COMP390

2022/23

Deep Learning for Commodity Price Prediction

Student Name: Haoming Guo

Student ID: 201600384

Supervisor Name: Anh Nguyen

DEPARTMENT OF

COMPUTER SCIENCE

University of Liverpool

Liverpool L69 3BX

COMP390

2022/23

Deep Learning for Commodity Price Prediction

DEPARTMENT OF

COMPUTER SCIENCE

University of Liverpool

Liverpool L69 3BX

Abstract:

This project aims to represent the implementation and performance of several models about the prediction of commodity price. The machine learning models like Linear Regression and Support Vector Regression, and the deep learning models like Long Short-term Memory and Convolution Neuron Networks + LSTM combined models all will be trained and evaluated their performance. All the models predict the close price based on the previous days' prices data. More specifically, the Linear Regression make predictions based on the hidden linear relationship behind the input price data and output prediction price. Meanwhile, the Support Vector Machine predict the target close price using the non-linear relationship. Moreover, the deep learning model, LSTM, will be trained and evaluated twice to figure out the input data with more indicators (features) whether improve the performance. As the result, more indictors will not benefit to enhance the models. Additionally, compared the deep learning model with the machine traditional learning model like Linear Regression and Support Vectors Regression, the deep learning model LSTM, as a refined type of RNNs, have more advantages in handling the sequential input data, capturing the hidden pattern, and predicting the price. In the end, a CNNs + LSTM combined model is trained. The three CNNs can extract the features of three different period from the input data. And the extracted features are fed into the LSTM to predict. However, since the parameters of this combined model are simply inherited from the optimal model before without explore the parameter space, the performance of it is not as good as the conclusion from other paper. In conclusion, for predicting commodity price, several models are implemented and evaluate, which support that the LSTM have the relatively best performance among them.

Table of contents

I. Introduction

1. Background and Motivation

A commodity market refers to a physical or virtual platform that facilitates the exchange of raw materials or primary products through a process of buying, selling, and trading[1]. The commodity market is complex and dynamic ecosystem and hard to predict, since it is influenced by many factors, for example, the dynamic relationship between supply and demand, weather and environment situation, geopolitical developments and economic policies. The prediction is undoubtable complicated task, since the financial time series are noisy, nonstationary, and irregular [2]. Although lots of statistic, analytic and computing method can be used for prediction of these series (*Atsalakis* & *Valavanis*, 2009), the issue of financial variables, particularly the price index, remains a complex challenge that persists to date. However, the price prediction is still and always the key topic today which every market participant is eager to pursue and learn, such as producers, processors, brokers, and hedgers. Today, lots of price prediction learning focus on the stock price prediction instead of commodity price, but because commodity heavily influent the national economy of every country and indirectly change normal life for everyone, the commodity price fluctuations are getting more attention than before. Therefore, this project will aim to discuss and study about the factors and influence behind the prediction the price of some common commodity from market based on the different techniques having today.

Common econometric models for commodity price prediction include the autoregressive moving average(*ARMA*), vector autoregressive (*VAR*), and vector error correction models (*VECM*) [3]. Although they are widely used in commodity price forecasting, it is important to note that they are all based on statistical methods which rely on linear relationship. This approach can limit the accuracy and reliability of the

models, as they may not sufficiently capture the complex interactions and nonlinear relationships that exist within commodity price data. As a result, these limitations highlight the importance of considering Deep learning modeling techniques which with the features like high-dimension and non-linear allow more accurately to capture the underlying dynamics of commodity price data.

2. Objectives of the Project and Overview of Research Questions

This project will put the main attention on commodity price prediction by using deep learning techniques. And the main question is how to get a better even best performance deep learning model for commodity price prediction based on the deep learning technique now. This project is basically developed following these steps and contents: 1. Data collection and preprocess: introduce the data source, as well as some data processing methods, such as data cleaning, data visualization, etc. 2. Feature engineering and selection: compute and get it if any other helpful feature index needed and evaluate them. 3. Model training: introduce the establishment process of the model in detail, such as network structure design, parameter setting, etc. 4. Model prediction and evaluation: after training, evaluate the performance of its prediction by using what kinds of error or loss index. And directly represent the evaluation by using visualization technique.

Moreover, based on different model structure and different performance, the most important objective is to develop and implement a well-performed model and its structure. And it is also important to find the impact from different features and parameters. Since this project is continue being updated during a long time and there always have a new idea to update the implement. So, the main work of this project will be in form of flow like **figure[1]** shown below. The first model is Linear regression model, which use the close price of previous days (in a specific window) to predict the newest close price. It shown that the performance of prediction using the linear relationship between old days and the newest day. Then, a non-linear machine model, SVR (support vector regression) will be developed and implemented. The best parameter combination will be found by *GridsearchCV* and it will show the performance of using non-linear relationship between several features from previous days and the newest day' close price. Moreover, following the topic, a LSTM model is the next to be explore. Firstly, the best parameter combination of the model for the same features with SVR used will be found and evaluate its performance, to simply check and prove the idea from Selvin [4]"Deep neural networks can be considered as non-linear function approximators which are capable of mapping non-linear functions." Next is to find and train another LSTM in the same steps but for additional new features like moving average and RSI (Relative Strength Index), to check the more indicator whether will help the model performance. In the end a prediction model which combine the multi-channel CNN with LSTM will be implemented. The three CNN-sliding windows will extract the features for both long- and short-term input. These features are subsequently concatenated and feed into the LSTM to predict the price. As the conclusion from Selvin[4], the combination of CNN and LSTM models will allow for

efficient utilization of multiple time frame information, leading to improved accuracy in predicting stock prices.

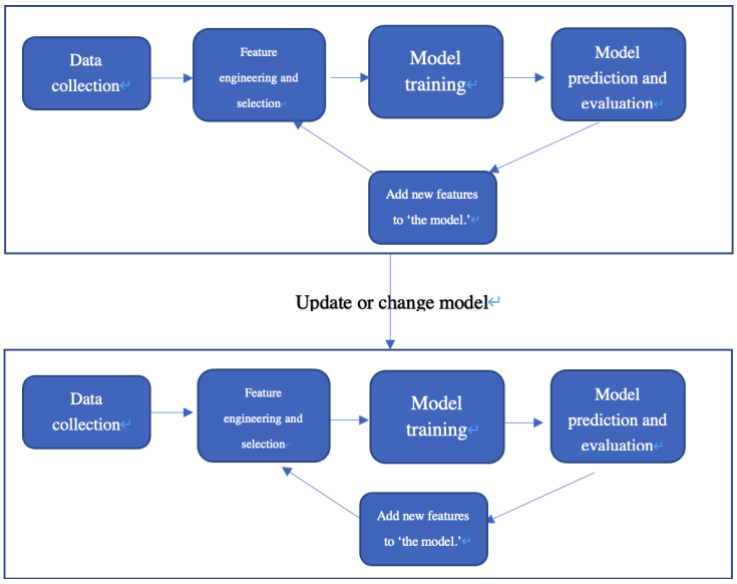

Figure 1: Workflow of this project

3. Scope of the Study

The scope of the objects of this project includes six different commodities from the Kaggle dataset[5], which are Gold, Palladium, Nickel, Brent Oil, Natural Gas, US Wheat respectively. The time span of historical price data on these commodities is from 2000 to 2022. As for the data processing, the normalization for this dataset will be implemented by the Min-Max scaler, which can enhance the accuracy and stability of training models. Since Min-Max scaler retain the relative size relationship of the original data by scaling data between 0 and 1, it makes easier and faster to train models and improve their accuracy. Moreover, the several deep learning models will be implemented and developed including like simple *RNN*(Recurrent Neural Network), *LSTM*(Long-Short-Term Memory), *CNN*(Convolutional Neural Network) and the hybrid model (combination of some model). This study will also aim to find what the difference performance between the pure individual model with the hybrid model, since lots of paper found the combination will reach a better performance than the individual models. For example, Kim[3] indicates that the hybrid model which combines the prediction from five different models (Autoregressive Integrated Moving Average (*ARIMA*), Vector Autoregression (*VAR*), Artificial Neural Network (*ANN*), Support Vector Regression (*SVR*) model, Random Forest (*RF*)) performs better than any single individual model. And also, from *Selvin*, *Vinayakumar* and *Gopalakrishnan*[4], the integration of CNN and LSTM models enables the authors to effectively leverage multiple time frame information and enhance the accuracy of their stock price predictions.

4. Changes of the Study

Compare with the detailed proposal and project video, this dissertion about the project

has changed a lot. The reason behind this situation is that many interesting and useful materials found push this project further and well-developed. In the detailed proposal, the plan of this project is influenced by *Forecasting directional movements of stock prices for intraday trading using LSTM and random forests*[6], which introduce a combination of *LSTM* and *RF* to predict the stock price movement. The task was defined as developing a python program to predict the 'Gold' and other commodities' price. However, using one model to predict all commodity price makes no sense since every different commodity has different price movement and the factors behind it. So the plan has been changed to develop lots of model, like firstly develop the linear regression, then implement other models like SVM (Support Vector Machine) or SVR (Support Vector Regression) and RF (Random Forests). However, this plan is a bit off the topic " Deep learning" since lots of the model from the plan is just machine learning model. As for the project video, it follows the structure to train a linear regression model for every commodity as the base line and then use it to compare with deep learning models, but the linear regression section occupied more section than it supposes to.

