Using Statistical Model And Machine Learning To Predict Gold, Silver And Platinum Daily Price

Nguyen T. Nhan
IS403.N22.HTCL
University of Information Technology
19521940@m.uit.edu.vn

Pham P. Tuan
IS403.N22.HTCL
University of Information Technology
20522125@gm.uit.edu.vn

Nguyen Q. Vinh
IS403.N22.HTCL
University of Information Technology
20522160@gm.uit.edu.vn

***Abstract* —** The high value and volatility of gold, silver, and platinum in the market have made them a prime target for investors and traders. To predict future price movements with reasonable accuracy, statistical models and machine learning algorithms are increasingly being used by analyzing historical data and identifying patterns. In this research, we use Gold, Platinum, and Silver daily datasets from December 2017 to June 2023 to make predictions using various statistical and machine learning models. By doing so, investors can make informed decisions about when to buy or sell these precious metals. The models that we will be using include ETS, HMM, RNN with Attention, RNN, KNN, Linear Regression, ARIMA, GRU and LSTM. Consequently, we can see that the **RNN** performed the best with the lowest MAPE **0.72** in 7 – 3 dataset and **GRU** performed the best with lowest MAPE **0.73** in the 6 – 4 dataset.

Keywords: precious metals, statical models, machine learning algorithms, price prediction

I. INTRODUCTION

One of the major factors that influence the prices of precious metals is supply and demand. The demand for these metals can be influenced by a variety of factors such as economic conditions, geopolitical tensions, inflation, and changes in consumer preferences. Historically, these metals have been viewed as a hedge against inflation and economic uncertainty. As inflation rates rise, investors may turn to precious metals as a way to protect their wealth. This can lead to an increase in demand for these metals, driving up their prices. Conversely, when inflation rates are low, the demand for these metals may decrease, leading to lower prices. Understanding the dynamics of supply and demand is essential for predicting future trends in the market and making informed investment decisions. In this research, we aim to explore the effectiveness of various statistical and machine learning models in predicting the future prices of gold, silver, and platinum.

Several studies have been conducted on predicting the future prices of precious metals using statistical models and machine learning algorithms. For example, the study by Tripathy, N. (2017)[1] used an autoregressive integrated moving average (ARIMA) model to predict the prices of gold. Manjula, K. A. et al. (2019)[2] used an ensemble model combining multiple machine learning algorithms to predict the prices of gold and silver. Another study by Boongasame, L. et al. **(**2022**)**[3] used long short-term memory and the association rule to forecast the gold price. These studies have shown promising results in predicting the future prices of precious metals using statistical models and machine learning algorithms.

The goal of this research is to compare the performance of various statistical and machine learning models in predicting the future prices of gold, silver, and platinum. We will use Gold, Platinum, and Silver daily datasets from December 2017 to June 2023 to make predictions using a variety of models such as linear regression, K-Nearest Neighbors, and neural networks. By doing so, we aim to provide insights into which models are most effective in predicting the future prices of these precious metals. This research can be beneficial for investors and traders who want to make informed decisions about buying or selling these metals based on predicted future prices.

II. RELATED WORK

The field of predicting metal prices has seen a range of techniques employed to try and accurately forecast future trends. One approach is to use ETS and ARIMA models, which was explored in a study by Azzutti, A. (2021) [4] for forecasting gold price. With respective average RMSE of 100.26 and 102.5, ARIMA and ETS have outperformed the other forecast models with the lowest RMSE and provided the most accurate forecast for the gold price.

We also found out Ho Thanh Tri's paper[5] about using ARIMA (5,1,5) model to forecast gold price in Viet Nam and the forecast error in this case is about 3.46%.

The performance GRU and LSTM have also been investigated for gold price prediction, as detailed in a research paper by Yurtsever, M. (2010)[6] by predict the gold price using LSTM and GRU, with RMSE of 61.728 and 87.425 respectively. This indicates that the model is good enough to make predictions.

Singh et al. (2018)[7] used a Recurrent Neural Network approach for mineral deposit modelling and achieved promising results with data from 1500 samples for training, 500 samples for validation, and another 500 samples for testing. R2 values for the training, validation, and testing sets were

0.98970, 0.97650, and 0.97210 respectively, while mean squared errors (MSE) for the training, validation, and testing sets were 45.2, 47.1, and 45.2.

Nguyet Nguyen's work[8] predicted daily Technology Stock using Hidden Markov Model to predict prices of three stocks: Apple, Google, and Facebook. They used AIC and BIC to choose numbers of states from HMM. Then they used the models to predict closes prices of these three stocks using both single observation data and multiple observation data. Finally, they used the predictions as signals for trading these stocks. The criteria tests' results showed that HMM with two states worked the best among two, three and four states for the three stocks.

Ranjan GSK's paper[9] demonstrated how to choose value of K in KNN regression using Grid Search CV. In this work, the KNN model was used to detect faults in a machine for industries and Grid Search CV was used to train and optimize the model for the best results. The proposed algorithm had an accuracy in prediction of 80%.

III. MODELING/AI

Our approach to forecasting the prices of silver, gold, and platinum involves a blend of machine learning techniques, which comprise of HMM, RNN with Attention, RNN, KNN, GRU, LSTM, ARIMA, ETS, and Linear Regression.

A. Linear Regression

Linear Regression predicts the value of the dependent variable y based on independent variable x. Coefficients (β) in a linear equation are estimated for each independent variable to make the best prediction for the dependent variable.[10]

The linear equation takes the form:

$$y = \beta_0 + \beta_1 x \text{ [11]}$$

Where,

y: the dependent variable

x: the independent variable

β0: the intercept

β1: estimated coefficients

Linear Regression can also be extended to more than one independent variable, also known as Multiple Linear Regression (MLR). The MLR model takes the form of an equation like this:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \text{ [11]}$$

B. ARIMA

ARIMA is a statistical model that uses time series data to understand the dataset or predict future trends. Autoregressive models assume that the future will resemble the past and predict future prices based on past performance, but it can be inaccurate during certain market conditions. ARIMA analyzes the strength of one dependent variable compared to other changing variables and aims to predict future market moves by examining differences between values in the series.[12]

1) Integrated

$$Differentation\ 1: \Delta y_t = y_t - y_{t-1}$$
$$Differentation\ 2: \Delta y_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$$

Where,

y_t: is the value at time t

y_{t-1}: is the value of the 1 unit of time ago

y_{t-2}: is the value of the 2 units of time ago

2) Auto regression

$$y_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} + \varepsilon_t$$

Where,

y_t: is the value of a dependent variable at time t

a_0: is a constant term

$y_{t-1}, y_{t-2}, y_{t-p}$: the dependent variables in the past

a_1, a_2, a_p: are the coefficients of the lagged values of the dependent variables respectively

ε_t: is the error term at time t

3) Moving average

$$y_t = \beta_0 + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \cdots + \beta_q \varepsilon_{t-q} + \mu_t$$

Where,

y_t: is the value of a dependent variable at time t

β_0: is a constant term

$\varepsilon_{t-1}, \varepsilon_{t-2}, \varepsilon_{t-q}$: the error terms in the past

$\beta_1, \beta_2, \beta_q$: are the coefficients of the lagged values of the error term respectively

μ_t: is the current error term or residual at time t

4) ARIMA

$$ARIMA(p,d,q) = AR(p) + I(d) + MA(q) \text{[13]}$$

Where,

p, d, q: are non-negative integers representing the order of autoregression, differencing, and moving average respectively.

