**VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**INFORMATION SYSTEM FACULTY**

🙡🕮🙣

Logo

Description automatically generated

**FINAL PROJECT REPORT**

**FORECASTING PRICE TRENDS OF STOCK USING FORECASTING MODELS  
AND MACHINE LEARNING**

SUBJECT: BUSINESS ANALYSIS

| **Instructor** | Ph.D Nguyen Dinh Thuan  Nguyen Minh Nhut  Nguyen Thi Viet Huong |
| --- | --- |
| **Group 5** | |
| Do Dang Kien Nam | 20521627 |
| Bui Thi Thanh Ngan | 20521643 |
| Le Anh Thu | 20521985 |

Ho Chi Minh City, June 2023

**ACKNOWLEDGMENTS**

I want to start by expressing my sincere gratitude to ***Assoc. Prof. Dr. Nguyen Dinh Thuan*** for his unwavering support and guidance throughout the course. Your vast knowledge and experience in the field have been invaluable to our team. I want to take this chance to thank ***Assoc. Prof. Dr. Nguyen Dinh Thuan*** for letting us work on this project. I want to say how grateful I am that our team was able to attend your session and learn from you.

I would also like to extend my sincerest thanks to to ***Mr. Nguyen Minh Nhut*** and ***Ms. Nguyen Viet Huong*** who has taught and helped our team throughout the course for unwavering support and commitment to our team's success. Your tireless efforts in teaching and guiding us through the course have been instrumental in our growth and development. We are grateful for your efforts in helping us navigate the challenging tasks of this project. I am grateful for the opportunity to have worked alongside such dedicated and knowledgeable instructors and I am confident that the skills and knowledge gained from their guidance will serve me well in future endeavors.

Many thanks to ***Assoc. Prof. Dr. Nguyen Dinh Thuan, Mr. Nguyen Minh Nhut*** and ***Ms. Nguyen Viet Huong***. We sincerely appreciate your tireless efforts in leading our team to success and motivating us to keep moving forward. Your tireless efforts have been instrumental in ensuring that our project was a resounding success. We would also like to extend our sincere gratitude to all of our classmates who generously volunteered their time and support to help us bring our project to fruition. Your selfless contributions and unwavering support were instrumental in our success and allowed us to achieve an exceptional outcome. Without your contributions, we would not have been able to achieve such an outstanding result.

Thank you all for your guidance and support.

Sincerely,

Team 5

**Contents**

[**I.**](#_heading=h.gjdgxs) **Introduction** 4

[**II.**](#_heading=h.30j0zll) **Data processing** 4

[**1.**](#_heading=h.1fob9te) **Data description** 4

[**2.**](#_heading=h.3znysh7) **Descriptive statistics** 4

[**III.**](#_heading=h.3dy6vkm) **Modeling** 7

[**1.**](#_heading=h.1t3h5sf) **Algorithm** 7

[**1.1.**](#_heading=h.4d34og8) **Linear Regression** 7

[**1.2.**](#_heading=h.2s8eyo1) **Vector Autoregression (VAR)** 7

[**1.3.**](#_heading=h.17dp8vu) **Bayesian vector autoregression (BVAR)** 7

[**1.4.**](#_heading=h.3rdcrjn) **ARIMA** 8

[**1.5.**](#_heading=h.26in1rg) **Gated Recurrent Units (GRU)** 8

[**1.6.**](#_heading=h.lnxbz9) **RNN - Recurrent neural network** 10

[**1.7.**](#_heading=h.35nkun2) **Long Short Term Memory (LSTM)** 10

[**1.8.**](#_heading=h.1ksv4uv) **SVR - Support Vector Regression** 10

[**1.9.**](#_heading=h.44sinio) **Recurrent Kalman Networks** 13

[**IV.**](#_heading=h.2jxsxqh) **Result** 14

[**1.**](#_heading=h.z337ya) **Performance measure** 14

[**2.**](#_heading=h.3j2qqm3) **Predictting next 30 days** 15

[**V.**](#_heading=h.1y810tw) **Conclusion** 17

# **Introduction**

# **Data processing**

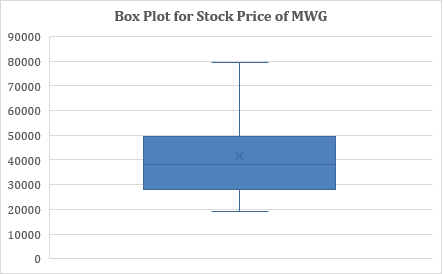
## **Data description**

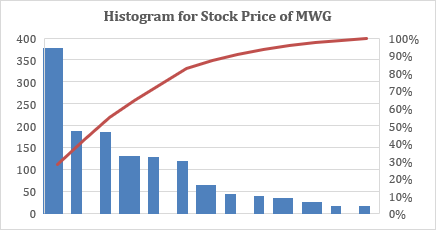
Datasets is collected through stock trends of 3 companies (Mobile World Investment Corp, Vingroup Joint Stock Company, FPT) on Website Investing.com between January 2 , 2017 and June 16, 2023. Every dataset includes 1360 rows and 7 attributes in total. The data contains trading information on high price, low price, open price, close price, trading volume and change. These columns mean respectively the timeline of data representation, closing price of the stock, opening price of the stock, the highest closing price, the lowest price at which a stock trades over the course of a trading day, the number of shares traded in a given period.

