Analytical techniques in ML based Non-Ferrous Metal Price Forecasting

A Project Report

submitted by

KR HARIHARAN

in partial fulfilment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2023

THESIS CERTIFICATE

This is to certify that the thesis titled **Analytical techniques in ML based Non-Ferrous Metal Price Forecasting**, submitted by **KR Hariharan**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. C Rajendran
Research Guide
Professor
Dept. of Management Studies
IIT-Madras, 600 036

Prof. N. S. Narayanaswamy
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 12th May 2023

ACKNOWLEDGEMENTS

I would like to thank Prof.C Rajendran for his guidance, enthusiasm, and suggestions throughout the project.

I would also like to thank Prof.Narayanaswamy for his support and encouragement, and for providing me with my first research opportunity.

I would like to thank my mother for her constant mental support throughout my college life, and for being a role model displaying immense tenacity in her career.

I would also like to thank my father, for sparking my passion for Computer Science, and for discussing my project with me, which led to my first performance improvement idea.

Finally, I would like to thank my friends in IITM for assisting me in times of need, and making my time during this project and throughout college more light-hearted.

ABSTRACT

The focus of this study was to come up with methods of using Operations Research techniques to compete or complement ML techniques in time series forecasting, with a particular focus on non ferrous metal price forecasting.

Existing copper price forecasting techniques were studied, a model based on LSTM was selected, and attempts were made at incorporating data transformation techniques popular in Operations Research into this model. We show two methods, deseasonalization of the data and first order differencing, which were able to significantly improve on the predictive ability of the existing model.

TABLE OF CONTENTS

ABBREVIATIONS

ML	Machine Learning
OR	Operations Research
VMD	Variable Mode Decomposition
IMF	Intrinsic Mode Function
ARIMA	Auto Regressive Integrative Moving Average
SARIMA	Seasonal Auto Regressive Integrative Moving Average
LSTM	Long Short-Term Memory Networks
RNN	Recurrent Neural Network
BPTT	Back Propagation Through Time
ELM	Extreme Learning machine
CNN	Convolutional Neural Networks
ANN	Artificial Neural Networks

0.1 Algorithm Notations

$f(t)$	historical price data
$y(t)$	forecasted price data
$f_d(t)$	historical price data - detrended component
$f_s(t), y_s(t)$	historical and extrapolated price data - seasonal component
$f_t(t)$, $y_t(t)$	historical and forecasted price data - trend component
$f_e(t)$, $y_e(t)$	historical and forecasted price data - residual component
$g'(t)$	$g(t) - g(t-1)$, or forecast of the same, for some $g(t)$
$g_i(t)$, $g_{a,i}(t)$	i^{th} component of $g(t)$ and $g_a(t)$ after VMD, or forecast of the same

CHAPTER 1

INTRODUCTION

1.1 Motivation

Time series analysis involves studying historical data to identify trends and patterns over time, and building models to forecast future values of the time series.

With the improvement in computing power and increased research into ML in recent years, most recent attempts at time series forecasting have made use of ML Models. However, there already exists a wide range of research into time series analysis in the field of Operations Research, as time series analysis is crucial to deconstructing, understanding, and predicting many business performance indicators, such as demand, sales, and revenue.

Thus, in this project, we look to build models that combine existing OR techniques of time series data handling, with ML models, so as to improve their predictive performance. For this purpose, we delve into research and models built for a particular use case: non ferrous metal price forecasting.

Prices of non-ferrous metals are subject to a wide range of factors, including global supply and demand, geopolitical events, and economic conditions. The price of these metals is of great importance to a number of stakeholders across industries.

The forecasting of non-ferrous metal prices is crucial for companies that rely on these metals as inputs for their products. For example, the construction industry uses copper, aluminum, and zinc in plumbing, electrical wiring, roofing, and insulation. Similarly, the energy industry relies on non-ferrous metals for the production of solar panels, wind turbines, and batteries. Any fluctuations in the prices of these metals can have a significant impact on profitability of these industries. Accurate price forecasting is critical for these industries to manage their supply chain risks and plan their production schedules. Thus, predicting fluctuations in advance can be critical to a company or industry's health.

The value of companies/projects may be influenced by the prices of non ferrous metals, either directly (in the case of mining projects, for example), or indirectly as in the previously mentioned industries. As a result, these price forecasts are of importance to investors dealing with such companies/projects.

Similarly, traders involved in the commodity markets rely on these price forecasts to make buying and selling decisions.

Economists also use the prices of non-ferrous metals to analyze the health of the global economy. The prices of these metals can signal shifts in global demand and supply, which can have implications for inflation, trade, and other economic indicators. For example, an increase in the prices can be a sign of economic growth.

As a result of the importance of prices of non ferrous metals, there has been a range of research done in this area, with a particular focus on copper, providing us with a number of OR and ML models to analyze and work with. The importance of this area, and the depth of existing research, are the reasons why we have chosen to focus on this area.

1.2 PREVIOUS WORK

1.2.1 OR Approaches

Because of its importance in the industry, a number of approaches for time series analysis and forecasting have been developed in Operations Research, as laid out by Makridakis *et al.* (1998).

Methods to break down data into a cyclic seasonal component, smoothened trend component, and residual component have been devised and utilised to better analyse and forecast the data based on the different factors affecting the data.

In its simplest form, time series forecasts are attempted using mean, where the next value is expected to be the average of all the previous values, or moving averages, where the next value is expected to be the average of the last n values, for some n. To weigh more recent values more highly, and slowly phase out the influence of older values, exponential smoothening methods are used.

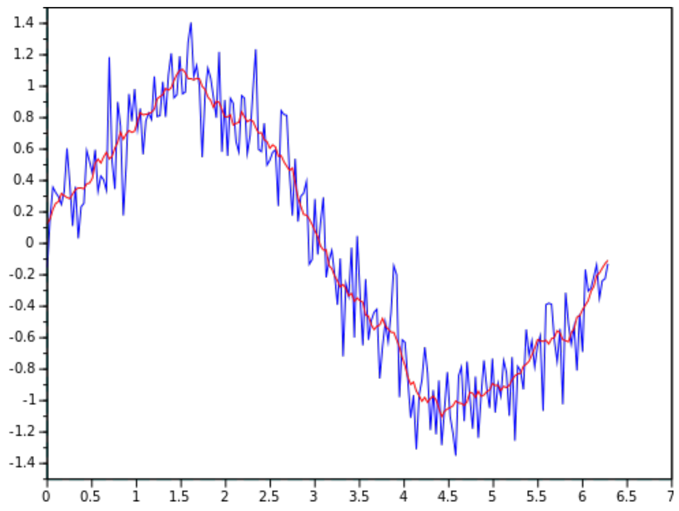


Figure 1.1: Moving Average : Wikimedia

In order to capture the effects of trends, the general rise or fall in values over time, and seasonality, the periodic variation in values, single exponential smoothening is extended to Holt's linear method (forecasting of data with trends) and Holt-Winter's method (forecasting of data with trends and seasonality).

$$\begin{aligned}
\text{Level:} \quad & L_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}) \\
\text{Trend:} \quad & b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \\
\text{Seasonal:} \quad & S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-s} \\
\text{Forecast:} \quad & F_{t+m} = L_t + b_t m + S_{t-s+m}.
\end{aligned}$$

Figure 1.2: Holt-Winter's Method : Makridakis *et al.* (1998)

Single and multiple regression is also widely used in Operations Research for time series forecasting. However, the autocorrelation (the degree of similarity between a given time series and a lagged version of itself over successive time intervals), trends, and seasonality of time series data is not handled well by regression techniques. For this purpose, ARIMA and SARIMA models have been developed.

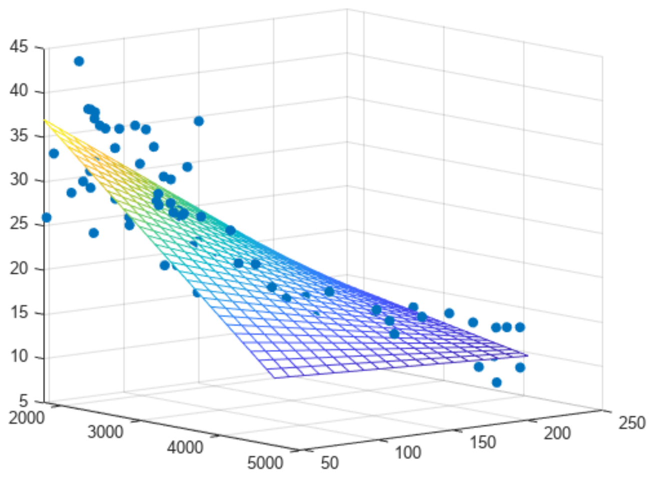


Figure 1.3: Multiple Regression : MATLAB

ARIMA models used autoregression, differencing, and moving averages based errors to make forecasts. SARIMA does the same, but also accounting separately for seasonality effects.

