COS30018 - Option B - Task 5: Machine Learning 2

Cody Cronin-Sporys

103610020

# Multi-Step:

The multi-step functionality allows for prediction to be made *k* days into the future. The code uses the variable LOOKUP\_DAYS, which gets passed into the load\_data() function as a parameter that sets the lookup\_step variable.





Inside the load\_data() function we use the lookup\_step variable to set how far the dataframe column should be shifted when setting the future array of the dataframe.

A screen shot of a computer code

Description automatically generated

We also use to to determine how many items are selected from the tail end when setting the last sequence values.



This last\_sequence array is what we use to predict the future values and is a combination of the window size (n\_steps) sequence and the lookup\_step sequence.

A screen shot of a computer code

Description automatically generated

Once the last\_sequence array has been setup we add it to the result so it’s returned from the function.



Going back to the stock\_prediction code, we can find the lasts\_sequence array being used in the to make a future predict based on the created model.

A screen shot of a computer code

Description automatically generated

The first line here takes the last\_sequence array and removes the n\_step sequence, leaving just the future prediction days. Then the array dimensions are expanded and we make a prediction based on the model. The prediction is then scaled based on the scaler set earlier in the code. The final line prints both the *k* days value and the final prediction as a 2 decimal float.

# Multi-Variate:

The simple multi-variate problem is solved in this code with the FEATURE\_COLUMNS variable. This variable sets all the columns we want to use to make the predictions with, and is first passed into the load\_data() function as a parameter.





The feature columns variable is used in multiple places in the load\_data() function. It is first used to check if the passed columns actually exist in the downloaded dataframe. Then it’s used to setup a scaler for each column.

A computer screen shot of text

Description automatically generated

A screen shot of a computer

Description automatically generated

Importantly for our multi-variate problem though, the feature columns variable is used when setting up both the last\_sequence and sequence\_data arrays. Since the last\_sequence code was explained with the multi-step problem there isn’t a need to explain it again, the feature columns is just used to make sure the each of the desired columns is accounted for when adding values to the array. For the sequence\_data array, the feature column is used to increment the for loop and add values to the sequences deque object (which is a list-like sequence).

A computer screen shot of a code

Description automatically generated

This sequence\_data and sequences variables are then used to set the X and Y arrays, which are then split into X\_train, y\_train, X\_test and y\_test arrays. These test and train arrays are returned through the result variable.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

This is important, because back in the stock\_prediction file, these arrays are used when fitting the model and making predictions.

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

The feature column variable itself is also used as a parameter for the model\_creation() function to make sure the model we generate is suitable for the desired number of columns.



# Summary:

In summary we use the FEATURE\_COLUMNS variable to setup the last\_sequence and sequence data variables in the load\_data() function. These variables are then passed back and used to fit the model through the x\_train and y\_train arrays, predict using the model through the x\_test array, and used to predict future prices through the last\_sequnce array.

The multi-step and multi-variate functionality both work simultaneously, so we can make predictions *k* days into the future while also including multiple feature\_columns to base our predictions on.

Much of this code was implemented earlier in the unit, and my experiments with implementing and fixing the code were included in the task 3 report. I’ll include them below:

Firstly, I needed to fix an issue that had occurred due to the change in code from last weeks task. The model was not being built due to the input shape being incorrect. This was because the x\_train array now stores 5 values per line instead of 1, and was fixed in the code here:



Next the test dataset was not being created correctly due to a key issue when concatenating the test and training data. I tried changing the total dataset to fix this but it created another issue with the scaler not having a transform function because it is a dict. This was fixed by making adjusting the scaler to use the adjclose feature column.

A screen shot of a computer code

Description automatically generated



This then caused another issue. Due to the fix with the x\_train data and the model shape, there was now a mismatch between test data and model expected shape. The model expected 5 feature columns but the current test data process only uses close. To fix this I removed total\_dataset[PRICE\_VALUE] line from the code. This caused other issues so I looked through the p1 code and realised that most of the code in v0.1 was unnecessary, so I commented it out and replaced it with the x\_test and y\_test datasets which fixed all the problems after some scaling and inverse scaling