**Title:** Task B-4 Report

**Author:** Robin Findlay-Marks, s103603871

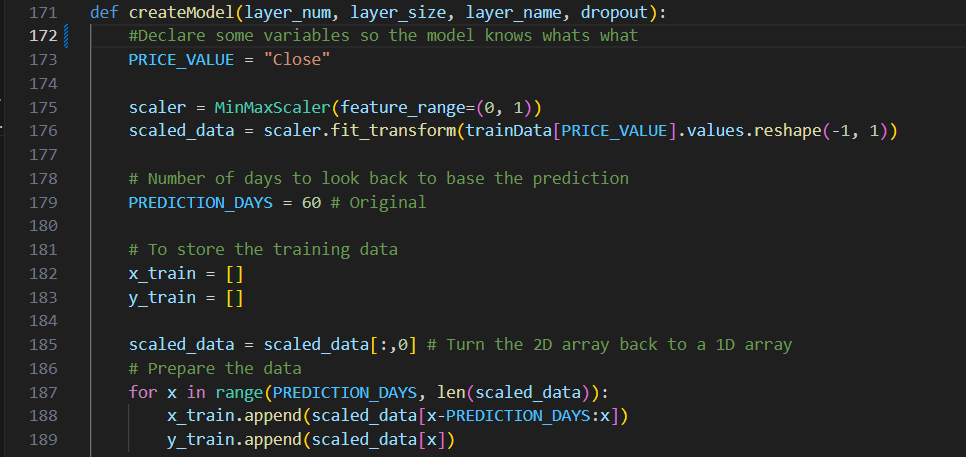
**Task information:**

Subtask 1:

This subtask was surprisingly easy as I could use P1 as a guide for how to write this function. The first thing I did was separate the model creation code into it’s own function. After a little bit of tweaking to make sure the it still worked I moved onto adding the function parameters. For this I used the ones listed in the assignment task; number of layers (layer\_num), layer size (layer\_size), deep learning network type (layer\_name) as well as dropout rate (dropout).

There are only a few real differences from the original code. The first one of course is that parameters from the parameters.py file are being used, rather than the hard coded ones that were previously present. Another difference is that the layers are now being generated in a for each loop, dependent on the parameters.py and can now handle as many new layers as requested (even if that may take a lot more time to process), rather than being again, hard coded.

The code for the new function can be found in the following images:



A screen shot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

Subtask 2:

I believe this subtask to just a further implementation of the first subtask by doing two things. The first is to get the program to run with different DL networks, such as LSTM, RNN and GRU. The program already runs LSTM, so no need to do anything there, the only things left to do is run RNN and GRU. The general structure of the model creation already supports this, especially since I replaced the LSTM with a parameter from the parameter.py file. All I had to do was import GRU and SimpleRNN as can be seen here.

The next step is to add them into the parameters so they can be changed. I did this in parameters.py and my work can be seen here

A screenshot of a computer program

Description automatically generated

The second part of subtask 2 was I believe to make several hyperparameter configurations (like sets of previously declared parameters) which use different combinations of the parameters for the the DL network. To do this I made a simple switch which checks the HYPERPARAMETER variable in the parameters.py file and uses this to decide which parameter set to use.

For this I made 5 different hyperparameters. The first is the default LSTM, which is using the parameters that were used in v0.1, the second and third are the same but using SimpleRNN and GRU respectively, the fourth is LSTM but with the parameters from the P1 example code, and the final is my custom version, which uses the combination of parameters that I have done tests with to find the best one I can.

The code for the switch and hyperparameters can be seen below

A screen shot of a computer

Description automatically generated

Below can be seen the graph results for the different setting presets

Default LSTM settings: Default RNN settings:

A graph showing price and price

Description automatically generated with medium confidence A graph showing the price of a stock market

Description automatically generated

Default GRU settings: P1 Settings:

A graph showing the price of a stock market

Description automatically generatedA graph showing price and price

Description automatically generated with medium confidence

Robin’s custom settings:

A graph showing the price of a stock market

Description automatically generated