FORECASTING SILVER PRICE TIME SERIES DATA USING DEEP LEARNING

PIYANUCH VIWATBORVORNWONG

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE

IN DATA SCIENCE AND ANALYTICS

KMITL DIGITAL ANALYTICS AND INTELLIGENCE CENTER SCHOOL OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2023

KMITL-2023-SC-M-017-039

Independent Study Title	Forecasting Silver Price Time Series Data Using Deep Learning
Student Name	Piyanuch Viwatborvornwong
Student ID	64605073
Degrees	Master of Science (Data Science and Analytics)
	KMITL-Digital Analytics and Intelligence Center
Year	2023
Independent Study Advisor	Dr.Jiraphat Yokrattanasak

ABSTRACT

Silver, a critical commodity in the global economy, has intricate and dynamic price fluctuations influenced by a multitude of factors including geopolitical events, market sentiment, and investor behavior. Our research explores various models for predicting silver prices, Linear Regression, Ensemble GRU-GRU, Ensemble LSTM-LSTM, GRU, Ensemble LSTM-GRU, LSTM, and XGBoost. The Linear Regression model demonstrated the best result with a MAPE of 1.1566 and RMSE of 0.3304. The Ensemble GRU-GRU model, being the most proficient among deep learning techniques, showed a MAPE of 2.0413 and RMSE of 0.5838. Other models yielded the following outcomes: the Ensemble LSTM-LSTM model had a MAPE of 2.5217 and RMSE of 0.7268, GRU had a MAPE of 2.6663 and RMSE of 0.7212, the Ensemble LSTM-GRU model had a MAPE of 3.0191 and RMSE of 0.7993, LSTM showed a MAPE of 3.5221 and RMSE of 0.8745, and lastly, the XGBoost model displayed a MAPE of 4.1206 and RMSE of 0.9856. Despite the inherent complexity in predicting silver prices due to their volatile nature, advanced techniques like deep learning and machine learning exhibited potential in improving forecast accuracy.

Keywords: Time Series, Silver Price Forecasting, Deep Learning, LSTM, GRU, Ensemble Models, XGBoost

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all the individuals who have contributed to the development and completion of this Individual Study on Forecasting Silver Price Time Series Data Using Deep Learning.

We extend our appreciation to our advisors and mentors for their guidance, valuable insights, and continuous support throughout the research process. Their expertise and feedback have been instrumental in shaping this Individual Study and refining our research.

We are also grateful to the academic community and researchers in the field of data science, time series analysis, and deep learning for their valuable contributions to the body of knowledge. Their groundbreaking work and publications have served as a solid foundation for our research endeavors.

Finally, we wish to express our genuine thanks to our friends, Pongpeera Jaruvijtrattan, Paweena Rouyprasert and Marit Asavamahakul, colleagues, and family for their support, motivation, and understanding during the creation of this research. Additionally, we extend our heartfelt thanks to our valued co-worker who has been instrumental in providing guidance and motivation during this research endeavor.

This research would not have been possible without the collective efforts, knowledge, and contributions of all those mentioned above, and for that, we express our heartfelt appreciation.

Piyanuch Viwatborvornwong

TABLE OF CONTENTS

Page

TABLE OF CONTENTS: Table

TABLE OF CONTENTS: Figure

CHAPTER 1

INTRODUCTION

This research addresses the complex challenge of silver price forecasting by using advanced deep learning models. These models are compared with traditional ones to enhance prediction accuracy. The focus is on short-term forecasts, considering multiple economic indicators from 2017 to 2022. Ultimately, the study aims to advance financial market analysis, offering valuable insights to stakeholders.

1.1 Statement of the problems

Silver holds a significant position in the global economy due to its unique properties and wide-ranging applications in industries, investments, and technology. The accurate prediction of silver prices is crucial for stakeholders across various sectors, as it influences decision-making, risk management, resource allocation, and strategic planning. Precise forecasting empowers stakeholders to make informed judgments and mitigate potential risks associated with future uncertainties. However, predicting silver prices can be challenging due to the diverse factors impacting price fluctuations, such as macroeconomic conditions, geopolitical events, market sentiment, and investor behavior. The arbitrary nature of the market and the presence of non-linear relationships further complicate the task of precise forecasting using traditional models.

The volatility of silver prices adds another layer of complexity to the prediction process. Fluctuations in silver prices can be influenced by factors like market sentiment, speculative trading, and investor behavior, making it challenging to quantify and predict the exact impact of these influences. Short-term price fluctuations may diverge from underlying fundamentals, making it difficult to accurately forecast silver prices using traditional models like ARIMA and GARCH.

To address these challenges, advanced techniques like deep learning, which can capture complex relationships and patterns in data, are required. Deep learning models, such as LSTM, GRU, Ensemble models and machine learning models;

XGBoost, have shown promise in improving the accuracy of silver price predictions. These models can consider multiple factors simultaneously and learn from historical data to make more accurate forecasts.

By leveraging advanced techniques, stakeholders can enhance their understanding of the underlying factors driving silver prices and make more informed decisions in various sectors.

1.2 Objectives

1. To create various deep learning-based models in order to enhance the precision of silver price predictions.
2. To evaluate the performance of traditional models and different deep learning techniques and architectures, such as LSTM, GRU, XGBoost and Ensemble Model, in detecting and comprehending the underlying patterns and relationships present within the data.

1.3 Scope of Study

1. The proposed scope of the study involves the study of short-term forecasts, specifically those that are related to a short-term forecast (a one-day duration).
2. The proposed scope of the study is to focus on the daily fluctuations of silver prices, gold prices, oil prices, the exchange rates of USD/Euro and Yuan, inflation rate and interest rate.
3. The temporal scope of this study is limited to the period spanning from 2017 to 2022, involving the past five years.

1.4 Contribution of knowledge

The research suggests potential benefits in enhancing the accuracy of silver price forecasting, gathering complex relationships, and providing valuable insights for various stakeholders. Through the application of deep learning methodologies, the algorithms could be developed to improve their effective predictions, which enhance the field of time series forecasting and financial market analysis.

CHAPTER 2

LITERATURE REVIEW

This literature review examines the current methodologies and insights in silver price prediction, emphasizing data management, modeling strategies, and influential variables. It explores a selection of models, from traditional Linear Regression to sophisticated LSTM, GRU, XGBoost, and Ensemble models, highlighting the application of deep learning and time series techniques. The review covers key influential factors such as exchange rates, gold and oil prices, interest rates, and inflation. Additionally, it reaches into the aspects of model evaluation and hyperparameter tuning, all while interpreting feature significance. The review shares a thorough discussion on the related studies in the field.

2.1 Silver Price Forecasting Overview

Silver Price Forecasting

Silver price forecasting has traditionally relied on methods like ARIMA, GARCH, and VAR models, but these struggle to capture non-linear relationships and complex patterns. Machine learning models, such as Random Forests and Support Vector Machines, have shown improved accuracy by incorporating various indicators. Deep learning models like LSTM and GRU networks have excelled at capturing temporal dependencies and long-term patterns. Ensemble models that combine multiple models further enhance accuracy. Hybrid models that blend statistical and machine learning approaches have also been explored. These techniques offer promising results for capturing complex patterns and supporting decision-making in the silver market. Ongoing research aims to improve models, feature engineering, and data sources for more accurate silver price forecasting.

Incorporating USD Exchange Rate Information

The USD exchange rate has been found to have a significant impact on silver prices, with studies highlighting an inverse relationship between the two.

A stronger USD is associated with lower silver prices, while a weaker USD is associated with higher silver prices. Including exchange rate data in silver price forecasting models allows for the consideration of global macroeconomic factors and can enhance the accuracy of predictions. By incorporating exchange rate information, models can capture the interplay between currency movements and silver prices, providing valuable insights for investors and policymakers in the silver market.

Gold Prices

Research studies have consistently found a strong relationship between gold and silver prices. Evidence suggests that gold and silver prices tend to move together in the long run, indicating that shocks in the gold market can impact the silver market as well. The similarity in their uses, including industrial applications and store of value, contributes to the correlation between the two metals. This relationship has implications for investors and market participants, as understanding the dynamics between gold and silver prices can provide valuable insights for forecasting and decision-making purposes.

Oil Prices

The relationship between oil prices and silver prices is complex and can be influenced by various factors. While some studies have found a positive relationship, indicating that increases in oil prices lead to increases in silver prices, others argue that the relationship may be nonlinear and affected by additional factors such as market volatility. The industrial uses of silver, coupled with production cost considerations influenced by oil prices, can contribute to the observed relationship. However, further research is needed to fully understand and characterize the dynamics between oil and silver prices.

Interest Rates

The relationship between interest rates and silver prices is complex and subject to ongoing research. While higher interest rates may increase the opportunity cost of holding silver, potentially leading to lower prices, other factors such as inflation expectations and economic uncertainty can also influence this relationship.

The impact of interest rates on silver prices requires further study to understand the underlying dynamics and potential nonlinearities.

Inflation Rates

The relationship between inflation rates and silver prices suggests that silver can serve as a hedge against inflation. Higher inflation rates may lead to increased demand for silver as investors seek to protect their wealth, potentially causing silver prices to rise. Studies have provided evidence of a long-run relationship between inflation rates and silver prices, further supporting the notion that silver can act as a hedge against inflation.

The terms "short-term" and "long-term" predictions are flexible, adapting to the context of specific research goals, dataset characteristics, and the market dynamics of the commodity in focus, such as silver.

Short-Term and Long-Term Predictions in Silver Price Forecasting

Generally, "short-term" forecasts in the field refer to projections spanning from a single trading day to several months. Such forecasts refine immediate market trends, daily financial indicators, and transient shifts in the supply-demand balance, as supported by the findings of Clements and Galvao (2008).

Contrarily, "long-term" predictions extend over a year to several years or even decades. These anticipate the impact of broader economic shifts, including interest and inflation rates, geopolitical events, consistent industrial demands, and expected mining output changes, as described by Tsay (2005).

Given these broad classifications, it's crucial to understand that these definitions are adaptable based on individual study requirements or the specifics of the silver market.

From this perspective, our study classifies "short-term" and "long-term" silver price predictions as follows:

Short-term silver price prediction: This term denotes the forecasting of silver prices over a period extending from a single trading day to a few months. These predictions aim to interpret immediate market behaviors and current economic events.

Long-term silver price prediction: This concept refers to the prediction of silver prices over an extended timeframe, usually from a year onwards. These predictions incorporate broader economic patterns, consistent industrial demand, and expected supply alterations.

2.2 Data Handling

Data Source

Yahoo Finance is a popular financial website that provides a wide range of financial information and services. It offers a comprehensive platform for investors and traders to access real-time stock quotes, historical price data, financial news, portfolio management tools, and other financial resources.

Quandl is a platform that provides access to a vast collection of financial and economic data. It serves as a data marketplace, offering a wide range of datasets from various sources, including financial markets, economic indicators, commodities, and more.

FRED stands for Federal Reserve Economic Data, which is a comprehensive and widely used economic database maintained by the Federal Reserve Bank of St. Louis. It provides access to a vast collection of economic and financial data from various sources, including government agencies, central banks, international organizations, and private sector entities, which contains a wide range of economic indicators, time series data, and financial statistics that cover various aspects of the economy. These include macroeconomic indicators such as GDP, inflation rates, employment figures, interest rates, exchange rates, and more. The data available in FRED spans multiple countries and covers different time periods, allowing researchers, economists, analysts, and policymakers to study and analyze economic trends and patterns.

Forward-filling

Forward-filling, also known as last observation carried forward (LOCF), is a method where missing values are filled with the most recently observed value. This technique assumes the concept of inertia, wherein the previous observation

persists until a new observation occurs. Therefore, in our daily data, if a silver price or USD exchange rate was missing for a particular day, we filled this missing value with the data from the preceding day. While this method is simple and often effective, it assumes that there is no significant change within the missing periods, which might not always be true. Thus, this technique was applied judiciously to prevent the introduction of any bias.

Interpolation

Interpolation is a technique where a missing value is filled with an estimated value derived from surrounding known values. Linear interpolation, the simplest and most used form, assumes a straight line between two known points and uses that to estimate the missing values. This technique can provide reasonable estimates for missing data points when there's a linear relationship between the data points. However, it might not be as effective if the data is characterized by frequent or large fluctuations.

Extrapolation

Extrapolation is a statistical method that involves estimating values beyond the range of the given data set. It is often used in time series analysis or predictive modeling when future data points are required but are not available in the existing data set. It relies on the assumption that the current trend in the data will continue. The method takes the known dataset, fits a model to this data, and then extends that model beyond the known range. While extrapolation can be a valuable tool in predictive analytics, it is important to be aware of its limitations. The main risk with extrapolation is that it assumes the future will continue to behave like the past. In reality, this is not always the case, especially over long-time horizons or when dealing with volatile data.

Lagged Variables

Lagged variables are fundamental in the field of time series forecasting due to their ability to feed past values from the series into the predictive model. This approach is predicated on the assumption that the future values of a series hinge

upon its past values. Such an aspect is especially valuable for models that do not inherently accommodate temporal dependencies. Additionally, it benefits models such as LSTM and GRU, which are tailored to handle sequential data.

Moving Averages

A moving average is a widely used statistical procedure that aims to analyze various data points by generating a series of averages from different subsets of a complete dataset. Especially pertinent to time series data, it effectively smoothens short-term volatility and highlights longer-term trends or cycles, reduces the impact of noise and outliers.

This technique involves the calculation of the average over a defined period, the window size. For instance, a 7-day moving average calculates the average of the preceding seven days and then moves forward to the subsequent seven days. In a similar, a 30-day moving average takes the average of the previous 30 days and advances to the next 30 days.

Data Normalization

Normalization is a scaling method used to ensure that different features contribute equitably to the final prediction. Without normalization, features with larger numerical ranges could overshadow those with smaller numerical ranges, possibly skewing the model. In this research, we utilized min-max normalization, a process that reshapes the data to fit within a certain range, usually between 0 and 1. This approach preserves the original data distribution while ensuring that all features have the same scale.

The formula for min-max normalization is:

Normalized value = (Original value - Minimum value) / (Maximum value - Minimum value)

Train-Test Split

The train-test split divides the dataset into a training set for model development and a test set for assessing the model's performance on unseen data. This strategy aids in preventing overfitting, a situation where a model excessively

learns the noise and outliers in the training data, which can impair its ability to generalize to new data.

The division process is usually random, ensuring that the training and test sets represent the overall data distribution. Although split ratios can vary, common practices include a 70:30 or 80:20 split for training and testing, respectively. In time-series analysis, the split is sequential rather than random due to the temporal nature of the data. The earlier observations form the training set, and the later ones constitute the test set.

2.3 Methodology and Model Approaches

Linear Regression

Linear regression is a statistical method used to examine the relationship between a dependent variable and one or more independent variables. It assumes a linear correlation between these variables, implying that changes in the independent variables lead to proportional changes in the dependent variable.

The objective in linear regression is to find the best-fit line that minimizes the difference between the predicted and actual values of the dependent variable, by estimating the coefficients that shape the equation of the line. The equation for simple linear regression can be represented as:

$$Y = \beta_0 + \beta_1 X + \varepsilon \quad (2.1)$$

Where:

- Y is the dependent variable
- X is the independent variable
- β_0 is the intercept (the value of Y when X is 0)
- β_1 is the slope (the change in Y for a one-unit change in X)
- ε is the error term representing the variability in Y that is not explained by the model

The coefficients β_0 and β_1 are estimated using methods like the least squares method, which minimizes the sum of the squared differences between the predicted and actual values. The model can be used the coefficients to predict

the dependent variable based on new values of the independent variable. Linear regression can be extended to multiple independent variables, resulting in multiple linear regression. In this case, the equation becomes:

$$Y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots + \beta nXn + \varepsilon \quad (2.2)$$

Where X1, X2, ..., Xn are the independent variables, and $\beta 1$, $\beta 2$, ..., βn are the corresponding coefficients.

In linear regression analysis, certain fundamental assumptions emphasize the validity and interpretability of the model. Here, we outline six essential assumptions:

Linearity: The assumption of linearity determines that there is a linear correlation between the independent and dependent variables. Essentially, a unit change in an independent variable result in a consistent change in the dependent variable.

Independence: This assumption asserts that the residuals (errors) of the model are independent of each other. The Durbin-Watson test can be applied to detect violations of this assumption, especially important in time series analysis where autocorrelation may be present.

Homoscedasticity: This proposes that the variance of residuals is constant across all levels of independent variables.

Normality of Errors: This assumption implies that the residuals (or errors) are normally distributed. It lies in making inferences about the regression parameters, and it becomes less important with larger sample sizes due to the Central Limit Theorem.

Absence of Multicollinearity: Multicollinearity occurs when independent variables are highly intercorrelated, leading to difficulties in distinguishing their individual effects on the dependent variable. While not compromising the model's overall predictive power, multicollinearity can destabilize the coefficient estimates.

Exogeneity of Regressors: This assumption indicates that there should be no correlation between the residuals and the independent variables. A violation may hint at simultaneity or omitted variable bias.

