NATIONAL UNIVERSITY OF HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROJECT REPORT

STATISTICAL ANALYSIS

TOPIC: COFFEE PRICE FORCASTING

Teacher: Ths. Đỗ Duy Thanh

Class: ACCT5123.N21.CTTT

Group: 4

Nguyễn Nhất Thưởng	20522000
Lê Quang Hoà	20521331
Kiều Xuân Diệu Hương	20521381

Ho Chi Minh City, July 10, 2023

Chapter 1: INTRODUCTION

1.1 Introduction

The coffee industry plays a significant role in the commodity agriculture sector, ranking as the fifth most consumed product in the global market with a value of approximately $70.68 billion (2011). In Vietnam, coffee is also one of the attractive industries, leveraging the advantages of a tropical climate, year-round seasonal winds, and vast fertile basaltic land. In fact, since the 1990s, coffee cultivation has experienced significant development, creating employment opportunities for a substantial portion of the workforce. Moreover, coffee cultivation helps reforest barren hills, providing additional avenues for economic development and minimizing social issues and natural disasters. In the era of global integration, coffee exports not only drive the modernization of the country's machinery but also serve as a significant component of the nation's trade balance in all international trade relations. Coffee exports account for 10% of the country's total export turnover, making it a key export commodity second only to rice. It can be said that the coffee industry holds a significant position and plays a vital role in economic development, providing employment opportunities, income for many households, and contributing to a substantial export value for the country.

Therefore, predicting coffee prices is crucial for coffee businesses to plan production, manage supply and demand, price their products accurately, and make effective business decisions.

1.2 Research Objectives

The objective of the project is to forecast coffee prices using deep learning models. Deep learning is a subset of machine learning that utilizes neural networks with multiple layers to extract complex patterns and make predictions. By applying deep learning techniques to historical coffee price data, the project aims to develop accurate forecasting models that can predict future coffee prices.

There some of main objectives of the project:

- Data Collection: Gather historical data on coffee prices and factors such as production, consumption, and market conditions that may impact prices.
- Data Processing: Clean and preprocess the collected data.
- Model Selection: Explore and select appropriate models for time series forecasting, such as LSTM network, or transformer models.
- Model Evaluation and Comparison: Compare the performance of different deep learning models to identify the most effective model.
- Performance Monitoring: Building an interactive website for monitoring the performance of the forecasting models.

1.3 Research Method

The data is collected from various sources, including reference materials such as books, newspapers, and the internet. It is then cleaned, preprocessed, normalized, and feature engineered. Subsequently, deep learning models are used for the prediction process. The predicted results are analyzed, summarized, evaluated, and compared. The models used for prediction include two deep learning models and one machine model. Additionally, the models are evaluated using commonly used performance metrics such as Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE).

1.4 Research Subjects and Scope

- Client: Targeted for business. The client wants to explore the global coffee market.
- Research Subject: Robusta instant coffee, espresso, and pre-mixed coffee sold in supermarkets. -Scope: Global coffee market.
- Timeframe: Period 1/7/2008-7/7/2023

Based on the desire to understand and develop time series forecasting models and explore the importance of ERP applications in business operations, we can proceed with building three following models

- GRU: A gated recurrent unit that addresses the vanishing gradient problem in RNNs.
- LSTM: A recurrent neural network architecture with memory cells and gates to capture long-term dependencies.
- XGBoost: An ensemble learning algorithm based on gradient boosting for efficient and accurate modeling.

Finally, we will build a website to display the forecasting results of the models. Additionally, the website will be used for comparing and evaluating the forecasted outcomes.

1.5 Structure of The Project

Chapter 1: Introduction The introduction emphasizes the economic significance of the coffee industry and its impact on employment opportunities. The chapter also outlines the research objective of developing deep learning models for precise coffee price forecasting and mentions the implementation of a website to display and evaluate the forecasted results.

Chapter 2: Situational Survey This chapter delves into the historical background of Vietnamese coffee, starting from its introduction by French missionaries in 1857 and its subsequent growth and expansion across different provinces and regions. It highlights the emergence of Robusta coffee as a distinguished product in the Central Highlands, specifically Buon Ma Thuot. The chapter also introduces market research and consumer surveys as essential components of the situational survey, which lay the foundation for further analysis and insights in subsequent sections.

Chapter 3: Dataset This chapter focuses on the data collection process and the dataset used for coffee price analysis. It includes historical records of variables such as Robusta coffee price, sugar price, oil price, and the US dollar index. The analysis highlights the positive correlations between sugar and oil prices with coffee prices, with a focus on using oil prices as a predictor for forecasting.

Chapter 4: Models This chapter presents LSTM and GRU as popular models for sequential data processing. GRU, a type of recurrent neural network, addresses the vanishing gradient problem and efficiently retains important information. These models have shown promise in various applications like natural language processing and time series forecasting.

Chapter 5: Implementation This chapter discusses the experimental setup for implementing the LSTM, GRU, and MLP models. It presents the configuration details and parameters used for each model.

Chapter 6: Conclusion The developed LSTM, GRU, and MLP models show promising performance in coffee price forecasting. However, limitations such as reliance on historical data and the exclusion of certain factors are acknowledged. Future directions include incorporating external factors, exploring ensemble models, and developing real-time forecasting capabilities with measures of uncertainty and risk.

Chapter 2: Situational Survey

2.1. History of Vietnamese Coffee

The arabica coffee variety was first introduced to Vietnam in 1857 by French missionaries. It was initially planted experimentally in Catholic churches in northern provinces such as Ha Nam and Phu Ly. Subsequently, coffee cultivation expanded to provinces like Thanh Hoa, Nghe An, and Ha Tinh. It then spread to the central provinces of Quang Tri and Quang Binh. Despite the war, coffee farms in these areas continued to thrive, leading to a high concentration of arabica coffee. Eventually, coffee production gradually extended to the Central Highlands and the Southeast region, with the realization that the Central Highlands, particularly, was the most suitable area for coffee cultivation.

After the introduction of arabica coffee to Vietnam in 1857, the French further imported two coffee varieties in 1908: Robusta and Liberia. Over time, the French colonizers found that arabica coffee did not yield high economic benefits, prompting them to bring Robusta coffee from Congo to be grown in the Central Highlands. The Robusta coffee plant thrived in this region, and the coffee cultivation area continued to expand. As a result, the Central Highlands, particularly Buon Ma Thuot, became known for having the largest and most renowned Robusta coffee plantations in the country, symbolizing a source of pride for the region.

Through the course of Vietnamese history, the land of the Central Highlands gradually emerged as the convergence point of all favorable ecological and soil conditions for coffee growth and development. Robusta coffee, therefore, became an iconic and prestigious product of the Central Highlands, especially Buon Ma Thuot, in the context of international economic integration.

2.2. Market Research

According to a recent USDA report, global coffee production for the 2023-2024 crop year is forecasted to increase by 4.3 million bags (60 kg/bag), a 2.5% rise from the previous year, reaching 174.3 million bags. Among them, higher production in Brazil and Vietnam is expected to compensate for the decline in Indonesia.

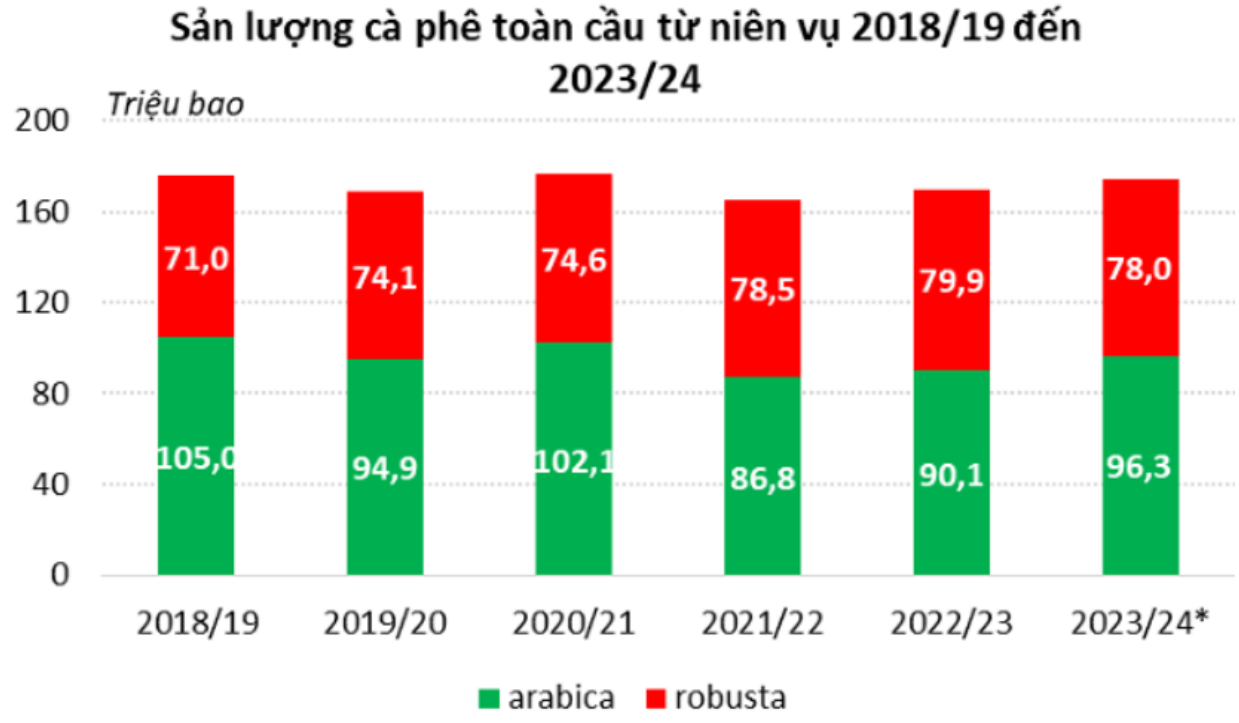


Figure 1 The global coffee production from the 2018/19 crop year to the 2023/24 crop year.

With the additional supply, global coffee exports are projected to increase by 5.8 million bags, reaching a record 122.2 million bags, primarily driven by significant export growth from Brazil. Global coffee consumption is also forecasted to reach a record 170.2 million bags in the 2023-2024 crop year, an increase of nearly 2 million bags compared to the previous year. As a result, end-of-season stocks are expected to remain low at 31.8 million bags.

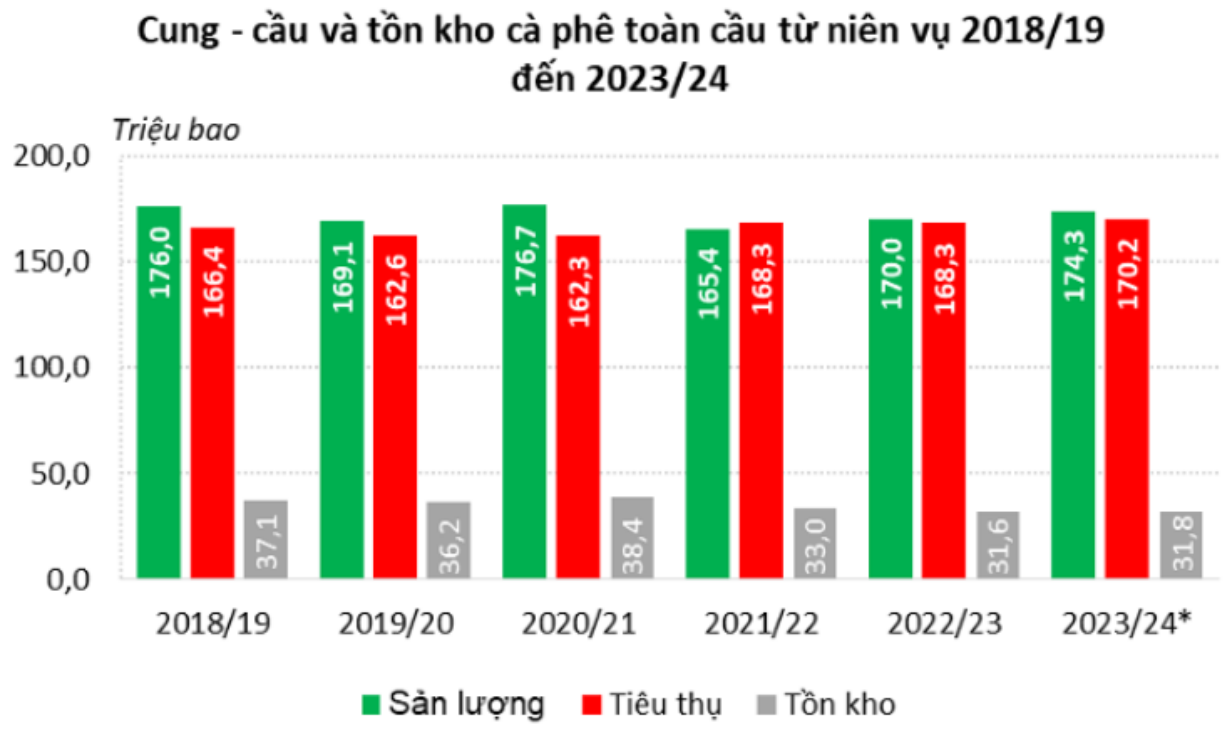


Figure 2: Supply and Demand and global inventory data from mission 2018/19 to 2023/24

Top Three Coffee Summary -Thousand 60-Kilogram Bags

	2018/19	2019/20	2020/21	2021/22	2022/23	Jun 2023/24
Robusta Production						
Vietnam	29,336	30,200	28,050	30,480	28,740	30,230
Brazil	16,800	18,500	20,200	21,700	22,800	21,700
Indonesia	9,400	9,450	9,400	9,300	10,500	8,400

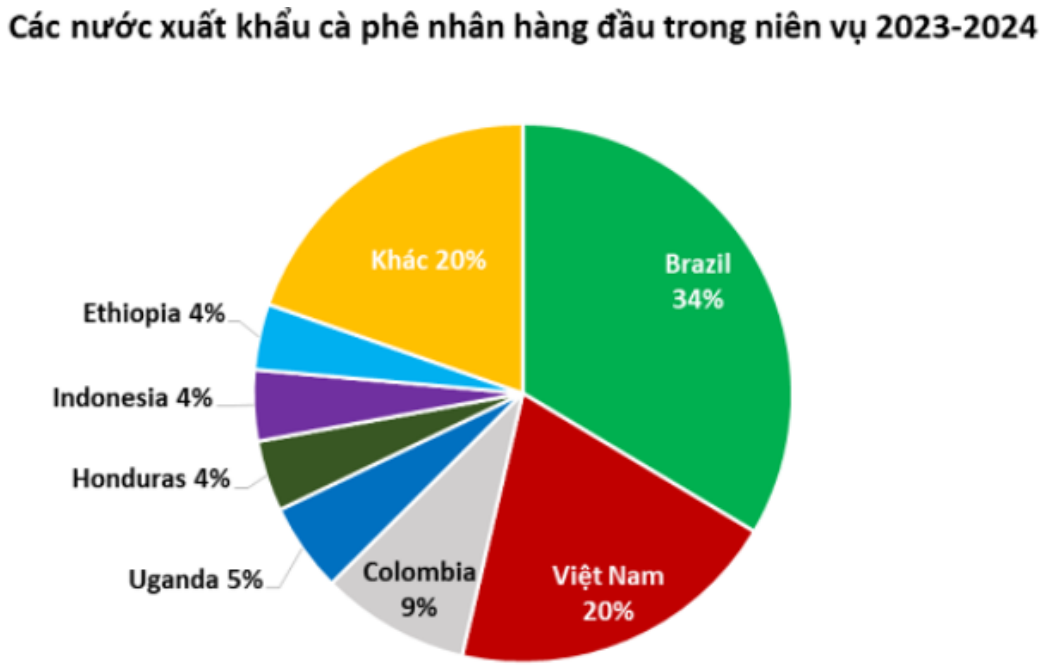


Figure 3: Pie chart of top coffee exporter

Brazil: Brazil's 2023-2024 harvest is forecasted to increase by 3.8 million bags to reach 66.4 million bags. Among them, arabica coffee production is projected to increase by 4.9 million bags to reach 44.7 million bags. Earlier this year, Brazil's leading coffee-producing region, Minas Gerais, experienced above-average rainfall during the coffee fruit development stage, causing difficulties for farmers in controlling diseases and pests.

