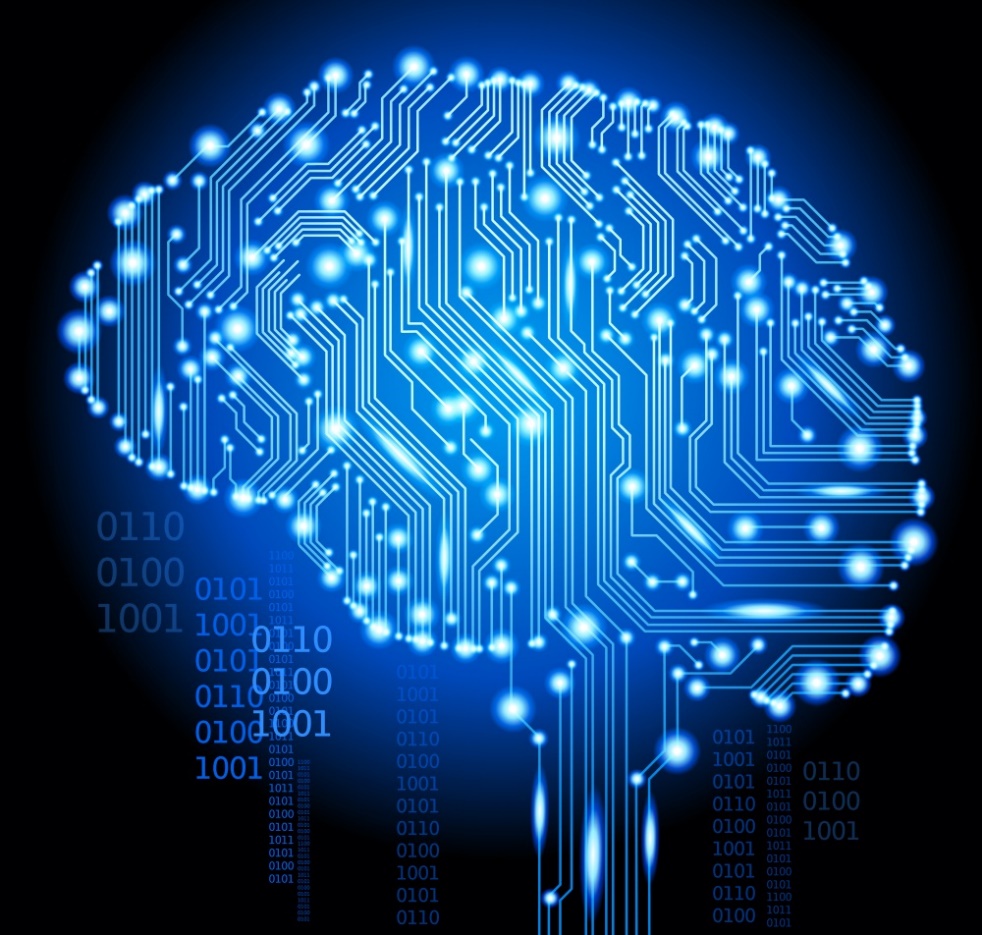


COS30018 – Intelligent Systems

**REPORT**

TASK B5 – MACHINE LEARNING 2



**TABLE OF CONTENTS**

*Table of contents…………………………………………………………………………………………………………………2*

*I. Change the method of processing data…………………………………………………………………………….3*

*II. Multi-variate problem……………………………………………………………………………………………………..4*

*1. “trainAndTestMultivariateDLModel” method…………………………………………………………4*

*2. Testing…………………………………………………………………………………………………………………..4*

*III. Multi-step problem………………………………………………………………………………………………………..6*

*1. “trainAndTestMultistepDLModel” method…………………………………………………………….6*

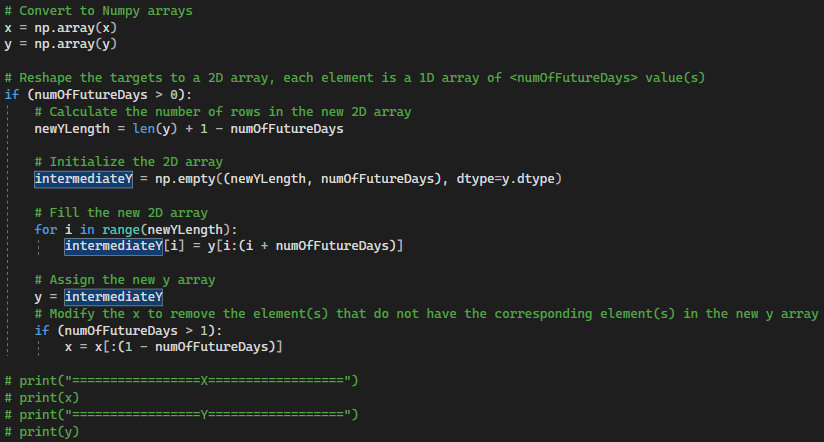
*2. Testing………………………………………………………………………………………………………………….9*

*IV. Summary…………………………………………………………………………………………………………………….11*

*V. References……………………………………………………………………………………………………………………*

**CHANGE THE PROCESSING DATA METHOD**

Besides a additional parameters of “numOfFutureDays” (k), I made some changes to the “processData()” method, before the splitting step:

****

The main idea for these new lines is reshaping the “y” (targets), from a 1D array to a 2D one, each of its element is a 1D array comtaining the “numOfFutureDays” consecutive values from the original array. E.g.

Given: numOfFutureDays = 3, y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Output: y = [ [0, 1, 2], [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6], [5, 6, 7], [6, 7, 8], [7, 8, 9] ]

Because of that process, the length of the new “y” array would decrease by “1 – numOfFutureDays”, if “numOfFutureDays” is greater than 1, so I also modified the “x” array to match that length, by removing the last “1 – numOfFutureDays” elements.

In addition, the parameter “predictionDays” is now named “numOfPastDays” for improving the readability of my codebase.

**MULTIVARAITE PREDICTION**

**1. “trainAndTestMultivariateDLModel” method**

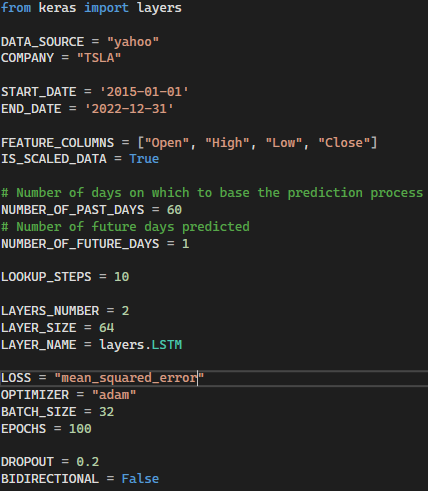
In the previous version of “v0.4”, I had already implemented most of the requirements for the multivariate prediction issue, as my model using multiple related columns (specified in the “FEATURE\_COLUMNS”, which is Open, High, Low, Close by default) to predict the future Close stock price. For the method of “trainAndTestMultivariateDLModel()”, I only made some modifications in the “yActualData” descaling process to match with the changes in the “dataProcessing.py” file:



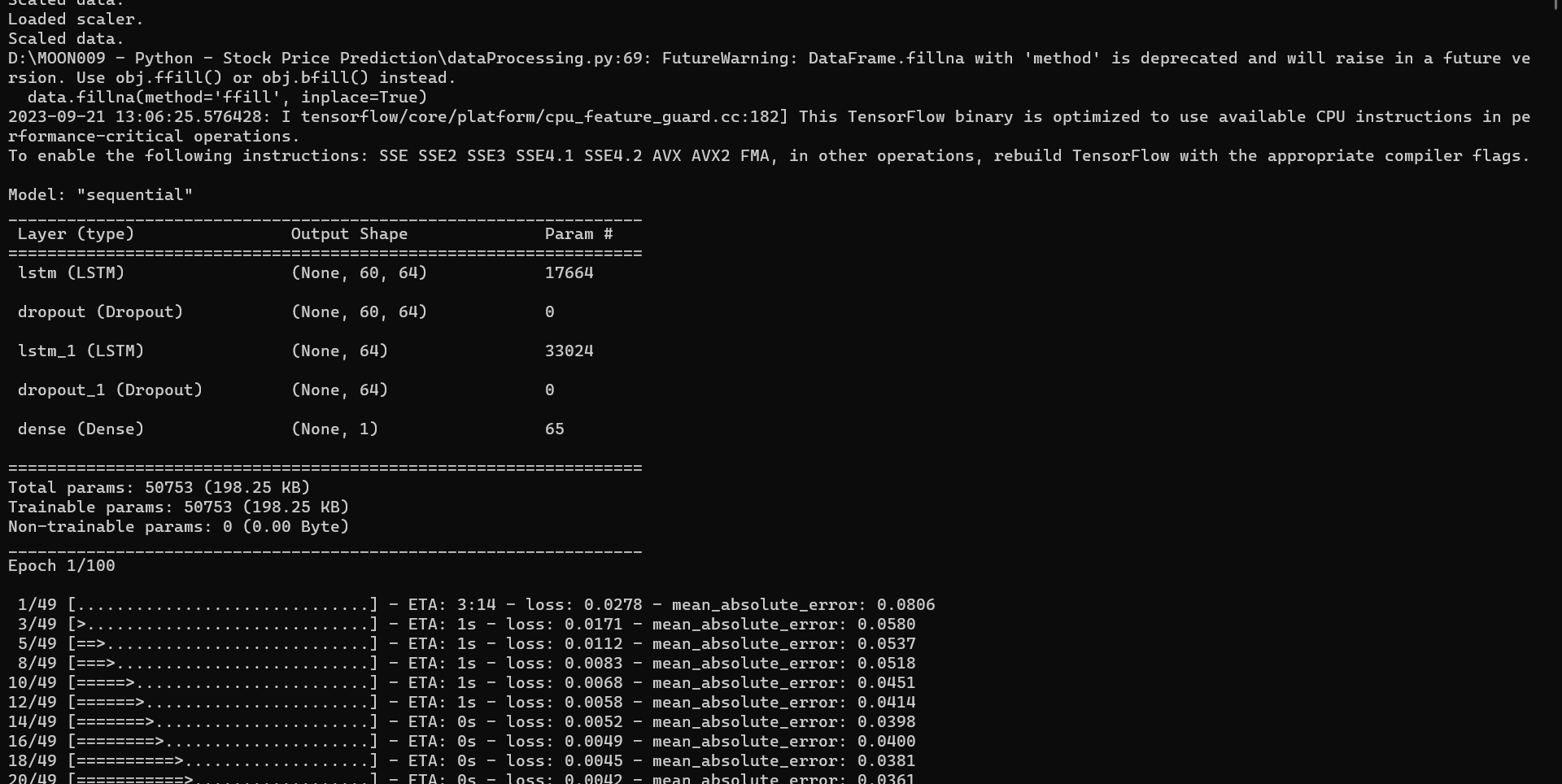
The “yActualData” would be a 2D array, even if “numOfFutureDays” is 1 as default, so the method “np.expand\_dims()” is unnecessary.