$AR(p)$: Autoregression of order p refers to the regression of the variable on its own lagged values up to p lags. The AR(p) component captures the linear dependence between the current value and its past values.

$I(d)$: Integration of order d refers to differing the time series d times until it becomes stationary. The I(d) component removes the trend from the time series.

$MA(q)$: Moving average of order q refers to the regression of the variable on the past errors up to q lags. The MA(q) component captures the linear dependence between the current value and its past errors.

C. ETS

Exponential smoothing is a time series forecasting method that uses a weighted average of past observations. The weights decrease exponentially as the observation gets older, hence the name "exponential" smoothing. It assumes that the future will be similar to the recent past. The only pattern that ETS learns from demand history is its level - the average value around which the demand varies over time.[14]

The component form of Simple Exponential Smoothing is given by:

$$F_t = F_{t-1} + \alpha(A_{t-1} - F_{t-1}) \text{ [15]}$$

Where,

F_t: represents the forecast for a given time period t
F_{t-1}: is the previous period's forecast value
A_{t-1}: is the actual value for the previous period
α: is a smoothing parameter that determines the weight given to the most recent observation versus the historical trend.

Simple Exponential Smoothing's predictions deviate from reality due to the presence of an uptrend or downtrend. Another approach is Holt's Linear Exponential Smoothing. It solves this by using two constants, α and β. Alpha is discussed as above, while β reduces the effect of errors between actual and predicted values caused by up and down trends. Which can be known as[16]:

$$F_{t+k} = L_t + k\,T_t$$
$$L_t = \alpha\,Y_t + (1-\alpha)(L_{t-1} + T_{t-1})$$
$$T_t = \beta\,(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

Where,
F_{t+k}: the forcasted value of (t + k) periods ahead
L_t: represents the smoothed value of the level component at time t
k : the number of periods ahead for which we want to forecast
T_t: represents the smoothed value of the trend component at time t
Y_t: represents the actual value of the time series at time t
α: is the smoothing parameter for the level component
β: is the smoothing parameter for the trend component

Holt's Winter Exponential Smoothing can model seasonality, trend, and level components for univariate time series data. Seasonal cycles are patterns in the data that occur over a standard number of observations. This method adds in the gamma (γ) parameter to account for the seasonal component. For this method, we must specify the period for the seasonal cycle. For example, these lengths include the following: weekly (7), monthly (12), or quarterly (3).

Its formula can be expressed as[17]:

$$L_t = \alpha\left(\frac{Y_t}{S_{t-M}}\right) + (1-\alpha)\,(L_{t-1} + T_{t-1}$$
$$T_t = \beta\,(L_t - L_{t-1}) + (1-\beta)\,T_{t-1}$$
$$S_t = \gamma\left(\frac{Y_t}{L_t}\right) + (1-\gamma)\,S_{t-M}$$
$$F_{t+k} = (L_t + k\,T_t) \times S_{t-M+k}$$

Where,
Y_t: is the actual value at time t
L_t: is the level of the series at time t
T_t: is the trend of the series at time t
S_t: is the seasonal component of the series at time t
M: is the number of seasons in a year
α, β, and γ are smoothing parameters between 0 and 1
k: is the number of time steps ahead we want to forecast

D. KNN

K-nearest neighbors (KNN) is a non-parametric, supervised learning classifier that uses proximity to make predictions about the grouping of an individual data point. It's commonly used for classification problems and operates on the assumption that similar points are located near each other. The algorithm finds the K closest data points in the feature space to the new input data point and predicts its class or value based on the majority class or average value of its K nearest neighbors.[18]

The KNN working can be explained based on the below algorithm:

Step 1: Select the number K of the neighbors by using Grid Search, choose from a minimum threshold of 3 and increase it by 1 until the optimal value is found.

Step 2: Calculate the Euclidean distance of K number of neighbors. The Euclidean formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where,
(x_1, y_1): are the coordinates of one point.
(x_2, y_2): are the coordinates of the other point.
d: is the distance between (x_1, y_1) and (x_2, y_2).

Step 3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step 4: Among these K neighbors, count the number of data points in each class.

Step 5: Make the prediction by assigning new data points to the class for which the number of the neighbor is maximum.[19]

E. RNN

RNN (Recurrent Neural Network) is a type of Neural Network that can be used to model sequence data by using a Hidden Layer. The Hidden state of RNN remembers information about a sequence, acting as a Memory State for previous inputs to the network. RNN uses the same parameters for each input and hidden layer, reducing parameter complexity compared to other neural networks.

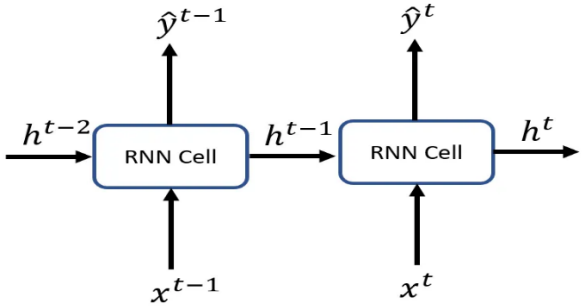

Figure 1. Unfold RNN Model

The Recurrent Neural Network consists of multiple fixed activation function units, one for each time step. Each unit has an internal state which is called the hidden state of the unit. This hidden state signifies the past knowledge that the network currently holds at a given time step. This hidden state is updated

at every time step to signify the change in the knowledge of the network about the past. The hidden state is updated using the following recurrence relation[20]:

$$h_t = f(h_{t-1}, x_t)$$

Where,
h_t : is the current state
h_{t-1} : is the previous state
x_t : is the input at the current timestamp

Formula for applying Activation function (tanh):

$$\tanh(x) = \frac{e^x - e^{x-1}}{e^x + e^{x-1}}$$

$$h_t = tanh(W_{hh}\, h_{t-1} + W_{hx}\, x_t)$$

Where,
W_{hh} : is the weight of recurrent neuron
W_{hx} : is the weight of input neuron

The formula for calculating output:

$$y_t = W_{hy}\, h_t$$

Where,
y_t : is the output
W_{hy} : is the weight of the output layer

F. LSTM

LSTM (Long Short-Term Memory) is a deep learning, sequential neural network that overcomes the vanishing gradient problem faced by RNN. It allows information to persist and works similarly to an RNN cell.

