Question 1

1. Time Horizon

In manufacturing, time horizons refer to the different time frames considered when crafting strategies regarding production processes. There are typically three time horizons:

- **Long Term**: These strategies cover one or more year and set the direction of a manufacturing firm in the long run (as its name suggests). Long term strategies may include strategic planning (e.g. when and where to build a new facility), partnerships and collaboration (e.g. who to purchase materials from), investment decisions (e.g. what new technologies and equipments to buy), and product development (e.g. improving current products through R&D).

- **Medium Term**: These strategies span a few weeks to months and are more tactical in nature. These decisions may include manpower management (e.g. recruiting new employees), resource allocation (e.g. deciding which markets will be supplied from which locations), inventory buildup planning and forecasting (e.g. what products to produce and how much).

- **Short Term**: These strategies cover a few hours to days and are operational in nature. They concern decisions regarding a manufacturing firm's daily operations, and may include decisions regarding delivery schedules, replenishment orders, and even quality control.

In short, a company survives in the long, medium, and short term by making good long, medium, and short term strategies respectively.

2. Focus

A company must **focus** in a certain area of manufacturing where they can thrive. For instance, a company like IKEA focuses on providing affordable, high quality furniture and other interior products. Meanwhile, a company like Apple chooses to focus on developing high technology gadgets such as the iPhone, iPad, and Mac. A company's focus can be developed by considering the following:

- **Process Technologies**: A company's focus determines what technologies it needs to use to manufacture its products. For instance, Tesla manufactures EVs (which means they need to develop Lithium-ion batteries) whereas Toyota prefers to develop hybrid and hydrogen based cars. Thus, Tesla and Toyota have different process technologies due to their differing focus.

- **Market Demands**: A company's focus must consider the market demands. For instance, Tesla develops cars for higher income owners (most of whom reside in the US/Europe) as compared to Toyota which tends to produce cars for a wider range of income (most of whom reside in Asia).

- **Production Volume**: A company can decide to produce in large batches to gain economies of scale or in small batches to enable higher customisation and product differentiation. For instance, DELL only manufactures laptops once it has received customized customer orders whereas other laptop companies like ASUS may choose to manufacture standardized laptops in bulk for a cheaper price.
- **Quality Level**: Some products have a higher quality (but also price) whereas others are cheaper but also lower in quality. For example, the typical Apple MacBook has a higher quality and price than the typical budget-version Lenovo notebook.
- **Manufacturing Tasks**: Different company focus means different manufacturing tasks to accomplish. For example, Tesla's manufacturing tasks involve automation, advanced robotics and gigafactories, whereas Toyota's involve Jidoka (an automation system with a human touch) and JIT (Just-in-time delivery).

3. Evaluation

A company needs to evaluate its performance through the following metrics.

- **Cost**: The company needs to identify what production costs can be minimized to ensure production efficiency.
- **Quality**: The company needs to assess the quality of its manufactured products, for example through random sampling every 1000 items produced. This is typically done by the Quality Control department.
- **Profitability**: The company needs to assess whether or not it is making money through its business by assessing financial measures such as ROA, ROE, and many more.
- **Customer Satisfaction**: The company needs to seek customer feedback to understand how a product satisfies (or dissatisfies) a customer's needs.

4. Consistency

A company needs to maintain the consistency of its manufactured products in the long term. They must ensure that future employees working in the same company are able to manufacture products while maintaining good quality. Achieving consistency can be done by ensuring professionalism across departments, product proliferation (expanding diversity of products offered), and making manufacturing tasks explicit (e.g. through proper technical documentation).

Question 2: Product-Process Matrix

(a) The product-process matrix describes the ideal combination of production volume and process complexity to select when manufacturing a certain product. The best combination typically follows the main diagonal (whereby the product volume increases as the process complexity decreases).

(b) Some examples of products and their manufacturing processes are shown in the matrix below.

Process \ Product	Low Volume	Medium Volume	High Volume	Very High Volume
Jumbled Flow (job-shop)	NASA Rocket			
Disconnected Line Flow (flow/batch)		Pfizer Vaccine		
Connected Line Flow (assembly)			Sausage & Canned Food	
Continuous Flow				Paper

Table 1: Product Process Matrix

(c) Companies must operate on the diagonal to ensure the highest rate of cost-effectiveness when manufacturing their products. Put it simply, the more complicated a product is, the harder it will be to produce them in higher volume (or setup production systems capable of high volume production). However, if a product is more simplistic, it will be worth the effort to setup mass production systems to produce these items in bulk since economies of scale can be achieved and increase overall company profitability. Beyond time and money is also quality: highly standardized systems such as assembly lines are more suitable for products requiring consistent quality (e.g. automobiles).

Operating outside of the diagonal for no good reason may cause a company to lose a lot of time and money unnecessarily. For instance, think of how absurd it would be to manufacture a NASA rocket through a continuous flow system. We would have to spend more time developing the system rather than the end product. Indeed, typically NASA develops a very small number of rockets every decade–but each has a very specific purpose (some rockets head to the moon, others to Mars, etc.). Thus, a lot of high-tech customisations are involved, making it impossible to produce such a rocket through a continuous flow system.

On the contrary, if, say, paper were to be manufactured through job-shops, each page of paper would take too much time and money to produce, and we would not be able to produce enough paper to everyone's need for paper.

Question 3

The five industrial revolutions occured thanks to various technological breakthroughs that took place in different periods which created a drastic increase in our overall manufacturing capacity. Each revolution enabled humans to manufacture products of not only higher quantity but also quality, and as a result drove a higher quality in life across the past three centuries.

First Industrial Revolution

The first industrial revolution took place following the development of the steam engine by James Watt in the 1760s. The steam engine allowed humans to harness greater amounts of energy than what was provided by the human or animal muscle. As such, various machines can now be used to perform manufacturing tasks as there was enough energy to power them. For instance, not long after Watt's steam engine, Richard Trevithick developed the world's first train which is powered by the steam engine. The train eventually enabled us to travel farther and faster than what the feet of man or horse can walk.

Second Industrial Revolution

The second industrial revolution took place when advancements in electricity, metallurgy, and petroleum between 1870 and 1920 enabled mass production systems to be erected. Now, not only were there even greater amounts of energy available, humans also had the capacity to build more sophisticated manufacturing systems through technologies such as the electric grid, railroad networks and telegraph. Examples of these breakthroughs include the hot blast technique (to produce iron), the Beseemer process (to produce steel), and the Fourdrinier machine (to produce paper).

