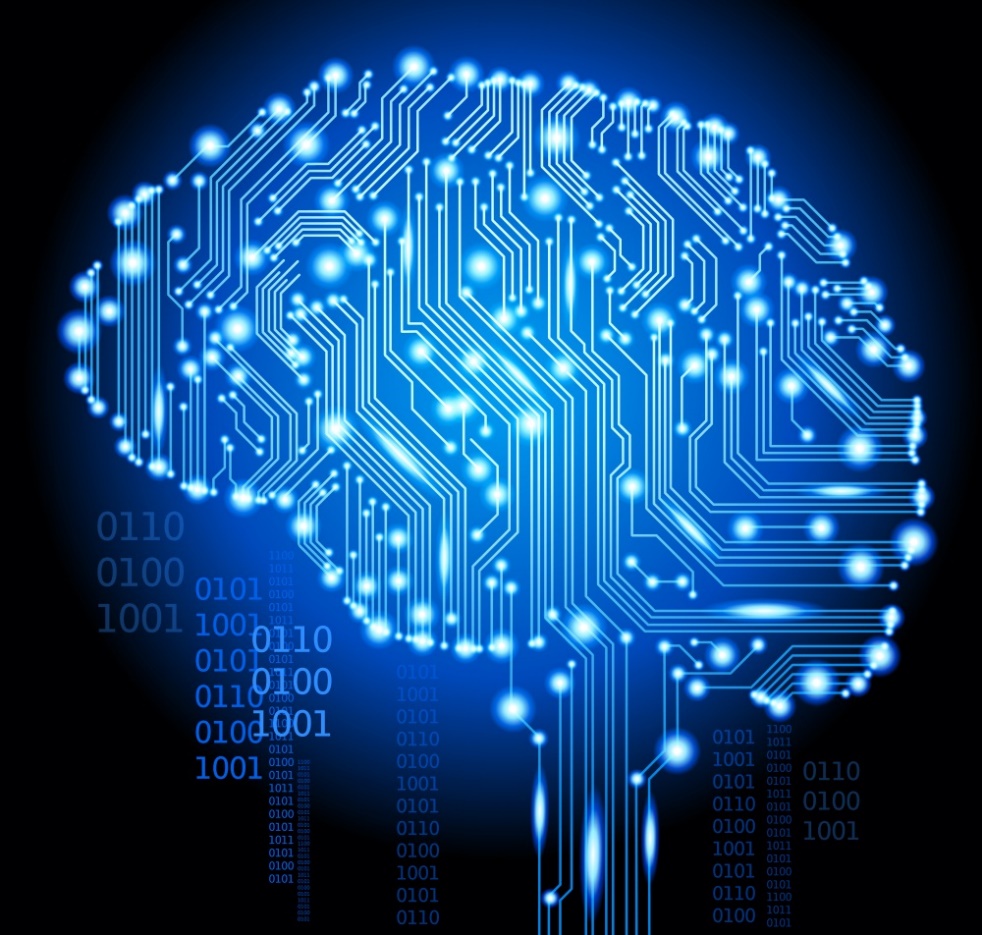


COS30018 – Intelligent Systems

**REPORT**

TASK B6 – MACHINE LEARNING 3



**TABLE OF CONTENTS**

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

*I. ARIMA model……………………………………………………………………………………………………………….....3*

*1.“createARIMAModel()” method………………………………………………………………………….....3*

*2. Test independently…………………………………………………………………………………………………5*

*3. Ensemble with LSTM/GRU/RNN model…………………………………………………………………..7*

*4. Summary……………………………………………………………………………………………………………..10*

*II. Random forest………………………………………………………………………………………………………………10*

1. *“createRTModel()” method…………………………………………………………………………………10*
2. *Test independently………………………………………………………………………………………………12*
3. *Ensemble with LSTM/GRU/RNN model………………………………………………………………..13*
4. *Summary…………………………………………………………………………………………………………….16*