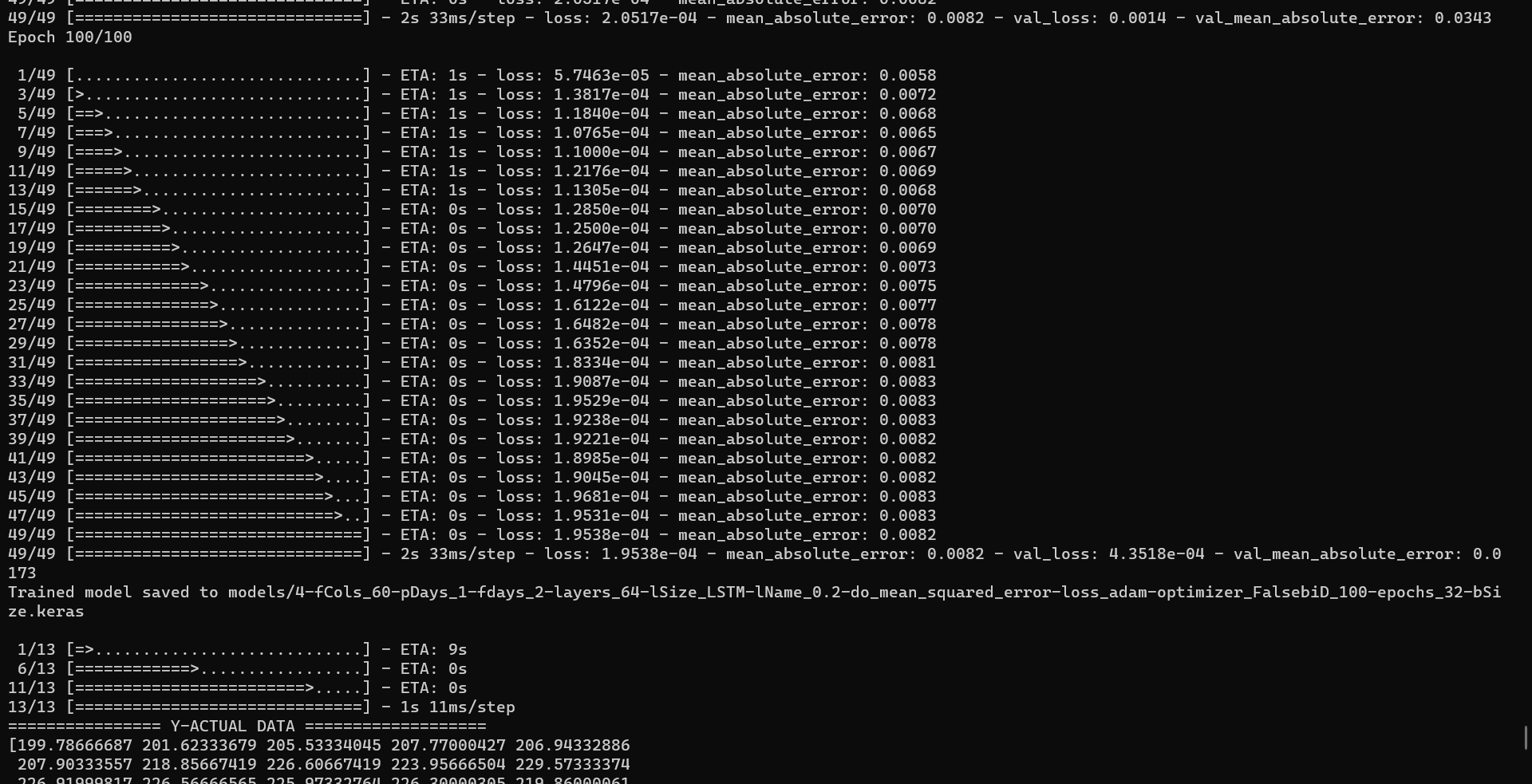
**2. Testing**

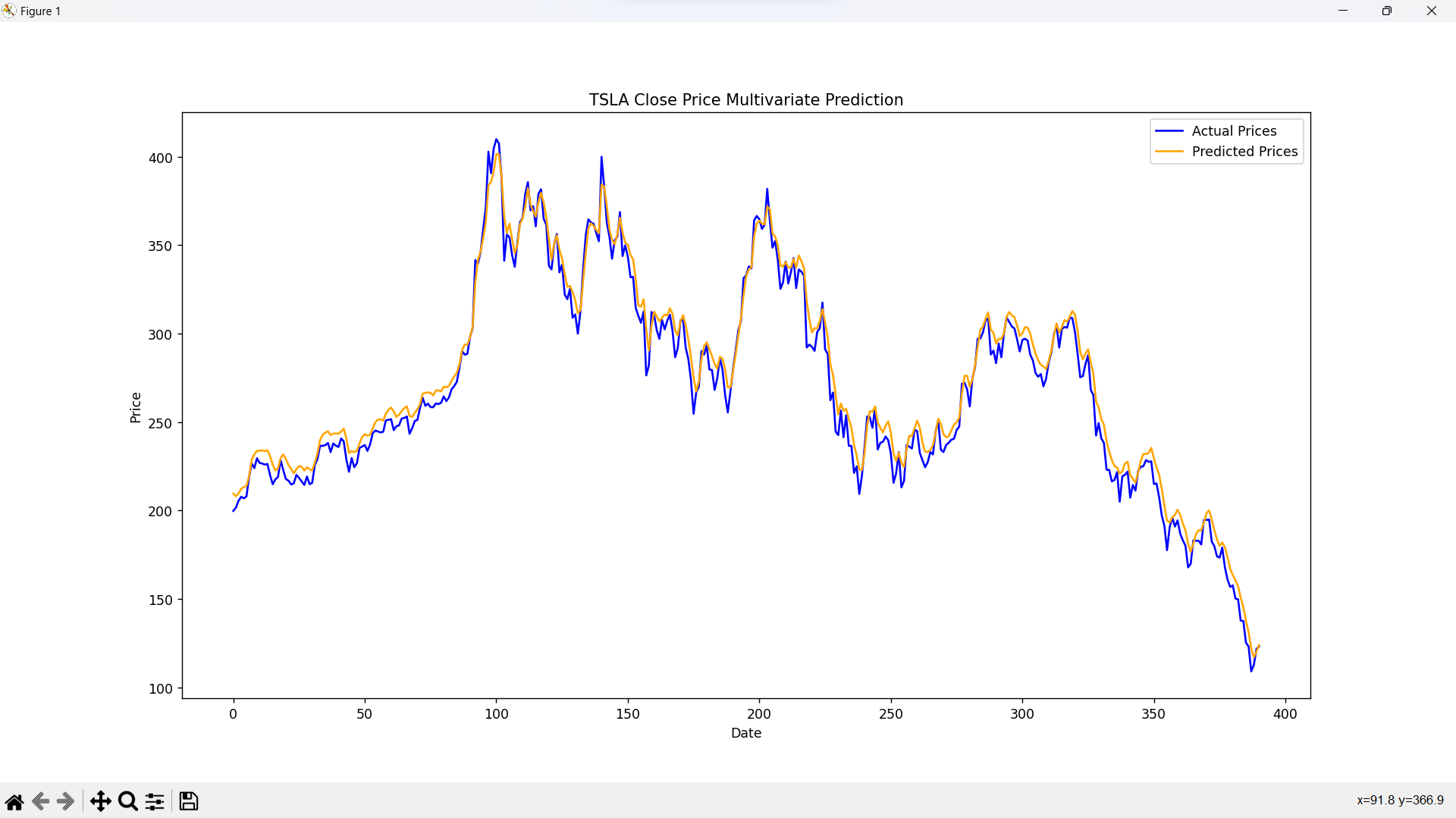
* For the test of this multivariate prediction (LSTM), I used the following parameters:



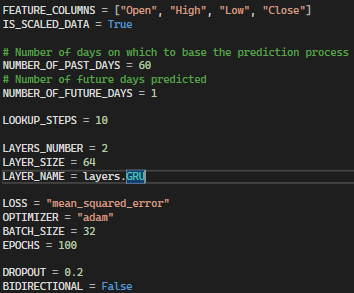
Output:





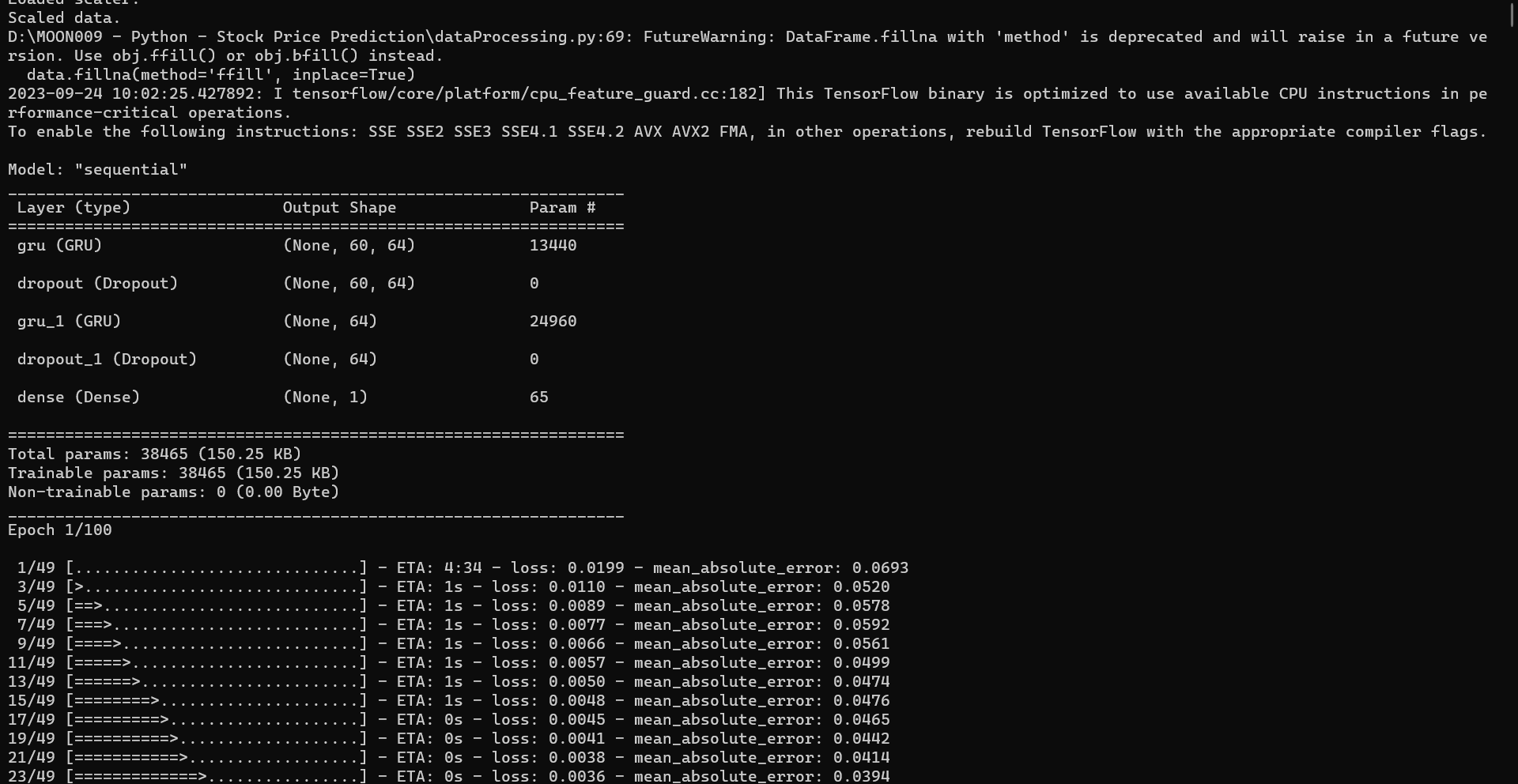


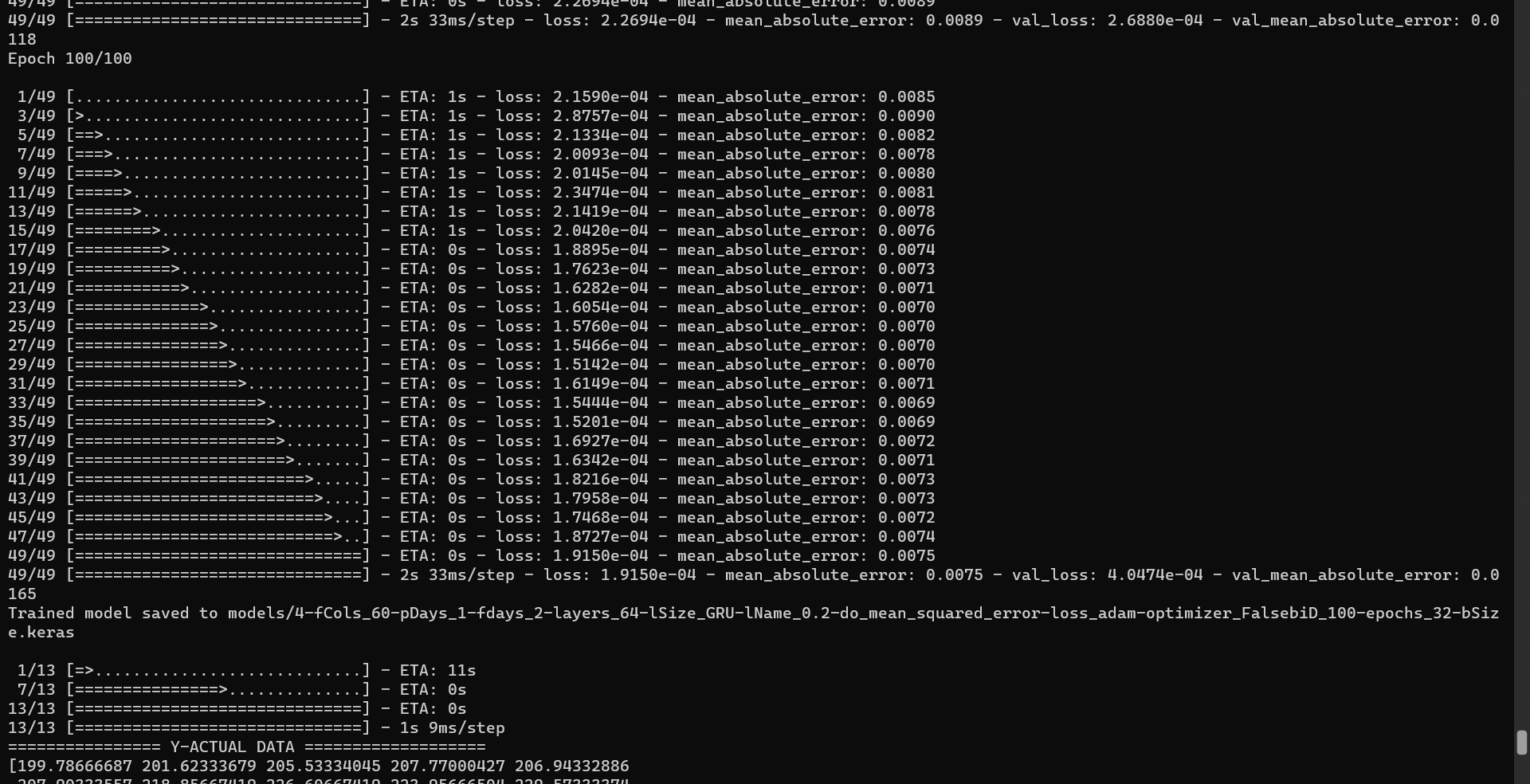
* The second experiment of multivariate problem was executed with GRU layers:

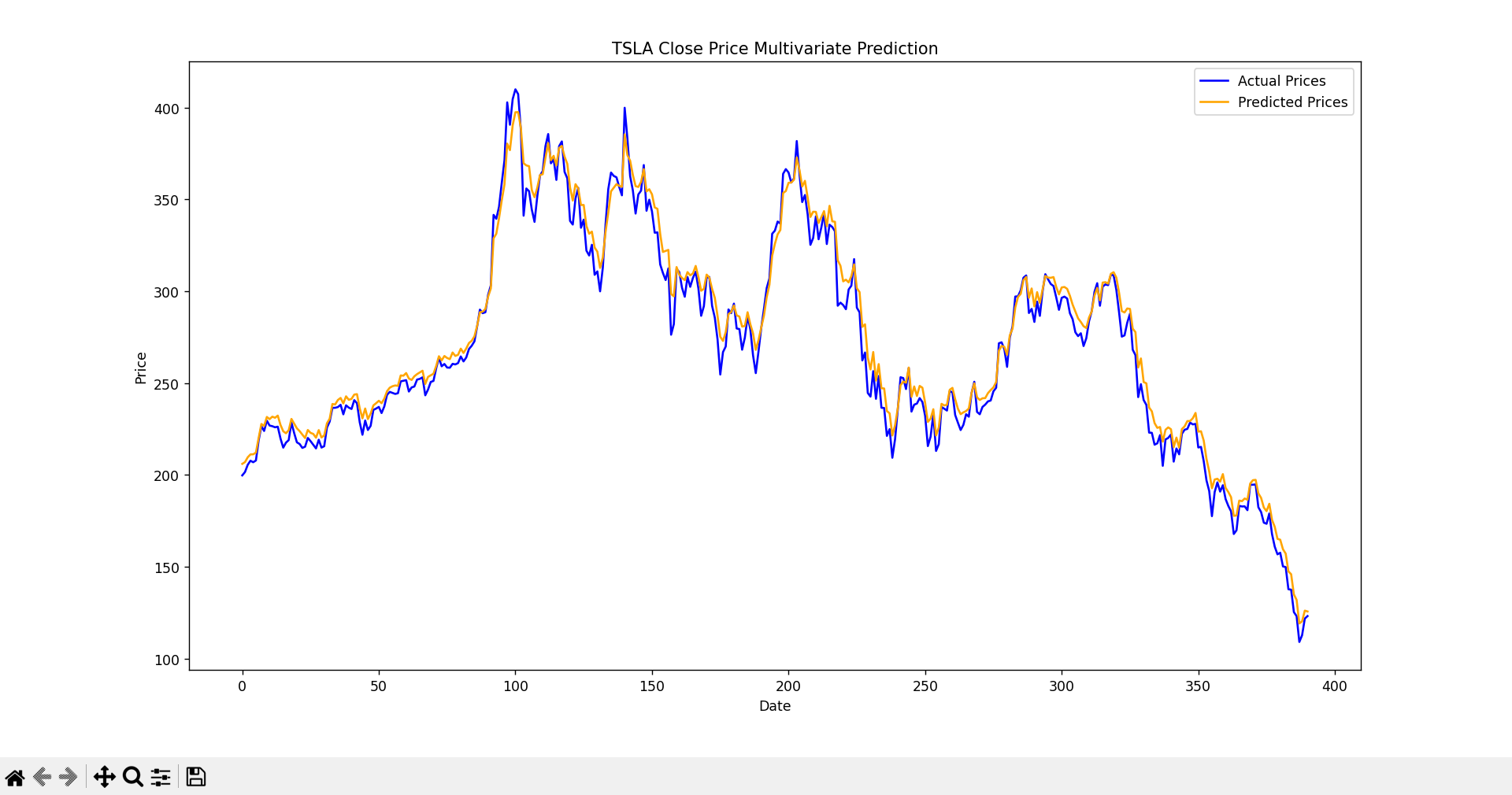


*The other parameters were kept similarly*

Output:







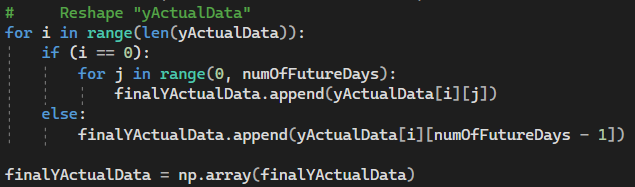
**MULTISTEP & COMBINED PREDICTION**

**1. “trainAndTestMultistepDLModel” method**

As the simple form of multivariate prediction had been implemented in the task B4, hence, in this step, instead of solving the multistep issue only, I dealt with the combined problem in one method, then when testing with multi-step-only and combined prediction, I will simply change the parameters “FEATURE\_COLUMNS”.

This method is basically similar to the metioned “trainAndTestMultivariateDLModel” method, excepts the code for reshaping the “yActualData” and “yPredictedData”, to become 1D arrays for plotting step.

* Reshaping “yActualData” to “finalYActualData:



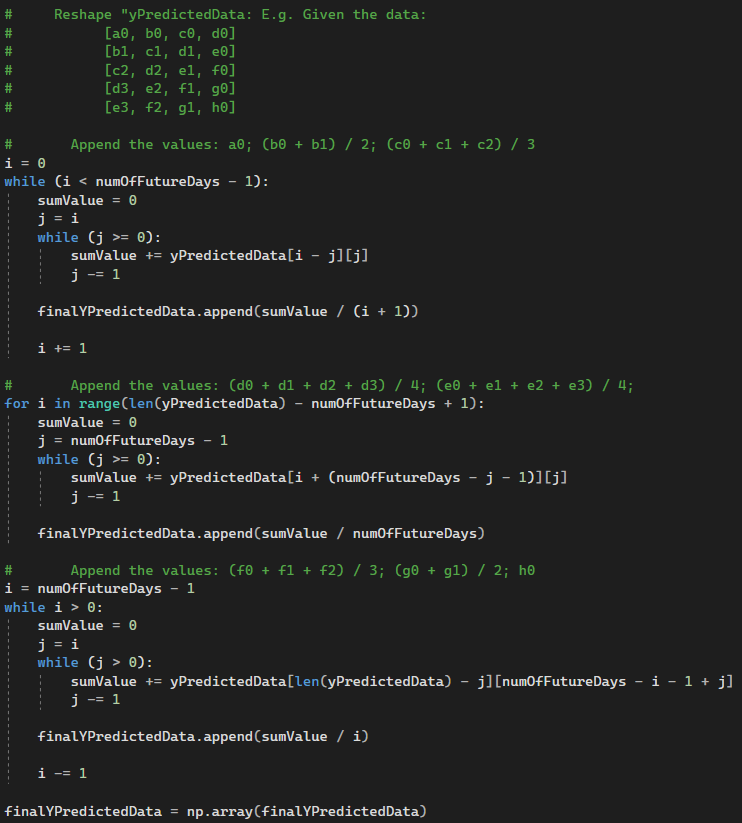
In the “yActualData” array, the last (“numOfFutureDays” – 1) sub-elements of an element, is the first (“numOfFutureDays” – 1) sub-elements of the next one, so, to reshape the data, I just appended all the sub-elements in the first element of “yActualData”, and the last sub-element in the others. E.g.:

Given numOfFutureDays = 3, yActualData = [ [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 6] ]

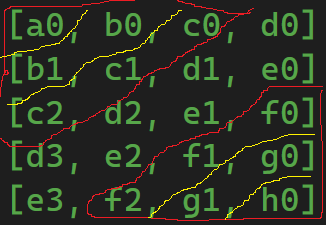
* + - All sub-elements of the first element are: 1, 2, 3
    - The last sub-element of the others are: 4, 5, 6

So, the “finalYActualData” would be: [ 1, 2, 3, 4, 5, 6 ]

* Reshaping “yPredictedData” to “finalYPredictedData”:



This process can be separated in three stages. E.g. Given “numOfFutureDays” = 4, the elements of “yActualData”:



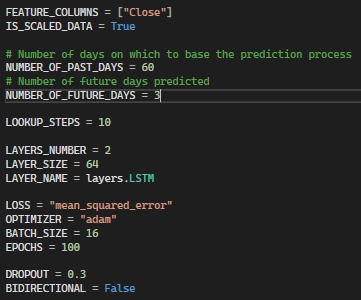
* + Calculate the first (“numOfFutureDays” – 1) element of the output array, which is the average value of 1, 2, … (“numOfFutureDays” – 1) sub-elements as shown in the diagram: a0; (b0 + b1) / 2; (c0 + c1 + c2) / 3.
  + Calculate the rest, except the last (“numOfFutureDays” – 1) element of the output, which is the average value of “numOfFutureDays” sub-elements diagonally: (d0 + d1 + d2 + d3) / 4; (e0 + e1 + e2 + e3) / 4
  + Calculate the last (“numOfFutureDays” – 1) element of the output array, which is the average of (“numOfFutureDays” – 1), (“numOfFutureDays” – 2), … 1 sub-elements as shown in the diagram: (f0 + f1 + f2) / 3; (g0 + g1) / 2; h0

In addition, the method of “constructDLModel()” now has an additional parameter of “numOfFutureDays”, and the Dense layer of the output model has “units” parameter of “numOfFutureDays” instead of “len(featureColumns)”. (This is also applied to the multivariate issue, when the units of the Dense layer should be the number of days of the future predicted sequence):



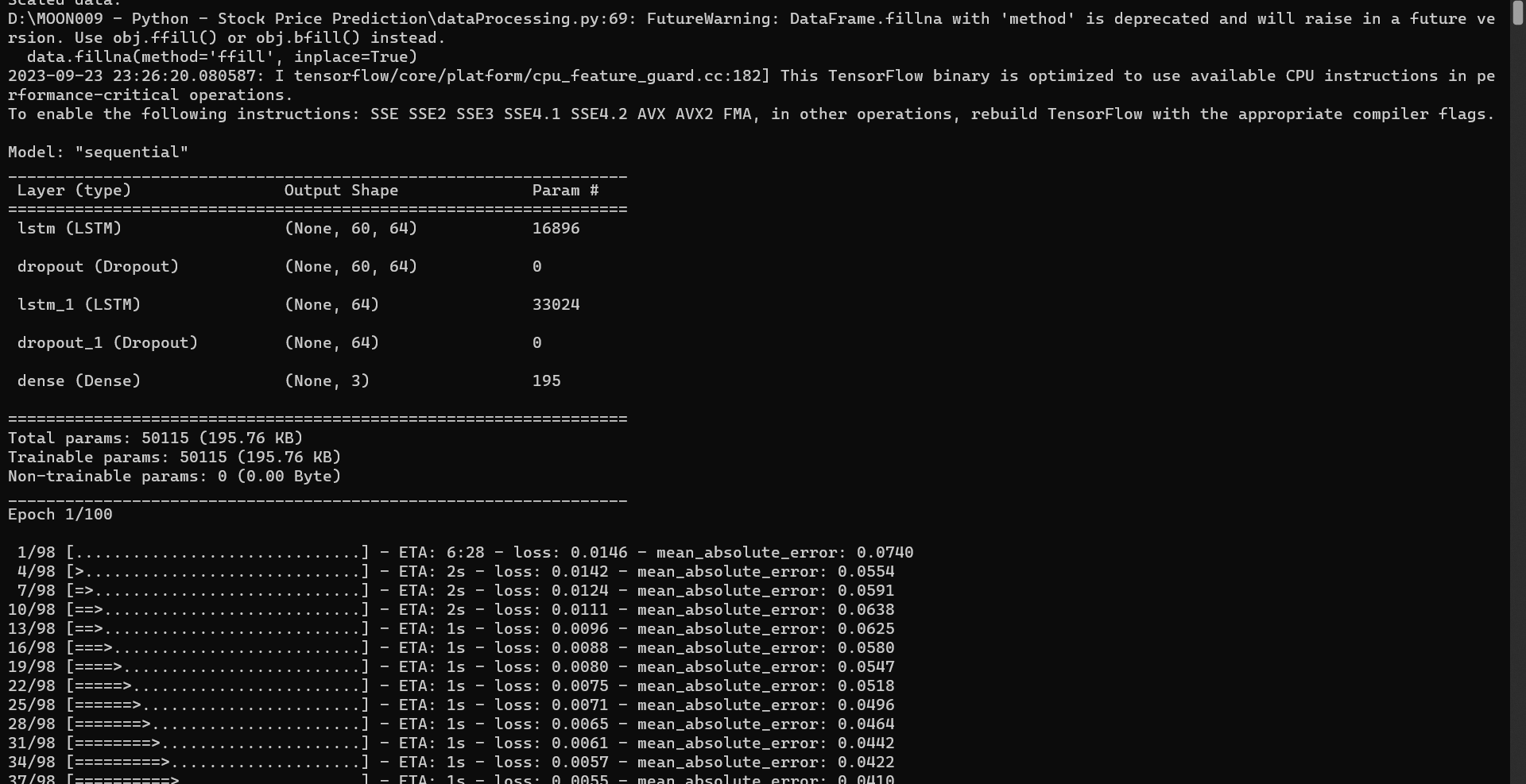
**2. Testing with multistep-only model**

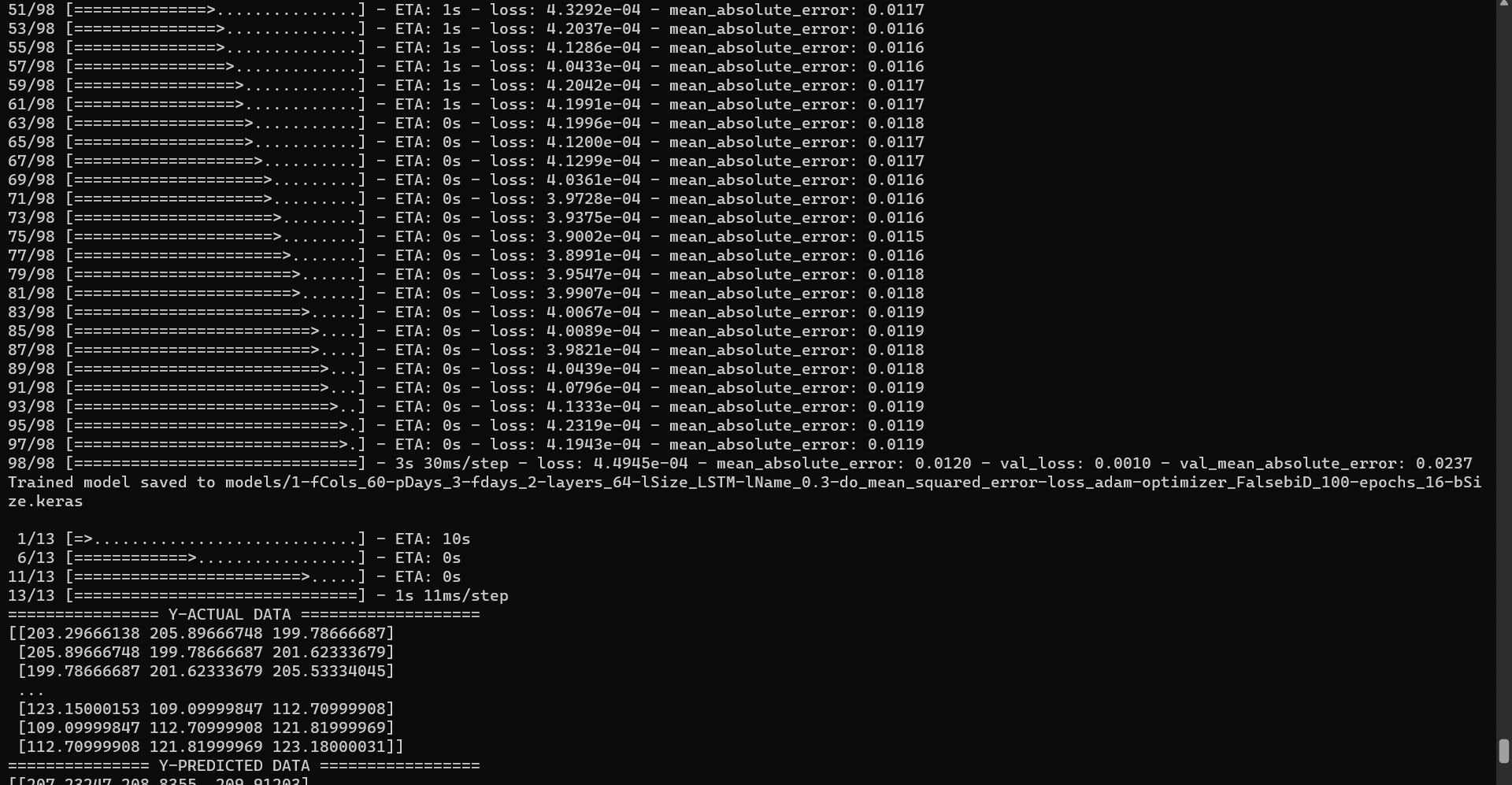
* In this test, I used the following parameters:

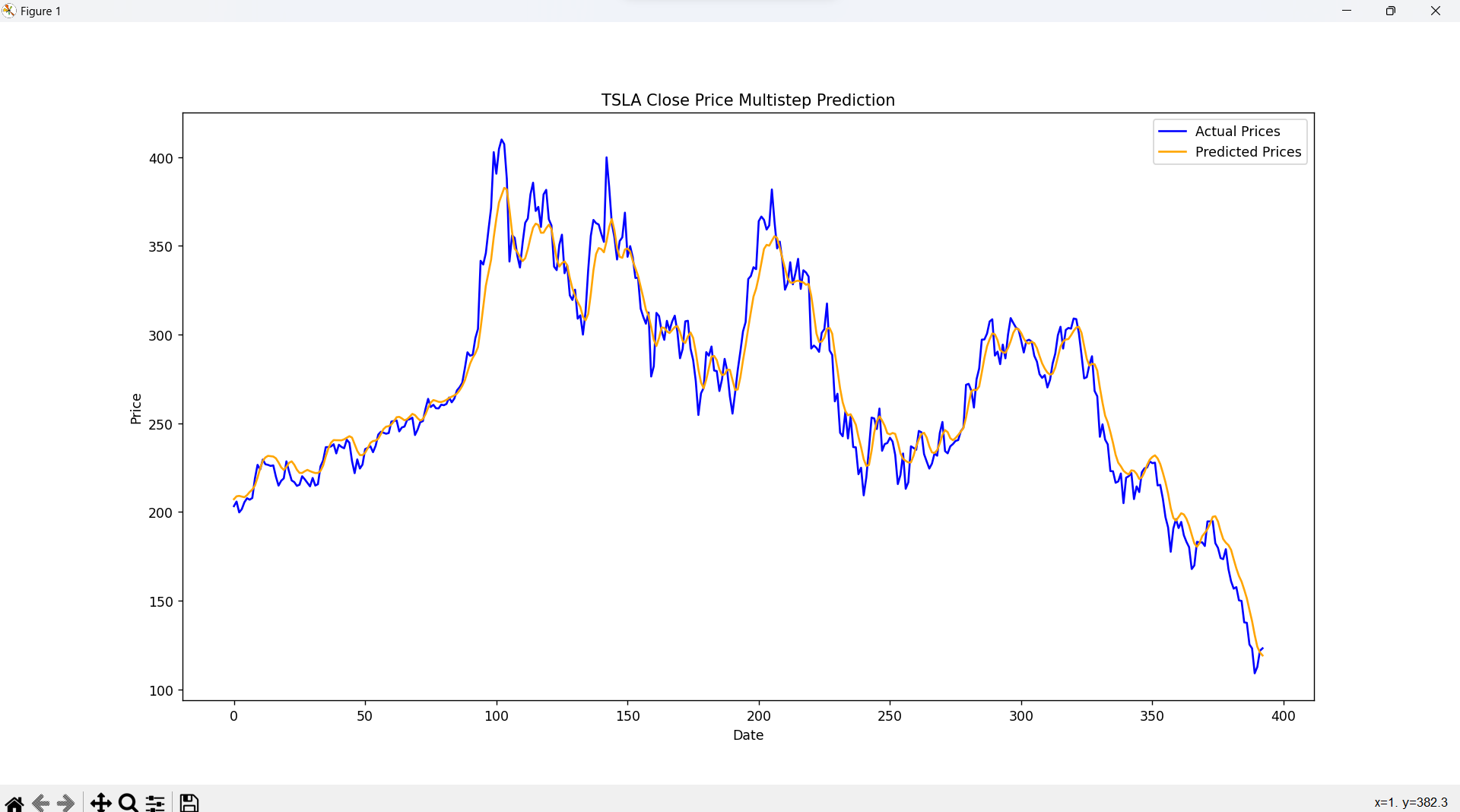
****

*LSTM layers*

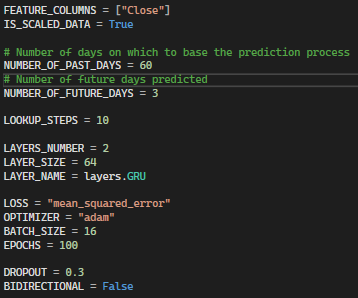
Output:

