Task B1: Set Up

Conducted by:

Group 1

Team Members	*Email*
Thanh Tam Vo	103487596@student.swin.edu.au
Phuc Khai Hoan Cao	103804739@student.swin.edu.au
Tai Minh Huy Nguyen	104220352@student.swin.edu.au

March 4, 2024

Table of Contents

1 Introduction

This report aims to demonstrate how to set up before running the code base of **V0.1** and **P1**. After demonstrating how to set up the environment for the entire project, this report will show the executed result of the tutorial from *Youtube* **V0.1** and the source code **P1** from a public *Github Repo*.

The Github Repository for the entire project *Stock Price Prediction* is also provided in this report.

2 Before You Can Run

Note: Due to the limitation of resources, the setting up process in this report is only demonstrated on Windows OS.

Link For the GitHub Repository: Please visit this **link** to download the entire project. At this moment, only my team members and lecturer can access the Repository. The Repository will be published publicly at the end of the semester (April 12, 2024).

2.1 Hardware Requirements

This project requires to use Tensorflow, which is an open-source for training machine learning model, so it is highly recommended to execute the program on the computer with **x86 Architecture** instead of **ARM Architecture**. Moreover, for users who experience in Apple Sillicon, please use a virtual machine or any cloud platforms (*Google Colab, etc.*) to run **V0.1** and **P1**.

2.2 Python Interpreter Requirements

Make sure that Python is already installed on your machine. Moreover, your Python Version should be **3.9-3.11**.

If Python is not installed in the local machine, please visit this link in order to install on your device.

2.3 Installation

Below is the step-by-step for the installation process, it is highly recommended to use a virtual environment for this project, if you decided to install all dependencies on your machine, you can skip the Creating Virtual Environment Step:

1. Open **PowerShell** as Administrator Privilege
2. Execute this Command Line: **`Set-ExecutionPolicy RemoteSigned`**
3. Execute this Command Line: **`python3 -m venv myenv`** to create the virtual environment
4. Execute this Command Line: **`myenv\Scripts\activate`** to activate the virtual environment
5. Execute this Command Line: **`.\requirements.bat`** to install all dependencies

From now, you are able to run **V0.1** and **P1**.

3 V0.1

The tutorial from the Youtube uses the **.py** extension, but the source code will be modified: instead of using **.py** file, the source code for **V0.1** is written in **.ipynb**, which is an extension for Jupyter Notebook file.