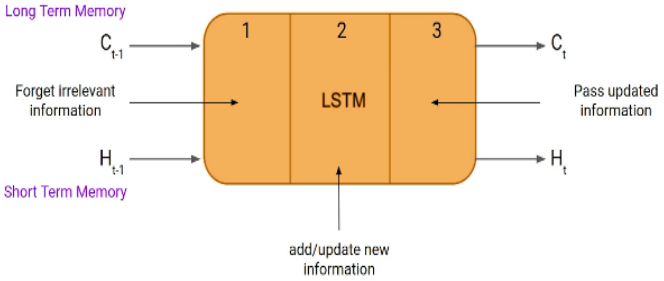

Figure 2. LSTM Architecture

An LSTM has a hidden state for short-term memory and a cell state for long-term memory. The cell state remembers important information over time while filtering out irrelevant data, allowing LSTMs to make accurate predictions.

Figure 3. Long and Short Term Memory

LSTM has three gates that control the flow of information in and out of the memory cell: Forget, Input, and Output.

First, the Forget gate chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten.

$$\sigma = \frac{1}{1 + e^{-x}}$$

$$f_t = \sigma(W_{hh}{}^{(f)}\, h_{t-1} + W_{hx}{}^{(f)}\, x_t)$$

It determines how much information from the previous cell state C_{t-1} should be retained for the current time step. It does this by taking a weighted sum of the previous hidden state h_{t-1} and current input x_t, with weights given by matrices $W_{hh}{}^{(f)}$ and $W_{hx}{}^{(f)}$, respectively.

The result is passed through a sigmoid activation function σ, which squashes the values between 0 and 1. A value of 0 means that all information from the previous cell state will be forgotten, while a value of 1 means that all information will be retained.

Next, the Input gate tries to learn new information from the input to this cell and determines how much to take from the intermediate cell state.

$$i_t = \sigma(W_{hh}{}^{(i)}\, h_{t-1} + W_{hx}{}^{(i)}\, x_t)$$

In contract, the Input gate identifies how much information from the current input x_t and previous hidden state h_{t-1} should be used to update the cell state C_t, with weights given by matrices $W_{hh}{}^{(i)}$ and $W_{hx}{}^{(i)}$, respectively.

Again, we have applied the sigmoid function over it. As a result, the value of i_t will be between 0 and 1.

$$\widetilde{C_t} = tanh(W_{hh}{}^{(c)}\, h_{t-1} + W_{hx}{}^{(c)}\, x_t)$$

LSTM may not take everything from the intermediate cell state, the $\widetilde{C_t}$ is there to distinguish between different weight matrices. Due to the $tanh$ function, the value of new information will be between -1 and 1. If the value of $\widetilde{C_t}$ is negative, the information is subtracted from the cell state, and if the value is positive, the information is added to the cell state at the current timestamp.

However, the $\widetilde{C_t}$ won't be added directly to the cell state. Here comes the updated equation:

$$C_t = (f_t\ C_{t-1} + i_t\ \widetilde{C_t})$$

The Output gate passes the updated information from the current timestamp to the next timestamp. Together, they function as a layer of neurons with hidden layers and a current state.

$$o_t = \sigma(W_{hh}{}^{(o)}\, h_{t-1} + W_{hx}{}^{(o)}\, x_t)$$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state, we will use o_t and tanh of the updated cell state. [21]

$$h_t = o_t * \tanh(C_t)$$

G. RNN – Attention

In Sequence to Sequence Learning, RNN is trained to map an input sequence to an output sequence which is not necessarily of the same length.

The Encoder RNN reads the input sequence and generates the fixed-size context vector which represents a semantic summary of the input sequence. The fixed-size context vector is given as input to the decoder RNN. The fixed-size context can be provided as the initial state of the Decoder RNN, or it can be connected to the hidden units at each time step. These two ways can also be combined.

However, this approach may limit the decoder's ability to extract information from the vector. Therefore, an attention mechanism was added to address this limitation.[22]

With the idea of using a context vector that can interact with the entire hidden state vector of the encoder instead of just using the final hidden state vector to generate the representation vector for the decoder. Example, the seq2seq model when applying the attention mechanism will have the following structure (blue blocks are encoder, red is decoder):

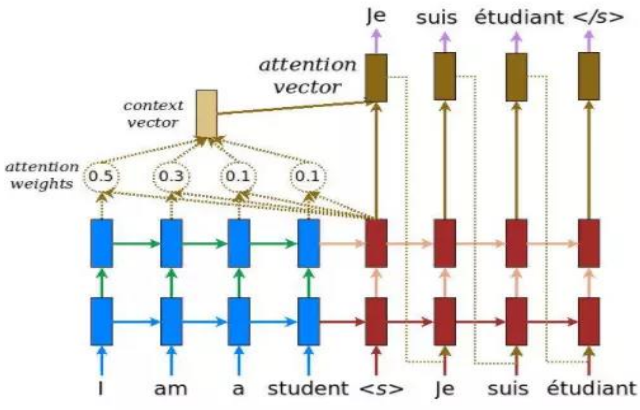

Figure 4. S2S RNN with Attention

The main idea of the mechanism is a network that learns at each time-step which part of the input is important. A model with the attention mechanism can be trained end-to-end without specifying which values to choose - the model will learn to do that itself.

H. GRU

GRU (Gated Recurrent Unit) is a type of recurrent neural network that addresses the vanishing gradient problem. It is similar to LSTM and can produce comparable results.

To solve the vanishing gradient problem of a standard RNN, GRU uses, Update gate and Reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or removing information which is irrelevant to the prediction.

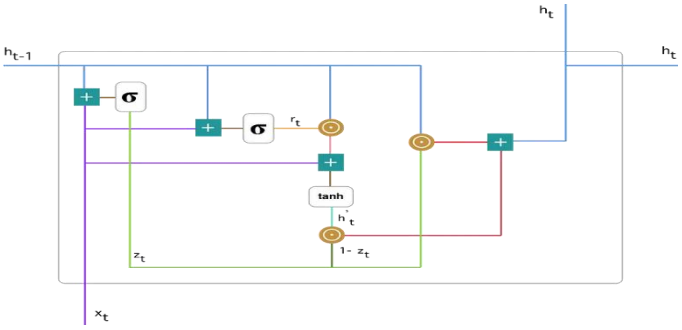

Figure 5. GRU Model

The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future and avoid the vanishing gradient problem. The formula using for Update gate:

$$z_t = \sigma(W_{hh}^{(z)}\, h_{t-1} + W_{hx}^{(z)}\, x_t)$$

When x_t is plugged into the network unit, it is multiplied by its own weight $W_{hx}^{(z)}$. The same goes for h_{t-1} which holds the information for the previous t - 1 units and is multiplied by its own weight $W_{hh}^{(z)}$. Both results are added together, and a sigmoid activation function is applied to squash the result between 0 and 1.

Reset gate is used from the model to decide how much of the past information to forget. To calculate it, we use:

$$r_t = \sigma(W_{hh}^{(r)}\, h_{t-1} + W_{hx}^{(r)}\, x_t)$$

This formula is the same as the one for the update gate. We plug in h_{t-1} — blue line and x_t — purple line, multiply them with their corresponding weights, sum the results and apply the sigmoid function. The difference comes in the weights and the gate's usage.