$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\begin{pmatrix}\text{Non-seasonal}\\ \text{part of the}\\ \text{model}\end{pmatrix}}\ \underbrace{(P,D,Q)_s}_{\begin{pmatrix}\text{Seasonal}\\ \text{part of}\\ \text{the model}\end{pmatrix}}$$

$$\text{ARIMA}(1,1,1)(1,1,1)_4$$

$$\underbrace{(1-\phi_1 B)}_{\binom{\text{Non-seasonal}}{\text{AR(1)}}}\underbrace{(1-\Phi_1 B^4)}_{\binom{\text{Seasonal}}{\text{AR(1)}}}\underbrace{(1-B)}_{\binom{\text{Non-seasonal}}{\text{difference}}}\underbrace{(1-B^4)}_{\binom{\text{Seasonal}}{\text{difference}}}Y_t = \underbrace{(1-\theta_1 B)}_{\binom{\text{Non-seasonal}}{\text{MA(1)}}}\underbrace{(1-\Theta_1 B^4)}_{\binom{\text{Seasonal}}{\text{MA(1)}}}e_t.$$

$$\begin{aligned} Y_t = {} & (1+\phi_1)Y_{t-1} - \phi_1 Y_{t-2} + (1+\Phi_1)Y_{t-4} \\ & - (1+\phi_1+\Phi_1+\phi_1\Phi_1)Y_{t-5} + (\phi_1+\phi_1\Phi_1)Y_{t-6} \\ & - \Phi_1 Y_{t-8} + (\Phi_1+\phi_1\Phi_1)Y_{t-9} - \phi_1\Phi_1 Y_{t-10} \\ & + e_t - \theta_1 e_{t-1} - \Theta_1 e_{t-4} + \theta_1\Theta_1 e_{t-5}. \end{aligned}$$

Figure 1.4: SARIMA with example : Makridakis *et al.* (1998)

1.2.2 ML Approaches

With the improvement in computing power and increased research into ML in recent years, most recent attempts at non ferrous metal price forecasting have made use of ML models.

Seguel *et al.* (2015) forecasted copper prices using a autoregressive-like function. The parameters of this function were learned using two meta-heuristic approaches: genetic algorithm and simulated annealing.

Liu *et al.* (2017) predicted copper prices using a decision tree learning model. Building on this, Díaz *et al.* (2020) obtained improved forecasting using random forests and gradient boosting regression trees.

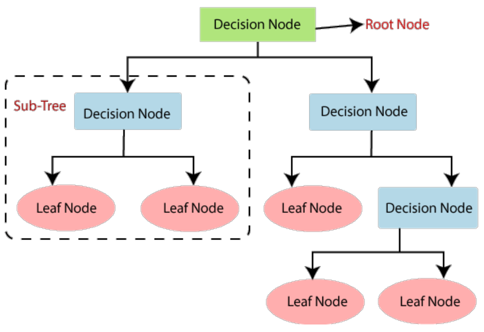


Figure 1.5: Decision Tree: Javatpoint

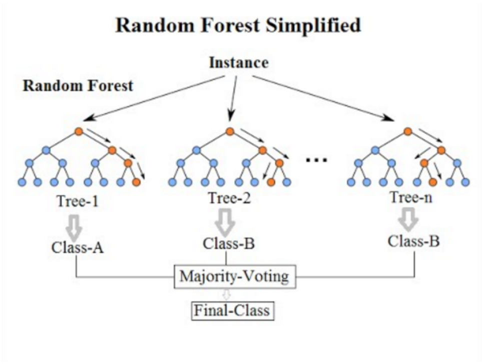


Figure 1.6: Random Forest: Wikimedia

Zhang *et al.* (2021) showed the effectiveness of ELM models for copper price forecasting. ELM models are neural networks with a single hidden layer; the input layer weights are randomly assigned, and the output layer weights are determined through a single backpropogation step, making this model very fast to train compared to ANN models. Similar to Seguel *et al.* (2015), Zhang *et al.* (2021) then proposed using two meta-heuristic approaches to determine the input layer weights: genetic algorithm and particle swarm optimization.

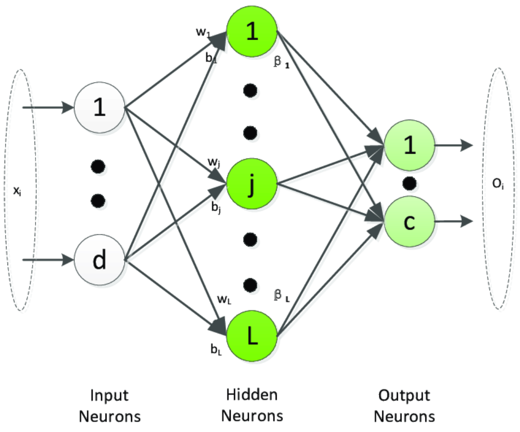


Figure 1.7: ELM : ResearchGate

For forecasting time series data, RNNs are typically more suited. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. Lasheras *et al.* (2015) proposed forecasting using an Elman RNN model, and compared its performance with and ARIMA model.

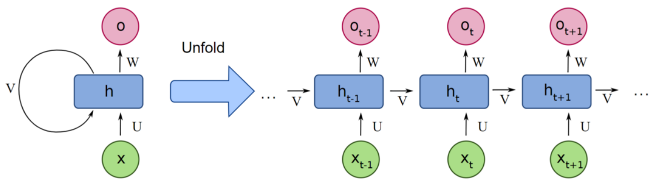


Figure 1.8: RNN: Wikimedia

Luo *et al.* (2022) proposed another such model, using LSTM models in 2 steps to predict copper prices. In the first step, copper prices are predicted using previous prices of copper, as well as associated non ferrous metals and currency fluctuations. In the second step, the errors of the previous first step predictions are used to predict the a correction term. The final prediction is a sum of these two.

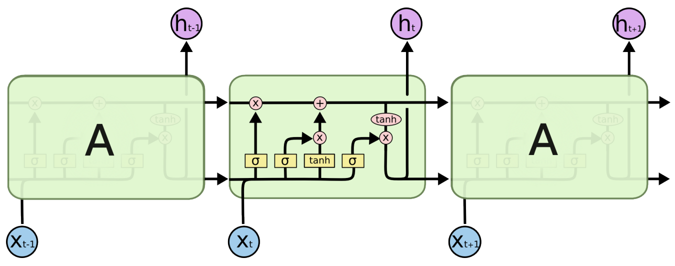


Figure 1.9: LSTM: Olah (2015)

In addition to application of ML techniques for forecasting, there has been work done on combining signal processing, OR, and ML techniques. For example, Selvam and Rajendran (2021) improved a model's forecasting performance by inputting parameters generated based on the Fourier series.

In another example of this, Liu *et al.* (2020) improved the forecasting performance of LSTM, by decomposing the copper price data using VMD and individually predicting each component. The sum of the prediction of each of these components is the final prediction. In this project, we are building on the model proposed by Liu *et al.* (2020)

CHAPTER 2

APPROACH AND ALGORITHM

2.1 Approach

2.1.1 LSTM

Traditional deep neural networks, such as convolutional neural networks, treat the different input-output pairs as independent of each other. However, when dealing with sequential or time series data, the different input-output pairs are not completely independent of each other. The upcoming values are likely to be influenced by the preceding values. Thus, if the model is able to store and account for these added contexts, it will be able to make better predictions.

In order to handle this, Recurrent Neural Networks were developed. RNNs differ from traditional neural networks, as the output is influenced not only by the input, but also a 'memory' that is stored in the RNN based on the previous input-output pairs.

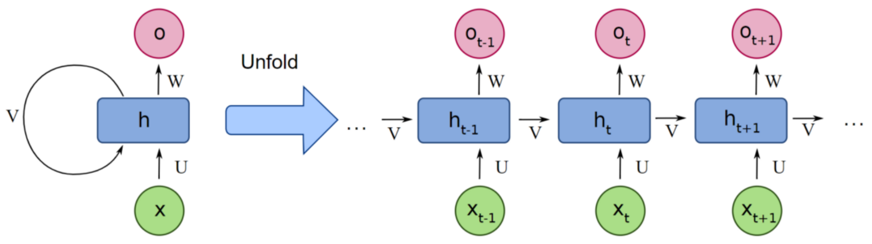


Figure 2.1: Wikimedia

Unlike CNNs, every layer of RNNs have the same parameters. Accordingly the learning method is slightly modified: where CNNs use Backpropogation for learning, RNNs use Backpropogation Through Time. In BPTT, the errors are backpropogated through the entire unfolded network, and then parameters (which are common to each layer) are modified.