Long Short-Term Memory (LSTM)

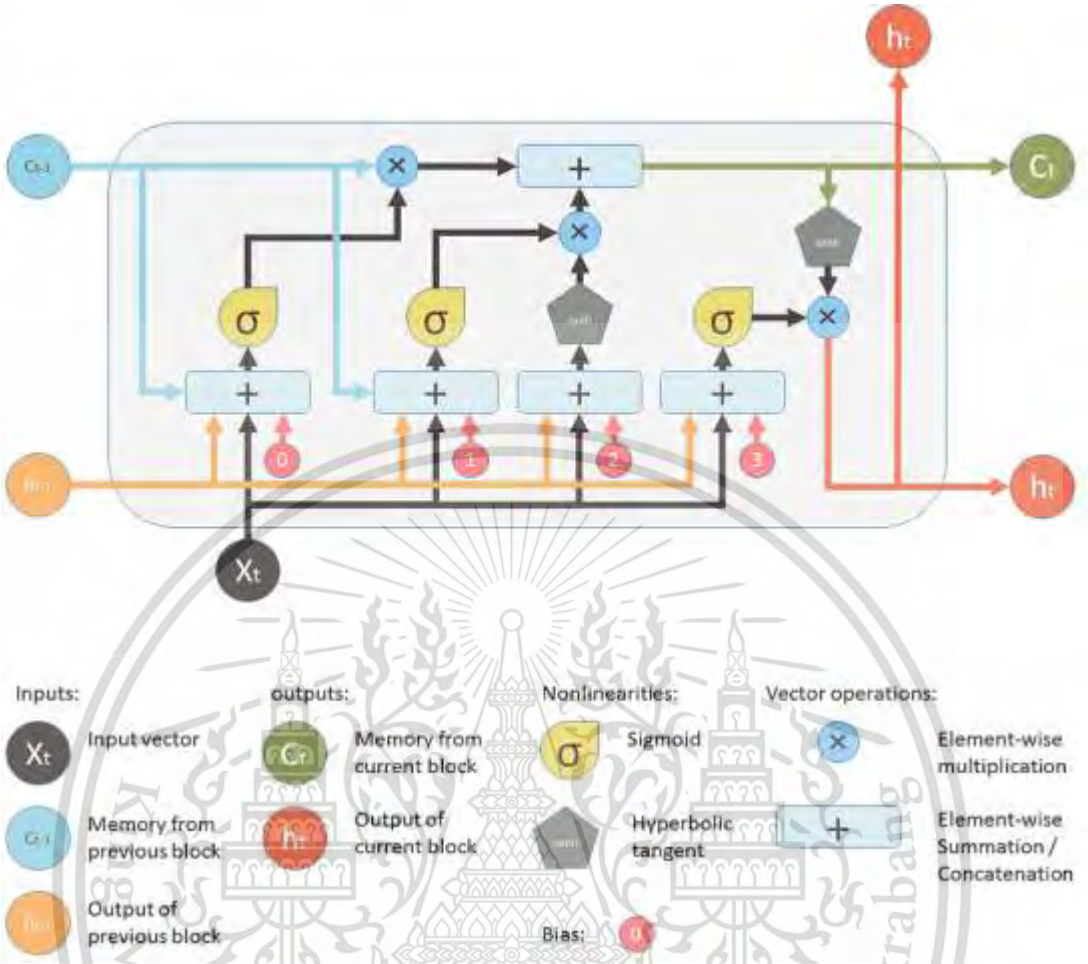

Figure 2.1: The structure of a long short-term memory (LSTM) algorithm.

Source: https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714

LSTMs are a type of recurrent neural network (RNN) and are particularly suited for time series forecasting due to their ability to capture long-term dependencies in the data. This is achieved through a unique cell state that can retain information over long periods, making them less prone to issues like vanishing or exploding gradients. The LSTM architecture (Figure 2.1) consists of several key components:

Cell State: The cell state serves as the long-term memory of the LSTM. It runs through the entire sequence of data, and information can flow across time steps without being significantly altered. The cell state acts as a conveyor belt, selectively updating and passing relevant information.

Gates: LSTMs use three types of gates to control the flow of information: the input gate, the forget gate, and the output gate.

Input Gate: The input gate determines the relevance of new input data to be added to the cell state. It considers the previous hidden state and the current input, passing the filtered information to the next steps.

Forget Gate: The forget gate decides which information to discard from the cell state. It evaluates the previous hidden state and the current input, selectively removing unnecessary information to avoid the accumulation of irrelevant data.

Output Gate: The output gate determines the relevance of the current hidden state, which contains the learned information, to the final output of the LSTM. It controls the information that is passed on to subsequent layers or used for predictions.

Hidden State: The hidden state represents the LSTM's short-term memory and provides a summary of the relevant information captured from the input sequence. It is updated based on the input, the previous hidden state, and the output of the input and forget gates. The hidden state carries information across time steps and serves as the output of the LSTM at each time step.

The equations for the gates in LSTM are:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad (2.3)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad (2.4)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad (2.5)$$

Where:

i_t is input gate

f_t is forget gate

o_t is output gate

σ is sigmoid function

w_x is weight for respective gate (x) neurons

h_{t-1} is output of previous LSTM block (at timestamps t-1)

x_t is input at current timestamp

b_f is biases for respective gates (x)

Gated Recurrent Units (GRUs)

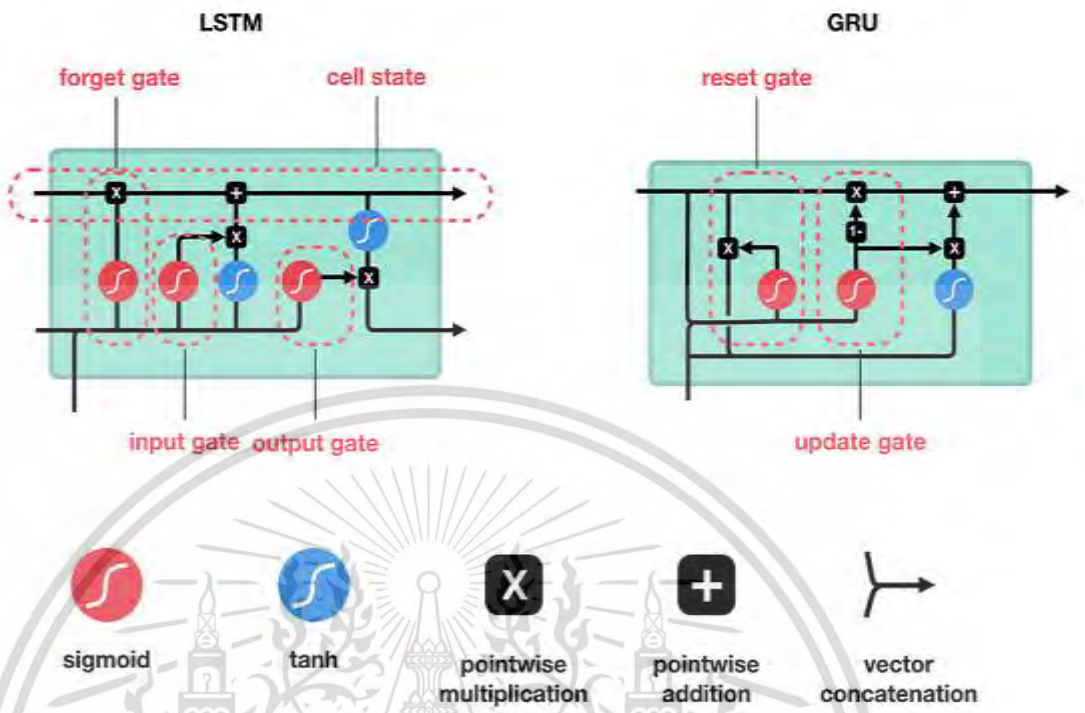

Figure 2.2: The structure of a long short-term memory (LSTM) algorithm with the cell model of a GRU block diagram.
Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

GRUs are another variant of recurrent neural network (RNN) architecture that, like LSTMs, can effectively model sequential and time-dependent data. GRUs were introduced as a simplified version of LSTMs with fewer gates, making them computationally efficient while still maintaining the ability to capture long-term dependencies. The GRU architecture (Figure 2.2) consists of the following key components:

Update Gate: The update gate in a GRU determines the degree to which the model incorporates new information from the current input. It considers the previous hidden state and the current input, generating an update value between 0 and 1. This gate controls the flow of information from the previous hidden state to the current hidden state.

Reset Gate: The reset gate in a GRU decides how much of the previous hidden state to forget or reset. It also considers the current input, generating a reset value between 0 and 1. This gate enables the model to selectively discard irrelevant information from the previous hidden state.

Hidden State: The hidden state in a GRU represents the model's memory or information accumulated from past inputs. It is updated based on the update and reset gates, allowing the model to retain relevant information while disregarding unnecessary or outdated information. The hidden state serves as the output of the GRU at each time step and can be passed on to subsequent layers or used for predictions.

The equations for the gates in GRUs are:

$$z_t = \sigma\left(w^{(z)}x_t + U^{(z)}h_{t-1}\right) \quad (2.6)$$

$$r_t = \sigma\left(w^{(r)}x_t + U^{(r)}h_{t-1}\right) \quad (2.7)$$

Where:

z_t is update gate

r_t is reset gate

σ is sigmoid function

w_x is weight for respective gate (x) neurons

h_{t-1} is output of previous GRU block (at timestamps t-1)

x_t is input at current timestamp

$U^{(h)}$ is weight for output of previous GRU block (at timestamps t-1)

Compared to LSTMs, GRUs have a simpler architecture by combining the forget and input gates of LSTMs into a single update gate. This reduces the number of parameters and computations required, making GRUs computationally efficient, especially on smaller datasets. However, since they have fewer gates, GRUs may have a slightly reduced capacity to model complex long-term dependencies.

XGBoost

The underlying algorithm behind XGBoost is gradient boosting. The Gradient Boosting algorithm works by combining weak learners, usually decision trees, into a weighted sum that represents the final output of the boosted classifier.

A weak learner is defined as a classifier that is only slightly correlated with the true classification. By combining these weak learners, we can create a model that makes fewer errors and provides more accurate predictions. Key features of XGBoost include:

Regularization: It provides options to penalize complex models through L1 and L2 regularization, avoiding overfitting in intricate machine learning models.

Parallel Processing: It uses parallel processing, enabling faster computation than other boosting algorithms. It features a block structure for parallel learning, creating decision nodes simultaneously.

Handling Missing Values: It can manage missing values automatically. Users need to assign a unique value for missing observations and pass it as a parameter.

Tree Pruning: Unlike GBM, which halts tree pruning when a negative loss is detected, XGBoost builds the tree up to max_depth and then prunes it to optimize the model.

Built-in Cross-Validation: It supports cross-validation at each boosting iteration, allowing users to determine the optimal number of boosting iterations in a single run.

Flexibility: Users can define custom optimization objectives and evaluation criteria, adding versatility to the model.

Scalability: It can efficiently handle big data scenarios, making it highly scalable.

Ensemble Model

An Ensemble model in machine learning is a technique that combines multiple models to create a more robust and generalized model that improves prediction accuracy. The idea behind Ensemble modeling is to aggregate the strengths of multiple diverse models, thereby decreasing variance (overfitting), bias

(underfitting), or improving predictions. Ensemble methods can be categorized into two main types:

Sequential Ensemble Methods: These methods have a particular dependency between the base learners. The base models are generated sequentially, such as in AdaBoost and Gradient Boosting. The motivation is to exploit the dependence between the base learners since the overall performance can be boosted by weighing previously mislabeled examples with higher weight.

Parallel Ensemble Methods: These methods build base learners independently, such as in Bagging and Random Forest. The motivation is to exploit independence between the base learners since the error can be reduced dramatically by averaging.

There are several strategies to build Ensemble models:

Bagging: Bagging stands for bootstrap aggregation. It works by creating multiple subsets of the original data with replacement, fitting a model on each subset, and averaging the predictions. The goal of bagging is to reduce the variance of a decision tree classifier. A famous example of bagging is the Random Forest algorithm.

Boosting: Boosting works sequentially by modifying the weights of instances in the training dataset based on the accuracy of the previous model. The objective of boosting is to reduce bias as well as variance of a classifier. Notable examples of boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost.

Stacking: Stacking, also known as Stacked Generalization, involves training a learning algorithm to combine the predictions of several other learning algorithms. The idea is to train a model to effectively learn how to best combine the input models to make a more accurate prediction.

In practice, Ensemble models often outperform single models and are widely used in image and speech recognition, natural language. However, they can be more computationally expensive and complex to implement, interpret, and understand.

Hyperparameters

In the context of LSTM and GRU, there are several hyperparameters. Some of these are:

Number of Hidden Units/Neurons: This is one of the most important hyperparameters for LSTM and GRU models. It determines the capacity of the model, i.e., the amount of information that the model can capture. A larger number of units increases the model's capacity.

Number of Layers: This is another critical hyperparameter. A deep network has more layers and can learn more complex patterns. However, it's also more likely to overfit and is harder to train.

Learning Rate: This determines the step size at each iteration while moving toward a minimum of a loss function, i.e., how quickly the model learns. It's one of the most difficult parameters to set, as a value too low will result in a long training process, while a value too high may lead to the model missing the minimum.

Batch Size: This is the number of training examples used in one iteration. For instance, if you have 1000 training examples and your batch size is 500, then each epoch will consist of 2 iterations. It's important to find a balance, as smaller batch sizes might lead to better model performance but slow the training process.

Dropout: This is a regularization technique that helps prevent overfitting in neural networks. It works by randomly dropping out (i.e., setting to zero) a number of output features of the layer during training. The "dropout rate" is the fraction of the features that are being zeroed-out.

As for ensemble models, they are typically constructed by combining multiple models to improve performance. The key hyperparameters here would be the number and types of models being combined.

Number of Models: This determines how many base models are included in the ensemble. Increasing the number of models can increase performance up to a certain point but may also lead to increased computational complexity and the risk of overfitting.

Types of Models: The types of models to use as base learners is another critical decision. Ideally, the base learners should be diverse (i.e., make different types of errors) to ensure that they complement each other.

XGBoost also has a number of hyperparameters that control its behavior:

n_estimators: This is the number built of trees before taking the maximum voting or averages of predictions. Higher values result in better performance but can also lead to overfitting.

max_depth: This sets the maximum depth of the tree and controls the complexity of the model. Deeper trees can model more complex relationships, but they can also lead to overfitting.

min_child_weight: This parameter defines the minimum sum of weights needed in a child. It's used to control overfitting and works similarly to min_child_leaf in GBM, but it considers the sum of weights of observations instead of just the number of observations in a leaf.

gamma: This parameter controls the minimum loss reduction required to make a split. It makes the algorithm more conservative, meaning that splits with a low loss reduction won't occur.

subsample: This is the fraction of observations (rows) to subsample for each tree. Lower values make the algorithm more conservative and can prevent overfitting, but they can also lead to underfitting.

colsample_bytree: This is the fraction of features (columns) to use for each tree. Like the subsample parameter, this can prevent overfitting, but too small a value might lead to underfitting.

learning_rate (eta): This makes the model more robust by shrinking the weights on each step, similar to learning rate in other models.

reg_lambda (lambda) and reg_alpha (alpha): These are L1 and L2 regularization terms on weights, respectively. Increasing these values will make the model more conservative.

2.4 Model Evaluation, Metrics and Tuning

Root Mean Squared Error (RMSE)

Formula: RMSE $= \sqrt{\left(\frac{1}{n}\right)\Sigma(Y - \hat{Y})^2}$ *(2.8)*

RMSE is a commonly used metric that calculates the square root of the average of the squared differences between the predicted values ($\hat{Y}$) and the actual values (Y). RMSE is sensitive to larger errors due to the squaring operation, giving more weight to larger discrepancies.

Mean Absolute Percentage Error (MAPE)

Formula: $$\text{MAPE} = \left(\frac{1}{n}\right) \Sigma \left(\left|\frac{Y-\hat{Y}}{Y}\right|\right) * 100 \quad (2.9)$$

MAPE calculates the average percentage difference between the predicted values ($\hat{Y}$) and the actual values (Y), expressed as a percentage. It provides a measure of the relative accuracy of the predictions, considering the magnitude of the errors in relation to the actual values.

In these formulas:

- Y represents the actual values.
- $\hat{Y}$ represents the predicted values from the model.
- n represents the total number of data points.

These evaluation metrics help quantify the performance of regression models, allowing us to assess their accuracy, precision, and ability to capture the true values. By calculating and analyzing these metrics, we gain valuable insights into the effectiveness of our models for forecasting silver prices.

Mean Absolute Error (MAE)

The MAE is the average absolute difference between observed and predicted outcomes and is represented mathematically as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \quad (2.10)$$

Where:

- n is the total number of observations or data points.
- y_i is the actual value of the outcome for observation i.
- $\hat{y}_i$ is the predicted value of the outcome for observation i.

In this formula, each difference between a model's prediction ($\hat{y}_i$) and the actual data point (y_i) is calculated. The absolute values of these differences are taken to remove any negative signs. This is important because without taking absolute values, the positive and negative errors could cancel each other out, giving a false sense of accuracy. These absolute differences are then summed up and divided by the total number of observations n, providing an average measure of the error magnitude.