****

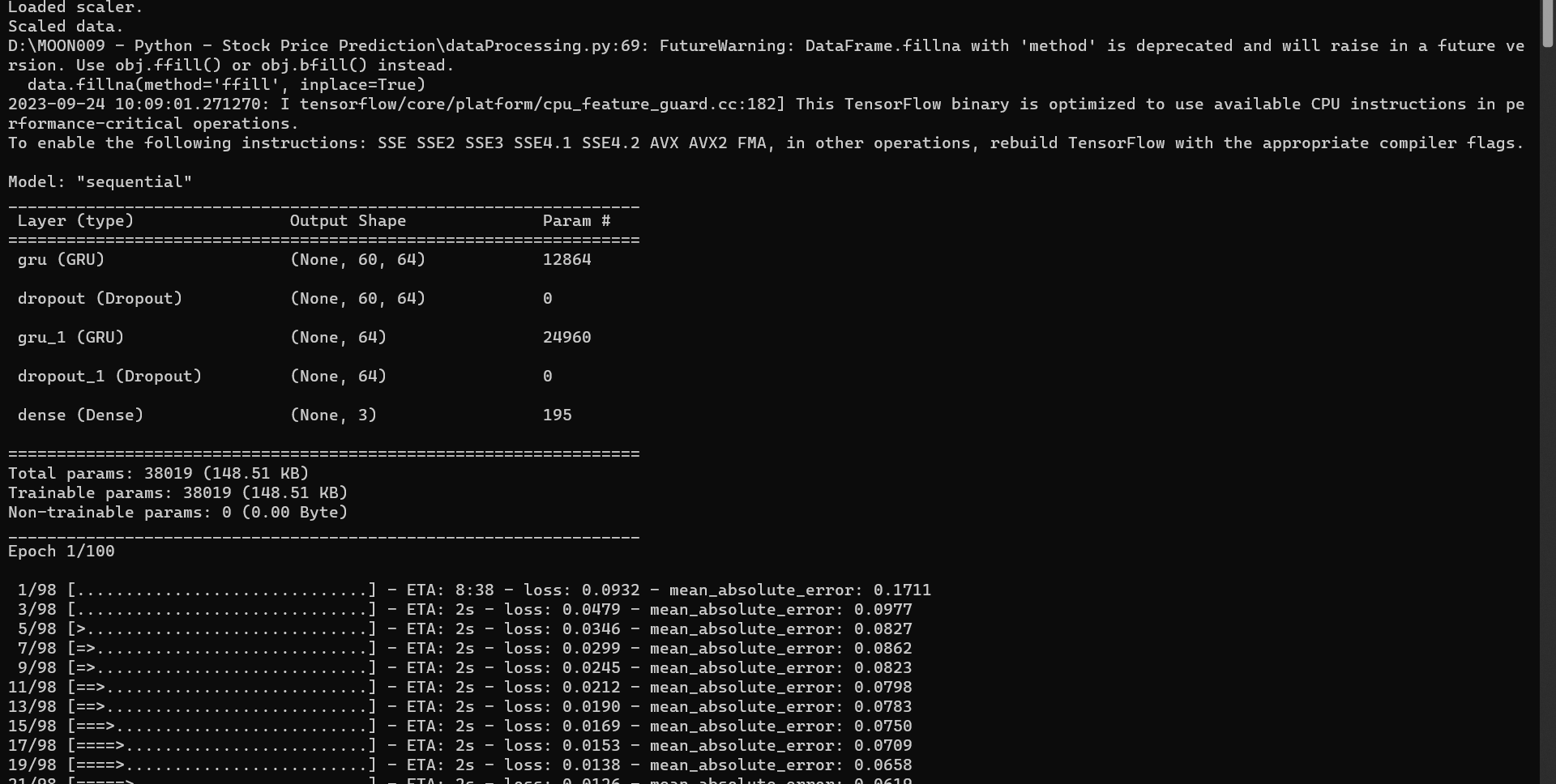


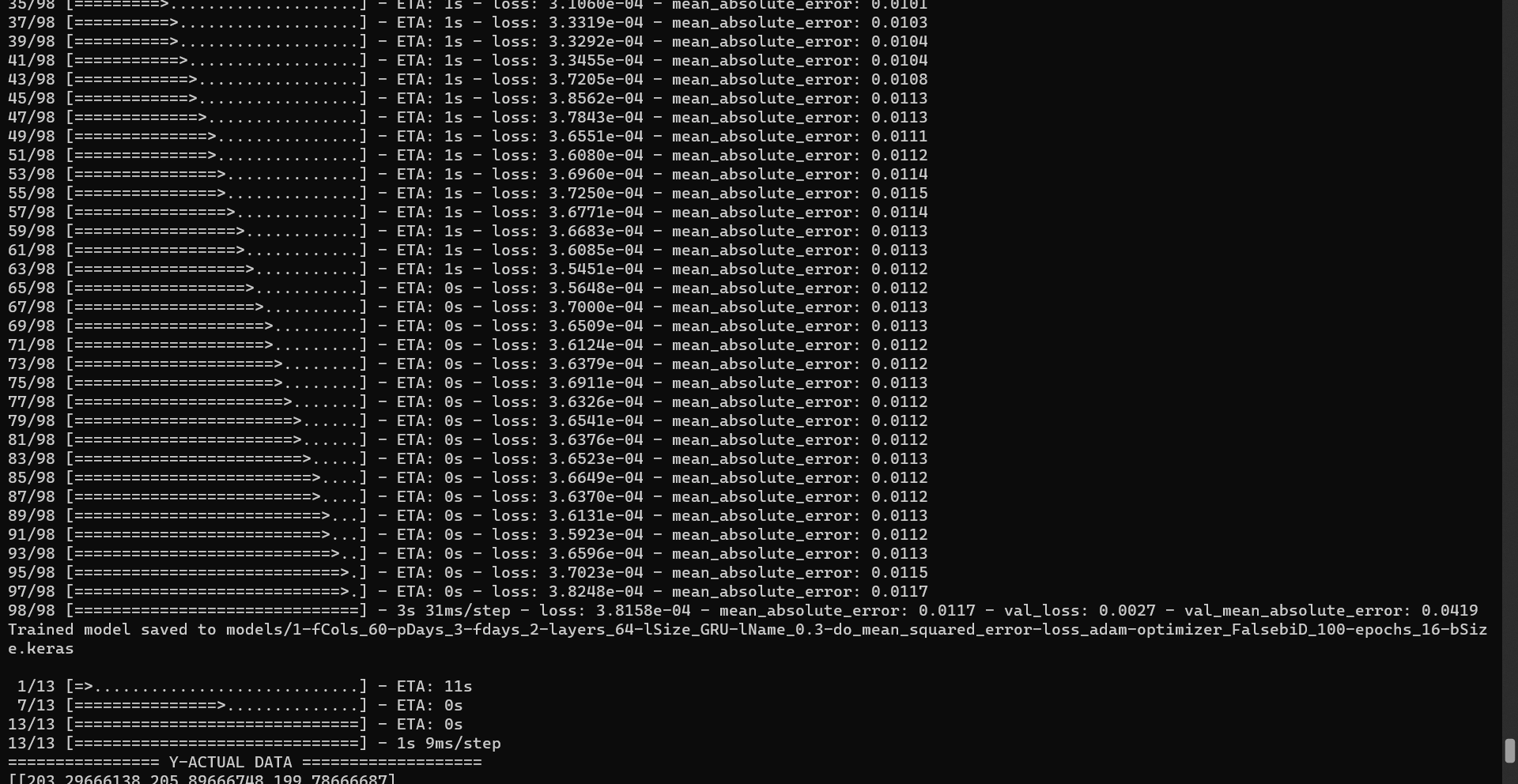
****

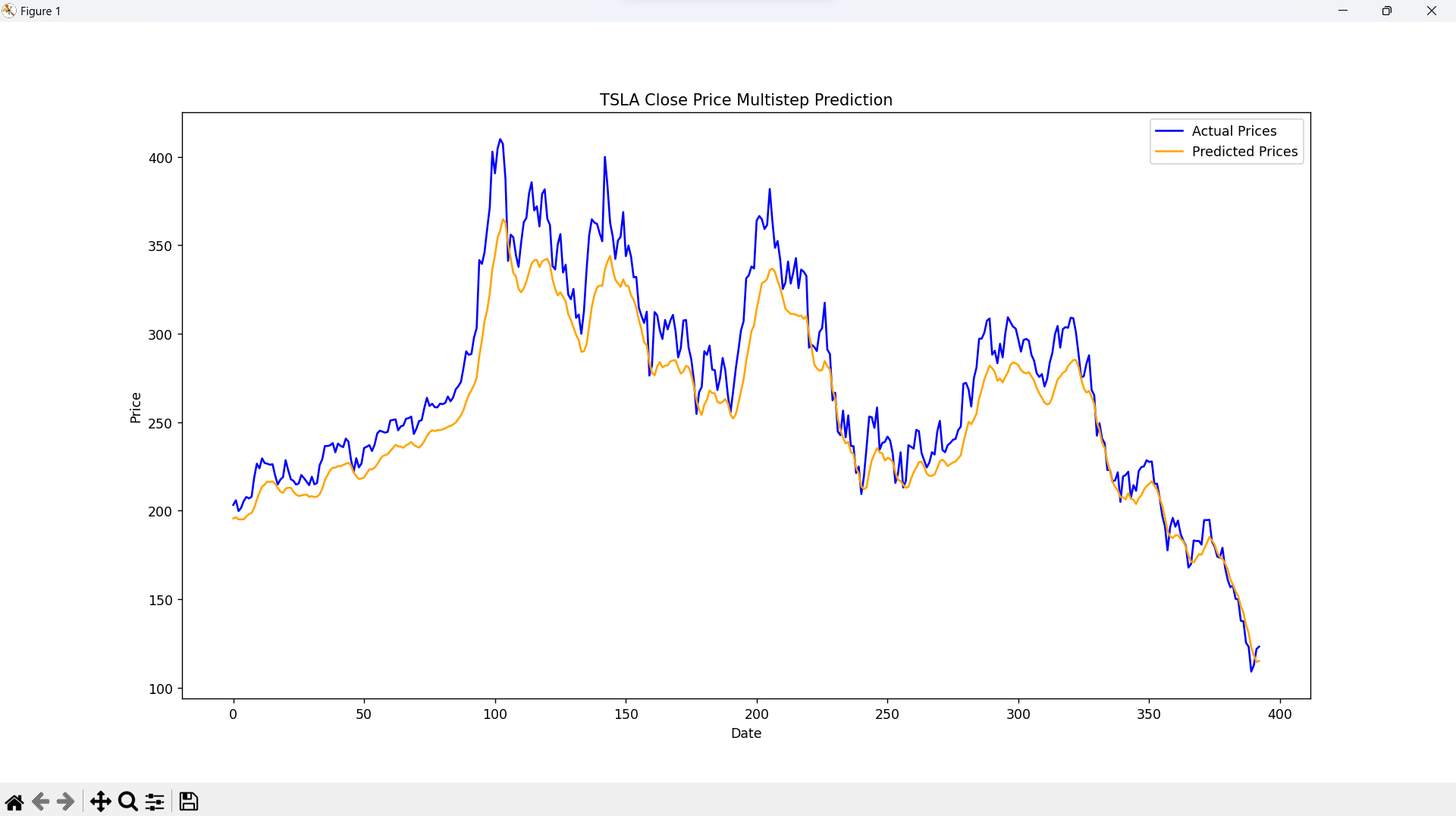
* When testing the multistep model with GRU layers, I still kept others the same:



Output:

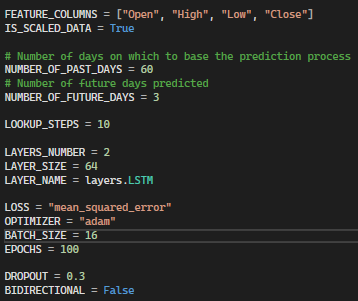






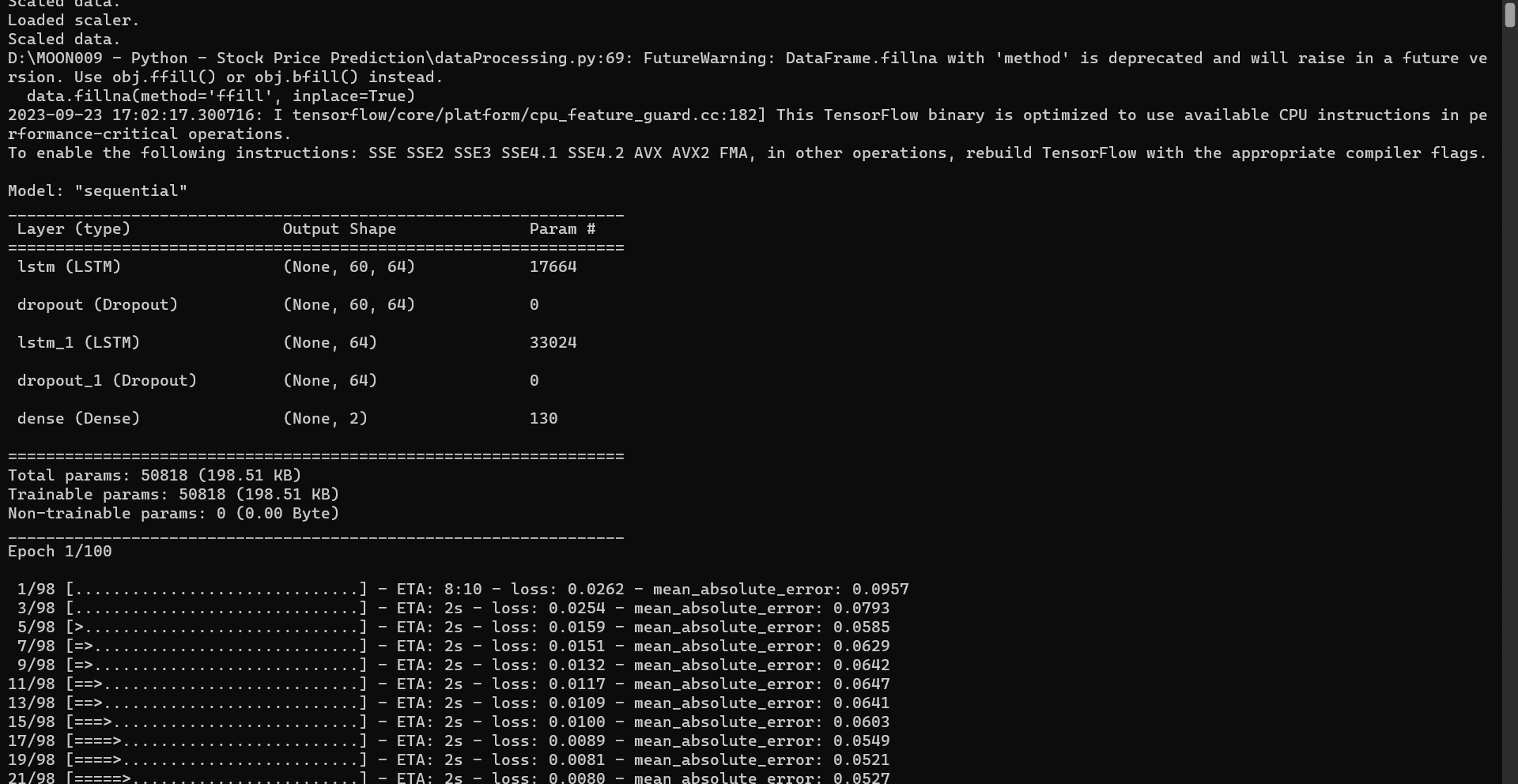
**3. Testing with combined model**

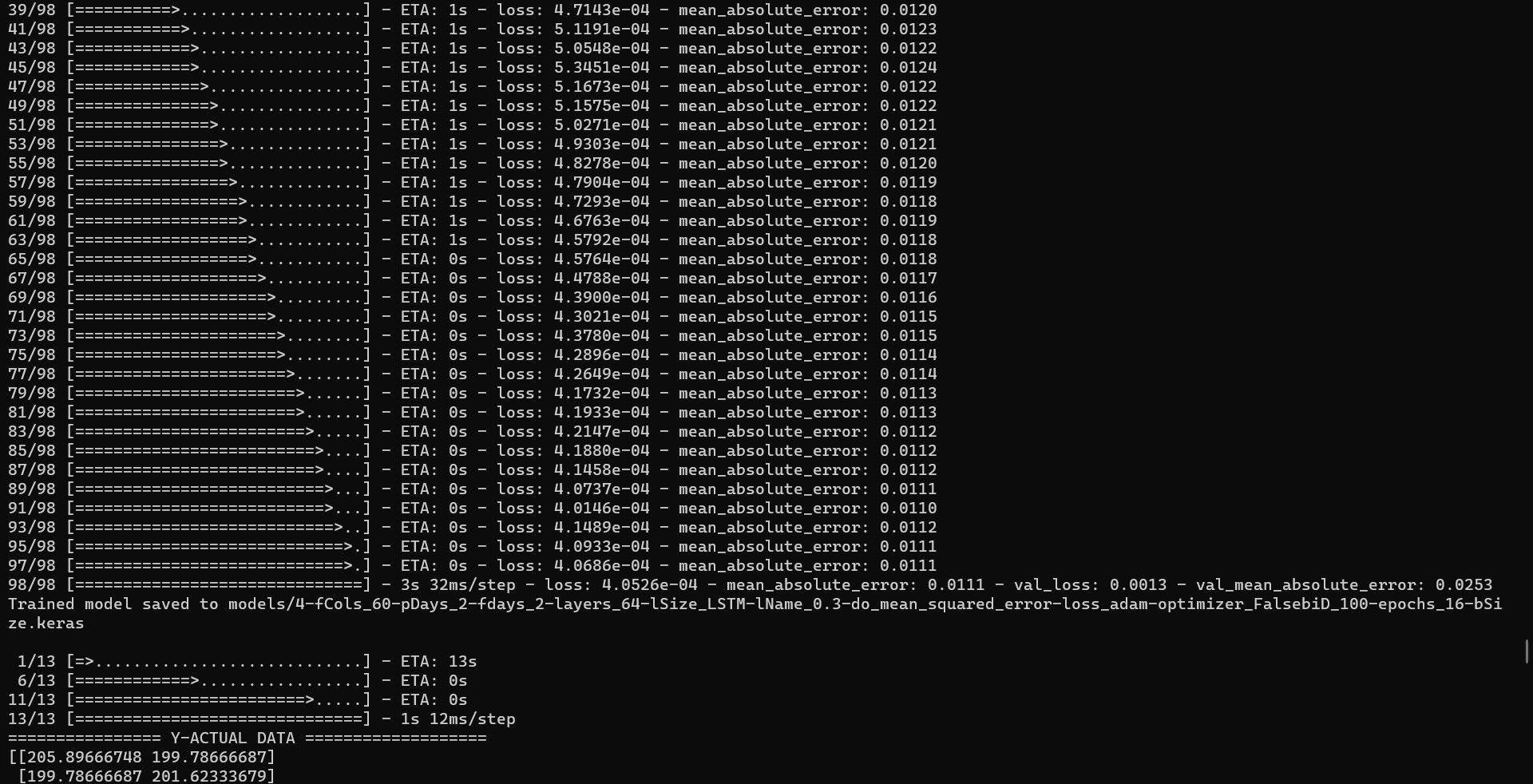
* In the first test with LSTM layers, I used the following parameters:

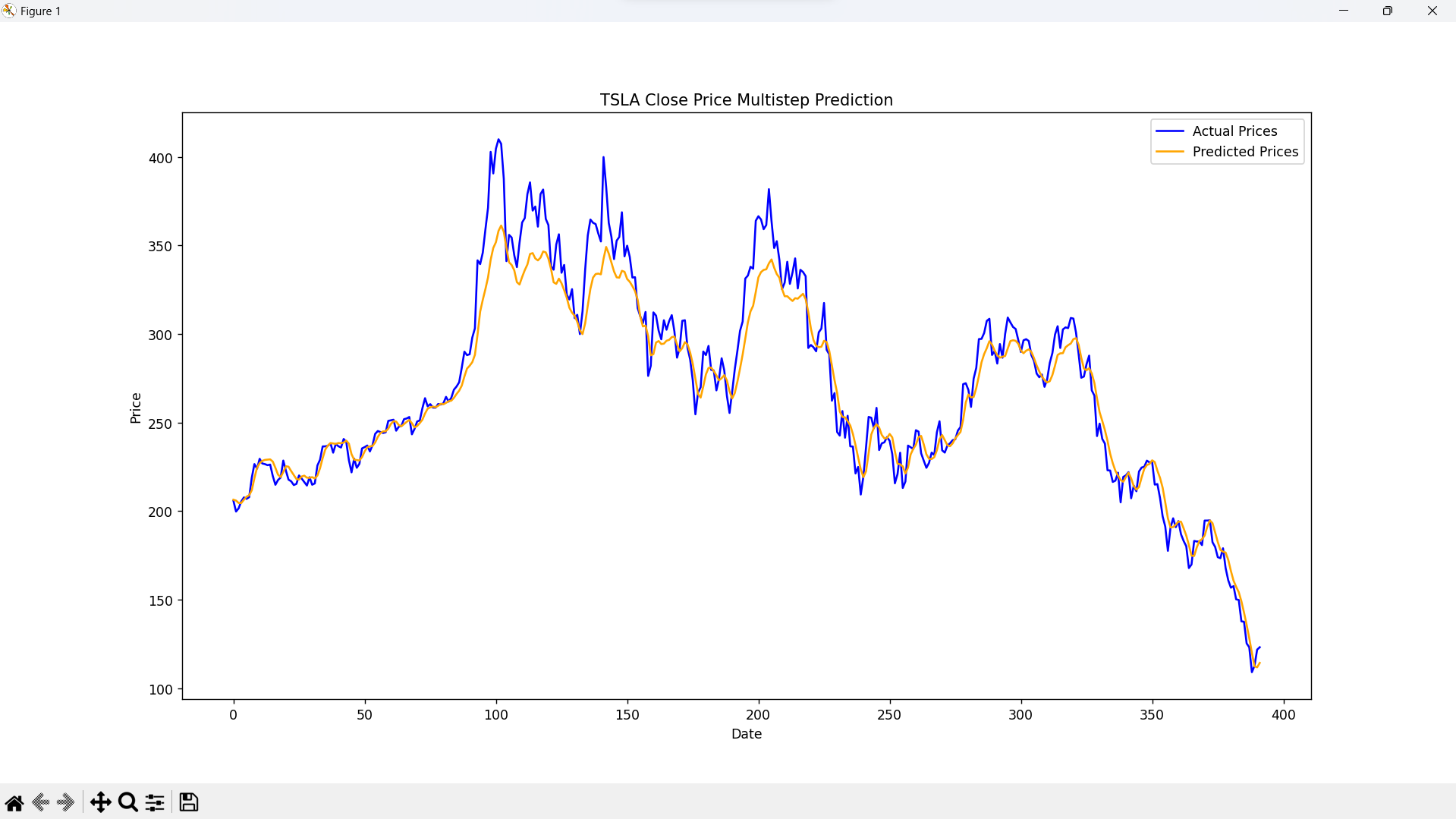


*As I have mentioned, the multi-step issue was basically solved in the previous task, so to combined the two types of prediction, I simply changed the “FEATURE\_COLUMNS”*

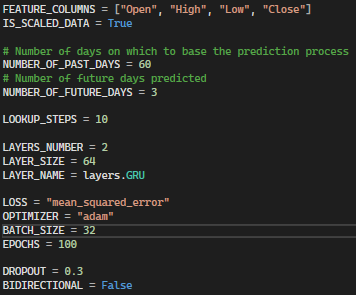
Output:



