**` COS30018**

**Intelligent Systems**

**Option B: Stock Prediction**

**Task B.1 – Set Up**

**Name: Duc Thuan Tran**

**Student ID: 104330455**

**Tutor: Dr. Ru Jia**

**Tutorial: Friday 2:30 – 4:30**

Table of Contents

[Introduction 3](#_Toc174381236)

[Environment Setup 3](#_Toc174381237)

[Codebase Testing 5](#_Toc174381238)

[Testing v0.1 5](#_Toc174381239)

[Testing P1 6](#_Toc174381240)

[Codebase Overview 10](#_Toc174381241)

[GitHub Repository Setup 10](#_Toc174381242)

# **Introduction**

The objective of Task B.1 is to set up a development environment for a stock prediction project based on an initial code base (v0.1) and an additional reference project (P1) from GitHub. This task involves downloading and testing these code bases, setting up a virtual environment to manage dependencies, and documenting the process. By exploring and understanding these projects, the goal is to identify potential improvements and prepare for future development work on the stock prediction model. This report details the steps taken to set up the environment, test the code, and understand the initial code base, providing a foundation for further development and refinement.

# **Environment Setup**

#### **Virtual Environment Setup in PyCharm**

To ensure a clean and organized workspace, a virtual environment was set up using PyCharm, which is widely regarded as one of the best Integrated Development Environments (IDEs) for Python development. PyCharm offers extensive support for Python, including code completion, debugging, and seamless integration with virtual environments, making it an ideal choice for managing Python projects efficiently.

The following steps outline the process of setting up the environment using PyCharm:

1. **Creating a New Project in PyCharm:**
   * Open PyCharm and select **"New Project"** from the welcome screen.
   * Choose a location for the project and make sure the option **"New environment using: Virtualenv"** is selected.
   * Select the base interpreter (e.g., Python 3.x) that will be used for this environment.
   * Click **"Create"** to initialize the new project and create the virtual environment.

A screenshot of a computer

Description automatically generated

Figure 1. PyCharm Create Project

1. **Installing Dependencies:**

* Once the project is created, open the terminal within PyCharm or use the **"Python Packages"** tool window.
* Install the required libraries by either:

Running the following command in the terminal:

*pip install -r requirements.txt*

Or manually adding each package using the **"Python Packages"** tool window:

+ Search for each required package (e.g., TensorFlow, NumPy, Pandas, etc.).

+ Click **"Install"** to add them to the environment.

A screen shot of a computer

Description automatically generated

Figure 2. Install requirement

A screenshot of a computer

Description automatically generated

Figure 3. Python interpreter (for downloading specific library)

Choosing PyCharm for this setup not only simplifies the process but also provides powerful tools for code analysis, version control, and project management, all within a single interface. This makes PyCharm a preferred choice for Python development, particularly when working with complex projects like machine learning and data analysis.

# Codebase Testing

The testing of the provided code bases (v0.1 and P1) was essential to ensure that the stock prediction models were set up correctly and could run successfully within the configured environment. The following steps outline the process and results of testing these code bases.

## Testing v0.1

**Setup and Initialization:**

* The v0.1 code base was downloaded and placed in the project directory created in PyCharm.
* The environment was activated, and all necessary dependencies were installed using the requirements.txt file.

**Running the Code:**

* The stock-prediction.py script was executed within PyCharm using the terminal:

*python stock-prediction.py*

* During the initial run, several issues were encountered, such as missing dependencies and deprecated functions. These were resolved by updating the code and installing the required packages.

**Model Training:**

* After resolving the issues, the model was trained on the provided dataset. The training process was monitored, and the model successfully completed the specified number of epochs.

A screenshot of a computer

Description automatically generated

Figure 4. Training process (25 Epochs)

**Results:**

* Upon completion of the training, the model's performance was evaluated using the test dataset.
* A plot was generated to compare the actual vs. predicted stock prices, demonstrating that the model could reasonably predict future prices based on historical data.
* **Screenshots:** The following screenshots illustrate the successful execution and results of the v0.1 code base:

A graph with lines and numbers on it

Description automatically generated

Figure 5. Stock prediction results

## Testing P1

**Setup and Initialization:**

* The P1 code base from GitHub was cloned into the PyCharm project directory.
* The virtual environment was used to install the dependencies specified in the requirements.txt file provided with P1.

**Challenges and Adjustments:**

* **Version Compatibility:** Since the P1 code base was developed a few years ago, several dependencies in the requirements.txt file were outdated or incompatible with the latest versions of Python and related libraries. To resolve this:

The “**requirements.txt**” file was updated to include the correct and compatible versions of all required libraries.

A screenshot of a computer

Description automatically generated

Figure 6. Requirements.txt for P1

* **Requests\_HTML Issue:** Although **requests\_html** was included in the code base, an issue persisted where the functionality requiring **requests\_html** was not working as expected.

A screenshot of a computer program

Description automatically generated

Figure 7. Request \_html error lxml.html error

The solution involved:

* + Explicitly importing **requests\_html** at the beginning of the script.
  + Installing the **lxml\_html\_clean** library using the terminal command:

*pip install lxml\_html\_clean*

* + After these adjustments, the script ran without issues.
* **Code Adjustments:**
  + Several parts of the t**rain.py** and **test.py** scripts needed modification to work with the updated libraries and Python versions.
  + The adjustments allowed the training to proceed for 100 epochs without interruption and ensured compatibility with the latest version of TensorFlow.