One of the main advantages of MAE is its direct interpretability. Unlike other metrics such as mean squared error (MSE), which squares the errors before averaging them.

Coefficient of Determination

The Coefficient of Determination, denoted as R^2, is a statistical measure that is used to assess the goodness-of-fit of a regression model. It quantifies the proportion of variance in the dependent variable that is predictable from the independent variable(s). R^2 provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

R^2 is mathematically represented as follows:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{2.11}$$

Where:

- R^2 is the coefficient of determination
- TSS is the total sum of squares, calculated as $\sum(y_i - y_{mean})^2$ It measures the total variance in the dependent variable y.
- RSS is the residual sum of squares, calculated as $\sum(y_i - \hat{y}_i)^2$. It measures the total residual variance.

The value of R^2 lies between 0 and 1:

An R^2 of 0 indicates that the dependent variable cannot be predicted from the independent variable.

An R^2 of 1 indicates the dependent variable can be perfectly predicted from the independent variable.

An R^2 between 0 and 1 indicates the extent to which the dependent variable is predictable. An R^2 of 0.10 means that 10 percent of the variance in y is predictable from X; an R^2 of 0.20 means that 20 percent is predictable, and so on.

However, a high R^2 does not necessarily mean the model is a good fit. It's possible to have a high R^2 for a poorly fitting model.

Hyperparameter Tunning

Hyperparameter tuning is a step in developing robust machine learning and statistical models, focusing on determining the best values for model parameters set before training. Unlike other parameters that can be learned from training data, hyperparameters, such as the learning rate, regularization parameters, or the depth of a decision tree, must be predefined. The settings of these parameters can significantly affect the model's performance and predictive power. Several strategies for hyperparameter optimization, including:

Grid Search: This method exhaustively tries all combinations of hyperparameters to identify the set resulting in the best model performance.

Random Search: This method samples random combinations of hyperparameters.

Bayesian Optimization: A more advanced technique, it constructs a probabilistic model mapping hyperparameters to a score of the model's performance on the test set. Acquisition functions guide the selection of the next set of hyperparameters to evaluate.

Timesteps

In time series analysis and recurrent neural networks (RNNs), a timestep signifies a single observation in a sequence of data points. It determines how much past data the model considers while making predictions.

The number of timesteps is a problem-specific decision that can impact the model's performance. Too few timesteps may not capture data trends and patterns

well enough, causing unsatisfactory predictions. Conversely, too many timesteps might lead to overfitting, where the model over-learns the training data and performs poorly on unseen data.

2.5 Data Analysis and Interpretation

Z-Score

The Z-score, often referred to as the standard score, is a fundamental concept in statistical data analysis. The score provides a means to understand an observation's relationship to the mean of a group of values. In other words, it offers a measure of how many standard deviations an observation is from the mean, thereby allowing for the comparison of data points from different normal distributions. The formula for calculating the Z-score is as follows:

$$Z = (X - \mu) / \sigma \quad (2.12)$$

Where:

- X is the value of the observation.
- μ is the mean of the population or sample.
- σ is the standard deviation of the population or sample.

A positive Z-score indicates that the observed value is above the mean, while a negative Z-score suggests that it is below the mean. Furthermore, the absolute value of the Z-score reflects the distance between the observation and the mean in terms of standard deviations.

In many applications, Z-scores are utilized for outlier detection. Outliers are observations that are significantly different from other data points, and they can distort statistical analyses and violate their assumptions. In a standard normal distribution, data points with a Z-score greater than 3 or less than -3 are typically considered outliers, as they are more than three standard deviations away from the mean. However, this threshold can be adjusted depending on the specific context and the degree of deviation considered acceptable.

Interquartile Range (IQR)

The Interquartile Range (IQR), also known as Midspread or H-spread, is a widely used statistical measure that describes the spread of a dataset, or essentially, the statistical dispersion. The IQR represents the middle 50% of the data and is the range between the first quartile (25th percentile) and the third quartile (75th percentile). The formula to compute the IQR is as follows:

$$IQR = Q3 - Q1 \tag{2.13}$$

Where:

- Q1 is the first quartile, which is the value below which 25% of the data fall.
- Q3 is the third quartile, which is the value below which 75% of the data fall.

The IQR is particularly effective in portraying a dataset's dispersion because it is resistant to the presence of outliers. Unlike the range, which is sensitive to extreme values, the IQR focuses solely on the data's central portion, providing a more robust measure of spread. The IQR is often used in conjunction with the box plot, a common graphical representation of a dataset's summary statistics. The box plot's box spans the IQR, and the whiskers extend to the minimum and maximum data values within a specific range, often 1.5 times the IQR away from Q1 and Q3, respectively. Data points outside this range are considered outliers.

In the financial field, the IQR can be used to understand the volatility or variability in data such as stock prices, economic indicators, and other financial time series. It can also be used for outlier detection, with outliers often defined as observations that fall below Q1 - 1.5*IQR or above Q3 + 1.5*IQR.

Feature Importance

Feature importance is a concept in machine learning and data analysis that aims to identify the relative significance of input variables in predicting the target variable. The idea is that not all features contribute equally to the predictive power of a model. Some features may have a stronger influence on the target variable, while others may be less informative or redundant.

One commonly used method to measure feature importance is permutation importance. This approach involves randomly shuffling the values of a feature while

keeping other features constant and observing the impact on the model's performance. The decrease in performance, such as accuracy or mean absolute error, indicates the importance of the shuffled feature.

2.6 Related Research

1. "Predicting Gold and Silver Price Direction Using Tree-Based Classifiers" (2021) by Perry Sadorsky. This research applied tree-based machine learning classifiers like bagging, stochastic gradient boosting, and random forests for forecasting the direction of gold and silver price exchange traded funds. These classifiers outperformed logit models. Thus, the combination of tree bagging and random forests was found to be most effective in predicting gold and silver prices direction.

2. "A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long-Short Term Memory" (2017) by Bao et al. This study proposed a deep learning framework for predicting financial time series data, demonstrating its effectiveness in comparison with other machine learning models.

3. "Prediction of gold and silver stock price using ensemble models" (2014) by Pradeep Kumar Mahato and Dr. Vahida Attar. explored machine learning algorithms to predict gold and silver prices, employing ensemble models. They utilized Majority Voting with various base classifiers. Their research revealed that stacking with output showed superior accuracy for gold, while bagging with random subspace excelled for silver.

4. "Copper Price Prediction Using LSTM Recurrent Neural Network and Simulation Annealing Algorithm" (2022) by Jiahao chena,Tongping Lia,Jinhua Chenga, and Chuandi Fanga. They undertook a study focusing on copper price prediction. Their approach utilized Long Short-Term Memory (LSTM) in conjunction with a simulated annealing model to determine the optimal hyperparameters. The model was built using three significant economic indicators: WTI oil price, gold price, and silver price. The outcomes of their study signify an enhancement in the copper price prediction model, in terms of both prediction time and accuracy.

CHAPTER 3

RESEARCH METHODOLOGY

This section outlines the methodology employed in this study for forecasting silver prices using time series data and deep learning models. Here, we describe the detailed process we followed for the methodology (Figure 3.1):

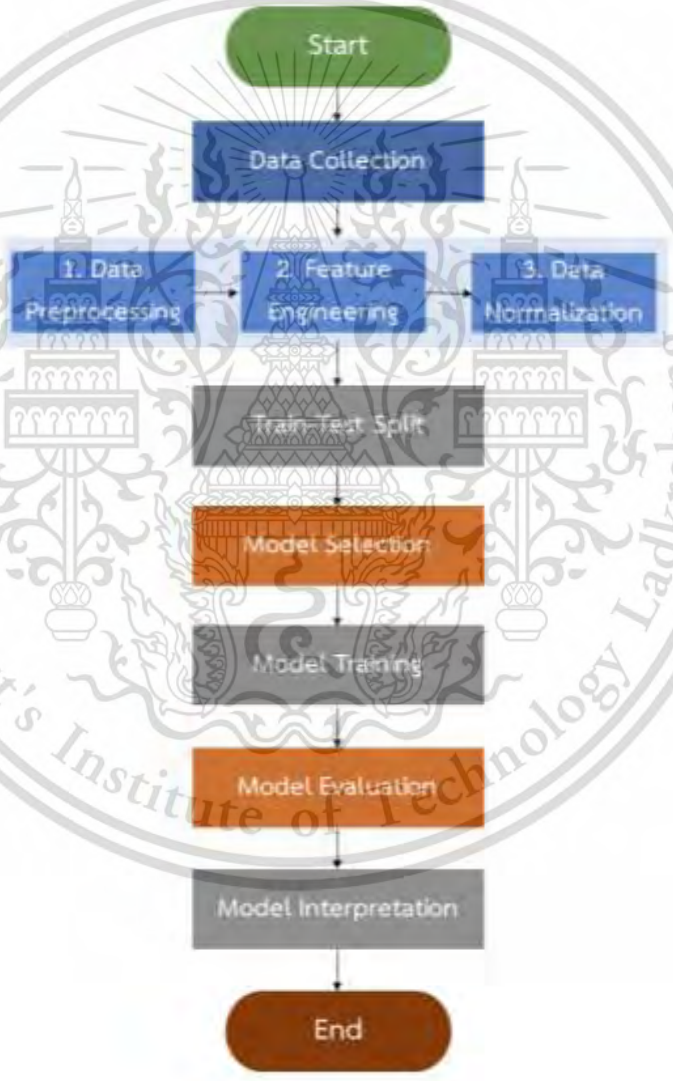

Figure 3.1: Flowchart of Research Methodology

This study utilized a high-performance personal computer with an AMD Ryzen 7 5800H processor and 16.0 GB of RAM, optimized for intensive processing tasks and large data volumes. Its 64-bit system facilitates efficient execution of complex instructions, making it suitable for machine learning modeling and data analysis.

3.1 Import Libraries

This analysis utilized various Python libraries and modules for data preprocessing, modeling, and performance evaluation. Fundamental libraries such as Pandas and Numpy were used for data manipulation and computation. Data visualization was handled by Matplotlib and Seaborn, while the Datetime module facilitated date and time manipulation. Machine learning was predominantly accomplished with Sklearn and its numerous tools for preprocessing and performance metrics, and the Scikit-learn integrated linear_model, and KerasRegressor for building and training models. Deep learning models were built using TensorFlow and Keras. Additional libraries like SciPy were used for scientific computation, and Ensemble methods such as GridSearchCV, also from Scikit-learn, were used for hyperparameter tuning and robust predictions. Collectively, these modules facilitated a holistic approach to data preprocessing, feature selection, model building, tuning, and evaluation.

3.2 Data Collection

We employed historical silver prices and related data from reputable financial databases such as Yahoo Finance, Quandl, and Federal Reserve Economic Data to ensure reliability. The daily closing prices of silver, USD exchange rates against the Euro and Yuan, gold prices, oil prices, interest rates, and inflation rates were all collected for the period 2017-01-12 to 2022-02-08, total 1,263 days. By selecting these variables, the study aimed to capture the complex interplay of factors that impact silver prices, enabling a multifaceted analysis. The use of daily data was specifically chosen to track granular market fluctuations and align with the research

objective of daily silver price forecasting. Here's a detailed description of the code and its components:

Data Sources

The research uses several libraries to fetch data from different financial sources. The yfinance library is utilized to gather historical price data for Silver (symbol: SLV) and Gold (symbol: IAU) from Yahoo Finance. The quandl library facilitates access to Quandl's financial and economic data, fetching data for USD/Euro exchange rates (dataset code: FRED/DEXUSEU), USD/Yuan exchange rates (dataset code: FRED/DEXCHUS), and Oil Price (dataset code: EIA/PET_RWTC_D). Finally, the fredapi library is employed to retrieve Interest Rate (dataset code: FEDFUNDS) and Inflation Rate (dataset code: FPCPITOTLZGUSA) data from the Federal Reserve Economic Data (FRED) API.

Data Retrieval and Preprocessing

The data collection processed by setting specific start and end dates, ranging from 2017-01-12 to 2022-02-08. Then, we used the API function to fetch historical price data for Silver and Gold, focusing on the 'Close' prices and converting the index to a datetime format and replicated for USD/Euro and USD/Yuan exchange rates, Oil Price, Interest Rate, and Inflation Rate data. These datasets are merged based on a common 'Date' column using the pandas merge function. Additional preprocessing steps include calculating the seconds from the start date and extracting day, month, and year information into new columns. For the interest rate and inflation rate data, we resampled, and interpolated daily data within the specified timeframe by a forward-fill method.

For inflation rate data, missing values are handled through interpolation. The data is then extrapolated up to a specified end date, providing a way to predict future values by extending current data trends. Both the interest and inflation rate data are filtered to include only information within the desired timeframe, from 2017-01-12 to 2022-02-08.

3.3 Data Exploration

The exploratory data analysis (EDA) begins with the visualization of time series data related to Silver and Gold Closing Prices, Oil Price, USD Exchange Rate, Interest Rate, and Inflation Rate. Visualizing the original values and their 7-day and 30-day moving averages allows for an examination of the overall trends, temporal patterns, and detection of short-term fluctuations and long-term cycles.

The collection of column names includes variables such as 'sil_close', 'gold_close', 'oil_value', 'usd_euro', 'usd_yuan', 'interest_rate', and 'inflation_rate', along with their moving averages. A figure and axes are initialized using Matplotlib's plt.subplots() function, with data plotted iteratively using a loop that traverses the column names. The x-axis, labeled 'Year', displays a maximum of six bins to balance detail and readability. The y-axis, labeled 'Value', broadly represents the financial metrics being plotted.

Next, histograms visualize the distribution of variables, offering insights into data distribution shape, skewness, outliers, and indicating if transformations are needed for normal distribution. A list, named 'variable_names', contains the variables of interest. For each variable, a histogram plot is created using Seaborn's sns.histplot() function.

Finally, a heatmap of the correlation matrix was created by using Seaborn's sns.heatmap() function, provides an intuitive understanding of the relationships between variables, emphasizing strong correlations critical for future modelling and feature selection processes.

3.4 Feature Engineering, Data Normalization, and Data Partitioning

The data is altered to accommodate a one-day time shift and a moving window of a 7-day period, which introduces null values into the initial 7 days of the dataset. This is expected in time-series manipulation and aids in the construction of lagged features for predictive modeling. Key steps in this process include:

Calculation of Moving Averages: A crucial step in time series analysis, moving averages smooth out short-term fluctuations to reveal underlying trends. Both 7-day

and 30-day moving averages were calculated for each variable using the .rolling(window=n).mean().shift(1) function.

Creation of Lagged Features: To account for temporal dependencies, lagged features were created using the .shift(day); day =1 function. These are data points from previous time steps.

Removal of Rows with Missing Values: Rows with insufficient historical data, resulting from moving averages and lagged feature creation, were removed to avoid feeding incomplete data into the models, leaving 1,243 days.

Data Normalization: MinMaxScaler() function was used for normalization, scaling the data between 0 and 1 to ensure all input features contribute equally to model predictions.

Data Partitioning: A 90/10 split divided the dataset into a training set (1124 days from 2017-01-12 to 2021-08-16) and a testing set (119 days, from 2021-08-17 to 2022-02-08).

Sequencing the Data: To train the model on sequences of data points, sequences containing 7 days' worth of data were created. This step is crucial for models like LSTM and GRU, which can capture long-term dependencies in sequences.

Feature Selection for Different Models: Different features were selected for different models based on their inherent strengths. LSTM, GRU, and ensemble models were trained in these feature sets. ['lagged_1_sil_close','lagged_1_gold_close', 'lagged_1_usd_euro', lagged_1_usd_yuan', 'lagged_1_oil_value', 'lagged_1_interest_rate', 'lagged_1_inflation_rate']

And Linear Regression and XGBoost models were trained in these feature sets. ['lagged_1_sil_close', 'lagged_1_gold_close', 'lagged_1_usd_euro', 'lagged_1_usd_yuan','lagged_1_oil_value','lagged_1_interest_rate', 'lagged_1_inflation_rate','lagged_2_sil_close','lagged_2_gold_close', 'lagged_2_usd_euro','lagged_2_usd_yuan','lagged_2_oil_value','lagged_2_interest_rate', 'lagged_2_inflation_rate', 'lagged_3_sil_close', 'lagged_3_gold_close', 'lagged_3_usd_euro', 'lagged_3_usd_yuan', 'lagged_3_oil_value', 'lagged_3_interest_rate', 'lagged_3_inflation_rate', 'lagged_4_sil_close',

'lagged_4_gold_close', 'lagged_4_usd_euro', 'lagged_4_usd_yuan',
'lagged_4_oil_value', 'lagged_4_interest_rate', 'lagged_4_inflation_rate',
'lagged_5_sil_close', 'lagged_5_gold_close', 'lagged_5_usd_euro', 'lagged_5_usd_yuan',
'lagged_5_oil_value', 'lagged_5_interest_rate', 'lagged_5_inflation_rate',
'lagged_6_sil_close', 'lagged_6_gold_close', 'lagged_6_usd_euro', 'lagged_6_usd_yuan',
'lagged_6_oil_value', 'lagged_6_interest_rate', 'lagged_6_inflation_rate',
'lagged_7_sil_close', 'lagged_7_gold_close', 'lagged_7_usd_euro', 'lagged_7_usd_yuan',
'lagged_7_oil_value', 'lagged_7_interest_rate', 'lagged_7_inflation_rate']

3.5 Model Training

Linear Regression

A Linear Regression model is instantiated using sklearn.linear_model module. Then, we checked linear regression included six assumptions. The model is trained on the dataset with the fit function. The dataset is divided into X_train1 (the features) and y_train1 (the target variable). Once trained, the model uses the prediction function on the test dataset (X_test1). Finally, coefficients and intercepts are calculated.