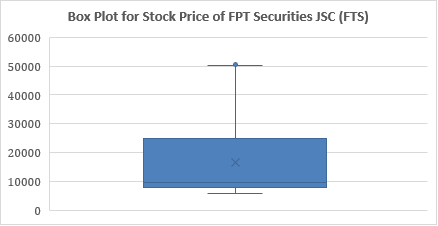
## **Descriptive statistics**

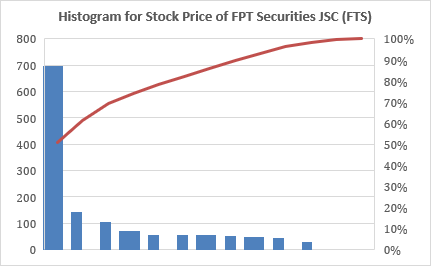
|  | MWG | FTS | VIC |
| --- | --- | --- | --- |
| Count | 1360 | 1360 | 1360 |
| Mean | 41602.58 | 16513.87 | 89532.10 |
| Std | 15199.76 | 11831.89 | 18628.47 |
| Min | 19198.00 | 5788.40 | 50500.00 |
| 25% | 28196.00 | 7726.32 | 78100.00 |
| 50% | 38008.00 | 9413.30 | 94133.00 |
| 75% | 48937.50 | 24685.30 | 103111.00 |
| Max | 79582.00 | 51873.90 | 128000.00 |

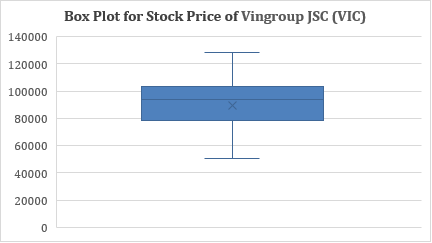
Table 1: Descriptive Statistics

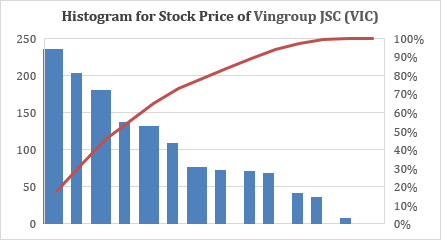












# **Modeling**

## **Algorithm**

### **Linear Regression**

Linear Regression is an algorithm of machine learning, based on supervised learning. This method is frequently used to forecast and determine the relationship between variables that cause and effect. The number of independent variables and the type of relationship between the independent and dependent variables are the primary differences between regression methods. The regression procedure enables you to confidently establish which elements are most important, which can be ignored, and how those factors interact with one another.

Linear regression provides the simplest model form to model the regression function as a linear combination of predictors. [1]

LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

The formula for a simple linear regression is [1]

y = +

Where

* y is the predicted value of dependent variables (y) for any given of independent variables (x)
* is the intercept , the predicted value of y when x is 0
* is the regression coefficient – how much we expect y to change as x increases
* is the independent variable ( the variable we expect is influenced by y)
* is the error estimate, or how much variation there is in our estimate of the regression coefficient.

### **Vector Autoregression (VAR)**

Vector Autoregression (VAR) is a forecasting algorithm that can be used when two or more time series influence each other

That means, the basic requirements in order to use VAR are:

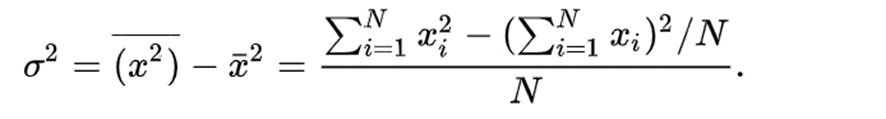
* You need at least two time series (variables)
* The time series should influence each other.

It is considered as an Autoregressive model because, each variable (Time Series) is modeled as a function of the past values, that is the predictors are nothing but the lags (time delayed value) of the series.

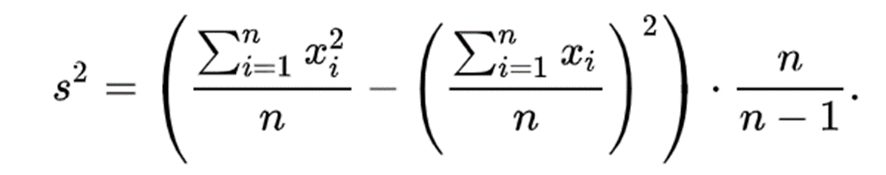
The structure is that each variable is a linear function of past lags of itself and past lags of the other variables. It is considered as an Autoregressive model because, each variable (Time Series) is modeled as a function of the past values, that is the predictors are nothing but the lags (time delayed value) of the series.

Algorithms for calculating variance play a major role in [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics). A key difficulty in the design of good [algorithms](https://en.wikipedia.org/wiki/Algorithm) for this problem is that formulas for the [variance](https://en.wikipedia.org/wiki/Variance) may involve sums of squares, which can lead to [numerical instability](https://en.wikipedia.org/wiki/Numerical_instability) as well as to [arithmetic overflow](https://en.wikipedia.org/wiki/Arithmetic_overflow) when dealing with large values.

