [10]). This would mean

that our model is overly complex and highly tailored to the specific training examples. As a result, it may not generalize well to new data points, leading to increased prediction errors and higher MSE. On the other hand, if the model is too simple or lacks the necessary complexity to capture the underlying patterns in the data, it may struggle to make accurate predictions as more data is added. This could also result in larger prediction errors and higher MSE. Another reason why this is observed is because of a higher level of noise that is introduced to the model. That would be a clear indication that the predicted values added back to the initial training set are adding more noise to the available dataset. In other words, the quality of the predictions is not very high (low MAE/MSE).

5.3 Part C: ARIMA(p,d,q)

In this section the results of the ARIMA models will be reviewed. The reader should remember that the ARIMA model serves as a benchmark to compare the results of the s_LSTM/s_GRU and d_LSTM/d_GRU.

Starting at Table: [13] it becomes clear that the initial datasets are not stationary. The p-values are much higher than the 0.05 threshold, while the ADF statistic is relatively close to zero. Therefore, the null is supported and the statistical evidence suggest that the time-series has a unit root and thus non-stationary. Applying a Differencing technique once and re-testing provide us with p-values of very close to zero and negative ADF statistic relative far from zero. This provides sufficient statistical evidence to reject the null hypothesis and therefore conclude that the time series has root different from one, thus stationary. Applying ACF/PACF to the now stationary dataset reveal the order of the ARIMA model for each subset of data. Table: [14] provides a summary. For 4 and 5 years of data an ARIMA (1,1,1) is preferred, while for all other datasets an ARIMA(0,1,0) are the optimal lagged observations to be included in the model.

Looking at the overall model evaluation metrics in Table: [15], one should observe that MAE/MSE keep increasing from 116.22/22983.15 for 1 year of data, to 157.17/39376.1 for 5 years of data. R^2 seems to increase in absolute terms but overall remains in the same level ranging from 0.9583 to 0.9762 across all years of data with an absolute difference of $|\Delta| = 0.0179$. These results are to be expected as MAE/MSE are the average of the sum of absolute differences and average of squared errors between the predicted and actual values. With higher number of observations these sums are expected to increase for a larger dataset.

When tha ARIMA model is used to forecast across the three different windows, the results are available in Table: [16]. R^2 clearly has an increasing trend for increasing forecasted days. Moreover, MAE/MSE drop for longer forecasting periods. In a typical ARIMA model the reverse would be expected. Meaning that the predictions should be-

come worst every time step into the future. With the Recursive Forecasting approach, the contrary is observed. Performance increases for longer periods. In the case of longer forecasting windows, such as the 63-day window, the model has more iterations and therefore more opportunities to update and adjust its predictions based on the accumulated historical data. As a result, the model can benefit from learning and adapting to the underlying patterns and trends in the data over a longer period. On the other hand, for shorter forecasting windows, the model has fewer iterations and less historical data to inform its predictions. This limited historical context may result in the model being less able to capture complex patterns or changes in the data, leading to relatively higher prediction errors.

6 Conclusion

Now that the results are presented and have been reviews for each model, its time to make a comparison between the results of each model. First, the LSTM and GRU models are compared for the 'static' and 'dynamic' variants. Secondly, the 'static' and 'dynamic' version are compared for each model. Finally, all models are compared with the ARIMA model.

Focussing first on the LSTM versus GRU models, it is observed that the LSTM outperforms consistently the GRU model, especially when small amount of data is available. Both MAE/MSE show lower values for the LSTM models; both 'static' and 'dynamic' versions, across most options. Some variations,where GRU performs better than the LSTM model can be observed but these can be explained by the specific characteristics of the dataset, for the most part. In any case, the general trend is that LSTM outperforms the GRU models. This can be further supported by the values in Tables: [5, 7]. For the 'dynamic' versions, the picture changes. GRU seem to outperform the LSTM model especially when it is well trained, meaning it has a big pool of data from which to train. On the other hand, LSTM seems to provide better metrics for 1 and 2 years of data. This notion can be further supported by the model evaluation metrics in Tables: [9,11].

Looking at 'static' vs 'dynamic' for the LSTM model. The d_LSTM produces better metrics across the board. Only for 5 years worth of data, when the s_LSTM is well trained, there is no improvement in the models' metrics. Looking at the model evaluation metrics the results are more blurry (Tables: [5,9]). D_LSTM outperforms s_LSTM for years 1,2 and 5 but does a worst job when trained for years 3 and 4. This can be attributed to the data specific characteristics or a different architecture could be used. Turning to the GRU 's' versus 'd' variants, the 'dynamic' version does a much better job in forecasting across windows. Apart from windows 21 and 63 for the 5 years data, all other metrics d_GRU significantly increases in performance. Tables:[7,11] further support this claim.