In the end, the project should be look like this: first a linear regression model is trained as the base line of linear relationship performance. Following is training a SVR model as the counterpart with linear regression to compare the different performance between non-linear and linear relationship models. Furthermore, deep learning models will be implemented and compare with some combination models' performance since lots of paper indicate that the combination will have a better performance.

II. Design and Implementation Chapters

1 Literature Review

1.1 Theoretical and Conceptual Framework

In this project, several machine learning models such as Linear Regression, Support Vector Regression and deep learning models like Long Short-Term Memory and Convolution Neuron Network will be implemented. More specifically, the two machine learning models represent the different prediction based on linear and non-linear respectively. And the *LSTM* as one of the deep learning models, outperforms the traditional models like Autoregressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ETS) in terms of forecasting accuracy, particularly for longer forecast horizons[7]. The below content will provide an overview with some visual representation of various models included in this project.

First technique used to predict the commodity price is Linear Regression analysis, which uses a mathematical model that describes the statistical relationship between one or more independent variables and a dependent variable. Based on the number of independent variables, it can be divided into simple linear regression and multiple linear

regression[8]. In this task, it will predict close price of the newest day based on a slide lookback window of close price of previous days, which is multiple linear regression. The independent variables may have data from 10 to 100 previous days. Then the model will try to find a best-fitting hyperplane, which allow it has the ability to make a prediction taking this hyperplane as reference.

Linear Regression has several strengths[8], for example, its simplicity and computational efficiency allow it can be trained and make the prediction quickly, even handling the large dataset. However, its limitations are also can not avoided, for example, it implies assumptions below: 1. Linearity assumption, which not hold in many real situations in the world, especially in the real-world market. The second is normality of the errors. Linear Regression assumes that the errors are normally distributed, which may not hold in real market cases. 3. It is sensitive to the outliers, which allow small noise can heavily influence the model's predictions. To find the non-linear relationship may hide behind the price data, the Support Vector Regression is implemented to evaluate the performance as comparation.

To predict the commodity price based on a time-series dataset, Support Vector Regression is also a suitable method. SVR is also one of the regression techniques as same as the MLR (Multiple Linear regression) and it is good at capture the long-term trends in the data[9]. SVR has three key principles: kernel function, margin and regularization. Kernel function is the key of SVR to overcomes the non-linearity issue in Linear Regression, by using a kernel function to transform the input variables into a higher-dimensional feature space to model the relationship between input variables and output variable easier. The margin, also known as support vectors, describes the distance between the hyperplane and the closest data points, which is a key parameter of SVR since the goal of it is to find an optimal hyperplane (defined by a set of weights and a bias term) that maximizes the margin between the predicted and actual values. The formular figure 2 shown below allows SVR find such a hyperplane, while still allowing some error [10]. Moreover, the regularization parameter of SVR determines the complexity of the model and also helps prevent overfitting, which heavily influent its performance. The results of the experiments based on house price prediction show that Support Vector Regression outperforms the linear regression techniques in terms of forecasting accuracy, with the lowest mean squared error (MSE) and root mean squared error (RMSE) values and the highest coefficient of determination (R^2) value [11].

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned}$$

Figure 2: Formular for SVR

Compared to Linear Regression, SVR has these strengths: it can handle the non-linear relationship for data with high dimension. Therefore, the feature selection is an important step in improving the forecasting accuracy of the SVR model [12]. And it has the ability to handle the data with noise, which may make a huge impact in linear regression. However, Support Vector Regression is limited by its high sensitivity to its parameters. And it also not as fast and explainable as the linear regression.

As for the deep learning models, they have more advantages about approximating non-linear functions, as their deep and complex structure them allow effectively map complex input-output relationships [4]. The common deep learning models are multiple layer perceptron(MLP), convolutional neural networks (CNNs) or recurrent neural networks (RNNs). As for this project, since the task is using the time-series price data to predict price, the RNN is more suitable. Loops in RNN's architecture allow it to persist information across time steps like figure 3shown below [13], enabling the network to capture temporal dependencies in the input data. However, standard RNNs is limited and suffered by the "vanishing gradient" problem, which occurs when the gradients of the loss function become too small to be effective in updating the weights during backpropagation. This problem is caused by the fact: a situation about exponential decay of the gradients will show when they are propagated backwards through the network, after the gradients are multiplied together over time[14].

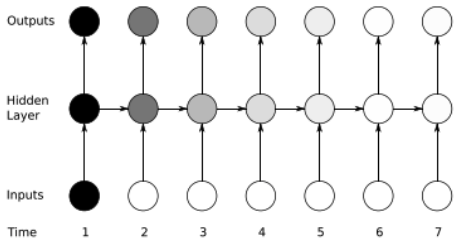

Figure 3: Structure of RNNs

Therefore, the Long Short-term memory as type of RNNs, is designed for handling the vanishing gradient problem in traditional RNNs models by adding memory cells and gates that enable the network to selectively learn or forget information as needed over long periods of time. Specifically, there are four key component types in *LSTM* structure: the input gate, the forget gate, the output gate, and the memory cell as figure 4 shown below[13]. During the forward pass, the input gate uses a sigmoid activation function to manage the flow of the new information. It takes the inputs and previous hidden state as input. And forget gate, using a sigmoid activation function, is similar with input gate, but to decide which parts of previous memory content (the current input and the previous hidden state) should be forgot. The output gate determines how much of the current cell state (the current input and the previous hidden state) should be used as output using a tanh activation function. Using a value between -1 and 1 from a hyperbolic tangent (tanh) function to represent the current state of cell, multiplies a value from 0 and 1, determined by another sigmoid function, to represent how much of

current state should be used to compute the output of the *LSTM* cell. Considering three gates existing, the *LSTM* model is optimized using a modified backpropagation algorithm, using gradients of the loss function to update the network parameters. In summary, *LSTM* is a nice choice to capture long-term dependencies in sequential data, especially for time-series price data of commodity.

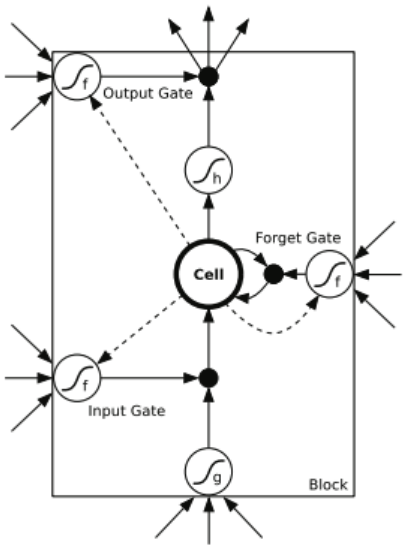

Figure 4: One *LSTM* block with one cell

In the end, as a recent development in deep learning for time series forecasting, the combination of CNNs and *LSTM* showing improved performance compared to standalone LSTM and other benchmark models about stock prices prediction[4]. The CNNs-LSTM model combines the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks by using the former to extract features from high-dimension input space, such as trends and patterns, and feed the features from CNNs to the latter LSTM to capture temporal dependencies from features. According to the evaluation experiment between CNNs-LSTM model with other traditional models like the ARIMA model and the random forest model, Cheng got the output which indicate that the hybrid model outperforms the traditional models in terms of forecasting accuracy, with an average improvement of around 20% [15].

1.2 Summary of Literature Review

In the previous chapter, the theoretical and conceptual framework of commodity price prediction using machine learning and deep learning models has been built for this project. Review literature on the models like Linear Regression, Support Vector Regression, Long Short-Term Memory, and Convolutional Neural Networks. Linear Regression is a simple and computationally efficient technique for predicting commodity prices. However, it has limitations, such as linearity assumption, normality of errors, and sensitivity to outliers. Support Vector Regression, on the other hand, is capable of handling non-linear relationships and noisy data, but is limited by its high

sensitivity to its parameters. Deep learning models, like RNNs and CNNs, are better at approximating non-linear functions due to their deep and complex structures, effectively mapping complex input-output relationships. Moreover, LSTM is a type of RNNs that is designed to handle the vanishing gradient problem by introducing memory cells and gates, making it a suitable choice for capturing long-term dependencies in sequential data, especially for time-series commodity price data. Lastly, recent developments have shown that the combination of CNNs and LSTMs can improve performance compared to individual LSTM and other benchmark models in stock price prediction, because the hybrid model combines the strengths from CNNs for feature extraction and LSTMs for capturing temporal dependencies.