To determine what to remove from the previous time steps, we will calculate current memory content by using reset gate to store relevant information from the past.

$$h'_t = tanh(r_t * W_{hh}\, h_{t-1} + W_{hx}\, x_t)$$

To process input x_t and h_{t-1}, we multiply them by weights W_{hx} and W_{hh} respectively. The reset gate r_t is used to determine what information to remove from previous time steps by calculating the Hadamard product with $(W_{hh}\, h_{t-1})$. The results of steps 1 and 2 are summed up and passed through the tanh activation function.

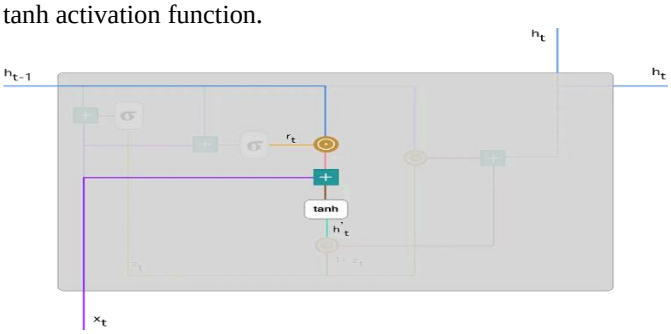

Figure 6. Current memory content

Finally, the network needs to calculate h_t — vector which holds information for the current unit and passes it down to the network. In order to do that the update gate is needed. It determines what to collect from the current memory content — h'_t and what from the previous steps — h_{t-1}. That is done as follows:

$$h_t = z_t * h_{t-1} + (1 - z_t) * h'_t$$

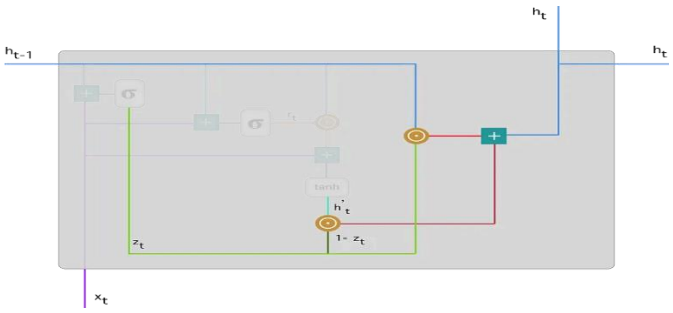

Figure 7. Final memory content

The model can learn to set the vector z_t close to 1 and keep a majority of the previous information by apply element-wise multiplication to the update gate z_t and h_{t-1}. Since z_t will be close to 1 at this time step, $(1 - z_t)$ will be close to 0 and ignore big portion of the current content, which is irrelevant for our prediction by apply element-wise multiplication to $(1 - z_t)$ and h'_t . Final, we sum the results.[23]

I. HMM

HMM (Hidden Markov Model) is a statistical model that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states. It is often used in situations where the system or process that generates the observations is unknown or hidden. It's often used for predicting future observations or classifying sequences, based on the underlying hidden process that generates the data.

It consists of two types of variables: hidden states and observations. The hidden states are the variables that generate the observed data, but they are not directly observable. The observations are the variables that are measured and observed.

The relationship between the hidden states and the observations is modeled using a probability distribution with two sets: the transition probabilities and the emission probabilities. The transition probabilities describe the probability of transitioning from one hidden state to another. The emission probabilities describe the probability of observing an output given a hidden state. [24]

HMM 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.

$$\pi = [\pi_0, \pi_1, \pi_2, \dots]^T$$

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 x_i given some hidden state z_i.

$$\theta = [\theta_0, \theta_1, \theta_2, \dots]^T$$

The three typical classes of problems which can be solved using hidden Markov models are:

Problem 1 (Likelihood): For a given set of model parameters $\lambda = (\pi, A, \theta)$ and a sequence of observations X, calculate $P(X|\lambda)$. This problem is solved using the forward algorithm.

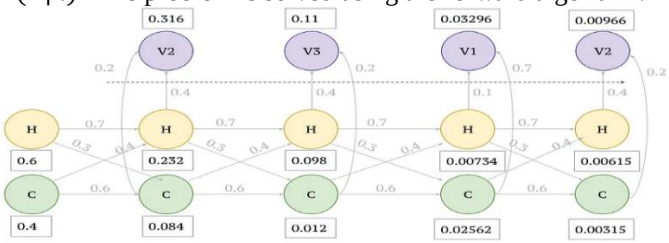

Figure 8. Likelihood problem

Problem 2 (Decoding): For a given set of model parameters $\lambda = (\pi, A, \theta)$ and a sequence of observations X, calculate the maximum a posteriori probability estimates of the most likely Z. This problem is solved using the Viterbi algorithm.

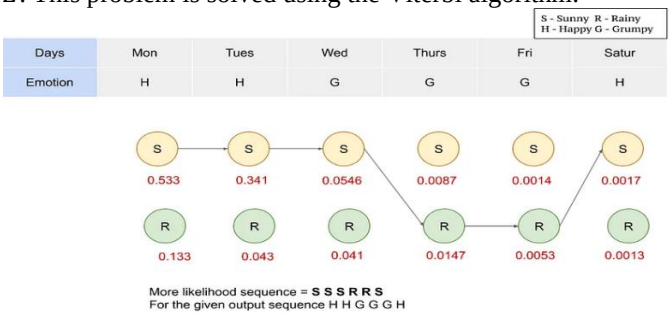

Figure 9. Decoding problem

Problem 3 (Learning): For a sequence of observations X, guess an initial set of model parameters $\lambda = (\pi, A, \theta)$ and use the forward and Viterbi algorithms iteratively to recompute $P(X|\lambda)$ as well as to readjust λ. The calculations stop when $P(X|\lambda)$ stops increasing, or after a set number of iterations. This problem is solved using the Baum-Welch algorithm.

IV. METHOD

In order to perform prediction in machine learning, it is essential to ensure that the input data is properly formatted and preprocessed.

Step 1: Data Preprocessing

First, we have to clean and preprocess the collected data by removing duplicates, handling missing values, and transforming the data into a format that can be used by machine learning algorithms. This may also involve feature engineering, where you create new features from existing ones to improve model performance.

Step 2: Data Splitting

Splitting the preprocessed data into training, testing and validate sets. The training set is used to train the machine learning model, while the testing set is used to evaluate the performance of the trained model. The validation set is used to evaluate the model prediction using the last set of the data to assets the model precision in the future.

Step 3: Model Training

Train the selected machine learning model using the training data. This involves feeding the data into the model, adjusting the model parameters, and evaluating the model performance.

Step 4: Model evaluation

Evaluate the performance of the trained model using testing and validate data. This can be done by calculating various metrics such as accuracy, precision, recall, and F1 score.

Step 5: Make predictions

Once the model has been trained and evaluated, it can be used to make predictions in the next 30 days.