Finally, ARIMA is compared with the other 4 variant models. First observation is that the Recursive ARIMA performs poorly for short forecasting windows. It is a model that cannot rely heavily on its model training and it needs time to start iterating predicted values into the model before it can start performing at the desired level. LST/GRU models spend much more time training and exploring different combinations of data so they can recognise patterns faster when compared to the ARIMA. For the 63-day forecasting window, the results are at the same level of accuracy as the rest of the models. Still, the 'dynamic' versions seem to do a better job at predicting especially when few amounts of data are initially available. Past the 4 year mark, all models seem to perform at an acceptable level, especially for longer predictions. Its worth noting that the ARIMA model is well trained across all available data. What could be viewed as a strength point for this model is that for all years of data the model metrics are above 0.9588 for the R^2. In the Appendix the diagnostics for the ARIMA models are included (Figure: [17]). Across the board, all residuals follow a white noise patterns and the their distribution is well-behaved.

Many studies have examined different approaches in the attempt to find the optimal forecasting model. In some papers ARIMA does a better job in forecasting stock prices (**Yamak 2019**) while in others the RNN seem to outperform simpler models (**Siami 2018**). This can be attributed to the different characteristics of the datasets, the parameters used and also the features used to train the models. In this approach, two innovative features where engineered and tested. It is true that when it comes to computational resources, ARIMA - a simpler model - is much more efficient. On the other hand, that sacrifices forecasting performance, especially in the short run. It is worth mentioning again that not a simple/typical ARIMA model was used. To produce meaningful comparisons between ARIMA and RNNs, a recursive element was incorporated. This way the ARIMA would be in the position to distinguish between long and short term influences of the data. On the same note, RNN's do seem to perform better, at least from the ARIMA model. On the other hand, the complexity of the training and architectural design can be a challenge to tackle. For each specific dataset the hyper-parameters need to be adjusted and different architectures need to be tested via trial and error. In this paper, a single layer architecture was preferred due to computational and other restrains. It would be interesting to further explore other attributes and architectures of these models or different datasets altogether. Further research could be conducted for start-up firms that historical data are not abundant. This approach could offer a potential approach to proxy historical data.

On a final note, for the 'dynamic' versions the results are encouraging. The aim was to investigate whether adding the predictions back to the RNN models and retrained them based on the new extended training set, would help increase their forecasting capabilities

across short (5days), medium (21days) and longer (63days) period of time. Results indicate that indeed this is the case. Such results were anticipated since these model rely heavily on the availability of data. This relation becomes more severe when there are dimensionality issues, meaning a lot of features for the model to train upon. This study has shown that when data are scarce, these RNN models can produce their own estimations and use them to extend their own training set.

7 Reference

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics, 2014*.

Adebiyi, A. A., Ayo, C. K., Adebiyi, M., & Otokiti, S. O. (2012). Stock price prediction using neural network with hybridized market indicators. *Journal of Emerging Trends in Computing and Information Sciences, 3*(1).

Baba, N., & Kozaki, M. (1992, June). An intelligent forecasting system of stock price using neural networks. In [Proceedings 1992] IJCNN International Joint Conference on Neural Networks (Vol. 1, pp. 371-377). IEEE.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one, 12*(7), e0180944.

Bratu, M. (2012). Forecasts accuracy comparisons for VAR, AR, and VARMA models. *Actual problems of the economy, (3)*, 340-349.

Cryer, J. D. (1986). *Time series analysis* (Vol. 286). Boston: Duxbury Press.

Dase, R. K., & Pawar, D. D. (2010). Application of Artificial Neural Network for stock market predictions: A review of literature. *International Journal of Machine Intelligence, 2*(2), 14-17.

Devadoss, A. V., & Ligori, T. A. A. (2013). Forecasting of stock prices using multi layer perceptron. *International Journal of Computing Algorithm, 2*(1), 440-449.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015, June). Deep learning for event-driven stock prediction. In Twenty-fourth international joint conference on artificial intelligence.

Gao, Y., Wang, R., & Zhou, E. (2021). Stock prediction based on optimized LSTM and GRU models. *Scientific Programming, 2021*, 1-8.

Hendtlass, T., & Ali, M. (Eds.). (2003). Developments in Applied Artificial In-

telligence: 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2002, Cairns, Australia, June 17-20, 2002. Proceedings (Vol. 2358). Springer.