Six Assumptions

Linearity: This assumption states that there is a linear relationship between the independent and dependent variables. A scatter plot was used to visualize this relationship, followed by the use of Pearson's correlation coefficient for quantification. Python's matplotlib and scipy.stats libraries were used for this purpose.

Independence of residuals: Durbin-Watson statistic was used to test this assumption, ensuring that there is no correlation between consecutive residuals. Python's statsmodels library provides a Durbin-Watson test function that was utilized.

Normality of residuals: Residuals should follow a normal distribution. Python's seaborn library, specifically the distplot function, was used to plot the distribution of residuals. The Q-Q plot and the Shapiro-Wilk test from the scipy.stats library were also used to statistically verify normality.

Homoscedasticity: This implies that residuals have constant variance. The Breusch-Pagan test from Python's statsmodels library was used for this purpose, accompanied by a visual inspection of a plot of residuals vs. predicted values.

Absence of multicollinearity: The Variance Inflation Factor (VIF) from the statsmodels library was used to ascertain that predictors are not highly correlated with each other.

No endogeneity of regressors: This assumption indicates that there's no correlation between the error term and the independent variables. The Durbin-Wu-Hausman test was used to validate this assumption.

Performance Evaluation: These coefficients signify the importance of each feature in predicting the silver price, ranging from -0.23 to 0.79. A positive coefficient indicates an increase in silver price with a unit increase in the feature, while a negative one indicates a decrease. The model's intercept is -0.00153. The model performance is measured using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2). With an RMSE of 0.02, an MAE of 0.02, and an R^2 of 0.82, the model shows a satisfactory predictive performance. While the model performed well, it's crucial to examine if the assumptions of linear regression hold in this context.

Deep Learning Data Dimension

In our study, the number of days used for prediction forms the `timesteps`, which are set to `X_train.shape[1]`, denoting the number of time intervals. This value is 7, implying that the model uses a sequence of 7 days to make predictions.

The `input_dim` parameter represents the number of variables in the input data for each timestep. For our model, `input_dim` is set to `X_train.shape[2]`, indicating the number of input features, which is 7.

Finally, `output_dim` parameter points to the number of variables the model is predicting. As our model predicts a single variable, the next day silver price, the `output_dim` is set to 1.

LSTM Model

First, an LSTM model was defined using a function named create_lstm_model. We then created a KerasRegressor wrapper around the LSTM model, enabling compatibility with the Scikit-Learn framework. The KerasRegressor was initialized with the create_lstm_model function and parameters specifying the number of training epochs and batch size.

Next, we defined a grid of hyperparameters to fine-tune the LSTM model; 'units': [100, 150, 300], 'batch_size': [4, 16, 32], 'epochs': [100, 200, 300]. We performed a grid search using Scikit-Learn's GridSearchCV, an automated process that identifies the optimal set of hyperparameters by performing cross-validation on the training data for each combination. The scoring method employed was the negative mean squared error, signifying our aim to minimize the mean squared error. Post grid search, the function .fit() was used to fit the model to the training data. The results revealed the best performing model comprised 300 units in the LSTM layer, a batch size of 4, and was trained for 200 epochs.

Finally, these optimal LSTM model was instantiated using the create_lstm_model function with the best number of units identified from the grid search.

GRU Model

First, we defined a `create_gru_model` function, followed by a Dense layer, and then returns a compiled model leveraging the Adam optimization algorithm and mean squared error as the loss function. Next, we created a `KerasRegressor` wrapper using this function, also incorporating parameters such as the number of training epochs, batch size, and verbosity level. A parameter grid for hyperparameter tuning was then defined; 'units': [100, 150, 300], 'batch_size': [4, 16, 32], 'epochs': [100, 200, 300].

Subsequently, a grid search cross-validation was conducted using `GridSearchCV` from Scikit-Learn. The configuration included the `KerasRegressor` wrapper, the parameter grid, a scoring metric of negative mean squared error, and a specific number of cross-validation folds. The grid search was then initiated on the

training data using the `fit` method. Post completion, the optimal hyperparameters were extracted from the `GridSearchCV` results.

Finally, a GRU model was subsequently trained using optimal hyperparameter results, that the best GRU model consisted of 300 units in the GRU layer, a batch size of 4, and was trained for 300 epochs.

Ensemble Models

Ensemble LSTM with GRU model

Initially, the `create_ensem_model` function was set up. The function puts together a Sequential model including an LSTM layer, a GRU layer, and a Dense output layer. The LSTM layer is configured to return sequences and its number of units is determined by the 'units' parameter. The GRU layer uses a number of units derived from the multiplication of the 'units' and 'multiply' parameters. The Dense layer sets its number of output units based on the 'output_dim' variable. Adam optimizer and Mean Squared Error are applied during the model's compilation, with Root Mean Squared Error serving as a performance metric. The `KerasRegressor` wrapper then creates the ensemble model, utilizing the 'create_ensem_model' function for model definition, and establishing the batch size, the number of epochs, and verbosity level.

Next, a hyperparameter grid is outlined for tuning, containing various values for 'units', 'batch_size', and 'epochs'. Using negative mean squared error as the scoring metric, grid search cross-validation is performed on the training data, initiated by calling the `fit` method on the `GridSearchCV` instance. Following this, the optimal parameters are extracted from the grid search results, resulting in 64 units in the LSTM layer, 200 training epochs, and a batch size of 4.

Finally, with these optimal parameters in hand, the ensemble model is fitted to the training data, utilizing the optimal batch size and epoch number ascertained from the grid search process.

Ensemble model composed of stacked LSTM layers.

The ensemble model was constructed using the `create_ensem_model` function. The initial LSTM layer, set to return sequences, comprises 'units' LSTM units while the following LSTM layer has 'units' times 'multiply' LSTM units. The Dense layer has one output unit. The model is compiled with Adam optimizer and Mean Squared Error loss, with Root Mean Squared Error as the performance metric. The model was instantiated using `KerasRegressor`, utilizing the 'create_ensem_model' function for model definition, and setting epochs, batch size, and verbosity level.

For hyperparameter tuning, `GridSearchCV` was deployed, 'units': [100, 150, 300], 'batch_size': [4, 16, 32], 'epochs': [100, 200, 300], using negative mean squared error as the scoring metric. The grid search was initiated on the training data by calling the `fit` method on the `GridSearchCV` instance. After the grid search, the best parameters were extracted, signifying the optimal configuration for the stacked LSTM model.

A final instance of the stacked LSTM model was then built with the 'create_ensem_model' function using the best number of units from the hyperparameter tuning. It was trained on the training data, with the optimal batch size and number of epochs ascertained from the grid search. Optimal model configuration was found to have 64 LSTM units, 200 epochs, and a batch size of 4.

Ensemble model composed of stacked GRU layers.

In the first step, we define a function, `create_ensem_model`, to construct the stacked GRU model. This Sequential model comprises an initial GRU layer (set with 'units' GRU units), a subsequent GRU layer (configured with 'units' times 'multiply' GRU units), and a final Dense output layer. The preceding GRU layer is set to return sequences, being followed by another GRU layer. The output size of the Dense layer corresponds to 'output_dim'. We compile the model with the Adam optimizer, the Mean Squared Error loss function, and we also use Root Mean Squared Error as a performance metric.

The model is then instantiated using the `KerasRegressor` wrapper, utilizing the 'create_ensem_model' function for model definition, and setting epochs, batch size, and verbosity level.

Subsequently, we leverage the `GridSearchCV` function from Scikit-Learn to perform a grid search cross-validation for hyperparameter tuning, using negative mean squared error as a scoring metric. By invoking the fit method on the `GridSearchCV` instance, we initiate the grid search on the training dataset, parameter set is, 'units': [100, 150, 300], 'batch_size': [4, 16, 32], 'epochs': [100, 200, 300]. The next step involves extracting the best parameters resulting from the grid search process. These parameters represent the optimal configuration for the stacked GRU model.

Finally, a final instance of the stacked GRU model is constructed using the 'create_ensem_model' function with the best number of units as determined by the hyperparameter tuning. The model is trained on the training dataset, utilizing the optimal batch size and number of epochs derived from the grid search. The results of the hyperparameter tuning process indicate an optimal model configuration of 64 units for GRU layers, training across 200 epochs, with a batch size of 4.

XGBoost Model

The process begins by defining an instance of the XGBoost Regressor model, denoted as `best_xgb`.

Next, a dictionary, `parameters`, is created which encapsulates a variety of hyperparameters that will be used in the model tuning process. The dictionary includes model performance parameters ('nthread':[4], 'objective':['reg:linear'], 'eval_metric':['rmse', 'mae'], 'reg_alpha':[0, 1], 'reg_lambda' : [0, 1], 'learning_rate': [0.01, 0.03, 0.05, 0.07, 0.1], 'max_depth': [3, 5, 7, 9, 12, 15], 'min_child_weight': range(1, 6, 2), 'subsample': [0.5, 0.7], 'n_estimators': [100, 150, 300, 500]).

A `GridSearchCV` instance, referred to as `xgb_grid`, is then configured using the defined model and parameter dictionary. The count of cross-validation folds is set to 5, and the number of parallel jobs to run is 4. The verbose parameter is enabled for output logging. The grid search is performed on the training data by calling the fit method on the `xgb_grid` object.

Following this, a final XGBoost Regressor model instance, labeled as `xgb_`, is instantiated using the optimal hyperparameters that were discovered during the tuning process. These hyperparameters include a learning rate of 0.1, a maximum tree depth of 7, a minimum child weight of 3, and subsampling set to 0.7 with 100 boosted trees. This final model is trained on the training dataset using the fit method.

3.6 Model Evaluation

In this phase of the research, the performance of multiple models, including LSTM, GRU, Ensemble, Linear Regression, and XGBoost, is assessed against a test dataset. To evaluate the prediction accuracy, key metrics such as Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) are employed. An identical evaluation process is also applied to the XGBoost model, facilitating a direct comparison across all these models.

The process begins with the setup of the evaluation environment. Empty lists, namely `evaluation_results`, `evaluation_results_lr`, and `evaluation_results_xgb`, are established to compile the evaluation outcomes for the various models, the Linear Regression model, and the XGBoost model respectively. Concurrently, two more lists, `models` and `model_names`, are defined to hold the models and their respective names.

The evaluation proceeds with each model within the `models` list. During this stage, predictions are generated on the test dataset, which are subsequently inverse transformed to return them to their original scale. Then, the MAPE and RMSE between the true and the predicted values are computed. This outcome, along with the model's name, is appended to the `evaluation_results` list.

This evaluation framework is also extended to the Linear Regression and XGBoost models, with their results being stored in `evaluation_results_lr` and `evaluation_results_xgb` respectively.

At the culmination of this comprehensive process, the `evaluation_results`, `evaluation_results_lr`, and `evaluation_results_xgb` lists consist of the evaluation metrics for each model. These collated results form the basis of a comparative study

across the various models, thereby aiding in identifying the model that delivers the best performance.

3.7 Model Interpretation

This segment delves into model interpretation through forecasting and feature importance analysis in Linear Regression, LSTM, GRU, XGBoost, and Ensemble models.

The process is initiated by defining two data frames: `forecast_df` and `result_df`. The former is designated to hold the forecasted values from each model, while the latter is assigned to contain both actual and predicted values.

Next, the models are put to work for forecasting, each one following an identical procedure. Initially, the model generates predictions on the test data which are subsequently inverse transformed to match the original scale. These predictions are then reshaped and fed into the `forecast_df` DataFrame as new columns. Concurrently, the `result_df` DataFrame is supplemented with the inverse transformed actual test values.

Once the prediction process is complete, the actual and forecasted values are combined into a unified DataFrame by concatenating `y_test_df`, `forecast_df`, `forecast_df_lr`, and `forecast_df_xgb`. As a result, `result_df` offers a comprehensive view of the actual target values along with corresponding forecasts from each model, allowing for a comparative performance analysis.

Following this, visualization is used to present a more intuitive comparison of the forecasting results from various models against the actual values. This includes both individual and combined plots for all models, with the dates serving as the x-axis and the prices on the y-axis.

During the plotting phase, a list of model names and their corresponding color codes is established for visual distinctions. Next, dates for the test period are extracted and designated as the index of the `result_df`. Then, each model is presented with individual line plots, comparing the actual values with the forecasted ones, with a unique color distinguishing each model's forecast line. Lastly, a combined line plot is created for an overarching comparison of all models' forecasts against the actual values.

The final aspect of this section is the analysis of feature importance. A specially designed function `get_feature_importance` is employed to calculate feature importance based on the permutation importance method. In this method, the decrease in Mean Absolute Error (MAE) when the values of each feature are randomly shuffled helps determine the feature's importance.

Feature importance is computed for each model using the test data, following which the values are sorted in descending order. A DataFrame is then created to display the sorted feature names along with their respective importance values, which is visualized via a horizontal bar plot. Each bar represents a feature, with the bar length corresponding to the feature's importance. The baseline MAE helps determine the importance of a feature; a significant increase in MAE upon feature permutation indicates a high importance.

CHAPTER 4

RESULTS AND DISCUSSION

This section of the research offers a comprehensive analysis of our findings as we explored silver price forecasting through various statistical and deep learning models. By exploring the intricacies of the models employed, this analysis provides insights into the strengths and limitations of each model in handling time-series data and complex patterns. The study also focuses on critical assumptions of linear regression models and evaluates their validity. This detailed examination of the models' performance and characteristics provides valuable takeaways for stakeholders aiming to predict silver prices accurately. The subsequent discussion details the models' performance, their ability to handle time-series data, the challenges they face, and the implications of these findings on future research and forecasting attempts.

4.1 Results

Data Exploration

1. Trend of Variables

Figure 4.1: Time series of silver price and other variables over time with 7-day and 30-day moving averages.

Figure 4.1 illustrates the time series plots of various financial variables along with their 7-day and 30-day moving averages.

Post-2020, silver prices experienced a considerable transformation. The beginning of 2020 was marked by a severe dip, swiftly followed by an unprecedented surge, reaching a peak by year-end. This shift set a new course for silver prices which did not revert to the previous trend.

Examining these plots closely reveals that recent trends in these variables align more closely with their 7-day moving averages as compared to the 30-day ones.

This indicates the potential stronger influence of the most recent week's prices on current prices, perhaps due to fast-evolving market conditions.

Around 2020-05-18, a conspicuous event transpires: a drastic dip in oil prices, while other variables display minor shifts. This sharp drop may be associated with geopolitical happenings or shifts in supply-demand dynamics. Yet, despite this stark deviation in the oil market, other variables show relative stability, suggesting their lower susceptibility to oil market volatility.

The trends observed in inflation and interest rates along with their 7-day and 30-day moving averages provide intriguing insights. A significant decrease in interest rates in 2020 corresponds with a substantial decline in oil prices. Meanwhile, the inflation rate saw a considerable rise that continued till 2022. These patterns are in alignment with established economic theories related to monetary policies.

The decline in interest rates during 2020 could be attributed to strategic decisions by the Federal Reserve to mitigate the economic slump triggered by factors like the oil price drop. This decision, however, had inflationary implications. Lower interest rates usually expand money supply, potentially increasing demand that may exceed supply, thereby causing price inflation as observed from 2020 to 2022.

These patterns mirror the theoretical expectations of the impacts of monetary policies, demonstrating the intricate relationship between interest rates, inflation, and commodity prices like oil. They also emphasize the multi-factorial influences on these macroeconomic phenomena and the need to understand them within a wider economic context.