2 Workflow Design and Implementation

2.1 Data Collection Design and Implementation

First of all, all the code of this project, including all the data preparation and details of every model will be implemented in the Google Colab, since Colab has the advantages such as easy to setup the environment and import the library, powerful hardware and synchronize between different devices. As for the data collection, this project develops different models based on the existing dataset 'Major commodity prices from 2000-2022' [5], which means that the collection step is simply finished by uploading the data in Colab and read into Python program. The existing dataset is from Kaggle, which is a popular online platform that hosts competitions about data science and provides a community for data scientists and machine learning enthusiasts to collaborate and share knowledge. Prasert Kanawattanachai creates this extensive collection of historical commodity price data spanning from 2000 to 2022, including daily price data for six commodities such as agricultural products: US Wheat, metals: Gold, Palladium and Nickel, and energy resources: Brent Oil and Natural Gas. More specificlly, the price data is stored as a spreadsheet in a CSV (Comma-Separated Values) file,which is a plain text format, meaning that it can be opened and edited easily. In a CSV file, each line represents a single row of data, and each value is separated by a comma within that row. The first row of the file typically is headers row for each column, while subsequent rows contain the actual data. In this dataset, the first row is six colume names of the table, which are Symbol, Date, Open, High, Low, Close, Volume. Moreover, the different commodities are distinguished by the 'Symbel' colume. The data collection step is not difficult to load into the Python program using Pandas. It is a popular open-source data manipulation library for Python, built based on the NumPy library and provides serveral powerful tools for working with structured data. The data will be store as dataframe after read by Pandas. Dataframe is two dimensional data structure like a table, but having many useful built-in functions. For example, using dataframe.info() can quickly get an overview of what just loaded, like figure 6 below.