Third Industrial Revolution

The third industrial revolution took place thanks to the introduction of computers and electronics to the industry after the 1940s. Now, manufacturing firms can automate various processes through the use of computers and electronics. Examples of these breakthroughs include the transistor (a critical component for digital devices), the computer, and database systems.

Fourth Industrial Revolution

The fourth industrial revolution is what many say is happening in the industry right now. Thanks to the introduction of the internet, the use and exchange of information has exploded, providing humans with an over-abundant amount of information. With the right steps, information can be processed into knowledge and actionable insights (e.g. through machine learning and analytics), further streamlining various industrial processes. Furthermore, our capacity for super-swift information exchange enables us to build cyber-physical systems, Internet of Things (IoT), and networks which increases connectivity between different elements or systems. All in all, these technological advancecements have pushed and will continue to push our productivity even further.

Fifth Industrial Revolution (& Bonus Question)

The fifth industrial revolution potentially takes place as artificial intelligence (AI) plays an even larger role in society. Through machine learning and neural networks, AI can be used to further streamline and industrialise various processes (not only in manufacturing). We are now entering an age where machines no longer perform the simplistic "manual" tasks in the industry but also the more "intelligent" ones (hence the name artificial intelligence). Put it simply, we are now creating machines which attempt to mimic (and even exceed) the human intelligence, with an overall aim to push the boundaries of what we can do. Indeed, thanks to AI, we are now able to do so many more things. From developing softwares at a faster speed, to writing essays while having to read less books, AI has enabled us to do so much more than we otherwise could.

At the same time, it is important to note that the fifth industrial revolution is taking place at such an unprecedented speed. Our society is being changed by AI perhaps far faster than what we can understand. In particular, through the constant use of AI tools such as ChatGPT, we are now increasingly tied down to these AI tools as we become more dependent on them for even the simplest of tasks. The lines separating humanity and technology are now blurring: we are increasingly unable to live without these technologies that they have practically become a part of our lives–and our bodies, even, (as we embed chips into our brains and organs).

Question 4

Given that the first unit took a worker 5 minutes to produce, by the equation for the learning curve, we have:

$$Y(u) = 5u^{-b}$$

Given a 75% learning curve, the time for the $2u$-th unit is 75% of the time for the u-th unit and thus:

$$\begin{aligned} 0.75 = \frac{Y(2u)}{Y(u)} &= \frac{5(2u)^{-b}}{5u^{-b}} = 2^{-b} \\ \ln(0.75) &= \ln\left(2^{-b}\right) = -b\ln(2) \\ b &= -\frac{\ln(0.75)}{\ln(2)} \\ &\boxed{b = 0.4150374993} \end{aligned}$$

Therefore, we have:

$$Y(100) = 5(100)^{-0.4150374993} \qquad Y(1000) = 5(1000)^{-0.4150374993}$$

$$\boxed{Y(100) = 0.73942649101 \text{ minutes}} \qquad \boxed{Y(1000) = 0.28435279827 \text{ minutes}}$$

Question 6

By taking the remainder of 92 (the last 2 digits of my student ID) divided by 26 (the total number of alphabets), we get 14, which refers to the letter O. The only country in the world with the letter O is Oman. 2 countries which start from a letter +/- 5 from O are Trinidad and Tobago and Jamaica.

An exploration of several credible websites, such as Our World in Data and World Bank provides the following data on the countries' GDP. Some key observations are obvious: Singapore has both the highest GDP and GDP per capita compared to the rest of the countries. Indeed, Singapore has the highest rate of development, followed by Oman (developed), Trinidad and Tobago (considerably developed), and Jamaica (developing) and this is reflected in each country's GDP per capita. Meanwhile, Oman has the second highest GDP per capita, followed by Trinidad and Tobago and Jamaica.