*III. Conclusion……………………………………………………………………………………………………………………16*

*IV. References……………………………………………………………………………………………………………………17*

**ARIMA Model**

**1. “createARIMAModel()” method**

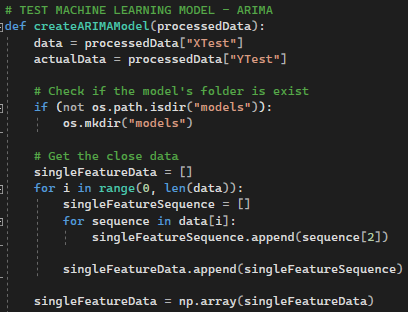
It is clear that, the LSTM outperforms ARIMA result, since the ARIMA model only give the “trend” of the future data, not specificly accurate data date-by-date as LSTM. For example:

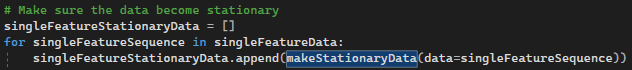
*Output of Forecasting of Stock Market using ARIMA. Retrieved from* [*https://medium.com/@raj.saha3382/forecasting-of-stock-market-using-arima-in-python-cd4fe76fc58a*](https://medium.com/@raj.saha3382/forecasting-of-stock-market-using-arima-in-python-cd4fe76fc58a)

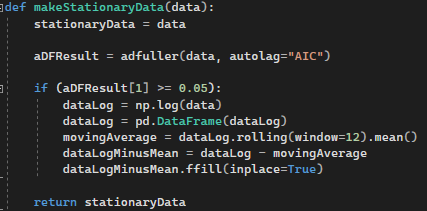
Therefore, instead of using a single sequence of previous data to predict a corresponding sequence of future data with a single ARIMA model, I using multiple ARIMA models. This means that, each daily data in the future sequence has a corresponding ARIMA model, using a correponding of sequence of previous days’ data, which has been splitted in the “dataProcessing.py” file (the “XTest”).

The first thing is to get multiple sequences of previous days’ closing data, this can be done easily in just some “for” loops:



After having the “singleFeatureData” array, which includes many sequences of closing data, I checked if each sequence in that array contains stationary data or not (in a nut shell, if there is no trend or any seasonal impacts, the data can be considered to be stationary), through a separete method named “makeStationaryData()”:



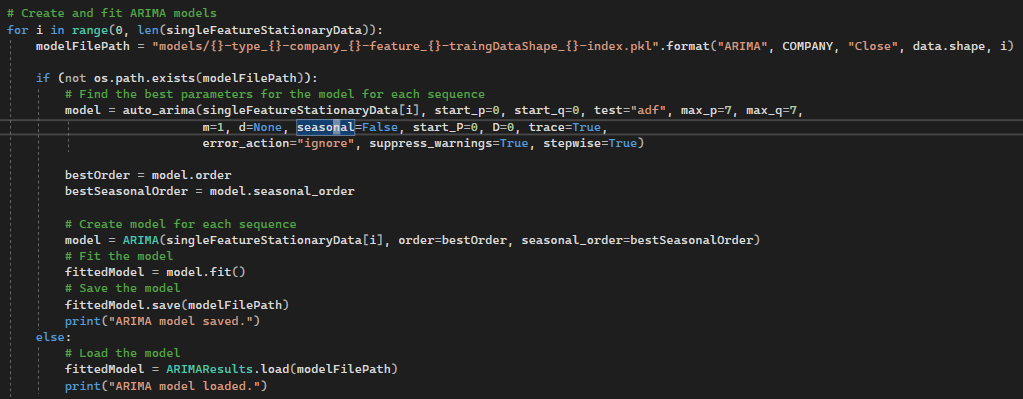


In this method, I used “adfuller” from “statsmodels.tsa” to get the p-Value, stored in “aDFResult[1]”. ADF (Augmented Dickey-Fulle) test on the input “data”, checking the stationarity of a time series.