While RNNs can handle time series dependencies, they suffer from the exploding/vanishing gradient problem during BPTT. If the gradients are small, then as the BPTT algorithm advances backwards, the gradients often get smaller and smaller and approach zero which eventually leaves the weights nearly unchanged. As a result, the gradient descent never converges to the optimum, leading to the vanishing gradients problem. Conversely, if the gradients are large, then as the BPTT algorithm advances backwards, the gradients often get larger and larger causing the weights to become NaN, known as the exploding gradients problem.

Also, while RNNs can handle short term dependencies well, they are not well suited to deal with long term dependencies, as the memory eventually diminishes. **To deal with both vanishing/exploding gradients problem and long term dependencies in the data, we use LSTMs.**

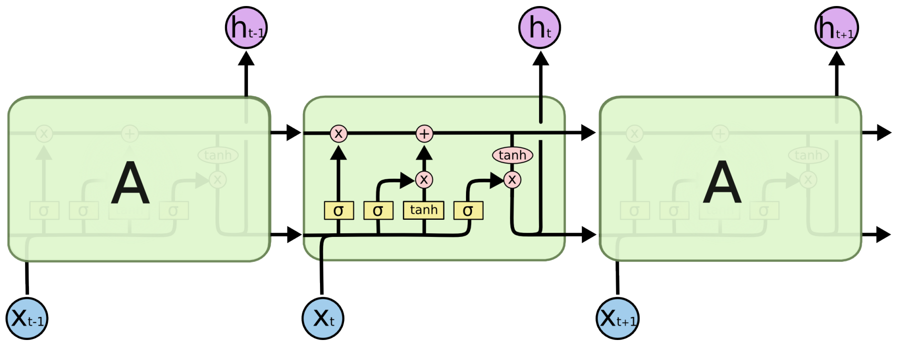
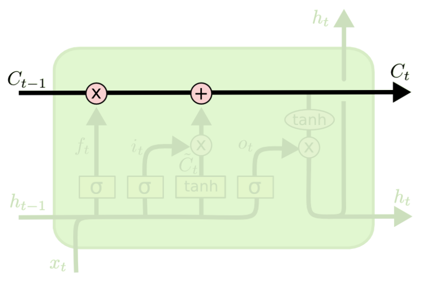
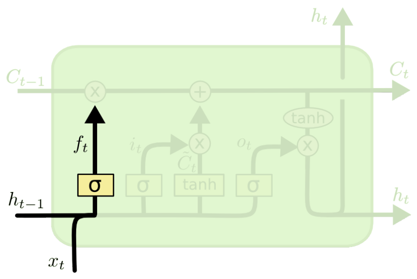


Figure 2.2: Olah (2015)

In LSTMs, memory about previous input-output pairs is stored in the form of the cell state. The cell state is modified through the forget gate and input gate.



Each LSTM cell has processes the input in 4 steps. The first step is the ’forget gate’. In this step, the portion of the cell state that is to be forgotten/remembered is determined, based on the input, via a sigmoid function.

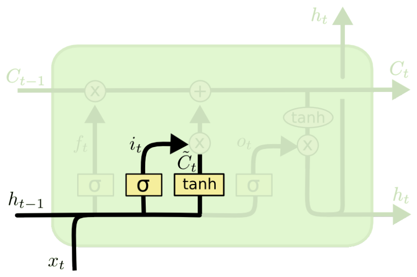


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 2.3:

W_f — forget gate weight
b_f — forget gate bias
h_{t-1} — previous output
x_t — input

The second step is the 'input gate'. In this step, the information to be added cell state based on the input is determined, via a sigmoid function and a hyperbolic tangent function.

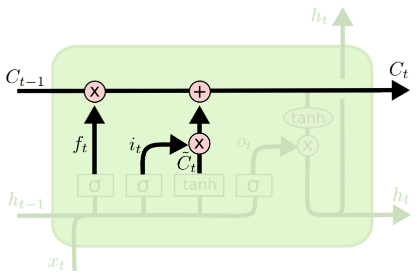


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 2.4:

W_i, W_c — input gate weights
b_i, b_c — input gate biases
h_{t-1} — previous output
x_t — input

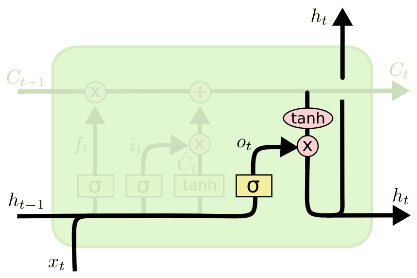
Based on the values obtained from the forget gate and the input gate, the cell state is updated to its new value. This value will be used to get the output, and will be passed on to the next cell iteration.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.5: C_t — cell state at t^{th} step

The final step is the 'output gate'. The input, modified by a sigmoid function, and the cell state, modified by a tanh function are multiplied to obtain the prediction.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.6:
W_o — output gate weight
b_o — output gate bias
h_{t-1} — previous output
x_t — input

In the models we build for this project, the time series data, or each of the components of the decomposed time series data, are individually forecasted using independently trained LSTM models.

The article by Brownlee (2016) was used as the base to write code for the LSTM model.

2.1.2 VMD

Variable Mode Decomposition is a signal processing technique used to decompose a signal into a set of intrinsic mode functions with different frequencies. **The VMD algorithm is designed to extract signals with a strong non-stationary behavior, such as the prices we are forecasting.** Such non-stationary signals cannot be effectively decomposed by other methods like traditional Fourier or wavelet transforms.

Another advantage of using VMD for signal decomposition, is that it is a data-driven technique that does not require any prior knowledge of the underlying signal properties. **Thus, VMD can be utilised with little prior understanding of the signal, making it somewhat domain-agnostic.** This aids in generalizing the prediction model across use cases.

The output of the VMD algorithm is a set of K IMFs that capture the non-stationary behavior of the signal, which can be used for further analysis. VMD has been successfully applied in various applications, such as biomedical signal processing as shown by Carvalho *et al.* (2020).

In the models we build for this project, we decompose the time series data using VMD, and each individual component is forecasted using an independently trained LSTM model. The package built by Carvalho *et al.* (2020) is used to perform VMD.

2.1.3 Trend and Seasonal Decomposing

Real world data, such as copper price data, is likely to not be stationary or random. This data is likely to follow a trend based on external factors. In the case of non ferrous metal pricing, the time series data is likely to follow an increasing trend, due to factors such as inflation, rising price levels, and increased demand with global growth.

Real world data is also likely to have a cyclic factor, based on economic, business, or seasonal cycles. The periodicity of this cyclic factor could be over multiple years, annual, bi-annual, quarterly, monthly, weekly, daily, or a combination of these, based on the factors affecting the data and the periodicity in which data is being recorded.

Thus, in addition to a data decomposition step independent of the signal properties, it can be beneficial to also have a decomposition step that is based on our prior knowledge of the data. Thus, we build a model where data is first split into a trend component, seasonal component assuming annual periodicity, and a residual component.

We can either perform additive decomposition, where the data is decomposed as,

$$Y_t = T_t + S_t + E_t$$

Or we can perform multiplicative decomposition, where the data is decomposed as,

$$Y_t = T_t * S_t * E_t$$

where,

$$Y_t - \text{original value}$$
$$T_t - \text{trend component}$$
$$S_t - \text{seasonal component}$$
$$E_t - \text{residual component}$$

For this project, as we are forecasting prices, we choose to focus on multiplicative seasonality. The reasoning being, when the price level increases/decreases due to factors such as inflation or supply and demand, the seasonal effects in price are also likely to proportionally increase/decrease in magnitude.

First, the trend of the series is determined using a moving average filter, and the data is split into the trend component and detrended data.

$$Y_t/T_t = S_t * E_t$$

The seasonal component is then determined, averaging the values in the detrended data across periods. The detrended data is then split into the seasonal component and residual component.

$$(Y_t/T_t)/S_t = E_t$$

The trend and residual component are decomposed using VMD, and predicted using LSTM models. The predictions are then combined with the seasonal component to obtain the total price prediction.