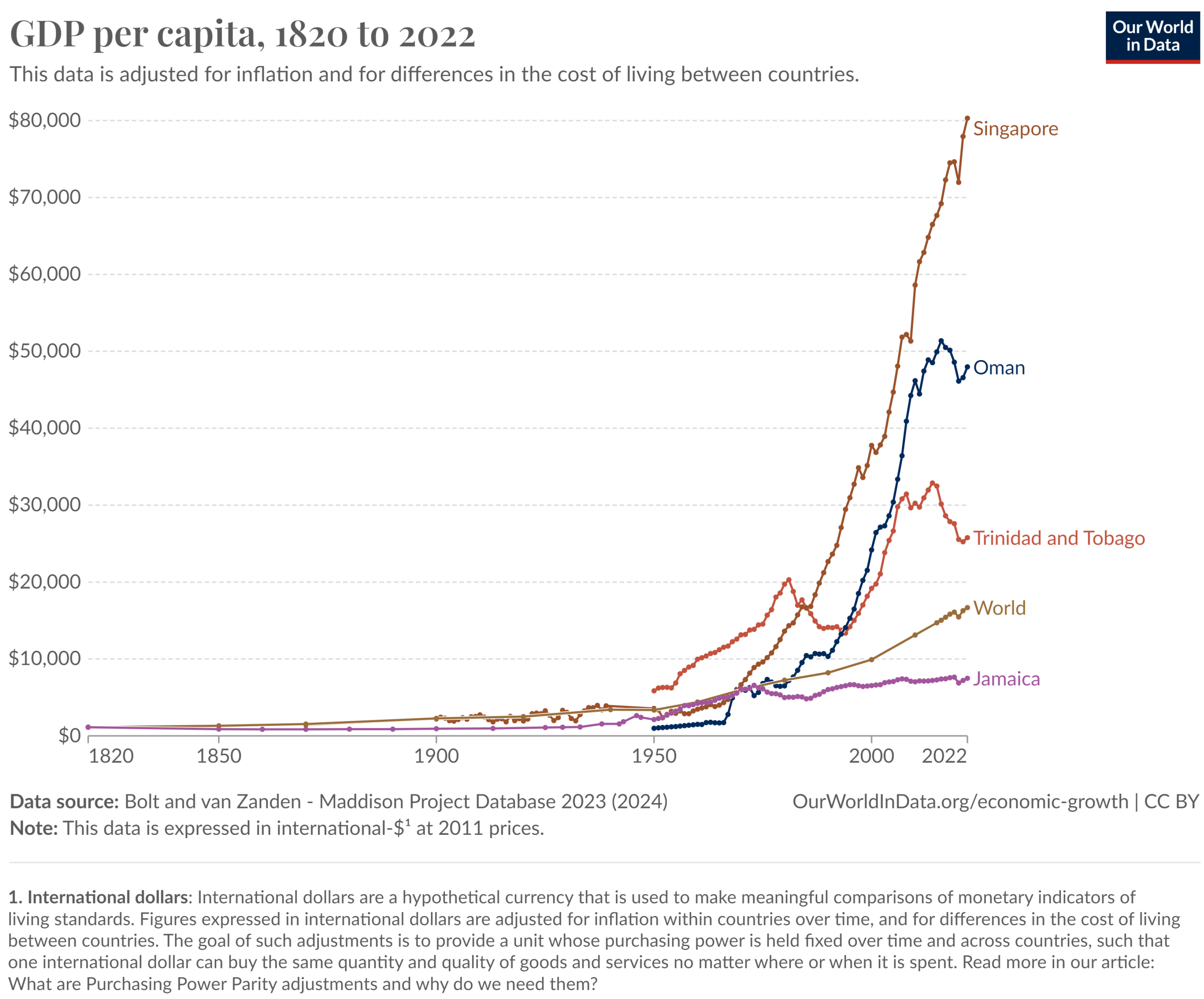

Figure 1: GDP per Capita of the 4 Countries, Source: Our World in Data

Singapore's GDP is the highest thanks to its effective economic policies which has hyperaccelerated growth in less than a decade. Meanwhile, Oman's GDP is second highest and this is thanks to their abundant oil reserves. Trinidad and Tobago, despite being a small country in size, is well developed especially compared to the rest of the Carribean (this explains their high GDP per capita but relatively lower GDP). Finally, Jamaica is a country that has been historically indebted, which may explain why their GDP is low.

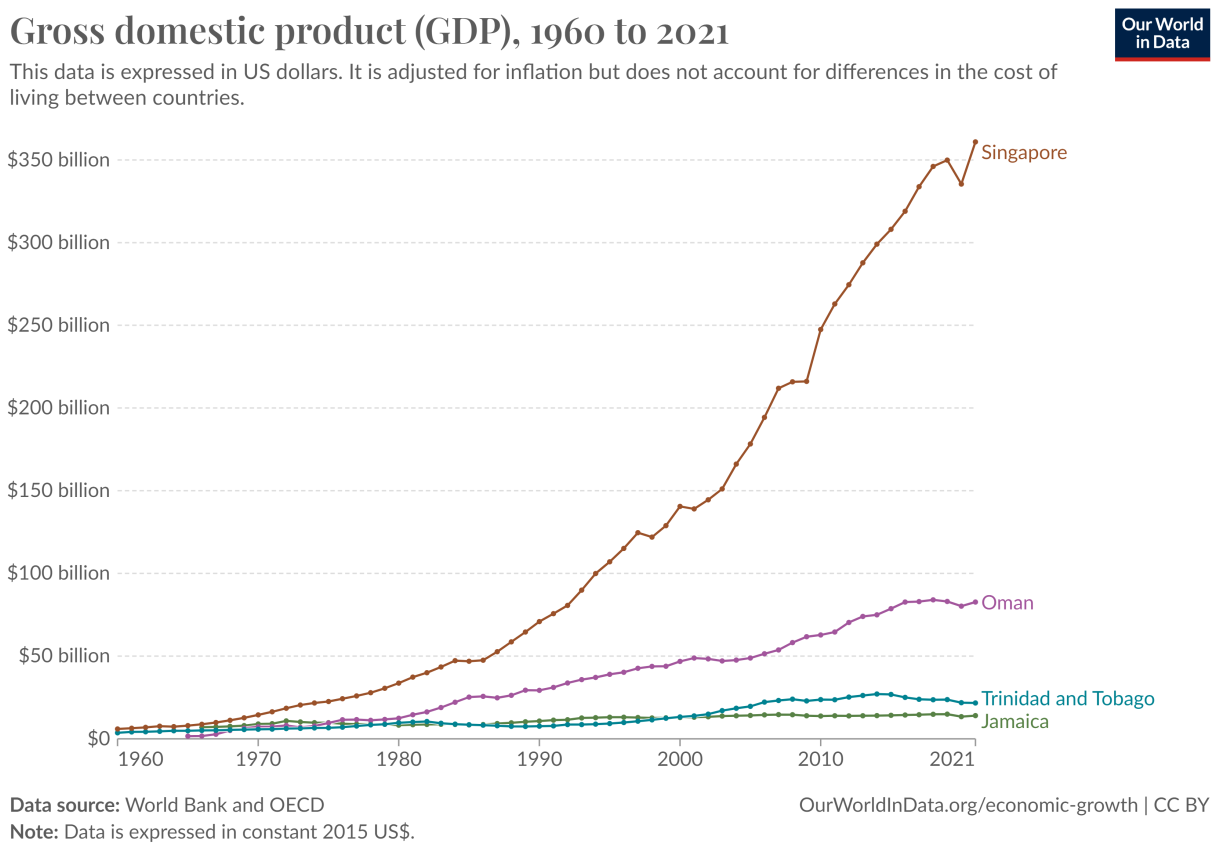

Figure 2: GDP of the 4 Countries, Source: Our World in Data

To understand how different economic sectors contribute in each of the countries, data was downloaded from the World Bank and plotted in Excel. In addition, the following table summarises the information given in the dataset.

	GDP ($B)		**Agriculture (%)**		**Industry (%)**		**Manufacturing (%)**		**Services (%)**	
	2015	**2022**	**2015**	**2022**	**2015**	**2022**	**2015**	**2022**	**2015**	**2022**
Singapore	308	466.8	0	0	24.3	24.2	18.1	20.5	70	70.9
Oman	78.7	114.7	1.7	1.8	53.1	57	8.9	10.5	50.1	44.5
Trinidad and Tobago	27	30.1	1.7	1.1	35.7	48.9	14.1	22.2	58.1	47.8
Jamaica	14.2	17.1	6.3	8.1	19.3	19.9	8	8.5	61.8	58.3