A formula for calculating the variance of an entire [population](https://en.wikipedia.org/wiki/Statistical_population) of size N is:



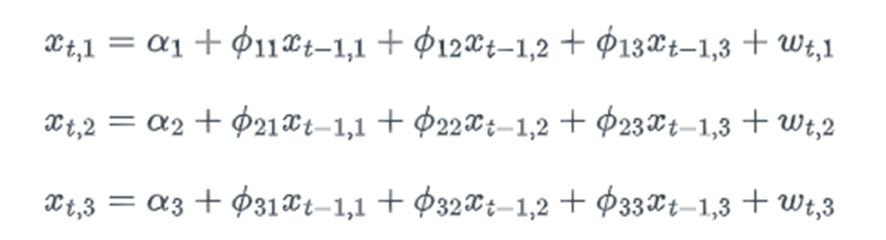
Using [Bessel's correction](https://en.wikipedia.org/wiki/Bessel%27s_correction) to calculate an [unbiased](https://en.wikipedia.org/wiki/Estimator_bias) estimate of the population variance from a finite [sample](https://en.wikipedia.org/wiki/Statistical_sample) of n observations, the formula is:



In the VAR model, each variable is modeled as a linear combination of past values of itself and the past values of other variables in the system. Since you have multiple time series that influence each other, it is modeled as a system of equations with one equation per variable (time series).

As an example suppose that we measure three different time series variables, denoted by x1, x2, x3

The vector autoregressive model of order 1, denoted as VAR(1), is as follows:

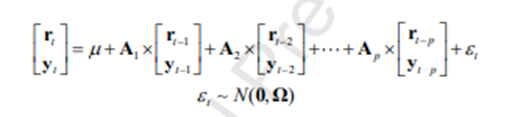


The primary difference of VAR different from other Autoregressive models like AR, ARMA or ARIMA is those models are uni-directional, where, the predictors influence the Y and not vice-versa. Whereas, Vector Auto Regression (VAR) is bi-directional. That is, the variables influence each other.

We are interested in modeling a T × K, multivariate time series Y, where T denotes the number of observations and K the number of variables. One way of estimating relationships between the time series and their lagged values is the vector autoregression process:

where Ai is a K × K coefficient matrix.

The [VAR](https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.var_model.VAR.html#statsmodels.tsa.vector_ar.var_model.VAR) class assumes that the passed time series are stationary. Non-stationary or trending data can often be transformed to be stationary by first-differencing or some other method. For direct analysis of non-stationary time series, a standard stable VAR(p) model is not appropriate



First and foremost, we calculate a class of VAR ( p) models, which consist of the linear relationship linking t r , a N 1 vector of commodity asset returns at time t , and t y , a M 1 vector of predictor s at time t , to the lags of both t r and t y . Specifically, the VAR ( p ) model can be written as:

wher e is a (N x M ) x 1 vector of intercepts, A1, A2,...,An are ( N + M) ( N + M) coefficient matrices, and t is a (N + M) X 1 vector of IID Gaussian residuals. Since the VAR model can generate the optimal strategic portfolio allocation by carefully selecting the lag order s and the predictor, we select the predictors before modeling, and consider five different lag order s : p 1, 2, 4, 6, 12 = to examine the performance of the VAR model in the asset allocation of commodity market. As the lag orders increase, more past information can be considered during modeling .

### **Bayesian vector autoregression (BVAR)**

 Bayesian vector autoregression (BVAR) uses [Bayesian methods](https://en.wikipedia.org/wiki/Bayesian_inference) to estimate a [vector autoregression](https://en.wikipedia.org/wiki/Vector_autoregression) (VAR) model. BVAR differs with standard VAR models in that the model parameters are treated as [random variables](https://en.wikipedia.org/wiki/Random_variable), with [prior probabilities](https://en.wikipedia.org/wiki/Prior_probability), rather than fixed values. Bayesian Vector Autoregression (BVAR) models have the same mathematical form as any other VAR model.

Where

* yt is a K×1vector of endogenous variables in period t
* yt is a vector of m economic variables at time t
* Ai is the cofficient matrix corresponding to the i th lag of yt
* c is a constant deterministic term
* ϵ is an error term with zero mean and variance-covariance Σ

The only difference between usual VAR models and BVAR models is the way parameter estimates are obtained and interpreted.

Bayesian estimators are slightly more complicated and more burdensome in terms of algebra and calculation power. [3]

### **ARIMA**

#### Algorithm

ARIMA is an acronym for “autoregressive integrated moving average.” It’s a model used in statistics and econometrics to measure events that happen over a period of time. The model is used to understand past data or predict future data in a series. It’s used when a metric is recorded in regular intervals, from fractions of a second to daily, weekly or monthly periods. ARIMA is a type of model known as a Box-Jenkins method

An ARIMA model has three component functions: AR (p), the number of lag observations or autoregressive terms in the model; I (d), the difference in the nonseasonal observations; and MA (q), the size of the moving average window. An ARIMA model order is depicted as (p,d,q) with values for the order or number of times the function occurs in running the model. Values of zero are acceptable.

To make a better explanation of ARIMA we can also write it as (AR, I, MA) and by this, we can assume that in the ARIMA, p is AR, d is I and q is MA. here our assumption is right. These parameters can be explained as follows:

* p is the number of autoregressive terms,
* d is the number of nonseasonal differences,
* q is the number of lagged forecast errors in the prediction equation.

For example: ARIMA(1, 1, 2) can also be called a damped-trend linear exponential smoothing where we are applying one time differencing on the time series if it is non-stationary and after that, we are performing autoregression on the series with one lag when the series is stationary by differencing and 2 average moving average order is applied.

How to choose values of p, d and q?