2.1.4 Differencing

As mentioned before, real world data, such as copper price data, is not likely to be stationary or random. This data is likely to follow an increasing trend. In order for a prediction model to account for this non-stationary behaviour, it must be able to capture long term trends.

CNNs and regression models fail to capture these trends, while RNNs and ARIMA models capture short term trends better than long terms. **LSTMs are better suited to capture long term trends, but can still perform better if the trends in data are reduced or eliminated through data preprocessing.** As mentioned by Selvam and Rajendran (2021), in ANNs, the bounded nature of the sigmoid or the hyperbolic tangent activation function makes it difficult to model time series with trends correctly.

One method of removing these trends, is through differencing. We define the differenced series as the change between each observation in the original series:

$$Y_t^{'} = Y_t - Y_{t-1}$$

Long run trends are removed from the time series through differencing. The differenced series resembles a random walk series:

$$Y_t^{'} = Y_t - Y_{t-1} = \epsilon_t$$

Random walks typically have long periods of apparent trends up or down which can suddenly change direction unpredictably.

2.2 Algorithms

2.2.1 Algorithm 1 (Benchmark)

The first algorithm is an implementation similar to the algorithm proposed by Liu *et al.* (2020). All the improvements made in this project are additions to this algorithm, and thus this algorithm will be used as a benchmark for the same.

In this algorithm, the time series data is decomposed into K components using VMD. Each component is individually forecasted using a separate LSTM model, and the forecasts are summed up to get the final price forecast. Each LSTM model takes l previous values of the time series component it is predicting as input, to predict the following value.

Algorithm 1 Benchmark: VMD + LSTM

```
  **Input:** The historical price data of non-ferrous metal $f(t)$,
  **Output:** The forecasted price of non-ferrous metal $y(t)$
  **Step 1: VMD**
  Perform VMD with mode number $K$ on $f(t)$ to get $f_i(t)$, $i = 1...K$
  **Step 2: LSTM**
  For each component $f_i(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y_i(t)$, $i = 1...K$
  **Step 3: Ensemble**
  Obtain difference forecast $y(t) = \sum_{i=1}^{K} y_i(t)$
```

2.2.2 Algorithm 2

The second algorithm is an attempt to improve on algorithm 1 by splitting the time series data into seasonal, trend and residual components. The idea is that the variations in the time series can be better characterized and explained when it is split based on the different causes: seasonal cyclicity, long term price level changes, and short term variations. This in turn will help in future prediction. Moreover, the seasonal component needn't be forecasted, only extrapolated. For the seasonal component, a period of 365 days was taken, effectively attempting to capture the annual cycles.

Once the copper price data is split into seasonal, trend and residual components, the trend and residual components are forecasted using algorithm 1. The forecasts are then combined with the extrapolated seasonal component through multiplication.

Algorithm 2 Seasonal and Trend Decomposition + VMD + LSTM

```
**Input:** The historical price data of non-ferrous metal $f(t)$
**Output:** The forecasted price of non-ferrous metal $y(t)$
**Step 1: Seasonal and Trend Decomposition**
**1a.** Decompose data into trend component $f_t(t)$ and detrended component $f_d(t)$
**1b.** Decompose detrended component $f_d(t)$ into seasonal component $f_s(t)$ with annual periodicity and residual component $f_e(t)$
**Step 2: VMD**
**2a.** Perform VMD with mode number $K$ on $f_t(t)$ to get $f_{t,i}(t)$, $i = 1...K$
**2b.** Perform VMD with mode number $K$ on $f_e(t)$ to get $f_{e,i}(t)$, $i = 1...K$
**Step 3: LSTM**
**3a.** For each component $f_{t,i}(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y_{t,i}(t)$, $i = 1...K$
**3b.** For each component $f_{e,i}(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y_{e,i}(t)$, $i = 1...K$
**Step 4: Ensemble**
**4a.** Obtain trend component forecast $y_t(t) = \sum_{i=1}^{K} y_{t,i}(t)$
**4b.** Obtain residual component forecast $y_e(t) = \sum_{i=1}^{K} y_{e,i}(t)$
**4c.** Extrapolate $f_s(t)$ to get $y_s(t)$ as $y_s(t) = f_s(mod(t, 365))$
**4d.** Obtain price forecast $y(t) = y_t(t) * y_s(t) * y_e(t)$
```

2.2.3 Algorithm 3

The third algorithm is another attempt on improving on algorithm 1, this time by differencing the data before implementing algorithm 1. Whereas the original data has non stationary trends, the differenced data more closely resembles a random walk, which should make it easier for the LSTM model to forecast.

In the case of price forecasting to make business decisions, it is often more critical to be able to accurately capture variations in the price, rather than the entire price itself. Also, on predicting the difference in data over time, we can use previous values to get a forecast of the full price, if required. Thus, even after performing this differencing, we are still capturing the crux of the data.

Algorithm 3 Differencing + VMD + LSTM

```
**Input:** The historical price data of non-ferrous metal $f(t)$
**Output:** The forecasted price difference of non-ferrous metal $y'(t)$
**Step 1: Difference**
Obtain the differenced time series data $f'(t) = f(t) - f(t-1)$
**Step 2: VMD**
Perform VMD with mode number $K$ on $f'(t)$ to get $f'_i(t)$, $i = 1...K$
**Step 3: LSTM**
For each component $f'_i(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y'_i(t)$, $i = 1...K$
**Step 4: Ensemble**
Obtain difference forecast $y'(t) = \sum_{i=1}^{K} y'_i(t)$
```

2.2.4 Algorithm 4

The fourth algorithm is an attempt to improve over the previous two algorithms, by combining the data handling techniques. The data is split into seasonal, trend, and residual components. The trend and residual components are forecasted using differencing and then using algorithm 1. The forecasts are then combined with the extrapolated seasonal component through multiplication.

Algorithm 4 Seasonal and Trend Decomposition + Differencing + VMD + LSTM

```
**Input:** The historical price data of non-ferrous metal $f(t)$
**Output:** The forecasted price of non-ferrous metal $y(t)$
**Step 1: Seasonal and Trend Decomposition**
**1a.** Decompose data into trend component $f_t(t)$ and detrended component $f_d(t)$
**1b.** Decompose detrended component $f_d(t)$ into seasonal component $f_s(t)$ with annual periodicity and residual component $f_e(t)$
**Step 2: Difference**
**2a.** Obtain the differenced trend component $f_t'(t) = f_t(t) - f_t(t-1)$
**2b.** Obtain the differenced residual component $f_e'(t) = f_e(t) - f_e(t-1)$
**Step 3: VMD**
**3a.** Perform VMD with mode number $K$ on $f_t'(t)$ to get $f_{t,i}'(t)$, $i = 1...K$
**3b.** Perform VMD with mode number $K$ on $f_e'(t)$ to get $f_{e,i}'(t)$, $i = 1...K$
**Step 4: LSTM**
**4a.** For each component $f_{t,i}'(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y_{t,i}'(t)$, $i = 1...K$
**4b.** For each component $f_{e,i}'(t)$, $i = 1...K$, input lagged train data to train individual forecasting models, and then input lagged test data to get individual forecasted result $y_{e,i}'(t)$, $i = 1...K$
**Step 5: Ensemble**
**5a.** Obtain differenced trend component forecast $y_t'(t) = \sum_{i=1}^{K} y_{t,i}'(t)$
**5b.** Obtain differenced residual component forecast $y_e'(t) = \sum_{i=1}^{K} y_{e,i}'(t)$
**5c.** Obtain trend component forecast $y_t(t) = y_t'(t) + f_t(t-1)$
**5d.** Obtain residual component forecast $y_e(t) = y_e'(t) + f_e(t-1)$
**5e.** Extrapolate $f_s(t)$ to get $y_s(t)$ as $y_s(t) = f_s(mod(t, 365))$
**5f.** Obtain price forecast $y(t) = y_t(t) * y_s(t) * y_e(t)$
```

CHAPTER 3

DATA, TESTING AND RESULTS

3.1 Data and Hyperparameters

The daily price data for all metals were sourced from investing.com. Copper and Zinc price data extends from 2010 to 2023, whereas Aluminium price data extends from 2014 to 2023.

In order to track the annual periodicity of data, values for missing days (weekends and holidays) were extrapolated as the average of the previously available and next available price.

The results shown below are based on modelling using 95% of the data as training data and 5% of the data as test data.

Liu *et al.* (2020) found mode number $K = 20$ to be most effective for effectively decomposing copper price data. Thus, we use the same in our implementation of VMD.

Each LSTM model has 4 units, takes the 10 previous values as inputs, and is trained for 10 epochs. These values were obtained by trial and error, accounting for forecasting accuracy as well as time taken.