3.1 Jupyter Notebook Organization for V0.1

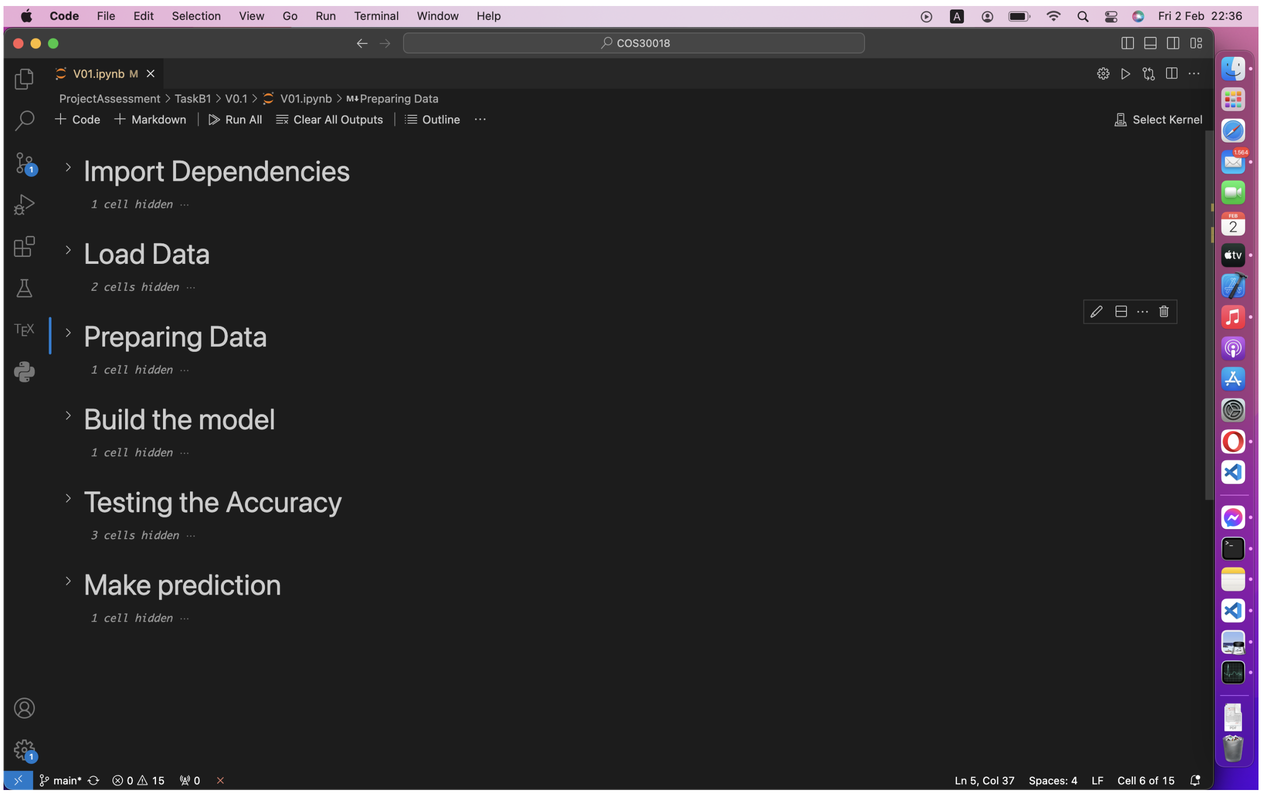


Figure 1: Structure Overview for **V01.ipynb**

The Figure 1 above is the overview for the **V01.ipynb** file. It is easy to observe that in each section, the cell will serve a specific responsibility:

- **Import Dependencies:** Importing all required packages to train the model
- **Load Data:** Downloading Stock Price dataset in order to train the model
- **Preparing Data:** Normalize the data, which is an important stage in training machine learning model
- **Build the model:** Configuring our model with specific hyper-parameter
- **Testing the Accuracy:** Testing the accuracy of the model based on the Test Dataset
- **Make prediction:** Making the prediction for the stock price based on the trained model

3.2 Code Base Explaination

3.2.1 Import Dependencies

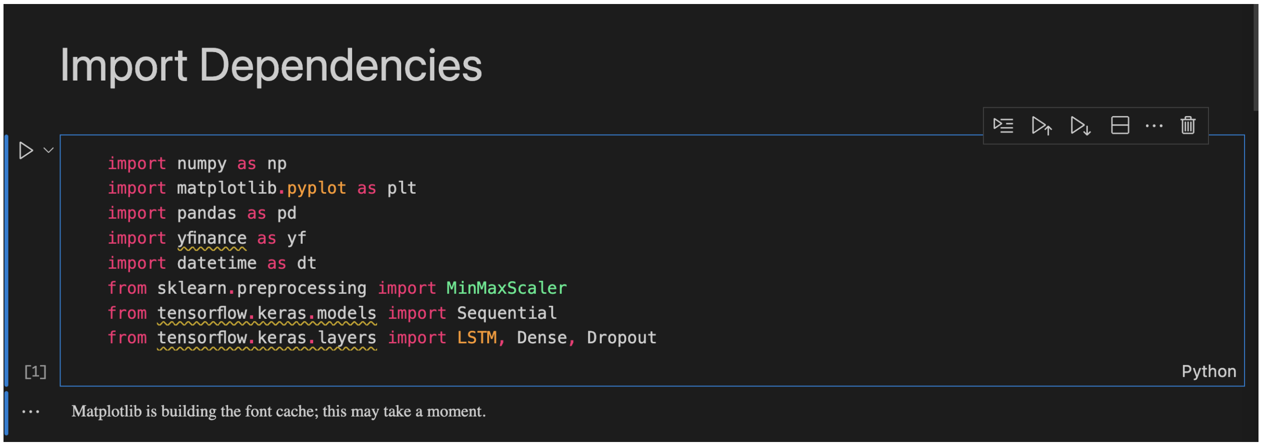


Figure 2: Importing required packages

The Figure 2 above shows the cell serving import packages responsibility. It is obvious so this part will not be discussed further.