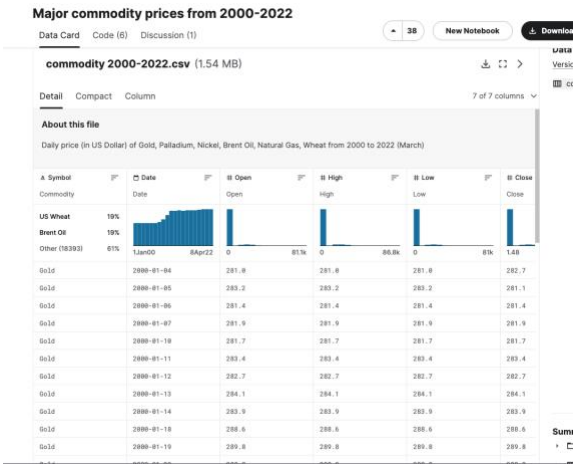

Figure 5: the overview of commodity price dataset.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5824 entries, 24111 to 29934
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Open    5824 non-null   float64
 1   High    5824 non-null   float64
 2   Low     5824 non-null   float64
 3   Close   5824 non-null   float64
 4   Volume  5824 non-null   int64  
dtypes: float64(4), int64(1)
memory usage: 273.0 KB
```

Figure 6: An example of dataframe.info()

Furthermore, after the first step: load the dataset, the next thing is dividing different commodity into different set, because everything further like analyzing and feature selection is only based on data from single commodity. All the model introduced before also will be implemented for six commodities respectively. Although there are so many methods can finish this divide job, the built-in function called groupby() is directly and simply used than others. As a well-defined functions from *Pandas* library, groupby() is powerful by to group rows of a Dataframe following some criteria, and also can apply a function to each group at same time. Based on current situation of the project, the original dataset will be grouped by the first column of the dataframe (*df.columns*[0]), which is Symbol column determine the different types of commodities.

2.2 Data Analysis Design and Implementation

After load the above dataset into the Python program, the following step is analyzing the data with no doubt. As the above figure 5, the author of this dataset has simply visualized the dataset by each column. However, considering the feature engineering and selection later, visualizing some key column will benefit for directly understand this dataset better. By using the function from *matplotlib.pyplot* library, it is easy to show the movement or distribution of data. On the one hand, the four columns about price are very similar every day, since they strongly have connection and will not have a huge gap or change between one day. On the other hand, the volume as one feature

may have relationship with price but will not be as strong as the relationship within other prices data, need to be considered as the key column which wait to visualize and analyze. Therefore, to reduce the unnecessary steps and keep the project in a clear code style, the close price, as both one of the four prices data and the predict target feature, will be visualized by the matplotlib with volume together. Here are six figures for visualization of every commodity respectively.

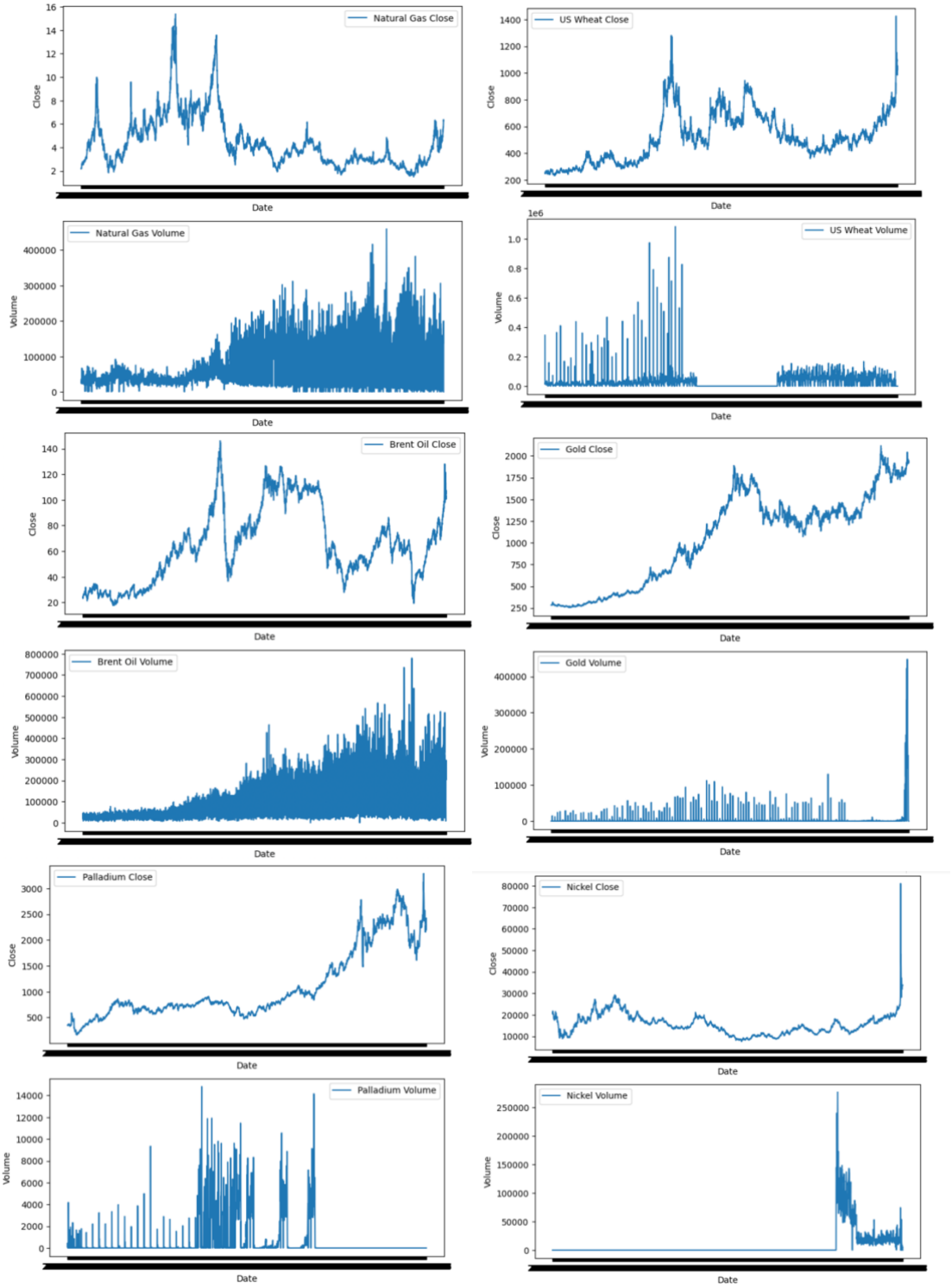

Figure 7-12: visualization of price and volume for every commodities.

Based on the figures above, the situation of the bonds between the price and the volume is shown clearly. Some of commodities may have the unknow connection between the price and volume, for example, the Nature Gas, Brent Oil. However, rest of commodities have the irregular distribution. Like figure 2, US Wheat, it have a high volume in the beginning and the end of the period, which expose that the market of the US Wheat is active and having trade frequently. However, in the middle period, volume of the US Wheat suddenly reduced to a very low level and keep this situation for a not short period. This situation means that the market of US Wheat is cool down and freeze during the period. But at same time, the price still have the movement as regular. Moreover, the situation about price and volume doesn't match also exist in other commodities. The volume of Gold meet a great peak in the end of period. And the Palladium and Nickel also have a low to zero level distribution. Combined the output of both price and volumn, those phenomenon are rare to happened in the real world market. At the same time, there have some comments from this dataset in kaggle pointed that the author had the problem in prevous version. More specificlly, other commodities have exact the same price data as the first commodity, Gold. Therefore, the Volume data is still possible get wrong since the auther is not the first and authoritative source of these data. Although it is no evidence show that the volume is wrong and just handle it as noise, to have a heathy train and performance of these models in project, do not use the volume as one of possible feature is not a bad choice. In this case, the data which will be split and generate features are only four different prices: 'Open','High','Low' and 'Close'.

To implement and handle the features more easily, this project will randomly pick one commodity to do the whole process first. As the result, the US Wheat data is picked. All the figure and result showing in next fewer chapter are all based on the four original price data of US Wheat. After finish all the work, from the data preparation to train and evaluate, from the Linear Regression to CNNs+LSTM, other commodities will also be processed in the same way (same pre-defined function and code).

2.3 Data split and Feature selection Design and Implementation

After confirmed the price columns of the dataframe will be used as potential feature for further engineering, the next step is other data preparation work like normalization, split, feature engineering and feature selection. First step is the data normalization. Normalization is the process which transform data to a common scale without change the differences in the range of values. Using data after Normalizing helps in improving the performance of machine learning algorithms. That is because it can reduce the effect of different scales and ranges of input features. As for this project, the prices are all in a large value. For example, price of gold is distributed from $200 in 2000 to around $2000 in 2022. Besides, the distribution of price does not follow any specific rule like Gaussian distribution or others. Therefore, it is suitable to apply Min-Max normalization for these data. More specifically, the Min-Max normalization do these

things to rescale the data: 1. Find the minimum value and the range of the distribution. 2. Every data need to subtract the minimum value of the feature. 3. divided by the range of the feature to finish the rescale steps. After knowing the basic rule behind the Min-Max scaler, , using the MinMaxScaler from sklearn.preprocessing library is the simplest way to implement this scaler. Create a scaler in range from 0 to 1. Then, use the scaler to take the value of data as input and output will be the rescaled data. The whole work of data normalization is finished until now. The next step is data split.

Data split usually represent a common technique used in machine learning to divide a dataset into two or more parts used for different period of a model, like training, validation, and testing steps. However, in this project, since several models will be implemented, they may have different requirement of data structures. So, it is necessary to arrange one more step in data split, which is using sliding windows to generate the sequential data. For Linear Regression and Support Vector Regression the original daily 'Close' data which each row represent one day's close price will be taken by using a lookback window as input and feed into the model. And for LSTM models, it also using a sliding window to get the sequential data but with all prices from previous days, since the complex deep learning structure all of them be able to handle the feature selection by themselves. More specifically, as one of the hyper-parameters, lookback window used to slide from beginning to end, to get the sequential daily price for training. For example, if the lookback window size is 60, it means the price of previous 60 days will be taken as input to predict the No.61 day's close price. Using the sequential price data from several days before to predict the final close price is more reasonable and more fit the real market situation because before having the close price of one day, it is not possible to confirm the High and Low price in advance. On the contrary, it is easy to get the previous price data in the real market. Therefore, Using the sliding lookback window to create the input dataset is more reasonable. Furthermore, it is slightly different for CNNs+LSTM hybrid model. Inspiring by Selvin[4], multi-channel CNNs is used to extract features. Here, three CNNs models are built for three different time frames and feed the combined the output into LSTM. Three CNNs for different model allow it can capture different period trend or pattern at the same for a single prediction. It will benefit the LSTM to finish the prediction task based on the new generated features. Based on the implement plan above, three channels CNNs will also require three input set together, and with one label at the same time. To feed data into the model correctly, a data split function different from these split function for LR and SVR and LSTM is implemented. It uses three stacked sliding window which the largest or longest one is outermost, and the smallest window is innermost. The outermost largest window can grantee that not only all three windows have enough data to generate the sequential data, but also all three sequential input set can have a same label, which is the close price from same day.

The pre-define function not only can generate sequential data as discussed above, but also can finish the traditional data split step. The input dataset will be divided into two parts, train and test dataset as the ratio 8:2. Too small test dataset may cause the

overfitting if the test set does not adequately represent the true distribution of the data. At the same time, too huge test dataset will make model does not have the enough data to be trained, and also cause the low accuracy since the model lack of generalization. Furthermore, each of the dataset both is divided into input dataset and the label set. All the work above can be finished using the function *train_test_split()* from the *sklearn.model_selection* library. It will return four datasets which are input train set, input test set, label train set and label test set respectively.

In conclusion, there will have three different methods to finish the work of data split, for Linear Regression model and Support Vector Regression model, for LSTM model and for CNNs+LSTM model respectively. Each method all has the function to generate the suitable sequential dataset using sliding window and split the data into train and test set following a predefined ratio as 8:2.

2.4 Model Structure and Parameter Design and Implementation

Before using the data from last step to train the model, one of the most important things is design the models and implement them. Although lots of predefined function from existing library can be use, it is still necessary to figure out what the problem the model will face and decide the possible structure from that. First, for the linear regression model, since it can only model for potential linear relationship, it seems like the base line of machine learning performance for this topic. Linear regression will predict the close price based on the close price from the previous days. For structure implementation of linear regression, it is relatively simplest among those models. It can be implemented by one row of code calling the *LinearRegression()* from library *sklearn.linear_model.* It will return a model ready to be trained. That is the model design and implementation of Linear regression.

As for Support Vector Regression model, it takes the same input as the Linear regression. However, SVR have more parameters than LR. First is the kernel function, which can map the input data to a higher-dimensional feature space where the optimal hyperplane can be found. There are several commonly used kernels: linear kernel, which assume there is a linear relationship between input features and output prediction. Polynomial kernel: maps input data to a higher dimensional space to find a linear hyperplane as model. Radial basis function(RBF): it can model non-linear relationship between input and output. Final one is called Sigmoid kernel, which also aim to find non-linear relationship, but used for the binary classification. In summary, RBF will be the kernel of the SVR used in this commodity price prediction project. Besides, there also have two parameters called Epsilon and Gamma which have the significant impact on the performance of the SVR model [17]. The epsilon parameter controls the width of the margin between the support vectors and the hyperplane, meanwhile the Gamma parameter define how much influence of each data point will deliver to model. To seek the relatively best parameter combination, two lists are built to store different values of these two parameters from an alternative range. These two parameter lists can expand

as the parameter space and ready to be trained in the next step to check the performance. In summary, the SVR has the parameters such as the kernel, the value of epsilon and gamma.

For design LSTM model, its main structure is clear and proved by many deep learning scientists. The simple layer combination is one LSTM layer with specific unit number, input size plus a dropout rate with a specific rate. The unit number is a hyperparameter that specifies the number of memory cells or hidden units in the layer. Adjust the value can allow people to determine the complexity and capacity of a LSTM layer. And dropout rate as a hyperparameter, it randomly sets some of the input units to be zero during each training iteration, to avoid overfitting and improve the generalization of the model. A LSTM model may have two or three such combined layers and a dense layer in the end. Dense layer in the last have the unit number equal 1, since it is used to produce the final one-dimension prediction result. It is fully connected with the last combined layer to map the output from high dimensional space to the desired lower dimensional space. However, the question is using how many this kind of layers combination with what the exact value for these parameters can build a LSTM model that outperform than other structures LSTM. Therefore, the parameters of LSTM need to be trained to adjust is the unit number of LSTM layer, dropout rate of dropout layer, the number of combined layers, and the input size which the first layer of LSTM need.

For the multi-channel CNNs + LSTM model, the structure and the parameter of LSTM is same as above while CNNs still need be analyzed. First thing can be confirmed is the number of CNNs channel is three, as the result of Selvin's paper[5]. These three CNNs extract the features from three different time frame, which can capture features from different length of periods. CNNs have the advantages in extracting the feature from the input space by applying the convolutions on the data. And CNNs have the ability to detect and extract the features from different levels of abstraction. For example, it can detect and get the feature from simple edges but also from complex patterns. Based on the understanding of CNNs, the parameters are structure relevant: layer number, kernel size, activation function, and specific unit number of layers. Generated features will be feed into the LSTM, then LSTM will use these features to capture the temporal dependencies hidden.

2.5 Training and Performance Evaluation Design and Implementation

After using the specific hyper-parameters to build the model with the responding structures, the next step is model training. It allows models have the ability to predict price not only based on the dataset but also the using pattern or rules it have leant during lots of times training. More specifically, an untrained model produces the output following the forward propagation mechanism. Since the initial weights or setting of the model can not support it to get the correct prediction as the label, an error like mean squared error will be computed to use in the backpropagation. Backpropagation is a

common algorithm used in neural networks to train it by propagating the error from the output layer back through the network to adjust the weights of the neurons in the hidden layers. The weights will be updated using the error gradient. And use the updated network to predict and compute the error again. The process above will be repeated for every training example in the dataset, until the updated weight allow model converges to a minimum error value. Model evaluation is a critical step to check the performance of model training. The trained model predicts on a new, never seen dataset, to check the performance in such aspects: 1. Generalization. The model having high level generalization can have an acceptable good performance when it meets new data. Evaluation the generalization can also check the model has not overfit the training dataset so that can generalize well on the new data. 2. Structure selection. evaluation can be used to compare the performance for the one model but using different structures to select the best one for the task. It is necessary because especially the deep learning model will have amounts of structures to choose during implementation. 3. Hyperparameter tuning. As the name, hyperparameter is the super key parameters can have influence on the model performance from the foundation. Finding a best solution from the parameter space is critical. For example, a higher learning rate will result in faster convergence but also have possibility to cause model to overshoot the situation with minimum error.

The training strategies for each model are slightly different. For training Linear Regression model, it takes the sequential input data produced by different sliding lookback windows of different features. Considering its simple structure, it needs people to select feature for it in advance, which deep learning model like CNNS can extract features from input by itself. To find the relatively best parameters of linear regression from the potential range, evaluate the performance of model trained with different feature is also important. A linear regression model will be trained for every lookback window size, also for every commodity since the different lookback window size influent the input features as well and different commodity has different price movement and different input pattern. To reduce the code redundancy, training several linear regression models for a range of lookback window sizes is implemented in only one function which allow user easily to call to finish the model training step. This function called *LinearRegression_lookback()* and takes a specific value of lookback, the scaled data from Min-Max scaler and a boolen value to instruct how to split dataset (shuffle or not) as parameters. In this function, it can call the data split function by taking the lookback and boolen value as input. Then using the dataset split function return, to train the model and make a prediction. After that, it can also compute the mean square error for prediction and label for further evaluation step. In the end, the function will return the mse(mean squared error), the input and output test set. In summary, this function integrates many functions by calling other functions such as data split, model training, model prediction and evaluation preparation. However, to finish the whole process with every potential lookback, this function needs to be called several times by using a for loop in the function called mseList. As its name, it calls the *LinearRegression_lookback()* for every lookback in range and store different mse from

them into a list for further evaluation. All the details above are shown in the below **Figure13**.

