**Task Report Cos30018 Option B**

**B.4: Machine Processing 1**

**Name: Le Bao Nguyen**

**Student Id: 104169837**

1. Importing libraries (From the task B.2):

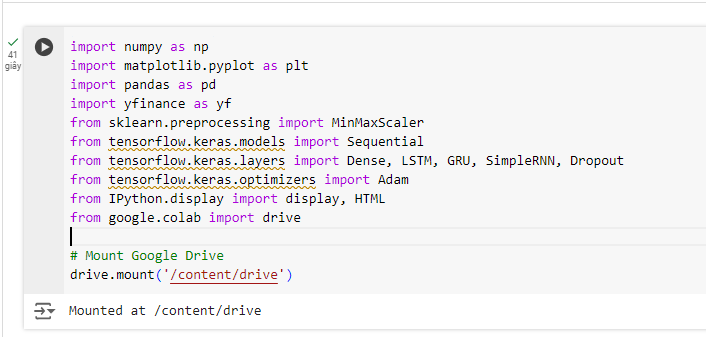


Figure 1: Importing libraries to run the code.

* The script import previous libraries and new libraries:
* "Sequential": A linear stack of layers in Keras.
* "Dense": A fully connected neural network layer.
* "LSTM", "GRU", "SimpleRNN": Various kinds of recurrent neural network layers applied to forecast time series.
* "Dropout": A regularization technique to prevent overfitting.
* "Adam": An optimization technique that uses training data to iteratively update the network weights.

1. Data loading and processing (From the task B.2):

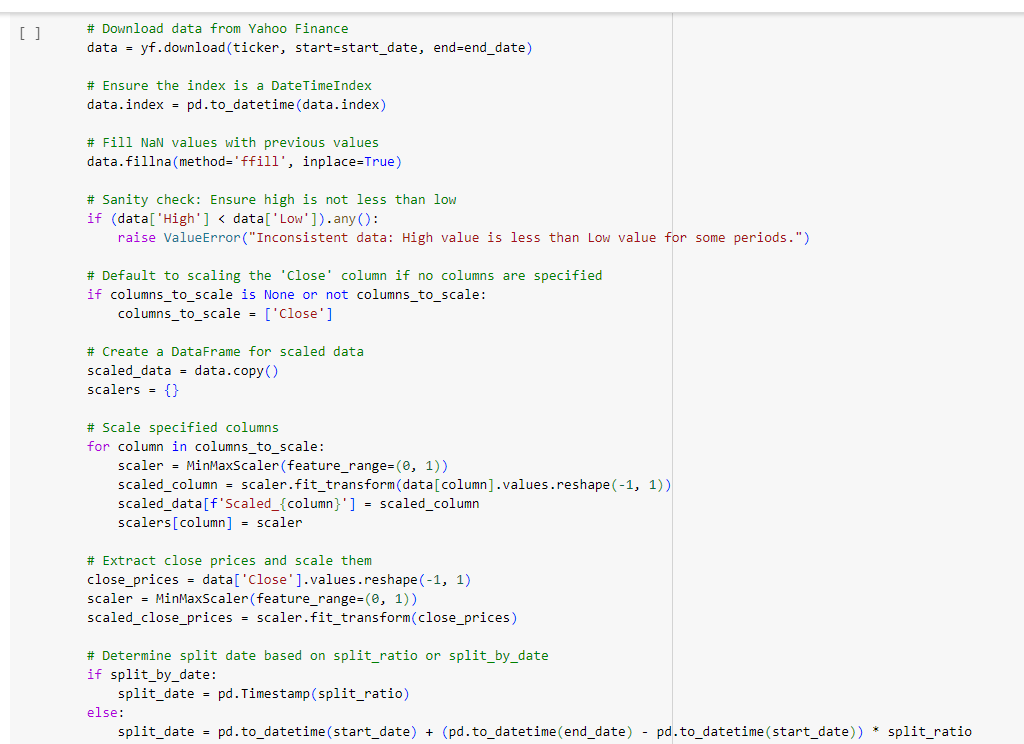


Figure 2: Loading and processing data (1).