However, the increased rainfall resulted in larger coffee bean sizes compared to the previous harvest, contributing to the production increase. Although the forecasted production is showing a rebound, it is still lower than the peak of nearly 50 million bags in the previous seasons. Arabica coffee plants in many Brazilian growing regions continue to recover from severe frost, high temperatures, and below-average rainfall in 2021, which led to decreased coffee production in the 2021-2022 and 2022-2023 crop years.

Meanwhile, Brazil's Robusta coffee production is forecasted to decrease for the first time in six consecutive years, with a reduction of 1.1 million bags to reach 21.7 million bags. Reduced rainfall and lower temperatures during the flowering stage have

affected production in Espiritu Santo, Brazil's main Robusta coffee region. Brazil's coffee exports are expected to recover and increase by 8 million bags compared to the previous year, reaching 41 million bags, driven by higher supply and reduced global inventories.

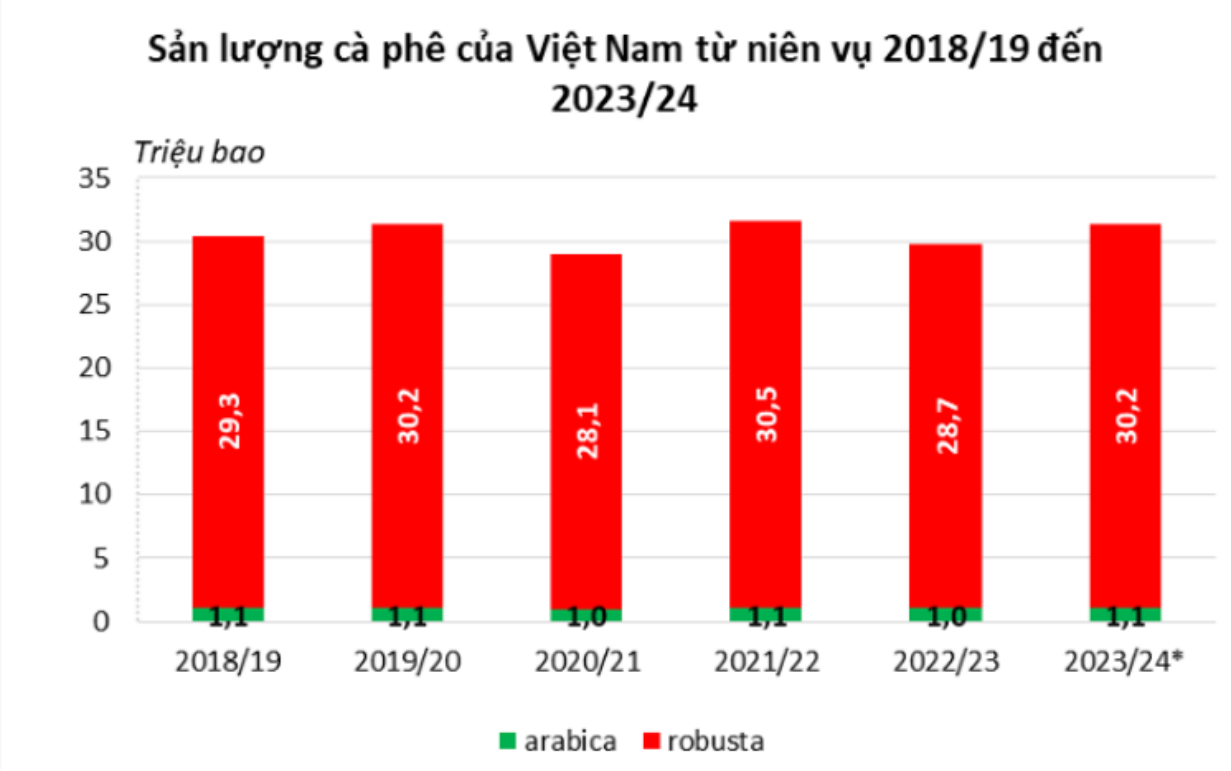


Figure 4 The coffee production of Viet Nam from the 2018/19 crop year to the 2023/24 crop year.

Vietnam: Vietnam's coffee production for the 2023-2024 crop year is forecasted to increase by 1.6 million bags (5%) to reach 31.3 million bags due to favorable weather conditions. The production area is expected to remain unchanged, with nearly 95% of the total production still being Robusta coffee. The forecasted rainfall is expected to be 10-20% higher than average, supporting irrigation and coffee plant development. Farmers are gradually replanting coffee trees with new varieties that have better productivity and disease resistance, resulting in increased crop yields.

Some USDA sources in the Vietnamese coffee industry are even more optimistic, predicting the possibility of total coffee production being 5-10% higher than the previous year. Farmers are actively replanting coffee trees with improved varieties,

leading to increased productivity for the coffee crop. However, Vietnam's green coffee exports are expected to decrease by 1.5 million bags to 24.5 million bags, based on lower starting stocks and stricter import regulations from the European Union. USDA forecasts that Vietnam's end-of-season stocks for the 2023-2024 crop year will increase by 1 million bags compared to the previous year, reaching 2.7 million bags.

2.3. Consumer Survey

According to statistics from the ICO, global coffee exports reached over 10.1 million bags in April this year, a 2.6% decrease compared to the 10.4 million bags in the same period last year. Accumulated data for the first 7 months of the 2022-2023 coffee year (October 22, 2022, to April 23, 2023) shows a 6.2% decline in global coffee exports, totaling 72.2 million bags compared to the previous year. Arabica coffee exports for the 12-month period ending in April 2023 totaled 75.2 million bags, a 9.6% decrease from the previous year, while Robusta exports reached 48.5 million bags, a 1% decrease.

Green coffee beans alone accounted for over 90% of the total global coffee export volume, with 9.2 million bags in April, a 2.9% decrease compared to the same period last year. This marks the fifth consecutive month of decline in global green coffee bean exports since the start of the 2022-2023 coffee year. Overall, global green coffee bean exports for the first 7 months of the 2022-2023 coffee year reached 64.9 million bags, a 6.4% decrease from the previous year.

In the past 7 months, exports of Brazilian arabica green coffee decreased by 9% to 21 million bags, while other arabica varieties decreased by 13.8% to 11.2 million bags. Colombian arabica exports also declined by 15.3% to 6.3 million bags. On the other hand, Robusta exports increased to 26.4 million bags compared to 25.8 million bags in the same period of the 2021-2022 coffee year.

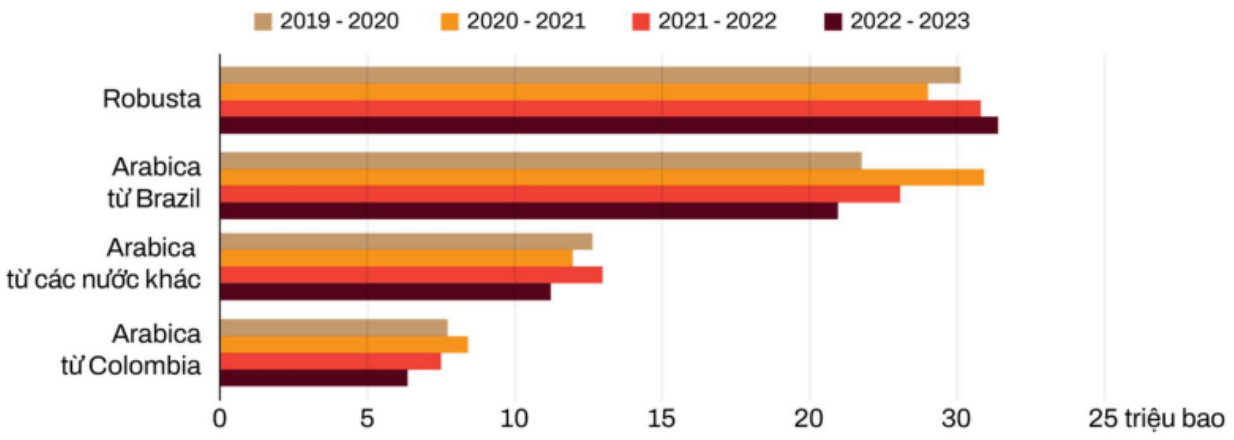


Figure 5: Global green coffee exports in the first seven months of the 2022-2023 crop year (October 2022 to April 2023) (Source: ICO).

With these results, the share of Robusta coffee in global coffee exports has increased to 40.6% from the previous crop year's 37.2%. This is also the highest proportion of Robusta in the structure of green coffee exports in recent crop years. In contrast, the share of arabica has decreased from 62.8% to 59.4%.

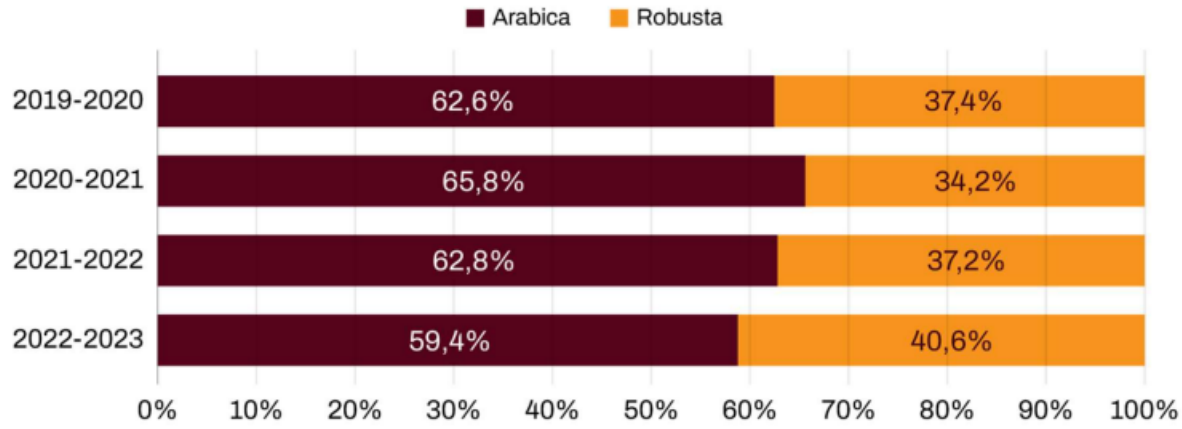


Figure 6:The proportion of Arabica and Robusta coffee in the total global green coffee exports (Source: ICO)

2.4. Supply Survey

Coffee exports have declined in most supplying countries. In South America, exports decreased by 6.4% to around 3.6 million bags in April, primarily due to Brazil,

Colombia, and Peru, which saw a combined export volume reduction of 17.9%. Colombia experienced a 2.5% decline, reaching 2.7 million bags, while unfavorable weather negatively impacted coffee production and exports, resulting in a 6% decrease. Peru witnessed a significant 62.5% drop in exports due to adverse weather conditions and political instability in key production regions.

Similarly, coffee exports from Africa declined by 9.8% compared to the same period last year, reaching 0.9 million bags in April. The cumulative export volume for the first seven months of the current season in Africa reached 6.9 million bags, down 5.9% compared to the same period last year. Ethiopia, Kenya, and Uganda, major coffee-exporting countries in the region, experienced declines of 17.6%, 25.8%, and 8.4% respectively. Uganda's coffee exports were affected by drought and lower exports to Sudan, while in Ethiopia, contract disputes arising from domestic and global price differentials impacted coffee exports.

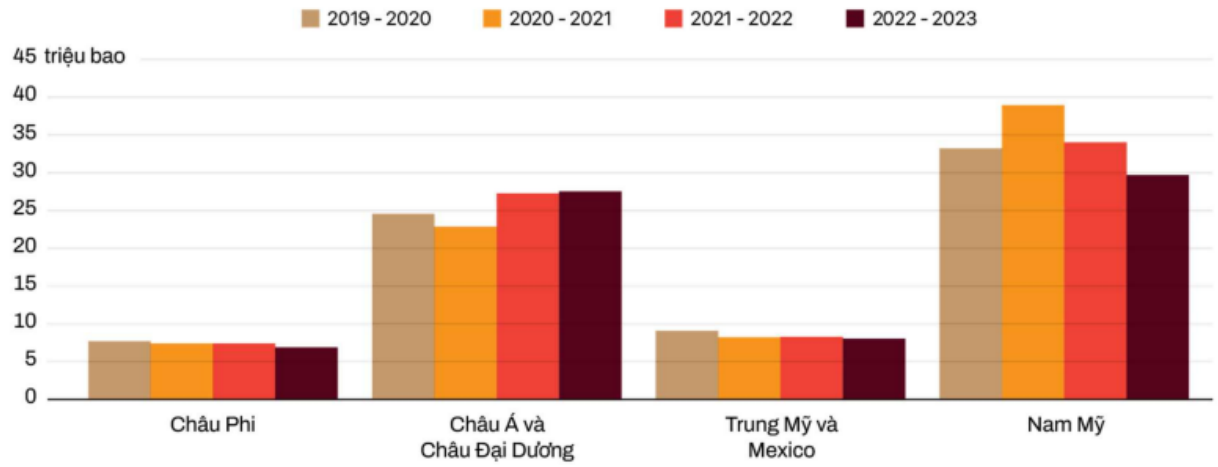


Figure 7:The coffee exports from different regions in the first seven months of the 2022-2023 season (Source: ICO)

In Central America and Mexico, coffee exports increased by 6.3% to 1.9 million bags in April. Costa Rica, Honduras, and Nicaragua showed growth rates of 27%, 13%, and 11.2% respectively. However, the cumulative exports for the first seven months of the current season in this region declined by 4% to 8.01 million bags. In Asia and Oceania, coffee exports decreased by 1% to slightly over 3.7 million bags in April 2023.

However, the cumulative exports for the first seven months still increased by 1.1% to 27.5 million bags, making it the only region with export volume growth in the 2022-2023 season. In April, Indonesia's coffee exports decreased significantly by 24.6%, surpassing the positive growth of 3.1% in Vietnam, the region's leading producer and exporter.

Vietnam's coffee exports increased by 6.7% to 19.18 million bags in the first seven months of the current season, compared to 17.97 million bags in the same period last year. This growth can be attributed to a 14.9% increase in domestic production during the 2021-2022 season and the challenges faced by other major Robusta coffee producers in sourcing supply.

During the same period, Brazil's Robusta coffee exports decreased by 36.1%, while India and Uganda experienced declines of 31.1% and 6.2% respectively. Indonesia's exports also appeared to be negatively affected, with an estimated 4.7% decrease in the 2021-2022 season due to lower production.

Chapter 3: Dataset

3.1. Data Collection

Several factors influence the price of coffee, which can vary over time and across different regions. Understanding these factors is crucial for analyzing and predicting coffee prices. Some factors we take into consideration:

- Costs of material for production: Materials or raw materials required for product manufacturing, including both direct and indirect materials.
- Currency Exchange Rates: Coffee is a globally traded commodity, and fluctuations in currency exchange rates can affect its price. As coffee is often priced in U.S. dollars, changes in the value of other currencies relative to the dollar can impact the affordability and competitiveness of coffee in international markets.

- Transportation Costs: Oil prices can influence transportation costs, including shipping and logistics expenses. As coffee is a globally traded commodity, it often undergoes extensive transportation from coffee-producing regions to consuming markets.

The data source contains daily historical records of several variables that are relevant to the analysis of coffee prices from 14/1/2008 to 6/7/2023, specifically focusing on Robusta coffee. These variables include the price of Robusta coffee, sugar price, oil price, and the US dollar index.

- Robusta Coffee Price: This data source contains daily historical records of the price of Robusta coffee, which is a variety of coffee extensively cultivated and traded globally, including in Vietnam. As Vietnam is one of the largest producers of Robusta coffee, monitoring its price is essential for understanding the local and international coffee markets.
- Sugar Price: The dataset for sugar prices contains the daily historical records of sugar prices. Sugar is often used as a key ingredient in various food and beverage products, including coffee. Changes in sugar prices can influence the cost of production for coffee products.
- Oil Price: This data source provides daily historical records of oil prices, which are important in understanding the broader economic and market dynamics. Oil prices can have indirect impacts on the coffee market through factors such as transportation costs and production costs.
- US Dollar Index: The US dollar index measures the value of the US dollar relative to a basket of other major currencies. It reflects the strength or weakness of the US dollar in international currency markets. The dataset for the US dollar index captures the daily historical records, which are relevant for analyzing the relationship between currency exchange rates and commodity prices, including coffee.

3.2. Data Analytics

The topic utilizes a dataset that is collected from various sources, including reference materials such as books, newspapers, and the internet. The dataset consists of 3867 rows, comprising historical data of price, oil price, and Us Dollar Index price from January 14, 2008, to July 6, 2023.

