**VIETNAM NATIONAL HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**------\*\*\*------**Logo

Description automatically generated

**IS403.N22.HTCL**

**FINAL PROJECT**

**BUSINESS ANALYST**

Instructor:

Associate professor PhD. Nguyen Dinh Thuan

Senior. Nguyen Thi Viet Huong

Student performance:

| 20521938 | Tran Phuong Thao |
| --- | --- |
| 20522176 | Dang Thi Tuong Vy |
| 20521582 | Nguyen Thanh Luan |

***Ho Chi Minh, March 2023***

**TEACHER’S COMMENT**

**Table of contents**

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

[**II. METHOD 5**](#_heading=h.30j0zll)

[**A. Dataset. 7**](#_heading=h.1fob9te)

[**B. Tool used. 7**](#_heading=h.3znysh7)

[**C. Model. 8**](#_heading=h.2et92p0)

[**1. Linear Regression 8**](#_heading=h.tyjcwt)

[**2. Support vector regression (SVR) 8**](#_heading=h.3dy6vkm)

[**3. K-Nearest Neighbors (KNN) 9**](#_heading=h.1t3h5sf)

[**4. Convolutional Neural Network (CNN) 11**](#_heading=h.4d34og8)

[**5. Hidden Markov Model (HMM) 13**](#_heading=h.2s8eyo1)

[**6. ARIMA 15**](#_heading=h.17dp8vu)

[**7. ARIMAX 16**](#_heading=h.3rdcrjn)

[**8. Long Short-Term Memory (LSTM) 17**](#_heading=h.26in1rg)

[**9. Gated Recurrent Unit (GRU) 19**](#_heading=h.lnxbz9)

[**D. Metrics. 21**](#_heading=h.35nkun2)

[**1. Mean Square Error (MSE) 21**](#_heading=h.1ksv4uv)

[**2. Root Mean Square Error (RMSE) 21**](#_heading=h.2jxsxqh)

[**3. Mean Absolute Percentage Error (MAPE) 22**](#_heading=h.z337ya)

[**III. RESULT 23**](#_heading=h.3j2qqm3)

[**A. AMAZON. 23**](#_heading=h.1y810tw)

[**B. NVIDIA. 24**](#_heading=h.4i7ojhp)

[**C. TESLA. 25**](#_heading=h.2xcytpi)

[**IV. CONCLUSION 26**](#_heading=h.1ci93xb)

[**V. REFERENCES 26**](#_heading=h.3whwml4)

# **I. INTRODUCTION**

Forecasting fluctuations in stock prices hold significant significance for investors, publicly listed corporations, and governmental bodies. The issue of whether the market can be reliably predicted has been a topic of enduring discussions. According to Malkiel's (1973) Random Walk Theory, stock prices are arbitrarily set, making it impractical to outperform the market. However, the emergence of artificial intelligence has demonstrated empirical proof that forecasting stock price movements is indeed achievable.

Additionally, the primary aim of this problem is to analyze the patterns and variations in stock prices to facilitate well-informed investment choices. The ability to predict stock prices can be instrumental in asset valuation, uncovering potential investment prospects, and effectively mitigating risks.

To predict stock prices, various methods, and tools are commonly used, including technical analysis, fundamental analysis, and data modeling. Technical analysis involves studying historical patterns and trends in stock prices to make predictions about the future. Fundamental analysis involves researching economic, financial, and company factors to evaluate the fundamental value of a company and thereby predict stock prices. Data modeling utilizes machine learning and artificial intelligence methods to analyze historical data and build predictive models for future stock prices.

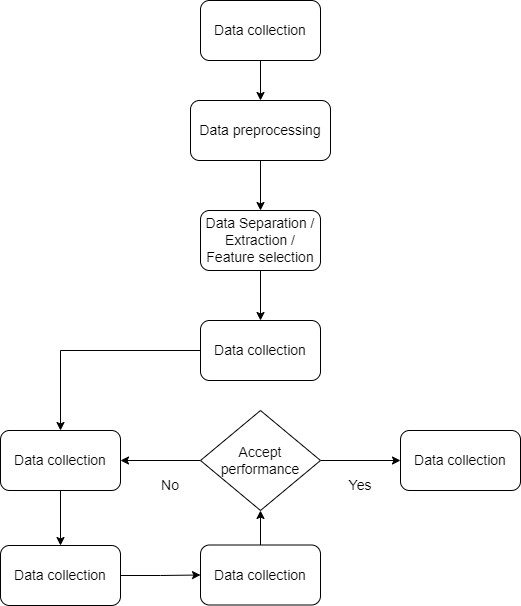
Linear regression is a widely used algorithm in financial analysis, predicting stock and providing a simple yet effective approach to modeling the relationship between independent and dependent variables. In addition, machine learning algorithms that are often used in stock price prediction problems such as KNN, SVR also give quite good and reliable prediction results. With ARIMA, ARIMAX models, on the other hand, are specifically designed for time series forecasting and have been successfully applied in various domains. LSTM and GRU are deep learning architectures that excel at capturing long-term dependencies. To evaluate the problem as well as the performance of the machine learning model used to predict in the above data set, we will conduct an analysis of how, data, accuracy, and efficiency in the stock prediction process. 9 models have been introduced.

In addition to evaluating the performance and comparing results of machine learning algorithms using stock price data, this research paper aims to delve deeper into understanding the strengths and weaknesses of each algorithm. By analyzing their accuracy, robustness, and computational efficiency, we seek to gain insights into their applicability and limitations in the context of stock price prediction.

By conducting a thorough analysis and comparison of these machine learning algorithms, we strive to provide valuable insights into their performance in the dynamic and unpredictable stock market. This research can potentially uncover new perspectives and approaches for forecasting stock prices, benefiting both individual investors and institutional traders in optimizing their investment strategies.