```
# use a lookback window including the previous close to predict the new close
# return the MSE,X_test,y_test,lr_model
def LinearRegression_lookback(lookback,scaled_data,random_Split):
    # Split test and train dataset
    X_train, X_test, y_train, y_test = data_split(lookback,scaled_data,random_Split)

    # build the linear regression model
    lr_model = LinearRegression()

    # train
    lr_model.fit(X_train, y_train)

    # predict
    y_pred = lr_model.predict(X_test)

    # compute the mse for evaluate
    mse = mean_squared_error(y_test, y_pred)

    return mse,X_test,y_test,lr_model

# store the mse in list for visualiazation
# return two mse list
def mseList(lookbacks,scaled_data,shuffle = True):
    mse_list = []
    for lookback in lookbacks:
        mse,_,_,_ = LinearRegression_lookback(lookback,scaled_data,shuffle)
        mse_list.append(mse)
    return mse_list
```

Figure 13: the detail of training and evaluation preparation.

Moreover, to evaluate the performance more directly, the mean square error is chosen as the indicator to represent how good the performance exactly is and show the change of it for different model setting in a figure. An evaluation figure example of linear regression will be shown below(**figure 14**). It shows that mean squared error for linear regression model for US Wheat with the input set generated by different lookback sliding window. However, only an exact error number can not tell people how good performance it has. Therefore, using the detail prediction result compared with its respective label can represent it more directly. Like the **figure 15** shown, the model is trained by the input with lookback window size equal 75 days, which is relatively best one selected based on the result from **figure 14**.

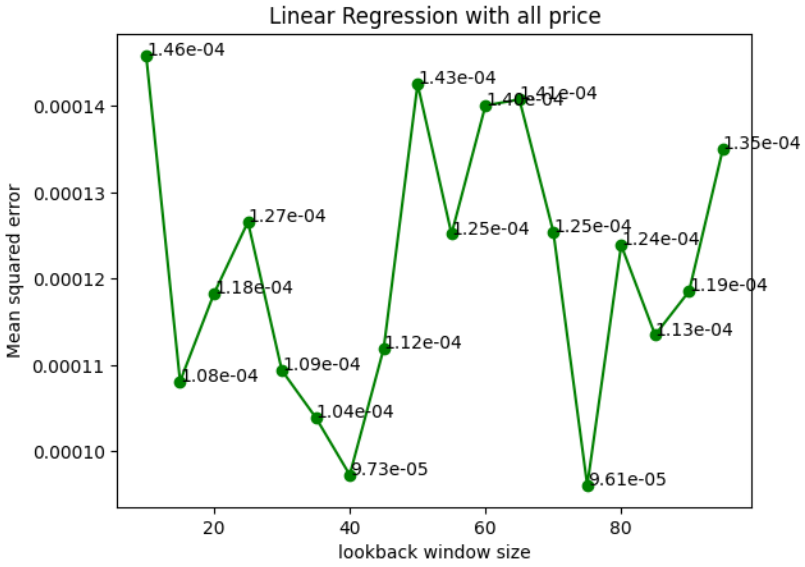

Figure 14: an evaluation based on the mean square error for linear regression model with different lookback size.

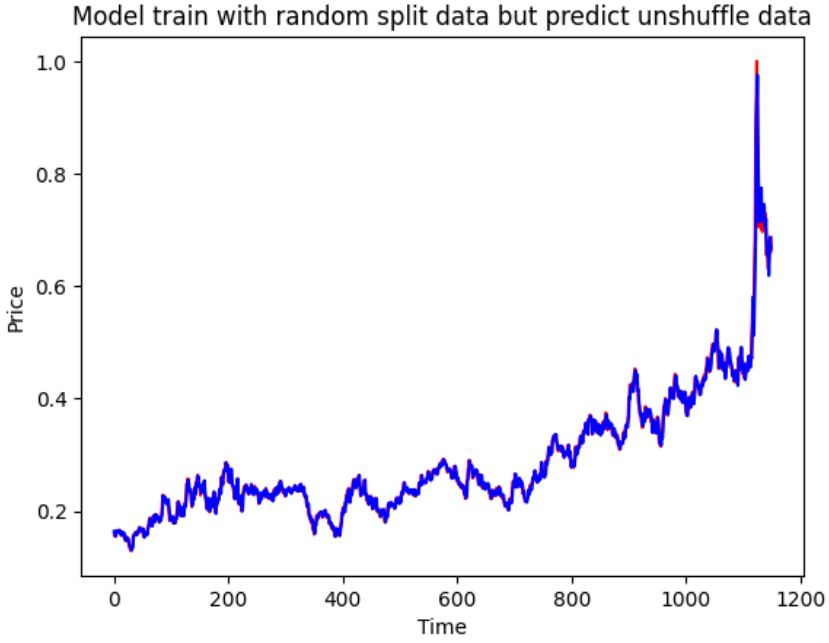

Figure 15: the specific performance of LR using the prediction result(blue line) and label(red line)

As for the Support vector regression model, first thing is defining the range of parameters: C and Gamma. Since the limitation of computing power does not allow setting a huge range for every parameter, the first four common value for each parameter will be trained and evaluated. For example, the potential search range of C is from 0.1, 1, 10 to 100. And range of Gamma is from 0.001, 0.01, 0.1 to 1. As for the lookback window size, the range of it will from 10 to 100 with step long is 5, which means there have 4*4*19 = 304 different potential parameter combination wait to be trained and evaluated. Therefore, a relatively easy solution for this situation is using *gridSearchCV* from the *sklearn.model_selection* library. The SVR model also can be

easily implemented by importing library *SVR* from the *sklearn.svm*. To train and evaluate SVR model for every parameter combination, the detail code design and implementation is shown below (Figure 16). The strategy to go through all the potential parameter combination is: 1. Using a for loop to go through all the lookback window size from beginning to end. 2. Implement the SVR model with kernel RBF. 3. Call the *GridSearchCV* to find the best combination from parameters C and Gamma range. However, the result returned by the *GridSearchCV* is not the best one. It is only the best one with its parameter under a certain lookback window. To find the global best parameters in the whole range, it needs to compare every best model from different lookback together. Therefore, the best model returned is kept being compared with other in the next step, still choose mean square error as the criteria of each model performance. The mean square error and the respective lookback are visualized in the figure 17, which allow people knows the model with lookback = 15, C = 100 and gamma = 0.1 has the relatively best performance in the range. Same as the Linear regression, another figure 18 directly shows the performance of best model by plotting prediction and the label movement. In conclusion, the SVR model has the best performance on predicting the price of US Wheat with the parameters: lookback = 15, C = 100 and gamma = 0.1.

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV

# define the hyperparameters for grid search
params = {
    # degree control low c-> wider margin
    'C': [0.1, 1, 10, 100],
    # controls the shape of the decision boundary bigger gamma -> more irregular
    'gamma': [0.001, 0.01, 0.1, 1]
}
# use the data-split function for linear regression.
mse_list = []
lookbacks = np.arange(10, 100, 5)
best_mse = float('inf')
for lookback in lookbacks:
    X_train, X_test, y_train, y_test = data_split(lookback,scaled_data)

    # perform grid search cross-validation
    svr = GridSearchCV(SVR(kernel='rbf'), param_grid=params, cv=5)
    svr.fit(X_train, y_train)

    # print the best hyperparameters and score
    print('The best paramters are {} with lookback = {}'.format(svr.best_params_,lookback))
    best_model = svr.best_estimator_
    y_pred = best_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mse_list.append(mse)

    if mse <= best_mse:
        final_model = best_model
        best_lookback = lookback
        best_mse = mse
```

Figure 16: implementation detail of SVR model.

