Project B
Task 1 - Setup

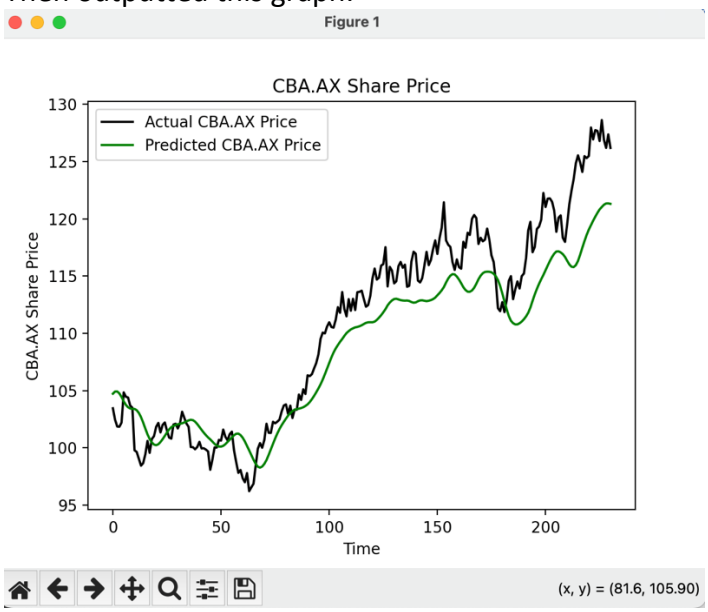
Created Anaconda venv to help manage dependencies.
Installed packages from github requirements.txt file. When trying to run the Swinburne stock_prediction.py, I was met with errors about missing packages. To resolve these errors, I made these amendments to my requirements file.

Changed sklearn to scikit-learn
Added requests_html
Add panda-datareader

I then ran the Swinburne stock-prediction.py again and received this output.
The model trained for 25 epochs:

```
Epoch 17/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0062
Epoch 18/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0061
Epoch 19/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0065
Epoch 20/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0065
Epoch 21/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0060
Epoch 22/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0059
Epoch 23/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0056
Epoch 24/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0048
Epoch 25/25
27/27 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0057
[**********100%%**********]  1 of 1 completed
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
Prediction: [[121.20436]]
(base) lukevalentino@192-168-1-103 ~ %
```

Then outputted this graph:



I then tried running the jupyter notebook from the Github repository, however I was met with many issues. The issues and their solutions are detailed below.

Issues in Github Jupyter Notebook:

Under the method “create_model”:

```
def create_model(sequence_length, n_features, units=256, cell=LSTM, n_layers=2, dropout=0.3,
                 loss="mean_absolute_error", optimizer="rmsprop", bidirectional=False):
    model = Sequential()
    for i in range(n_layers):
        if i == 0:
            # first layer
            if bidirectional:
                model.add(Bidirectional(cell(units, return_sequences=True), batch_input_shape=(None,
sequence_length, n_features)))
            else:
                model.add(cell(units, return_sequences=True, batch_input_shape=(None, sequence_length,
n_features)))
```

Needs to change to:

```
def create_model(sequence_length, n_features, units=256, cell=LSTM, n_layers=2, dropout=0.3,
                 loss="mean_absolute_error", optimizer="rmsprop", bidirectional=False):
    model = Sequential()
    for i in range(n_layers):
        if i == 0:
            # first layer
            if bidirectional:
                model.add(Bidirectional(cell(units, return_sequences=True), input_shape=(sequence_length,
n_features)))
            else:
                model.add(cell(units, return_sequences=True, input_shape=(sequence_length, n_features)))
```

In the block that trains the model here:

```
# some tensorflow callbacks
checkpointer = ModelCheckpoint(os.path.join("results", model_name + ".h5"), save_weights_only=True,
save_best_only=True, verbose=1)
tensorboard = TensorBoard(log_dir=os.path.join("logs", model_name))
```

```
# train the model and save the weights whenever we see
# a new optimal model using ModelCheckpoint
history = model.fit(data["X_train"], data["y_train"],
                    batch_size=BATCH_SIZE,
                    epochs=EPOCHS,
                    validation_data=(data["X_test"], data["y_test"]),
                    callbacks=[checkpointer, tensorboard],
                    verbose=1)
```

The file extension in checkpointer needs to be changed to this:

```
checkpointer = ModelCheckpoint(os.path.join("results", model_name + ".weights.h5"), save_weights_only=True, save_best_only=True, verbose=1)
```

In the code block where the hyperparameters are set, the “LOSS” variable needs to be changed to match the variable in create_model.

From:

```
LOSS = "huber_loss"
```

To:

```
LOSS = "mae"
```

Changing it the other way (changing create_model to match hyperparameters as opposed to changing hyperparameters to match create_model) leads to errors.