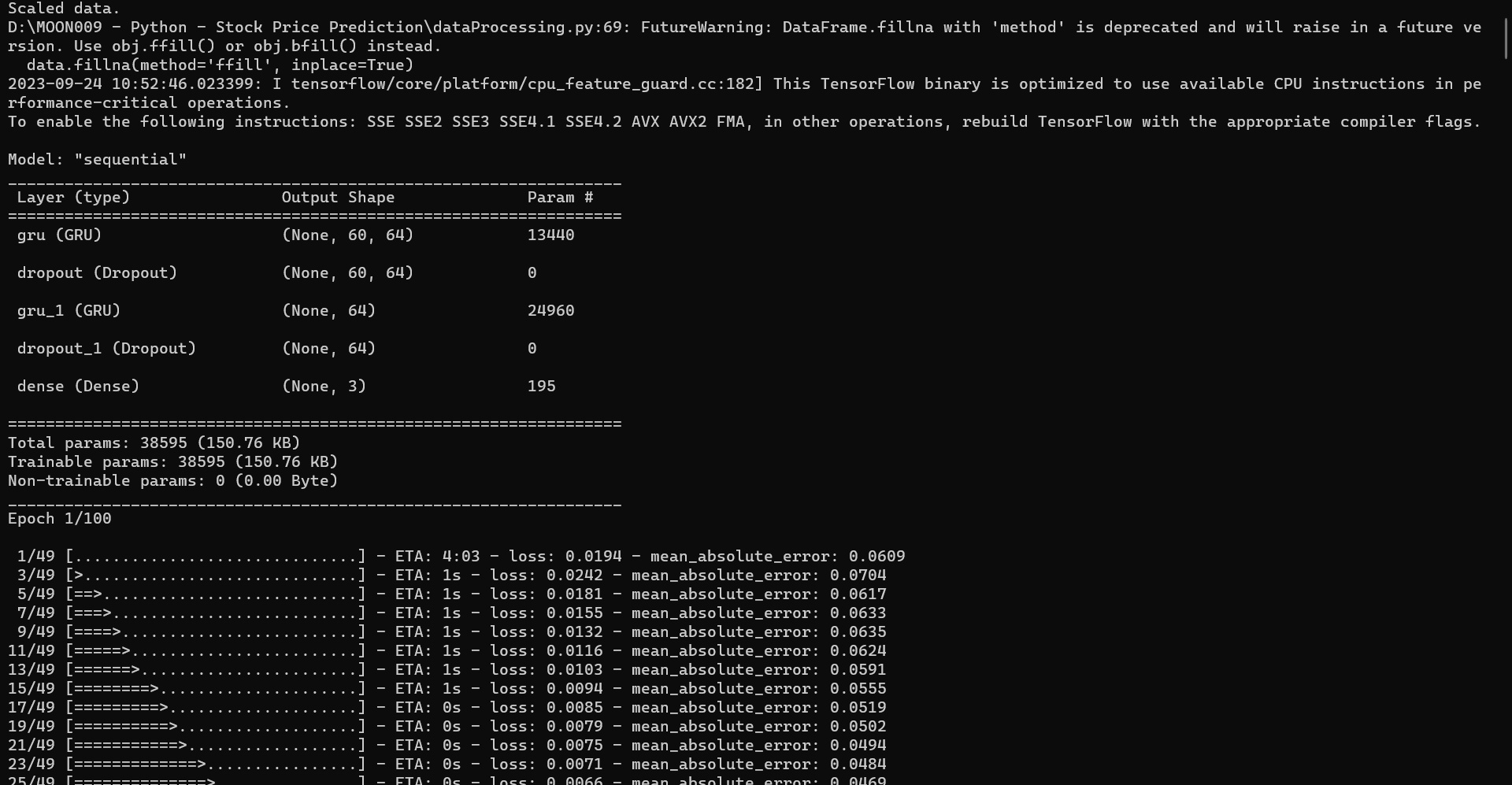


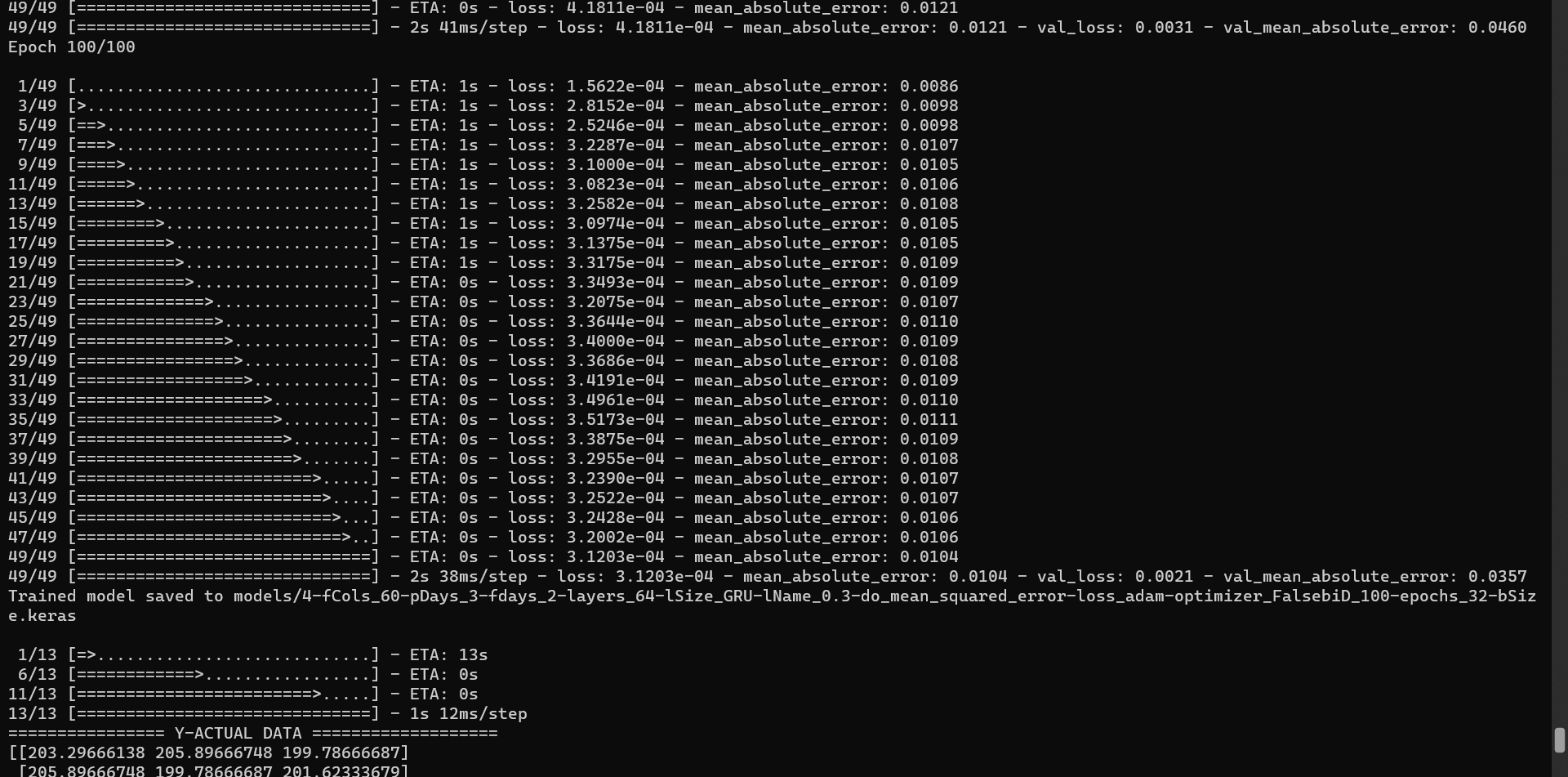


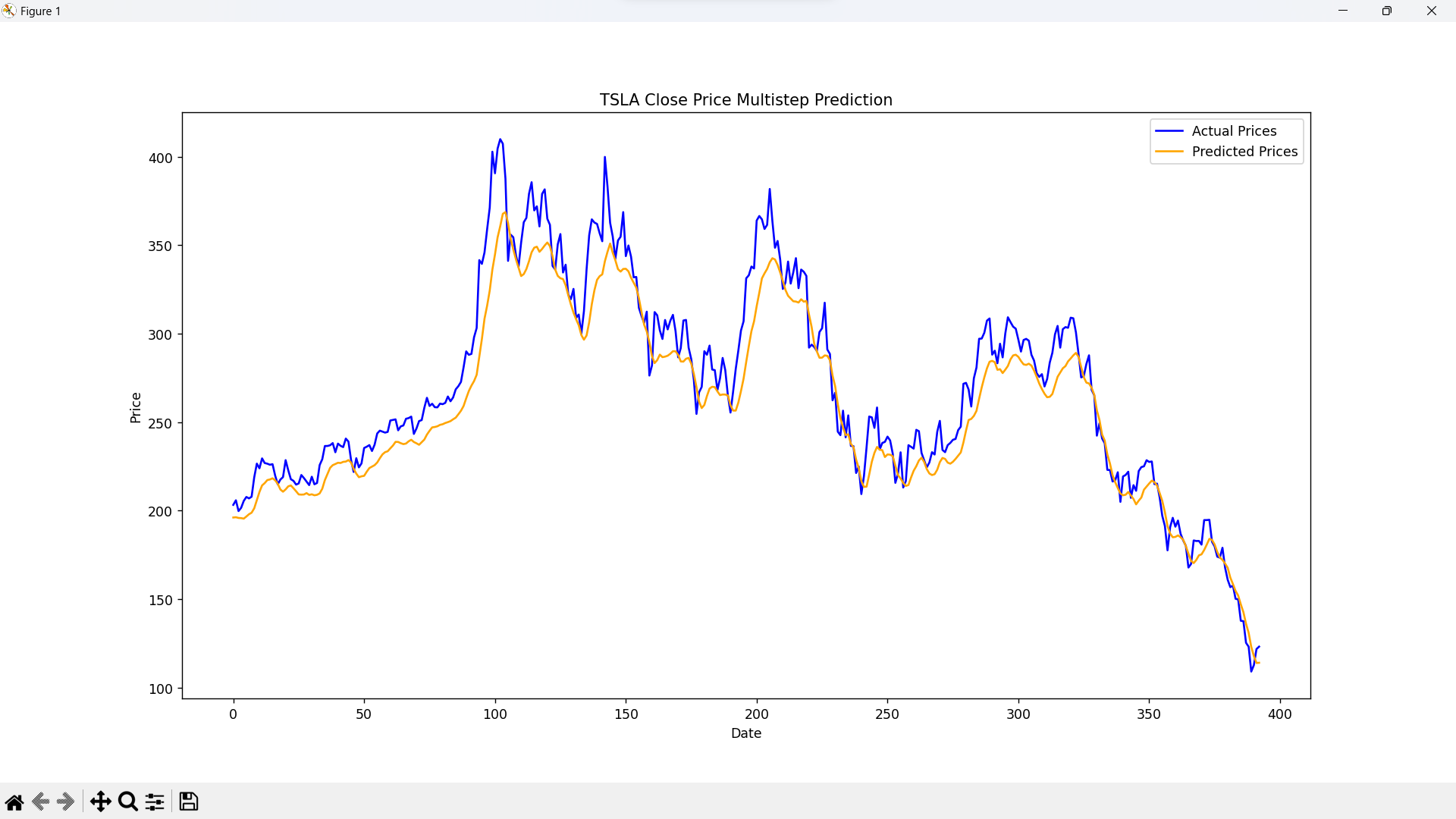
* The GRU layers’ experiment was conducted with the following parameters:



Output:







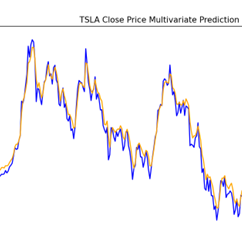
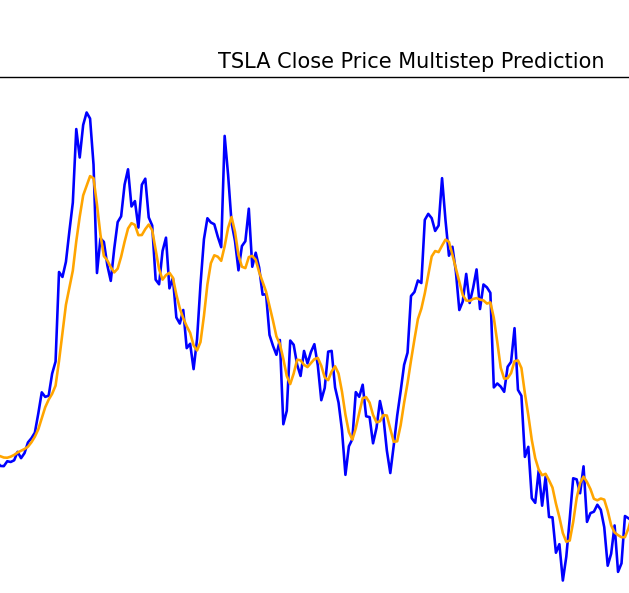
**SUMMARY**

This part will summarise the the results of my experements with multivariate, multistep, and combined prediction issue.

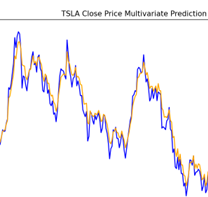
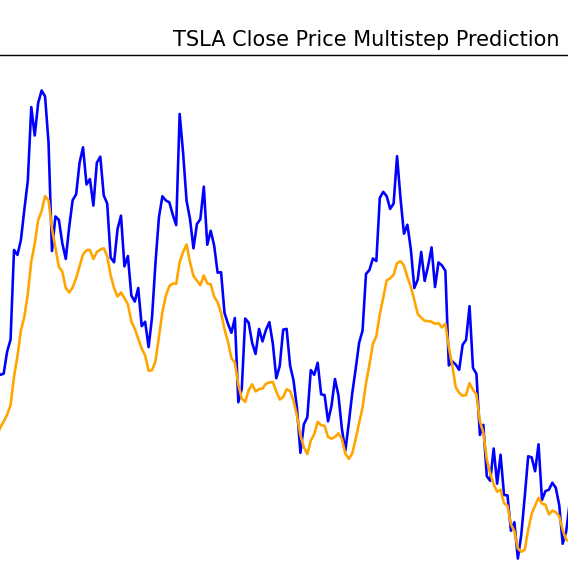
When comparing the results from the experiments of the two cases of using LSTM and GRU layers, it is clear that predicting with multivariate model singularly seems to be more accurate than with multistep-only functionality.

Specially, the most significant difference comes in the middle time of the tested period, when the actual data witnessed the crucial fluctuation in closing price:

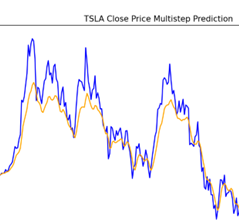
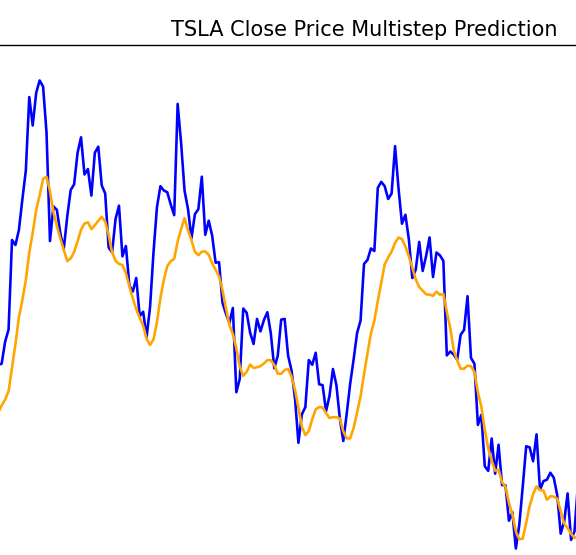
* Using LSTM layers (multivariate-only and multistep-only):

* Using GRU layers (multivariate-only and multistep-only):

I assumed that, with a complex and hard-to-predict data type like stock price, especially when having to use a lot of feature data to make predictions for one type, using the multistep feature and then averaging the found values will make the final result tend to be more "steady" than predicting a single step. This disadvantage also happened even when predicting the closing price with combined of the two functionality: (LSTM – left; GRU – right)

**REFERENCES**

[1] T. Hurson, “Stock Price Prediction with LSTM/Multi-Step LSTM,” Kaggle. <https://www.kaggle.com/code/thibauthurson/stock-price-prediction-with-lstm-multi-step-lstm/notebook>

[2] S. Khankary, “Multivariate Time Series Forecasting using RNN(LSTM),” MLearning.ai, Jan. 27, 2022. <https://medium.com/mlearning-ai/multivariate-time-series-forecasting-using-rnn-lstm-8d840f3f9aa7>