3.2 Copper

Figure 3.1: Copper - LSTM

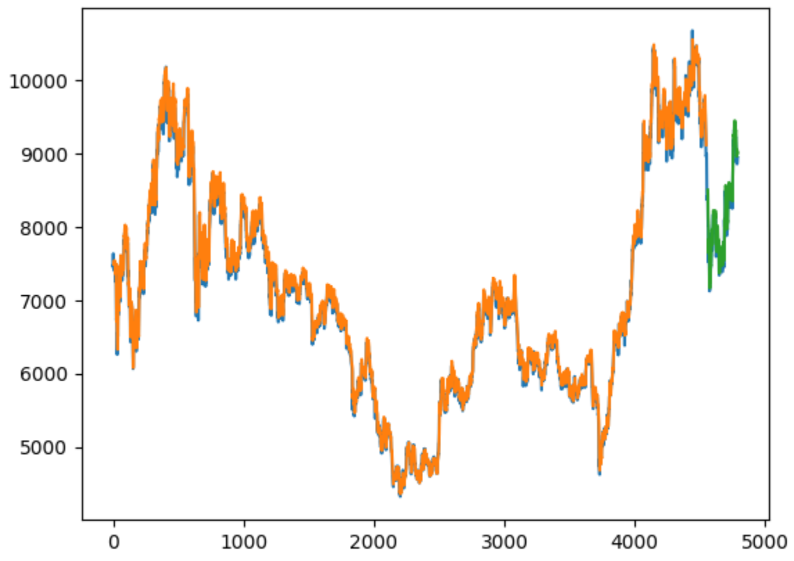


Figure 3.2: Copper - Algorithm 1 : VMD + LSTM

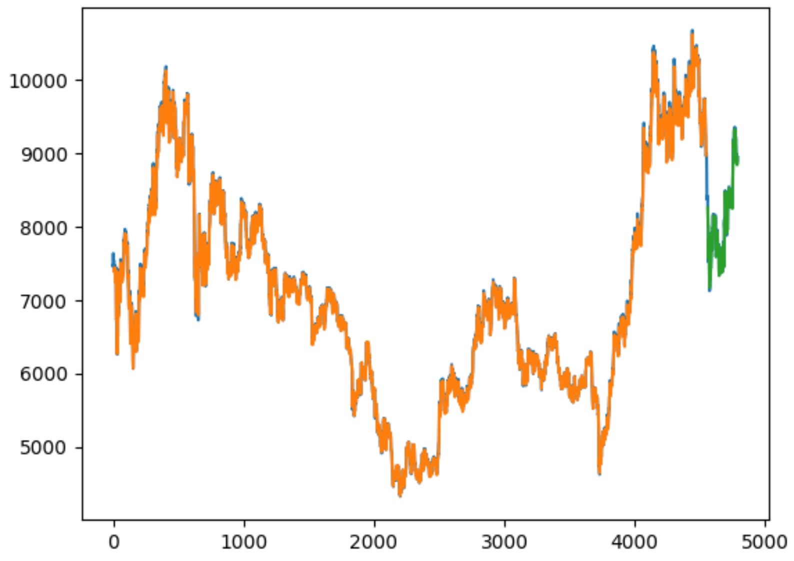


Figure 3.3: Copper - Algorithm 3 : Differencing + VMD + LSTM

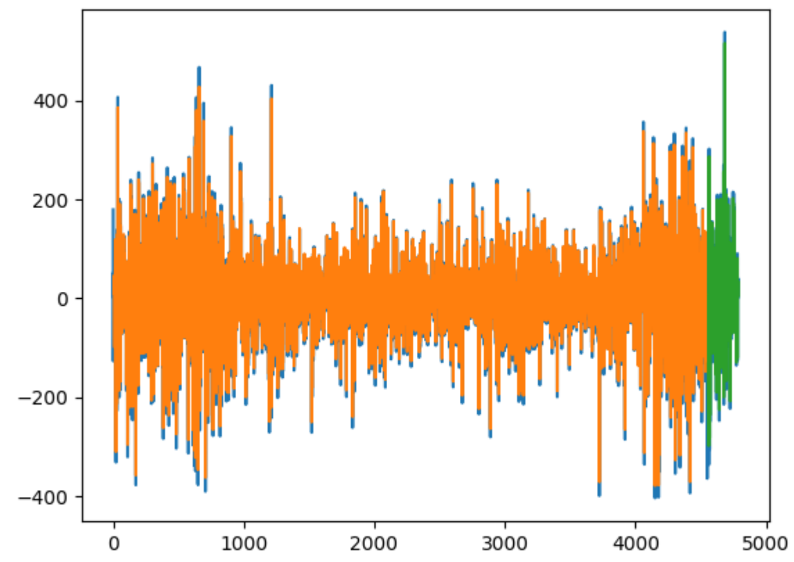


Figure 3.4: Copper - Algorithm 4 : Seasonal and Trend Decomposition + VMD + LSTM

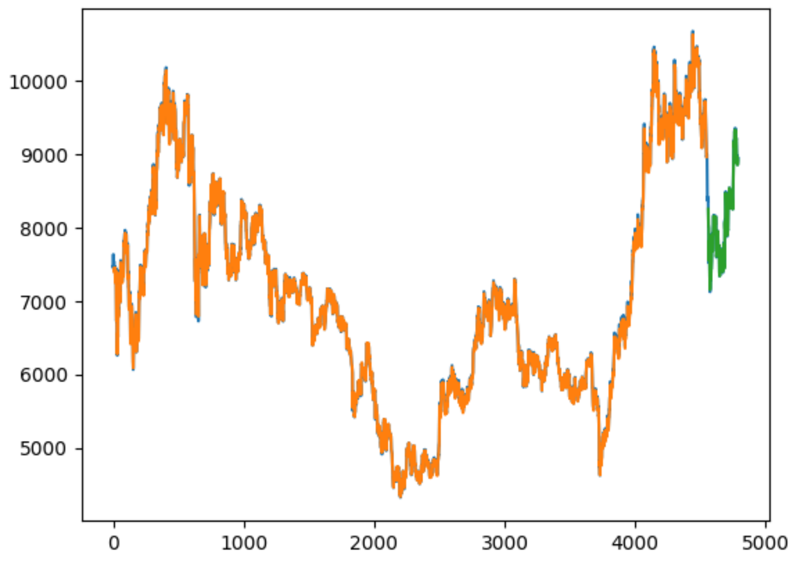
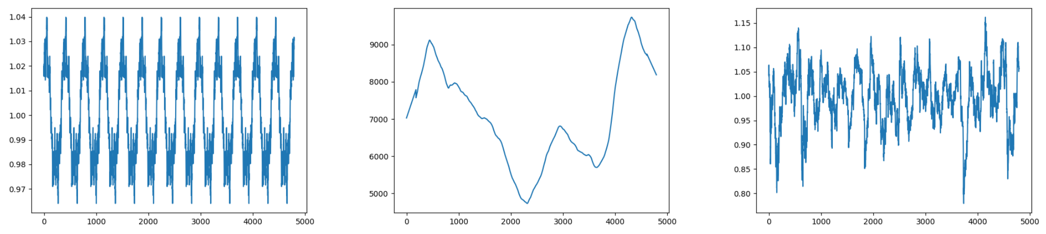


Figure 3.5: Copper - Seasonal, Trend, and Residual Components



	Training Error (RMSE)	**Test Error (RMSE)**
LSTM	97.508947	131.104977
Benchmark	25.009985	18.070448
Algorithm 2	18.915492	11.644881
Algorithm 3	7.228020	8.747397
Algorithm 4	6.641380	8.065471

The benchmark algorithm reduces test data error by more than 85% from a naive LSTM implementation. From here, algorithm 2, which splits the data in seasonal/trend/residual components further reduces the test error by almost 40%.

Algorithm 3, which differences the data, reduced the test error to less than half of that of the benchmark algorithm. Algorithm 4, which combines seasonal/trend/residual decomposition with differencing, produces a marginal forecasting improvement, reducing test error by around 8% from algorithm 3.