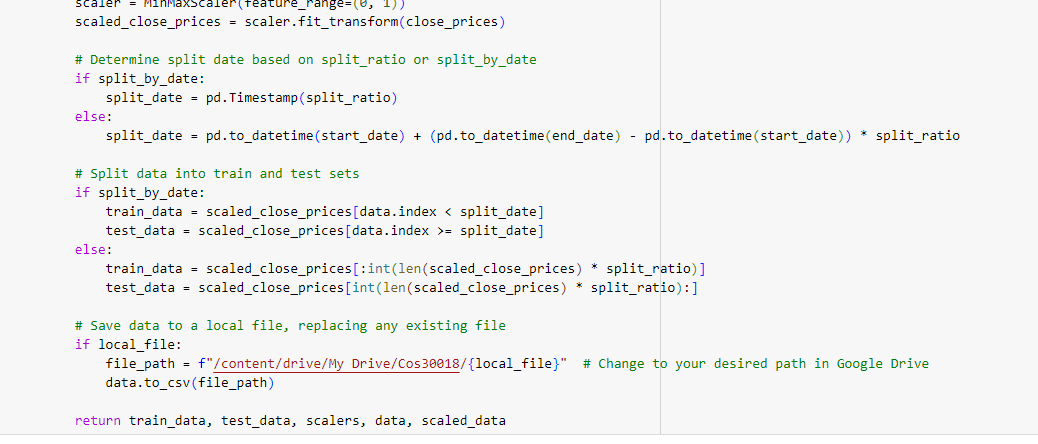


Figure 3: Loading and processing data (2).

* We still use the same data loading and processing just like B.2.

1. Displaying data in a custom table (From the task B.2):

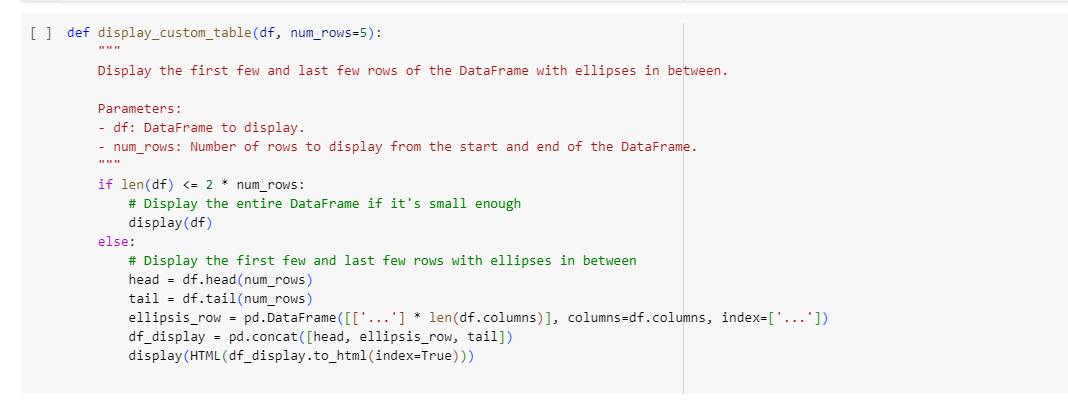


Figure 4: Displaying the data from csv file.

* We still use the same displaying data function just like B.2.

1. Model Creation:



Figure 5: Code to create the model.

* We go over the dictionaries in the "layers\_config" list, which define the type, units, activation function, and other parameters for every layer.
* We add each layer to the model after identifying if it is an LSTM, GRU, or RNN. An "input\_shape" that specifies the form of the input data is needed for the first layer.
* When stacking RNN layers, the "return\_sequences" argument is used. By doing this, the layer makes sure to return the entire output sequence rather than just the end result.
* The addition of dropout layers eliminates overfitting. A Dropout layer is created after the current RNN layer if "dropout\_rate" is larger than 0.
* The predicted stock price is output by the Dense layer, which is the last layer and has a single unit. The mean squared error (MSE) loss function and Adam optimizer are used in the construction of the model, making it appropriate for regression applications like stock price prediction.