2. **Distribution of variables**

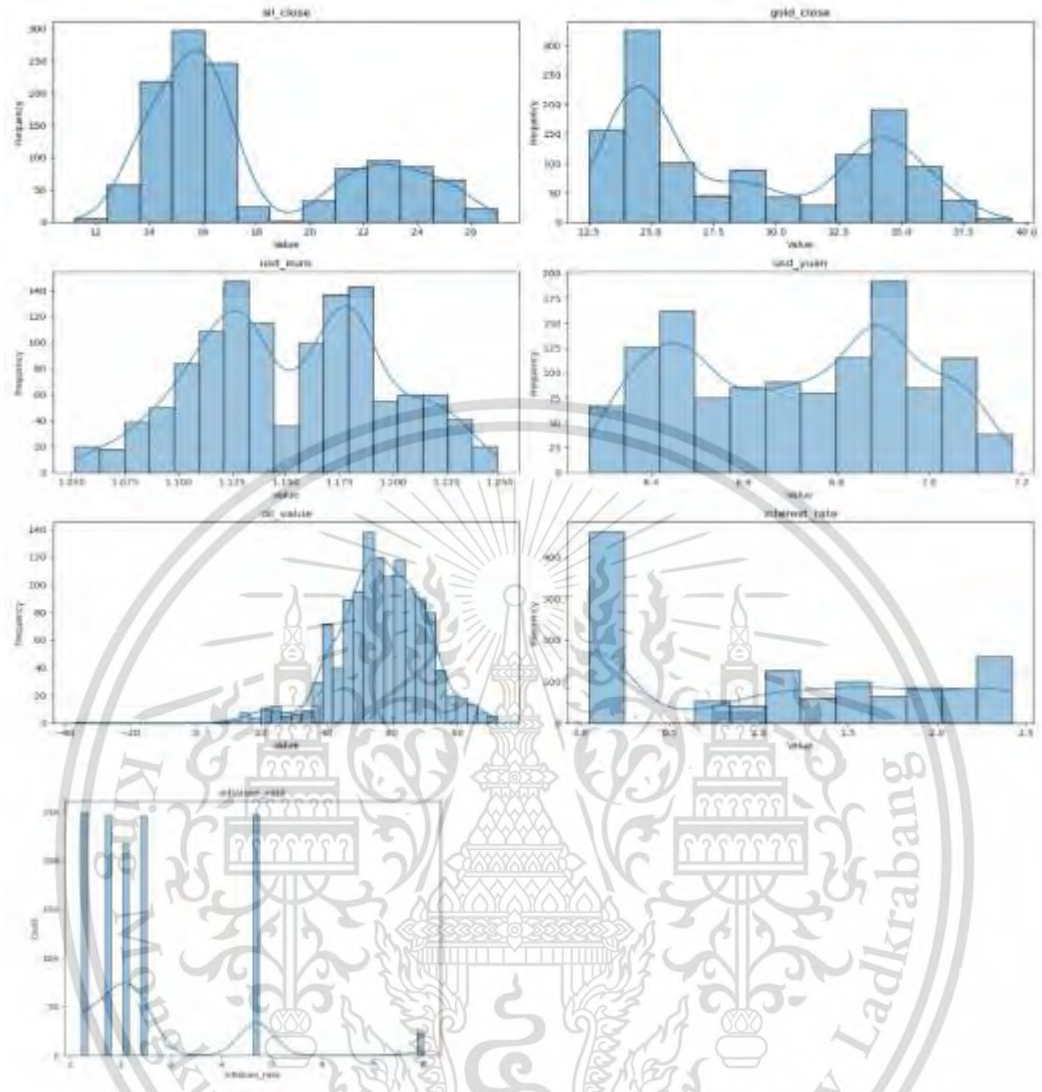

Figure 4.2: Distribution of variables

Upon evaluating the variable distributions presented in Figure 4.2, it becomes apparent that the oil price distribution deviates from normal distribution. Noteworthy are the outliers, which align with the period of a sharp decline in oil prices. While these outliers are statistically significant, they are not viewed as anomalies but rather as manifestations of exceptional global events - specifically, the fallout from the COVID-19 pandemic and the ensuing economic turmoil.

Recognizing these factors' influence on oil prices is critical for our analysis. By incorporating these extreme yet significant instances into our dataset, we intend to enable our predictive models to recognize and adapt to similar impactful events in

the future. Therefore, we have opted to preserve these outliers, valuing them as informative data points rather than anomalies to be dismissed.

This decision emphasizes the need for judicious data preprocessing, acknowledging that real-world scenarios. For this study, the oil price distribution serves dual purposes: it records the historical shifts in prices and illustrates the profound influence global events can exert on commodity markets.

3. Correlation Matrix Heat Map

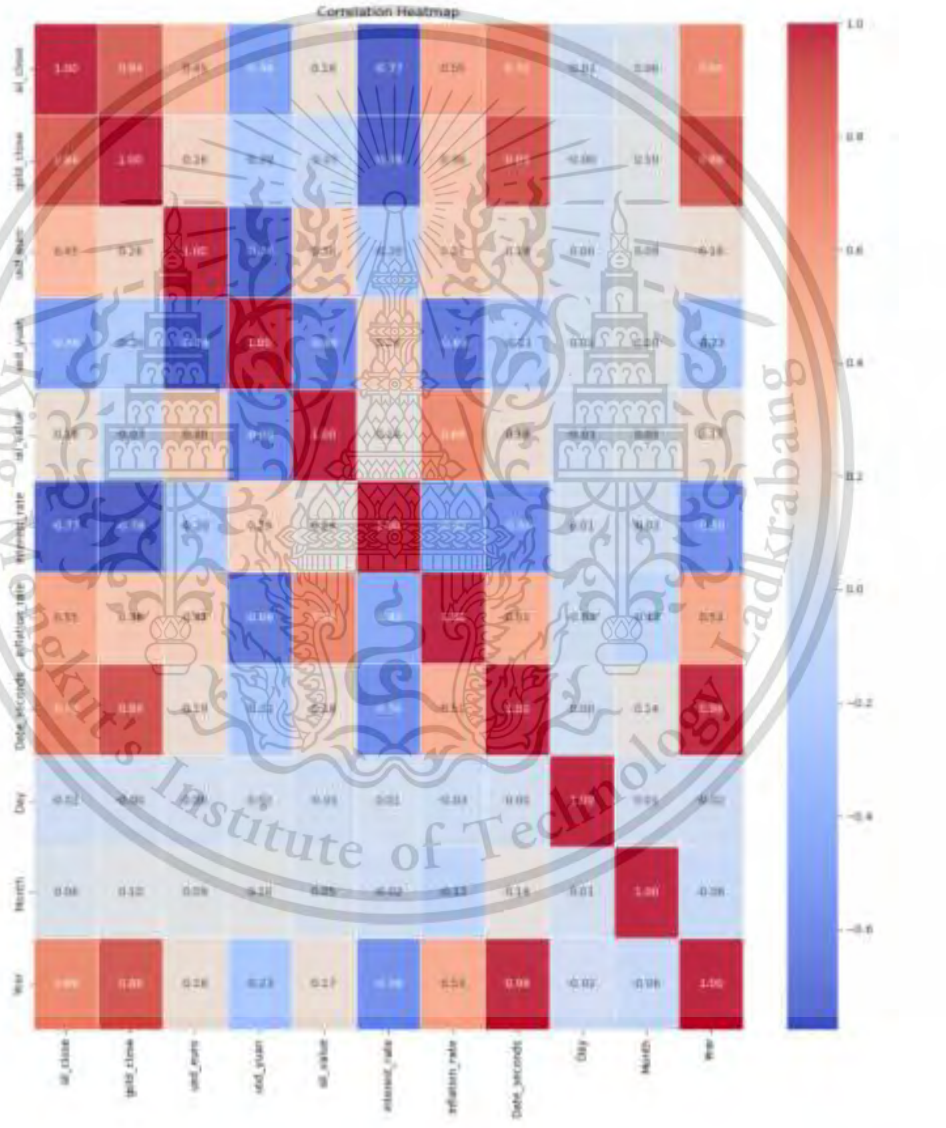

Figure 4.3: Correlation Matrix Heat Map

The correlation matrix, depicted as a heatmap in Figure 4.3, provides a numerical representation of the interrelations among various features in our dataset. One key observation from this matrix is the absolute correlation among selected

features falls below 0.8, a threshold we have set for our study. This correlation level indicates a moderate yet not excessively strong connection among the different variables. This approach helps us curtail the impact of multicollinearity, thereby enhancing the stability and interpretability of our machine learning models.

On the contrary, variables like 'Day', 'Month', 'Year', and 'Date_seconds' exhibit a correlation exceeding our predetermined limit. This high correlation signifies a significant amount of redundancy, multicollinearity, and autocorrelation. These are undesirable traits for our models, as they may lead to overfitting and hamper the interpretability of model results. Hence, these variables have been omitted from the model's feature set.

Significantly, the chosen features exhibit a correlation with the target variable that goes beyond our cutoff (correlation = -0.8 and 0.8). This denotes a substantial influence of these features on the target variable, affirming their relevance and significance in predicting the target.

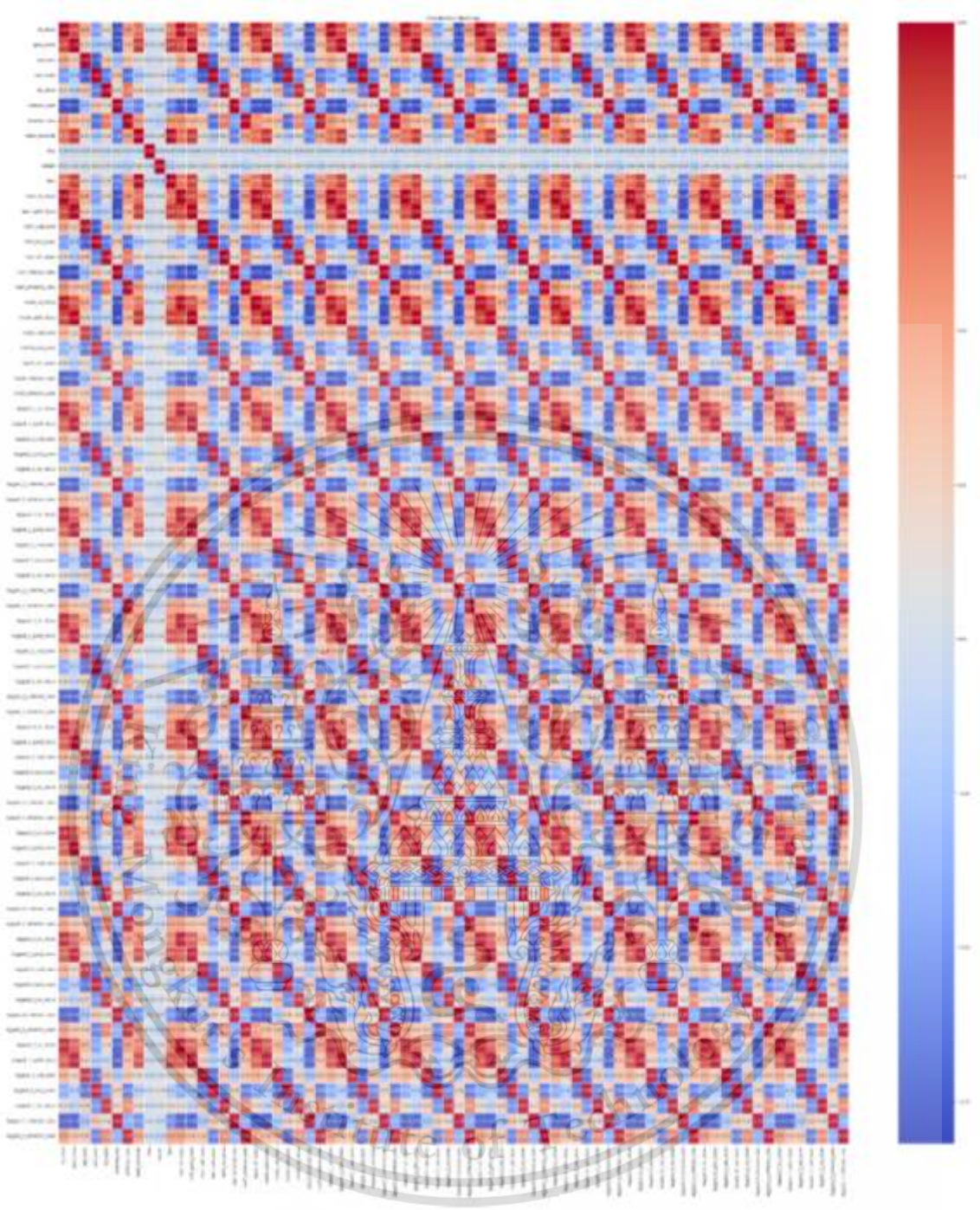

Figure 4.4: Correlation Matrix Heat Map with Feature Engineering

The correlation heatmap featuring engineered transformations such as lagged variables and moving averages was demonstrated in Figure 4.4. It reveals distinct instances of high correlations between variables. The varying degrees of color intensity (red and blue) in the heatmap visually signify these high correlation coefficients, indicating a potential risk of multicollinearity among the studied variables.

Linear Regression Assumptions

The validity of linear regression models is contingent on certain assumptions. This section examines these assumptions in the context of our model to predict silver prices, yielding the following results:

Linearity: The p-value of 0.9992, which is greater than 0.05, supports the assumption of linearity between the independent and dependent variables in our model. This suggests that changes in predictor variables are consistently associated with changes in the response variable, establishing a linear relationship in our model.

Independence of residuals: Our Durbin-Watson statistic of 1.9972 indicates the presence of positive autocorrelation. Despite this, we must consider the nature of our data where yesterday's price influences today's price, an inherent characteristic in financial time-series data. This introduces a degree of autocorrelation, even though it challenges the assumption of independence.

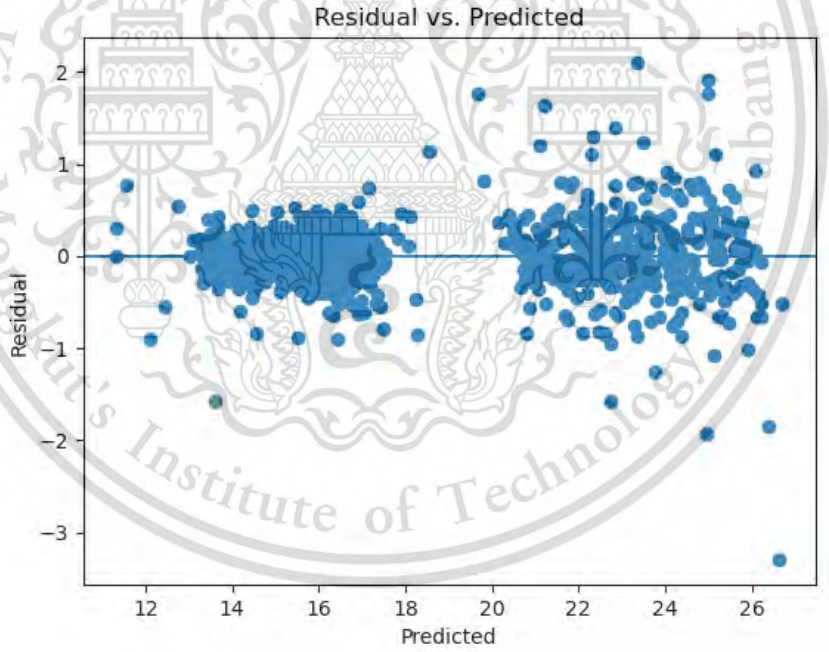

Figure 4.5: Homoscedasticity: Residual vs Predicted plot

Homoscedasticity: Upon observing the Residual vs. Predicted plot in Figure 4.5, we notice a division into two distinct clusters, indicative of heteroscedasticity. This reveals non-constant variance in our model's residuals, deviating from the assumption of homoscedasticity. Therefore, the model's performance might vary across different levels of our predictors, which could affect the reliability of our predictions.

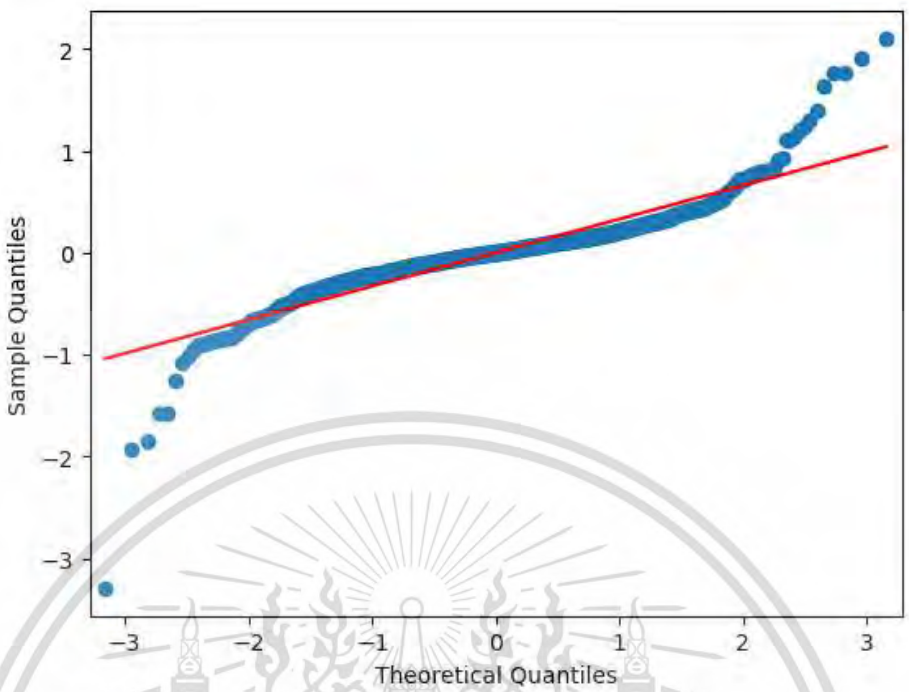

Figure 4.6: Normality: Q-Q plot

Normality of residuals: Both the Q-Q plot point towards non-normality in our residuals (Figure 4.6). This suggests that our model's residuals deviate from a normal distribution, potentially impacting the model's generalization capability and the robustness of statistical inferences derived from the model.