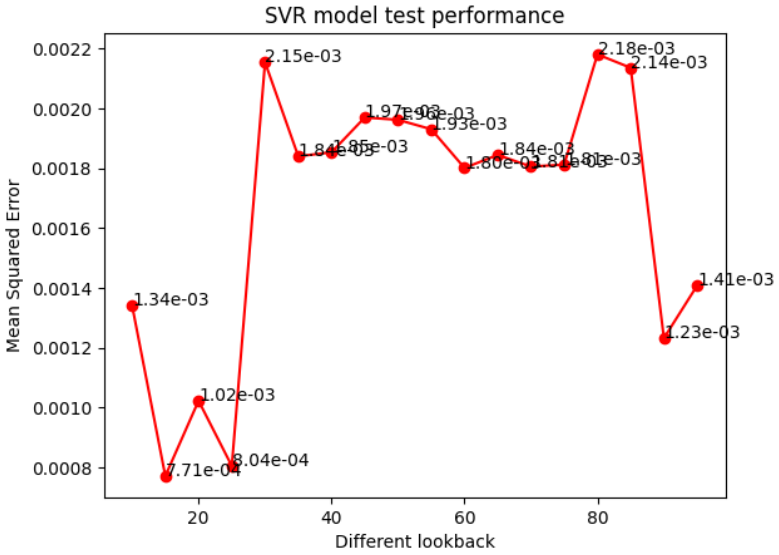

Figure 17: the mean squared error for local best SVR model under different lookback window sizes.

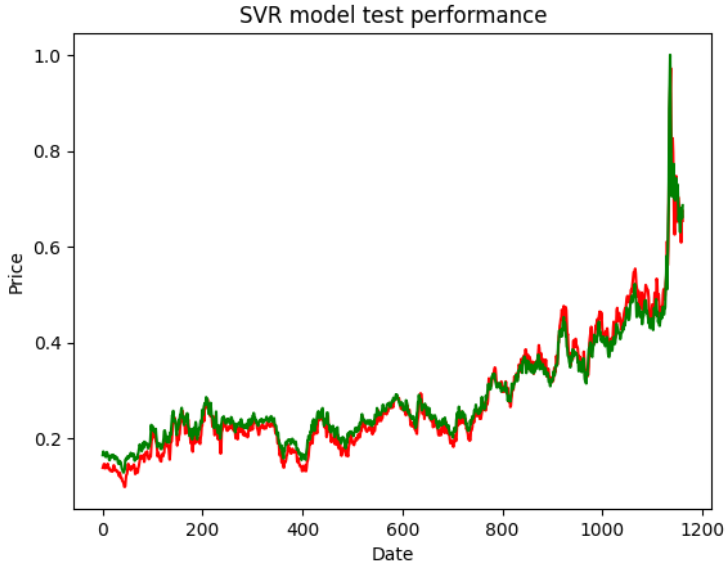

Figure 18: the performance of global SVR using the prediction and label.

To call and use easily to build, train and evaluate the LSTM with different parameters and different structure, there also design an integrated function for LSTM, like figure 19 shown below. First is a data split function, which is slightly different from the data split function for Linear Regression model. That is also the reason it is called data_split2. In detail, the data split for linear regression store only the close price from the previous days as input data when applying the sliding lookback window, while the data split function for LSTM takes all the data from the data scaled by Min-Max scaler as input. That is because the LSTM have more complex structure to handle the input features and capture the complex relationship or pattern behind. Excluding the generated different input, rest of the two data split functions are same. It will return four datasets

for input and output, train and test respectively. The other function in figure 19 is responsible for build a LSTM model with a specific structure. The *layer_num* instruct the function to build LSTM with a corresponding layer number by using a for loop and other condition control code. The first LSTM layer need confirm the *input_shape*, which is the shape of example from input training set. The last layer is a dense layer to strict the one output dimension. Moreover, the middle layer is LSTM with Dropout layer. The *unit_num* is the number of units in each layer LSTM. Considering the limitation of computing power of author, the unit number of each layer are same as default. This function is built for easy call when training and evaluation in the further step.

However, even under the help of these two predefine function, there still have a huge parameter space need to be explore, like figure 20. There are 3(*layer_num*) * 3(*unit_num*) * 3(*dropout_rate*) * 10(*lookbacks*) = 270 parameter combinations wait to be trained and evaluated. So like the SVR, use the imported *GridSearchCV* to explore. It will be called under the for loop to find the local best parameter combination for this lookback. furthermore, the mean squared error of each local best is compared to find the best parameter in this range, like *figure 21*. In conclusion, the LSTM model have the relatively best performance which the mean squared error is $7.59e^{-5}$, with the parameters is: '*dropOut_rate*': 0.1, '*input_shape*': (60, 5), '*layer_num*': 2, '*units_num*': 150} with lookback = 60.

Pre-define functions

```
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasRegressor

def data_split2(lookback,scaled_data,random_Split=True):
    # Initialize empty lists for X and y data
    x = []
    y = []
    for i in range(lookback,len(scaled_data)):
        x.append(scaled_data[i-lookback:i])
        y.append(scaled_data[i,3])
    # convert list to numpy array
    x,y = np.array(x),np.array(y)
    if random_Split:
        # Split and return
        return train_test_split(x, y, test_size=0.2, random_state=42)
    else:
        # Split and return
        return train_test_split(x, y, test_size=0.2, shuffle = False)


def LSTM_model(layer_num,units_num,dropOut_rate,input_shape):
    # define and build model as instruction
    model = Sequential()
    for i in range(layer_num):
        # first layer
        if i == 0:
            model.add(LSTM(units_num, input_shape=input_shape, return_sequences=True))
            model.add(Dropout(dropOut_rate))
        # last layer
        elif i == layer_num - 1:
            model.add(LSTM(units_num, return_sequences=False))
            model.add(Dropout(dropOut_rate))
        # middle layer
        else:
            model.add(LSTM(units_num, return_sequences=True))
            model.add(Dropout(dropOut_rate))
    model.add(Dense(units = 1))
    model.compile(optimizer = 'adam',loss = 'mean_squared_error')
    return model
```

Figure 19: the predefined function to help LSTM implementation and training.

```
# define the hyperparameter space
param_grid = {
    'layer_num':[2,3,4],
    'units_num':[50,100,150],
    'dropOut_rate':[0.1,0.2,0.5],
    'input_shape':[(X_train.shape[1], X_train.shape[2])]
}
```

Figure 20: The potential parameter range of LSTM

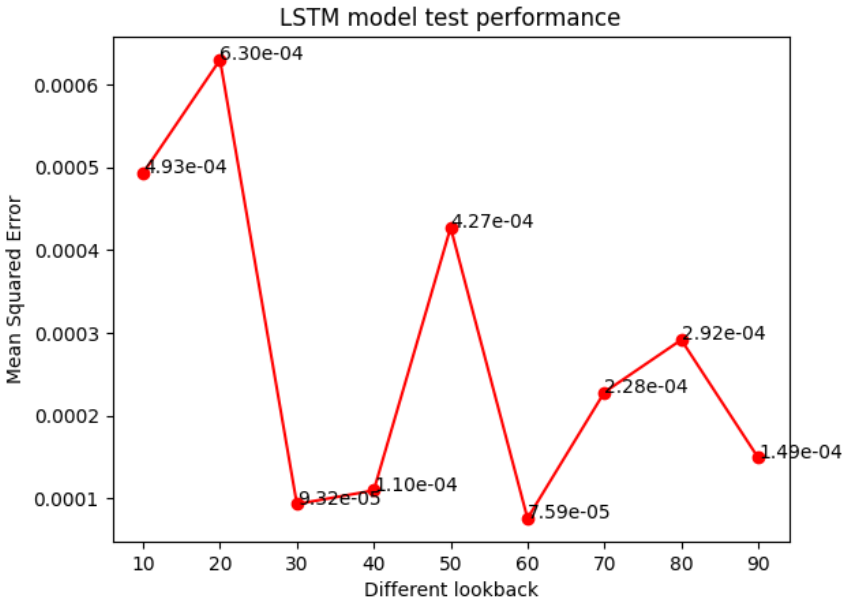

Figure 21: the MSE of each local best LSTM model under different lookback.