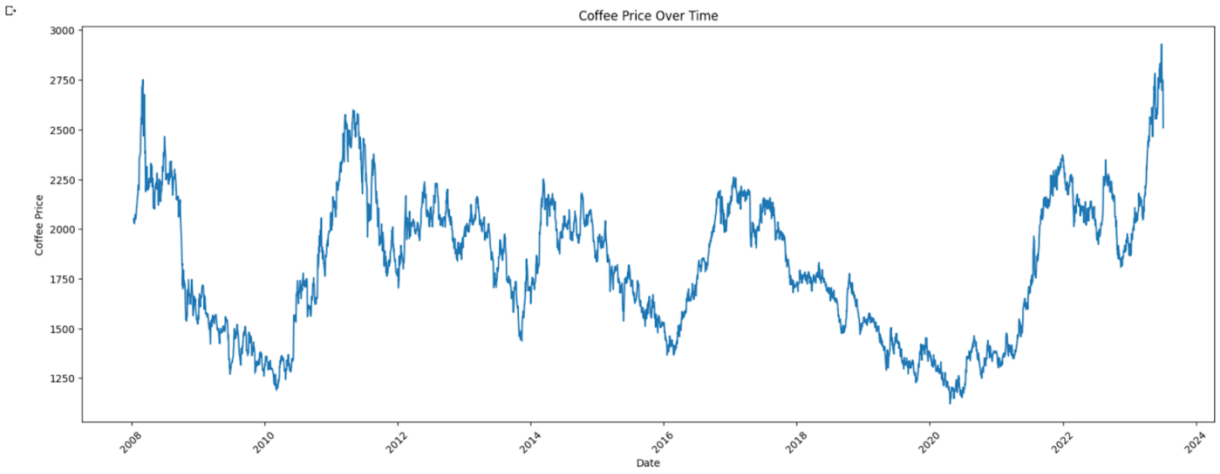


Figure 8: Historical Price of Coffee (USD/ MetricTon)

Correlation coefficient is a concept in statistics that measures the level of relationship or correlation between two variables. It quantifies the extent to which two variables change together and have a directional correlation, meaning that when one variable increases, the other variable also increases (positive correlation), or when one variable increases, the other variable decreases (negative correlation). The correlation coefficient is measured using a coefficient that ranges from -1 to 1. A value of -1 indicates a perfect negative correlation, 1 indicates a perfect positive correlation, and 0 indicates no linear correlation between the two variables.

$$\rho_{xy} = \frac{\mathrm{Cov}(x, y)}{\sigma_x \sigma_y}$$

Equation 1: Correlation Coefficient

Sugar Price	Oil Price	US Index
0.3	0.54	-0.13

Table 1: Correlation Coefficient between factors and coffee price

There are three features that may have an impact on coffee prices: Sugar Price, Oil Price, and US Index. The positive correlation between Sugar Price and coffee prices (0.3) indicates that as the price of sugar increases, there may be a modest upward influence on coffee prices. Similarly, the stronger positive correlation between Oil Price and coffee prices (0.54) suggests that as oil prices rise, coffee prices may experience a more substantial increase. On the other hand, the weak negative correlation between the US Index and coffee prices (-0.13) suggests that changes in the US dollar's value may have a minor downward influence on coffee prices. Based on the data and their correlation values, it appears that Oil Price has a relatively stronger positive correlation with coffee prices compared to Sugar Price and the US Index. So, the project is focused on using Oil Price to predict Coffee Price.

Chapter 4: Models And Evaluation Matrix

4.1. LSTM

4.1.1. LSTM network concept

Long Short-Term Memory (LSTM) is an improvement of RNN that allows learning of long correlations for which the recurrent neural network is constrained. means the influence of the input values at the beginning times on the output values at later times.

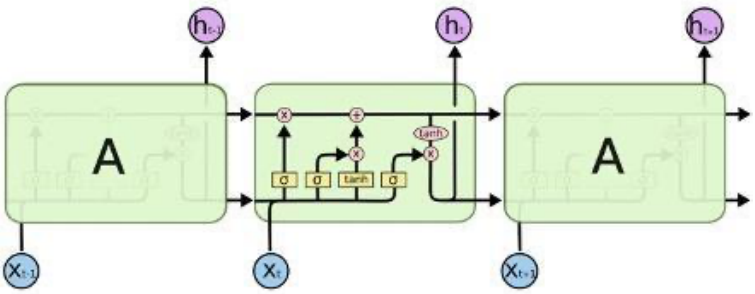


Figure 9: LSTM network.

4.1.2. Internal architecture of an LSTM cell

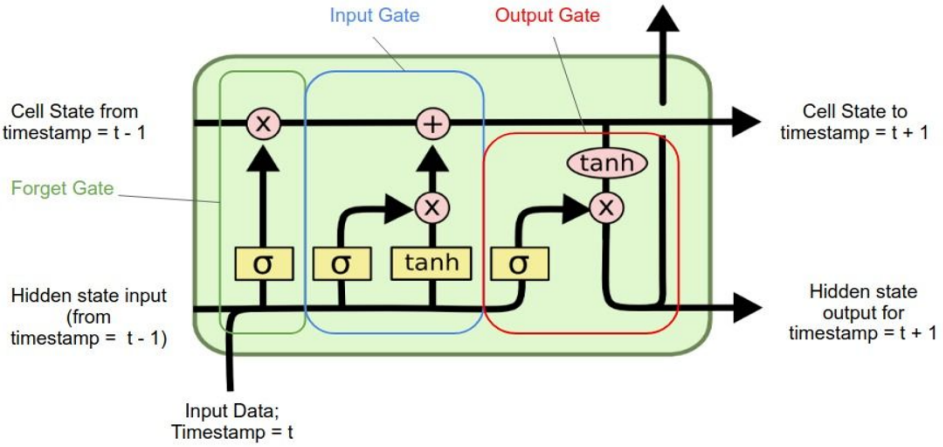
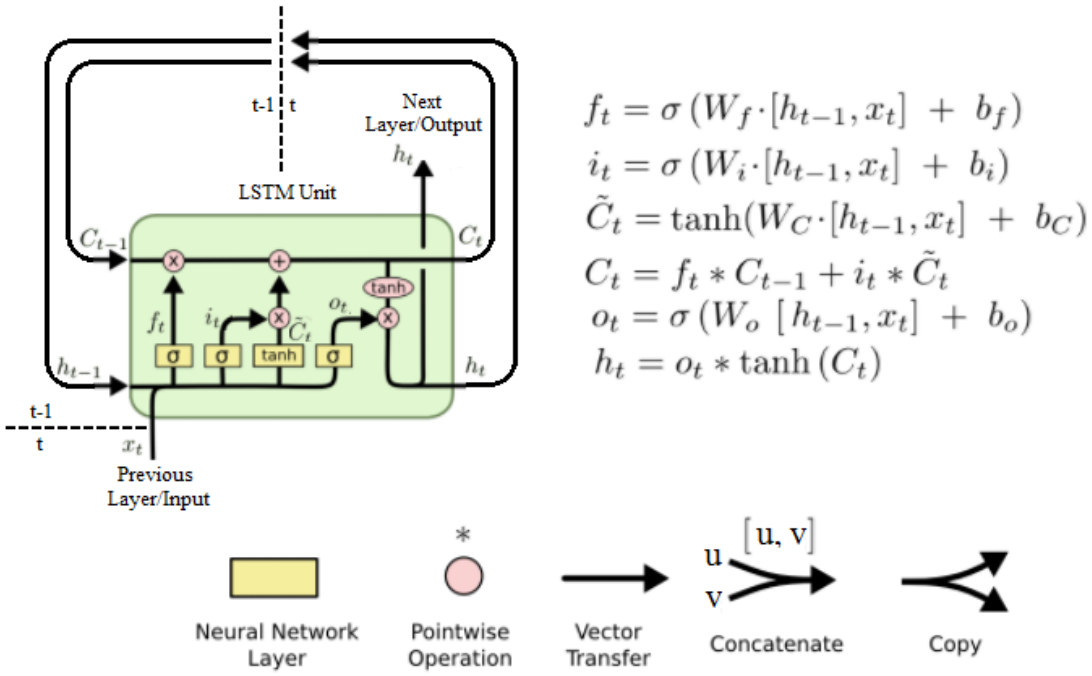


Figure 10: LSTM area.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 11: LSTM Cell

Where:

- Ct is cell's state
- ht is a hidden state
- xt is input of the cell at time t
- Ct-1 and ht-1 is the state of the cell and the hidden state at time t-1
- ft is forget gate
- it is input gate
- ot is output gate

4.1.3. Execution of a cell LSTM

The first step is to determine what input will go to the cell states. The forget gates determine the value of the previous states, how much h(t-1) is passed. The forget gate value of (0,1) is determined by the sigmoid function.

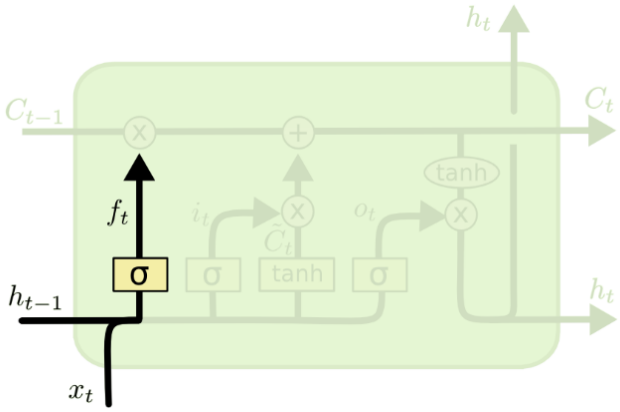


Figure 12: Forget gates

$$ft = \sigma(Wfht\text{-}1,xt+bf)$$

In the second step the input gates will determine the effect of the current input values.

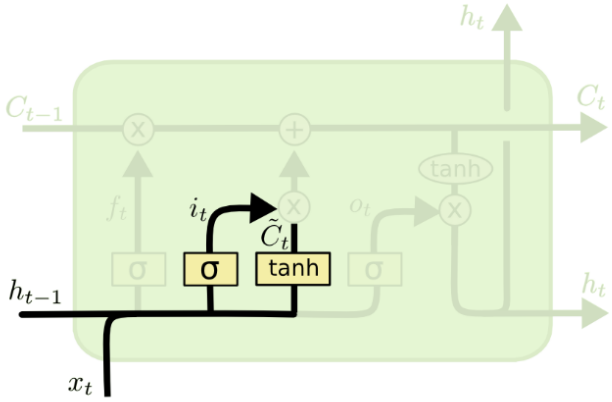


Figure 13: Input gates.

$$Ct = \tanh(WCht\text{-}1,xt+bC) \qquad it = \sigma(Wiht\text{-}1,xt+bi)$$

Next, calculate the value for cell state c_t

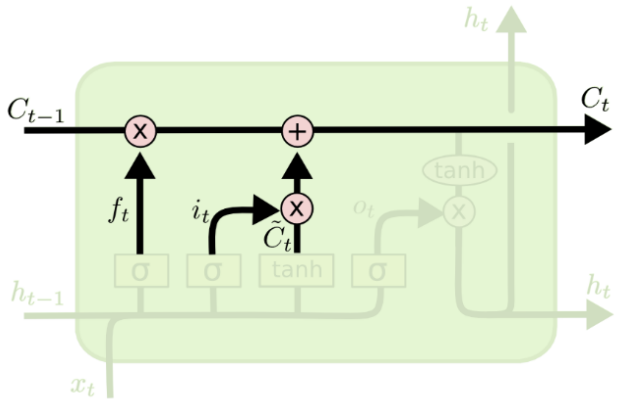


Figure 14: Calculate the value for cell state c_t.

Finally, calculate the value for the emergency layer state and the output gates.

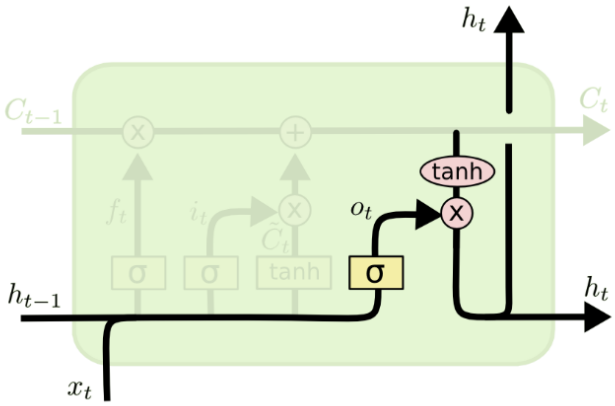


Figure 15: Output gates.

The application of LSTM that we will use to predict the future gold price is to predict the series of numbers. This is the process of predicting the next value for a given series of numbers, this application is often used to forecast the stock market. , weather, Bitcoin price...

Below is an example of a series forecast:

Figure 16: Example of series forecast.

4.2. GRU

GRU, short for Gated Recurrent Unit, is a type of recurrent neural network (RNN) architecture that is widely used in deep learning for sequential data processing tasks. It was introduced as an alternative to the traditional RNN and addresses the vanishing gradient problem, which can occur when training RNNs on long sequences.

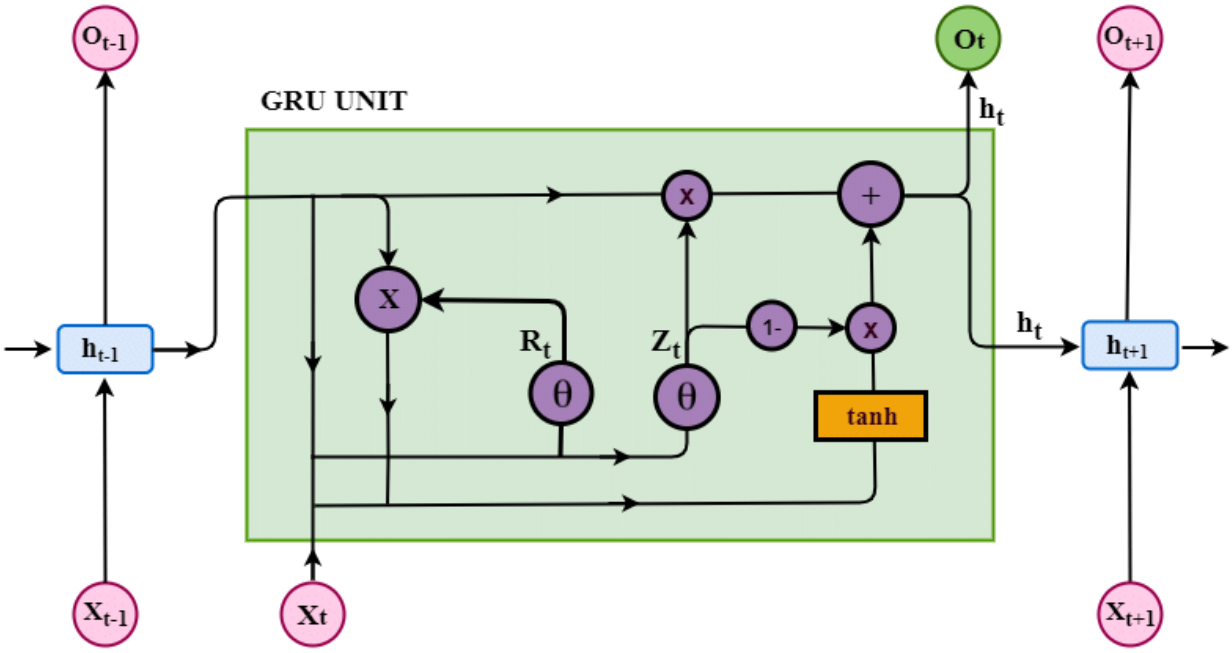


Figure 17: Architecture of basic GRU unit

GRU incorporates gating mechanisms that allow the network to selectively update and reset its internal state. These gates help the GRU model capture and retain relevant information while discarding unnecessary information. The two primary gates in a standard GRU cell are the update gate and the reset gate. The update gate controls how much of the previous hidden state should be passed on to the current state, while the reset gate determines how much of the previous hidden state should

be forgotten or ignored. By adaptively updating and resetting the internal state, GRU cells are capable of modeling long-range dependencies in sequential data.

Compared to traditional RNNs, GRU networks have fewer parameters and are computationally more efficient, making them easier to train and less prone to overfitting. They have shown promising results in various applications, including natural language processing, speech recognition, machine translation, and time series forecasting.

4.3. MLP

MLP models, or Multi-Layer Perceptron models, are a popular choice in machine learning. They consist of multiple layers of interconnected artificial neurons, enabling them to capture complex patterns in data. MLPs are versatile and can handle various types of inputs, making them suitable for a wide range of tasks.