A screen shot of a computer

Description automatically generated

Figure 8. Training process (100 Epochs)

**Running the Code:**

* With the environment set up and the necessary adjustments made, the **stock\_prediction.py** script was executed:

*python stock\_prediction.py*

* The model was successfully initialized and began training on the provided dataset.

**Model Training and Evaluation:**

* The model trained for 100 epochs, after which the test.py script was executed to evaluate the model's performance.
* The predictions were visualized through plots, showing a close alignment between actual and predicted stock prices.

**Results:**

* The P1 code base, after adjustments, provided accurate stock prediction results. The results were displayed using Matplotlib plots, like those generated by the v0.1 code base, but with improved accuracy.

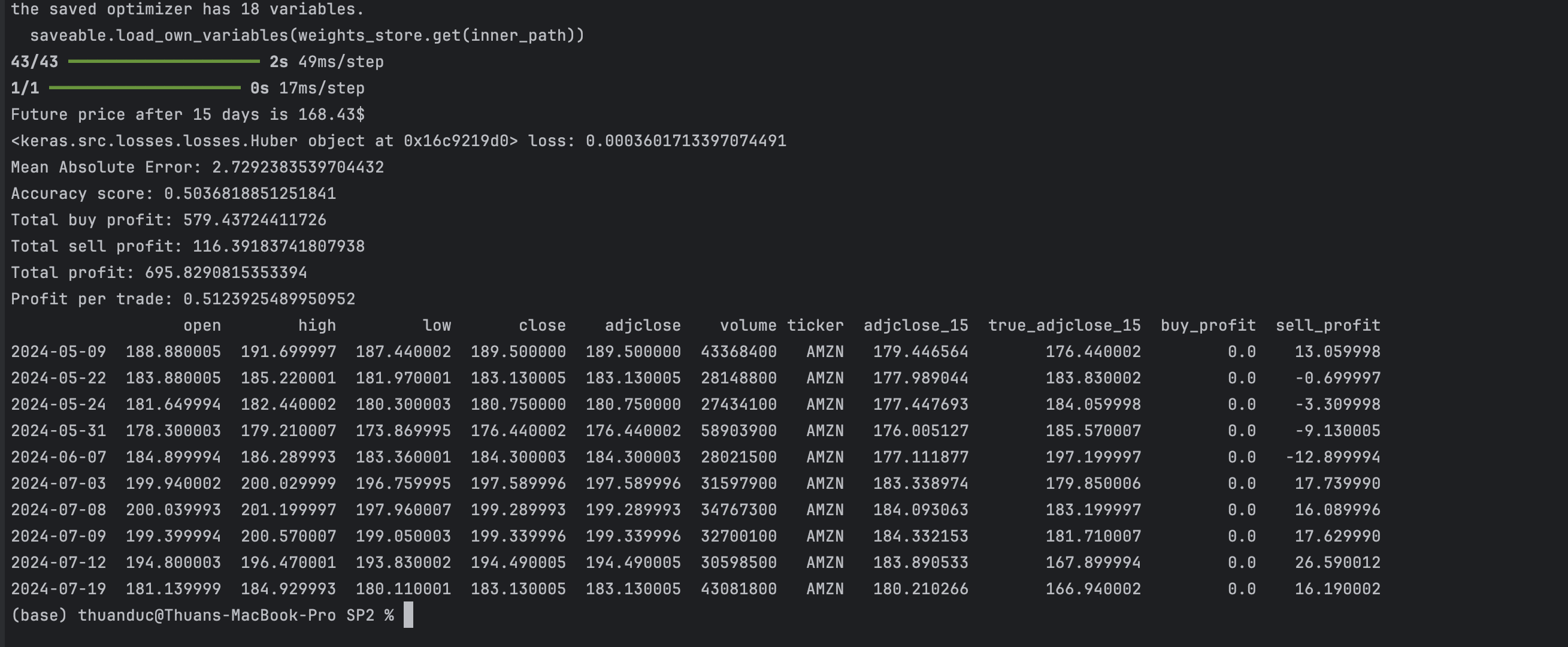


Figure 9. test.py output

A graph with red and blue lines

Description automatically generated

Figure 10. Stock prediction results

# Codebase Overview

#### **Summary**

The v0.1 code base is a Python script designed to predict future stock prices using a Long Short-Term Memory (LSTM) neural network, a type of recurrent neural network (RNN) suitable for time series forecasting.

#### **Key Functions**

The v0.1 code base consists of several essential functions:

* **Data Loading:** Retrieves historical stock data using yfinance, with options to save or load the data locally.
* **Data Preparation:** Scales the stock prices and creates sequences of data for training the LSTM model.
* **Model Building:** Constructs an LSTM model with multiple layers, designed to predict future stock prices. The model is compiled with the Adam optimizer and Mean Squared Error (MSE) as the loss function.
* **Training:** Trains the model on the prepared data, adjusting the weights over multiple epochs to minimize prediction error.
* **Testing and Evaluation:** Predicts future stock prices on test data and compares them with actual prices. The results are plotted to visualize model performance.
* **Visualization:** Plots actual vs. predicted prices, providing a clear view of the model's effectiveness.

# GitHub Repository Setup

The project's code base is hosted on GitHub for version control and review. Everyone can access the repository via the following link: [GitHub Repository](https://github.com/ThuanDanchoi/COS30018/tree/main). This repository contains all necessary files for the project and is available for the tutor to review the work.

A screenshot of a computer

Description automatically generated

Figure 11. GitHub source

# 