V. IMPLEMENTATION

A. Dataset

1) Description

The dataset includes daily prices for gold, silver and platinum from December 1st, 2017, to June 10th, 2023. Each row represents a single day and contains the following information:

- Date: The date of the trading day.
- Price: The closing price of the metal on that day.
- Open: The opening price of the metal on that day.
- High: The highest price that the metal reached during the trading day.
- Low: The lowest price that the metal reached during the trading day.
- Vol.: The volume of the metal that was traded during the day.
- Change %: The percentage change in price from the previous day's closing price.

The dataset was collected from investing.com[25] and has been preprocessed to remove any missing or erroneous data points.

2) Descriptive Statistical

a) Gold data

	Price	Open	High	Low
Mean	1632.738357	1633.000874	1644.131259	1621.049336
Standard Error	6.759192784	6.765594039	6.845279686	6.667887499
Median	1729	1732.1	1743.35	1718.725
Mode	1814.1	1817	1812	1939.1
Standard Deviation	255.6011787	255.8432443	258.8565842	252.1484382
Sample Variance	65331.96255	65455.76567	67006.73118	63578.83489
Minimum	1184	1180.7	1189	1167.1
Maximum	2069.4	2076.4	2089.2	2049
Sum	2334815.85	2335191.25	2351107.7	2318100.55
Count	1430	1430	1430	1430

Figure 10. Descriptive statistics for gold data

b) Silver data

	Price	Open	High	Low
Mean	20.13270014	20.14578713	20.39096605	19.86692504
Standard Error	0.113431266	0.113663791	0.116408007	0.110165433
Median	19.1205	19.095	19.375	18.7825
Mode	14.18	16.72	16.565	14.54
Standard Deviation	4.265377604	4.274121287	4.377312578	4.14257187
Sample Variance	18.1934461	18.26811278	19.1608654	17.1609017
Minimum	11.772	11.975	12.33	11.64
Maximum	29.418	29.235	30.35	28.155
Sum	28467.638	28486.143	28832.826	28091.832
Count	1414	1414	1414	1414

Figure 11. Descriptive statistics for silver data

c) Platinum data

	Price	Open	High	Low
Mean	945.5853027	945.4471769	956.5855355	934.0923458
Standard Error	2.710385967	2.705991039	2.756120841	2.669819487
Median	934.05	934.425	944.45	925
Mode	973.6	937.3	1012	1072
Standard Deviation	112.3421471	112.1599827	114.2378011	110.6607167
Sample Variance	12620.75801	12579.86173	13050.27519	12245.79421
Minimum	595.2	596.05	615.4	562
Maximum	1318.75	1292.8	1351.2	1283.15
Sum	1624515.55	1624278.25	1643413.95	1604770.65
Count	1718	1718	1718	1718

Figure 12. Descriptive statistics for platinum data

3) Visualization

a) Gold data

Figure 13. Visualize gold data

b) Silver data

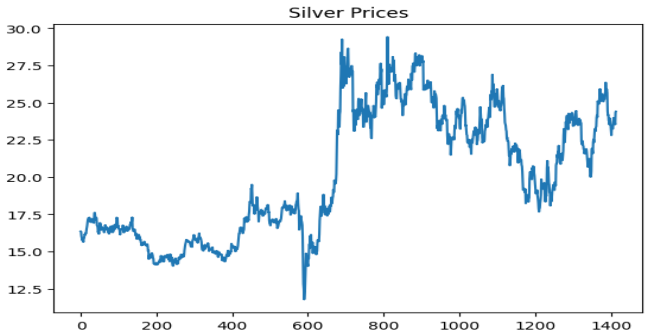

Figure 14. Visualize silver data

c) Platinum data

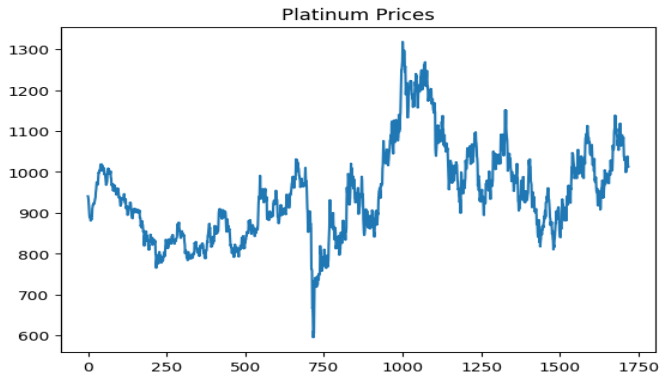

Figure 15. Visualize platinum data

B. Technology

In this paper, Google Collab will be used for data analysis. It's a free cloud-based platform that offers powerful hardware, seamless integration with Google Drive, collaboration through shared notebooks, making it a user-friendly choice.

Library used: Pandas, mathplotlib, Tensorflow, keras, sklearn, statsmodels…

C. Spliting data

We will split the dataset into 3 ratios (Train: Test: Validate): 7:2:1, 6:3:1, 5:3:2. To build a better machine learning model that generalizes well on new data and avoids overfitting.

D. Evaluation

Evaluating a model is crucial to ensure that it is accurate, reliable, and generalizes well to new data. The following metrics will be used: MAE, MAPE and RMSE.

1) MAE

MAE (Mean Absolute Error) is a measure of the average size of the mistakes in a collection of predictions, without taking their direction into account. It is measured as the average absolute difference between the predicted values and the actual values and is used to assess the effectiveness of a model.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|A_t - F_t|$$

Where,

n: is the number of observations in the dataset

A_t: is the actual value at time t

F_t: is the predicted value at time t

2) MAPE

MAPE (Mean absolute percentage error) is a metric that defines the accuracy of a forecasting method. It represents the average of the absolute percentage errors of each entry in a dataset to calculate how accurate the forecasted quantities were in comparison with the actual quantities. MAPE is often effective for analyzing large sets of data and requires the use of dataset values other than zero.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

3) RMSE

The root mean square error (RMSE) measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals. Residuals represent the distance between the regression line and the data points.

$$RMSE = \sqrt{\sum_{t=1}^{n}\frac{(F_t - A_t)^2}{n}}$$

E. Result

1) Gold data

Gold							
Model	Ratio	Testing			Validation		
		RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE
LN	7-2-1	270.6	13.44	236.44	302.49	15.68	296.84
	6-3-1	302.6	14.9	263.84	410.76	21.43	407.04
	5-3-2	413.48	7.04	128.59	388.89	21.04	380.28
ARIMA	7-2-1	94.86	4.16	76.47	167.35	7.51	146.7
	6-3-1	115.9	5.25	96.98	213.16	10.14	197.08
	5-3-2	353.73	18.01	327.81	274.8	14.57	262.47
ETS	7-2-1	63.1	2.59	45.47	122.19	5.34	103.82
	6-3-1	2.41	4.35	78.35	203.62	8.24	161.67
	5-3-2	123.52	5.56	101.78	183.75	8.58	155.46
GRU	7-2-1	17.57	0.73	13.27	17.38	0.73	13.93
	6-3-1	23.98	1.43	25.78	31.6	1.44	27.56
	5-3-2	21.92	0.96	17.73	21.36	0.91	16.84
LSTM	7-2-1	18.15	0.77	14.03	20.77	0.85	16.32
	6-3-1	33.21	1.6	28.94	33.39	1.49	28.56
	5-3-2	19.42	0.79	14.62	19.67	0.82	15.07
KNN	7-2-1	8.99	0.39	7	9.45	0.37	7.08
	6-3-1	9.57	0.41	7.39	15.4	0.59	11.44
	5-3-2	16.35	0.66	12.15	12.25	0.51	9.39
RNN	7-2-1	17.27	0.72	13.09	18.76	0.74	14.17
	6-3-1	19.14	0.81	14.71	19.16	0.75	14.47
	5-3-2	19.45	0.75	13.8	18.17	0.75	13.82
RNN Attention	7-2-1	19.16	0.81	14.47	22.21	0.92	17.75
	6-3-1	30.43	1.32	24.1	37.93	1.58	30.51
	5-3-2	28.95	1.18	21.62	28.26	1.19	21.74
HMM	7-2-1	542.77	29.38	533.36	361.22	17.61	332.78
	6-3-1	163.96	7.4	131.7	133.47	5.46	105.6
	5-3-2	305.16	15.67	288.15	181.09	8.72	154.39