1. Experimentation with Different Configurations:

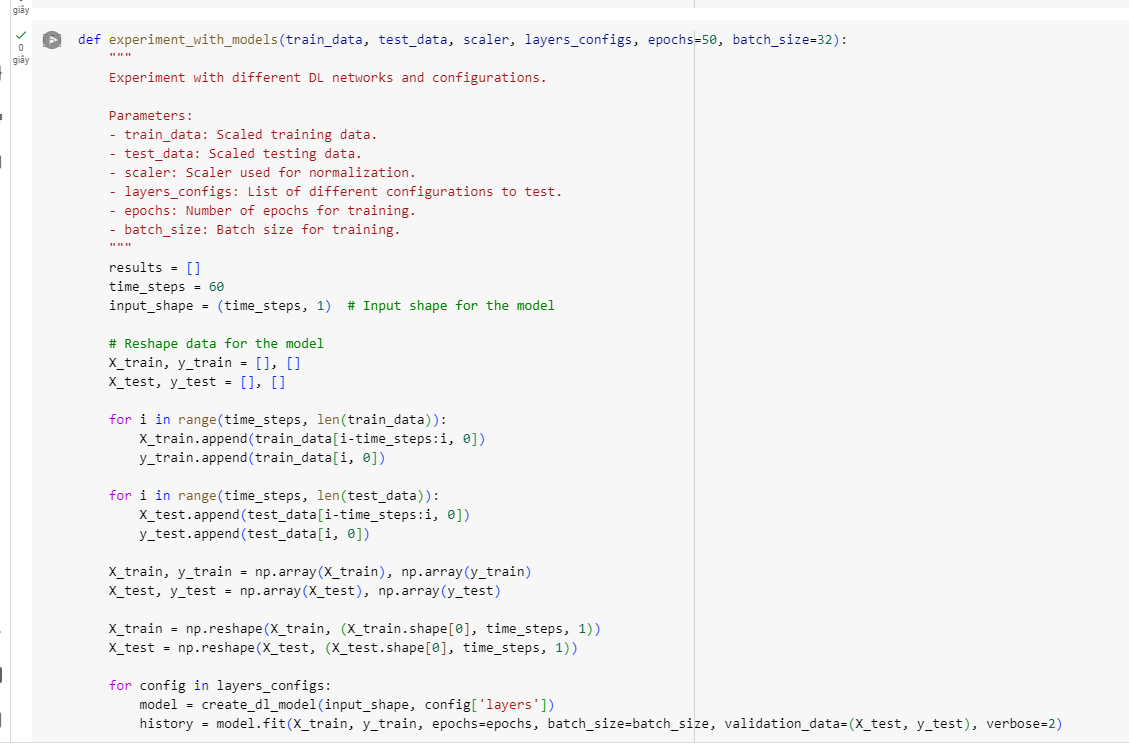


Figure 6: Code to experiment with model (1).