Table 2: GDP by Economic Sector, Source: World Bank

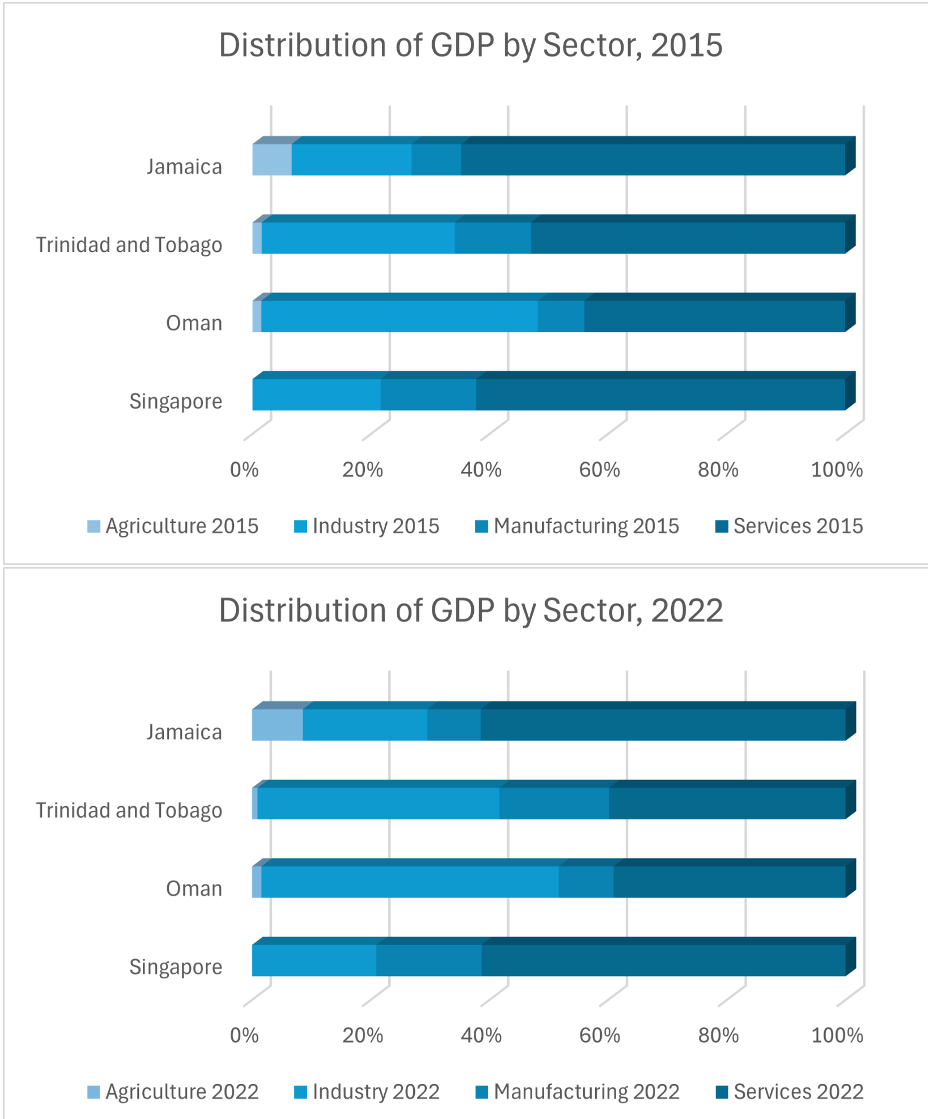

Figure 3: GDP by Economic Sector, Source: World Bank

Oman

Compared to the rest of the countries, Oman's industry sector contribution is largest at above 50% for both 2015 and 2022. This is once again due to the fact that Oman has abundant oil reserves and majorly relies on this resource for its economy. Meanwhile, Oman's manufacturing sector plays a smaller role at about 10%. Between 2015 and 2022, there was only a <2% growth in Oman's manufacturing sector, which may be a sign that Oman's manufacturing sector can further be improved. Another key sector to Oman's economy is the services industry, which is a sector where most of its middle income citizens earn their money from.

Trinidad and Tobago

Trinidad and Tobago's agriculture sector is the second lowest across the four countries, and this may be due to their geographical location in the Caribean which makes agriculture and farming difficult. Trinidad and Tobago's main economic sectors are industry and services. Like Oman, Trinidad's economy is largely supported by its large reserves of petroleum and petrochemicals, especially liquified natural gas (LNG). In addition, Trinidad has earned a reputation as an excellent investment site for LNG, enabling the industrial sector to grow further within the country.

Trinidad's manufacturing sector contributes an awesome 22.2% of their economy in 2022. For one, Trinidad is better than Oman when it comes to performing "downstream processing" of its petrochemicals. This adds a lot of value into the products they sell, enabling them to earn more money and have a more diversified, robust economy. The country also has a robust logistics infrastructure and an excellent international reputation which draws foreign investment into the country. This boosts growth in manufacturing, explaining why the sector performs well in Trinidad's economy.

Jamaica

With a population of 2.8 million people (smaller than Singapore), the economy of Jamaica is heavily reliant on the services sector. Jamaica also has a very suitable geography for agriculture and tourism, which explains why the agriculture sector contributed to 6.3 and 8.1% of the economy in the years 2015 and 2022 respectively (as compared to 0% for Singapore and <2% for Oman and Trinidad and Tobago).

Jamaica's manufacturing sector has a relatively low contribution to the economy. Nonetheless, Jamaica's manufacturing still plays an important role, providing various goods the country needs. In particular, the garment industry plays a prominent role in contributing to Jamaica's manufacturing sector.

Singapore

What is most impressive of Singapore is its highly developed economy which is one of the most open, least corrupt, and most pro-business in the world. Equally interesting is the pivotal role the country's manufacturing plays in its economy, characterized by a sophisticated approach to intermediary and entrepôt trade. The country excels in purchasing raw materials, refining them, and then re-exporting the finished products, particularly in industries like wafer fabrication and oil refining. This strategy is supported by Singapore's strategic port, which facilitates efficient transshipment activities, making it a competitive global player. Singapore is also well known for its being a high technology manufacturing hub.

Singapore's services sector also plays a really impressive role in their knowledge-based economy. Many financial and technological companies branch out in Singapore, and this contributes to a large percentage of their economic growth. Apart from high tech companies, however, the services sector are also backed by the food industry (after all, food is a human's primary need). Be it hawker centers or restaurants in malls, food plays an important role in Singapore's economy.

Of course, having limited land, Singapore's agriculture contributes to 0% of their economy. Singapore thus needs to import food and water from neighbouring countries, and in this sense they are dependent towards their neighbours. Nonetheless, they have developed a diverse range of sources such that they are not overly dependent on a single country to source their food and water.