Ho, S. L., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. *Computers & Industrial Engineering, 35*(1-2), 213-216.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

IBM. (2023). Artificial Intelligence. Retrieved May 25, 2023, from `https://www.ibm.com/topics/artificial-intelligence`.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. arXiv preprint arXiv:1706.10059.

Köppen, M. (2000, September). The curse of dimensionality. In 5th Online World Conference on Soft Computing in Industrial Applications (WSC5) (Vol. 1, pp. 4-8).

Mondal, P., Shit, L., & Goswami, S. (2014). Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices. *International Journal of Computer Science, Engineering and Applications, 4*(2), 13.

Rumelhart, D. E. (1986). Learning internal representations by error propagation, in parallel distributed processing. Explorations in the Microstructure of Cognition, 318-362.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1643-1647). IEEE.

Sethia, A., & Raut, P. (2019). Application of LSTM, GRU and ICA for stock price prediction. In *Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2018, Volume 2* (pp. 479-487). Springer Singapore.

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science, 131*, 895-903.

Si, W., Li, J., Ding, P., & Rao, R. (2017, December). A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In 2017 10th International symposium on computational intelligence and design (ISCID) (Vol. 2, pp. 431-436). IEEE.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018, December). A comparison of ARIMA and LSTM in forecasting time series. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1394-1401). IEEE.

Spector, L. (2006). Evolution of artificial intelligence. *Artificial Intelligence, 170*(18), 1251-1253.

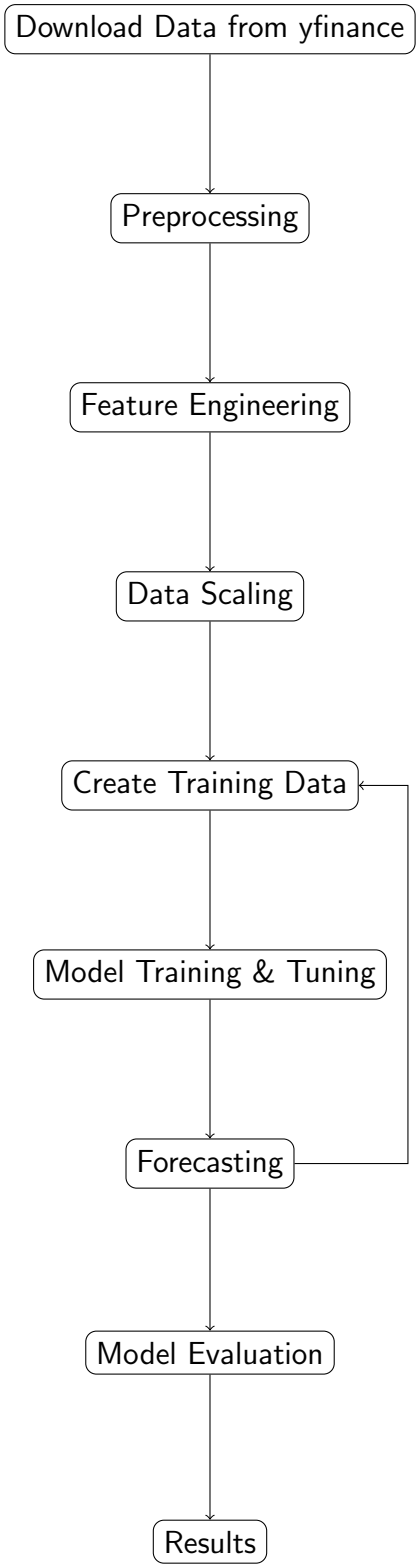
Sunny, M. A. I., Maswood, M. M. S., & Alharbi, A. G. (2020, October). Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES) (pp. 87-92). IEEE.

Vrbka, J., & Rowland, Z. (2017). Stock price development forecasting using neural networks. In *SHS Web of Conferences (Vol. 39, p. 01032)*. EDP Sciences.

