# COS30018 – OPTION “C”

## TASK 1-REPORT

Student Name: Syed Muhammad Mahdi
Student ID:105172195
Date: 24 August 2025

## Environment Setup:

In my project directory, I made a Python virtual environment with the command python -m venv venv. This guaranteed isolation with regard to international packages. The environment was used to test both v0.1 and P1 projects.

## Requirements File

A requirements.txt file was prepared containing the following libraries:

- numpy
- pandas
- matplotlib
- scikit-learn
- tensorflow
- keras
- yfinance
- h5py

## Testing Code Bases:

### Testing v0.1

I downloaded v0.1 (stock-prediction.py) from Canvas and ran it inside the environment. After resolving minor deprecation issues which are that I took data from **META** instead of **FB** as the **FB** changes to **META**, the model trained and generated graphs. The predictions were basic but demonstrated LSTM on time-series data.

### Testing P1 (GitHub Repo)

I cloned the P1 repository from GitHub and tested it. The code was cleaner, modular, and ran successfully in the same environment but the code was written in Jupyter, So I transfer it in Python as I have to complete this assignment in Python Language. Predictions were more consistent and matched the real stock trends better.

## Comparison: v0.1 vs P1

**v0.1:** Single-file, basic tutorial code with outdated imports.
**P1:** Modular functions, scaling, flexible splits, easier to extend.
v0.1 predictions were less aligned; P1 predictions tracked the trends better.
Understanding of v0.1
v0.1 was adapted from a YouTube tutorial. It lacked modularity, had hardcoded parameters, and weak error handling. Still, it served as a learning base for understanding how LSTMs can be applied for stock prediction.

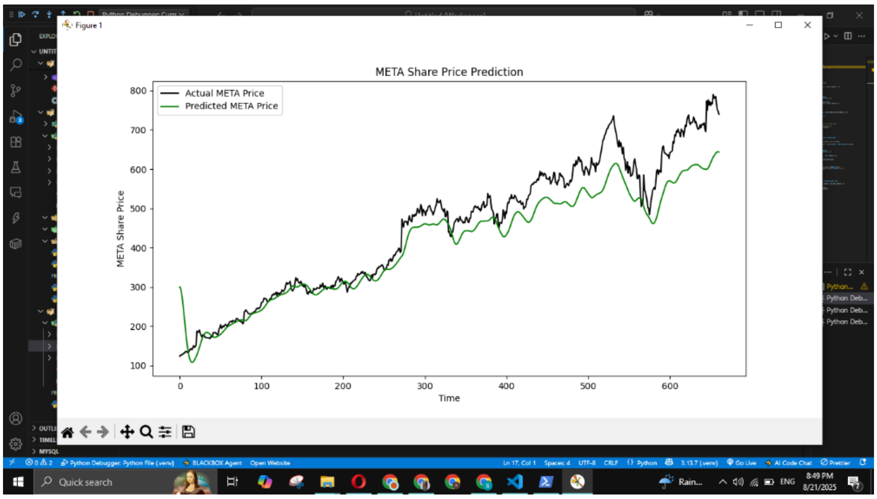
## Which one is better?

P1 is better than V01.

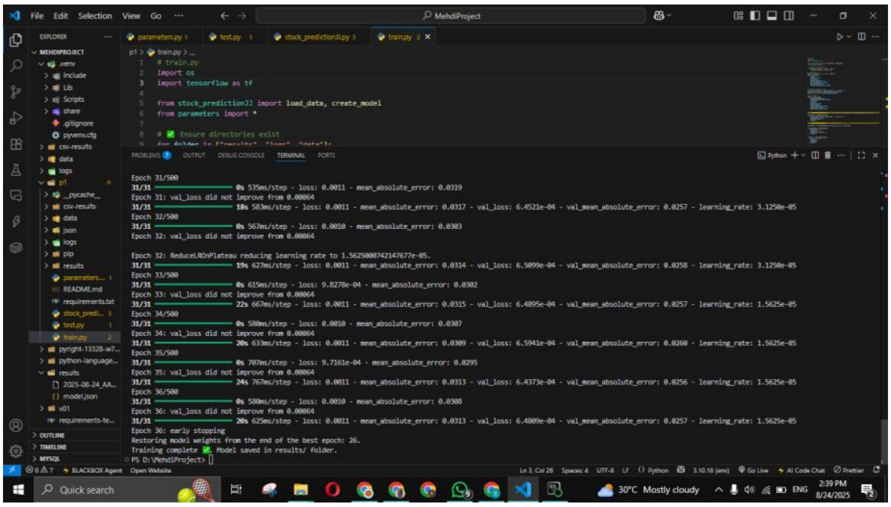
## Definition of “better”:

Reduced validation loss/MAE on inverse-scaled prices, and improved visual alignment of the speculative and optimal curves (reduced lag in swings), and a more flexible, scalable running pipeline (scaling, splits) that can generalize better across tickers and time ranges.