If that p-Value is greater than 0.05, the data is non-stationary, and it will be converted by:

* Taking the log of the data, using “Numpy.log”
* Calculating the moving average by computing the rolling mean for the data log
* Subtracting the moving average from that

The next part is creating the models:

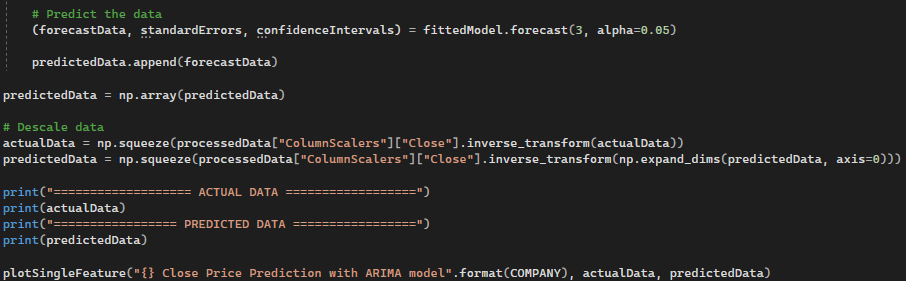


For each sequence in the “singleFeatureStationaryData”, if the corresponding model had not been created, it would be created:

* Initially, the best parameters of “order” and “seasonal\_order” for the model are found using the “auto\_arima()” method from the module “pmdarima”.
* Then, with the method “ARIMA” from the module “statsmodels.tsa.arima.model”, the suitable ARIMA model is created, and fitted.

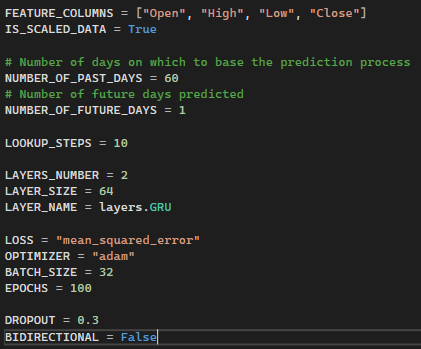
Otherwise, the “ARIMAResults.load()” (module “statsmodels.tsa.arima.model”) method is used to load the corresponding model from the “models” folder.

After that, each created model only predicts its corresponding days (the following days of the used sequence), the output is appended to the predicted array, which is then plotted and returned with “actualData” (the “Ytest” in “processedData”)