3.2.2 Load Data

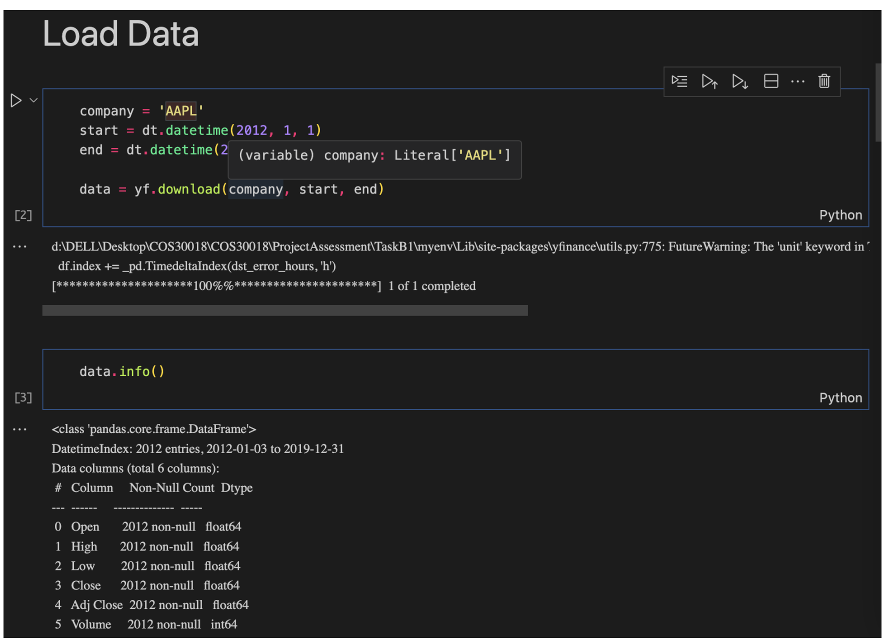
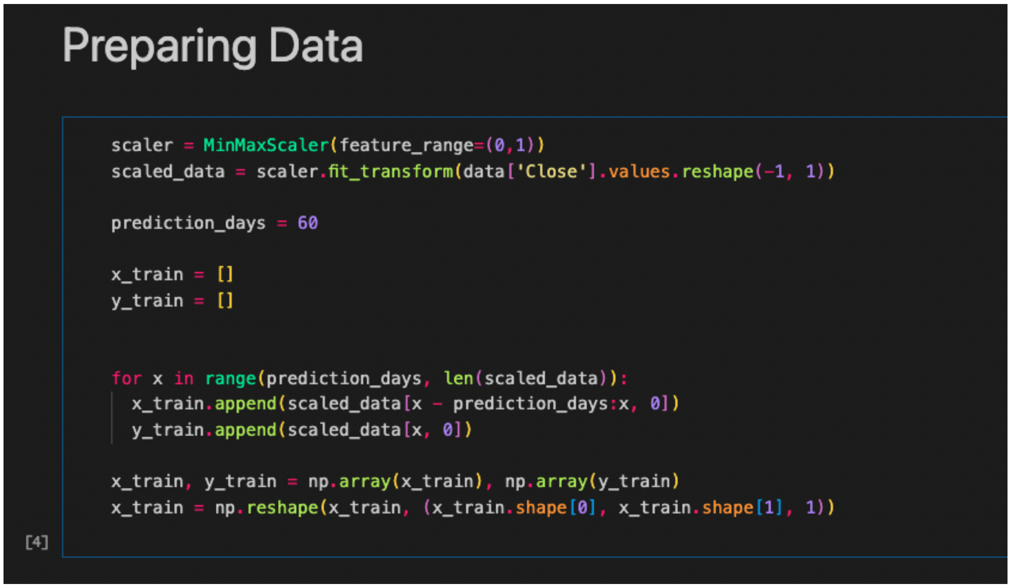


