Task 1: Setup

Min Khant Aung

Student ID: 103833225

—

COS30018

—

Dr Ru Jia

Introduction

In this report, I will outline the process of setting up the environment for a stock price prediction project and provide an overview of the codebases in the v0.1 and p1 directories. The goal of this project is to leverage machine learning techniques, specifically Long Short-Term Memory (LSTM) networks, to predict stock prices based on historical data.

Environment setup

To ensure reproducibility and isolate dependencies, I have set up the environment (Name: en1) using **Anaconda Navigator**.

For v0.1 and P1:

The following packages (with their version) were installed within the virtual environment to support the codebase:

- **Numpy**: For numerical computations and array manipulations. (1.24.4)
- **Matplotlib**: For plotting and visualizing stock price trends. (3.9.1.post1)
- **Pandas**: For data manipulation and analysis. (2.2.2)
- **Pandas-datareader**: To retrieve financial and economic data from various online sources into Pandas DataFrames for analysis (0.10.0)
- **TensorFlow**: For building and training the LSTM model. (2.14.0)
- **Scikit-learn**: For data preprocessing, such as scaling and splitting the dataset. (1.5.1)
- **Yfinance & yahoo_fin:** For fetching stock price data from online sources. (0.2.41) & (0.8.9.1)

Notes: Tensorflow version **2.14.0** was used in the environment because of “Issue of unrecognised "batch_input_shape" argument for LSTM” in P1 as announced by Convener. Therefore, Python version was required to drop down to **Python 3.10.14.**

The environment setup ensures that all dependencies are correctly installed, enabling seamless execution of the provided code.

[Verification of Installed Packages Command > *pip list*]

Option B – Task 1: Setup

```
(en1) C:\Users\13min\OneDrive - Swinburne Unive
n>pip list
Package                      Version
---------------------------- -----------
absl-py                      2.1.0
appdirs                      1.4.4
astunparse                   1.6.3
beautifulsoup4               4.12.3
bs4                          0.0.2
cachetools                   5.4.0
certifi                      2024.7.4
charset-normalizer           3.3.2
colorama                     0.4.6
contourpy                    1.2.1
cssselect                    1.2.0
cycler                       0.12.1
fake-useragent               1.5.1
feedparser                   6.0.11
flatbuffers                  24.3.25
fonttools                    4.53.1
frozendict                   2.4.4
gast                         0.6.0
google-auth                  2.33.0
google-auth-oauthlib         1.0.0
google-pasta                 0.2.0
grpcio                       1.65.4
h5py                         3.11.0
html5lib                     1.1
idna                         3.7
importlib_metadata           8.2.0
joblib                       1.4.2
keras                        2.14.0
kiwisolver                   1.4.5
libclang                     18.1.1
lxml                         5.2.2
lxml_html_clean              0.2.0
Markdown                     3.6
markdown-it-py               3.0.0
```

This command confirms that all required packages are installed, and their versions match the requirements.

[Checking Installed package Version Command: *python -c "import **tensorflow** as tf; print(tf.__version__)"*]

```
(en1) C:\Users\13min\OneDrive - Swinburne University\option2\stock-predictio
n>python -c "import tensorflow as tf; print(tf.__version__)
2.14.0
```

- **Output:** Indicates that TensorFlow is successfully installed.