There are various ways to choose the values of parameters of the ARIMA model. Without being confused we can do this using the following steps:

1. Test for stationarity using the augmented dickey fuller test or visualization

2. If the time series is stationary try to fit the ARMA model, and if the time series is non-stationary then seek the value of d.

3. If the data is getting stationary then draw the autocorrelation and partial autocorrelation graph of the data.

4. Draw a partial autocorrelation graph(ACF) of the data. This will help us in finding the value of p because the cut-off point to the PACF is p.

5. Draw an autocorrelation graph(ACF) of the data. This will help us in finding the value of q because the cut-off point to the ACF is q.

Stationary Series is one whose statistical properties such as mean, variance, covariance, and standard deviation do not vary with time, or these stats properties are not a function of time. In other words, stationarity in Time Series also means series without a Trend or Seasonal components.

Types of Stationary Series

* Strict Stationary – Satisfies the mathematical definition of a stationary process. Mean, variance & covariance are not a function of time.
* Seasonal Stationary – Series exhibiting seasonality.
* Trend Stationary – Series exhibiting trend.

#### StatisticalTest (ADF test, KPSS test)

##### Dickey-Fuller Test

Before going into the ADF test, let’s first understand what the Dickey-Fuller test is.

It uses an autoregressive model and optimizes an information criterion across multiple different lag values. A Dickey-Fuller test is a unit root test that tests the null hypothesis that α=1 in the following model equation. alpha is the coefficient of the first lag on Y.

Null Hypothesis (H0): alpha=1

where,

* y(t-1) = lag 1 of time series
* delta Y(t-1) = first difference of the series at the time (t-1)

Fundamentally, it has a similar null hypothesis as the unit root test. That is, the coefficient of Y(t-1) is 1, implying the presence of a unit root. If not rejected, the series is taken to be non-stationary.

The Augmented Dickey-Fuller test evolved based on the above equation and is one of the most common forms of the Unit Root Test.

##### Augmented Dickey-Fuller test

Statistical tests make strong assumptions about your data. They can only be used to inform the degree to which a null hypothesis can be rejected or fail to be rejected. The result must be interpreted for a given problem to be meaningful.

However, they provide a quick check and confirmatory evidence that the time series is stationary or non-stationary.

The Augmented Dickey-Fuller test is a type of statistical test called a unit root test.

In probability theory and statistics, a unit root is a feature of some stochastic processes (such as random walks) that can cause problems in statistical inference involving time series models. In simple terms, the unit root is non-stationary but does not always have a trend component.

ADF test is conducted with the following assumptions:

* Null Hypothesis (HO): Series is non-stationary, or series has a unit root.
* Alternate Hypothesis(HA): Series is stationary, or series has no unit root.

If the null hypothesis is failed to be rejected, this test may provide evidence that the series is non-stationary.

##### Dickey-Fuller Test

Before going into the ADF test, let’s first understand what the Dickey-Fuller test is.

It uses an autoregressive model and optimizes an information criterion across multiple different lag values. A Dickey-Fuller test is a unit root test that tests the null hypothesis that α=1 in the following model equation. alpha is the coefficient of the first lag on Y.

Null Hypothesis (H0): alpha=1

where,

* y(t-1) = lag 1 of time series
* delta Y(t-1) = first difference of the series at the time (t-1)

Fundamentally, it has a similar null hypothesis as the unit root test. That is, the coefficient of Y(t-1) is 1, implying the presence of a unit root. If not rejected, the series is taken to be non-stationary.

The Augmented Dickey-Fuller test evolved based on the above equation and is one of the most common forms of the Unit Root Test.

##### Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test

The KPSS test, short for, Kwiatkowski-Phillips-Schmidt-Shin (KPSS), is a type of Unit root test that tests for the stationarity of a given series around a deterministic trend. In other words, the test is somewhat similar in spirit to the ADF test. A common misconception, however, is that it can be used interchangeably with the [ADF test](https://www.machinelearningplus.com/time-series/kpss-test-for-stationarity/). This can lead to misinterpretations about stationarity, which can easily go undetected, causing more problems down the line.

Further in this article, you will see how to implement the KPSS test in python, how it is different from the ADF test, and when and what all things you need to take care of when implementing a KPSS test.

### **Gated Recurrent Units (GRU)**

These networks are designed to handle the vanishing gradient problem. They have a reset and update gate. These gates determine which information is to be retained for future predictions.

A Gated Recurrent Unit (GRU) is a variant of the RNN architecture, and uses gating mechanisms to control and manage the flow of information between cells in the neural network. GRUs can be considered a relatively new architecture, especially when compared to the widely-adopted LSTM. The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.[4]

A picture containing text, screenshot, diagram, font

Description automatically generated

Figure 7: Overview of GRU

The structure of the GRU allows it to adaptively capture dependencies from large sequences of data without discarding information from earlier parts of the sequence. This solve the vanishing/exploding gradient problem of traditional RNNs.

A picture containing text, diagram, font, screenshot

Description automatically generated

Figure 8: GRU Architecture

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.

A picture containing diagram, line, design

Description automatically generated

the training process for a GRU network is also diagrammatically similar to that of a basic Recurrent Neural Network and differs only in the internal working of each recurrent unit.

The ability of the GRU to hold on to long-term dependencies or memory stems from the computations within the GRU cell to produce the hidden state. While LSTMs have two different states passed between the cells — the cell state and hidden state, which carry the long and short-term memory, respectively — GRUs only have one hidden state transferred between time steps. This hidden state is able to hold both the long-term and short-term dependencies at the same time due to the gating mechanisms and computations that the hidden state and input data go through.