MLP models are trained using optimization algorithms like backpropagation. These algorithms adjust the weights and biases of the connections between neurons to minimize the difference between predicted and actual values. MLPs can approximate any continuous function and are known as universal approximators.

MLP models have found success in various domains, including image recognition, natural language processing, and time series forecasting. However, they can be prone to overfitting, so careful regularization techniques and validation strategies should be employed to ensure their generalization to unseen data. MLPs offer a powerful tool for capturing and learning complex relationships in data, making them an asset in the field of machine learning.

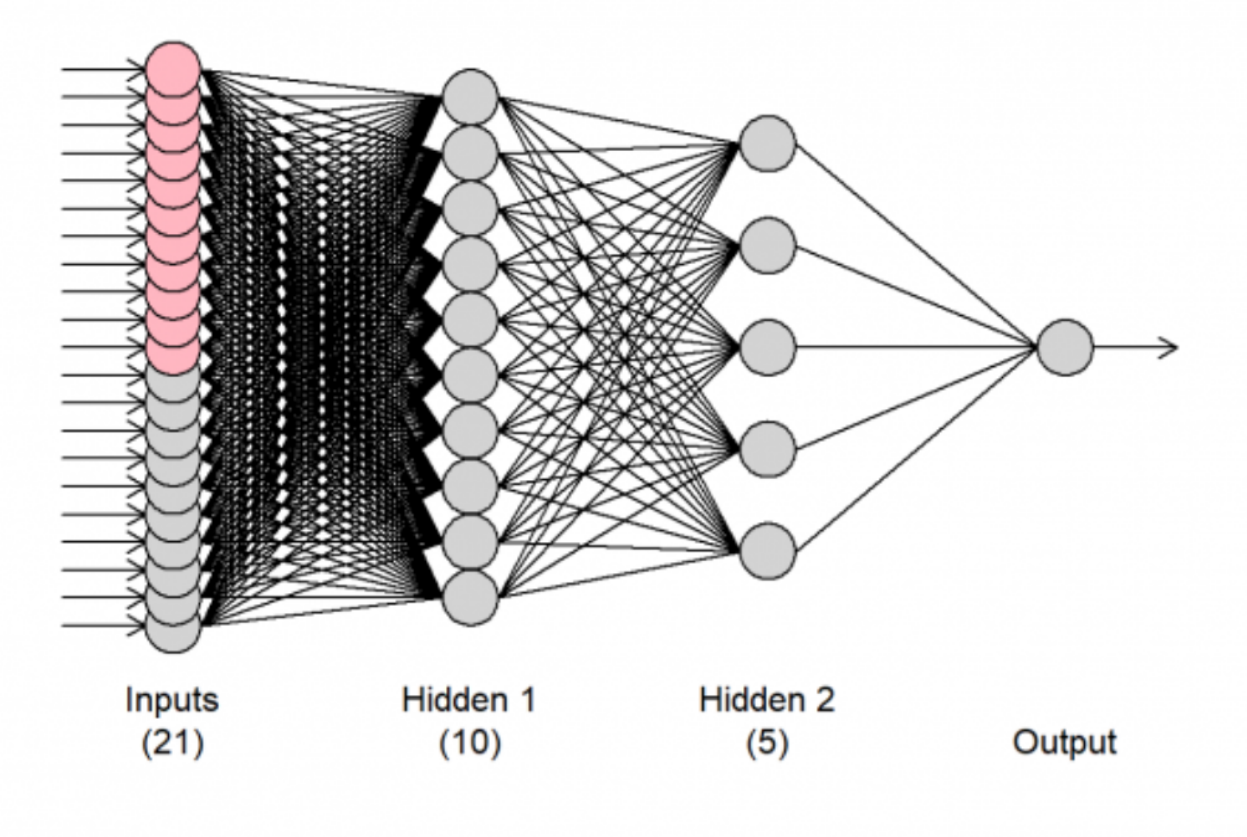


Figure 18: Architecture of MLP

Initialization: At the beginning, the weights, and biases of the MLP model's neurons are initialized randomly or with predefined values.

Forward Propagation: During this step, the input data is fed into the MLP model's input layer. Each neuron in the subsequent hidden layers receives weighted inputs from the neurons in the previous layer, applies an activation function to the weighted sum, and passes the output to the neurons in the next layer. This process is repeated layer by layer until the output layer is reached, generating the model's prediction.

Calculation of Loss: After obtaining the predicted output, the model compares it with the actual target values and calculates a loss or error metric, such as mean squared error or cross-entropy loss. This loss metric quantifies the discrepancy between the predicted and actual values.

Backpropagation and Weight Update: In the backpropagation step, the gradient of the loss with respect to the model's weights and biases is computed using the chain rule of calculus. This gradient information is then used to update the weights and biases in a way that reduces the loss.

Prediction: Once the model is trained, it can be used to make predictions on new, unseen data. The input is fed through the trained model, and the output layer produces the predicted values.

Like any investment, MLPs have their pros and cons. MLPs may not work for all investors. An investor must weigh the disadvantages against the benefits of holding units of MLPs before they invest.

Advantages

- MLPs are known for offering slow yet steady investment returns. The slow returns stem from the fact that MLPs are often in slow-growing industries, like pipeline construction. This slow and steady growth means MLPs are low risk. They earn a stable income often based on long-term service contracts. MLPs offer steady cash flows and consistent cash distributions.
- The cash distributions of MLPs usually grow slightly faster than inflation. For limited partners, 80% to 90% of the distributions are often tax-deferred. Overall, this lets MLPs offer attractive income yields—often substantially higher than the average dividend yield of equities. Also, with the flow-through entity status (and the absence of double taxation), more capital is available for future projects. The availability of capital keeps the MLP competitive in its industry.
- Further, for the limited partner, cumulative cash distributions could exceed the portion taxed at the capital gains rate once units are sold.
- Limited partners' liability for an MLP's debts and obligations is limited to the amount of their capital contribution.

- Until the 2017 tax cuts act benefit expires in 2025, investors can deduct 20% of their distributions from their taxable income, reducing the tax they would otherwise pay.
- There are benefits to using MLPs for estate planning, as well. When unitholders gift or transfer the MLP units to beneficiaries, both unitholders and beneficiaries avoid paying taxes during the time of transfer. The cost basis will readjust based on the market price during the time of the transfer if the transfer happens because of death. There is no step-up in basis if the MLP's units are gifted. Should the unitholder die and the investment pass to heirs, the units pass tax free and fair market value is determined to be the value as of the date of death.

Disadvantages

- MLPs are extremely tax-efficient for investors. However, the filing requirements for this business structure are complex. An MLP’s income, deductions, credits, and other items are detailed each year on an Internal Revenue Service (IRS) Schedule K-1 form that is sent to the investor.
- The K-1 can be complicated and create extra work for investors (or the tax professionals they hire).
- MLP investors are required to pay state income taxes on their allocated portion of income in each state in which the MLP operates (which can be more than one). This can increase their costs.
- Another tax-related disadvantage of MLPs is that you cannot use a net loss (more losses than profits) to offset other income.
- However, net losses may carry forward to the following year.
- When you eventually sell all of your units, a net loss can then be used as a deduction against other income.
- MLPs have limited upside potential. However, this might be expected from an investment that should produce a gradual yet reliable income stream over several years.

4.4. Evaluation Matrix

We utilize the Root Mean Square Error to assess the system's effectiveness (RMSE). The RMSE value is used to reduce error or the discrepancy between the desired and obtained output values. The root mean square error (RMSE) is the sum of all square errors.

$$RMSE = \sqrt{\frac{1}{n}\sum(\hat{Y}_i - Y_i)^2}$$

Equation 2: Formula of RMSE

The RMSE error metric is widely used and is a great all-purpose error metric for numerical forecasts. RMSE amplifies and harshly penalizes big errors in comparison to the analogous Mean Absolute Error. The mean absolute percentage error, or MAPE, is expressed as a percentage. The accuracy of a forecasting system is statistically measured by the mean absolute percentage error (MAPE).

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Equation 3: Formula of MAPE

Chapter 5: Implementation

5.1. LSTM

Step 1: Import Library.

```python
#Prepare and process data
import math
import pandas as pd #read data file csv
import matplotlib.pyplot as plt #draw chart
import matplotlib.ticker as ticker #Format
import numpy as np #data processing
from keras.callbacks import ModelCheckpoint #keep good training
from tensorflow.keras.models import load_model #download model
from sklearn.preprocessing import MinMaxScaler #normalize data 0->1
#Training, model building
from keras.models import Sequential #input
from keras.layers import Dropout #avoid overfitting
from keras.layers import LSTM #dependent learning
from keras.layers import Dense #output
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras import regularizers

#Phân tích chỉ số
from sklearn.metrics import r2_score #measure suitability
from sklearn.metrics import mean_squared_error #measure MSE and RMSE = sqrt(MSE)
from sklearn. metrics import mean_absolute_error #measure mean absolute error
from sklearn.metrics import mean_absolute_percentage_error #measure the mean percent absolute error
```

Figure 19: Import Library

Step 2: Read Data.

```python
df=pd.read_csv('/content/drive/MyDrive/Coffee Forcasting/data.csv')
df
```

	Date	price_coffee	price_oil
0	2008-01-14	2051.0	94.20
1	2008-01-15	2036.0	91.90
2	2008-01-16	2040.0	90.84
3	2008-01-17	2037.0	90.13
4	2008-01-18	2027.0	90.57
...	...	...	...
3861	2023-06-30	2697.0	70.64
3862	2023-07-03	2750.0	69.79
3863	2023-07-04	2744.0	71.06
3864	2023-07-05	2713.0	71.79
3865	2023-07-06	2510.0	71.80

Figure 20: The dataset

Checking the data we see that there are 3 columns totaling 3866 rows each, and no rows are left blank, with the column format price_coffee and price_oil as "float" and date as '"object" for the malformed data we need. So, we're going to reformat the date for the date column

```
[4] df.info() # kiểm tra dữ liệu đã được định dạng

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3866 entries, 0 to 3865
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          3866 non-null   object
 1   price_coffee  3866 non-null   float64
 2   price_oil     3866 non-null   float64
dtypes: float64(2), object(1)
memory usage: 90.7+ KB
```

```
#Định dạng lại cấu trúc của cột Date (định dạng năm-tháng-ngày)
df["Date"]=pd.to_datetime(df.Date,format="%Y/%m/%d")
```

Figure 21: Structure of data

Make the date column the DateFrame df's index. Each row in the DataFrame has a label called an index, which is used to specify the row's location.

```
df=pd.DataFrame(df,columns=['Date','price_coffee','price_oil'])
df.index=df.Date
df.drop("Date",axis=1, inplace=True)
```

Figure 22: Set date into index

Data visualization of the coffee and oil prices from January 2008 to July 2023

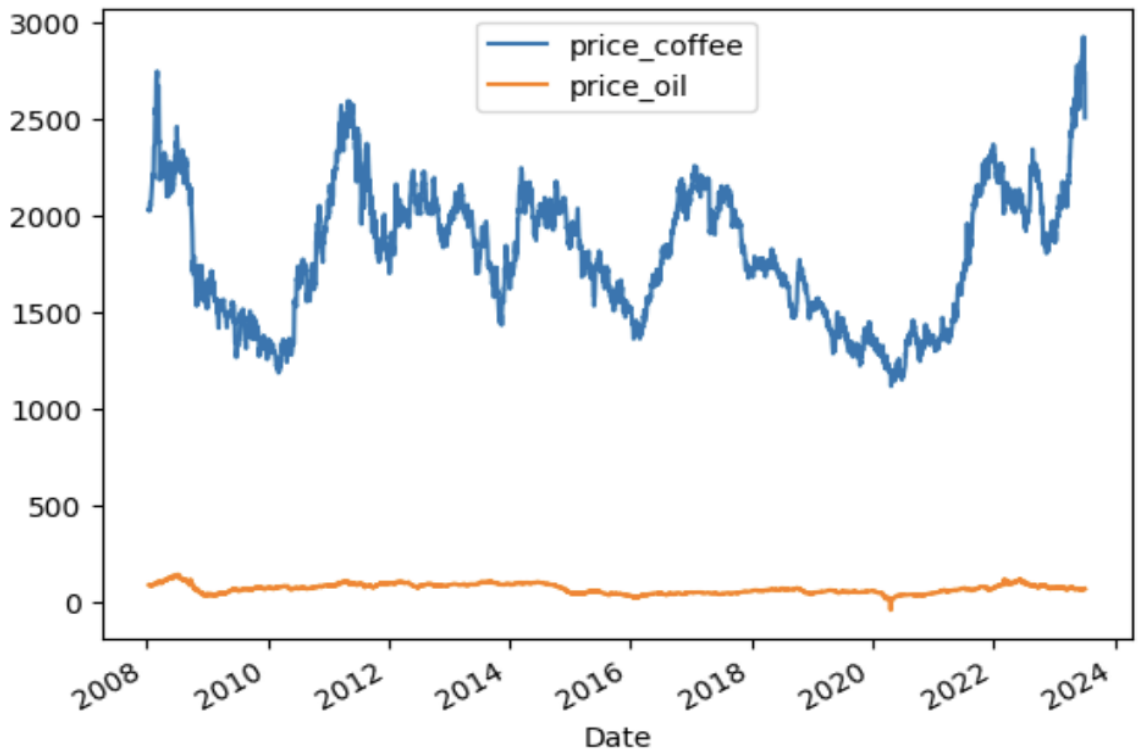


Figure 23: Data visualization

Step 3: Split Dataset.

We proceed to divide the dataset to conduct model training, here I will divide the data into 3 rates, the first rate is 70% train and 30% test, the second type is 80% train

20% test, the third type is 90% train and 10% test. For convenience in division, use the math.ceil() function to divide the data with the result rounded.

```
[12] #8-2
     data=df.values
     training_data_len = math.ceil(len(data)* 0.8)
     train_data = data[0: training_data_len, :]
     train_data.shape
     test_data=data[training_data_len:]
     test_data.shape

     (773, 2)

[13] #7-3
     data=df.values
     training_data_len_73 = math.ceil(len(data)* 0.7)
     train_data73 = data[0: training_data_len_73, :]
     train_data73.shape
     test_data73=data[training_data_len_73:]
     test_data73.shape

     (1159, 2)

[14] #9-1
     data=df.values
     training_data_len_91 = math.ceil(len(data)* 0.9)
     train_data91 = data[0: training_data_len, :]
     train_data91.shape
     test_data91=data[training_data_len_91:]
     test_data91.shape

     (386, 2)
```

Figure 24: Train test split into 7-3, 8 – 2 and 9 - 1

Step 4: Data normalization.

The next step after we split the dataset is to normalize the data to 0 to 1 to conduct model training.

```
[16] #Tiến hành chuẩn hoá lại dữ liệu với các giá trị được đưa về từ 0-1
     sc=MinMaxScaler(feature_range=(0,1))
     #Chuẩn hoá dữ liệu có tên data
     sc_train=sc.fit_transform(data)

[17] sc_train

     array([[0.51409619, 0.72069757],
            [0.50580431, 0.70812377],
            [0.50801548, 0.70232889],
            ...,
            [0.89718076, 0.59419418],
            [0.88004422, 0.598185  ],
            [0.76782753, 0.59823967]])
```

Figure 25: Data Normalization

Step 5: Training set data processing.