Question 7

About the competition

The Makridakis Competitions (otherwise known as the M Competition) are a series of open competitions to test out different time series forecasting methods. The first M competition (M1) was held by its founder Spyrow Makridakis in 1982, where 1001 time series were forecasted using 24 different techniques. Over time, the competition was held again until M6 (and still currently taking place). Today, the competition is well known and various postdoctoral researchers and graduate students from around the world participate in the competition to test out forecasting methods they develop.

The M5 Competition

As I was interested to learn more about neural networks, I decided to explore a Long Short-Term Memory (LSTM) model to create the forecasts. The LSTM is a type of Recurrent Neural Network that can be used to develop forecasts. However, it does require an ample amount of data and as such, the M1 dataset is insufficient to be used here. Instead, I used the M5 competition dataset and this provided me with a lot more data. More information regarding the competition can be found in the official M5 website or in kaggle.

The goal of the M5 competition is to create 28 point forecasts for 42,840 timeseries. Each timeseries represent the sales of a certain item in a certain Walmart store from Texas, California, or Wyoming. The M5 competition is different from previous M competitions in several ways. Most notably, its datasets provides information beyond the timeseries themselves as there are explanatory variables such as sell prices, days of the week, and special events. These information can be accounted for when creating the forecast model, enabling for greater forecasting results to be achieved. Another interesting feature of the competition is that its timeseries display intermittency (i.e. sporadic demand including zeros).

Exploratory Data Analysis

The dataset used in the M5 competition was provided by Walmart. The dataset contains the unit sales of various products sold in the USA, organized in the form of grouped time series.

The unit sales data span across 3,049 different products of 3 different categories (Hobbies, Foods, and Household) and 7 product departments. There are ten stores located in the three states. So essentially the dataset has a hierarchical structure as illustrated in figure 4.

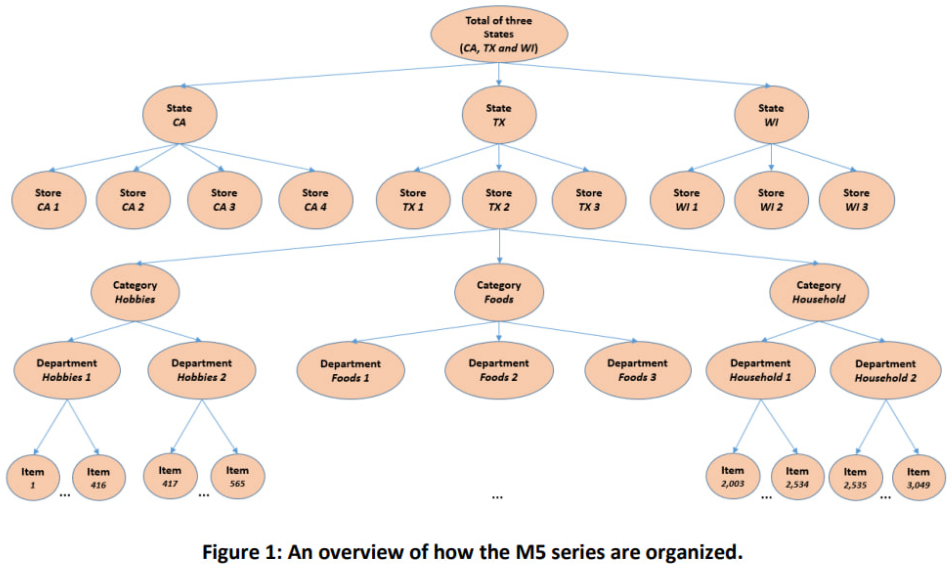

Figure 4: Taken from the M5 Competition Guide

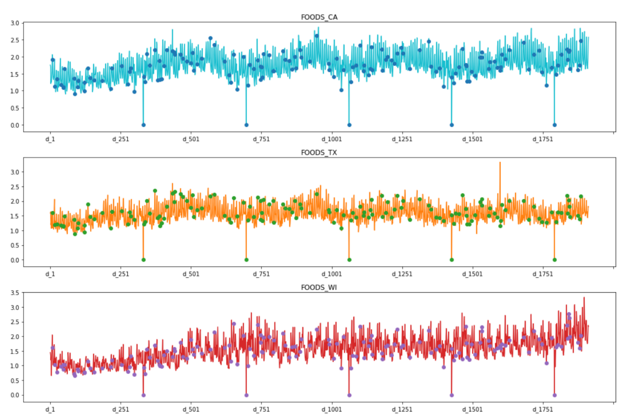

Figure 5: Taken from https://github.com/rruss2/M5_competition/tree/master

To understand the data, we can plot out several charts which help to visualise the data. The plots below will help us in identifying certain patterns and findings related to the dataset. For example, figure 5 displays the total sales for the FOODS category in each state. Then we also have figure 6 where each plot and category is plotted. These plots are taken from a GitHub repository made by Robert Russ.