Figure 3: Loading Data for Training

The Figure 3 above shows 2 hidden cells for the Data Loading Stage. Our dataset is a Pandas Dataframe for the Stock Price of *Apple Inc.* This dataframe contains 2012 records of Apple Inc. stock price from 2012 to 2020, in which the index for each record is a date.

3.2.3 Preparing Data



Preparing Data

```python
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1, 1))

prediction_days = 60

x_train = []
y_train = []


for x in range(prediction_days, len(scaled_data)):
    x_train.append(scaled_data[x - prediction_days:x, 0])
    y_train.append(scaled_data[x, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

[4]

Figure 4: Preparing Data for Training

The **MinMaxScaler** class was imported from the famous library **sklearn** to scale our data (*Close price*) such that all records are in the range $[0, 1]$.

The `prediction_days` variable indicates the number of looking back days to perform the prediction. In Figure 4, we will observe the last 60 days to perform the *Close* price prediction for our Stock Ticket. Next, we prepare two arrays for training model: `x_train` and `y_train`. In `y_train`, the first element is the *Close Price* (of AAPL) at the 60^{th} date among 2012 records in the total dataset. On other hands, the first element in `x_train` is an array of 60 *Close Price* records before the 60^{th} date, more specifically, the first element is an array containing *Close Price* from the 1^{st} date to the 59^{th} date. You can look at Figure 5 for more details.