Reset Gate: It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.

Update Gate: It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit.

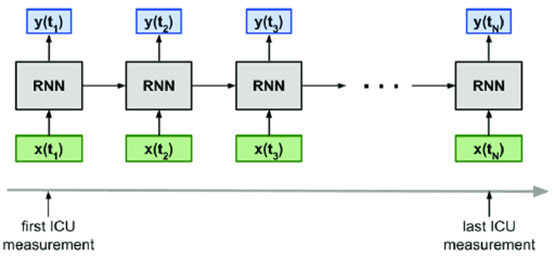
Current Memory Gate( ): It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the future.

Combining the outputs:

[4]

### **RNN - Recurrent neural network**

Recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs.



RNN Architecture

A diagram of a machine

Description automatically generated with low confidence

Variant RNN Architectures

Bidirectional recurrent neural networks (BRNN): These are a variant network architecture of RNNs. While unidirectional RNNs can only drawn from previous inputs to make predictions about the current state, bidirectional RNNs pull in future data to improve the accuracy of it.

Long short-term memory (LSTM): This is a popular RNN architecture, which was introduced by Sepp Hochreiter and Juergen Schmidhuber as a solution to vanishing gradient problem. If the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state.

Gated recurrent units (GRUs): This RNN variant is similar the LSTMs as it also works to address the short-term memory problem of RNN models. Instead of using a “cell state” regulate information, it uses hidden states, and instead of three gates, it has two—a reset gate and an update gate. Similar to the gates within LSTMs, the reset and update gates control how much and A Recurrent Neural Networks is a more flexible model, since it encodes the temporal context in its feedback connections, which are capable of capturing the time varying dynamics of the underlying system RNNs are learning machines that recursively compute new states by applying transfer functions to previous states and inputs. Typical transfer functions are composed by an affine transformation followed by a nonlinear function, which are chosen depending on the nature of the particular problem at hand.

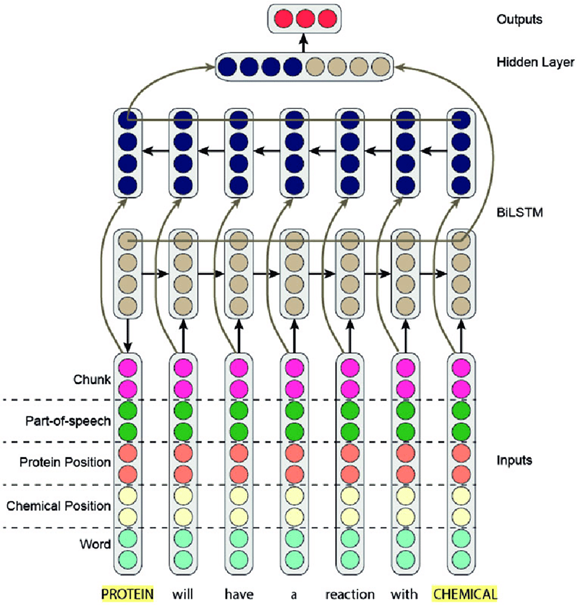
An activation function determines whether a neuron should be activated. The nonlinear functions typically convert the output of a given neuron to a value between 0 and 1 or -1 and 1. [5]

A picture containing text, diagram, line, font

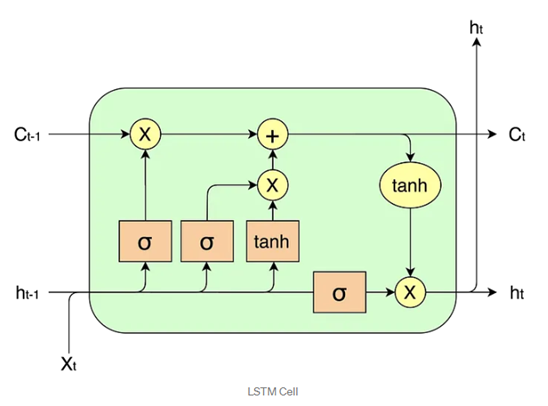
Description automatically generated

### **Long Short Term Memory (LSTM)**

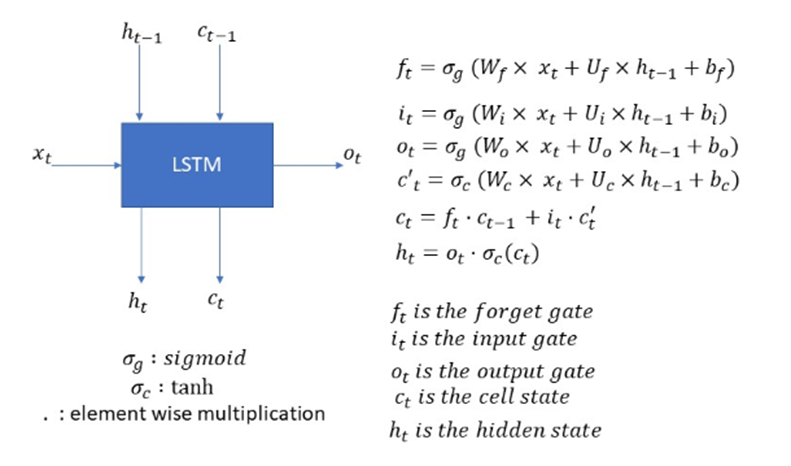
LSTMs were also designed to address the vanishing gradient problem in RNNs. LSTMs use three gates called input, output, and forget gate. Similar to GRU, these gates determine which information to retain.



Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that can learn order dependence in sequence prediction problems. This is a necessary characteristic in complex problem domains such as machine translation, speech recognition, and others.

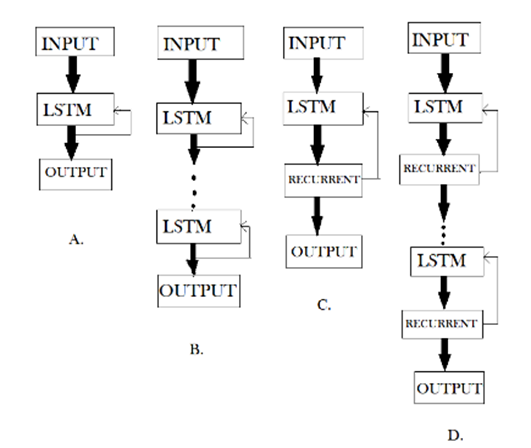


The basic difference between the architectures of RNNs and LSTMs is that the hidden layer of LSTM is a gated unit or gated cell. It consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got the only single neural net layer of tanh, LSTMs comprises of three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where ‘0’ means ‘reject all’ and ‘1’ means ‘include all’.



Variations:

With the increasing popularity of LSTMs, various alterations have been tried on the conventional LSTM architecture to simplify the internal design of cells to make them work in a more efficient way and to reduce the computational complexity. Gers and Schmidhuber introduced peephole connections which allowed gate layers to have knowledge about the cell state at every instant. Some LSTMs also made use of a coupled input and forget gate instead of two separate gates that helped in making both the decisions simultaneously. Another variation was the use of the Gated Recurrent Unit(GRU) which improved the design complexity by reducing the number of gates. It uses a combination of the cell state and hidden state and also an update gate which has forgotten and input gates merged into it.



LSTM(Figure-A), DLSTM(Figure-B), LSTMP(Figure-C) and DLSTMP(Figure-D)

Figure-A represents what a basic LSTM network looks like. Only one layer of LSTM between an input and output layer has been shown here.

Figure-B represents Deep LSTM which includes a number of LSTM layers in between the input and output. The advantage is that the input values fed to the network not only go through several LSTM layers but also propagate through time within one LSTM cell. Hence, parameters are well distributed within multiple layers. This results in a thorough process of inputs in each time step.

Figure-C represents LSTM with the Recurrent Projection layer where the recurrent connections are taken from the projection layer to the LSTM layer input. This architecture was designed to reduce the high learning computational complexity (O(N)) for each time step) of the standard LSTM RNN.

Figure-D represents Deep LSTM with a Recurrent Projection Layer consisting of multiple LSTM layers where each layer has its own projection layer. The increased depth is quite useful in the case where the memory size is too large. Having increased depth prevents overfitting in models as the inputs to the network need to go through many nonlinear functions.

In 1997, Hochreiter and Schmidhuber proposed LSTM, which had achieved surprising performance in the NLP field. LSTM aims at resolving long-term dependence problem based on improved RNN (Annotation) neural network. Keeping information in mind for a long time is an inherent characteristic of LSTM. All RNN models have a chain form of repetitive neural network modules. [6]

Similarly, as a variant of RNN, LSTM also has this chain module structure, but with different repetitive modules and layers. LSTM has three more gates than RNN with only the tanh layer.

A picture containing screenshot, circle, line, design

Description automatically generated

LSTM equations:

Forget Gate:

Input Gate:

Cell Gate:

Output Gate:

Cell State: [7]

Bidirectional long-short term memory (Bi-LSTM) is a technique that allows any neural network to store sequence information both forward and backward. Bi-LSTM allows input flow in both directions, whereas normal LSTM only allows input flow in one direction

A picture containing diagram, text, plan, screenshot

Description automatically generated

Advantages:

LSTM cells have several advantages over simple RNN cells, such as their ability to learn long-term dependencies and capture complex patterns in sequential data. For example, they can predict the next word in a sentence based on the previous words and the context, or generate captions for images based on the visual features and the language model. Additionally, LSTM cells can avoid the vanishing or exploding gradient problem, allowing them to learn from longer sequences without losing or amplifying the information. This means they can be used to translate a long sentence from one language to another without forgetting or distorting the meaning. Furthermore, they can handle noisy or missing data better than simple RNN cells, such as filling in the blanks or correcting errors in a text based on the surrounding words and grammar.

Drawbacks:

LSTMs became popular because they could solve the problem of vanishing gradients. But it turns out, they fail to remove it completely. The problem lies in the fact that the data still has to move from cell to cell for its evaluation. Moreover, the cell has become quite complex now with the additional features (such as forget gates) being brought into the picture.

They require a lot of resources and time to get trained and become ready for real-world applications. In technical terms, they need high memory-bandwidth because of linear layers present in each cell which the system usually fails to provide for. Thus, hardware-wise, LSTMs become quite inefficient.

With the rise of data mining, developers are looking for a model that can remember past information for a longer time than LSTMs. The source of inspiration for such kind of model is the human habit of dividing a given piece of information into small parts for easy remembrance.

LSTMs get affected by different random weight initialization and hence behave quite similar to that of a feed-forward neural net. They prefer small weight initialization instead.