The model will use 14 consecutive days to predict the next days, in this step we proceed to create a loop for the values.

```
[20] # 8-2
    for i in range(n_past, len(train_data) - n_future +1):
        trainX.append(sc_train[i - n_past:i, 0:train_data.shape[1]])
        trainY.append(sc_train[i + n_future - 1:i + n_future, 0])

    trainX, trainY = np.array(trainX), np.array(trainY)

    print('trainX shape == {}.'.format(trainX.shape))
    print('trainY shape == {}.'.format(trainY.shape))

    trainX shape == (3079, 14, 2).
    trainY shape == (3079, 1).
```

```
[21] # 7-3
    for i in range(n_past, len(train_data73) - n_future +1):
        trainX73.append(sc_train[i - n_past:i, 0:train_data73.shape[1]])
        trainY73.append(sc_train[i + n_future - 1:i + n_future, 0])

    trainX73, trainY73 = np.array(trainX73), np.array(trainY73)

    print('trainX shape == {}.'.format(trainX73.shape))
    print('trainY shape == {}.'.format(trainY73.shape))

    trainX shape == (2693, 14, 2).
    trainY shape == (2693, 1).
```

```
[87] # 9-1
    for i in range(n_past, len(train_data91) - n_future +1):
        trainX91.append(sc_train[i - n_past:i, 0:train_data91.shape[1]])
        trainY91.append(sc_train[i + n_future - 1:i + n_future, 0])

    trainX91, trainY91 = np.array(trainX91), np.array(trainY91)

    print('trainX shape == {}.'.format(trainX91.shape))
    print('trainY shape == {}.'.format(trainY91.shape))

    trainX shape == (3466, 14, 2).
    trainY shape == (3466, 1).
```

Figure 26: Processing the data

Step 6: Build Model

In this step we proceed to build a model of 9 layers, the first layer is serial() to create the network layer for input data, the next is the lstm layer whose unit is 64 and in this layer we have to describe the input information. . input and return_sequence = True to notify the next lstm layer. Next, we come to the Dropout class, which is used to assume that part of the units is hidden during training, thus reducing the mixing product. Next is the lstm class with a unit number of 32 and after that is the dropout class with an index of 0.5. The last lstm layer has a unit number of 16 followed by 2 Dense composite layers with the last 25 Dense 1 is an output layer with 1-way output to predict a value.

With the. compile function, we have loss index = 'mean_absolute_error' to measure the average error index and optimizer = 'adam' the optimizer helps the learning process of the model.

```
# 8-2
from keras import regularizers

model = Sequential()
model.add(LSTM(units=64, activation='relu', input_shape=(trainX.shape[1], trainX.shape[2]), return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(units=32, activation='relu', return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(units=16, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(25, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.2))
model.add(Dense(trainY.shape[1]))

model.compile(loss='mean_absolute_error', optimizer='adam')
model.summary()
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_6 (LSTM)                (None, 14, 64)            17152

 dropout_8 (Dropout)          (None, 14, 64)            0

 lstm_7 (LSTM)                (None, 14, 32)            12416

 dropout_9 (Dropout)          (None, 14, 32)            0

 lstm_8 (LSTM)                (None, 16)                3136

 dropout_10 (Dropout)         (None, 16)                0

 dense_4 (Dense)              (None, 25)                425

 dropout_11 (Dropout)         (None, 25)                0

 dense_5 (Dense)              (None, 1)                 26

=================================================================
Total params: 33,155
Trainable params: 33,155
Non-trainable params: 0
_________________________________________________________________
```

Figure 27: Build the Model

Step 7: Model Training

In the 7th step, we train the model with the epochs index of 50. The model after training will be saved in the save_model.h5 file, above all, only the best model is saved when training.

```
#8-2
#huấn luyện mô hình mô hình
save_model="save_model.h5" #lưu lại mô hình với dạng file .h5
best_model=ModelCheckpoint(save_model,monitor='loss',verbose=2,save_best_only=True, mode='auto')
model.fit(trainX, trainY, epochs=50,verbose=2,callbacks=best_model) #lưu dữ liệu phần epochs
```

Figure 28: Train the Model

After training we are going to check loss value, see if it fits the model and if enough epochs the graph will be stable after runtime:

```
loss_per_epoch91 = model91.history.history['loss']
plt.plot(loss_per_epoch91)
```

```
[<matplotlib.lines.Line2D at 0x7c2e18157eb0>]
```

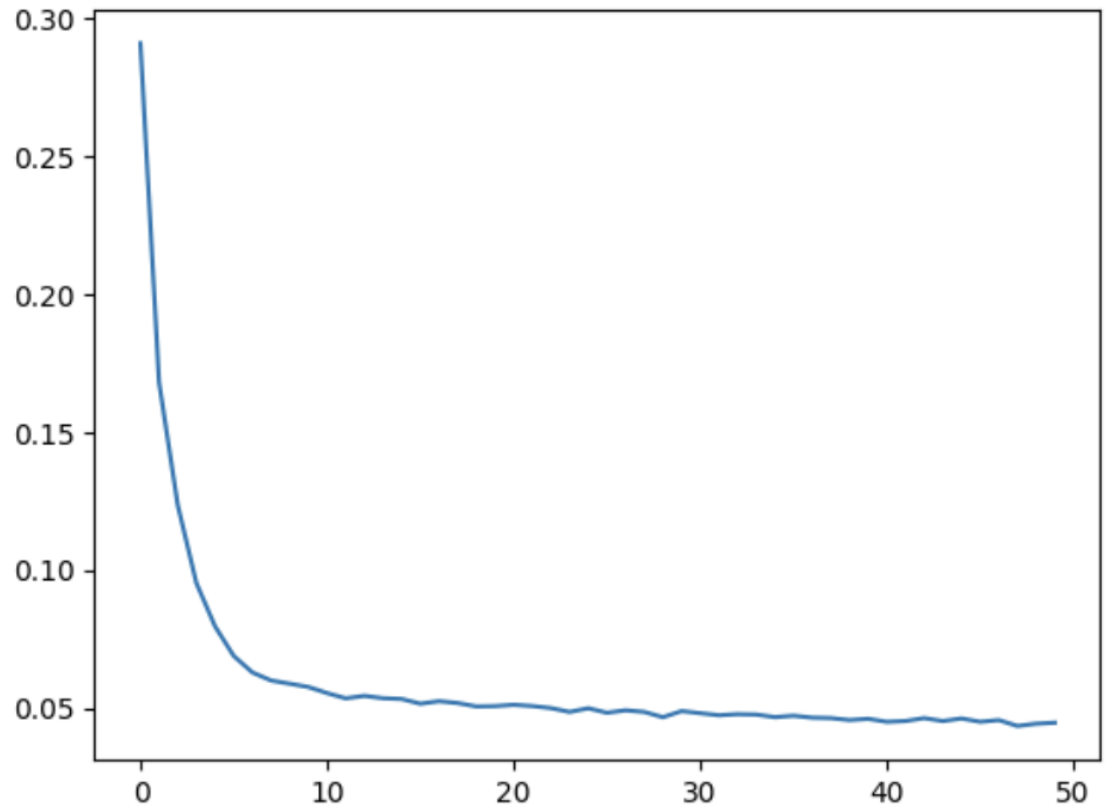


Figure 29: Loss per Epoch

Step 8: Data processing of test sets and train sets.

Re-upload the saved model and apply it to the x_train prediction and format the data back to the original format.

```
prediction_copies = np.repeat(trainY, train_data.shape[1], axis=-1)
y_train=sc.inverse_transform(prediction_copies) #tiến hành xử lí số chuẩn hoá 0-1
final_model=load_model('save_model.h5') #load lại mô hình tốt nhất đã lưu
y_train_predict=final_model.predict(trainX) #giá dự đoán
y_train_predict = np.repeat(y_train_predict, train_data.shape[1], axis=-1)
y_train_predict=sc.inverse_transform(y_train_predict) #tiến hành xử lí số chuẩn hoá 0-1
```

Figure 30: Process train set and test set

In the test set, we process the data as in the training set.

```
test=df[len(train_data)-14:].values #trừ đi 14 ngày đầu tiên dùng để dự đoán ngày tiếp theo
test=test.reshape(-1,2)
sc_test=sc.transform(test)

x_test=[]
for i in range(n_past, len(test) - n_future +1):
    x_test.append(sc_test[i - n_past:i, 0:train_data.shape[1]])
x_test=np.array(x_test)

y_test=data[training_data_len:]
y_test_predict=final_model.predict(x_test)
y_test_predict = np.repeat(y_test_predict, train_data.shape[1], axis=-1)
y_test_predict=sc.inverse_transform(y_test_predict)
```

Step 9: Data histogram between the set of train – test and predict

In the 9th step, we proceed to draw a graph to be able to observe the train - test data with the predicted data.

```
train_data1=df[14:training_data_len]
test_data1=df[training_data_len:]

format = ticker.StrMethodFormatter('{x:,.0f}')
title = 'Coffee Price'
ylabel = 'Price (USD)'
xlabel = 'Date'
plt.figure(figsize=(21,5))
plot = df['price_coffee'].plot(label=f"Prices Data",color='red')
train_data1['prediction']=y_train_predict
plot_pdtrain = train_data1['prediction'].plot(label=f"Prediction train", color='green')
test_data1['prediction']=y_test_predict
plot_pdtest = test_data1['prediction'].plot(label=f"Prediction test", color='blue')

plot.autoscale(axis='x', tight=True)
plot.set(xlabel=xlabel, ylabel=ylabel)
plot.yaxis.set_major_formatter(format)
plt.legend()
plt.grid(True)
plt.show()
```

Figure 31: Histogram of Predicted train and Predicted test results.

Step 10: Evaluation

At the first data rate of 70% train and 30% test, MAPE reuse is at 7.16% and RMSE is 140.50 USD with data after analysis of data set 7-3 is good.

Date	price_coffee	price_oil	prediction
2018-11-23	1611.0	50.42	1631.893799
2018-11-26	1637.0	51.63	1623.210693
2018-11-27	1647.0	51.56	1617.663696
2018-11-28	1630.0	50.29	1616.551147
2018-11-29	1625.0	51.45	1618.275391
...	...	...	...
2023-06-30	2697.0	70.64	2433.645752
2023-07-03	2750.0	69.79	2418.678223
2023-07-04	2744.0	71.06	2399.622314
2023-07-05	2713.0	71.79	2386.112061
2023-07-06	2510.0	71.80	2377.776855

Figure 32: Forecast next 30 days

```
[202] #7-3 train-data
      mape_t73 = mean_absolute_percentage_error(test_data173['price_coffee'],test_data173['prediction'])
      mape_percentage = mape_t82 * 100
      print(f"MAPE: {mape_percentage:.2f}%")
      print('RMSE: ',np.sqrt(mean_squared_error(test_data173['price_coffee'],test_data173['prediction'])))

      MAPE: 7.16%
      RMSE:  140.50251771777454
```

Figure 33: Model Evaluation

At the second data rate of 80% train and 20% test, the MAPE reuse is 6.15% and RMSE is 139.381 USD with the data after analysis of data set 8-2 as good.

Date	price_coffee	price_oil	prediction
2020-06-05	1245.0	39.55	1442.653931
2020-06-08	1263.0	38.19	1443.581177
2020-06-09	1247.0	38.94	1444.913818
2020-06-10	1253.0	39.60	1446.367920
2020-06-11	1227.0	36.34	1447.847656
...	...	...	...
2023-06-30	2697.0	70.64	2433.645752
2023-07-03	2750.0	69.79	2418.678223
2023-07-04	2744.0	71.06	2399.622314
2023-07-05	2713.0	71.79	2386.112061
2023-07-06	2510.0	71.80	2377.776855

Figure 34: Forecast next 30 days

```
[192] # 8-2 test data
      mape_t82 = mean_absolute_percentage_error(test_datal['price_coffee'],test_datal['prediction'])
      mape_percentage = mape_t82 * 100
      print(f"MAPE: {mape_percentage:.2f}%")
      print('RMSE: ',np.sqrt(mean_squared_error(test_datal['price_coffee'],test_datal['prediction'])))

      MAPE: 6.15%
      RMSE:  139.38116070587472
```

Figure 35: Model Evaluation

At the final data rate of 90% train and 10% test, MAPE reuse at 7.16% and RMSE is 199.608 USD with data after analysis of dataset 9-1 as good.

Date	price_coffee	price_oil	prediction
2021-12-13	2287.0	71.29	2095.857422
2021-12-14	2297.0	70.73	2092.911865
2021-12-15	2303.0	70.87	2090.422607
2021-12-16	2299.0	72.38	2090.385498
2021-12-17	2333.0	70.86	2091.281982
...	...	...	...
2023-06-30	2697.0	70.64	2338.383301
2023-07-03	2750.0	69.79	2317.108398
2023-07-04	2744.0	71.06	2298.255615
2023-07-05	2713.0	71.79	2287.446533
2023-07-06	2510.0	71.80	2280.099609

Figure 36: Forecast next 30 days .

```
[203] #9-1 test-data
      mape_t91 = mean_absolute_percentage_error(test_data191['price_coffee'],test_data191['prediction'])
      mape_percentage = mape_t82 * 100
      print(f"MAPE: {mape_percentage:.2f}%")
      print('RMSE: ',np.sqrt(mean_squared_error(test_data191['price_coffee'],test_data191['prediction'])))

      MAPE: 7.16%
      RMSE:  199.60895348360881
```

Figure 37: Model Evaluation

The fact that the ratio between the trains - tests is not different, as well as the MAPE at ~6-7%, we can see that the LSTM model works very well, after many times of training the model, we find that the model runs at the ratio. 80% train 20% test rate is giving the best results so in the next part we will proceed to predict the next 30 days of future gold prices.

Step 11: Prediction Coffee price next 30 days

```
[209] lst_output = []
      n_steps = 14
      i = 0

      while i < 30:
          if len(temp_input) > n_steps:
              x_input=np.array(temp_input[0:])
              x_input = x_input.reshape((1, n_steps, 2))
              yhat = model.predict(x_input, verbose=0)
              temp_input.extend(yhat[0].tolist())
              temp_input = temp_input[1:]
              lst_output.extend(yhat.tolist())

              x_input=np.array(temp_input[0:])
              x_input = x_input.reshape((1, n_steps, 2))
              yhat = model.predict(x_input, verbose=0)
              temp_input.extend(yhat[0].tolist())
              temp_input = temp_input[1:]
              i += 1
          else:
              x_input = np.array(temp_input).reshape((1, n_steps, 2))
              yhat = model.predict(x_input, verbose=0)

              temp_input.extend(yhat[0].tolist())
              lst_output.extend(yhat.tolist())
              i += 1
```

[212]
```
y_train_predict_inv = y_train_predict_inv[:, :-1]
y_train_predict_inv
```

```
array([[2387.8414945 ],
       [2350.17099881],
       [2302.04327291],
       [2250.92242569],
       [2204.11008865],
       [2165.97429758],
       [2133.94528323],
       [2108.09750557],
       [2087.36013913],
       [2068.28165185],
       [2052.36427099],
       [2036.89703929],
       [2022.93913132],
       [2011.33958137],
       [2003.06647497],
       [1995.20215699],
       [1988.22976989],
       [1983.06777686],
       [1978.10215479],
       [1973.91693252],
       [1970.41805536],
       [1967.54472667],
       [1965.26993752],
       [1963.41193354],
       [1961.8696695 ],
       [1960.6116879 ],
       [1959.60854369],
       [1958.90784678],
       [1958.44774136],
       [1958.22049546]])
```

Figure 38: Visualization of the next 30 days

Thus, it is predicted that in the next 30 days the Coffee price will decrease.

5.2. GRU

Step 1: Import the dataset and normalize data.

```
# Prepare the data for training
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[['price_coffee', 'price_oil']])
```

Figure 39: Normalize the dataset

The 'Date' column in the dataset is parsed as dates, and the dataframe is sorted by date. The 'Date' column is then set as the index for easier manipulation. The MinMaxScaler from scikit-learn is used to normalize the input data. The 'price_coffee' and 'price_oil' columns are selected from the dataframe, and the scaler is fitted on this subset of data. The scaler applies a transformation that scales the values between 0 and 1, ensuring that all features are on a similar scale.

Step 2: Divide the dataset into input sequences.

```
# Split the data into input sequences and corresponding target values
X = []
y = []
for i in range(n_steps, len(scaled_data)):
    X.append(scaled_data[i - n_steps:i])
    y.append(scaled_data[i, 0])

# Convert the lists to numpy arrays
X = np.array(X)
y = np.array(y)
```

Figure 40: Input Sequence of feature and target value

The step of dividing the dataset into input sequences and target value is used for training the sequential model like GRU. The GRU (Gated Recurrent Unit) is a type of recurrent neural network (RNN) that processes sequential data. It takes a sequence of input data and learns to make predictions based on the patterns and dependencies within the sequence. In this step we use 14 steps for prediction (use previous 14 days to predict the next day). It includes 2 arrays X is input sequence and Y is target value.

Step 3: Split the data into training and testing set.

```
# Split the data into training and testing sets
train_size = int(0.7 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
# Split the data into training and testing sets
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
# Split the data into training and testing sets
train_size = int(0.9 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Figure 41: Train test split into 3 cases

The input sequence is split into 3 cases:

- 70% of the training set and 30% of the testing set.
- 80% of the training set and 20% of the testing set.
- 90% of the training set and 10% of the testing set.

The data is split into training and testing sets to evaluate the performance of the model. The common practice is to use a larger portion of the data for training and a smaller portion for testing. In this code, the data is split using a 70:30 ratio. The first 70% of the data is used for training, and the remaining 30% is used for testing. The train-test split is performed on both the input sequences (X) and the target values (y), resulting in four variables: X_train, X_test, y_train, and y_test.

Step 4: Initialize the model.