# **II. METHOD**

Input is time series historical data of stock price. After the data goes through the preparation step: the selection of required features and preprocessing splits the data according to the training, validation, and testing dataset accordingly. Then will use Linear Regression, ARIMA, ARIMAX, LSTM, GRU, CNN, HMM, SVR, KNN to train. The selected model is trained using the prepared data set. During training, the model learns the underlying patterns, relationships, and structures in the data. The training process involves tuning the model’s parameters to minimize the difference between the predicted output and the real output from the training data. After the model is trained, it needs to be evaluated to evaluate its performance and generalizability, this step involves using a separate evaluation dataset or using techniques such as cross-validate to estimate the model’s performance on unseen data. Different metrics are used to measure performance depending on the specific problem, if after evaluating the model performance is not effective then go back to adjust the hyperparameters and then do the training again. Finally, the results are presented in the form of graphs.

**

*Figure 1: Implementation process*

The research process goes through the following steps:

* Step 1: Data collection - collect a representative and relevant dataset to train the model. The dataset must include input data (features) and corresponding output data (labels or targets) in this study we choose time series historical data of stock price.
* Step 2: Data preprocessing - After the data set is collected, it usually requires preprocessing. This step involves cleaning the data, dealing with missing values, removing outliers, and converting the data to a format suitable for training.
* Step 3: Data Separation / Extraction / Feature selection. In this step, we separate the data at different rates including train, test, validation data, and then select features to conduct training.
* Step 4: Model selection - in this study, we will perform on models including Linear Regression, ARIMA, ARIMAX, LSTM, GRU, CNN, HMM, SVR, KNN.
* Step 5: Tune the hyperparameters - set the appropriate hyperparameters for several models.
* Step 6: Model Training - The selected model is trained using the prepared dataset.
* Step 7: Model evaluation: After the model is trained, it needs to be evaluated to evaluate its performance and generalizability. If the performance is not acceptable, go back to step 5.
* Step 8: Show result.

## A. Dataset.

The dataset is a trio of stock prices including Amazon (AMZN), Nvidia (NVDA), Tesla (TSLA). The data is taken from Yahoo! Financeii, a media website that is part of Yahoo! network. It provides financial news, data, and commentary including stock quotes, press releases, financial reports, and original content. It also offers a number of online tools for managing personal finances. In this paper, the time period represented starts from 2017-12-01 to 2023-06-09, and the attributes of the dataset are described in table I.

*TABLE I. ATTRIBUTES IN THE STOCK PRICE DATASET*

| **Attribute** | **Describe** |
| --- | --- |
| Date | Stock price trading day |
| Open | The initial opening/price at a certain time |
| High | Highest opening price |
| Low | Lowest price of opening price |
| Close | The closing/final price of the stock at a certain time |
| Adj Close | Closing price after adjustment |
| Volume | Number of transactions during the day |

Table II describes the measure of features selected for training in the dataset. In this paper, we choose the closing price of the stock price to perform.

*TABLE II. DESCRIBES THE MEASURE OF FEATURES SELECTED FOR TRAINING IN THE DATASET*

| **Measure** | **Amazon (AMZN)** | **Nvidia (NVDA)** | **Tesla (TSLA)** |
| --- | --- | --- | --- |
| Count | 1389 | 1389 | 1389 |
| Mean | 117.56 | 126.35 | 133.88 |
| Standard deviation | 1227.99 | 6621.81 | 12986.89 |
| Min | 56.70 | 31.77 | 11.93 |
| Max | 186.57 | 401.11 | 409.97 |

## B. Tool used.

During the experiment, we use Python language and the support tool is Google Colab. In addition, we use Python’s built-in libraries such as Pandas to process data in the form of data frames. Matplotlib plots to visualize data. The figure below illustrates the result after completing a model. Numpy helps with math and matrix operations in this experiment. And finally, the Scikit-learn library supports machine learning and regression models.

## C. Model.

### 1. Linear Regression

Linear regression is a type of statistical analysis used to predict the relationship between two variables. It assumes a linear relationship between the independent variable and the dependent variable, and aims to find the best-fitting line that describes the relationship. [1]

Linear Regression is a simple yet powerful and mostly used algorithm in data science.

The formula is given as:

* represents the dependent variable (output) that we want to predict or explain.
* are the independent variables (inputs) used to predict Y.
* are the regression coefficients corresponding to the independent variables. They represent the extent of influence of each independent variable on the dependent variable.
* is the error term, representing the difference between the actual value of Y and the predicted value by the model.

Linear regression is commonly used in many fields, including economics, finance, and social sciences, to analyze and predict trends in data. It can also be extended to multiple linear regression, where there are multiple independent variables, and logistic regression, which is used for binary classification problems.

### 2. Support vector regression (SVR)

Support Vector Regression (SVR) [2] is a powerful machine learning algorithm for regression tasks. It is a variant of Support Vector Machines (SVM) and is specifically designed for continuous target variables. SVR aims to find a function that approximates the relationship between input variables and the target variable. It uses support vectors, kernel functions, and regularization to achieve this. SVR is commonly used in stock market prediction, time series analysis, and financial forecasting.

Unlike traditional regression models, SVR focuses on minimizing the prediction error rather than fitting the data precisely. It accomplishes this by finding an optimal hyperplane that maximizes the margin, or distance, between the predicted values and the actual data points. SVR achieves a balance between simplicity and flexibility by allowing a specified tolerance, or margin of error, around the predicted values.

SVR is effective in dealing with intricate connections between input variables and the target variable, thanks to its utilization of a kernel function. By mapping the data to a higher-dimensional space, SVR can handle non-linear relationships. This capability enhances its strength in regression tasks that involve complex associations between the input variables and the target variable.