LSTMs are prone to overfitting and it is difficult to apply the dropout algorithm to curb this issue. Dropout is a regularization method where input and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network.

### **SVR - Support Vector Regression**

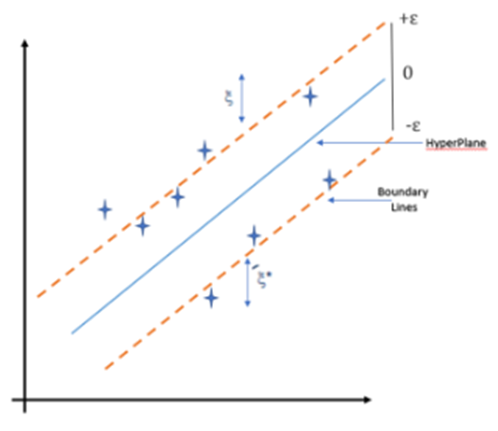
Support Vector Regression as the name suggests is a regression algorithm that supports both linear and non-linear regressions. The goal of SVR is to find a function that approximates the relationship between the input variables and a continuous target variable, while minimizing the prediction error.[8] This method works on the principle of the Support Vector Machine. SVR differs from SVM in the way that SVM is a classifier that is used for predicting discrete categorical labels while SVR is a regressor that is used for predicting continuous ordered variables.

Support Vector Regression is similar to Linear Regression in that the equation of the line is y= wx+b In SVR, this straight line is referred to as hyperplane. The data points on either side of the hyperplane that are closest to the hyperplane are called Support Vectors which is used to plot the boundary line.

SVR can handle non-linear relationships between the input variables and the target variable by using a kernel function to map the data to a higher-dimensional space. This makes it a powerful tool for regression tasks where there may be complex relationships between the input variables and the target variable.

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value (Distance between hyperplane and boundary line), a. Thus, we can say that SVR model tries satisfy the condition -a < y-wx+b < a. It used the points with this boundary to predict the value.

In simple regression, the idea is to minimize the error rate while in SVR the idea is to fit the error inside a certain threshold which means, work of SVR is to approximate the best value within a given margin called ε- tube. Look at the figure below to understand SVR:



The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample. So let’s now dive deep and understand how SVR works actually.

A picture containing line, diagram, screenshot, text

Description automatically generated

These two red lines as the decision boundary and the green line as the hyperplane. Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line. Our best fit line is the hyperplane that has a maximum number of points.

To find the linear function

f(x)=x′β+b,

and ensure that it is as flat as possible, find f(x) with the minimal norm value (β′β). This is formulated as a convex optimization problem to minimize

J(β)=12β′β

subject to all residuals having a value less than ε; or, in equation form:

∀n:∣yn−(xn′β+b)∣≤ε .

It is possible that no such function f(x) exists to satisfy these constraints for all points. To deal with otherwise infeasible constraints, introduce slack variables ξn and ξ\*n for each point. This approach is similar to the “soft margin” concept in SVM classification, because the slack variables allow regression errors to exist up to the value of ξn and ξ\*n, yet still satisfy the required conditions.

Including slack variables leads to the objective function, also known as the primal formula[[5]](https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html#buytaw5):

J(β)=12β′β+CN∑n=1(ξn+ξ∗n) ,

subject to:

∀n:yn−(xn′β+b)≤ε+ξn∀n:(xn′β+b)−yn≤ε+ξ∗n∀n:ξ∗n≥0∀n:ξn≥0 .

The constant C is the box constraint, a positive numeric value that controls the penalty imposed on observations that lie outside the epsilon margin (ε) and helps to prevent overfitting (regularization). This value determines the trade-off between the flatness of f(x) and the amount up to which deviations larger than ε are tolerated.

Advantages of SVR

SVR is robust to the outliers.

Decision model can be easily updated

Can use multiple classifiers trained on the different types of data using the probability rules.

It can improve the prediction accuracy by measuring the confidence in classification.

SVR performs lower computation compared to other regression techniques.

Its implementation is easy.

### **Recurrent Kalman Networks**

The prediction and observation updates result in a new type of recurrent neural network, that we call Recurrent Kalman Network, which allows working in high dimensional state spaces while keeping numerical stability, computational efficiency and (relatively) low memory consumption.

The RKN provides a principled method to deal with absent inputs by just omitting the update step and setting the posterior to the priority.

The Recurrent Kalman Network, an end-to-end learning approach for Kalman filtering and prediction. While Kalman filtering in the original state space requiresapproximations due to the non-linear models, the RKN uses a learned high-dimensional latent state representation that

allows for efficient inference using locally linear transition models and a factorized belief state representation. Exploiting this representation allows us to avoid the expensive and numerically problematic matrix inversions involved in the KF equations. A picture containing text, diagram, line, font

Description automatically generated

Similar to the input gate in LSTMs (Hochreiter & Schmidhuber, 1997) and GRUs (Cho et al., 2014) the Kalman gain can be seen as a gate controlling how much the current

observation influences the state. However, this gating explicitly depends on the uncertainty estimates of the latent state and observation and is computed in a principled manner. While the sparse transition models and factorization assumptions may seem restrictive, they allow stable and efficient computations in high dimensional spaces. Since the high dimensional representation is learned jointly with the dynamics, this can yield very powerful models, as shown in

our experiments. In comparison to LSTMs and GRUs the number of parameters is considerably smaller. For a fixed bandwidth b and number of basis matrices k it scales linearly in the state size

for the RKN while it scales quadratically for LSTMs and GRUs. Moreover, the RKN provides a principled method to deal with absent inputs by just omitting the update step and setting the posterior to the

A picture containing diagram, plan, line, technical drawing

Description automatically generated[9]

An encoder network extracts latent features from the current observation . Additionally, it emits an estimate of the uncertainty in the features via the variance . The transition model is used to predict the current latent prior () using the last posterior () and subsequently update the prior using the latent observation (). As we use a factorized representation of , the Kalman update simplifies to scalar operations. The current latent state zt consists of the observable units as well as the corresponding memory units. Finally, a decoder produces either (), a low dimensional observation and an element-wise uncertainty estimate, or , a noise free image.