Figure 16. Evaluate gold data

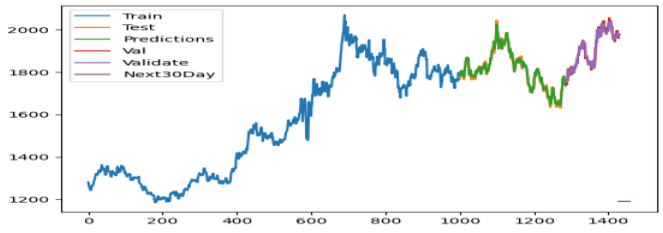

Figure 17. KNN visualization for gold data

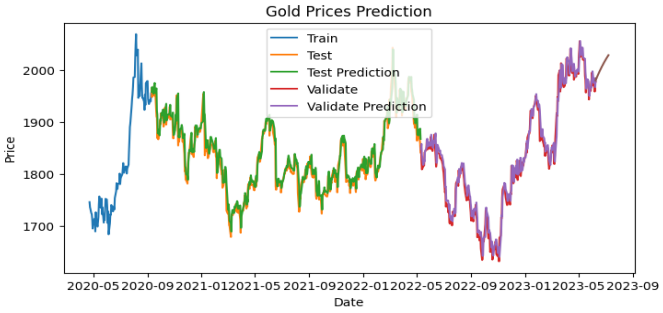

Figure 18. RNN visualization for gold data

Figure 19. GRU visualization for gold data

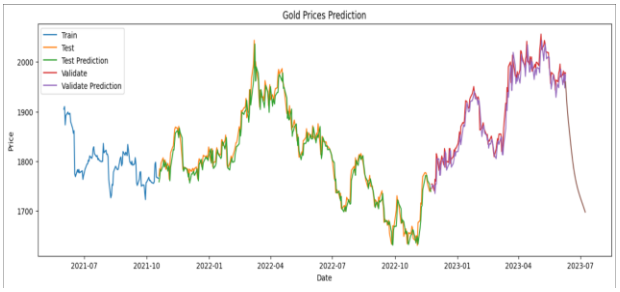

Figure 20. LSTM visualization for gold data

2) Silver data

Silver							
Model	Ratio	Testing			Validation		
		RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE
LN	7-2-1	6.07	25.43	5.22	6.62	28.12	6.48
	6-3-1	4.65	17.53	3.73	5.70	24.04	5.53
	5-3-2	5.60	20.65	5.24	1.90	7.43	1.61
ARIMA	7-2-1	2.66	10.63	2.18	1.42	4.87	1.12
	6-3-1	3.36	12.72	2.69	2.25	8.27	1.85
	5-3-2	28.09	102.51	24.97	22.18	96.44	21.11
ETS	7-2-1	5.21	21.02	4.30	1.91	6.47	1.50
	6-3-1	4.09	16.06	3.63	5.07	20.59	4.72
	5-3-2	8.30	32.17	7.85	11.49	52.93	11.22
GRU	7-2-1	0.44	1.55	0.34	0.42	1.37	0.32
	6-3-1	0.44	1.45	0.33	0.42	1.40	0.32
	5-3-2	0.57	1.67	0.42	0.47	1.65	0.36
LSTM	7-2-1	0.54	2.02	0.44	0.51	1.79	0.41
	6-3-1	0.48	1.63	0.37	0.47	1.61	0.37
	5-3-2	0.96	3.37	0.83	0.73	2.87	0.63
KNN	7-2-1	0.29	1.04	0.22	0.31	1.03	0.23
	6-3-1	0.43	1.34	0.30	0.43	1.37	0.31
	5-3-2	1.45	0.36	0.48	1.74	0.38	0.51
RNN	7-2-1	0.42	1.47	0.32	0.42	1.35	0.32
	6-3-1	0.42	1.38	0.32	0.41	1.35	0.31
	5-3-2	0.55	1.56	0.39	0.43	1.53	0.33
RNN Attention	7-2-1	0.53	1.83	0.40	0.57	1.93	0.45
	6-3-1	0.91	2.76	0.68	0.73	2.70	0.58
	5-3-2	0.91	2.76	0.68	0.73	2.70	0.58
HMM	7-2-1	4.84	19.54	4.18	3.9	15.2	3.46
	6-3-1	8.51	33.87	8.03	7.42	30.45	7.19
	5-3-2	2.62	8.46	2.06	5	20.89	4.29