Kernel Functions: SVR uses kernel functions to transform the input variables into a higher-dimensional feature space. This transformation enables SVR to capture complex relationships between the features and the target variable. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. As SVR can only interpolate within the known data points in the training dataset, the accuracy of its estimates beyond that domain relies on the selected kernel and optimization methods employed. [3]

| **Kernel** | **Introduce** | **Function** |
| --- | --- | --- |
| Linear | This is the simplest and most basic kernel in SVR. It performs a linear projection of the input data into a linear space. |  |
| Polynomial | This kernel transforms the input data into a polynomial space, determined by a degree parameter and a bias coefficient. |  |
| RBF | This kernel uses the RBF function to transform the input data into a non-linear space. The RBF function is defined by a slope parameter |  |
| Sigmoid | This kernel transforms the input data into a non-linear space using the sigmoid function. It can be used for problems with binary data. |  |

### 3. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN)[4] is a versatile machine learning algorithm widely employed in both classification and regression tasks. Although it is commonly recognized for its application in classification, KNN can also be adapted to tackle regression analysis, which involves predicting continuous target variables.

In KNN regression, the algorithm operates by identifying the K closest neighbors to a specific data point within the feature space. Unlike KNN classification, where a class label is assigned, KNN regression computes the average or weighted average of the target variable values for these K neighbors. This aggregated value is then used as the predicted value for the target variable of the query data point.

Several distance formulas are commonly used to calculate the k nearest neighbors for 2 data points, x and y, with k attributes[4]

* Euclidean
* Manhattan
* Minkowski

-

The basic steps of the K-Nearest Neighbors[5] algorithm are:

Step 1: First, we need to determine the number of nearest neighbors (K) to use in the prediction process. Typically, K is a number that the user determines beforehand. However, there are also some algorithms that help users easily determine the parameter k to make the algorithm run more efficiently.

Step 2: Calculate the distance between the data point being predicted and all the data points in the training set. The distance is usually calculated using the Euclidean or Manhattan distance metric.

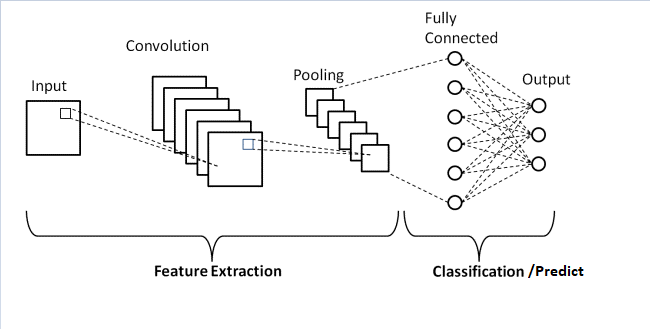
Step 3: Select K data points with the closest distance to the data point being predicted. These are the nearest neighbors. These nearest data points will play an important role in the prediction process.

Step 4: In regression problems, take the average value of the K nearest data points (neighbors) as the predicted value for the data point.

### 4. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) have emerged as a powerful deep learning architecture for image and video processing tasks [10]. Originating from the 1980s, early developments in CNNs laid the foundation for their success. LeNet-5, proposed by Yann LeCun in 1998, played a significant role in popularizing CNNs and demonstrated their effectiveness in handwritten digit recognition. Subsequent advancements have shown that CNNs can be applied to various problems and tasks. The CNN allows users to build classification and prediction systems with extremely high accuracy.

A convolutional neural network consists of an input layer, hidden layers, and an output layer. In a convolutional neural network, the hidden layers include one or more layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU[11]. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, flatten layer, fully connected layers, and normalization layers[10].



*Figure 2: Convolutional Neural Network(CNN) architecture diagram[10]*

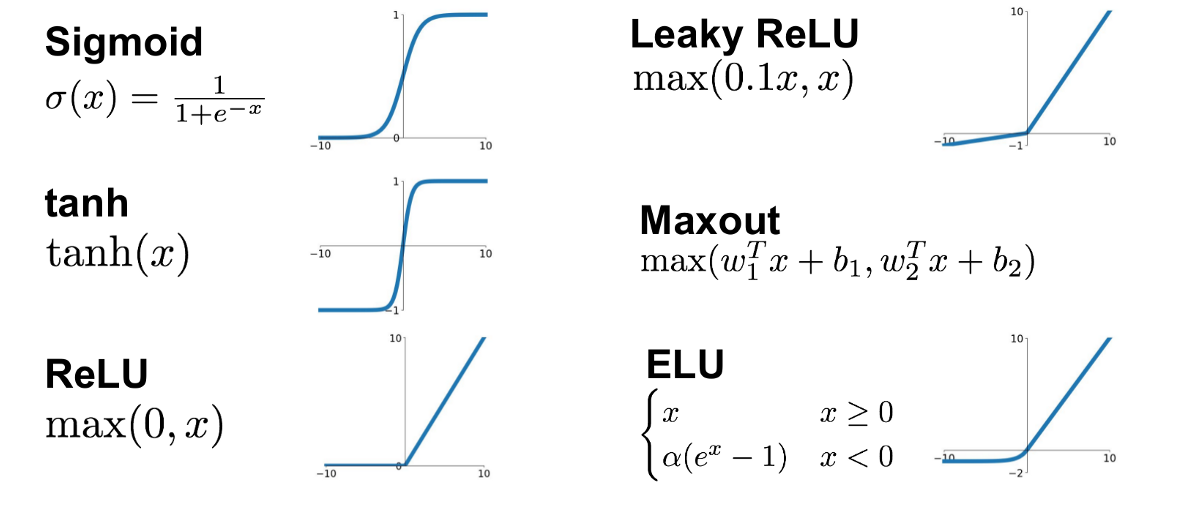
Convolution layer: When using Convolutional layer in a CSV dataset, we perform convolution between the columns in the data table and the filters (kernels) to generate feature maps. These feature maps are used to extract features from the data.