After training when reloading the model for evaluation, the model path must be amended:

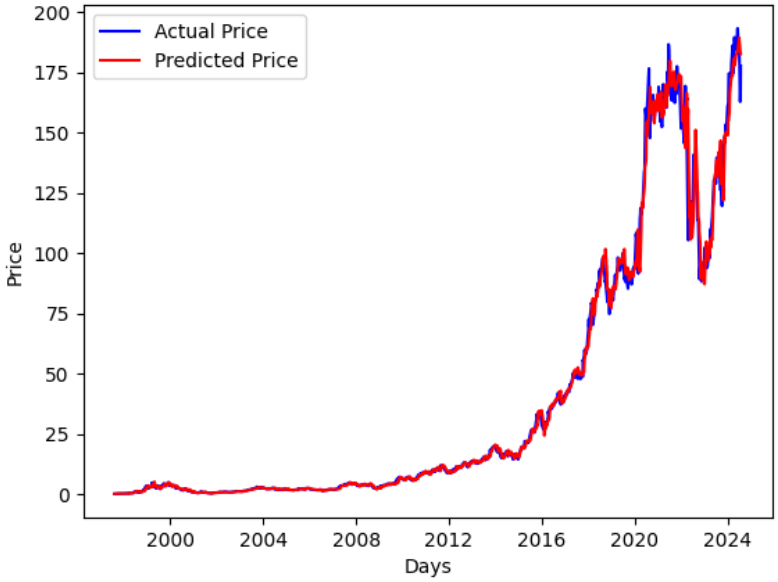
From:

```
# load optimal model weights from results folder
model_path = os.path.join("results", model_name) + ".h5"
model.load_weights(model_path)
```

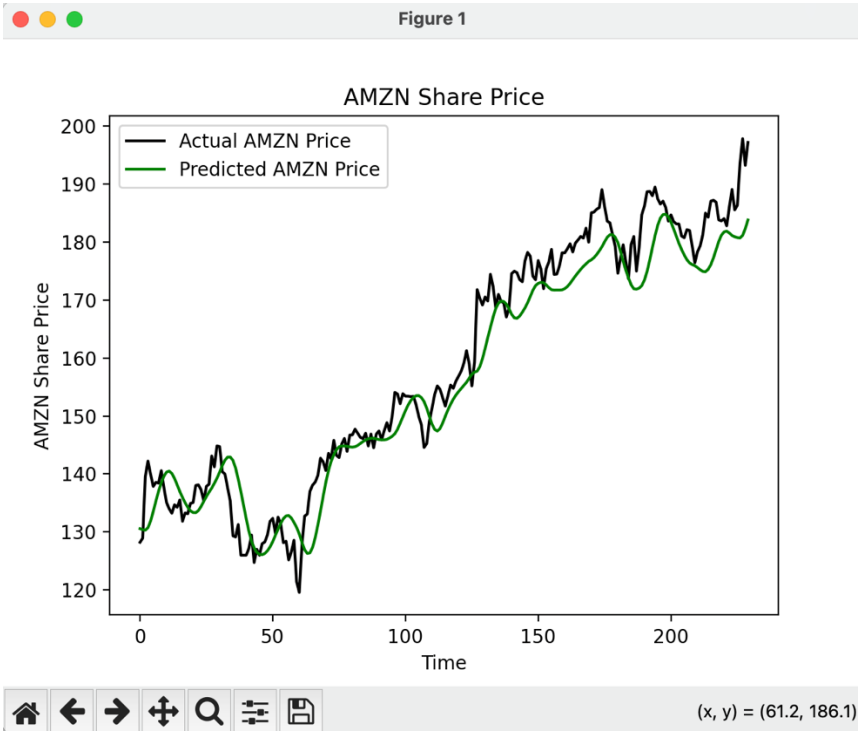
To:

```
# load optimal model weights from results folder
model_path = os.path.join("results", model_name) + ".weights.h5"
model.load_weights(model_path)
```

The notebook now works, and I received this outputted graph (along with other data points)



As a comparison I changed the swinbrune stock_prediction.py to analyse the Amazon stock as well.



As can be seen, the github repository's attempt at stock price prediction was considerably more accurate.
I believe this is mainly due to the epochs each was trained for. The swinburne model was trained for 25 epochs whereas the github repository model was trained for 500 epochs.

However there were also many other differences in the models hyperparameters. For example, both models are LSTM but use different amounts of units, 50 for Swinburne and

256 for Github Repository.

The models also use different loss functions. Swinburne uses mean squared error while github repository uses mean absolute error.

Another difference in training to go along with the epochs is the difference in batch size. The batch size when training the Github repository model is 64 compared to 32 for Swinburne.