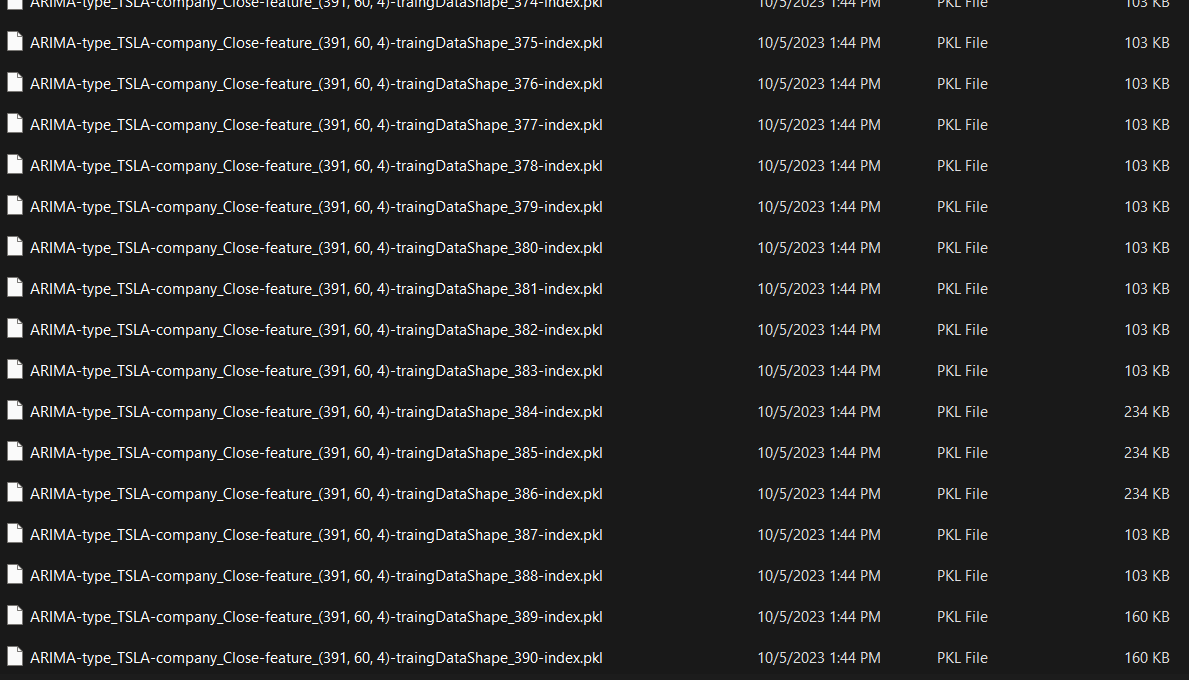
**2. Test independently**