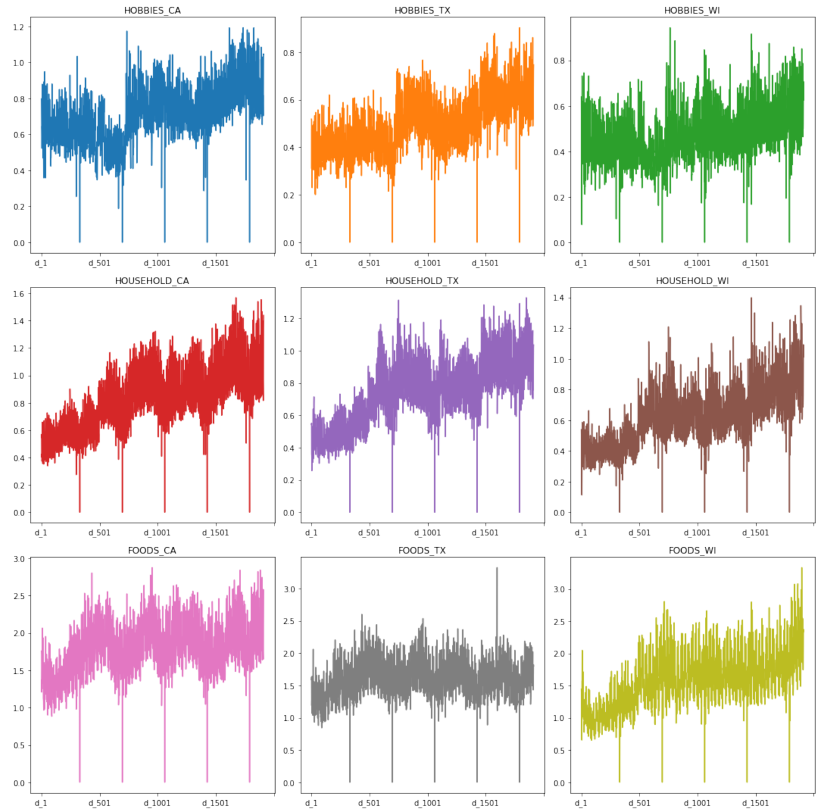

Figure 6: Taken from https://github.com/rruss2/M5_competition/tree/master

Self-Exploration, Iteration 1

To start exploring the LSTM, I followed a tutorial made by Gregg Hogg which explores a simple LSTM model to predict a univariate timeseries. Since there are 42,840 series, I decided to start with a simple forecasting model first which predicts the sale prices for a joint time series: the time series of item 'FOODS_3_827' for 4 stores from California (CA_1, CA_2, CA_3, CA_4).

	store_id	item_id	wm_yr_wk	sell_price
2029680	CA_3	FOODS_3_827	11409.0	1.0
2029681	CA_3	FOODS_3_827	11410.0	1.0
2029682	CA_3	FOODS_3_827	11411.0	1.0
2029683	CA_3	FOODS_3_827	11412.0	1.0
2029684	CA_3	FOODS_3_827	11413.0	1.0

(379, 4)

Figure 7: Timeseries used for the model

Afterwards, I initiated an LSTM Model using Python's TensorFlow module and trained the model. The dataset was split into a training, validation and test set. My full workings can be accessed here.

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (None, 64)                16896

 dense_10 (Dense)            (None, 8)                 520

 dense_11 (Dense)            (None, 1)                 9

=================================================================
Total params: 17425 (68.07 KB)
Trainable params: 17425 (68.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 8: Model initialization

```
[16] model1.fit(X_train1, y_train1, validation_data=(X_val1, y_val1), epochs=10, callbacks=[cp1])

Epoch 1/10
6/6 [==============================] - 6s 905ms/step - loss: 0.8413 - root_mean_squared_error: 0.9172 - val_loss: 0.8287 - val_root_mean_squared_error: 0.9103
Epoch 2/10
6/6 [==============================] - 31s 6s/step - loss: 0.8199 - root_mean_squared_error: 0.9055 - val_loss: 0.8073 - val_root_mean_squared_error: 0.8985
Epoch 3/10
6/6 [==============================] - 5s 946ms/step - loss: 0.7986 - root_mean_squared_error: 0.8936 - val_loss: 0.7859 - val_root_mean_squared_error: 0.8865
Epoch 4/10
6/6 [==============================] - 3s 685ms/step - loss: 0.7771 - root_mean_squared_error: 0.8815 - val_loss: 0.7644 - val_root_mean_squared_error: 0.8743
Epoch 5/10
6/6 [==============================] - 4s 811ms/step - loss: 0.7569 - root_mean_squared_error: 0.8700 - val_loss: 0.7457 - val_root_mean_squared_error: 0.8636
Epoch 6/10
6/6 [==============================] - 5s 915ms/step - loss: 0.7376 - root_mean_squared_error: 0.8589 - val_loss: 0.7258 - val_root_mean_squared_error: 0.8520
Epoch 7/10
6/6 [==============================] - 4s 897ms/step - loss: 0.7175 - root_mean_squared_error: 0.8470 - val_loss: 0.7053 - val_root_mean_squared_error: 0.8398
Epoch 8/10
6/6 [==============================] - 4s 823ms/step - loss: 0.6967 - root_mean_squared_error: 0.8347 - val_loss: 0.6844 - val_root_mean_squared_error: 0.8273
Epoch 9/10
6/6 [==============================] - 3s 693ms/step - loss: 0.6757 - root_mean_squared_error: 0.8220 - val_loss: 0.6629 - val_root_mean_squared_error: 0.8142
Epoch 10/10
6/6 [==============================] - 5s 931ms/step - loss: 0.6539 - root_mean_squared_error: 0.8087 - val_loss: 0.6408 - val_root_mean_squared_error: 0.8005
<keras.src.callbacks.History at 0x78a5b7299900>
```

Figure 9: Model Training

To assess the model, I plotted the predictions against the original data for all test sets. The plots show that there is a very large difference between the predictions and the original data, meaning that the model still needs a lot more tweaking. It is largely possible that the model is incapable of making good predictions due to the way the data is fed into the model and also the very small unit price dataset available. As such, in the next iteration, I decided to forecast the unit sales instead as more data was available.

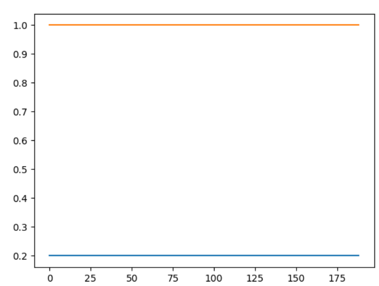

Figure 10: Forecasts (blue) vs Original (orange), Train Set

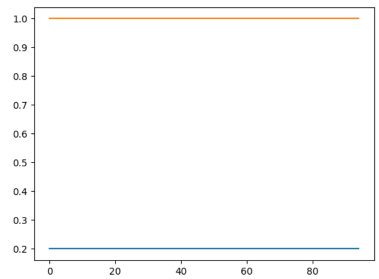

Figure 11: Forecasts (blue) vs Original (orange), Validation Set

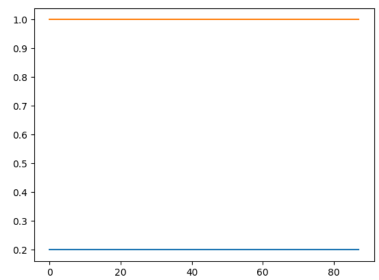

Figure 12: Forecasts (blue) vs Original (orange), Test Set

Self-Exploration, Iteration 2

In my second try, I tried to implement a different code in my model (although still using an LSTM) following a tutorial here. This time, I tried to perform the model training locally using my own GPU. I also selected a single item in a single store instead of several stores ('HOBBIES_1_001_CA_1'). The model took about 20 minutes to complete, and as such I could not train one model for each the series. All my work can be found here.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np
import torch

# GPU SETUP
print(f"Number of devices available: {torch.cuda.device_count()}")
if torch.cuda.is_available():
    print(f"GPU: {torch.cuda.get_device_name(0)} is available.")
else:
    print("No GPU available. Training will run on CPU.")
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Pytorch version: {torch.__version__}")
```

[31] ✓ 0.0s Python