Table 4.1: Absence of multicollinearity: VIF Factor

features	VIF Factor	features	VIF Factor
lagged_1_sil_close	7593.95	lagged_4_usd_yuan	365040.23
lagged_1_gold_close	34679.84	lagged_4_oil_value	991.05
lagged_1_usd_euro	83562.60	lagged_4_interest_rate	3070.20
lagged_1_usd_yuan	200538.37	lagged_4_inflation_rate	916.71
lagged_1_oil_value	717.90	lagged_5_sil_close	13644.48
lagged_1_interest_rate	1557.36	lagged_5_gold_close	70129.94
lagged_1_inflation_rate	470.11	lagged_5_usd_euro	172418.71
lagged_2_sil_close	13431.64	lagged_5_usd_yuan	361160.42
lagged_2_gold_close	68736.60	lagged_5_oil_value	986.92
lagged_2_usd_euro	171999.61	lagged_5_interest_rate	3112.73
lagged_2_usd_yuan	361748.94	lagged_5_inflation_rate	915.81
lagged_2_oil_value	966.78	lagged_6_sil_close	13614.49
lagged_2_interest_rate	3065.10	lagged_6_gold_close	69779.49
lagged_2_inflation_rate	925.67	lagged_6_usd_euro	171971.52
lagged_3_sil_close	13627.15	lagged_6_usd_yuan	357087.18
lagged_3_gold_close	71026.80	lagged_6_oil_value	947.31
lagged_3_usd_euro	172926.59	lagged_6_interest_rate	3046.16
lagged_3_usd_yuan	365497.24	lagged_6_inflation_rate	916.63
lagged_3_oil_value	988.72	lagged_7_sil_close	7730.22
lagged_3_interest_rate	3057.50	lagged_7_gold_close	35816.34
lagged_3_inflation_rate	922.55	lagged_7_usd_euro	83156.34
lagged_4_sil_close	13617.48	lagged_7_usd_yuan	197501.34
lagged_4_gold_close	70377.59	lagged_7_oil_value	689.22
lagged_4_usd_euro	172547.33	lagged_7_interest_rate	1542.67
		lagged_7_inflation_rate	467.07

Absence of multicollinearity: Since all Variance Inflation Factor (VIF) values in Table 4.1 are greater than 1, there is evidence of multicollinearity in our model. This indicates a high correlation among our predictors, which could obscure the individual effects of predictors on the response variable and inflate the variance of the regression coefficients.

No endogeneity of regressors: Lastly, the mean of our residuals is a very small number, nearly zero, which satisfies the assumption that the error term has an expected value of zero. This indicates that there's no systematic bias in our error terms, which supports the model's overall reliability.

While our model adheres to the linearity and zero mean error term assumptions, it violates the assumptions of independence, homoscedasticity, normality, and absence of multicollinearity. This highlights the importance of carefully evaluating model assumptions and potentially considering other modeling techniques or additional preprocessing steps to better adhere to these assumptions when forecasting silver prices.

Performance of the Models in Comparison

1. **Evaluation Results**

Table 4.2: Evaluation Results of the models in MAPE and RMSE

Model	MAPE	RMSE
Linear(Baseline)	1.1566	0.3304
LSTM	3.5221	0.8745
GRU	2.6663	0.7212
Ensemble LSTM GRU	3.0191	0.7993
Ensemble LSTM LSTM	2.5217	0.7268
Ensemble GRU GRU	2.0413	0.5838
XGBoost	4.1206	0.9856

The evaluation of model performance results is summarized in Table 4.2 that presents the metrics - Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE).

A process of prediction and comparison was executed across the entire testing dataset unseen during model training. For every data point, the models generated predictions based on the historical lagged prices within the dataset. These forecasted

values were contrasted with the actual target values in the test dataset, and the ensuing differences were recorded as absolute percentage errors and squared errors. Once all test data points underwent this process, the collected errors were aggregated and utilized to compute the MAPE and RMSE.

The results revealed notable differences in the performance of the various state-of-art models. Particularly, the ensemble model that incorporated two Gated Recurrent Unit (GRU) layers delivered the most impressive results, posting the lowest MAPE of 2.0413 and RMSE of 0.5838 as shown in Table 4.2. This outcome indicates a higher predictive accuracy compared to the other models. Conversely, the XGBoost model posted the highest error values, with a MAPE of 4.1206 and RMSE of 0.9856, suggesting its relatively lower accuracy in this instance.

While the ensemble model with two Gated Recurrent Unit (GRU) layers demonstrates strong performance, it falls short when compared to the traditional Linear Regression model. Despite its relatively simple approach, Linear Regression manages to outperform the more complex ensemble model in this case, highlighting the value of traditional methods in certain contexts.

2. Model Interpretation

2.1 The actual versus predicted price plots

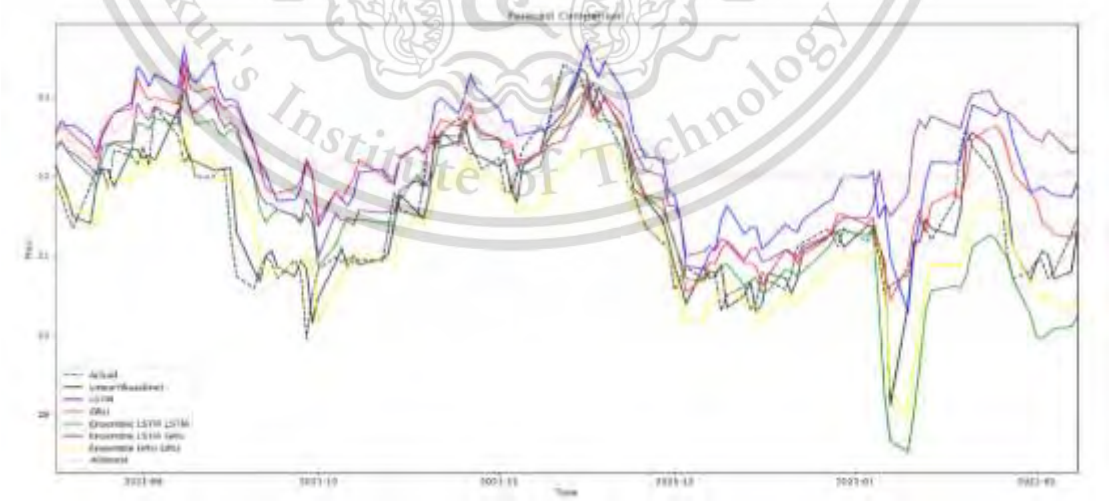

Figure 4.7: Forecasting Comparison

The visual comparison of the actual price (represented by a dashed black line) versus the predicted prices from each model powerfully illustrates the quantitative results that were derived. The level of congruity between the predicted line and the actual line provides the model's predictive accuracy (Figure 4.7). It's noteworthy that the Ensemble GRU with GRU model (displayed by a yellow line) closely aligns with the actual line, which is consistent with the evaluation results presented in Table 4.2.

Interestingly, between January and February 2022, the forecasted price pattern mirrors the significant drop in oil price that occurred in 2020, as reflected in the training dataset. However, this pattern isn't replicated in the predictions generated by the Linear Regression model (shown as a solid black line).

For the XGBoost model (represented by a pink line), it was observed that the model grapples with accurately tracing the price trend in this study.

Moreover, we observed that post-2022, the models' predictions failed to accurately mirror the actual prices or capture the evolving trend, a stark contrast to the period before 2022. This observation is consistent with the silver price graph developed during the data exploration phase, illustrating a disparity between predicted and actual prices in this later period.

2.2 Feature Importance

The significance of each feature within the models, quantified via feature importance scores and visualized in Figures 4.6 to 4.12, revealed insights of models.

Feature Importance Result: Linear Regression

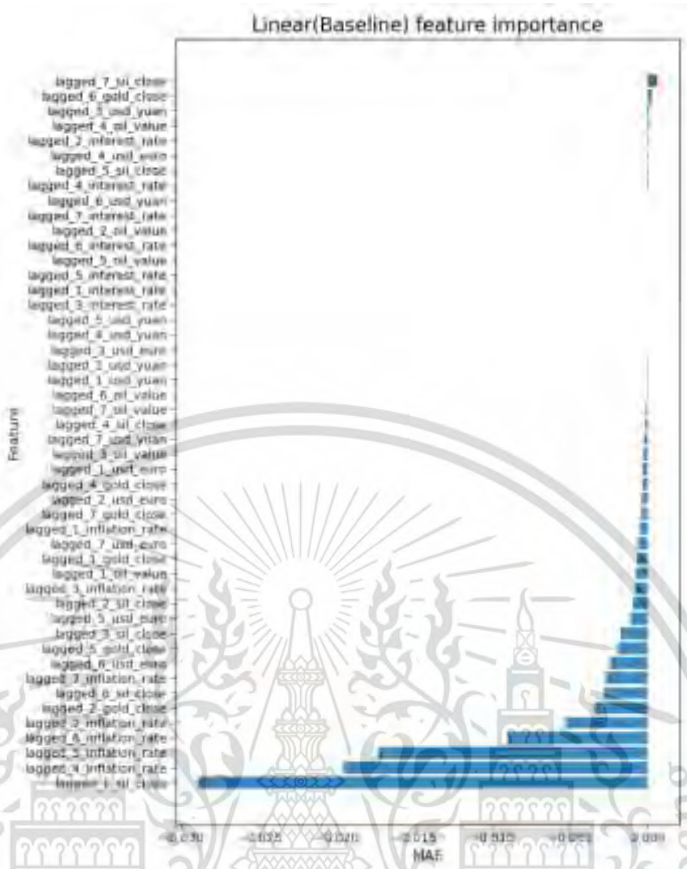

Figure 4.8: Feature Importance of Linear Regression Model

The feature importance of the Linear (Baseline) model as determined by Mean Absolute Error (MAE) revealed interesting patterns, sumarized the result in Figure 4.8. The most impactful feature was "lagged_1_sil_close," with the highest negative MAE value of -0.0289, suggesting a strong influence on model predictions. The importance of lagged inflation rate was also considerable, especially with a four to six-day lag.

Other influential features included lagged closing prices of gold and silver, and the lagged exchange rate between USD and Euro. Interestingly, the lagged oil values and interest rates had less impact on the model, with relatively smaller negative MAE values.

On the other hand, the least significant features in terms of MAE were lagged oil values, lagged interest rates, and lagged USD/Yuan exchange rates, as well as some lagged silver and gold closing prices.

Notably, while most features had negative MAE, a handful displayed positive MAE values, suggesting these variables may have counterintuitive or indirect effects on the model's output. These features included lagged oil values, lagged USD/Yuan and USD/Euro exchange rates, and lagged silver and gold closing prices.

Feature Importance Result: LSTM

Figure 4.9: Feature Importance of LSTM Model

In the LSTM model, the feature importance, as indicated by the Mean Absolute Error (MAE), has been shown in Figure 4.9. The most influential feature in the LSTM model was "lagged_1_inflation_rate" with a positive MAE value of 0.0022, which indicated its considerable positive impact on the model's predictions.

Other positively impacting features included the lagged exchange rates of USD/Euro and USD/Yuan. However, the influence of lagged interest rates was minimal, as suggested by the small negative MAE value.

Interestingly, "lagged_1_sil_close", which was the most influential feature in the Linear model, had a significantly higher negative MAE value in the LSTM model (-0.0133). This indicated a strong inverse relationship with the model's output.

Other features such as "lagged_1_oil_value" and "lagged_1_gold_close" also exhibited negative MAE values, suggesting that they have negative impacts on the model's predictions.

Overall, the LSTM model appeared to be more sensitive to the immediate past (lagged by one day), with lagged inflation rates and exchange rates.

Feature Importance Result: GRU

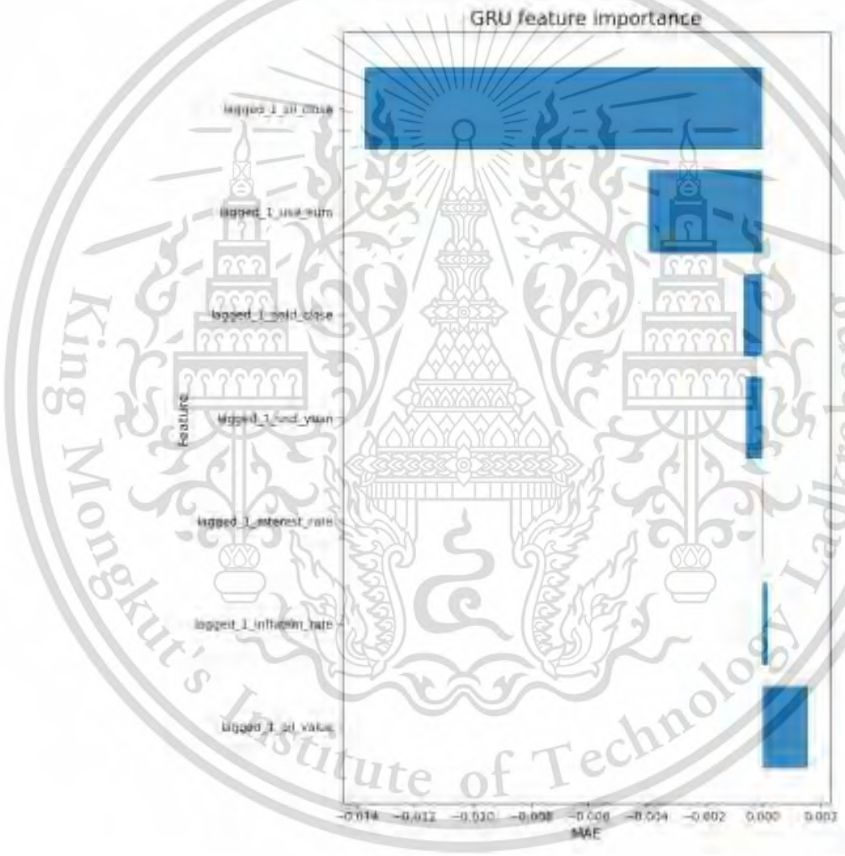

Figure 4.10: Feature Importance of GRU Model

In the case of the GRU model, the feature importance was depicted in Figure 4.10. Here, the feature "lagged_1_oil_value" emerged as the most influential one, displaying a positive Mean Absolute Error (MAE) of 0.0015, indicating that the model's predictions were largely influenced by the oil price from the previous day.

The "lagged_1_inflation_rate" feature also had a positive MAE, demonstrating its positive contribution to the model's output. Conversely, "lagged_1_interest_rate" had a marginal negative impact, as suggested by its small negative MAE value.

Interestingly, "lagged_1_usd_yuan", "lagged_1_gold_close", and "lagged_1_usd_euro" had negative MAE values, suggesting that these features inversely affected the model's predictions.

The most influential feature in the previous models, "lagged_1_sil_close", continued to have a strong negative influence in the GRU model as well, with a significantly high negative MAE of -0.014. This implies that the GRU model's output has a strong inverse relationship with the silver closing price of the previous day.

Overall, the GRU model demonstrated a higher sensitivity to immediate past data, particularly the previous day's oil prices and inflation rates, while also being significantly impacted by the silver closing price.

Feature Importance Result: Ensemble LSTM with LSTM

Figure 4.11: Feature Importance of Ensemble LSTM with LSTM Model

For the Ensemble LSTM model, the feature importance was depicted in Figure 4.11. In this model, "lagged_1_usd_yuan" and "lagged_1_usd_euro" were the most influential features, showing positive MAE values of 0.0014 and 0.0011, respectively.

This highlights the model's sensitivity towards recent changes in the USD to Yuan and Euro exchange rates.

The feature "lagged_1_inflation_rate" also contributed positively to the model's forecasts, as indicated by a positive MAE value. However, "lagged_1_interest_rate" demonstrated a very slight negative impact, suggested by its small negative MAE value.

On the other hand, the features "lagged_1_oil_value" and "lagged_1_gold_close" had negative influences on the model's prediction, as their negative MAE values indicate an inverse relationship with the model's output.

The silver closing price of the previous day, "lagged_1_sil_close", continued its trend of strong negative influence from the previous models, with the highest negative MAE of -0.01532445. This confirms its significant inverse relationship with the model's predictions.