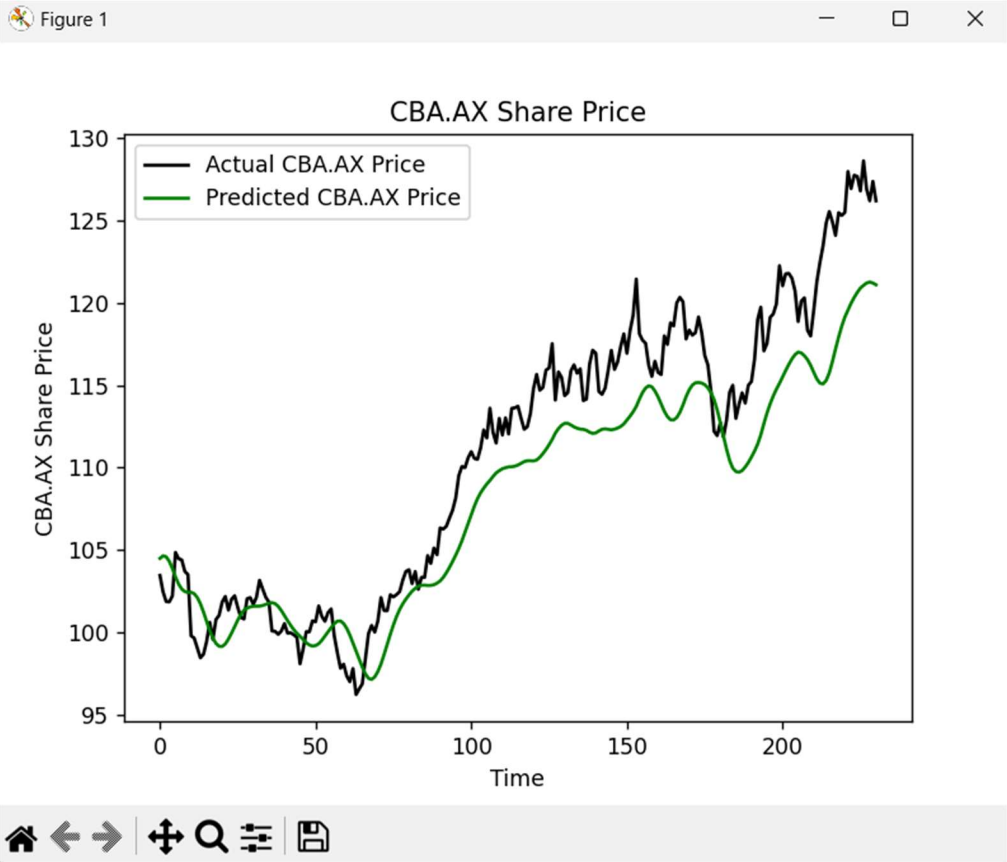
Testing Provided code bases (with Screenshots)

For v0.1:

```
Epoch 15/25
27/27 [==============================] - 1s 38ms/step - loss: 0.0073
Epoch 16/25
27/27 [==============================] - 1s 37ms/step - loss: 0.0062
Epoch 17/25
27/27 [==============================] - 1s 38ms/step - loss: 0.0055
Epoch 18/25
27/27 [==============================] - 1s 37ms/step - loss: 0.0061
Epoch 19/25
27/27 [==============================] - 1s 42ms/step - loss: 0.0069
Epoch 20/25
27/27 [==============================] - 1s 39ms/step - loss: 0.0057
Epoch 21/25
27/27 [==============================] - 1s 38ms/step - loss: 0.0063
Epoch 22/25
27/27 [==============================] - 1s 36ms/step - loss: 0.0059
Epoch 23/25
27/27 [==============================] - 1s 38ms/step - loss: 0.0051
Epoch 24/25
27/27 [==============================] - 1s 38ms/step - loss: 0.0048
Epoch 25/25
27/27 [==============================] - 1s 40ms/step - loss: 0.0051
[**********100%**********]  1 of 1 completed
8/8 [==============================] - 1s 13ms/step
1/1 [==============================] - 0s 26ms/step
Prediction: [[120.92509]]
```