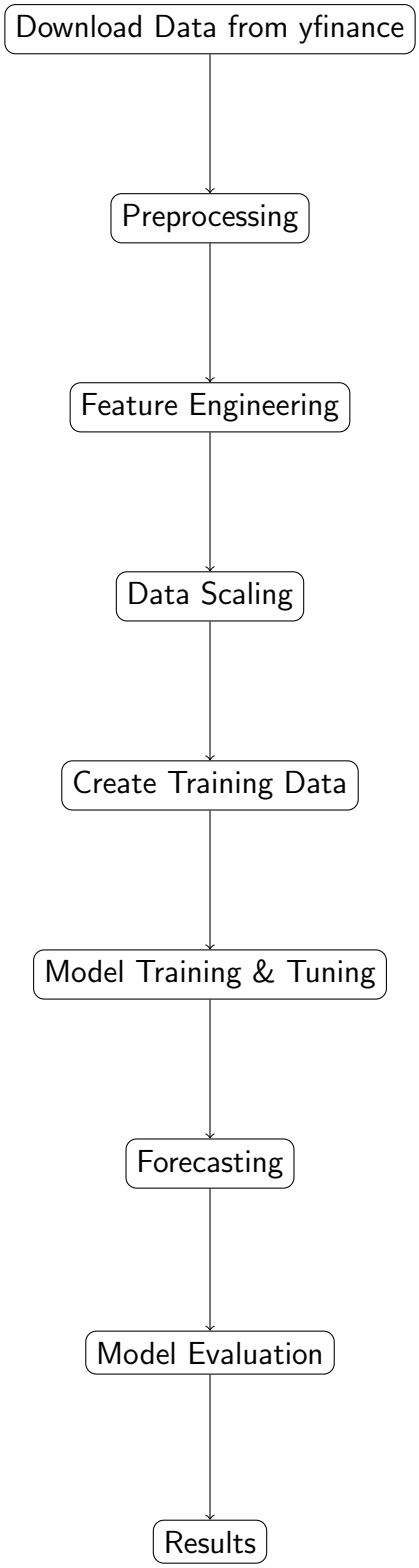
Yamak, P. T., Yujian, L., & Gadosey, P. K. (2019, December). A comparison between ARIMA, LSTM, and GRU for time series forecasting. In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence (pp. 49-55).

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50*, 159-175.

8 Appendix



Dynamic Version



Static Version

Figure 15: Flowchart of the Processes

	Open	High	Low	Medium	Range	Close
count	1259.000000	1259.000000	1259.000000	1259.000000	1259.000000	1259.000000
mean	10973.835512	11064.430032	10877.666519	10971.048275	186.763512	10977.115936
std	2930.468604	2951.479362	2904.311415	2927.359231	121.581981	2929.788396
min	5969.080078	6075.169922	5895.120117	5985.145020	20.750000	5899.350098
25%	7815.895020	7856.199951	7767.239990	7812.314941	93.814941	7826.770020
50%	11465.209961	11577.690430	11331.259766	11451.015137	161.690430	11501.650391
75%	13285.000000	13403.689941	13193.479980	13292.484863	240.135254	13306.615234
max	16644.769531	16764.859375	16523.830078	16574.844727	923.659180	16573.339844