```
Number of devices available: 1
GPU: NVIDIA GeForce RTX 3050 Laptop GPU is available.
Pytorch version: 2.3.0+cu118
```

Figure 13: Setting up the GPU

```python
# DATA VISUALIZATION

series = 'HOBBIES_1_001_CA_1'
timeseries_train_evaluation = df_train_evaluation[[series+'_evaluation']].values.astype('float32')
timeseries_train_validation = df_train_validation[[series+'_validation']].values.astype('float32')

def remove_leading_zeros(series):
    series = list(series)
    first_non_zero_index = next((index for index, value in enumerate(series) if value != 0), None)
    if first_non_zero_index is not None:
        return series[first_non_zero_index:]
    return series

timeseries_train_evaluation = np.array(remove_leading_zeros(timeseries_train_evaluation))
timeseries_train_validation = np.array(remove_leading_zeros(timeseries_train_validation))

plt.figure(figsize=(30, 6))
plt.plot(timeseries_train_evaluation)
plt.title(f"Timeseries Plot for {series}, evaluation")
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()

plt.figure(figsize=(30, 6))
plt.plot(timeseries_train_validation)
plt.title(f"Timeseries Plot for {series}, validation")
plt.xlabel('Time')
plt.ylabel('Value')
plt.show()
```
[47] ✓ 0.3s Python

Figure 14: Data Visualization

To visualize the original data, I plotted the original data as shown below.

Figure 15: Plot for the Timeseries, Evaluation and Validation Set

Afterwards, I simply implemented and modified the code given by the website to complete the model training. To evaluate the results, I calculated and plotted the prediction results.

```python
# DATA LOADING FOR THE MODEL

def create_dataset(dataset, lookback):
    """Transform a time series into a prediction dataset

    Args:
        dataset: A numpy array of time series, first dimension is the time steps
        lookback: Size of window for prediction

    Source:
        https://machinelearningmastery.com/lstm-for-time-series-prediction-in-pytorch/
    """
    X, y = [], []
    for i in range(len(dataset)-lookback):
        feature = dataset[i:i+lookback]
        target = dataset[i+1:i+lookback+1]
        X.append(feature)
        y.append(target)
    return torch.tensor(X), torch.tensor(y)

lookback = 7
X_train, y_train = create_dataset(timeseries_train_evaluation, lookback=lookback)
X_test, y_test = create_dataset(timeseries_train_validation, lookback=lookback)

# Transfer datasets to the GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
X_train, y_train = X_train.to(device), y_train.to(device)
X_test, y_test = X_test.to(device), y_test.to(device)
```

✓ 0.0s Python

Figure 16: Dataset Preparation

```python
# MODEL DEFINITION
import torch.nn as nn
import torch.optim as optim
import torch.utils.data as data

class MyLSTM(nn.Module):
    def __init__(self):
        super(MyLSTM, self).__init__()
        self.lstm = nn.LSTM(input_size=1, hidden_size=50, num_layers=1, batch_first=True)
        self.linear = nn.Linear(50, 1)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = self.linear(x)
        return x
```

✓ 0.0s Python

```python
# MODEL TRAINING

model = MyLSTM().to(device)
optimizer = optim.Adam(model.parameters())
loss_fn = nn.MSELoss()
loader = data.DataLoader(data.TensorDataset(X_train, y_train), shuffle=True, batch_size=8)

n_epochs = 2000
train_rmse_list, test_rmse_list = [], []
for epoch in range(n_epochs):
    model.train()
    for X_batch, y_batch in loader:
        y_pred = model(X_batch)
        loss = loss_fn(y_pred, y_batch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    # Validation
    if epoch % 100 == 0 or epoch == (n_epochs-1):
        model.eval()
        with torch.no_grad():
            y_pred_train = model(X_train)
            train_rmse = np.sqrt(loss_fn(y_pred_train, y_train).item())
            y_pred_test = model(X_test)
            test_rmse = np.sqrt(loss_fn(y_pred_test, y_test).item())
            train_rmse_list.append(train_rmse)
            test_rmse_list.append(test_rmse)
        print(f"Epoch {epoch}: train RMSE {train_rmse:.4f}, test RMSE {test_rmse:.4f}")
print("Training complete.")

# Plotting the RMSE
plt.figure(figsize=(10, 5))
plt.plot(range(0, n_epochs+1, 100), train_rmse_list, label='Train RMSE')
plt.plot(range(0, n_epochs+1, 100), test_rmse_list, label='Test RMSE')
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Line Plot from List')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.title('Model Evaluation RMSE')
plt.grid(True)
plt.show()
```

✓ 7m 42.5s Python

Figure 17: Model Definition & Training