Feature Importance Result: Ensemble GRU with GRU

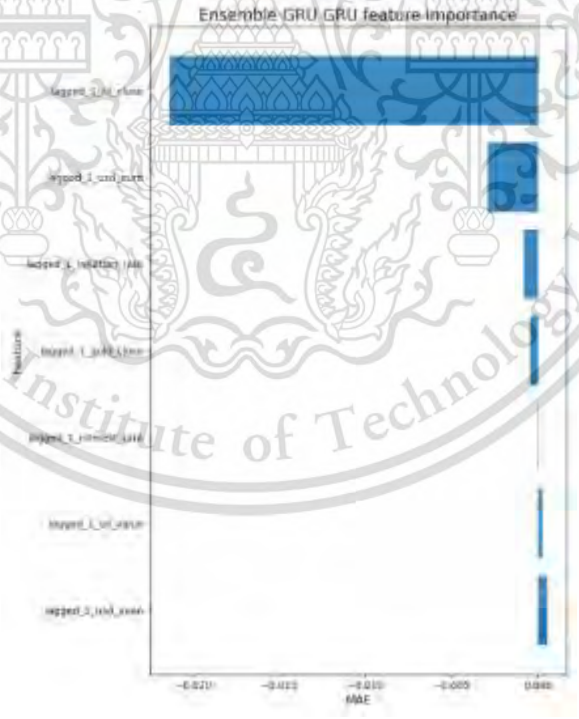

Figure 4.12: Feature Importance of Ensemble GRU with GRU Model

In the Ensemble GRU model, feature importance was depicted in Figure 4.12. The most influential feature, in a positive sense, was "lagged_1_usd_yuan" with an MAE of 0.0006.

The "lagged_1_oil_value" also demonstrated a modest positive influence on the model, denoted by an MAE of 0.0003. This indicates the model's consideration of recent oil price trends in its predictions.

Contrastingly, the model seemed to deem "lagged_1_interest_rate" as slightly negative but nearly negligible, as suggested by a small negative MAE value.

The features "lagged_1_gold_close", "lagged_1_inflation_rate", and "lagged_1_usd_euro" had negative impacts on the model's prediction.

Finally, the feature "lagged_1_sil_close" stood out with a highly negative MAE of -0.0214, the largest magnitude among all the features. This reflects a strong inverse relationship between the prior day's silver closing price and the model's prediction output.

Feature Importance Result: Ensemble LSTM with GRU

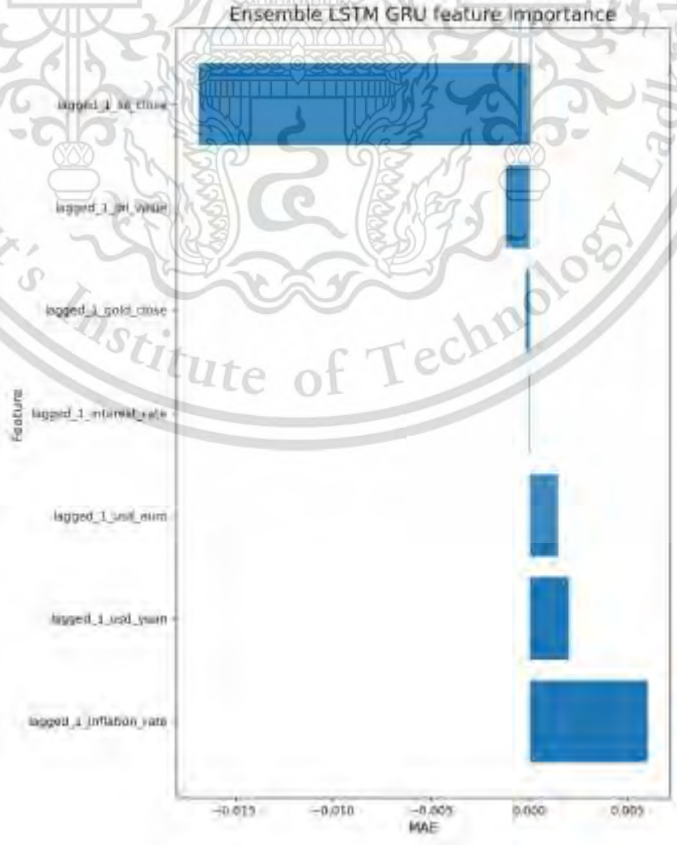

Figure 4.13: Feature Importance of Ensemble LSTM with GRU Model

In the Ensemble LSTM GRU model, feature importance results were demonstrated in Figure 4.13. The feature "lagged_1_inflation_rate" appeared to be the most influential with a positive Mean Absolute Error (MAE) of 0.0060.

Further, the model was positively affected by "lagged_1_usd_yuan" and "lagged_1_usd_euro", with their respective MAE values suggesting that the model took the prior day's exchange rates into account while making its predictions.

On the other hand, "lagged_1_interest_rate" had a negligible negative influence on the model. The model also demonstrated a negative response to the features "lagged_1_gold_close", "lagged_1_oil_value", and notably,"lagged_1_sil_close" which had the highest magnitude of MAE value at -0.0169.

In summary, the Ensemble LSTM GRU model's predictions seem to be primarily driven by recent inflation rates, exchange rates, and silver closing prices, with each of these factors influencing the model in different ways.

Feature Importance Result: XGBoost

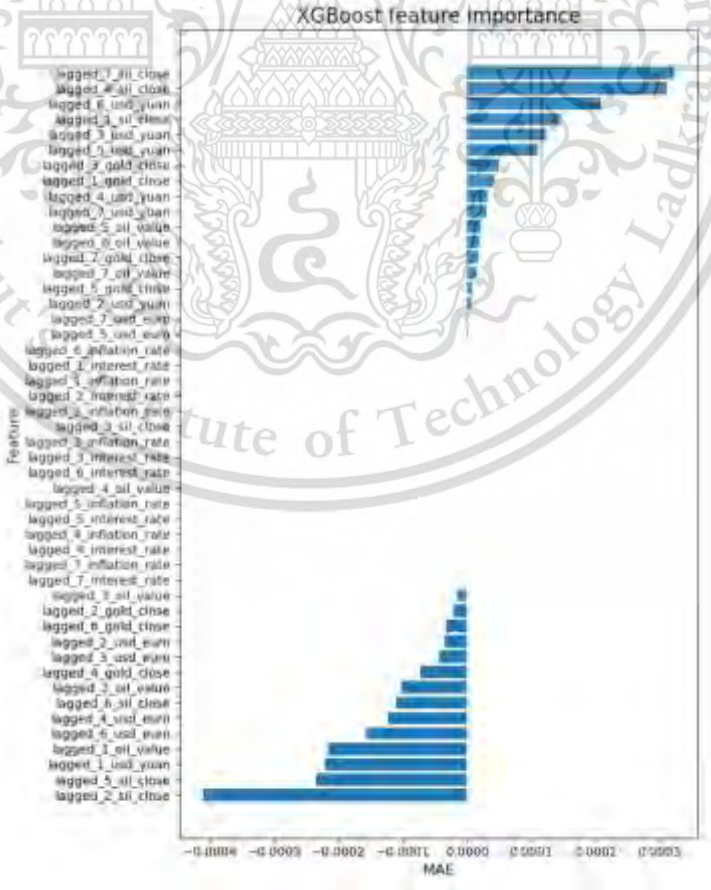

Figure 4.14: Feature Importance of XGBoost Model

In the XGBoost model, the feature importance results demonstrated in Figure 4.14. Negative influences on the model were led by "lagged_2_sil_close" with an MAE of -0.0004. Other notable features included "lagged_5_sil_close", "lagged_1_usd_yuan", and "lagged_1_oil_value", suggesting the model's predictions decrease with an increase in these features.

Interestingly, several features, including various inflation rates, interest rates, and closing prices, registered zero feature importance, indicating no discernible impact on the model's predictions.

On the contrary, a few factors registered a positive influence on the model. "Lagged_7_sil_close" exhibited the strongest positive effect with an MAE of 0.0003, followed by "lagged_4_sil_close" and "lagged_6_usd_yuan".

4.2 Discussion

In this research, we employed a variety of models, including linear regression, deep learning techniques such as LSTM and GRU, ensemble models, and XGBoost to predict silver prices.

Starting with linear regression, it showcased a commendable predictive performance, especially its capability to highlight the impact of macroeconomic factors like inflation on silver prices. Yet, its limitations in identifying long-term trends suggest it might not be the optimal choice as a standalone model for time-series forecasts.

Deep learning models, LSTM, and GRU, shone through their ability to detect complex temporal relationships within the data, thus surpassing linear regression in this regard. However, their susceptibility to overfitting due to the high-dimensional nature of the data and the complexity inherent to these models underlines the importance of judicious hyperparameter tuning.

Interestingly, the ensemble models, which amalgamate the merits of various models, showcased competitive performance, indicating the potential benefits of such an approach. Specifically, the ensemble model featuring two GRU models recorded the lowest RMSE, suggesting that harnessing similar architectures could enhance predictive capabilities.

In contrast, the XGBoost model underperformed compared to the other models, hinting that gradient boosting might struggle to capture complex temporal patterns within time-series data. This highlights the significance of model selection based on the data characteristics and the task at hand.

In addition, we observed a noticeable shift in the models' predictive capabilities post-2022, where they failed to closely track the actual prices or identify the evolving trend, a significant departure from the accuracy seen in the period leading up to 2022. This aligns with the trends observed in the silver price graph generated during our data exploration process, emphasizing the models' difficulty in accurately forecasting prices in this later period.

CHAPTER 5

CONCLUSION

This section encapsulates the significant findings from our research into silver price prediction, utilizing a variety of machine learning and deep learning models. The insights gleaned from the study uncover the models' unique attributes, strengths, and areas of improvement in predicting silver prices. From the limitations of linear regression in recognizing long-term trends to the proficiency of LSTM and GRU in managing complex temporal dependencies, our findings shed light on the pivotal role of discerning model selection and hyperparameter tuning. The report also highlights future avenues of exploration that could enhance the precision and robustness of silver price forecasts. This includes the incorporation of diverse features and events influencing silver prices, broadening the timeframes, and experimenting with various recurrent neural network architectures. These findings and recommendations pave the way for more refined and accurate silver price prediction models in future research.

5.1 Key Findings

In our study, we delved into silver price prediction using an array of machine learning and deep learning models. This exercise revealed the strengths, limitations, and unique merits of each model.

Linear Regression, though excellent for simple predictions, assumes a consistent relationship over time between variables. This feature can limit its utility in scenarios with abrupt trend changes, as witnessed in May 2020. Even though it captures the effect of pivotal factors like inflation effectively, the model's ability to retain long-term trends is somewhat limited.

On the other hand, models such as LSTM and GRU, famed for their proficiency in managing complex temporal dependencies, outperformed Linear Regression in recognizing long-term patterns. These models, with their inherent memory

mechanisms, adapt more fluidly to trend changes, giving them an edge in times of significant market shift.

We found XGBoost, while powerful in identifying non-linear relationships, has shortcomings in time-series prediction. It doesn't natively consider the chronological ordering of data, which could hinder its predictive accuracy during trend changes.

Interestingly, the ensemble models amalgamating GRUs demonstrated the compelling potential of recurrent architectures in time-series prediction. The ensemble's exceptional performance underscores the impact of blending similar architectures for enhanced prediction capabilities.

Our feature engineering process was instrumental in these outcomes. We created multiple lagged features to factor in historical data. While recent prices were vital for future predictions, it became evident that creating lagged features for Linear Regression and XGBoost could lead to redundancy.

Key variables influencing silver prices included the silver price itself, USD exchange rate, gold price, and oil price. However, we also recognized that model performance could significantly vary based on data constraints and hyperparameters, underscoring the need for robust tuning strategies.

Ultimately, our findings reiterate the importance of discerning model selection, bearing in mind aspects like model complexity, overfitting risk, and the nature of input data. Moreover, while our models adeptly predicted the next day's price, questions arose about their efficacy in forecasting prices over longer durations, especially beyond 2022.

In conclusion, our study highlights the importance of careful model selection, taking into account model complexity, overfitting risk, and input data type. It presents a rich exploration into silver price prediction using machine learning and deep learning, pointing out potential directions for future research.

5.2 Further Study

Upon interpreting our models, we found the variable prices trend was captured effectively by deep learning models, potentially benefiting long-term predictions.

This study did not consider certain features such as labels denoting specific periods, such as the COVID-19 pandemic or economic crises, which could be important factors influencing silver prices. Extending the timeframe beyond 2022 might also reveal new patterns and contribute to improving the model's performance.

Future studies could expand their focus from forecasting the silver price of the subsequent day to predicting prices over extended periods, ranging from 15 or 30 days to an entire year, or even up to five years. This approach would reveal the long-term forecasting capabilities of these models.

Furthermore, future research could experiment with incorporating various types of Recurrent Neural Networks into ensemble models or explore other models adept at efficiently handling time-series data. These investigations should particularly target models that can account for the sequential data nature, where past price information impacts future prices.

In the quest to enhance model performance, it would be worth considering techniques to reduce data complexity and oppose overfitting. This can be achieved through feature selection to minimize dimensionality and applying regularization techniques to mitigate overfitting.

Moreover, upcoming studies could consider gathering more diverse features, including notable events like pandemics (for instance, COVID-19) or economic crises, which might impact silver price significantly. An extended data collection timeframe stretching beyond 2022 might also be beneficial to capture the emerging trends in silver prices.

REFERENCES

Clements, M. P., & Galvao, A. B. (2008). Macroeconomic Forecasting with Mixed Frequency Data: Forecasting Output Growth in the United States. Journal of Business & Economic Statistics, 26(4), 546-554.

D. V. A. Pradeep Kumar Mahato (2014), "Prediction of gold and silver stock price using ensemble models," in *2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014)*.

Edel Tully, B. M. (2007). A power GARCH examination of the gold market. *21*(2).

Felix A. Gers, J. S. (2000). Learning to Forget: Continual Prediction with LSTM. *12*.

Forrest Capie, T. C. (2005). Gold as a hedge against the dollar. *15*(4).

Frankel, J. A. (1984). Commodity Prices and Money: Lessons from International Finance. *66*(5).

Frankel, J. A. (2006). *The Effect of Monetary Policy on Real Commodity Prices.* NATIONAL BUREAU OF ECONOMIC RESEARCH.

Gareth James, D. W. (2013). *An Introduction to Statistical Learning With Applications in R.* Springer New York.

Ian Goodfellow, Y. B. (2016). *Deep Learning.* MIT Press.

Ilya Sutskever, O. V. (2014). Sequence to Sequence Learning with Neural Networks.

James Bergstra, Y. B. (2012). Random Search for Hyper-Parameter Optimization. *13*.

Kyunghyun Cho, B. v. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.

P. Sadorsky (2021), "Predicting Gold and Silver Price Direction Using," *Journal of Risk and Financial Management,* vol. 14, p. 198.

Ramazan Sari, S. H. (2010). Dynamics of oil price, precious metal prices, and exchange rate. *32*(2).

Reboredo, J. C. (2013). Is gold a safe haven or a hedge for the US dollar? Implications for risk management. *37*(8).

Sepp Hochreiter, J. S. (1997). Long Short-Term Memory.

Spyros Makridakis, E. S. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward.

Tianqi Chen, C. G. (2016). XGBoost: A Scalable Tree Boosting System.

T. L. J. C. a. C. F. Jiahao chena (2022), "Copper Price Prediction Using Lstm Recurrent Neural Network and Simulation Annealing Algorithm," Social Science Research Network (SSRN).

TSAY, R. S. (2005). *Analysis of Financial Time Series.* John Wiley & Sons, Inc.

Wei Bao, J. Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory.

Zhou, Z.-H. (2012). *Ensemble Methods: Foundations and Algorithms.* CRC Press.

Appendix

Table A: Comparison of model-predicted versus actual silver prices for 1-day ahead forecast from August 17, 2021, to February 8, 2022.