In additional, to explore and evaluate the LSTM model ability on this topic and dataset, more indictors are computed and used for training the model. The dataframe expand by adding 8 columns, which are represent seven days moving average of close price, the return of each day computed by using the gap between close price of two adjacent days, the gain and loss computed by using the return value former, the average gain and loss of 14 days. Based on the average loss and gain, the Relative Strength (RS) and the Relative Strength Index (RSI), which is a technical indicator used to measure the strength of a security's price action, can be easily computed [16]. It is expected that the LSTM can using a more complex structure to predict better with the help from these new indictors. However, the specific experiment result is shown below (figure 22), which shows that the more indictors do not help the model to reach a higher performance.

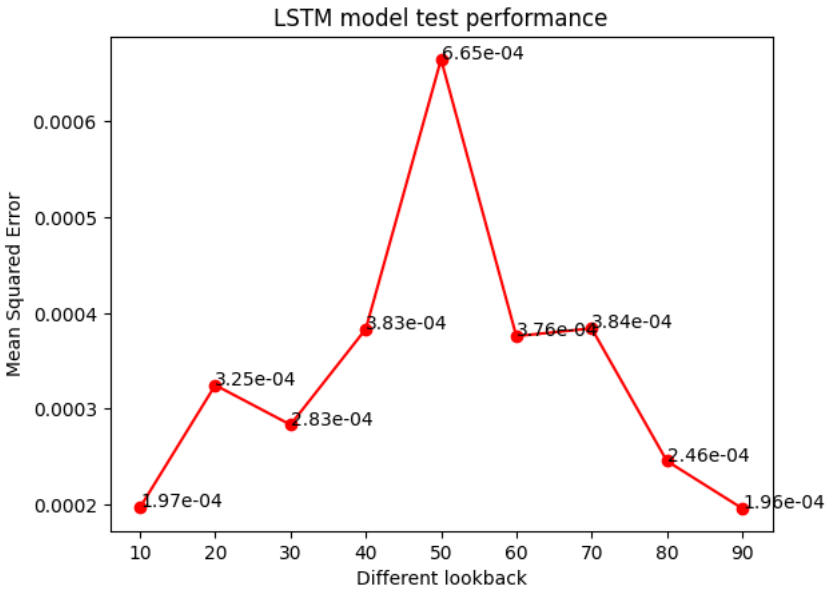

Figure 22: the LSTM performance with more new indictors.

For the CNNs + LSTM model, a function called data_split3 are built following plan in 2.3 'Data Split and Feature Selection Design and Implementation'. After data preparation, three CNNs are defined based on the three different timeframes. These three CNNs take different shape of data generated by sliding window as input, to extract the features from three different period. These features are fed into the LSTM model later. The parameter combination of this LSTM will also be found through the *GridSearchCV* as the previous LSTM training. However, since the limited time is not enough to develop and evaluate a global optimal LSTM for the combined model, the LSTM is just defined as the instruction from the previous LSTM result. It is a significant limitation of this project. Since the previous global optimal parameter combinations only guarantee under its only situation, which means this structure and the parameter combination is not the best solution for this CNNs+LSTM model. More detail will be analyzed in the later limitation chapter. The Train result of such a combined model with this LSTM structure is shown below as figure 23. The most possible reason of such a model with a higher mean squared error (worse performance) than other model is the structure are not optimized. The result only shows that the three CNNs with the timeframe = 10, 20, 30 respectively plus the LSTM model whose structure is two LSTM layer with units number = 100 and two Dropout layer with the dropout rate = 0.1, this model setting do not have a better performance than the previous optimal individual LSTM model, which do not prove the conclusion of Selvin's experiment [4].

```
Epoch 42/50
145/145 [==============================] - 1s 8ms/step - loss: 2.2706e-04
Epoch 43/50
145/145 [==============================] - 1s 8ms/step - loss: 2.0617e-04
Epoch 44/50
145/145 [==============================] - 1s 9ms/step - loss: 1.9031e-04
Epoch 45/50
145/145 [==============================] - 1s 8ms/step - loss: 2.2179e-04
Epoch 46/50
145/145 [==============================] - 1s 9ms/step - loss: 2.5128e-04
Epoch 47/50
145/145 [==============================] - 2s 12ms/step - loss: 2.3333e-04
Epoch 48/50
145/145 [==============================] - 2s 12ms/step - loss: 2.1505e-04
Epoch 49/50
145/145 [==============================] - 1s 9ms/step - loss: 2.0565e-04
Epoch 50/50
145/145 [==============================] - 1s 8ms/step - loss: 2.2839e-04
37/37 [==============================] - 1s 3ms/step
The mean squared error is 1.23e-04
```

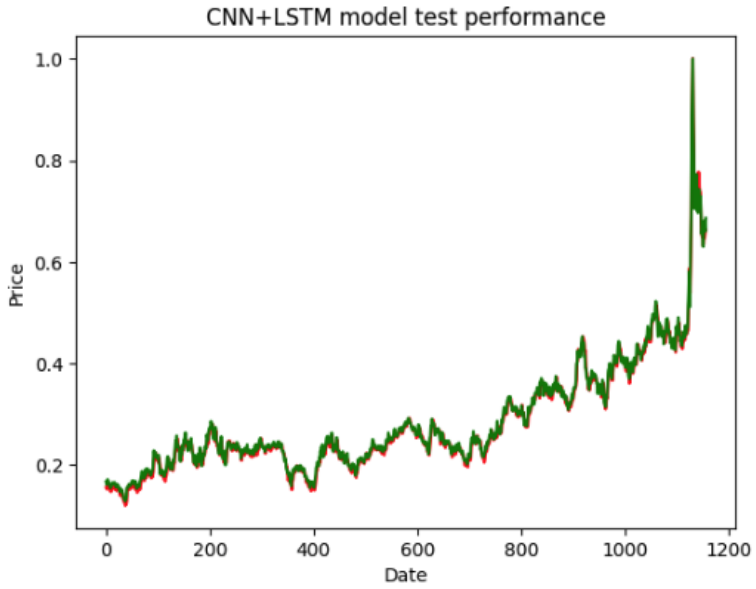

Figure 23: the training result and evaluation of combined model with a specific structure

2.6 Limitation about Design and Implementation

There still exist the limitation of the project design and implementation. First is setting the range of the parameters. The options of potential parameters range are limited by the computing power, since adding the options will cause the parameter space to expand exponentially, making it more challenge and exhaustive to explore the entire space. It also may need more times and epoch to optimize. More importantly, if the granularity of the search space is too fine, the optimization algorithm may be trapped in local optimum and fail to converge to the global optimum.

Secondly, use only one non-linear machine learning models (SVR), which compared with the linear regression to figure the relationship type behind the price on the topic, linear or non-linear. If more non-linear models can be implemented, like Decision Tree Regression or other possible models. Using result of several model performances

compared with LR, have more convincing to prove the conclusion got from it.

The test dataset used in the evaluation is not same, which is not a really controlled experiment to compare every model and parameter combination performance. However, as for the comparation between different parameter combinations, it is hard to use the same test dataset since the *lookback* parameter indicate how sliding windows work, which will generate different input shape dataset. In other words, the different parameter combination from one basic model will create models which take different input dataset shape. Based on this reason, the different optimal models from different basic model also can not be compared using a same input dataset to complete a controlled experiment, because they may have different input shape requirement. Therefore, there do not exist a so-called controlled experiment for this project, but the experiment is indeed controlled by fixing the random seeds used for each time are same, which always equal 42. Moreover, the SVR and Linear regression model use different input, so it can not be seen as a controlled experiment about the relationship type behind the commodities prices.

Anther limitation is optimization issue of CNNs+LSTM structure. As above discussion, the parameters such as timeframe for three CNN, the structure of LSTM are all not full explored. The result is not persuasive because of that.

III. Testing & Evaluation Result

1. Analysis and Comparison Different Model

The results of Evaluation are all shown in figures above. Based on the figure 14, it shows that the linear regression model has the relatively best performance with lookback = 75 using only the close price to predict. Since it has the lowest mean squared error, $9.61e^{-05}$. Meanwhile, the relatively best model for SVR has the lowest mean squared error is $7.71e^{-04}$, with the parameter combination is: *lookback* = 15, *C*: 100, *gamma*: 0.1. This result between these two figures shows that, only under these potential parameter range, the optimal Linear regression model have a better performance than SVR. However, as the limitation talked above, since the different input test dataset, it has lots of problem with the conclusion. The purpose of setting these two non-deep learning models is showing potential of their performance.

Furthermore, the deep learning models like LSTM (Long Short-term memory model) with original input data and with additional new indicators like gap, gain, loss, RSI (Relative Strength Index), their performance figures are Figure 21 and Figure 22 respectively. The LSTM with original input dataset can reach the lowest mean squared error = $7.59e^{-05}$, while the LSTM model with new indicators have $1.96\ e^{-04}$ as the lowest mean squared error. This shows that having more new indicators as input will not help the complex structure deep learning models to improve their performance. The

preliminary feature selection is still necessary for the model training.