Max pooling layer: This layer is used to reduce the size of the feature maps and the number of parameters in the model. The Max pooling layer works by dividing the feature map into non-overlapping pooling regions and selecting the maximum value from each region to include in the output. This selection of the maximum value preserves the important features of the data while ignoring small irrelevant details.

Flatten layer: After using the Convolutional layer and Max pooling layer to extract features from the CSV data, we obtain feature maps. These feature maps can be processed by the Flatten layer to convert them into a vector that can be input to the Fully connected layer.

Fully connected layer (FCL): When using FCL, the features of the data are represented as a vector and are input into FCL layers to perform classification or prediction. These FCL layers learn the complex relationships between the features and provide predictions about the outcome.

In this paper, the activation function that our group used is ReLU, which has the following formula[11]



*Figure 3: Activation of Convolutional Neural Network*

During the training process, CNN uses the backpropagation algorithm to adjust the weights of the network layers in order to minimize the loss function between the predicted output and the actual output. Optimization algorithms such as Gradient Descent and its variants are commonly applied to update the network weights.

Step1- Preparing the data: You need to prepare the data for the CNN model, including both the training and testing datasets. The data can be in the form of CSV files with columns containing information about the features of the data and a column for the target output.

Step2- Building the CNN model: You need to construct a suitable CNN model for your problem. First, you need to convert the input data into a tensor that can be used by the CNN model. Then, you can build a CNN model with convolutional layers, pooling layers, and fully connected layers to extract features and predict the data.

Step3-Training the model: You need to train the CNN model with the training dataset. The goal is to optimize the model's weights to maximize accuracy and minimize errors.

Step4-Testing and evaluating the model: After training, you need to test and evaluate the model with the testing dataset. Evaluate the model by calculating metrics such as accuracy, sensitivity, and specificity.

### 5. Hidden Markov Model (HMM)

Out of the several models available for modelling discrete sequential data, hidden Markov models are one of the more simple but powerful ones— they have been applied successfully to a wide variety of fields such as statistical mechanics, DNA analysis, speech recognition and stock market predictions. [6]

The first order Markov process makes a very important simplification to observed sequential data—the current system state depends only on the previous system state.

Additionally, hidden Markov models make one more important modification to the Markov process — the actual system states are assumed to be unobservable and are hidden. For a sequence of hidden states Z, the hidden Markov process emits a corresponding sequence of observable processes X. Using the observed processes X, we try to guess what Z really is using hidden Markov models!

A picture containing diagram, sketch, circle, line

Description automatically generated

*Figure 4: Graph of a hidden Markov process. We are unable to observe the actual hidden states of the system Z, and can only observe the observable processes X.*

Hidden Markov models are used to model a hidden Markov process. Hidden Markov models are defined by the following 3 model parameters:

* Initial hidden state probabilities . This vector describes the initial probabilities of the system being in a particular hidden state
* Hidden state transition matrix A. Each row in A corresponds to a particular hidden state, and the columns for each row contain the transition probabilities from the current hidden state to a new hidden state.
* Observable emission probabilities This vector describes the emission probabilities for the observable process Xᵢ given some hidden state .

The Hidden Markov Model (HMM) algorithm can be implemented using the following steps:

Step 1: Define the state space and observation space. The state space is the set of all possible hidden states, and the observation space is the set of all possible observations.

Step 2: Define the initial state distribution. This is the probability distribution over the initial state.

Step 3: Define the state transition probabilities. These are the probabilities of transitioning from one state to another. This forms the transition matrix, which describes the probability of moving from one state to another.

Step 4: Define the observation likelihoods: These are the probabilities of generating each observation from each state. This forms the emission matrix, which describes the probability of generating each observation from each state.

Step 5: Train the model. The parameters of the state transition probabilities and the observation likelihoods are estimated using the Baum-Welch algorithm, or the forward-backward algorithm. This is done by iteratively updating the parameters until convergence.

Step 6: Decode the most likely sequence of hidden states. Given the observed data, the Viterbi algorithm is used to compute the most likely sequence of hidden states. This can be used to predict future observations, classify sequences, or detect patterns in sequential data.

Step 7: Evaluate the model. The performance of the HMM can be evaluated using various metrics, such as accuracy, precision, recall, or F1 score. [7]

### 6. ARIMA

ARIMA is an autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends.

A statistical model is autoregressive if it predicts future values based on past values. For example, an ARIMA model might seek to predict a stock's future prices based on its past performance or forecast a company's earnings based on past periods.

Understanding Autoregressive Integrated Moving Average (ARIMA)

An ARIMA model can be understood by outlining each of its components as follows:

* Autoregression (AR): refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
* Integrated (I): represents the differencing of raw observations to allow the time series to become stationary (i.e., data values are replaced by the difference between the data values and the previous values).
* Moving average (MA):  incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each component in ARIMA functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

* p: the number of lag observations in the model, also known as the lag order.
* d: the number of times the raw observations are differenced; also known as the degree of differencing.
* q: the size of the moving average window, also known as the order of the moving average. [8]

The formula for a simple ARIMA(p,d,q) is:

In this formula:

* is the value of the time series at time t.
* is a constant term representing the mean value of the time series.
* are the autoregressive (AR) coefficients for the lagged values in the model.
* are the moving average (MA) coefficients for the lagged error terms in the model.
* is the error term or residual at time t.

ARIMA models have strong points and are good at forecasting based on past circumstances.

* Good for short-term forecasting
* Only needs historical data.
* Models non-stationary data

ARIMA is a method for forecasting or predicting future outcomes based on a historical time series. It is based on the statistical concept of serial correlation, where past data points influence future data points.