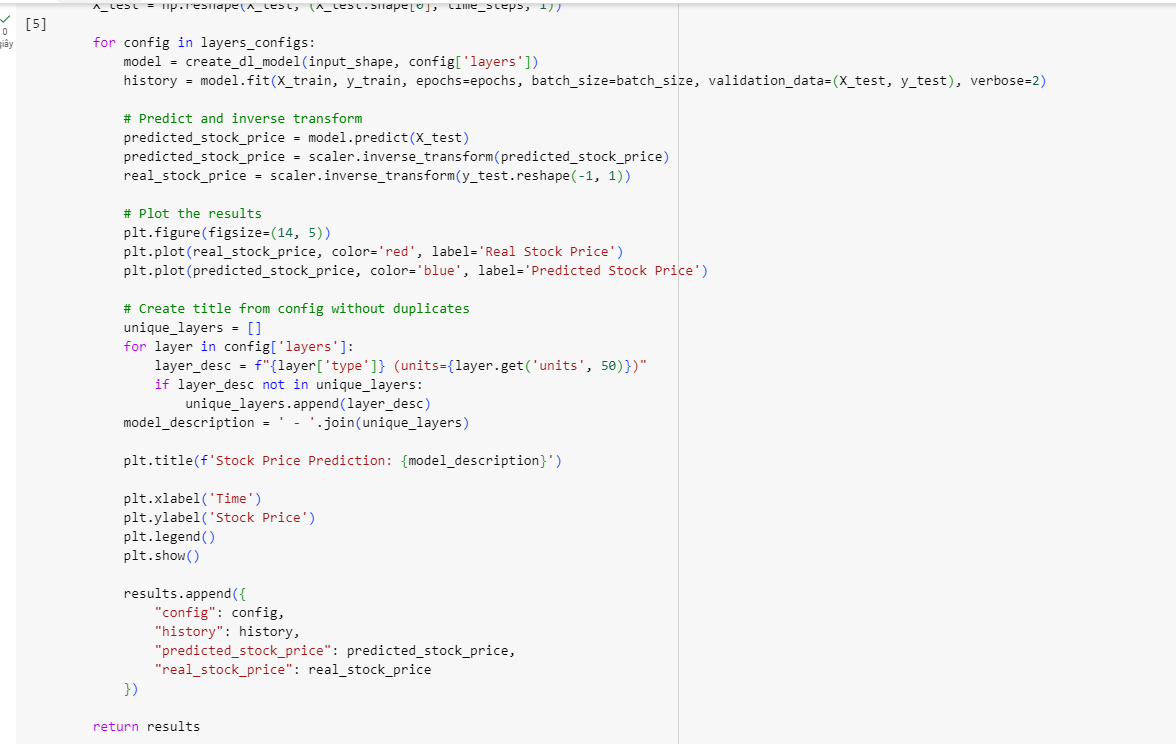


Figure 7: Code to experiment with model (2).

* We define "time\_steps" as 60, which means each input sequence will contain 60 previous data points.
* We construct sliding windows of 60 data points for each time step in the training and testing data ("X\_train" and "X\_test"), with the next data point serving as the desired output ("y\_train" and "y\_test").
* The input data is reshaped to fit the required input shape for the RNN models.
* We use "create\_dl\_model" to make a model for each configuration in "layers\_configs" and then train it with the training set. To train the model, we define the batch size and number of epochs using the "fit" method.
* Using test data, we apply the model to forecast stock prices once it has been trained. The scaler is then used to inversely translate these predictions back to their original scale.
* We plot the real and predicted stock prices to visually evaluate the model's performance. The unique layer configurations are used as the title for each plot to differentiate between the models.
* A list including the model configuration, training history, and predictions is created as a result and kept for additional examination.

1. Main script run:



Figure 8: The script to run the code and the prediction.

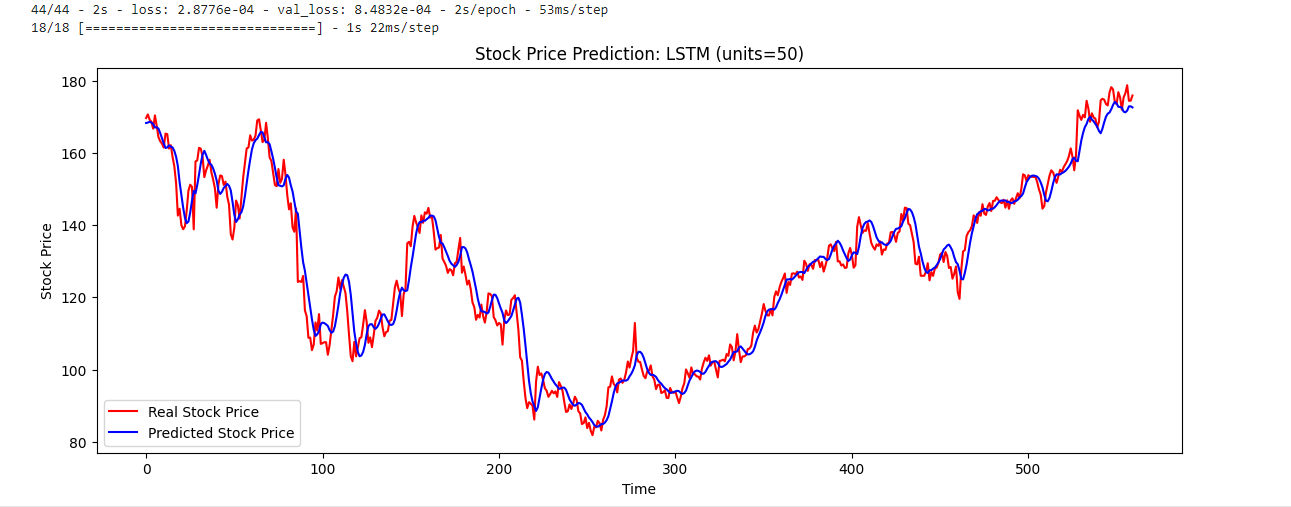
* LSTM Configuration: Two LSTM layers with 50 units each, the first one returning sequences.
* GRU Configuration: Two GRU layers with 50 units each, the first one returning sequences.
* RNN Configuration: Two SimpleRNN layers with 50 units each, the first one returning sequences.
* The line calls the "experiment\_with\_models" function to train and evaluate the defined DL models on the training and testing data.
* 