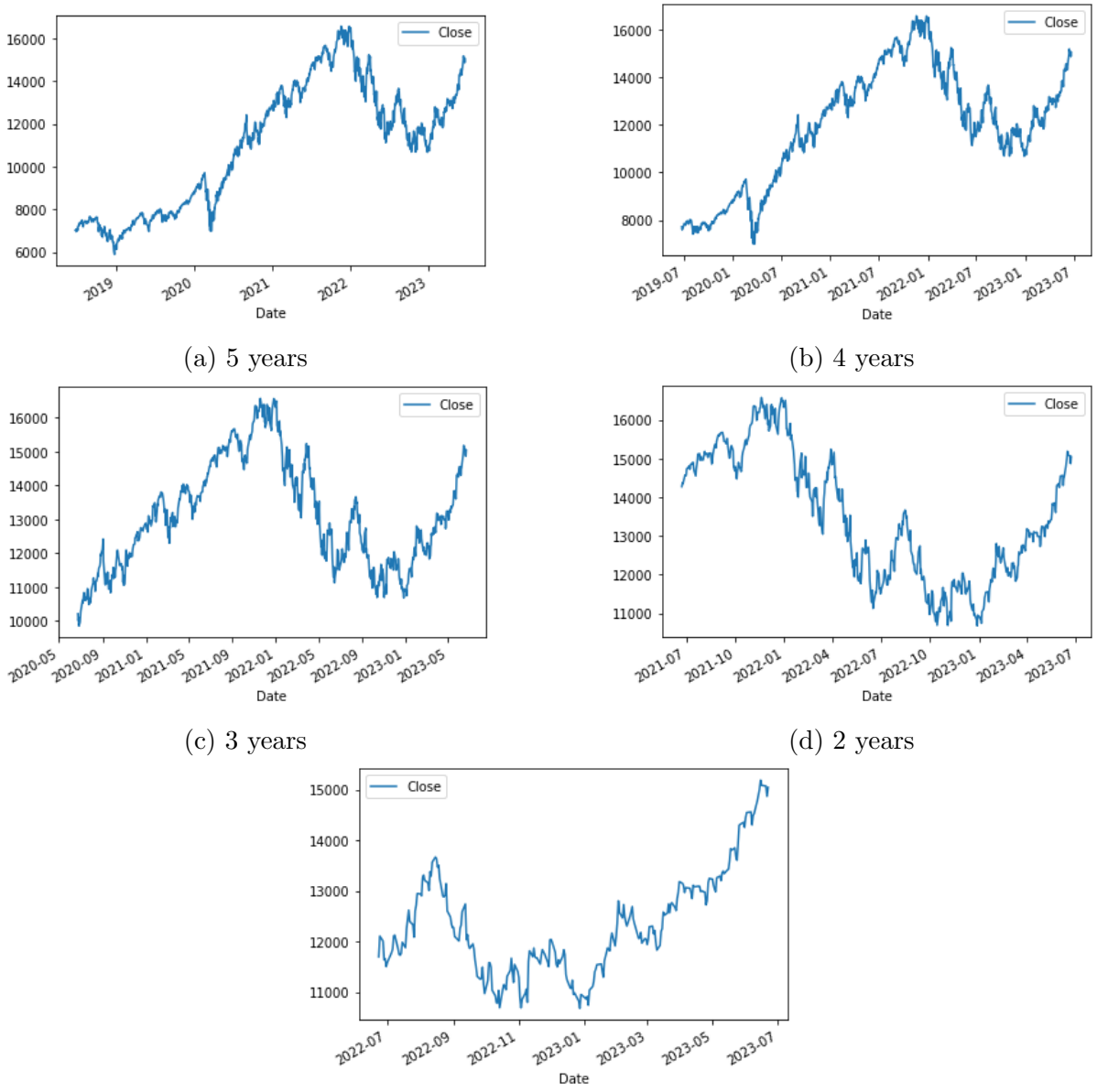
Table 17: Description of 5 yr Data

	Open	High	Low	Medium	Range	Close
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000
mean	11913.986492	12014.808928	11807.152961	11910.980944	207.655966	11917.815109
std	2495.148864	2507.019940	2476.470728	2491.023251	123.794726	2493.033985
min	6952.709961	7145.290039	6771.910156	6958.600098	20.750000	6994.290039
25%	9970.209961	10010.679688	9923.330078	9977.905273	120.669922	9975.860352
50%	12131.070312	12258.370117	12012.980469	12128.250000	187.719727	12125.690430
75%	13730.730469	13827.099609	13662.910156	13733.344727	269.709961	13758.509766
max	16644.769531	16764.859375	16523.830078	16574.844727	923.659180	16573.339844

Table 18: Description of 4 yr Data

	Open	High	Low	Medium	Range	Close
count	754.000000	754.000000	754.000000	754.000000	754.000000	754.000000
mean	13077.687528	13188.775928	12957.274761	13073.025345	231.501167	13080.051437
std	1591.364280	1585.286224	1591.368907	1587.238825	117.755968	1589.061047
min	9517.139648	9809.410156	9489.580078	9649.495117	51.429688	9663.780273
25%	11782.044678	11943.764648	11686.044678	11815.502563	150.117920	11827.924805
50%	12950.379883	13054.569824	12836.040039	12948.039795	206.114746	12938.430176
75%	14244.050049	14401.477539	14022.057129	14217.774780	289.217773	14254.027344
max	16644.769531	16764.859375	16523.830078	16574.844727	923.659180	16573.339844