```
model = Sequential()
model.add(GRU(50, activation='relu', input_shape=(n_steps, 2)))
model.add(layers.Dropout(rate=0.3))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

Figure 42: Initialize the model.

The GRU model is initialized using the Sequential API from Keras. The Sequential API allows you to build a model by stacking layers one after the other. In this code, the model consists of a single GRU layer with 50 units (neurons) and a ReLU activation function. The input shape is specified as (n_steps, 2), indicating the number of time steps and the number of features (price_coffee and price_oil).

A dense layer with a single output unit is added to the model. The output unit will predict the next 'price_coffee' value. The model is compiled with the Adam optimizer, which is an efficient optimization algorithm commonly used for training deep learning models.

The mean squared error (MSE) loss function is used to measure the difference between the predicted and actual values during training.

Step 5: Train the model

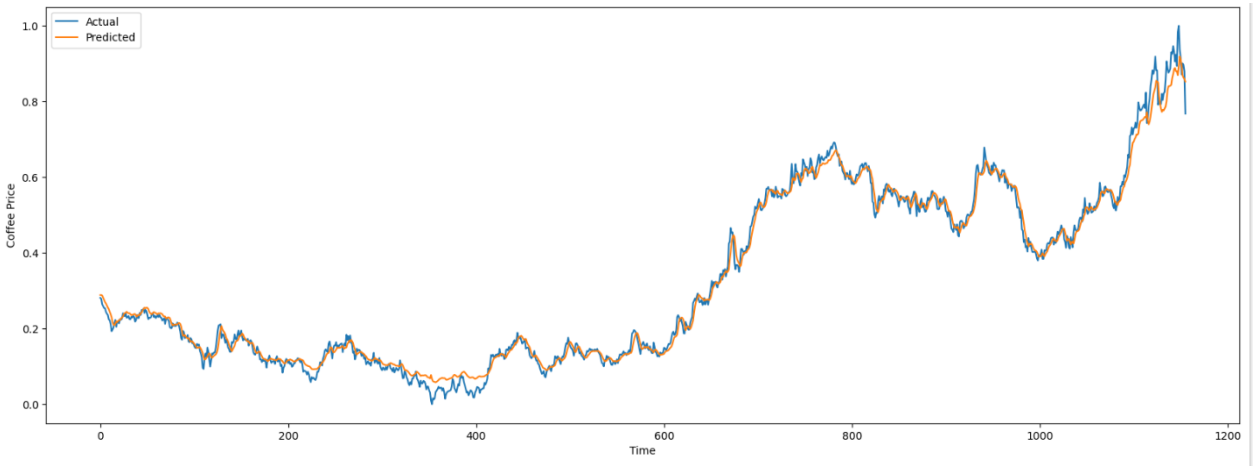
```
# Train the model
model.fit(X_train, y_train, epochs=200, batch_size=32, verbose=1)
```

Figure 43: Train the Model

The model is trained using the fit() method. The training data (X_train and y_train) are provided as inputs to the fit() function. The model is trained for 10 epochs, which means it goes through the entire training dataset 10 times. The batch size is set to 32, indicating that the model will update the weights after processing 32 samples. The verbose parameter is set to 1, which displays the training progress and the loss value at each epoch.

Step 6: Predict and Evaluate on the test set.

We use the trained GRU model to make predictions on new data and evaluate its performance.



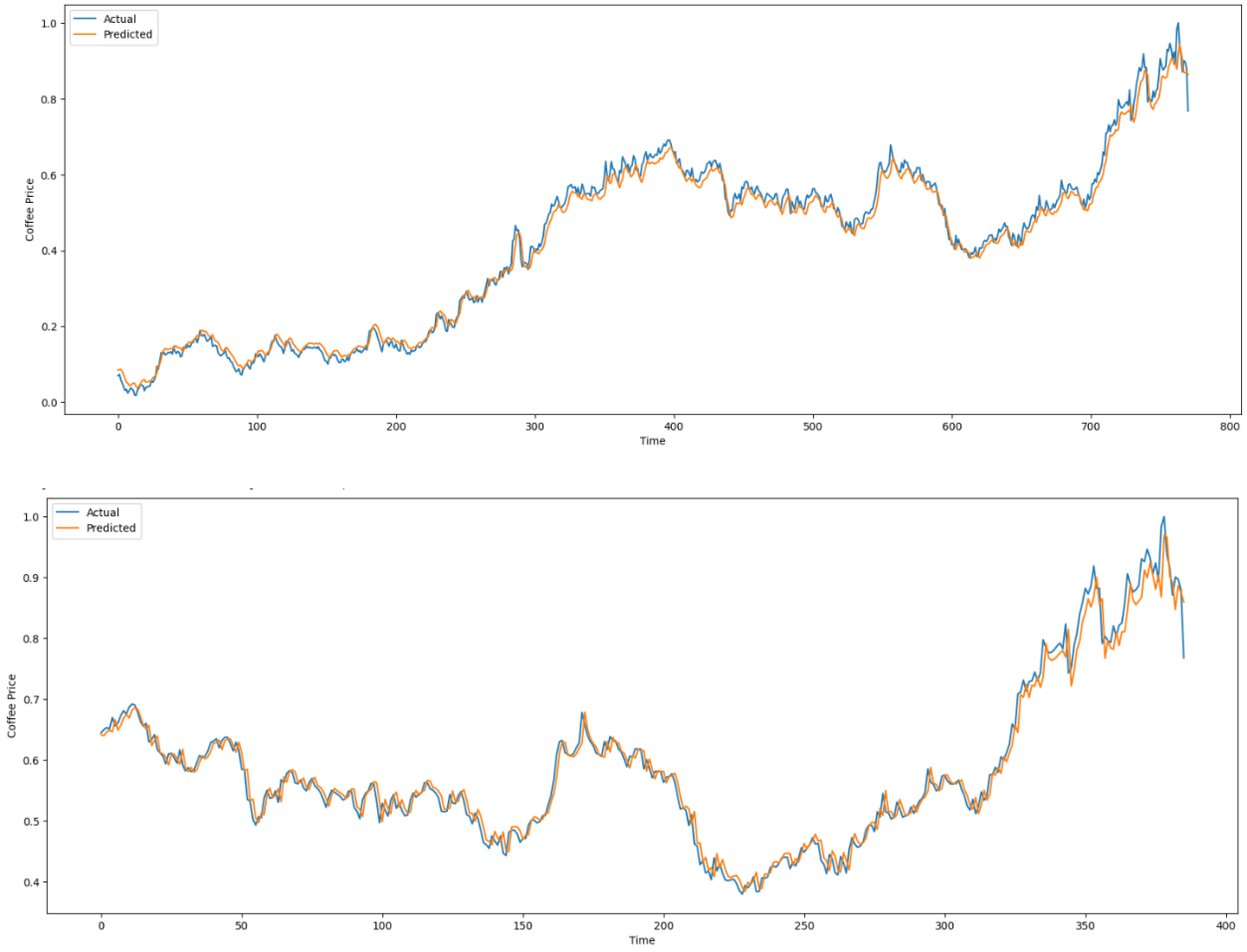


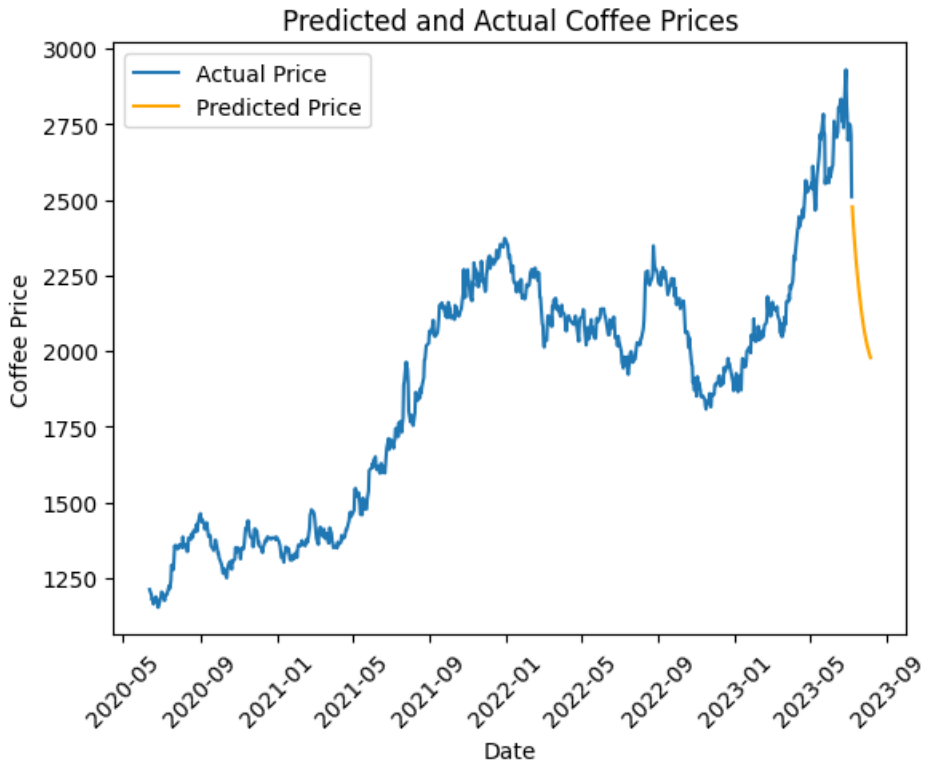
Figure 44 Predict on Test Set

GRU	7 - 3	140.02	6.57
	8 – 2	121.32	6.23
	9 - 1	94.73	4.51

Table2 : Train test Evaluaion

After training the model, it can be used to make predictions on new, unseen data. In this code, the trained model is used to predict the 'price_coffee' values for the test set (X_test). The predictions can be compared with the actual values (y_test) to evaluate the model's performance.

Step 7: Forecast the next 30 days



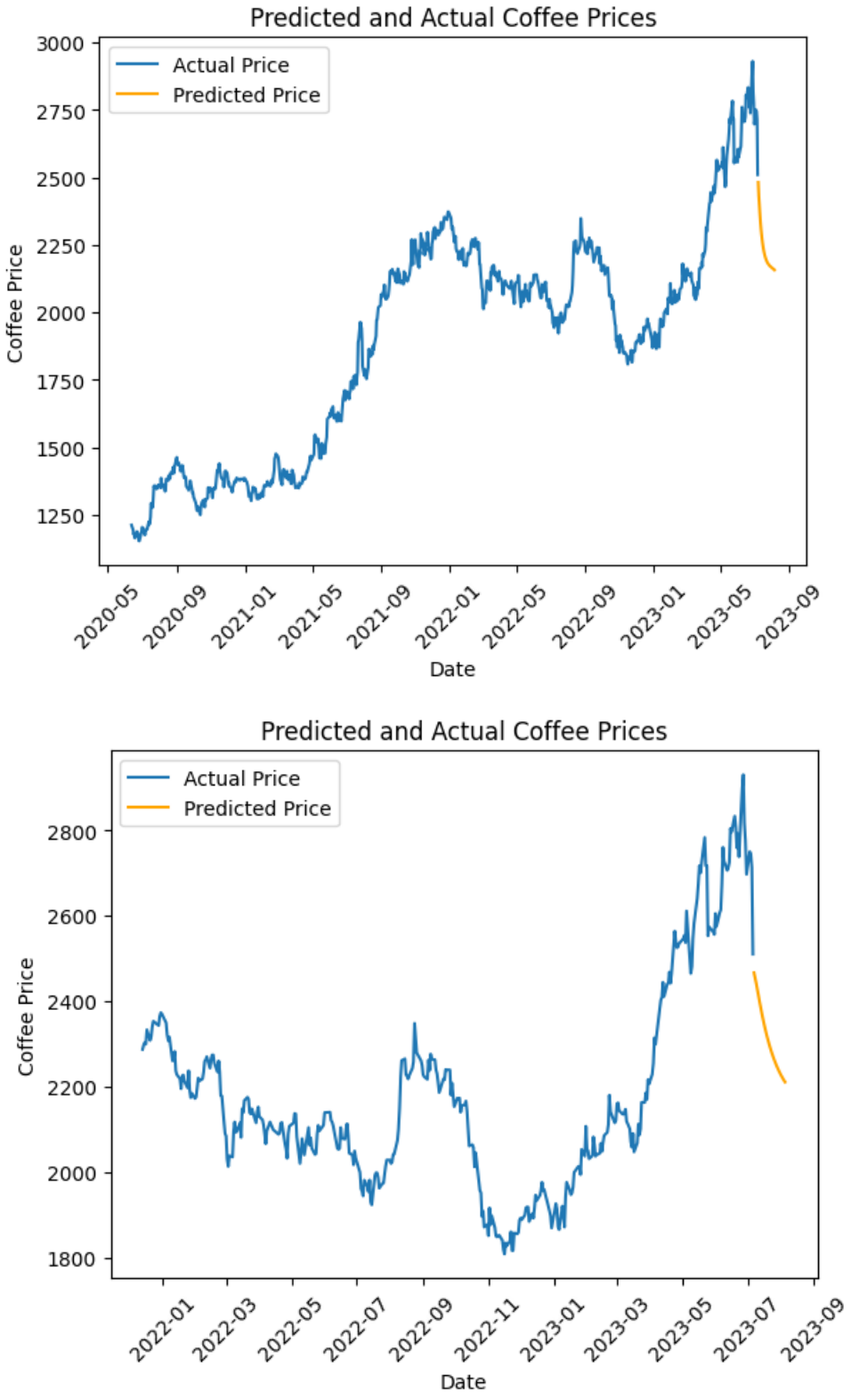


Figure 45: Visualization of the next 30 days

5.3. MLP

Step 1: Importing Libraries, Loading and Preprocessing the Data

```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Load the dataset from CSV
df = pd.read_csv('/content/data_final.csv', parse_dates=['Date'], dayfirst=True)

# Sort the dataframe by date
df = df.sort_values('Date')

# Set 'Date' as the index
df.set_index('Date', inplace=True)
```

Figure 46: Import Libraries and Loading the data

In this step, the necessary libraries (pandas, numpy, MLPRegressor, MinMaxScaler, and matplotlib.pyplot) are imported to support subsequent data processing and visualization tasks.

The dataset is loaded from a CSV file using pd.read_csv, . The parse_dates parameter is used to indicate that the 'Date' column should be parsed as dates. The dayfirst parameter specifies that the dates in the 'Date' column are in day-month-year format.

Step 2: Extracting input features and target variable

```
# Extract input features (coffee price and oil price) and target variable (coffee price)
X = df[['price_oil', 'price_coffee']].values
y = df['price_coffee'].values
```

Figure 47: Extract input Features

The input features are extracted from the DataFrame df by selecting the 'price_oil' and 'price_coffee' columns using df[['price_oil', 'price_coffee']]. These features are stored in the NumPy array X. The target variable is extracted from the DataFrame df

by selecting the 'price_coffee' column using df['price_coffee']. The target variable is stored in the NumPy array y.

Step 3: Train Test Split

```
# Split the data into training and testing sets
train_size = int(0.7 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
# Split the data into training and testing sets
train_size = int(0.8 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
# Split the data into training and testing sets
train_size = int(0.9 * len(X))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Figure 48: Train test split into 3 cases

The input sequence is split into 3 cases:

70% of the training set and 30% of the testing set.

80% of the training set and 20% of the testing set.

90% of the training set and 10% of the testing set.

The data is split into training and testing sets to evaluate the performance of the model. The common practice is to use a larger portion of the data for training and a smaller portion for testing. In this code, the data is split using a 70:30 ratio. The first 70% of the data is used for training, and the remaining 30% is used for testing. The train-test split is performed on both the input sequences (X) and the target values (y), resulting in four variables: X_train, X_test, y_train, and y_test.

Step 4: Normalize the input and create input sequence.