Figure 9: The predicted chart if we use LTSM layer type.

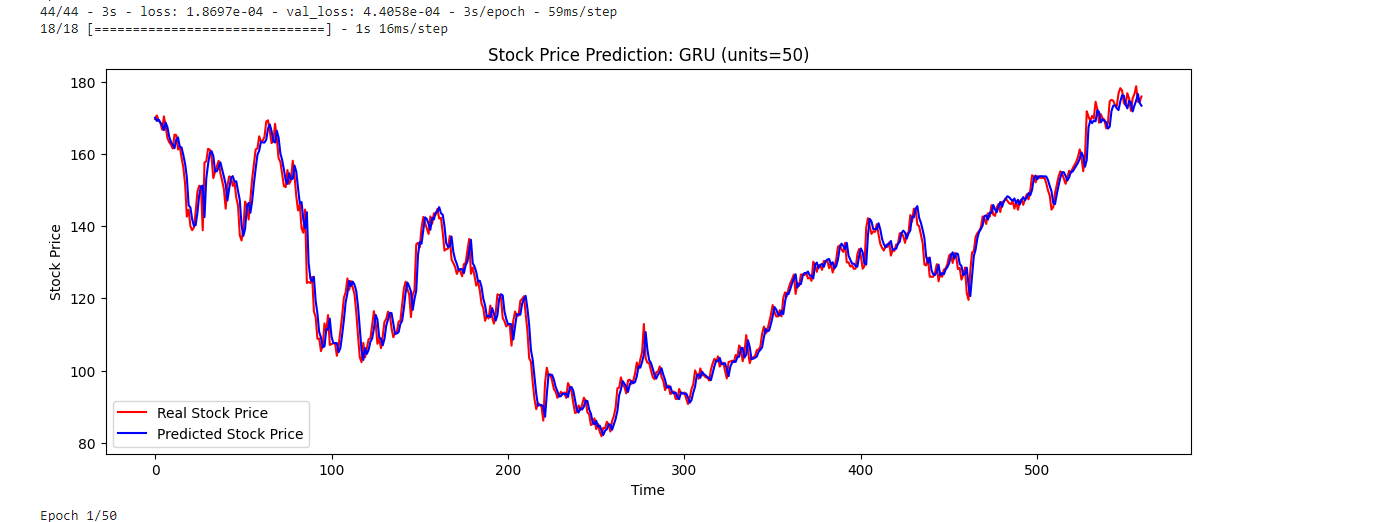


Figure 10: The predicted chart if we use GRU layer type.