Table 19: Description of 3 yr Data



(a) 5 years

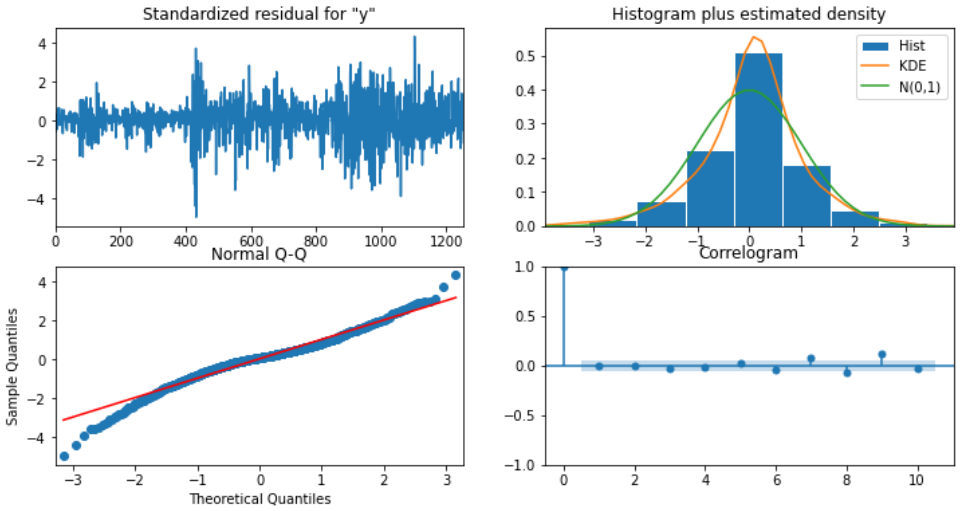
(b) 4 years

(c) 3 years

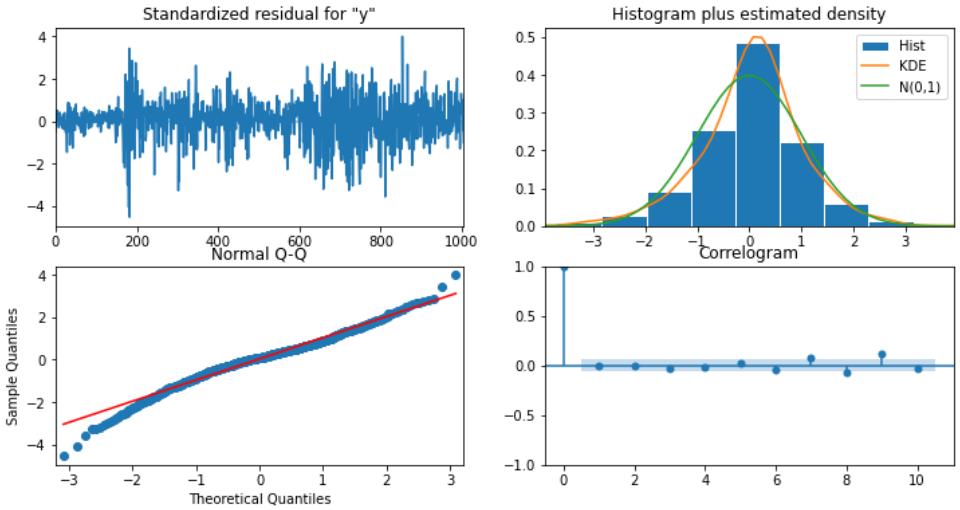
(d) 2 years

(e) 1 year

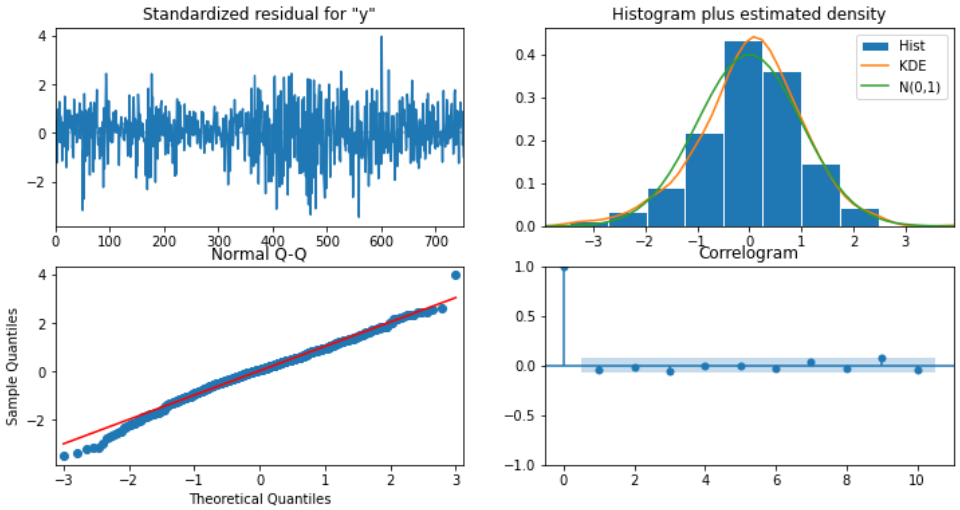
Figure 16: NDX's closing prices across different years worth of data.