### 7. ARIMAX

ARIMAX (Autoregressive Integrated Moving Average with Exogenous Variables) is an extension of the ARIMA model that incorporates exogenous or external variables into the time series analysis. X added to the end stands for “exogenous”. In other words, it suggests adding a separate different outside variable to help measure our endogenous variable.

X is the exogenous variable and it can be any variable. It can be a time-varying measurement like the inflation rate or the price of a different index. Or a categorical variable separating the different days of the week. It can also be a Boolean accounting for the special festive periods. Finally, it can stand for a combination of several different external factors.

The formula for a simple ARIMAX(p,d,q) is:

* are the exogenous variables at time t.
* are the coefficients associated with the exogenous variables, representing their impact on the dependent variable.

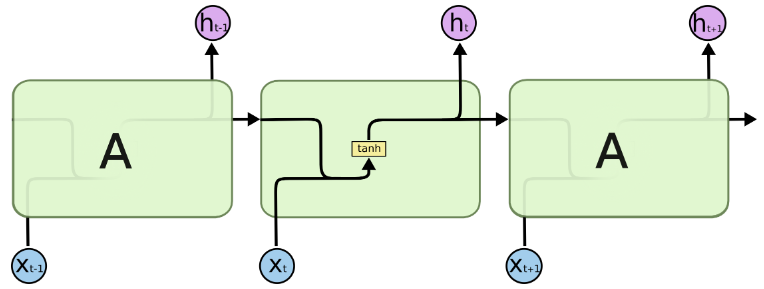
ARIMAX models are used when we want to analyze and forecast time series data, taking into account not only past values but also the impact of exogenous variables.[9]

### 8. Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) [12] model is a type of recurrent neural network (RNN) that is specifically designed to handle long-term dependencies in sequential data. It was introduced by Hochreiter and Schmidhuber in 1997 and has since gained popularity in various domains, including natural language processing, speech recognition, and time series forecasting.

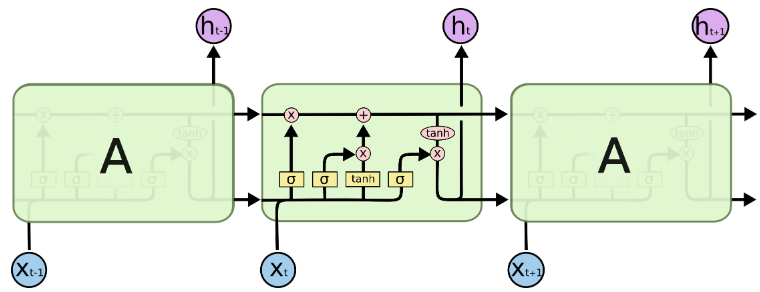
Unlike standard feedforward neural networks, LSTM incorporates feedback connections, making it a recurrent neural network (RNN)[12]. This enables LSTM to not only process individual data points like images but also handle sequential data such as speech or video. LSTM, being a special type of RNN, has demonstrated exceptional performance across a wide range of problem domains. And LSTM is designed to address the issue of long-term dependencies. It has the inherent ability to remember information over extended periods of time without the need for explicit training. This means that LSTMs can retain information by default, without any additional intervention.

Every recurrent neural network (RNN) has a sequential structure consisting of repeating modules of neural networks. In standard RNNs, these modules have a very simple structure, often a single tanh layer. [15]



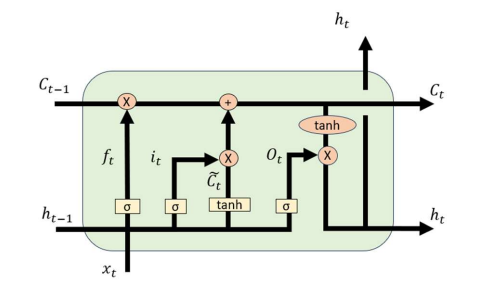
*Figure 5: he repeating module in a standard RNN contains a single layer*

LSTM also has a similar sequential structure, but its modules have a unique structure with 4 interacting layers.



*Figure 6: the LSTM model architecture*

Can be briefly presented as follows.



*Figure 7: the LSTM model architecture*

In state t of the LSTM model:[15]

Output:

Input:

* c is cell state, h is hidden state.
* output of previous layer.
* input in state t.

corresponding to forget gate, input gate, and output gate.

* Forget gate:

[15]

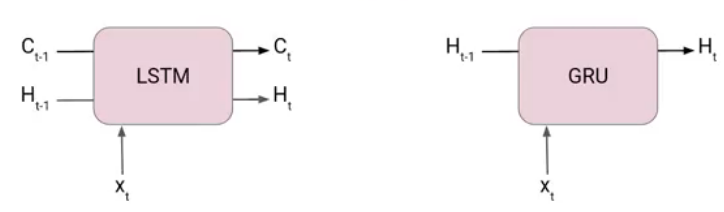
* Input gate:

[15]

* Output gate:

### 9. Gated Recurrent Unit (GRU)

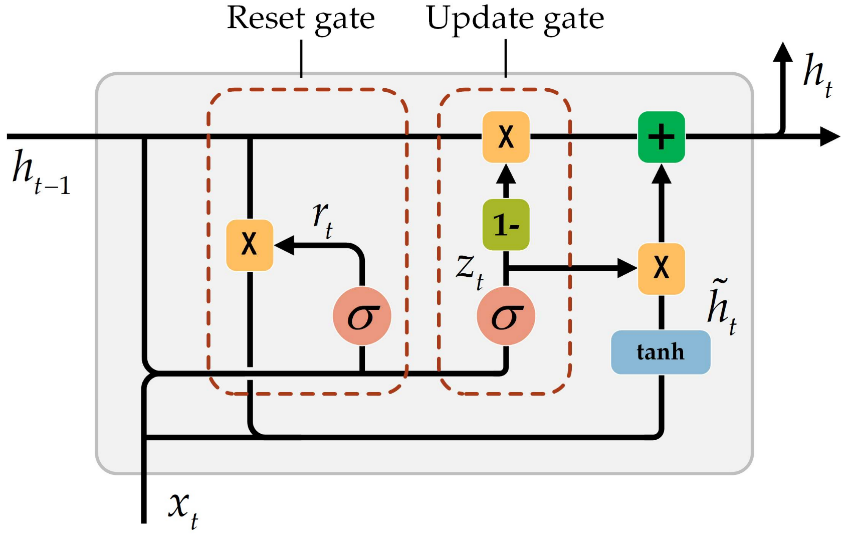
The GRU (Gated Recurrent Unit)[13] algorithm is a recurrent neural network (RNN) architecture used to process natural language and time series data and is an improved version of the LSTM (Long Short-Term Memory) network. GRUs are very similar to Long Short Term Memory(LSTM). Just like LSTM, GRU uses gates to control the flow of information. They are relatively new as compared to LSTM. This is the reason they offer some improvement over LSTM and have simpler architecture.



*Figure 8: Difference between LSTM and GRU*

Another Interesting thing about GRU is that, unlike LSTM, it does not have a separate cell state (Ct). It only has a hidden state(Ht). Due to the simpler architecture, GRUs are faster to train.[13]

The architecture of Gated Recurrent Unit:



*Figure 9: Model of GRU architectural*

Output:

Input:

* h is the hidden state.
* output of previous layer.
* input in state t.

corresponding to reset gate, and update gate.

At each timestamp t, it takes an input and the hidden state from the previous timestamp t-1. Later it outputs a new hidden state which again passed to the next timestamp.

The LSTM gate equation it is very similar to that. The value of rt will range from 0 to 1 because of the sigmoid function. Here and are weight matrices for the reset gate.

* Reset Gate (Short-term memory)[14]
* Update Gate (Long-term memory)[14]

From the above theories and formulas, we can conclude GRU uses less training parameter and therefore uses less memory and executes faster than LSTM whereas LSTM is more accurate on a larger dataset. One can choose LSTM if you are dealing with large sequences and accuracy is concerned, GRU is used when you have less memory consumption and want faster results. [14]

## D. Metrics.

### 1. Mean Square Error (MSE)

Mean Square Error (MSE) is perhaps a widely used measure in regression problems to calculate the average squared difference between estimated values and true values. It represents the average squared difference between the predicted and actual values.

In there

+ the actual value of the data at the sample

+ is the predicted value at the point

+ is the number of data points.

In regression analysis, plotting a graph is a more intuitive way to visualize the overall trend of the data. Simply put, MSE tells you how close the regression line is to a set of points. It does this by measuring the distances between the points and the regression line (these distances are called "residuals") and squaring them. Squaring is important to reduce complexity and give more weight to larger deviations or negative indications. Therefore, a lower MSE indicates better predictive accuracy.

### 2. Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a commonly used metric or measure to quantify the differences between predicted values (by a model or an estimation) and observed values. It provides an overall assessment of the accuracy of the predictions by considering the average squared differences. By taking the square root of the mean squared differences, RMSE gives a measure that is in the same unit as the original values, making it easier to interpret and compare the performance of different models or estimations.

In other words, Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals measure the distance from data points to the regression line, and RMSE measures the spread of these residuals. Essentially, it informs you about the concentration of data around the best-fit line. RMSE is commonly used in climatology, forecasting, and regression analysis to validate experimental results.

In there

+ the actual value of the data at the sample

+ is the predicted value at the point

+ is the number of data point.

Root Mean Square Error (RMSE) is a measure of the effectiveness of your model. It accomplishes this by quantifying the difference between predicted values and actual values. A smaller RMSE indicates smaller errors, which implies a higher level of confidence in the model's estimates.

### 3. Mean Absolute Percentage Error (MAPE)

Root Mean Square Error (RMSE) is a commonly used metric or measure to quantify the differences between predicted values (by a model or an estimation) and observed values. It provides an overall assessment of the accuracy of the predictions by considering the average squared differences. By taking the square root of the mean squared differences, RMSE gives a measure that is in the same unit as the original values, making it easier to interpret and compare the performance of different models or estimations.

In other words, Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals measure the distance from data points to the regression line, and RMSE measures the spread of these residuals. Essentially, it informs you about the concentration of data around the best-fit line. RMSE is commonly used in climatology, forecasting, and regression analysis to validate experimental results.

In there

+ the actual value of the data at the sample

+ is the predicted value at the point

+ is the number of data points.

# **III. RESULT**

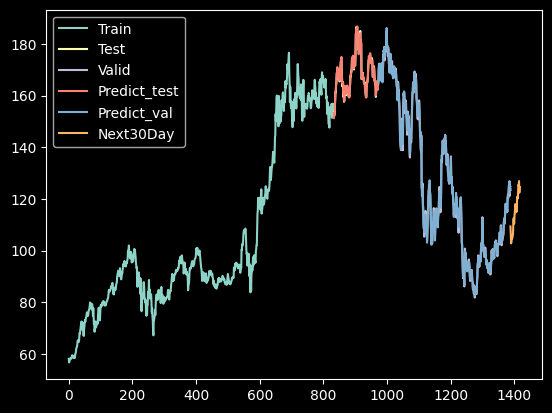
## A. AMAZON.