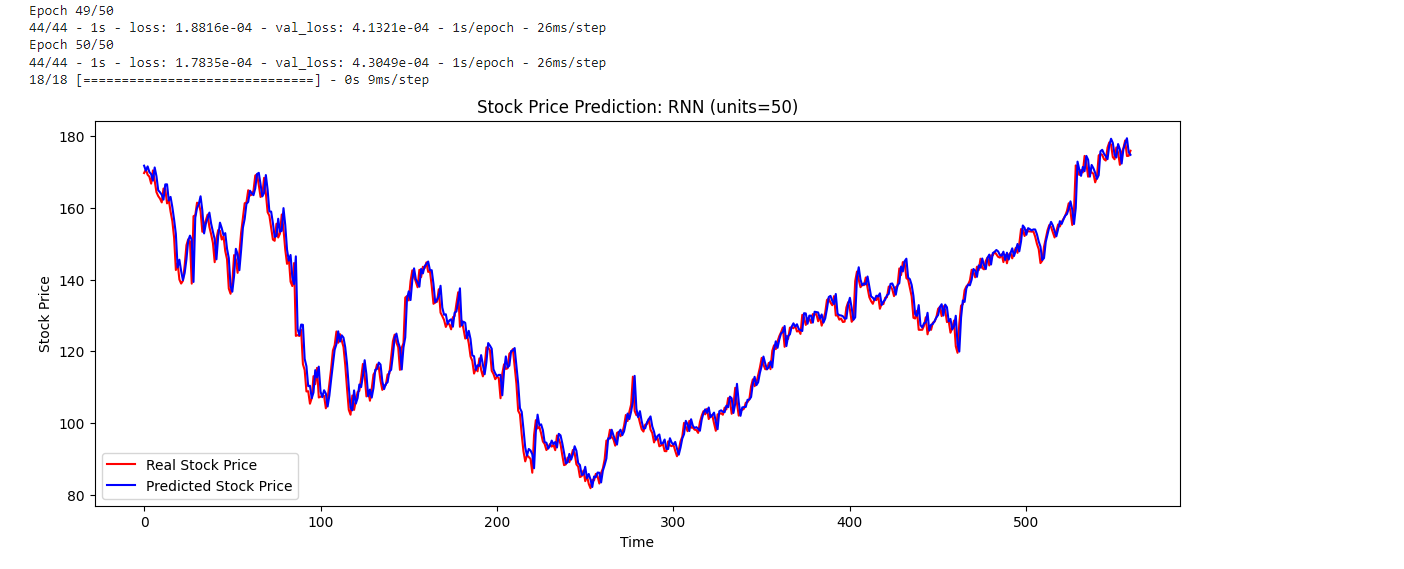


Figure 11: The predicted chart if we use RNN layer type.

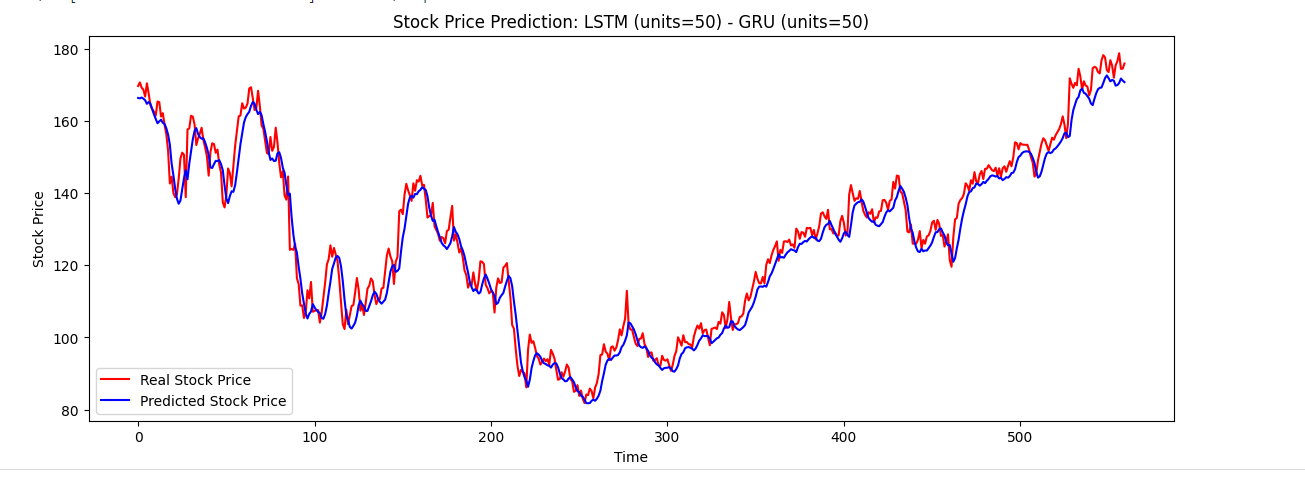


Figure 12: The predicted chart if we use LTSM and GRU layer types.

1. References:

SAYAH, F. (2023). *Stock Market Analysis + Prediction using LSTM*. <https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm>

OZTURK, O. (2020). *Stock Price prediction by simple RNN and LSTM*. <https://www.kaggle.com/code/ozkanozturk/stock-price-prediction-by-simple-rnn-and-lstm>