```
Epoch 0: train RMSE 0.8863, test RMSE 0.8637
Epoch 100: train RMSE 0.8188, test RMSE 0.8054
Epoch 200: train RMSE 0.7783, test RMSE 0.7678
Epoch 300: train RMSE 0.7650, test RMSE 0.7551
Epoch 400: train RMSE 0.7631, test RMSE 0.7528
Epoch 500: train RMSE 0.7597, test RMSE 0.7497
Epoch 600: train RMSE 0.7581, test RMSE 0.7491
Epoch 700: train RMSE 0.7567, test RMSE 0.7476
Epoch 800: train RMSE 0.7562, test RMSE 0.7472
Epoch 900: train RMSE 0.7563, test RMSE 0.7472
Epoch 1000: train RMSE 0.7553, test RMSE 0.7464
Epoch 1100: train RMSE 0.7552, test RMSE 0.7463
Epoch 1200: train RMSE 0.7548, test RMSE 0.7459
Epoch 1300: train RMSE 0.7547, test RMSE 0.7457
Epoch 1400: train RMSE 0.7549, test RMSE 0.7458
Epoch 1500: train RMSE 0.7545, test RMSE 0.7452
Epoch 1600: train RMSE 0.7542, test RMSE 0.7453
Epoch 1700: train RMSE 0.7542, test RMSE 0.7456
Epoch 1800: train RMSE 0.7549, test RMSE 0.7457
Epoch 1900: train RMSE 0.7538, test RMSE 0.7450
Epoch 1999: train RMSE 0.7540, test RMSE 0.7449
Training complete.
```

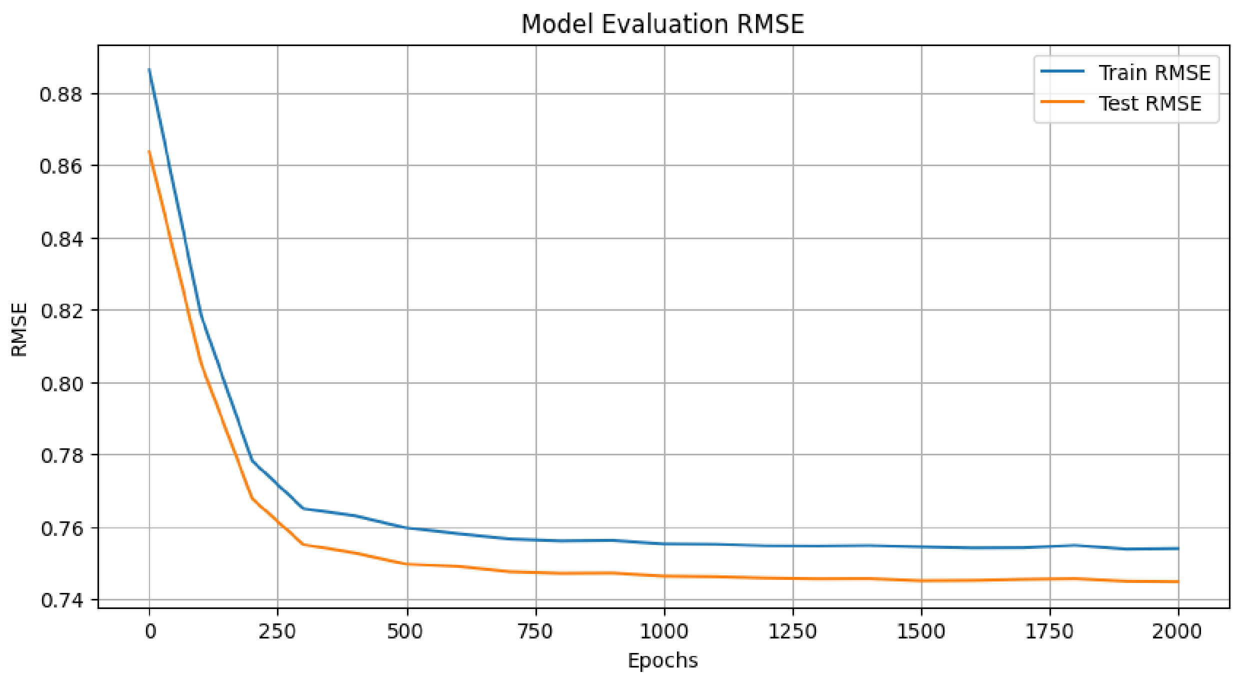

Figure 18: Model Training

Evaluation

Overall, the LSTM model I developed so far is incapable of making the best predictions. Although the Neural Network "works", more tweaking has yet to be done to fully capture the nature of the datasets (and to do so in the most efficient manner). Indeed, the hierarchical nature of the dataset might have been lost during the model training, as I did not consider any of the explanatory variables (e.g. state, department, event, etc.). Due to the time limitations, I was only able to develop the most basic time series forecast, and in this

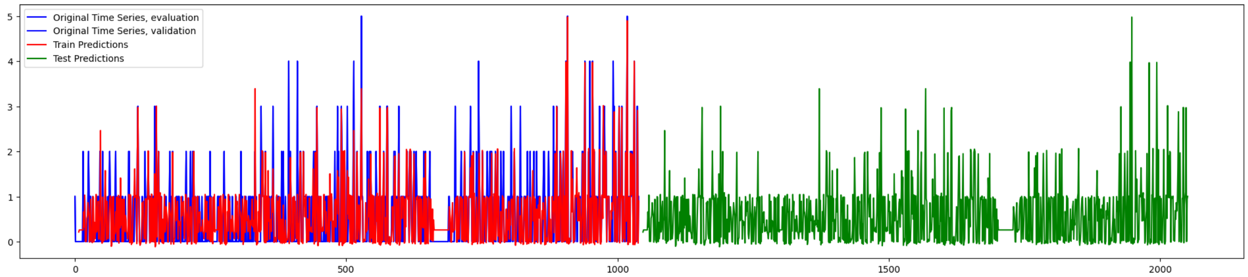

Figure 19: Prediction Results

```python
# sales = df_train_validation[series+"_validation"].tail(7)
past_7_days_sales = np.array(df_train_evaluation[series+"_evaluation"].tail(7))

input_data = torch.tensor(past_7_days_sales, dtype=torch.float32).view(1, 7, 1).to(device)

model.eval()

dict_predictions = {}
for i in range(4):
    weekly_prediction = []
    for _ in range(7):
        with torch.no_grad():
            if i == 0:
                input_data = torch.tensor(past_7_days_sales, dtype=torch.float32).view(1, 7, 1).to(device)
            else:
                past_prediction = dict_predictions[i-1]
                input_data = torch.tensor(past_prediction, dtype=torch.float32).view(1, 7, 1).to(device)

            prediction = model(input_data)
            predicted_value = prediction[0, -1, 0].item()
            weekly_prediction.append(predicted_value)

            input_data = torch.cat((input_data[:, 1:, :], torch.tensor([[[predicted_value]]], dtype=torch.float32).to(device)), dim=1)
    dict_predictions[i] = np.array(weekly_prediction)

for prediction_key in dict_predictions.keys():
    print(list(dict_predictions[prediction_key]))
```

✓ 1.3s Python

```
[1.4748221635818481, 1.4748221635818481, 1.4748221635818481, 1.4748221635818481, 1.4748221635818481, 1.4748221635818481, 1.4748221635818481]
[0.4461688995361328, 0.4461688995361328, 0.4461688995361328, 0.4461688995361328, 0.4461688995361328, 0.4461688995361328, 0.4461688995361328]
[0.3961995244026184, 0.3961995244026184, 0.3961995244026184, 0.3961995244026184, 0.3961995244026184, 0.3961995244026184, 0.3961995244026184]
[0.4760608673095703, 0.4760608673095703, 0.4760608673095703, 0.4760608673095703, 0.4760608673095703, 0.4760608673095703, 0.4760608673095703]
```

Figure 20: Prediction Results

particular competition, it would seem that a more sophisticated technique would be required to perform well. The top winners of the M5 competition spent half a year developing their models, and as such getting a model close to their performance would also take a lot of time.