*TABLE III: RESULTS WHEN TRAINING ON AMAZON DATASETS*

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMA | 7 : 2 : 1 | 9.66 | 6.11 | 11.50 |
| 6 : 3 : 1 | 20.45 | 11.96 | 22.08 |
| 5 : 3 : 2 | 8.68 | 5.25 | 10.59 |
| LN | 7 : 2 : 1 | 20.63 | 13.72 | 25.98 |
| 6 : 3 : 1 | 9.03 | 5.23 | 11.18 |
| 5 : 3 : 2 | 28.02 | 17.05 | 28.85 |
| SVR | 7 : 2 : 1 | 9.28 | 7.82 | 11.16 |
| 6 : 3 : 1 | 7.46 | 5.80 | 9.51 |
| 5 : 3 : 2 | 29.32 | 18.09 | 30.14 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| CNN | 7 : 2 : 1 | 6.01 | 4.60 | 7.41 |
| 6 : 3 : 1 | 5.02 | 3.51 | 6.23 |
| 5 : 3 : 2 | 3.69 | 2.29 | 4.81 |
| LSTM | 7 : 2 : 1 | 3.59 | 2.77 | 4.58 |
| 6 : 3 : 1 | 3.22 | 2.23 | 4.08 |
| 5 : 3 : 2 | 3.13 | 1.98 | 4.11 |
| GRU | 7 : 2 : 1 | 2.97 | 2.23 | 3.99 |
| 6 : 3 : 1 | 2.61 | 1.87 | 3.56 |
| 5 : 3 : 2 | 2.73 | 1.71 | 3.72 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMAX | 7 : 2 : 1 | 1.06 | 0.69 | 1.32 |
| 6 : 3 : 1 | 0.59 | 0.35 | 0.77 |
| 5 : 3 : 2 | 0.77 | 0.48 | 1.04 |
| HMM | 7 : 2 : 1 | 2.62 | 1.64 | 3.31 |
| 6 : 3 : 1 | 1.66 | 0.98 | 2.11 |
| 5 : 3 : 2 | 2.04 | 1.29 | 2.65 |
| KNN | 7 : 2 : 1 | 1.69 | 1.28 | 2.32 |
| 6 : 3 : 1 | 1.78 | 1.23 | 2.51 |
| 5 : 3 : 2 | 1.75 | 1.07 | 2.48 |



*Figure 10: ARIMAX on AMAZON DATASET WITH TRAIN: TEST: VALID - 6:3:1*

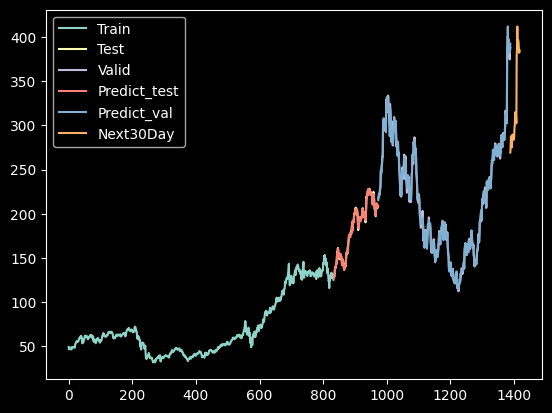
## B. NVIDIA.

*TABLE III: RESULTS WHEN TRAINING ON NVIDIA DATASETS*

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMA | 7 : 2 : 1 | 57.75 | 20.83 | 67.23 |
| 6 : 3 : 1 | 61.67 | 31.56 | 68.79 |
| 5 : 3 : 2 | 35.95 | 19.26 | 50.89 |
| LN | 7 : 2 : 1 | 88.69 | 32.67 | 97.41 |
| 6 : 3 : 1 | 64.53 | 33.63 | 69.36 |
| 5 : 3 : 2 | 79.28 | 48.34 | 84.79 |
| SVR | 7 : 2 : 1 | 42.28 | 26.16 | 47.11 |
| 6 : 3 : 1 | 33.76 | 14.46 | 45.08 |
| 5 : 3 : 2 | 38.70 | 16.83 | 48.17 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| CNN | 7 : 2 : 1 | 17.44 | 6.03 | 21.74 |
| 6 : 3 : 1 | 21.43 | 7.48 | 28.73 |
| 5 : 3 : 2 | 26.64 | 10.61 | 32.68 |
| LSTM | 7 : 2 : 1 | 11.29 | 3.83 | 14.35 |
| 6 : 3 : 1 | 8.68 | 3.26 | 11.38 |
| 5 : 3 : 2 | 25.20 | 10.32 | 32.46 |
| GRU | 7 : 2 : 1 | 7.25 | 3.40 | 9.53 |
| 6 : 3 : 1 | 7.84 | 2.83 | 10.79 |
| 5 : 3 : 2 | 10.31 | 3.92 | 14.27 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMAX | 7 : 2 : 1 | 3.18 | 1.24. | 4.13 |
| 6 : 3 : 1 | 1.06 | 0.58 | 1.31 |
| 5 : 3 : 2 | 1.07 | 0.71 | 1.41 |
| HMM | 7 : 2 : 1 | 7.59 | 2.83 | 9.82 |
| 6 : 3 : 1 | 2.88 | 1.70 | 3.16 |
| 5 : 3 : 2 | 2.37 | 1.81 | 3.02 |
| KNN | 7 : 2 : 1 | 1.67 | 1.28 | 2.32 |
| 6 : 3 : 1 | 1.77 | 1.23 | 2.51 |
| 5 : 3 : 2 | 1.75 | 1.07 | 2.48 |