Figure 3.6: Components of Copper price data after VMD with $K = 20$

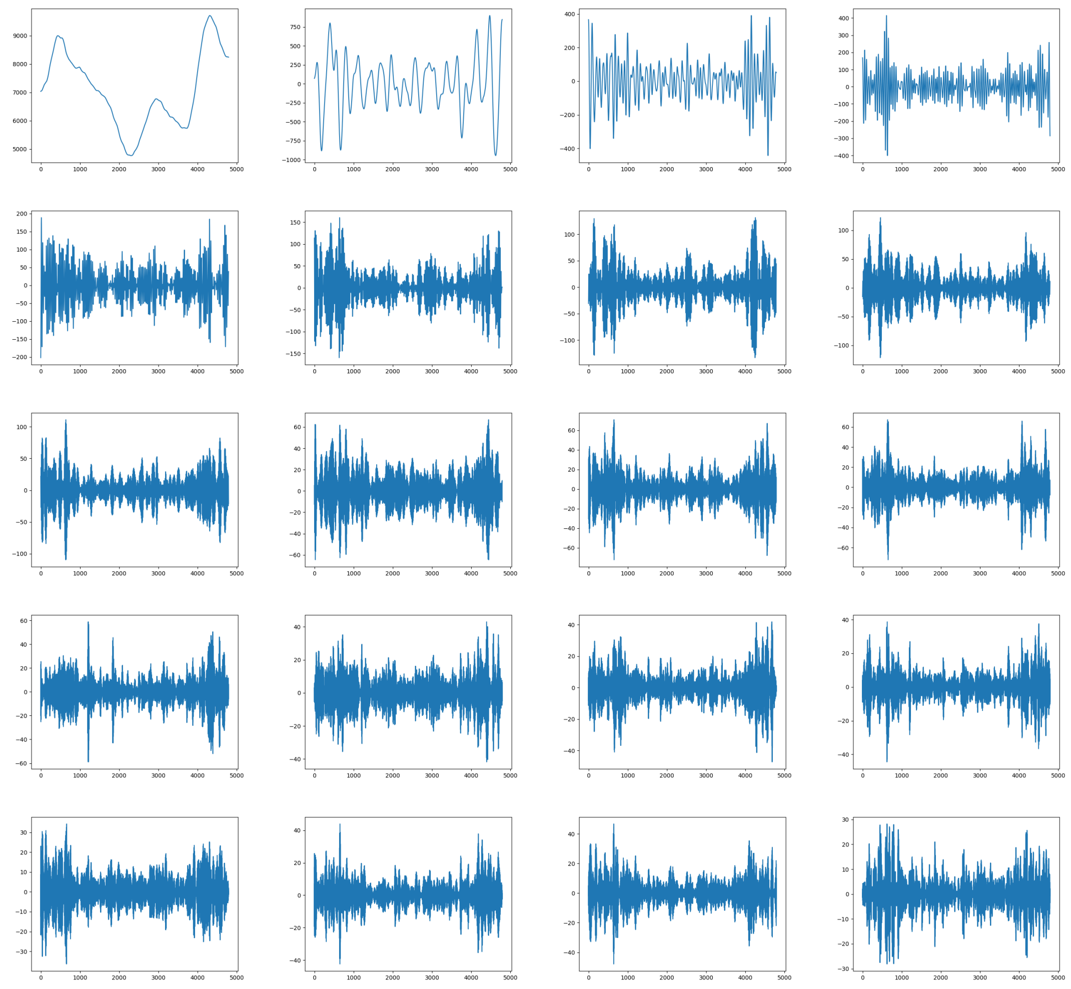
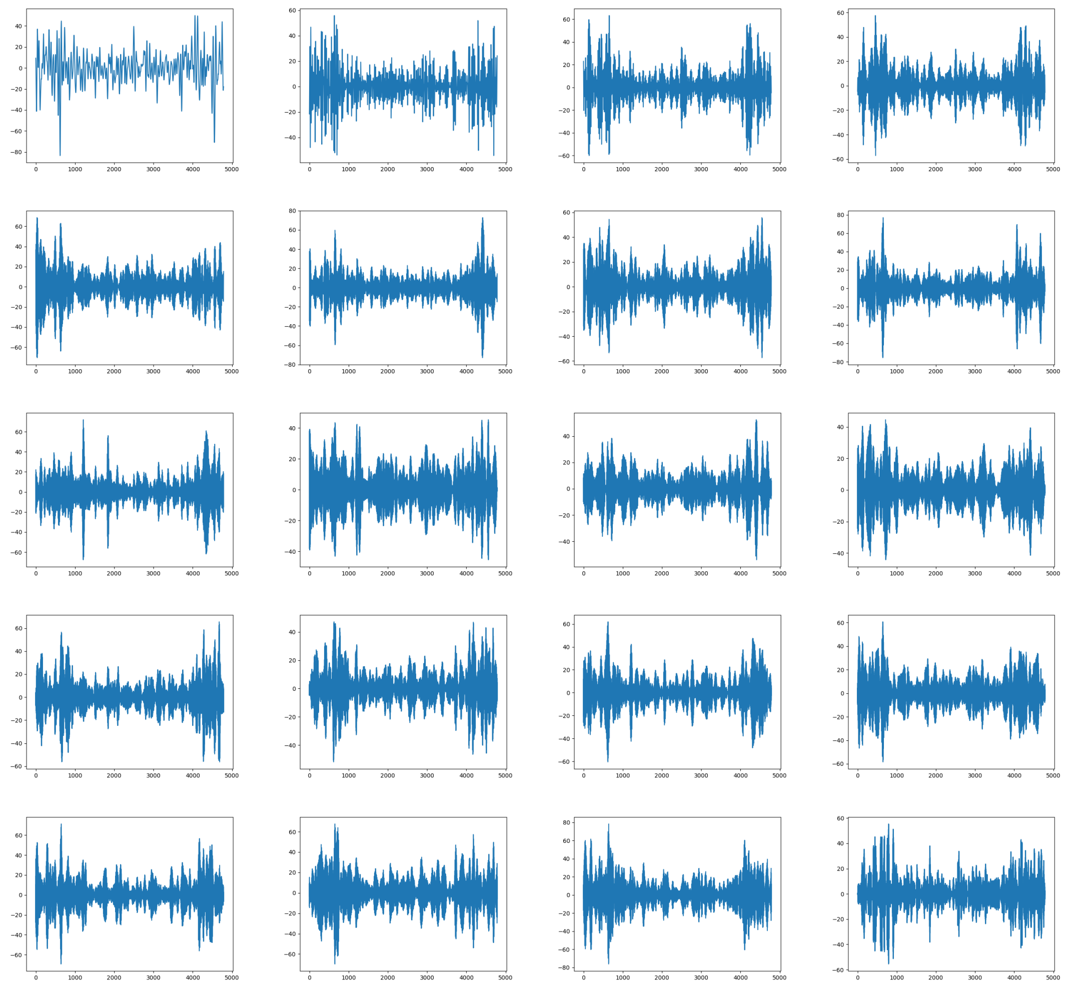


Figure 3.7: Components of Differenced Copper price data after VMD with $K = 20$



A significant portion of the forecasting error by the benchmark algorithm occurs when forecasting the first 2-3 components obtained from after Variable Mode Decomposition. Clearly, the first component follows a lifetime non stationary trend, whereas the second and third components follow some medium term trends.

On the other hand, after performing VMD on differenced data, the components have very little, if any, medium to long term trends. This data transformation helps in improving LSTM performance.

3.3 Aluminium

Figure 3.8: Seasonal, Trend, and Residual Components

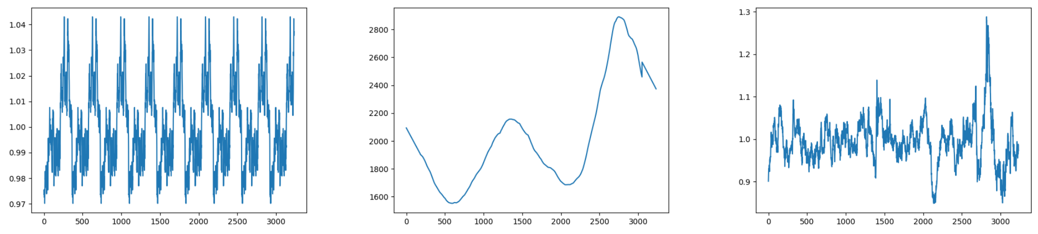
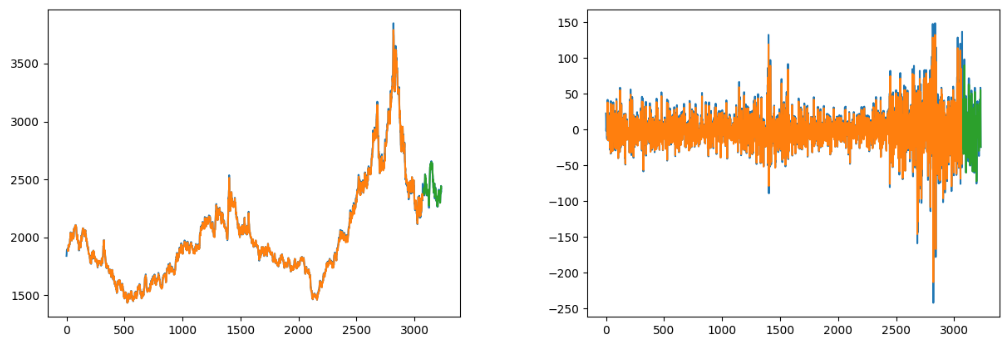