Figure 21. Evaluate silver data

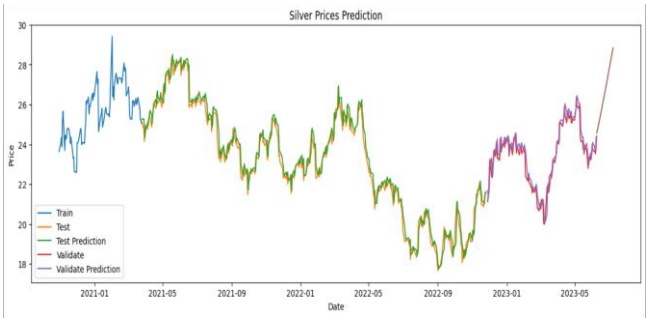

Figure 22. GRU visualization for silver data

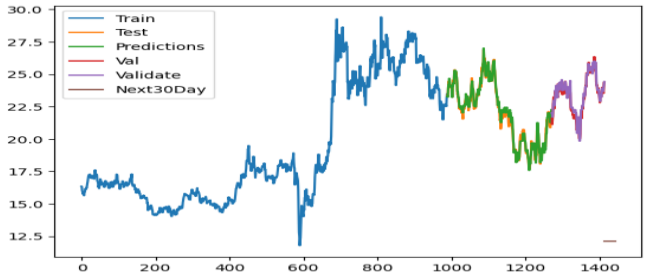

Figure 23. KNN visualization for silver data

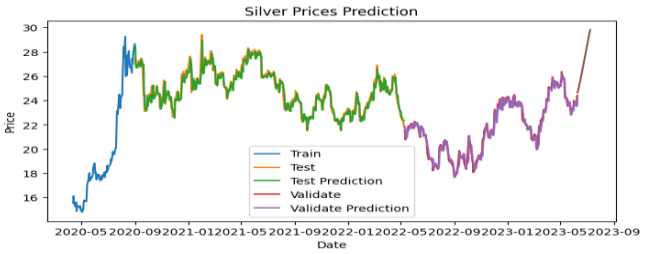

Figure 24. RNN visualization for silver data

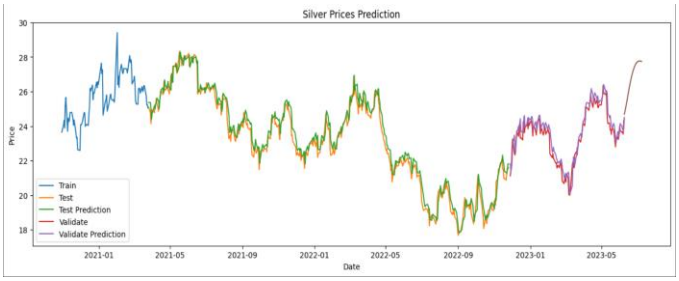

Figure 25. LSTM visualization for silver data

3) Platinum data

Platinum							
Model	Ratio	Testing			Validation		
		RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE
LN	7-2-1	139.79	12.86	117.79	111.98	10.1	100.67
	6-3-1	119.49	9.07	93.52	49.79	4.05	41.06
	5-3-2	222.2	17.81	195.05	153.92	13.34	134.28
ARIMA	7-2-1	92.16	8.27	75.91	50.43	4.07	41.49
	6-3-1	207.62	19.5	185.49	173.03	16.47	165.52
	5-3-2	138.92	9.8	108.86	75.44	6.5	63.2
ETS	7-2-1	123.9	10.7	98.04	86.63	7.41	75.2
	6-3-1	282.65	26.91	258.86	259.28	24.5	246.85
	5-3-2	104.53	7.65	83.51	90.4	7.92	74.21
GRU	7-2-1	18.25	1.45	13.95	17.64	1.33	13.59
	6-3-1	18.17	1.37	13.78	17.33	1.3	13.29
	5-3-2	25.79	1.99	21.1	22.38	1.84	17.92
LSTM	7-2-1	19.96	1.63	15.19	20.04	1.55	15.86
	6-3-1	18.25	1.4	14.06	17.72	1.35	13.73
	5-3-2	24.4	1.75	18.63	19.39	1.55	15.1
KNN	7-2-1	7.63	0.6	5.84	8.5	0.67	6.91
	6-3-1	8.8	0.65	6.64	9.29	0.7	7.25
	5-3-2	93.21	4.95	57.66	56.34	1.46	15.18
RNN	7-2-1	18	1.43	13.79	17.43	1.31	13.39
	6-3-1	17.89	1.37	13.68	17.16	1.28	13.1
	5-3-2	23.62	1.77	18.82	19.5	1.56	15.26
RNN Attention	7-2-1	18.72	1.52	14.64	18.16	1.4	14.25
	6-3-1	26.16	2.02	20.54	23.72	1.8	18.52
	5-3-2	29.56	2.22	23.12	25.03	2.09	20.19
HMM	7-2-1	131.41	11	109.67	214.08	19.1	197.98
	6-3-1	254.04	24.82	242.87	204.98	19.6	197.54
	5-3-2	149.08	10.62	117.05	147.64	11.88	119.53

Figure 26. Evaluate platinum data

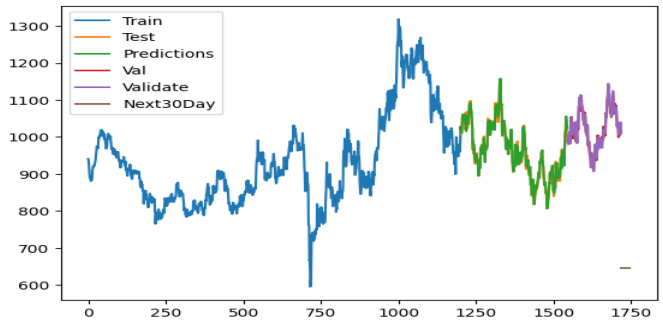

Figure 27. KNN visualization for platinum data

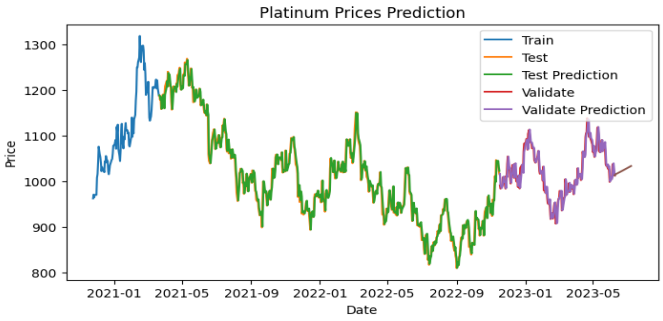

Figure 28. RNN visualization for platinum data

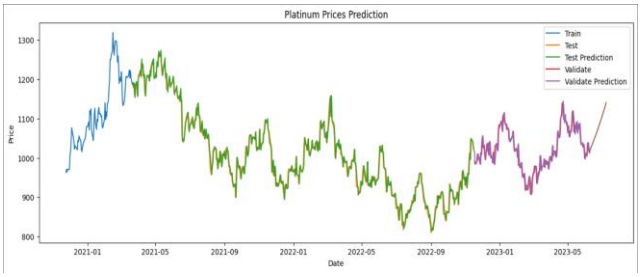

Figure 29. GRU visualization for platinum data

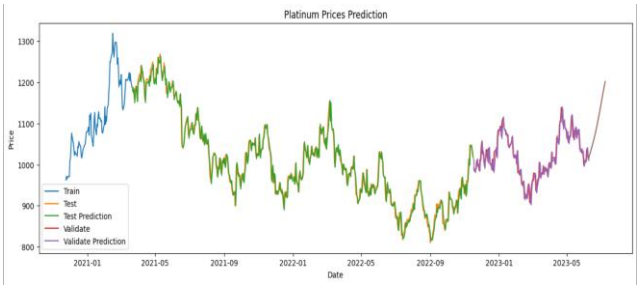

Figure 30. LSTM visualization for platinum data

4) Remark

Based on the result of evaluating 9 models, we have GRU, LSTM, KNN and RNN appear to be the best performing models based on their low RMSE, MAE, and MAPE values for both testing and validation sets. The KNN model had the lowest RMSE and MAE values among all models for the testing set, indicating that it had the smallest average error in predicting actual values.

However, in the long run, KNN model cannot perform well when predicting for longer periods, such as 30 days, due to its reliance on nearest neighbors in the training data. As the prediction horizon increases, there may not be enough historical observations available to accurately predict future values.