*Figure 11: ARIMAX on AMAZON DATASET WITH TRAIN:TEST:VALID - 6:3:1*

## C. TESLA.

*TABLE III: RESULTS WHEN TRAINING ON TESLA DATASETS*

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMA | 7 : 2 : 1 | 66.80 | 19.40 | 75.53 |
| 6 : 3 : 1 | 26.41 | 11.02 | 32.48 |
| 5 : 3 : 2 | 87.02 | 37.43 | 97.98 |
| LN | 7 : 2 : 1 | 96.52 | 28.41 | 105.00 |
| 6 : 3 : 1 | 62.32 | 26.94 | 64.42 |
| 5 : 3 : 2 | 145.86 | 66.92 | 151.27 |
| SVR | 7 : 2 : 1 | 20.56 | 7.73 | 26.63 |
| 6 : 3 : 1 | 75.26 | 28.20 | 85.26 |
| 5 : 3 : 2 | 58.42 | 22.96 | 66.25 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| CNN | 7 : 2 : 1 | 17.44 | 6.03 | 21.74 |
| 6 : 3 : 1 | 21.43 | 7.48 | 28.73 |
| 5 : 3 : 2 | 26.68 | 10.61 | 32.68 |
| LSTM | 7 : 2 : 1 | 11.29 | 3.83 | 14.35 |
| 6 : 3 : 1 | 8.68 | 3.26 | 11.38 |
| 5 : 3 : 2 | 25.20 | 10.32 | 32.46 |
| GRU | 7 : 2 : 1 | 7.25 | 3.40 | 9.53 |
| 6 : 3 : 1 | 7.84 | 2.83 | 10.79 |
| 5 : 3 : 2 | 10.31 | 3.92 | 14.27 |

| **Model** | **Train : Test : Valid** | **Measure** | | |
| --- | --- | --- | --- | --- |
| **MAE** | **MAPE** | **RMSE** |
| ARIMAX | 7 : 2 : 1 | 4.32 | 1.34 | 5.44 |
| 6 : 3 : 1 | 1.56 | 0.69 | 2.03 |
| 5 : 3 : 2 | 2.10 | 1.02 | 2.76 |
| HMM | 7 : 2 : 1 | 11.43 | 3.51 | 14.78 |
| 6 : 3 : 1 | 4.47 | 2.03 | 5.89 |
| 5 : 3 : 2 | 4.81 | 2.59 | 6.24 |
| KNN | 7 : 2 : 1 | 1.69 | 1.28 | 2.32 |
| 6 : 3 : 1 | 1.77 | 1.23 | 2.51 |
| 5 : 3 : 2 | 1.75 | 1.07 | 2.48 |

A picture containing text, screenshot, graphics software, font

Description automatically generated

*Figure 12: ARIMAX on TESLA DATASET WITH TRAIN:TEST:VALID - 6:3:1*

# **IV. CONCLUSION**

Stocks present themselves as attractive investment opportunities, but they also come with inherent risks stemming from their volatile and unpredictable nature. Fortunately, these advanced algorithms leverage the power of neural networks to capture complex patterns and trends in stock price data. By analyzing historical data and identifying anomalies, these algorithms can provide valuable insights for forecasting future price movements. The robust performance. ARIMAX algorithms underscore their effectiveness in handling the dynamic and volatile nature of stock markets. With their ability to deliver accurate predictions, these algorithms hold significant potential for informing investment decisions and risk management strategies in the stock predict.

Among the nine models tested (Linear Regression, ARIMA, ARIMAX, SVR, KNN, LSTM, CNN, GRU, and HMM), the results of this study demonstrate that the ARIMAX models were the most suitable for accurately predicting the future prices of. AMAZON, NVIDIA, TESLA stocks in the given time series. This study highlights the importance of exploring diverse modeling approaches when predicting stock prices and highlights the potential effectiveness of employing ARIMAX models for forecasting future stock prices. The findings suggest that these models can provide valuable insights for investors and analysts in making informed decisions in the stock market.

The utilization of regression and artificial intelligence algorithms, including ARIMAX, has shown significant effectiveness in forecasting stock prices. However, additional research and implementation of further techniques are necessary to enhance these models by incorporating additional external variables that impact stock prediction. With ongoing advancements in data analysis, artificial intelligence, and the continuous improvement of accurate prediction techniques, investors and businesses can confidently navigate the complex financial landscape and achieve high success in their investment endeavors.

# **V. REFERENCES**

[1] Everything you need to Know about Linear Regression <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/> (accessed Jun. 15, 2023).

[2] Harris Drucker, Chris J.C.Burges, Linda Kaufman, Alex Smola, Vladimir Vapoik: “Support Vector Regression Machines”

[3] " Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin "A Practical Guide to Support Vector Classification (2010)

[4] Jacek Biesiada và Adam Wierzbicki "k-Nearest Neighbor Algorithm for Real-Time Continuous Data Stream Regression" (2008)

[5] "Local k-Nearest Neighbors Regression" của S. Shekhar và S. Chawla (1998)

[6] Hidden Markov Models with Python <https://medium.com/@natsunoyuki/hidden-markov-models-with-python-c026f778dfa7> (accessed Jun. 18, 2023).

[7] Hidden Markov Model in Machine learning <https://www.geeksforgeeks.org/hidden-markov-model-in-machine-learning/> (accessed Jun. 19, 2023).

[8] Autoregressive Integrated Moving Average (ARIMA) Prediction Model <https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp#toc-how-to-build-an-arima-model> (accessed Jun. 14, 2023).

[9] What Is an ARIMAX Model? <https://365datascience.com/tutorials/python-tutorials/arimax/> (accessed Jun. 19, 2023).

[10] Yann LeCun, Yoshua Bengio và Geoffrey Hinton "Deep Learning" (2015)

[11] Pramit Saha and Dipti Prasad Mukherjee "A Comparative Analysis of Activation Functions for Deep Neural Networks" (2018).

[12] "Long Short-Term Memory" Sepp Hochreiter và Jürgen Schmidhuber (1997)

[13] Introduction to Gated Recurrent Unit (GRU)? <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/> (accessed Jun. 19, 2023).

[14] Introduction to Gated Recurrent Unit (GRU)? <https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/#:~:text=One%20can%20choose%20LSTM%20if,consumption%20and%20want%20faster%20results.> (accessed Jun. 20, 2023).

[15] The Ultimate Guide to Building Your Own LSTM Models: https://www.projectpro.io/article/lstm-model/832