Figure 3.9: VMD + LSTM without and with differencing



	Training Error (RMSE)	**Test Error (RMSE)**
LSTM	28.841870	32.412891
Benchmark	6.639317	5.752907
Algorithm 2	5.319174	9.143038
Algorithm 3	2.835562	3.227119
Algorithm 4	1.975466	2.607139

3.4 Zinc

Figure 3.10: Seasonal, Trend, and Residual Components

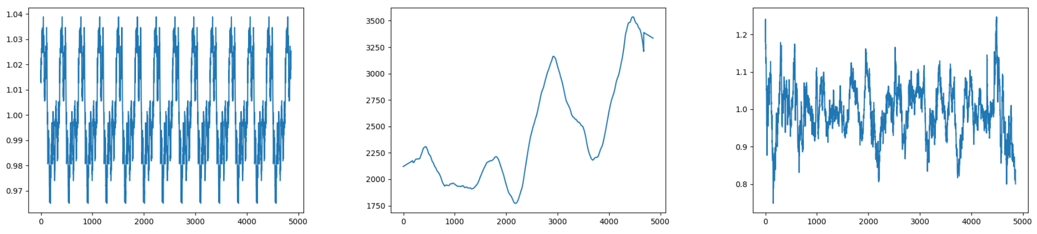
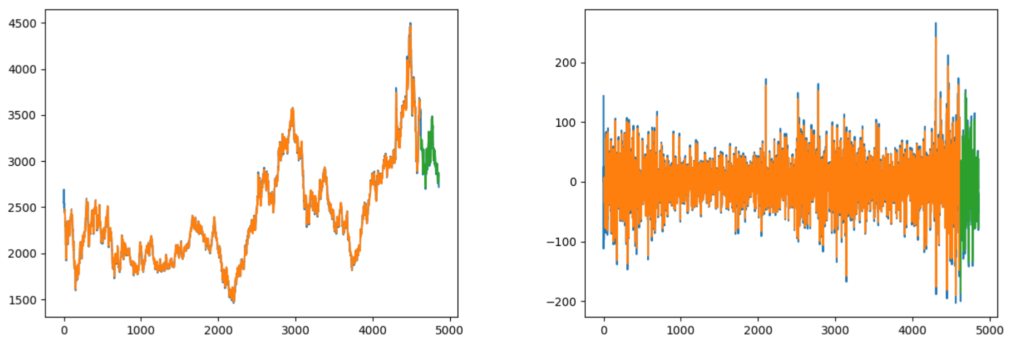


Figure 3.11: VMD + LSTM without and with differencing



	Training Error (RMSE)	Test Error (RMSE)
LSTM	37.952139	60.809865
Benchmark	7.105957	10.178958
Algorithm 2	5.317178	23.769801
Algorithm 3	2.731173	3.637032
Algorithm 4	3.044958	7.578704

While splitting the data into seasonal, trend and residual component improves explainability of data, the LSTM model will still have to forecast long term trends while forecasting the trend component. On the other hand, on differencing the data, the trends to be forecasted by the LSTM model are reduced in length and magnitude. Thus, algorithm 3 tends to perform better than algorithm 2. Upon differencing the trend data in algorithm 4, forecasting performance catches up, and slightly improves on algorithm (for aluminium and copper).

While algorithm 3, involving differencing, improves performance consistently, algorithm 2 shows a performance drop for aluminium, whereas algorithm 2 and 4 shows a performance drop for zinc.

On observing the trend and residual component plots for the 3 metals, we can see that near the end of the series, where the test data is, the residual component is not as random as expected, and the trend does not accurately reflect the data. The residual component of copper has an upward trend, of aluminium has a sharp spike, and of zinc has a downward trend. These changes should have rather been reflected in the trend data.

We use `statsmodels.tsa.seasonal.seasonal_decompose` for seasonal, trend, and residual component decomposition, which uses a moving average to determine trend. However, this is a naive method, which especially breaks down near the ends of the time series. The performance of algorithm 2 and 4 for these time series is likely to be improved if a different mathematical method is used for this decomposition.

CHAPTER 4

CONCLUSIONS

- Compared to a naive LSTM implementation, the VMD-LSTM model shows a significant improvement in prediction.
- Splitting the price data into seasonal, trend, and residual data prior to using the VMD-LSTM model further improves the prediction performance.
- As there is not as much of a trend in the differenced values, there is a further significant reduction in errors when predicting price fluctuation using the VMD-LSTM model, as opposed to predicting the entire price.
- Splitting the price data into seasonal, trend, and residual data, and then using the VMD-LSTM model to predict trend and residual fluctuation, as opposed to full trend and residual values, also marginally improves the prediction performance.
- VMD is a data independent decomposition method, whereas seasonal decomposition is a data dependent decomposition method. We can see that both these methods, as well as techniques such as differencing, can aid in improving the performance of forecasting models.

APPENDIX A

Parameter Generation

According to Domingos (2012) feature engineering is an important part of Machine Learning. Ideally, the use of additional explanatory variables will improve the predictive performance of a model, but in the absence of such explanatory variables, feature engineering becomes key.

Feature engineering is of 2 types:

- Endogenous features: These features are engineered through some transformation of the time series data itself.
- Exogenous features: These features are external explanatory variables that have a notable statistical and/or real world correlation with the time series data being forecasted.

Exogenous feature engineering is typically domain specific. Background knowledge regarding the time series data must be known to be able to choose the right explanatory data.

To overcome this domain specificity, Selvam and Rajendran (2021) came up with a set of time-related features, that are domain agnostic. These engineered features were shown to improve predictive performance even in robust forecasting models.

A similar attempt was made as a part of this project, to come up with mathematically generated domain agnostic features. The idea was to improve the predictive performance of the benchmark model discussed here, by assisting in modelling trends and seasonality.

A few of the generated parameters are shown here:

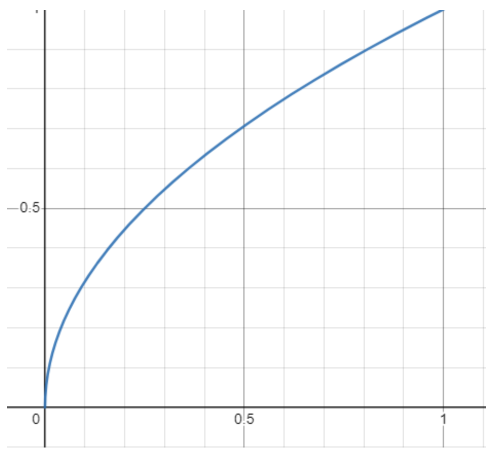


Figure A.1: t^p

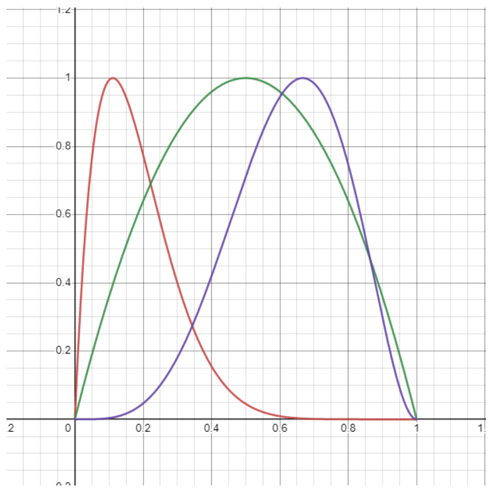


Figure A.2: $\frac{t^p(1-t)^q}{\left(\frac{p}{p+q}\right)^p\left(\frac{q}{p+q}\right)^q}$
(beta distribution like)