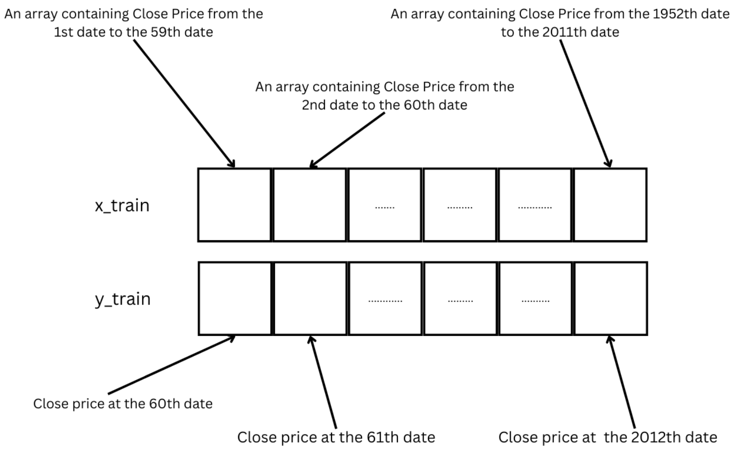


Figure 5: `x_train` and `y_train` arrays

Next, we will convert them into *np.array* for manipulation. Until now, `x_train` is matrix with a shape of $(1952, 60)$ and `y_train` is a vector with a shape of $(1952, 1)$. Then, we will reshape `x_train` into a cube, so its final shape is $(1952, 60, 1)$.

3.2.4 Build the Model

Figure 6: Model Configuration

The Figure 6 above is the configuarion of the Model, since we have to deal with Sequential Dataset, choosing the LSTM model and adding Dropout to reduce overfitting would be suitable. In order to find the perfect parameters efficiently, Mean Square Error is the metric for the loss function and Adam Optimizer Algorithm is applied. After 25 epochs of training (about 30 minutes), the Figure 7 below is our result:

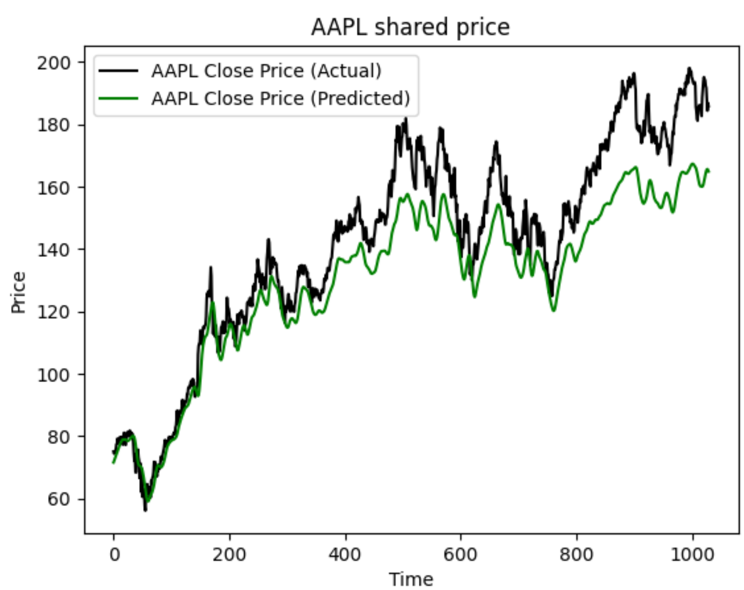


Figure 7: Result for the Training Process

4 P1

In **P1**, the model will predict the price of Amazon based on the last 15 days. Two Figures below are our attempt to test the code base.

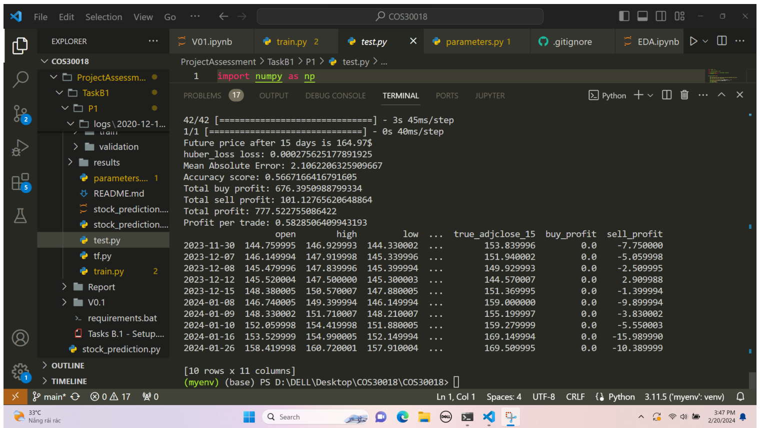


Figure 8: The Console Output when predicting the stock price

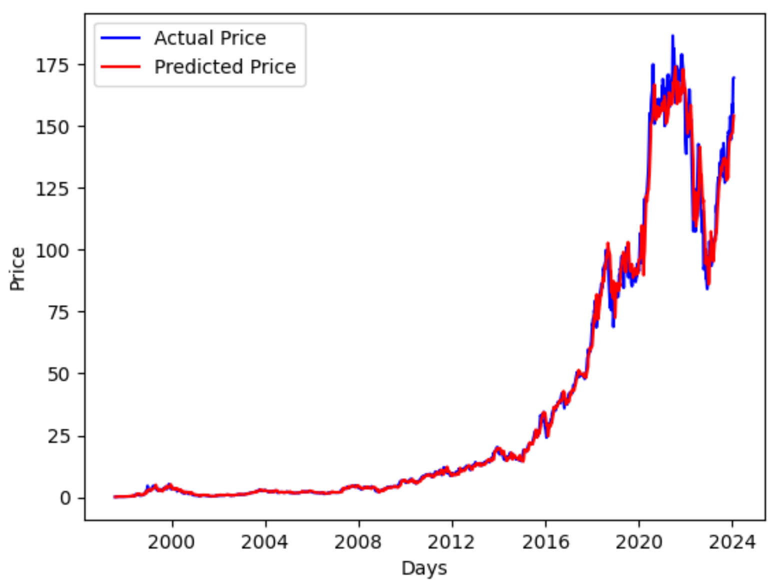


Figure 9: Stock Prediction of Amazon Ticker