1. Training in progress for v0.1

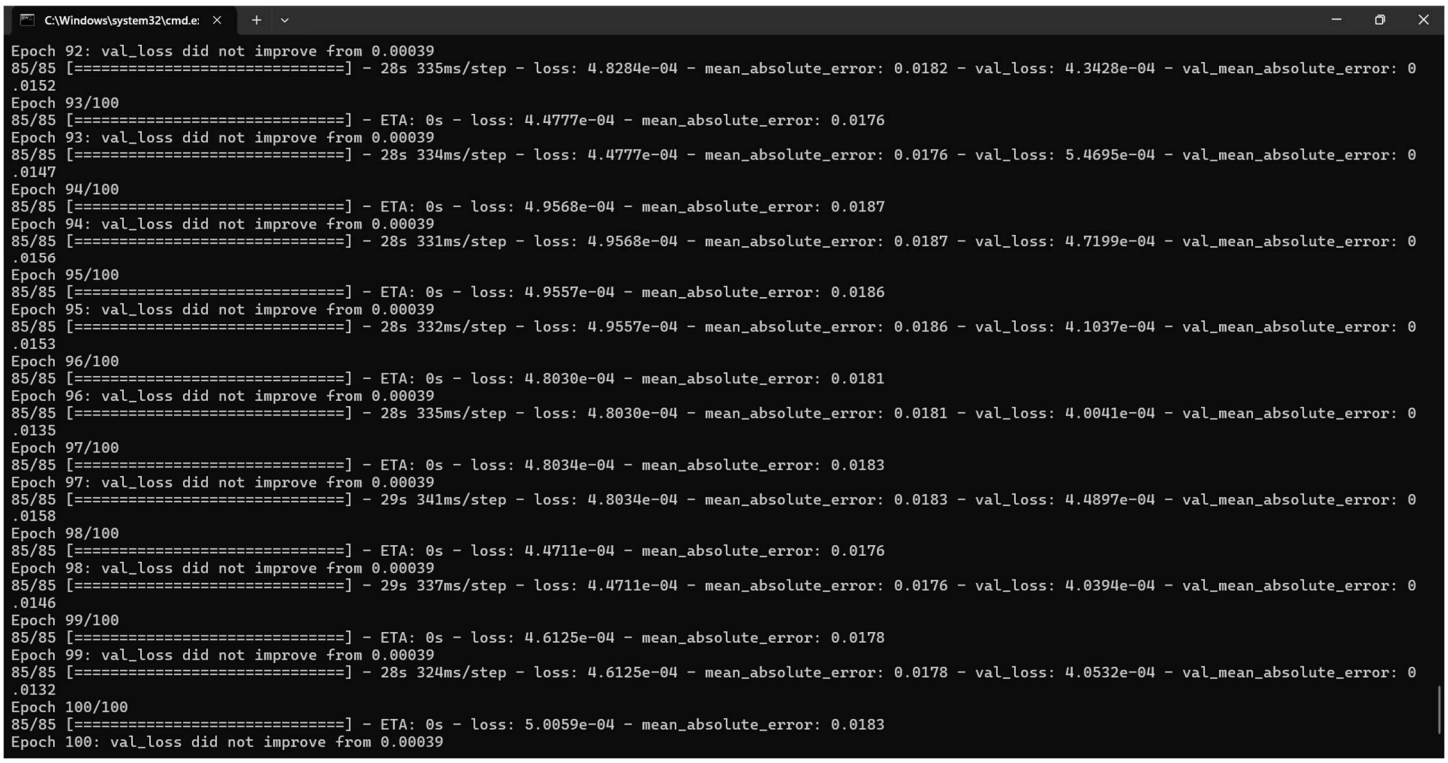
Option B – Task 1: Setup



2. Figure result

For P1:

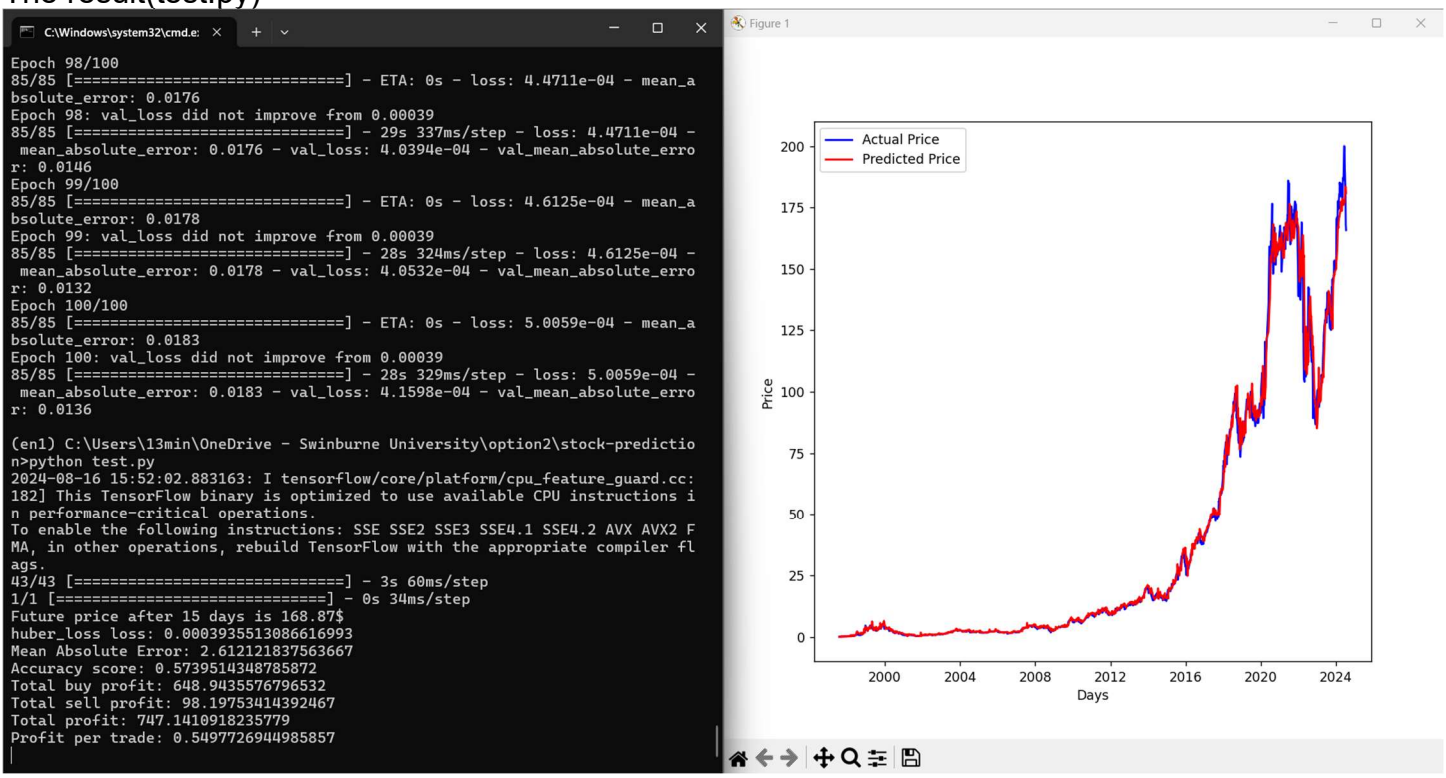
In training process (Train.py)