```
# Prepare the data for training
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(dataset[['price_coffee']])

# Define the number of time steps to use for prediction
n_steps = 14

# Split the data into input sequences and corresponding target values
X = []
y = []
for i in range(n_steps, len(scaled_data)):
    X.append(scaled_data[i - n_steps:i])
    y.append(scaled_data[i, 0])

# Convert the lists to numpy arrays
X = np.array(X)
y = np.array(y)
```

Figure 49: Normalize the Input

The code utilizes the MinMaxScaler from scikit-learn to normalize the input features. Normalization is an important step in machine learning to ensure that all features are on a similar scale, which can help improve the performance and stability of the model. The MinMaxScaler scales the input features by mapping the minimum and maximum values of each feature to a specified range, often between 0 and 1. By applying fit_transform() to the training data, the scaler computes the scaling parameters based on the training set and applies the transformation. Then, the transform() method is used on the testing data, applying the same scaling parameters derived from the training data. This ensures that the testing data is scaled consistently with the training data.

Defining the number of time steps: n_steps = 14 specifies the number of previous time steps to consider as input for the model. In this case, the model will use the past 14 time steps to predict the next time step.

Splitting the data into input sequences and target values: This part of the code creates input sequences (X) and corresponding target values (y) for the model. It loops over

the scaled data (scaled_data) starting from the n_steps index. For each index i, it takes the previous n_steps values as the input sequence (X) and the value at index i as the target value (y). This way, X and y are formed as sliding windows of data where each input sequence of length n_steps corresponds to the target value at the next time step.

Converting to numpy arrays: Finally, the lists X and y are converted to numpy arrays using np.array(). This is necessary as the input to most machine learning models, including the MLP, should be in the form of numpy arrays.Step 5: Initialize and train the model.

```
# Create and train the MLP model
model = Sequential()
model.add(Dense(50, activation='relu', input_shape=(n_steps * 1,)))
model.add(Dropout(rate=0.3))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train_seq, y_train_seq, epochs=20, batch_size=16, verbose=1)
```

Figure 50: Initialize and train model

Creating the model: model = Sequential() initializes an instance of the Sequential class, which allows us to create a sequential stack of layers for the model.

Adding layers: The code adds layers to the model using model.add(). The layers added in the code snippet are as follows:

Dense(50, activation='relu', input_shape=(n_steps * 1,)): This line adds a fully connected (dense) layer with 50 units and ReLU activation function. The input_shape parameter is set to (n_steps * 1,), which specifies the shape of the input data. In this case, the input shape is (n_steps, 1) because we are using a univariate time series.

Dropout(rate=0.3): This line adds a dropout layer with a dropout rate of 0.3. Dropout is a regularization technique that randomly sets a fraction of input units to 0 during training, which helps prevent overfitting.

Dense(1): This line adds the output layer with a single unit. Since we are performing regression for time series forecasting, the output layer has only one unit without an activation function.

Compiling the model: model.compile(optimizer='adam', loss='mean_squared_error') compiles the model by specifying the optimizer and loss function to be used during training. In this case, we use the Adam optimizer and mean squared error (MSE) as the loss function.

Training the model: model.fit(X_train_seq, y_train_seq, epochs=20, batch_size=16, verbose=1) trains the model using the training data. The X_train_seq and y_train_seq are the input sequences and target values, respectively. The epochs parameter determines the number of times the model will iterate over the entire training dataset. The batch_size parameter specifies the number of samples per gradient update. The verbose parameter is set to 1, which displays the training progress during each epoch..

Step 6: Predict and Evaluate.

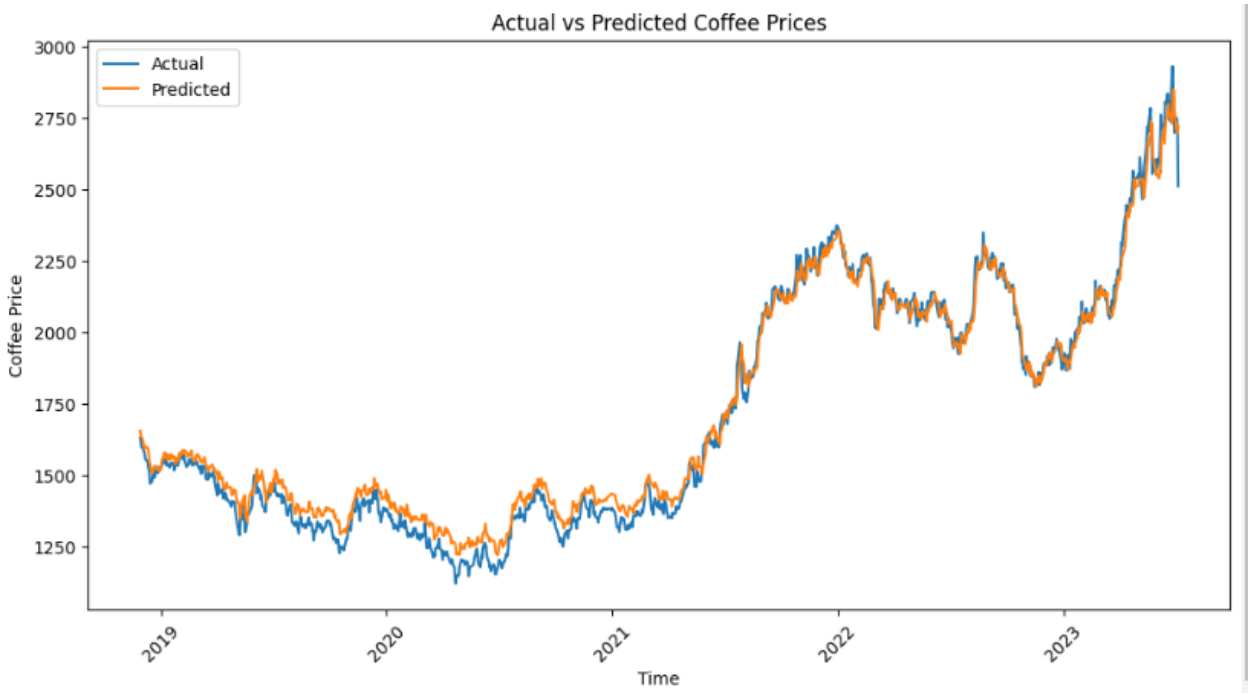


Figure 51: Predict on each test set (70/30,80/20,90/10)

MLP	7-3	48.42	2.61
	8 - 2	40.60	1.80
	9 - 1	40.57	1.36

Table 3: Model evaluation

Step 7: Calculate Loss per Epoch:

```
from sklearn.neural_network import MLPRegressor
import pandas as pd
import matplotlib.pyplot as plt

# Create an instance of the MLPRegressor and fit the model to the training data
mlp = MLPRegressor(hidden_layer_sizes=(50, 50), activation='relu', random_state=42)
mlp.fit(X_train_seq, y_train_seq)

# Get the loss per epoch during training
loss_per_epoch = mlp.loss_curve_

# Save the loss per epoch to a CSV file
loss_df = pd.DataFrame({'Epoch': range(1, len(loss_per_epoch) + 1), 'Loss': loss_per_epoch})
loss_df.to_csv('/content/loss_per_epoch.csv', index=False)