As for the combined model, CNNs+LSTM, has the performance with the mean squared error = $2.2839e^{-04}$. It is not a good performance regarding the performance of individual LSTM. However, the main reason is its parameters and structure have not optimized yet.

2. Discussion of Findings

Based on the limitation and compared result analyzed above, some conclusions can be got under conditions: 1. Only in the parameters range introduced before, the SVR does not show that it has a better performance than Linear regression. 2. In the same parameter range, the LSTM with more indictor as input do not have the performance as good as the LSTM model with original dataset.

IV. Conclusion

1. Summary of the Study

This project aims to using different models to predict commodity price from the Kaggle dataset [5]. The Linear Regression model, Support Vector Regression model, Long Short-term memory model and the CNNs+LSTM combined model are implemented respectively. These models are different kinds of machine learning model, to explore the different performance of prediction with this dataset. More specifically, the Linear Regression model represent the prediction based on the linear relationship between the previous price and prediction. The Support Vector Regression is one of non-linear machine learning model, which try to model the non-linear relationship between input price data with prediction price output to make the prediction. And LSTM as a refined type of RNNs deep learning model, can handle large-scale datasets, capture nonlinear relationships, and automatically extract relevant features from raw input data[18]. Depending on the complex network structure, for this topic, LSTM network is an excellent model which can learn patterns and relationships in the historical data and make predictions based on past price movements [19]. Therefore, to check the ability to learn and handle the features from original input dataset, more indicators are developed and trained with a completely new LSTM model. In the end, a CNNs+LSTM combined model is simply implemented. As conclusion from the material found, this combined model structure allow it has a improve performance. But since the limitation of time and hardware, the model uses the same parameters returned from *GridSearchCV*. And performance of this parameter does not support the conclusion.

2. Recommendations for Future Research

However, limited by time and hardware, only a range of the parameter are explored. So

for the future research which also wants to check and compared several models for this dataset, training and evaluated larger range of parameter combination and more models are recommended. More specifically, explore more advanced deep learning architectures, such as Transformer-based models (e.g., BERT), attention mechanisms, or graph neural networks, which can capture complex patterns and dependencies in commodity price data. Moreover, combined more model to get a outperform model. For example, using weight to build a hybrid model which can combines several well-trained deep learning models in forecasting daily commodity prices. Individual model may have different advantages and disadvantages. And a hybrid model can have more possible prediction from different models for a specific period. The disadvantages from model's algorithm and structure can be offset by other models. The final output will be decided by all models. For example, use the Mean Absolute Percentage Error (MAPE) to measure the accuracy from different models and sign a weight based on this indicator. So that, the hybrid model can finally have a relatively reasonable output.

Moreover, to train the model well, future research can find and consider other data sources. Like using additional data source which contain the commodity price not only from 2000 to 2022, but also have the data from 1990 to 2000 and from 2022 to 2023. Additional data will help with model training and allow them learnt and predict in a higher accuracy.

As for the evaluation, future research can use more technique to represent the performance. Not only use the *matplotlib.pyplot* library to visualize. More vivid represent would help enhance the understanding in the training and prediction, especially in a project about financial market.

Furthermore, there are still more topics waiting to be explored. Like using reinforcement learning model to create an active and dynamic trading strategies or implement a real-time prediction model to handle the streaming and act in a high speed and high frequency.

These recommendations aim to inspire any possible future research about commodity price prediction. Most of them are thinking or personal conjecture, which can not guarantee the direction is right and helpful.

V. BCS Criteria & Self-Reflection

1. *'An ability to apply practical and analytical skills gained during the degree programme'*: this point shows everywhere in this project. For example, the programming ability which support to implement the whole project.

2. *'Innovation and/or creativity'*: as concept of deep learning model says, compared to machine learning model need human to finish the features selection work, the deep

learning has the ability to exact the feature itself. Therefore, a controlled experiment is set. Two LSTM model using same parameter range using two different datasets as input. They are trained in the same method and show their performance of trained models. As the result, more indicators will not improve the model performance at 100%. The simple feature selection is still necessary to help the model well to be trained.

3. *'Synthesis of information, ideas and practices to provide a quality solution together with an evaluation of that solution'*: this point can be shown by the example of previous one.

4. '*That your project meets a real need in a wider context.*': like the evaluation of performance above, these model all have the ability to predict commodity price at a relatively high accuracy.

5. '*An ability to self-manage a significant piece of work.*': although the project is slightly changed from the *Detailed Proposal,* the project continue being pushed during the semester as plan.

6. '*Critical self-evaluation of the process.*': This project implements several models for commodity price prediction, the Linear Regression model, the Support Vector Regression model, the Long Short-term memory model with original and new indicators and the CNNs+LSTM combined model. However, since the different controlling condition, compared these different models can not give any powerful and convincing conclusion this model outperform than other models. Therefore, it indicates that the design of whole project workflow is not reasonable. The main reason is author did not have a full understanding about the topic at first. In the beginning the plan is showing a well-trained LSTM model with its parameter combination. During doing the work of this topic and reading more and more materials and paper, more useful and meaningful models are found, which author really wants to implement and check their performance. And an assumption about 'the non-linear model like SVR will outperform than linear regression, and CNNs+LSTM will outperform than the individual LSTM' is pre-set before. To prove this assumption, change the design of this project to current one. However, some results of experiment are unexpected.

VI. References

[1]: "Commodity Market Definition." Investopedia, 19 May 2021, www.investopedia.com/terms/c/commodity-market.asp

[2]: Miah, S. J., Kerr, D., & Gammack, J. (2020). A Framework for Industry 4.0 Implementation: The Case of SMEs. Intelligent Systems in Accounting, Finance and Management, 27(1), 30-44. doi: 10.1002/isaf.1459.

[3]: Kim, D. H., Kim, Y. H., & Yoo, S. H. (2020). Commodities: Fundamentals and Time Series Forecasting Models of Daily Prices. Energies, 13(18), 4691. doi: 10.3390/en13184691.

[4]: S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 2017, pp. 1643-1647, doi: 10.1109/ICACCI.2017.8126078.

[5]: Prasertk, R. (2021). Major commodity prices from 2000-2022. Kaggle. Retrieved April 26, 2023, from https://www.kaggle.com/datasets/prasertk/historical-commodity-prices-from-20002022.

[6]: Araújo, R. A., de Almeida, D. F., & Ohashi, R. H. (2021). Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. Finance Research Letters, 42, 102280. doi: 10.1016/j.frl.2021.102280.

[7]: Majhi, S., & Panda, G. "Forecasting Commodity Prices Using Long-Short-Term Memory Neural Networks." In A. Pandey et al. (eds.), Intelligent Techniques and Their Applications in Environmental Monitoring and Pollution Control, pp. 39-47, Springer, 2021, ISBN 978-981-16-0765-5.

[8]: Montgomery, D.C., Peck, E.A., and Vining, G.G. "Introduction to Linear Regression Analysis," 5th ed., Wiley, 2012, ISBN 978-0470542811.

[9]: G. Bathla, "Stock Price prediction using LSTM and SVR," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2020, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.

[10]: Smola, A.J., Schölkopf, B. A tutorial on support vector regression. Statistics and Computing 14, 199–222 (2004). https://doi.org/10.1023/B:STCO.0000035301.49549.88

[11]: C. R. Madhuri, G. Anuradha and M. V. Pujitha, "House Price Prediction Using Regression Techniques: A Comparative Study," 2019 International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 2019, pp. 1-5, doi: 10.1109/ICSSS.2019.8882834.

[12]: I. A. Budiastuti, S. M. S. Nugroho and M. Hariadi, "Predicting daily consumer price index using support vector regression method," 2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering, Nusa Dua, Bali, Indonesia, 2017, pp. 23-28, doi: 10.1109/QIR.2017.8168445.

[13]: Graves, A. (2012). Long Short-Term Memory. In: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, vol 385. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24797-2_4.

[14]: S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[15]: L. -C. Cheng, Y. -H. Huang and M. -E. Wu, "Applied attention-based LSTM neural networks in stock prediction," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4716-4718, doi: 10.1109/BigData.2018.8622541.

[16]: Qiujun Huang, Jingli Mao and Yong Liu, "An improved grid search algorithm of SVR parameters optimization," 2012 IEEE 14th International Conference on Communication Technology, Chengdu, 2012, pp. 1022-1026, doi: 10.1109/ICCT.2012.6511415.

[17]: Wilder, J.W. (1978). New Concepts in Technical Trading Systems. Trend Research.

[18]: B. Batres-Estrada, 'Deep learning for multivariate financial time series', Dissertation, 2015.

[19]: K. Chen, Y. Zhou and F. Dai, "A LSTM-based method for stock returns prediction: A case study of China stock market," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 2015, pp. 2823-2824, doi: 10.1109/BigData.2015.7364089.

VII. Appendices

All the Figure and supporting material are represented during the main body. To support the reading and understanding better.