On the other hand, LSTM, GRU, RNN models have been shown to perform better in predicting longer horizons due to their ability to capture temporal dependencies and patterns in the input data. The recurrent connections in the RNN allow the model to remember past information and use it to make predictions for future time steps. These models had relatively low RMSE and MAE values for both testing and validation sets.

VI. Conclusion

In conclusion, the use of statistical models and machine learning algorithms has proven to be an effective method for predicting daily prices of gold, silver, and platinum. The results obtained from this study show that **LSTM, GRU and RNN models** can provide accurate predictions of precious metal prices based on historical data and other relevant factors.

Though, predicting financial markets is a challenging task for our group and requires continuous monitoring, updating models to account for new trends and changes in market conditions. The toughness of this problem cannot be overstated, but with the right approach and tools, we have overcome this problem perfectly.

In future, there is much potential for our team to do further research in this area. As more data becomes available, it will be possible to develop even more sophisticated models that can capture subtle patterns and relationships in the market. Additionally, advances in computing power and machine learning techniques will continue to enhance the accuracy and efficiency of these models. Overall, the future looks bright for those working to improve the prediction of precious metal prices using statistical modeling and machine learning.

VII. Acknowledgements

This paper and the research behind it would not have been possible without the exceptional support of our supervisor, **Assoc. Prof. Dr. Nguyễn Đình Thuân**. His enthusiasm, knowledge and exacting attention to detail have been an inspiration for the final draft of this paper. We are grateful for his time, effort, and dedication in helping us to achieve our goals, and we could not have completed this work without his unwavering support and encouragement.

We would also like to extend my heartfelt thanks to **Mr. Nguyễn Minh Nhựt** for his invaluable contribution to this paper. His expertise in the technical aspects of this work has been invaluable, and his insights and feedback have helped us to refine our ideas and arguments. He has been a patient and supportive mentor, always willing to share his knowledge and experience, and we are grateful for his guidance and advice throughout this process.

Finally, we would like to express my gratitude for the invaluable support and assistance provided by **Ms. Nguyễn Thị Viết Hương** throughout the development of this paper. She has been a dependable and helpful mentor, readily offering feedback and suggestions that have been instrumental in shaping the direction and content of this work.

REFERENCES

[1] N. Tripathy, "Forecasting Gold Price with Auto Regressive Integrated Moving Average Model," *Int. J. Econ. Financ. Issues*, vol. 7, pp. 324–329, Aug. 2017.

[2] K. A. Manjula and P. Karthikeyan, "Gold Price Prediction using Ensemble based Machine Learning Techniques," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India: IEEE, Apr. 2019, pp. 1360–1364. doi: 10.1109/ICOEI.2019.8862557.

[3] L. Boongasame, P. Viriyaphol, K. Tassanavipas, and P. Temdee, "Gold-Price Forecasting Method Using Long Short-Term Memory and the Association Rule," *J. Mob. Multimed.*, pp. 165–186, 2023, doi: 10.13052/jmm1550-4646.1919.

[4] A. Azzutti, "Forecasting Gold Price: A Comparative Study," Feb. 2016. doi: 10.13140/RG.2.1.4206.5686.

[5] "(PDF) Application of ARIMA Model to Forecast Gold Price in Vietnam." https://www.researchgate.net/publication/324226832_Application_of_ARIMA_Model_to_Forecast_Gold_Price_in_Vietnam (accessed Jun. 21, 2023).

[6] M. Yurtsever, "Gold Price Forecasting Using LSTM, Bi-LSTM and GRU," *Eur. J. Sci. Technol.*, Dec. 2021, doi: 10.31590/ejosat.959405.

[7] R. K. Singh, D. Ray, and B. C. Sarkar, "Recurrent neural network approach to mineral deposit modelling," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad: IEEE, Mar. 2018, pp. 1–5. doi: 10.1109/RAIT.2018.8389063.

[8] "Risks | Free Full-Text | An Analysis and Implementation of the Hidden Markov Model to Technology Stock Prediction." https://www.mdpi.com/2227-9091/5/4/62 (accessed Jun. 21, 2023).

[9] G. S. K. Ranjan, A. Verma, and R. Sudha, "K-Nearest Neighbors and Grid Search CV Based Real Time Fault Monitoring System for Industries," Mar. 2019, pp. 1–5. doi: 10.1109/I2CT45611.2019.9033691.

[10] "About Linear Regression | IBM." https://www.ibm.com/topics/linear-regression (accessed Jun. 21, 2023).

[11] "Linear regression - Wikipedia." https://en.wikipedia.org/wiki/Linear_regression (accessed Jun. 21, 2023).

[12] "Autoregressive Integrated Moving Average (ARIMA) Prediction Model," *Investopedia*. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp (accessed May 18, 2023).

[13] "Khoa học dữ liệu." https://phamdinhkhanh.github.io/2019/12/12/ARIMAmodel.html (accessed Jun. 21, 2023).

[14] F. Sohil, M. U. Sohali, and J. Shabbir, "An introduction to statistical learning with applications in R: by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, $41.98, eISBN: 978-1-4614-7137-7," *Stat. Theory Relat. Fields*, vol. 6, no. 1, pp. 87–87, Jan. 2022, doi: 10.1080/24754269.2021.1980261.

[15] T. Stasik, "Trevor Stasik: Blog (Archive 2007-2019): Time-Series Forecasting: Exponential Smoothing Part 1," *Trevor Stasik*, Oct. 05, 2007. https://trevorstasik.blogspot.com/2007/10/time-series-forecasting-exponential.html (accessed May 18, 2023).

[16] "Smoothing 5: Holt's exponential smoothing - YouTube." https://www.youtube.com/watch?v=DUyZl-abnNM (accessed Jun. 21, 2023).

[17] *7.3 Holt-Winters' seasonal method | Forecasting: Principles and Practice (2nd ed)*. Accessed: May 18, 2023. [Online]. Available: https://otexts.com/fpp2/holt-winters.html

[18] "What is the k-nearest neighbors algorithm? | IBM." https://www.ibm.com/topics/knn (accessed Jun. 21, 2023).

[19] "K-Nearest Neighbor (KNN) Algorithm for Machine Learning - Javatpoint." https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning (accessed Jun. 21, 2023).

[20] "Introduction to Recurrent Neural Network - GeeksforGeeks." https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (accessed Jun. 21, 2023).

[21] "LSTM | Introduction to LSTM | Long Short-Term Memory Algorithms." https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/ (accessed Jun. 21, 2023).

[22] P. Radhakrishnan, "Sequence to Sequence Learning," *Medium*, Oct. 10, 2017. https://towardsdatascience.com/sequence-to-sequence-learning-e0709eb9482d (accessed Jun. 21, 2023).

[23] S. Kostadinov, "Understanding GRU Networks," *Medium*, Nov. 10, 2019. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be (accessed Jun. 21, 2023).

[24] "Hidden Markov Model in Machine learning - GeeksforGeeks." https://www.geeksforgeeks.org/hidden-markov-model-in-machine-learning/ (accessed Jun. 21, 2023).

[25] "Investing.com - Stock Market Quotes & Financial News." https://www.investing.com/ (accessed Jun. 21, 2023).