# **Result**

## **Performance measure**

To assess the predictive power of our proposed models, we use two performance measures: the root means square error (RMSE) and the MAPE. When we train models, we use RMSE as a loss function, and MAPE is a statistical measure of prediction accuracy. The following are the equations:

Where:

* f = forecasts (expected values or unknown results),
* o = observed values (known results).

Where:

* n is the number of fitted points,
* At is the actual value,
* Ft is the forecast value.
* Σ is summation notation (the absolute value is summed for every forecasted point in time).

MSLE is the relative difference between the log-transformed actual and predicted values.

The formula of the MSLE is :

Where:

A picture containing text, screenshot, number, parallel

Description automatically generated

A picture containing text, screenshot, number, font

Description automatically generated

## **Predictting next 30 days**

* Linear Regression (LN)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Support Vector Regession (SVR)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Auto Regressive Integrated Moving Average (ARIMA)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Vector Autoregression (VAR)

| A picture containing text, screenshot, plot, line  Description automatically generated | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Bayesian Vector Autoregression (BVAR)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* A Gated Recurrent Unit (GRU)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Recurrent Neural Network (RNN)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Long short-term memory (LSTM)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* A bidirectional LSTM (BiLSTM)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

* Recurrent Kalman Network (RKN)

| A blue line graph on a white background  Description automatically generated with medium confidence | A picture containing text, screenshot, plot, line  Description automatically generated |
| --- | --- |

# **Conclusion**

The following table are the experimental results of six models for training and testing, this study show that the BiLSTM model was the best for predicting the future price of the FTS, MWG, and VIC stocks in the resultant time series out of the nine models tested (LN, SVR, ARIMA, VAR, BVAR, RNN, BiLSTM, GRU, LSTM). The performance of the other models, such as the SVR and RNN, was subpar. This study emphasizes the significance of taking into account a range of modeling strategies in financial analysis, as well as the potential benefit of employing the BiLSTM model to forecast future stock prices. Future research could be done to confirm the findings of this study and to examine how well the other models perform on various stock price prediction tasks.ARIMA MODEL.

**REFERENCES**

[1] R. Bevans, “Simple Linear Regression | An Easy Introduction & Examples,” *Scribbr*, Feb. 19, 2020. https://www.scribbr.com/statistics/simple-linear-regression/ (accessed Jun. 17, 2023).

[2] G. Koop, “Bayesian Multivariate Time Series Methods for Empirical Macroeconomics,” *Found. Trends® Econom.*, vol. 3, no. 4, pp. 267–358, 2009, doi: 10.1561/0800000013.

[3] P. Dua and S. C. Ray, “A BVAR model for the connecticut economy,” *J. Forecast.*, vol. 14, no. 3, pp. 167–180, May 1995, doi: 10.1002/for.3980140303.

[4] “Gated Recurrent Unit (GRU) With PyTorch,” *FloydHub Blog*, Jul. 22, 2019. https://blog.floydhub.com/gru-with-pytorch/ (accessed Jun. 17, 2023).

[5] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, *An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting*. 2017. doi: 10.1007/978-3-319-70338-1.

[6] “Impact of News on the Trend of Stock Price Change: an Analysis based on the Deep Bidirectiona LSTM Model,” *Procedia Comput. Sci.*, vol. 174, pp. 128–140, Jan. 2020, doi: 10.1016/j.procs.2020.06.068.

[7] “Deep Learning | Introduction to Long Short Term Memory,” *GeeksforGeeks*, Jan. 16, 2019. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/ (accessed Jun. 17, 2023).

[8] A. Sethi, “Support Vector Regression Tutorial for Machine Learning,” *Analytics Vidhya*, Mar. 27, 2020. https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/ (accessed Jun. 18, 2023).

[9] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, J. Taylor, and G. Neumann, “Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces.” arXiv, May 17, 2019. Accessed: Jun. 18, 2023. [Online]. Available: http://arxiv.org/abs/1905.07357

**JOB ASSIGNMENT**

| **Nhiệm vụ** | **Đỗ Đặng Kiến Nam** | **Lê Anh Thư** | **Bùi Thị Thanh Ngân** |
| --- | --- | --- | --- |
| **Quản lý và phân chia nhiệm vụ** |  |  |  |
| **Tìm hiểu nghiên cứu kiến thức** |  |  |  |
| **Viết report và paper** |  |  |  |
| **Làm slide** |  |  |  |
| **Thực hiện thuật toán LN, SVR, ARIMA** |  |  |  |
| **Thực hiện thuật toán VAR, BVAR, GRU** |  |  |  |
| **Thực hiện thuật toán RNN, LSTM, RKN** |  |  |  |