# Plot the loss per epoch
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(loss_per_epoch) + 1), loss_per_epoch)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss per Epoch')
plt.show()
```

Figure 52: Calculate Loss per epoch

Importing libraries: The required libraries, including sklearn.neural_network.MLPRegressor and matplotlib.pyplot, are imported.

Creating an instance of MLPRegressor: mlp = MLPRegressor(hidden_layer_sizes=(50, 50), activation='relu', random_state=42) creates an instance of MLPRegressor with two hidden

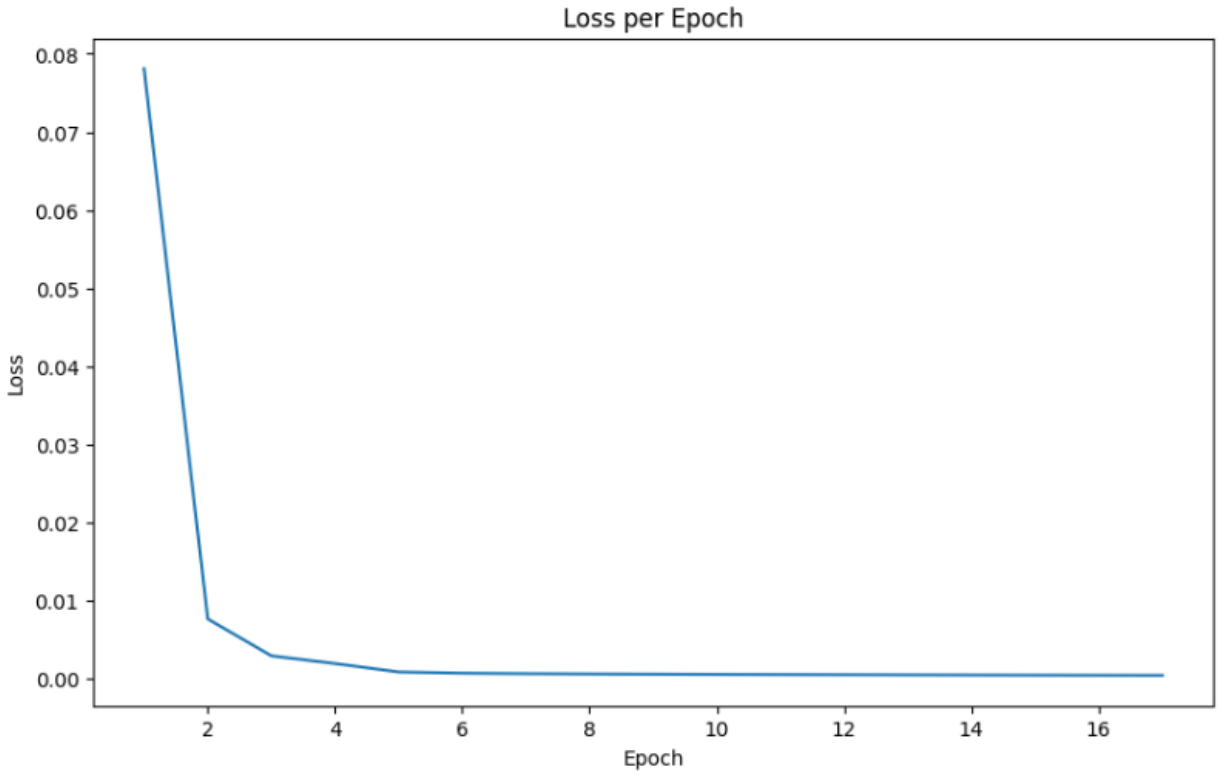
layers, each containing 50 units. The activation parameter is set to 'relu', which specifies the rectified linear unit activation function. The random_state parameter sets the seed value for random number generation, ensuring reproducibility.

Fitting the model: mlp.fit(X_train_seq, y_train_seq) trains the MLPRegressor model using the training data X_train_seq and y_train_seq.

Getting the loss per epoch: loss_per_epoch = mlp.loss_curve_ retrieves the loss per epoch during the training process. The loss_curve_ attribute stores the loss value at each epoch.

Saving the loss per epoch to a CSV file: The code creates a pandas DataFrame loss_df with two columns: 'Epoch' and 'Loss'. It saves the loss per epoch data to a CSV file named 'loss_per_epoch.csv' using the to_csv() function.

Plotting the loss per epoch: The loss per epoch is visualized using a line plot. plt.plot(range(1, len(loss_per_epoch) + 1), loss_per_epoch) plots the loss values against the epoch number. The remaining code sets the x-axis label, y-axis label, and title of the plot.



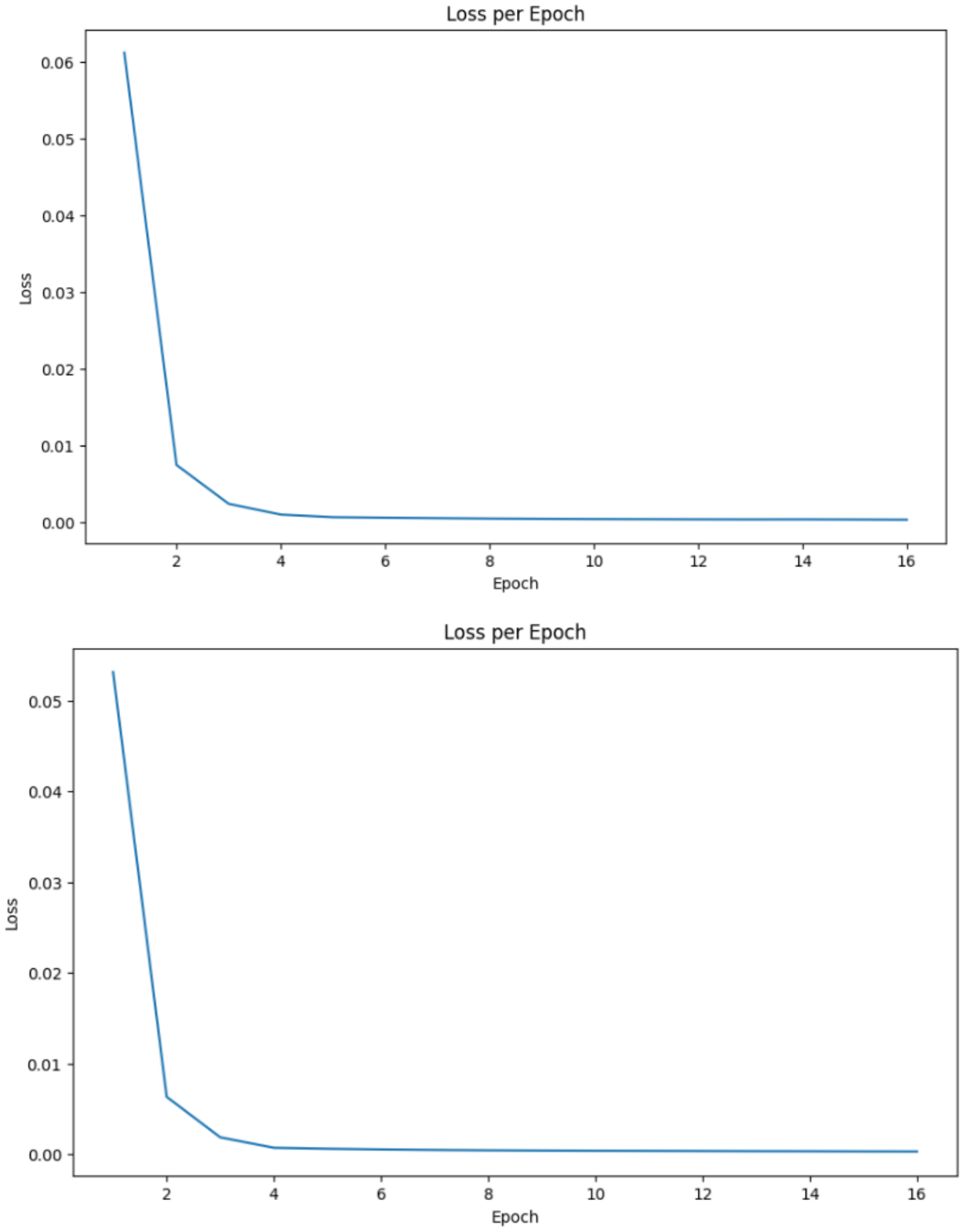
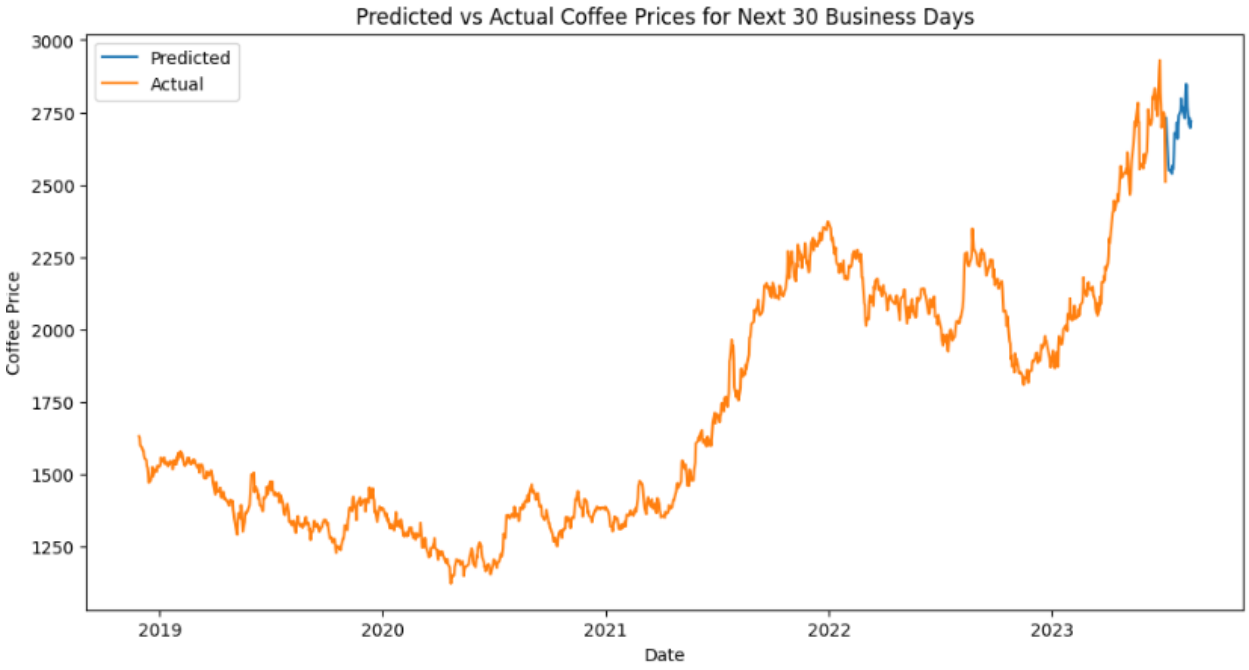
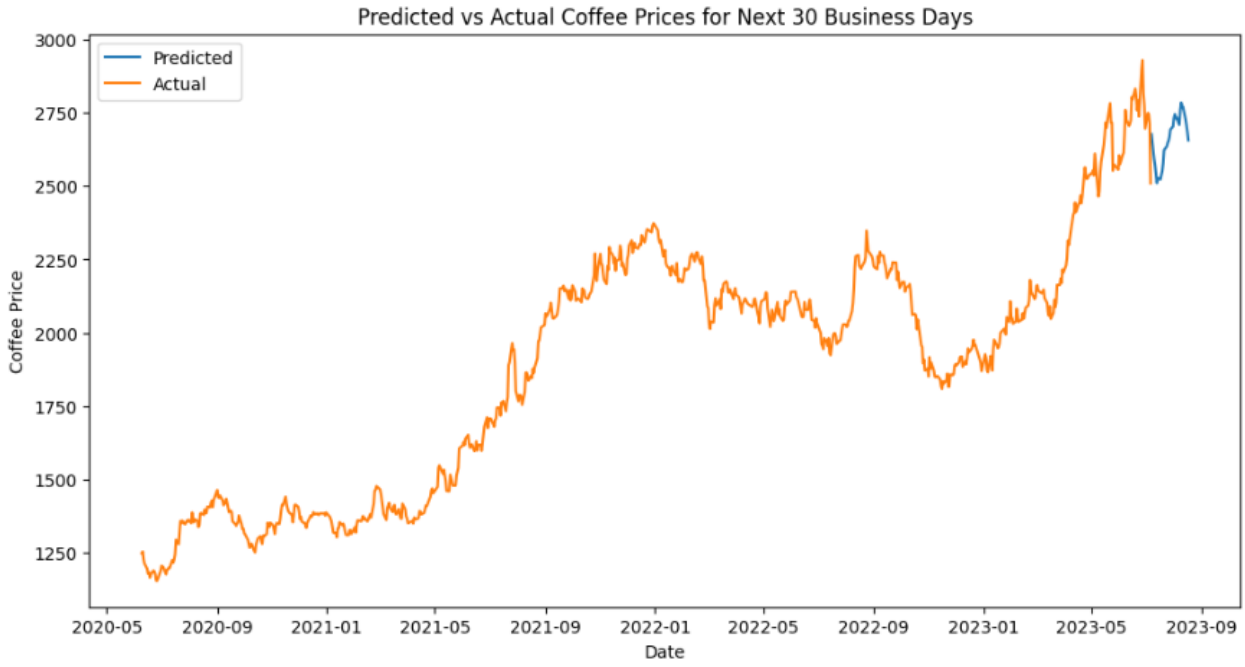


Figure 53: The loss of 70-30; 80-20; 90-10

Step 8: Forecast the next 30 days


Predicted vs Actual Coffee Prices for Next 30 Business Days
Predicted
Actual
Coffee Price
3000
2750
2500
2250
2000
1750
1500
1250
2019
2020
2021
2022
2023
Date


Predicted vs Actual Coffee Prices for Next 30 Business Days
Predicted
Actual
Coffee Price
3000
2750
2500
2250
2000
1750
1500
1250
2020-05
2020-09
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
Date

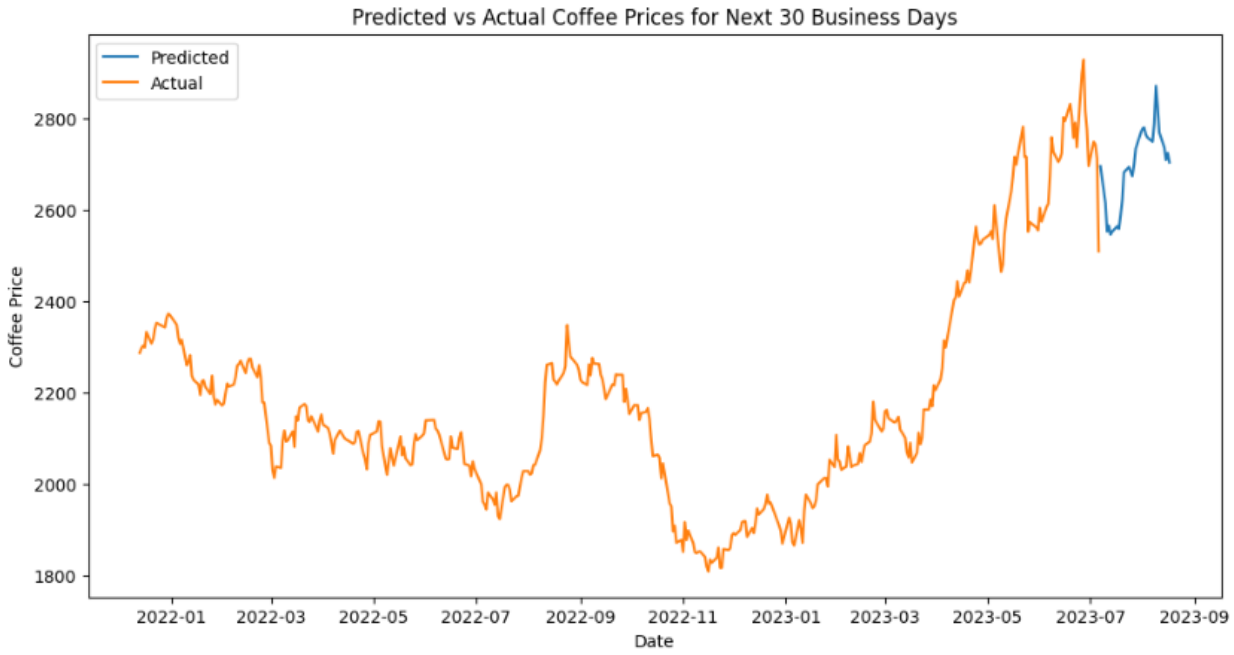


Figure 54: Visualization of the next 30 days for eachh test (70/30,80/20,90/10)

5.4. Experimental Result

MODEL	Train - Test	RMSE	MAPE(%)
LSTM	7-3	140.5	7.16
	8 – 2	139.38	6.15
	9 - 1	199.60	7.16
	7-3	136.02	6.57

GRU	8 – 2	121.32	6.23
	9 - 1	94.73	4.51
MLP	7-3	48.42	2.61
	8 - 2	40.60	1.80
	9 - 1	40.57	1.36

Figure 55: Result of all Models

For the LSTM model, the performance varies based on the train-test split ratio. In general, the RMSE ranges from 140.5 to 199.60, with the lowest value obtained in the 8-2 split. The MAPE ranges from 6.15% to 7.16%. The GRU model shows better performance compared to LSTM in terms of RMSE and MAPE. The RMSE ranges from 94.73 to 136.02, with the lowest value obtained in the 9-1 split. The MAPE ranges from 4.51% to 6.57%. The MLP model performs the worst among the three models, with higher RMSE and MAPE values. The RMSE ranges from 40.57 to 48.42, with the lowest value obtained in the 9-1 split. The MAPE ranges from 1.36% to 2.61%.

Chapter 6: Dashboard

6.1. Overview of Dashboard

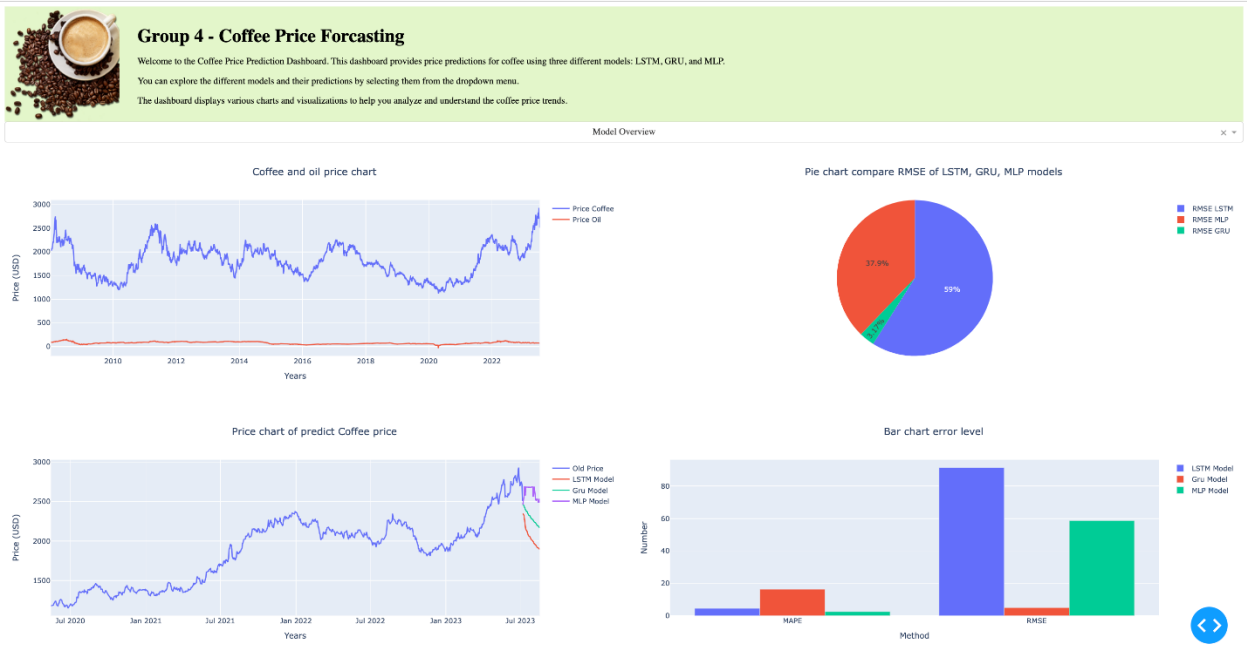


Figure 56: Overvierw Page

6.2. LSTM Model

Date	price_coffee_pre
2023-07-07	2347.110937654972
2023-07-10	2321.7067422866817
2023-07-11	2279.034930348396
2023-07-12	2227.522188544273
2023-07-13	2182.374873876571
2023-07-14	2151.5694179534908
2023-07-17	2127.0637203454967
2023-07-18	2106.017180860042
2023-07-19	2088.6689611077304
2023-07-20	2077.5299402475357

Figure 57: LSTM Page

6.3. MLP

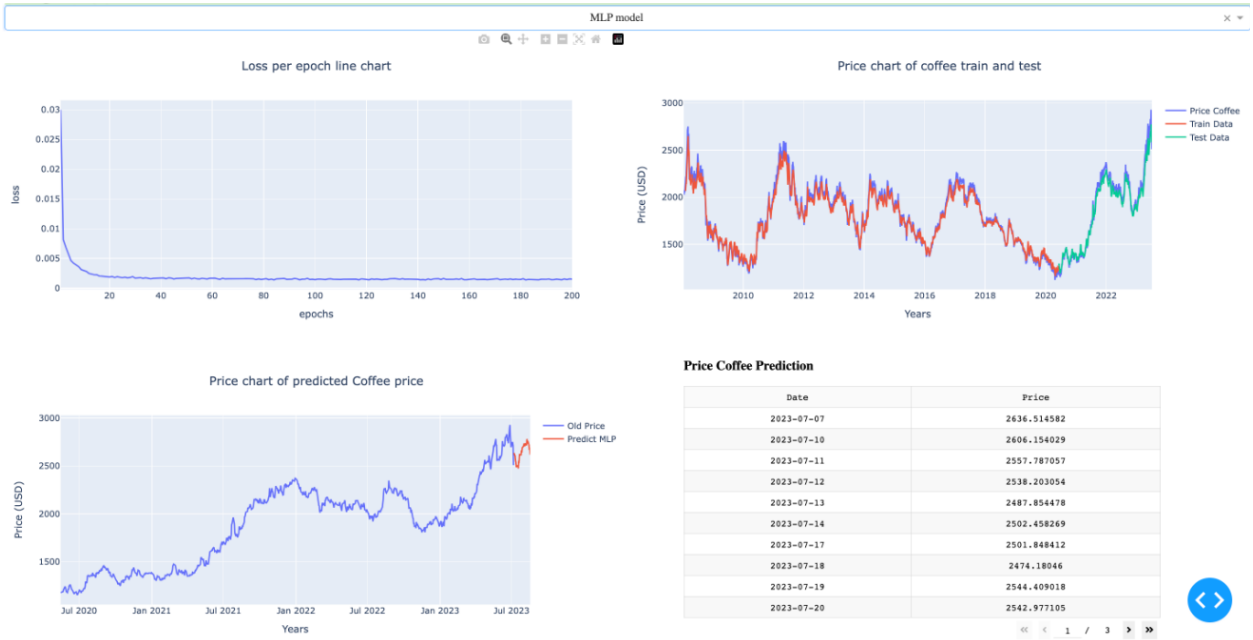


Figure 58: MLP Page

7.4. GRU

Figure 59: GRU Page

Chapter 7: Conclusion

6.1. Result

Coffee plays a significant role in the economic landscape of Vietnam. As one of the world's largest producers and exporters of coffee, particularly Robusta coffee. The income generated from coffee exports helps drive economic growth, create employment opportunities, and improve livelihoods for many individuals and communities. Predicting coffee prices is crucial for coffee businesses to effectively plan production, manage risks, and make informed decisions in a competitive market. The research findings demonstrate promising performance of the developed LSTM, GRU, and MLP models in forecasting coffee prices. These models have shown the potential to accurately predict coffee price movements based on historical data and relevant factors.

After analyzing the provided performance metrics for the LSTM, GRU, and MLP models, several insights can be drawn. The LSTM model stands out for its strong performance in terms of both RMSE and MAPE. Among the three configurations, the 8-2 configuration yields the best results, achieving the lowest RMSE value of 138.81 and a relatively low MAPE of 6.15%. This indicates that the LSTM model exhibits high accuracy and a smaller average percentage difference between its predictions and the actual values. Similarly, the GRU model shows good performance, with the 9-1 configuration achieving the lowest RMSE of 94.73 and the lowest MAPE of 4.51%. These results suggest that the GRU model is effective in predicting the data, particularly when considering the 9-1 configuration. On the other hand, the MLP model performs less favorably, with higher RMSE and MAPE values compared to the LSTM and GRU models. However, within the MLP configurations, the 8-2 configuration stands out with the lowest RMSE of 40.57and the lowest MAPE of 1.36%. These findings emphasize the importance of model selection and configuration in achieving accurate predictions. Ultimately, the choice of the best

model depends on specific requirements and priorities, such as interpretability, computation time, and the significance of different performance metrics.

6.2. Future Works

In the future, this research can be expanded by incorporating external factors such as weather conditions and economic indicators, which can have a significant impact on coffee prices. By integrating these factors into the forecasting models, the accuracy of predictions can be improved. Additionally, exploring ensemble models and hybrid approaches that combine the strengths of different models can further enhance the performance of the forecasting models. Advanced deep learning architectures, such as Transformer models and GANs, can be investigated to capture complex dependencies and improve the accuracy of predictions. Developing real-time forecasting capabilities would enable timely insights for decision-making in the coffee industry. Furthermore, incorporating measures of uncertainty and risk into the models can provide decision-makers with a better understanding of the potential range of outcomes and facilitate more informed decision-making processes. These future directions have the potential to enhance the applicability and effectiveness of coffee price forecasting models in practical settings.

Chapter 8: Duty Roster

Member	Name	Student ID	Works
1	Nguyễn Nhất Thưởng	20522000	LSTM
2	Lê Quang Hoà	20521331	GRU
3	Kiều Xuân Diệu Hương	20521381	MLP