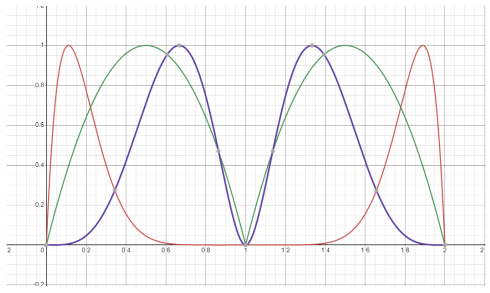


Figure A.3: $0 \leq t \leq 1 : \frac{t^p(1-t)^q}{\left(\frac{p}{p+q}\right)^p\left(\frac{q}{p+q}\right)^q}$
$1 < t \leq 2 : \frac{(2-t)^p(t-1)^q}{\left(\frac{p}{p+q}\right)^p\left(\frac{q}{p+q}\right)^q}$

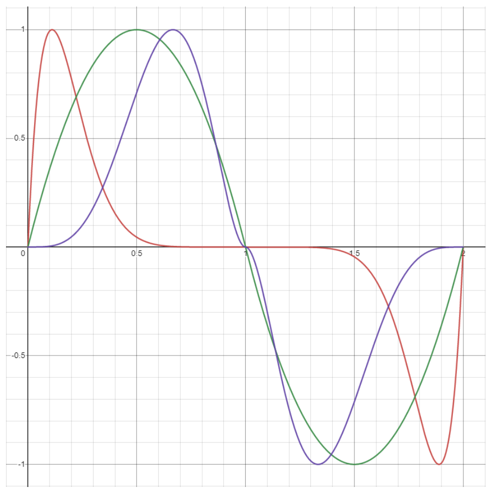


Figure A.4: $0 \leq t \leq 1 : \frac{t^p(1-t)^q}{\left(\frac{p}{p+q}\right)^p\left(\frac{q}{p+q}\right)^q}$
$1 < t \leq 2 : -\frac{(2-t)^p(t-1)^q}{\left(\frac{p}{p+q}\right)^p\left(\frac{q}{p+q}\right)^q}$

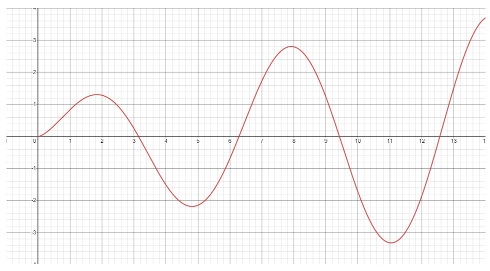


Figure A.5: $t^p sin(t)$

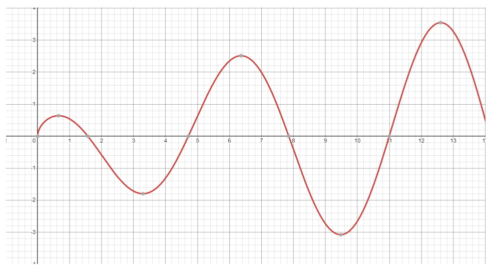


Figure A.6: $t^p cos(t)$

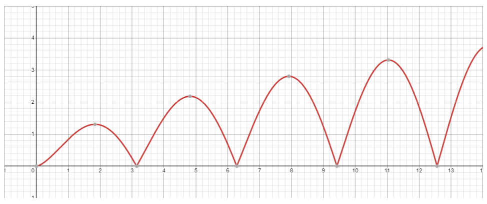


Figure A.7: $t^p sin(t \text{ mod } \pi)$

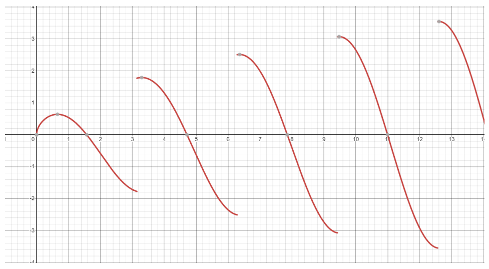


Figure A.8: $t^p cos(t \text{ mod } \pi)$

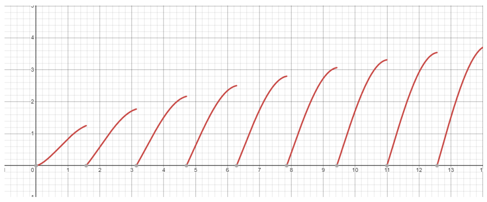


Figure A.9: $t^p sin(t \text{ mod } \pi/2)$

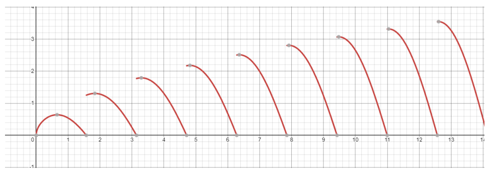


Figure A.10: $t^p cos(t \text{ mod } \pi/2)$

By varying parameters p and q in the range $[0, 0.5, 1, 2, 4, 8]$, and periodicity in the range [7 (weekly), 30 (monthly), 121 (quarterly), 365 (annual), 730 (2 years), 1825 (5 years), 3650 (10 years), full data set size (no periodicity)], about 800 parameters were generated to give as input to the forecasting models in addition to previous time series values.

The correlation between these generated parameters and time series data was calculated, and an arbitrary number of parameters with the strongest correlation were given as input to the forecasting models.

Some of these parameters have high correlation with the data. This is especially true for the parameters generated without much periodicity, so as to mimic the long term trend of the data.

Table A.1: Correlation of selected generated parameters with Copper price

	Correlation Coefficient
3650_bd_2_8	0.834775
3650_bd_1_4	0.793108
4789_sym_bd_2_8	0.790078
4789_sym_bd_1_4	0.778471
4789_sym_bd_0.5_4	0.773920

However, forecasting did not change much, and even worsened on incorporating these parameters. Thus, further work needs to be done with regards to parameter generation and selection, in order to effectively make use of feature engineering for non-ferrous metal price forecasting.

Table A.2: Copper Price forecasting - simple LSTM

	Training Error (RMSE)	**Test Error (RMSE)**
without generated parameters	97.508947	131.104977
with 5 generated parameters	125.815891	176.661053

The correlation significantly reduced upon performing VMD, and was virtually non existent for most components, making the generated parameters redundant.

REFERENCES

1. **Brownlee, J.** (2016). Time series prediction with lstm recurrent neural networks in python with keras. URL `https://shorturl.at/prvBC`.

2. **Carvalho, V. R.**, **M. F. Moraes**, **A. P. Braga**, and **E. M. Mendes** (2020). Evaluating five different adaptive decomposition methods for eeg signal seizure detection and classification. *Biomedical Signal Processing and Control.*

3. **Domingos, P.** (2012). A few useful things to know about machine learning. *Communications of the ACM*, **55**.

4. **Díaz, J. D.**, **E. Hansen**, and **G. Cabrera** (2020). A random walk through the trees: Forecasting copper prices using decision learning methods. *Resources Policy*, **69**.

5. **Lasheras, F. S.**, **F. J. de Cos Juez**, **A. S. Sánchez**, **A. Krzemień**, and **P. R. Fernández** (2015). Forecasting the comex copper spot price by means of neural networks and arima models. *Resources Policy*, **45**.

6. **Liu, C.**, **Z. Hu**, **Y. Li**, and **S. Liu** (2017). Forecasting copper prices by decision tree learning. *Resources Policy*, **52**.

7. **Liu, Y.**, **C. Yang**, **K. Huang**, and **W. Gui** (2020). Non-ferrous metals price forecasting based on variational mode decomposition and lstm network. *Knowledge-Based Systems*, **188**.

8. **Luo, H.**, **D. Wang**, **J. Cheng**, and **Q. Wu** (2022). Multi-step-ahead copper price forecasting using a two-phase architecture based on an improved lstm with novel input strategy and error correction. *Resources Policy*, **79**.

9. **Makridakis, S. G.**, **S. C. Wheelwright**, and **R. J. Hyndman**, *Forecasting: Methods and Applications*. Wiley, 1998.

10. **Olah, C.** (2015). Understanding lstm networks. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

11. **Seguel, F.**, **R. Carrasco**, **P. Adasme**, **M. Alfaro**, and **I. Soto**, A meta-heuristic approach for copper price forecasting. *In International Conference on Informatics and Semiotics in Organisations*. 2015.

12. **Selvam, S. K.** and **C. Rajendran** (2021). tofee-tree: automatic feature engineering framework for modeling trend-cycle in time series forecasting. *Neural Computing and Applications*.

13. **Zhang, H.**, **H. Nguyen**, **X.-N. Bui**, **B. Pradhan**, **N.-L. Mai**, and **D.-A. Vu** (2021). Proposing two novel hybrid intelligence models for forecasting copper price based on extreme learning machine and meta-heuristic algorithms. *Resources Policy*, **73**.