Date	Actual	LSTM	GRU	Ensemble LSTM GRU	Ensemble LSTM LSTM	Ensemble GRU GRU	Linear (Baseline)	XGBoost
8/17/2021	21.90	22.52	22.60	22.38	22.34	21.72	22.15	22.72
8/18/2021	21.76	22.70	22.63	22.42	22.45	21.87	21.94	22.55
8/19/2021	21.51	22.63	22.59	22.33	22.32	21.70	21.76	22.48
8/20/2021	21.35	22.65	22.51	22.29	22.21	21.64	21.49	22.33
8/23/2021	21.88	22.48	22.33	22.12	22.02	21.50	21.41	22.20
8/24/2021	22.08	22.30	22.21	22.00	21.87	21.35	21.91	22.41
8/25/2021	22.09	22.56	22.51	22.39	22.22	21.70	22.10	22.69
8/26/2021	21.85	22.69	22.65	22.40	22.38	21.89	22.08	22.76
8/27/2021	22.33	22.80	22.80	22.43	22.41	21.98	21.87	22.54
8/30/2021	22.26	22.93	22.85	22.43	22.32	21.93	22.34	22.78
8/31/2021	22.14	23.38	23.23	22.92	22.67	22.24	22.34	22.79
9/1/2021	22.37	23.29	23.07	22.79	22.66	22.12	22.22	22.79
9/2/2021	22.14	23.13	22.96	22.71	22.59	22.06	22.26	22.80
9/3/2021	22.88	23.29	22.99	22.88	22.73	22.25	22.20	22.90
9/7/2021	22.51	23.12	22.91	22.72	22.64	22.13	22.84	23.10
9/8/2021	22.17	23.63	23.45	23.27	23.10	22.63	22.43	22.80
9/9/2021	22.27	23.25	23.11	22.84	22.86	22.29	22.21	22.75
9/10/2021	21.99	23.18	23.06	22.81	22.66	22.13	22.19	22.71
9/13/2021	21.99	23.45	23.13	23.00	22.71	22.25	22.08	22.66
9/14/2021	22.10	23.11	22.93	22.74	22.54	21.97	22.09	22.68
9/15/2021	22.07	22.95	22.84	22.62	22.51	21.91	22.09	22.73
9/16/2021	21.25	22.98	22.90	22.68	22.59	21.99	22.10	22.75
9/17/2021	20.74	22.94	22.87	22.62	22.55	21.95	21.29	22.45
9/20/2021	20.59	22.41	22.37	22.01	21.97	21.37	20.81	22.23
9/21/2021	20.84	22.17	22.12	21.78	21.58	21.00	20.68	22.30
9/22/2021	20.99	22.05	21.85	21.66	21.42	20.76	20.99	22.25
9/23/2021	20.91	21.84	21.80	21.64	21.51	20.75	21.06	22.27
9/24/2021	20.71	21.69	21.78	21.58	21.58	20.81	20.85	22.27
9/27/2021	20.94	21.70	21.85	21.51	21.50	20.88	20.74	22.28
9/28/2021	20.78	21.84	21.93	21.44	21.40	20.89	20.94	22.19
9/29/2021	19.95	22.18	22.22	21.71	21.54	21.04	20.83	22.21
9/30/2021	20.52	22.03	22.03	21.56	21.41	20.83	20.15	22.19
10/1/2021	20.83	21.37	21.46	20.99	20.86	20.18	20.47	22.03
10/4/2021	20.98	21.81	21.80	21.53	21.13	20.59	20.92	22.13
10/5/2021	20.92	21.71	21.70	21.44	21.33	20.71	21.10	22.24
10/6/2021	20.96	21.63	21.85	21.45	21.48	20.88	20.85	22.19
10/7/2021	20.90	21.75	21.97	21.38	21.45	20.95	21.00	22.10
10/8/2021	20.95	22.05	22.21	21.57	21.46	21.04	20.89	22.14
10/12/2021	20.93	22.08	22.02	21.54	21.41	20.91	20.96	22.12
10/13/2021	21.33	21.98	21.92	21.56	21.42	20.94	21.02	22.12

Table A: Comparison of model-predicted versus actual silver prices for 1-day ahead forecast from August 17, 2021, to February 8, 2022. (Continue)

Date	Actual	LSTM	GRU	Ensemble LSTM GRU	Ensemble LSTM LSTM	Ensemble GRU GRU	Linear (Baseline)	XGBoost
10/14/2021	21.80	21.89	21.96	21.52	21.42	20.93	21.40	22.11
10/15/2021	21.58	22.22	22.25	21.83	21.71	21.23	21.77	22.21
10/18/2021	21.49	22.38	22.36	21.95	22.02	21.49	21.54	22.13
10/19/2021	21.89	22.29	22.28	21.87	21.91	21.39	21.46	22.16
10/20/2021	22.53	22.47	22.36	21.97	21.87	21.49	21.76	22.46
10/21/2021	22.35	22.91	22.57	22.41	22.10	21.77	22.53	22.77
10/22/2021	22.54	22.99	22.71	22.56	22.52	22.05	22.36	22.82
10/25/2021	22.74	22.79	22.63	22.38	22.50	21.88	22.44	23.02
10/26/2021	22.37	22.99	22.83	22.56	22.63	22.05	22.72	23.05
10/27/2021	22.29	23.28	22.92	22.76	22.77	22.25	22.39	22.64
10/28/2021	22.27	23.11	22.66	22.61	22.57	22.02	22.26	22.75
10/29/2021	22.09	23.05	22.53	22.56	22.47	22.00	22.24	22.86
11/1/2021	22.23	22.85	22.48	22.46	22.46	21.95	22.12	22.56
11/2/2021	21.80	22.68	22.43	22.37	22.26	21.73	22.22	22.63
11/3/2021	21.81	22.71	22.41	22.37	22.34	21.76	21.85	22.33
11/4/2021	22.00	22.51	22.20	22.12	22.13	21.58	21.67	22.43
11/5/2021	22.35	22.52	22.20	22.17	22.03	21.55	22.13	22.65
11/8/2021	22.65	22.58	22.30	22.29	22.12	21.60	22.30	22.63
11/9/2021	22.53	22.57	22.44	22.34	22.37	21.71	22.58	22.79
11/10/2021	22.87	22.78	22.52	22.40	22.59	21.96	22.64	22.78
11/12/2021	23.42	22.89	22.72	22.45	22.62	22.10	22.80	22.88
11/15/2021	23.24	23.42	23.02	22.88	22.77	22.36	23.36	23.61
11/16/2021	22.96	23.66	23.08	23.15	23.02	22.46	23.29	23.38
11/17/2021	23.17	23.40	22.74	22.93	22.94	22.29	22.86	23.43
11/18/2021	22.96	23.24	22.87	22.79	22.69	22.22	23.11	23.51
11/19/2021	22.75	23.45	22.99	23.02	22.72	22.38	22.97	23.50
11/22/2021	22.34	23.12	22.58	22.79	22.60	22.18	22.63	22.92
11/23/2021	21.92	22.94	22.42	22.63	22.38	21.96	22.22	22.47
11/24/2021	21.79	22.56	22.13	22.35	22.01	21.62	21.83	22.55
11/26/2021	21.37	22.24	21.76	22.07	21.74	21.42	21.73	22.42
11/29/2021	21.16	22.21	21.71	22.00	21.58	21.31	21.44	22.31
11/30/2021	21.06	21.81	21.39	21.63	21.39	21.01	21.18	22.10
12/1/2021	20.57	21.76	21.18	21.63	21.15	20.81	20.87	22.02
12/2/2021	20.69	21.48	20.90	21.47	20.97	20.49	20.64	22.08
12/3/2021	20.85	20.86	20.53	21.00	20.71	20.15	20.39	22.08
12/6/2021	20.71	20.83	20.73	21.05	20.67	20.19	20.87	22.10
12/7/2021	20.81	21.04	20.91	21.12	20.80	20.39	20.72	22.02
12/8/2021	20.76	21.07	20.90	21.07	20.79	20.39	20.74	22.04
12/9/2021	20.30	21.46	21.08	21.21	20.86	20.56	20.89	22.04
12/10/2021	20.50	21.63	21.16	21.18	20.89	20.65	20.34	22.01
12/13/2021	20.61	21.27	20.90	20.95	20.62	20.37	20.50	22.00
12/14/2021	20.31	21.35	20.91	21.06	20.68	20.37	20.67	22.01

Table A: Comparison of model-predicted versus actual silver prices for 1-day ahead forecast from August 17, 2021, to February 8, 2022. (Continue)

Date	Actual	LSTM	GRU	Ensemble LSTM GRU	Ensemble LSTM LSTM	Ensemble GRU GRU	Linear (Baseline)	XGBoost
12/15/2021	20.41	21.36	20.86	21.10	20.73	20.37	20.29	22.00
12/16/2021	20.79	21.08	20.62	20.92	20.53	20.14	20.39	22.03
12/17/2021	20.68	21.15	20.74	20.94	20.55	20.22	20.77	22.06
12/20/2021	20.57	21.39	21.07	21.07	20.82	20.52	20.72	21.99
12/21/2021	20.77	21.34	21.06	21.02	20.81	20.46	20.52	22.01
12/22/2021	21.10	21.29	20.90	20.97	20.73	20.39	20.69	22.01
12/23/2021	21.16	21.45	21.01	21.12	20.81	20.51	21.06	22.07
12/27/2021	21.33	21.67	21.20	21.24	21.07	20.77	21.22	22.06
12/28/2021	21.27	21.70	21.33	21.29	21.17	20.86	21.29	21.98
12/29/2021	21.11	21.86	21.52	21.41	21.32	21.00	21.29	21.97
12/30/2021	21.34	22.00	21.51	21.48	21.31	21.01	21.10	22.00
1/3/2022	21.18	21.99	21.39	21.48	21.25	20.95	21.37	22.00
1/4/2022	21.32	22.07	21.48	21.62	21.36	21.05	21.38	22.00
1/5/2022	21.00	21.47	21.12	22.07	20.40	21.11	21.14	21.98
1/6/2022	20.51	21.63	20.79	21.65	19.24	20.70	20.92	21.97
1/7/2022	20.62	20.96	20.45	21.50	18.66	19.31	19.12	22.01
1/10/2022	20.76	20.28	20.87	21.78	18.52	18.94	20.37	22.00
1/11/2022	21.05	21.05	21.36	22.49	19.00	19.59	21.17	22.00
1/12/2022	21.45	21.46	21.58	22.71	19.61	20.11	21.17	22.07
1/13/2022	21.33	21.94	21.30	22.61	20.30	20.61	21.45	22.03
1/14/2022	21.20	22.19	21.65	22.76	20.56	20.89	21.34	21.98
1/18/2022	21.69	22.16	21.80	22.63	20.61	20.89	21.25	22.02
1/19/2022	22.37	22.21	21.71	22.60	20.59	20.87	21.55	21.97
1/20/2022	22.62	22.58	22.00	22.87	20.76	21.19	22.45	22.53
1/21/2022	22.38	22.90	22.46	23.03	21.10	21.58	22.55	22.61
1/24/2022	22.11	22.80	22.61	23.08	21.26	21.66	22.38	22.60
1/25/2022	22.02	22.76	22.63	22.93	21.21	21.61	22.17	22.65
1/26/2022	21.72	22.84	22.55	22.84	21.08	21.54	21.96	22.61
1/27/2022	21.02	22.81	22.33	22.81	20.97	21.39	21.73	22.38
1/28/2022	20.71	22.46	22.03	22.77	20.70	21.08	21.07	22.17
1/31/2022	20.80	21.81	21.79	22.51	20.25	20.66	20.69	22.09
2/1/2022	20.94	21.82	21.69	22.47	19.98	20.46	20.91	22.03
2/2/2022	20.93	21.87	21.41	22.44	19.96	20.49	21.06	21.98
2/3/2022	20.71	21.81	21.37	22.57	20.01	20.49	21.01	22.03
2/4/2022	20.79	21.77	21.26	22.51	20.08	20.43	20.71	22.03
2/7/2022	21.27	21.74	21.23	22.30	20.10	20.32	20.79	22.04
2/8/2022	21.45	21.93	21.29	22.32	20.22	20.48	21.32	22.01

Table A: The initial 10 days of the dataset, post feature engineering.

Date	1/12/2017	1/13/2017	1/17/2017	1/18/2017	1/19/2017	1/23/2017	1/24/2017	1/25/2017	1/26/2017	1/27/2017
sil_close	15.91	15.94	16.28	16.19	16.12	16.29	16.22	16.10	15.93	16.22
gold_close	23.02	23.10	23.42	23.20	23.18	23.40	23.30	23.12	22.88	22.92
usd_euro	1.07	1.06	1.07	1.07	1.06	1.07	1.07	1.07	1.07	1.07
usd_yuan	6.89	6.90	6.85	6.84	6.87	6.85	6.86	6.88	6.88	6.88
oil_value	53.01	52.36	52.45	51.12	51.39	52.77	52.38	52.14	53.24	53.18
interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
Date_seconds	777600.00	864000.00	1209600.00	1296000.00	1382400.00	1728000.00	1814400.00	1900800.00	1987200.00	2073600.00
Day	12	13	17	18	19	23	24	25	26	27
Month	1	1	1	1	1	1	1	1	1	1
Year	2017	2017	2017	2017	2017	2017	2017	2017	2017	2017
ma7_sil_close	15.70	15.77	15.82	15.89	15.97	16.03	16.09	16.14	16.16	16.16
ma7_gold_close	22.67	22.77	22.86	22.96	23.04	23.10	23.18	23.23	23.25	23.21
ma7_usd_euro	1.05	1.06	1.06	1.06	1.06	1.06	1.06	1.07	1.07	1.07
ma7_usd_yuan	6.93	6.92	6.91	6.91	6.90	6.89	6.88	6.86	6.86	6.86
ma7_oil_value	52.62	52.71	52.58	52.39	51.99	51.91	52.18	52.21	52.09	52.21
ma7_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
ma7_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_1_sil_close	15.87	15.91	15.94	16.28	16.19	16.12	16.29	16.22	16.10	15.93
lagged_1_gold_close	22.92	23.02	23.10	23.42	23.20	23.18	23.40	23.30	23.12	22.88
lagged_1_usd_euro	1.05	1.07	1.06	1.07	1.07	1.06	1.07	1.07	1.07	1.07
lagged_1_usd_yuan	6.94	6.89	6.90	6.85	6.84	6.87	6.85	6.86	6.88	6.88
lagged_1_oil_value	52.19	53.01	52.36	52.45	51.12	51.39	52.77	52.38	52.14	53.24
lagged_1_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_1_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_2_sil_close	15.91	15.87	15.91	15.94	16.28	16.19	16.12	16.29	16.22	16.10
lagged_2_gold_close	22.88	22.92	23.02	23.10	23.42	23.20	23.18	23.40	23.30	23.12
lagged_2_usd_euro	1.06	1.05	1.07	1.06	1.07	1.07	1.06	1.07	1.07	1.07
lagged_2_usd_yuan	6.92	6.94	6.89	6.90	6.85	6.84	6.87	6.85	6.86	6.88
lagged_2_oil_value	50.82	52.19	53.01	52.36	52.45	51.12	51.39	52.77	52.38	52.14
lagged_2_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_2_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_3_sil_close	15.70	15.91	15.87	15.91	15.94	16.28	16.19	16.12	16.29	16.22
lagged_3_gold_close	22.76	22.88	22.92	23.02	23.10	23.42	23.20	23.18	23.40	23.30
lagged_3_usd_euro	1.06	1.06	1.05	1.07	1.06	1.07	1.07	1.06	1.07	1.07
lagged_3_usd_yuan	6.94	6.92	6.94	6.89	6.90	6.85	6.84	6.87	6.85	6.86
lagged_3_oil_value	51.95	50.82	52.19	53.01	52.36	52.45	51.12	51.39	52.77	52.38
lagged_3_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_3_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_4_sil_close	15.64	15.70	15.91	15.87	15.91	15.94	16.28	16.19	16.12	16.29
lagged_4_gold_close	22.60	22.76	22.88	22.92	23.02	23.10	23.42	23.20	23.18	23.40
lagged_4_usd_euro	1.06	1.06	1.06	1.05	1.07	1.06	1.07	1.07	1.06	1.07
lagged_4_usd_yuan	6.92	6.94	6.92	6.94	6.89	6.90	6.85	6.84	6.87	6.85

Table A: The initial 10 days of the dataset, post feature engineering. (Continue)

Date	1/12/2017	1/13/2017	1/17/2017	1/18/2017	1/19/2017	1/23/2017	1/24/2017	1/25/2017	1/26/2017	1/27/2017
lagged_4_oil_value	53.98	51.95	50.82	52.19	53.01	52.36	52.45	51.12	51.39	52.77
lagged_4_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_4_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_5_sil_close	15.76	15.64	15.70	15.91	15.87	15.91	15.94	16.28	16.19	16.12
lagged_5_gold_close	22.76	22.60	22.76	22.88	22.92	23.02	23.10	23.42	23.20	23.18
lagged_5_usd_euro	1.06	1.06	1.06	1.06	1.05	1.07	1.06	1.07	1.07	1.06
lagged_5_usd_yuan	6.89	6.92	6.94	6.92	6.94	6.89	6.90	6.85	6.84	6.87
lagged_5_oil_value	53.77	53.98	51.95	50.82	52.19	53.01	52.36	52.45	51.12	51.39
lagged_5_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_5_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_6_sil_close	15.58	15.76	15.64	15.70	15.91	15.87	15.91	15.94	16.28	16.19
lagged_6_gold_close	22.42	22.76	22.60	22.76	22.88	22.92	23.02	23.10	23.42	23.20
lagged_6_usd_euro	1.05	1.06	1.06	1.06	1.06	1.05	1.07	1.06	1.07	1.07
lagged_6_usd_yuan	6.93	6.89	6.92	6.94	6.92	6.94	6.89	6.90	6.85	6.84
lagged_6_oil_value	53.26	53.77	53.98	51.95	50.82	52.19	53.01	52.36	52.45	51.12
lagged_6_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_6_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13
lagged_7_sil_close	15.44	15.58	15.76	15.64	15.70	15.91	15.87	15.91	15.94	16.28
lagged_7_gold_close	22.32	22.42	22.76	22.60	22.76	22.88	22.92	23.02	23.10	23.42
lagged_7_usd_euro	1.04	1.05	1.06	1.06	1.06	1.06	1.05	1.07	1.06	1.07
lagged_7_usd_yuan	6.96	6.93	6.89	6.92	6.94	6.92	6.94	6.89	6.90	6.85
lagged_7_oil_value	52.36	53.26	53.77	53.98	51.95	50.82	52.19	53.01	52.36	52.45
lagged_7_interest_rate	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
lagged_7_inflation_rate	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.13

AUTHOR BIOGRAPHY

Name	Piyanuch Viwatborvornwong
Date of Birth	4 December 1997
Address	Sanseab, Minburi Bangkok 10510
Education	(2020) Bachelor's degree of Economics Thammasat University
Work Experience	(2021 – Present) Plan and Policy Analyst National Housing Authority