3. Training in progress

Option B – Task 1: Setup

The result(test.py)



4. Text result (left) and Figure result (right)

My Understanding of the Initial Code Base (v0.1)

Authors: Bao Vo and **Cheong Koo**
Dates: 14/07/2021 (v1); 19/07/2021 (v2); 02/07/2024 (v3)
Source: Adapted from a **NeuralNine** YouTube tutorial on stock price prediction.

Overview:
The version v0.1 of the code is all about predicting stock prices using an **LSTM** (Long Short-Term Memory) neural network. Below is how I see the key parts of the code:

1. **Environment Setup:**
   - The code needs several Python packages, which should be installed in a virtual environment to manage dependencies easily. These include libraries like **numpy**, **matplotlib**, **pandas**, **tensorflow**, **scikit-learn**, **pandas-datareader**, and **yfinance**.
2. **Data Loading:**
   - The script fetches stock data for a specific company (CBA.AX – the Commonwealth Bank of Australia's stock on the ASX) using the **yfinance library**.
   - It also checks if the data is already saved to avoid unnecessary downloads.
3. **Data Preparation:**
   - The data is normalized using **MinMaxScaler**, scaling the stock prices to a range of 0 to 1.
   - It then processes the data into sequences based on a set look-back period (PREDICTION_DAYS), creating features (x_train) and targets (y_train).
4. **Model Building:**
   - A Sequential model with **LSTM** layers is set up:
     1. It has **3 LSTM** layers with dropout to reduce overfitting.
     2. The model ends with a Dense layer to predict the next closing price.
   - The model uses the Adam optimizer and mean squared error as the loss function and is trained over **25** epochs with a batch size of **32**.
5. **Model Testing and Evaluation:**
   - The model is tested using data prepared similarly to the training data.
   - Predictions are made and compared against actual prices, and these results are visualized with a plot.
6. **Future Predictions:**
   - The model predicts the stock price for the next day using the latest data available.
   - There's a note suggesting that the prediction quality could be better and recommends exploring other methods or models for improved accuracy.

Code Analysis:

- **Data Handling:**
  - Data is fetched from **Yahoo Finance**, scaled, and reshaped for **LSTM input**.
  - Training data is created by slicing the scaled data into sequences.
- **Model Architecture:**
  - The model consists of three **LSTM layers** with dropout to help avoid overfitting.
  - These layers are followed by a Dense layer that produces the final prediction.
  - The architecture is designed to capture temporal patterns in stock price data.
- **Performance:**
  - The model's performance is measured by comparing its predictions to actual prices and visualizing the results.
  - There's a mention of the model's limitations and some suggestions for improving accuracy, like trying more advanced techniques or mixing different approaches.

Conclusion:
This initial code base (v0.1) lays the groundwork for predicting stock prices using **LSTM networks**. It covers the basics: data loading, preparation, model building, training, and evaluation. However, it seems the model might not be performing at its best, so there's more room for experimenting with different techniques and optimizations for the maximum accuracy. That's exactly what I plan to do for this project.