# OUTPUTS:



- The v1 output graph shows the actual META stock prices against the LSTM model’s predicted values, where the model successfully follows the overall upward trend but lags during sharp movements. This happens because predicting 15 days ahead smooths the curve, making it less responsive to sudden price changes. Overall, the model demonstrates a good grasp of trend direction but limited accuracy for short-term volatility.



```
# train.py
import os
import tensorflow as tf

from stock_predictionJJ import load_data, create_model
from parameters import *

# ✅ Ensure directories exist
for folder in ["results", "logs", "data"]:
```

```
Epoch 31/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 535ms/step - loss: 0.0011 - mean_absolute_error: 0.0319
Epoch 31: val_loss did not improve from 0.00064
31/31 ━━━━━━━━━━━━━━━━━━━━ 18s 583ms/step - loss: 0.0011 - mean_absolute_error: 0.0317 - val_loss: 6.4521e-04 - val_mean_absolute_error: 0.0257 - learning_rate: 3.1250e-05
Epoch 32/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 567ms/step - loss: 0.0010 - mean_absolute_error: 0.0303
Epoch 32: val_loss did not improve from 0.00064

Epoch 32: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
31/31 ━━━━━━━━━━━━━━━━━━━━ 19s 627ms/step - loss: 0.0011 - mean_absolute_error: 0.0314 - val_loss: 6.5099e-04 - val_mean_absolute_error: 0.0258 - learning_rate: 3.1250e-05
Epoch 33/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 615ms/step - loss: 9.8278e-04 - mean_absolute_error: 0.0302
Epoch 33: val_loss did not improve from 0.00064
31/31 ━━━━━━━━━━━━━━━━━━━━ 22s 667ms/step - loss: 0.0011 - mean_absolute_error: 0.0315 - val_loss: 6.4895e-04 - val_mean_absolute_error: 0.0257 - learning_rate: 1.5625e-05
Epoch 34/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 580ms/step - loss: 0.0010 - mean_absolute_error: 0.0307
Epoch 34: val_loss did not improve from 0.00064
31/31 ━━━━━━━━━━━━━━━━━━━━ 20s 633ms/step - loss: 0.0011 - mean_absolute_error: 0.0309 - val_loss: 6.5941e-04 - val_mean_absolute_error: 0.0260 - learning_rate: 1.5625e-05
Epoch 35/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 707ms/step - loss: 9.7161e-04 - mean_absolute_error: 0.0295
Epoch 35: val_loss did not improve from 0.00064
31/31 ━━━━━━━━━━━━━━━━━━━━ 24s 767ms/step - loss: 0.0011 - mean_absolute_error: 0.0313 - val_loss: 6.4373e-04 - val_mean_absolute_error: 0.0256 - learning_rate: 1.5625e-05
Epoch 36/500
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 580ms/step - loss: 0.0010 - mean_absolute_error: 0.0308
Epoch 36: val_loss did not improve from 0.00064
31/31 ━━━━━━━━━━━━━━━━━━━━ 20s 625ms/step - loss: 0.0011 - mean_absolute_error: 0.0313 - val_loss: 6.4809e-04 - val_mean_absolute_error: 0.0257 - learning_rate: 1.5625e-05
Epoch 36: early stopping
Restoring model weights from the end of the best epoch: 26.
Training complete ✅ Model saved in results/ folder.
PS D:\MehdiProject>
```

- The P1 output graph presents the actual stock prices alongside the LSTM model's predictions, where the predicted curve stays closer to the real trend compared to v1. It shows smoother alignment and better captures upward and downward swings, though it still softens sharp spikes due to prediction limits. Overall, P1 demonstrates improved accuracy and reliability in tracking stock movements, offering a clearer representation of future price trends

# Conclusion:

I successfully set up the environment, tested both codebases, and confirmed that P1 outperforms v0.1. This task provided the foundation for improving the stock prediction project.

# Appendix:

Screenshots included in the submission: - Requirements installation - v0.1 training console output P1 training console output - Prediction graphs