(a) 5-year dataset

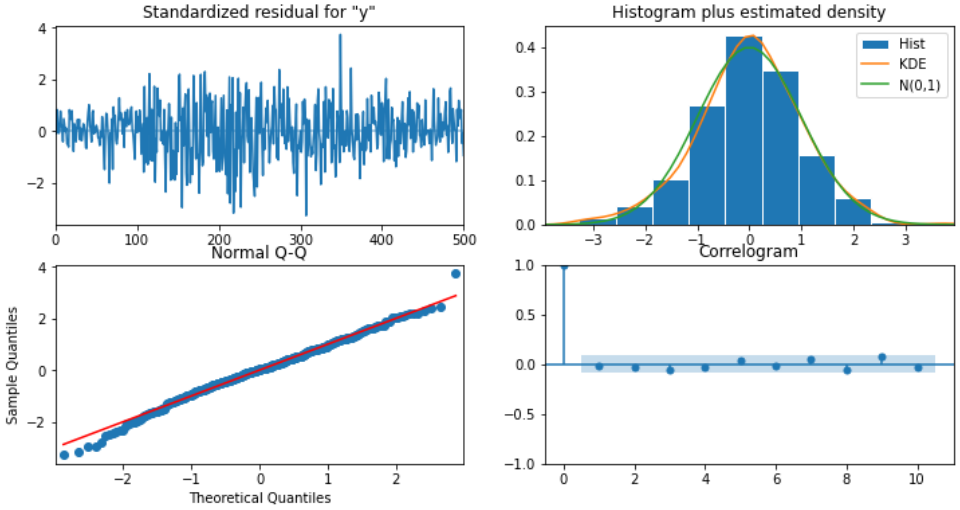


(b) 4-year dataset

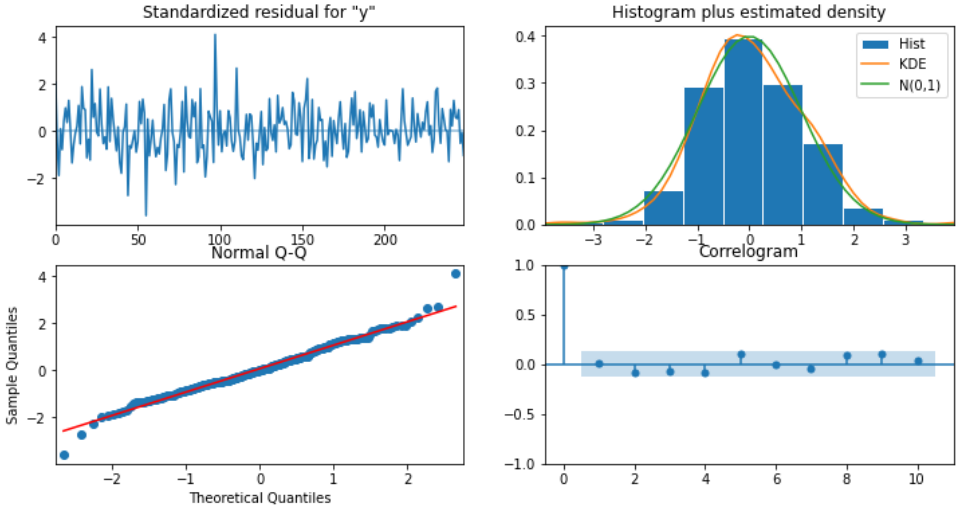


(c) 3-year dataset

Figure 17: ARIMA full model diagnostics for different year datasets.



(d) 2-year dataset



(e) 1-year dataset

(continued) ARIMA full model diagnostics for different year datasets.

	Open	High	Low	Medium	Range	Close
count	502.000000	502.000000	502.000000	502.000000	502.000000	502.000000
mean	13472.980798	13593.884019	13340.274203	13461.186227	253.609816	13475.830383
std	1123.503593	1113.399422	1123.913989	1120.859097	96.959303	1122.245745
min	9757.900391	10086.799805	9711.349609	9934.683594	51.429688	9938.584961
25%	12636.285645	12778.869629	12555.767090	12616.745605	192.525513	12643.834961
50%	13609.979492	13730.219727	13461.430176	13602.893555	242.579102	13598.430176
75%	14381.699707	14489.949707	14235.985352	14354.123535	310.682617	14384.559570
max	15680.000000	15763.429688	15550.559570	15650.299805	637.429688	15658.129883

Table 20: Description of 2 yr Data

	Open	High	Low	Medium	Range	Close
count	251.000000	251.000000	251.000000	251.000000	251.000000	251.000000
mean	13939.048007	14061.308115	13835.041013	13937.628176	226.267102	13943.777411
std	1554.656770	1543.369431	1557.346999	1548.436942	105.469618	1554.438056
min	11160.139648	11360.950195	11150.040039	11237.694336	103.680176	11138.209961
25%	12661.700195	12777.400391	12577.549805	12642.404785	154.930176	12654.400391
50%	13759.049805	13825.099609	13695.660156	13759.720703	202.149414	13759.110352
75%	15238.194824	15376.157227	15092.184570	15237.848145	280.259766	15254.957520
max	16644.769531	16764.859375	16523.830078	16574.844727	923.659180	16573.339844

Table 21: Description of 1 yr Data

	Open	High	Low	Close	Adj Close	Volume
count	1257.000000	1257.000000	1257.000000	1257.000000	1257.000000	1257.000000
mean	10998.205004	11089.087931	10901.819727	11001.586105	11001.586105	3970203532.219571
std	2929.648769	2950.503308	2903.724474	2929.082842	2929.082842	1515093734.924904
min	5969.080078	6075.169922	5895.120117	5899.350098	5899.350098	958950000.000000
25%	7825.640137	7874.990234	7778.229980	7841.299805	7841.299805	2421470000.000000
50%	11472.629883	11606.730469	11360.730469	11504.519531	11504.519531	4196130000.000000
75%	13329.860352	13427.419922	13205.589844	13329.519531	13329.519531	4920210000.000000
max	16644.769531	16764.859375	16523.830078	16573.339844	16573.339844	11621190000.000000

Table 22: 5 years descriptive statistics

	Open	High	Low	Close	Adj Close	Volume
count	1007.000000	1007.000000	1007.000000	1007.000000	1007.000000	1007.000000
mean	11944.207300	12045.374358	11837.042014	11948.122022	11948.122022	4393002482.621649
std	2484.825186	2496.327254	2466.705131	2482.885997	2482.885997	1389712661.891824
min	6952.709961	7145.290039	6771.910156	6994.290039	6994.290039	1014530000.000000
25%	10006.250000	10115.979980	9942.675293	10010.344727	10010.344727	3776340000.000000
50%	12158.879883	12290.330078	12026.980469	12166.599609	12166.599609	4459000000.000000
75%	13766.594727	13861.864746	13678.799805	13786.744629	13786.744629	5115860000.000000
max	16644.769531	16764.859375	16523.830078	16573.339844	16573.339844	11621190000.000000

Table 23: 4 years descriptive statistics

	Open	High	Low	Close	Adj Close	Volume
count	755.000000	755.000000	755.000000	755.000000	755.000000	755.000000
mean	13126.759770	13238.164981	13006.451019	13129.503847	13129.503847	4905834622.516557
std	1571.699969	1565.326434	1572.118315	1570.374866	1570.374866	1053878465.949215
min	9850.549805	9965.080078	9742.889648	9849.360352	9849.360352	2184080000.000000
25%	11828.520020	11983.584961	11726.130371	11849.089844	11849.089844	4266355000.000000
50%	12970.320312	13078.099609	12861.519531	12969.759766	12969.759766	4710720000.000000
75%	14367.725098	14491.750000	14218.050293	14350.084961	14350.084961	5311340000.000000
max	16644.769531	16764.859375	16523.830078	16573.339844	16573.339844	11621190000.000000

Table 24: 3 years descriptive statistics

	Open	High	Low	Close	Adj Close	Volume
count	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000
mean	13488.054812	13609.282240	13361.698254	13491.788971	13491.788971	4885193359.840954
std	1637.855303	1625.048804	1638.988144	1634.664698	1634.664698	780985881.381714
min	10481.580078	10842.330078	10440.639648	10679.339844	10679.339844	2184080000.000000
25%	12035.729980	12149.469727	11903.990234	12031.839844	12031.839844	4376305000.000000
50%	13253.769531	13394.540039	13103.799805	13267.610352	13267.610352	4773520000.000000
75%	14918.419922	15035.839844	14817.410156	14915.449707	14915.449707	5264400000.000000
max	16644.769531	16764.859375	16523.830078	16573.339844	16573.339844	9468130000.000000

Table 25: 2 years descriptive statistics

	Open	High	Low	Close	Adj Close	Volume
count	251.000000	251.000000	251.000000	251.000000	251.000000	251.000000
mean	12318.818013	12439.738316	12210.323448	12334.130089	12334.130089	4921451314.741035
std	1007.470029	1000.466090	1022.329012	1018.286682	1018.286682	754415158.142163
min	10481.580078	10842.330078	10440.639648	10679.339844	10679.339844	2184080000.000000
25%	11550.040039	11668.540039	11471.754883	11584.109863	11584.109863	4442120000.000000
50%	12110.709961	12257.059570	12016.429688	12134.400391	12134.400391	4808710000.000000
75%	12982.645020	13081.354980	12909.829590	12993.375000	12993.375000	5226980000.000000
max	15272.650391	15284.650391	15073.450195	15185.480469	15185.480469	9468130000.000000

Table 26: 1 year descriptive statistics