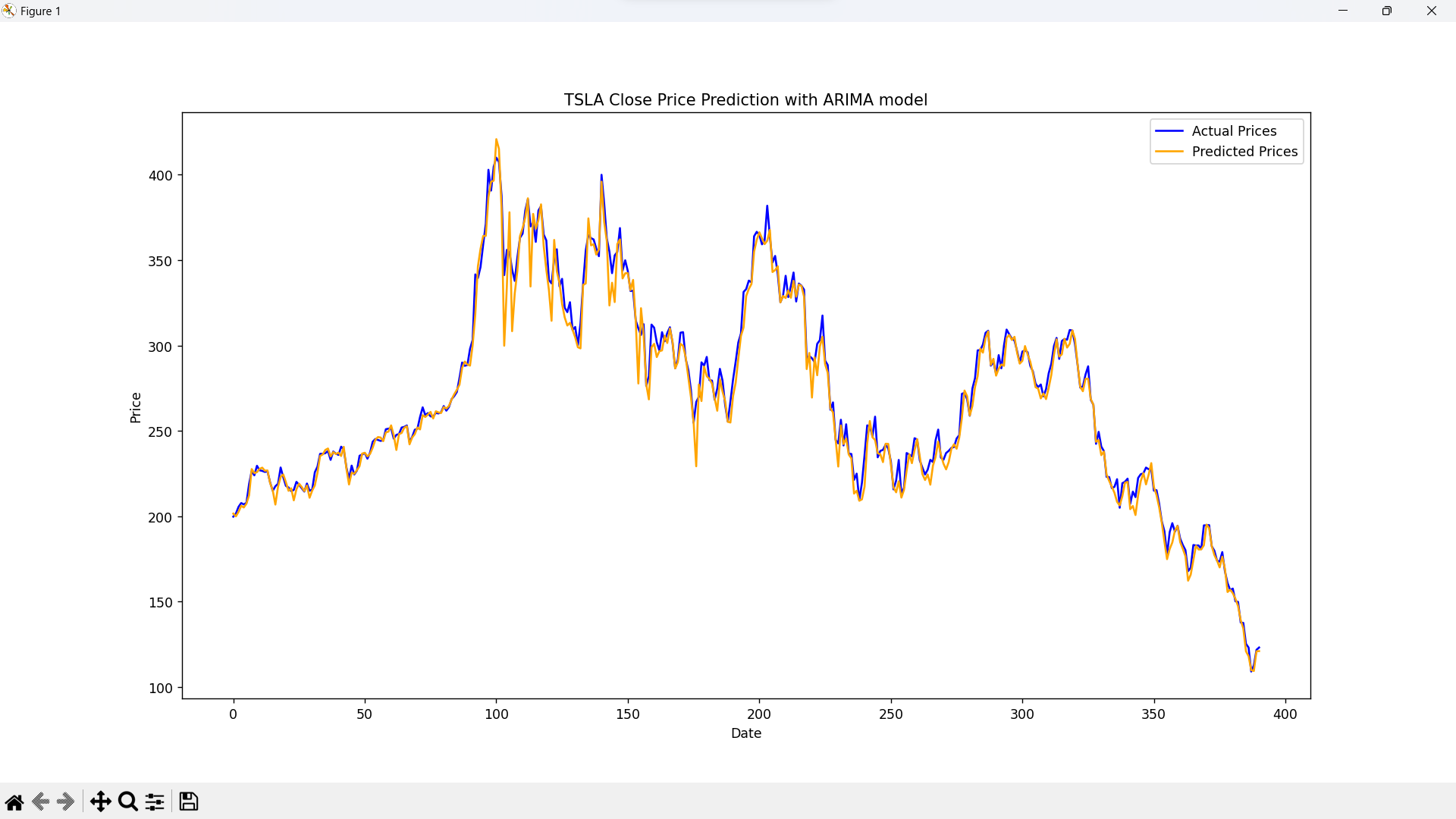
I have tested the ARIMA models, using these parameters:



Result:

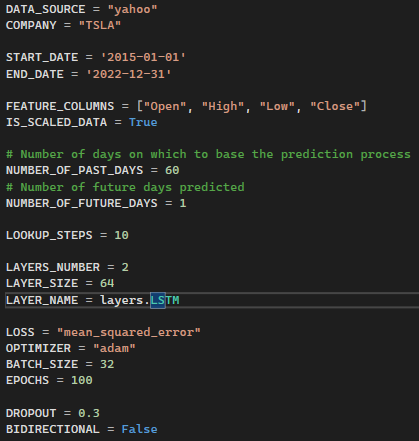
There was a total of 390 models created for predicting 390 corresponding days of the output:



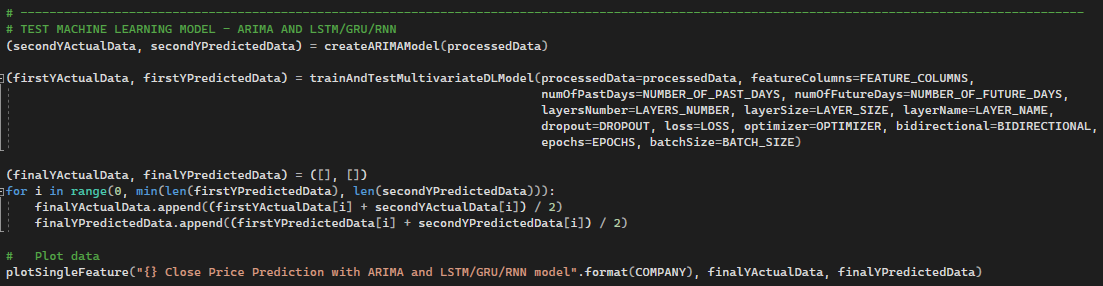


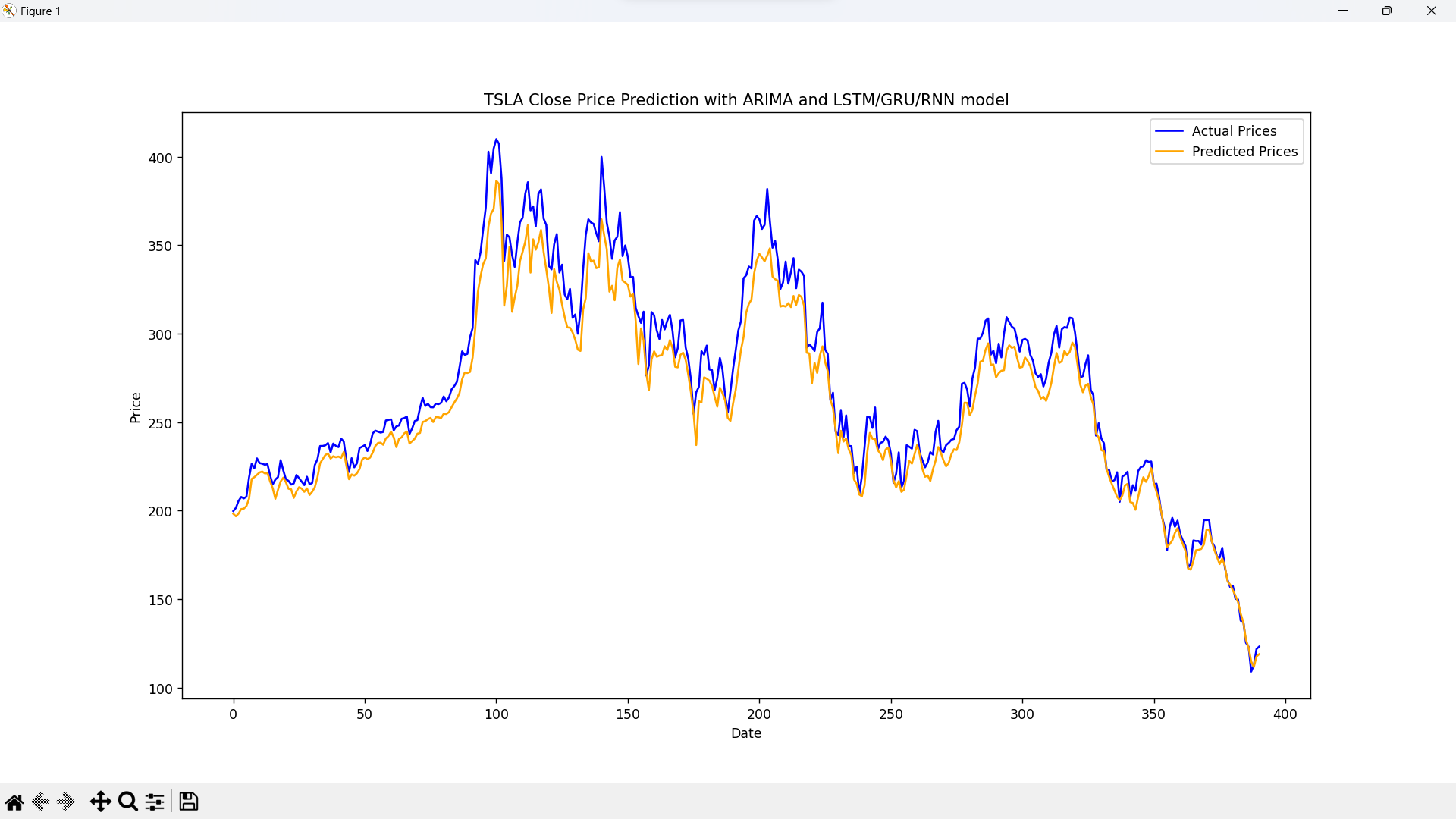
**3. Ensemble with LSTM/GRU/RNN model**

* For testing the ensembled model of ARIMA and LSTM (multivariate), I used the parameters as below:

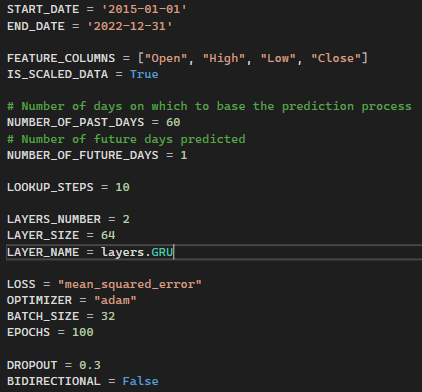


Then plotted the average value of the two predicted data:

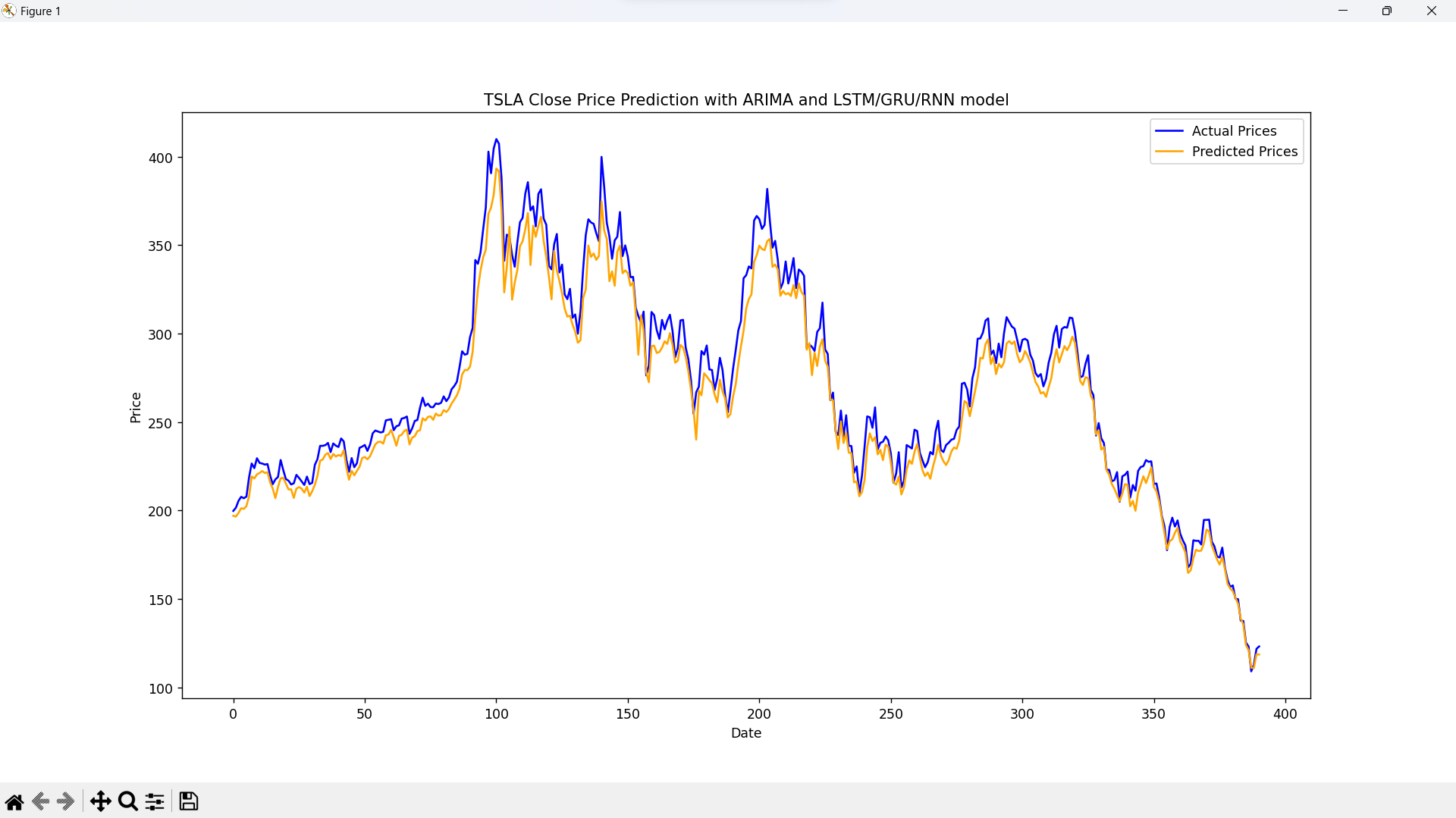




* Then, I have tested to ensemble the ARIMA model and GRU model, using the following parameters:



Result:



**4. Summary**

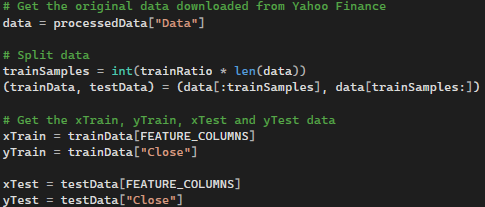
Through the output from the above test, even though models like ARIMA is under-estimated compared to LSTM/GRU model, but when I using many ARIMA models, each takes the responsibility to predict the price data of one corresponding day, the output seems to be much more accurate.

**RANDOM FOREST REGRESSOR**

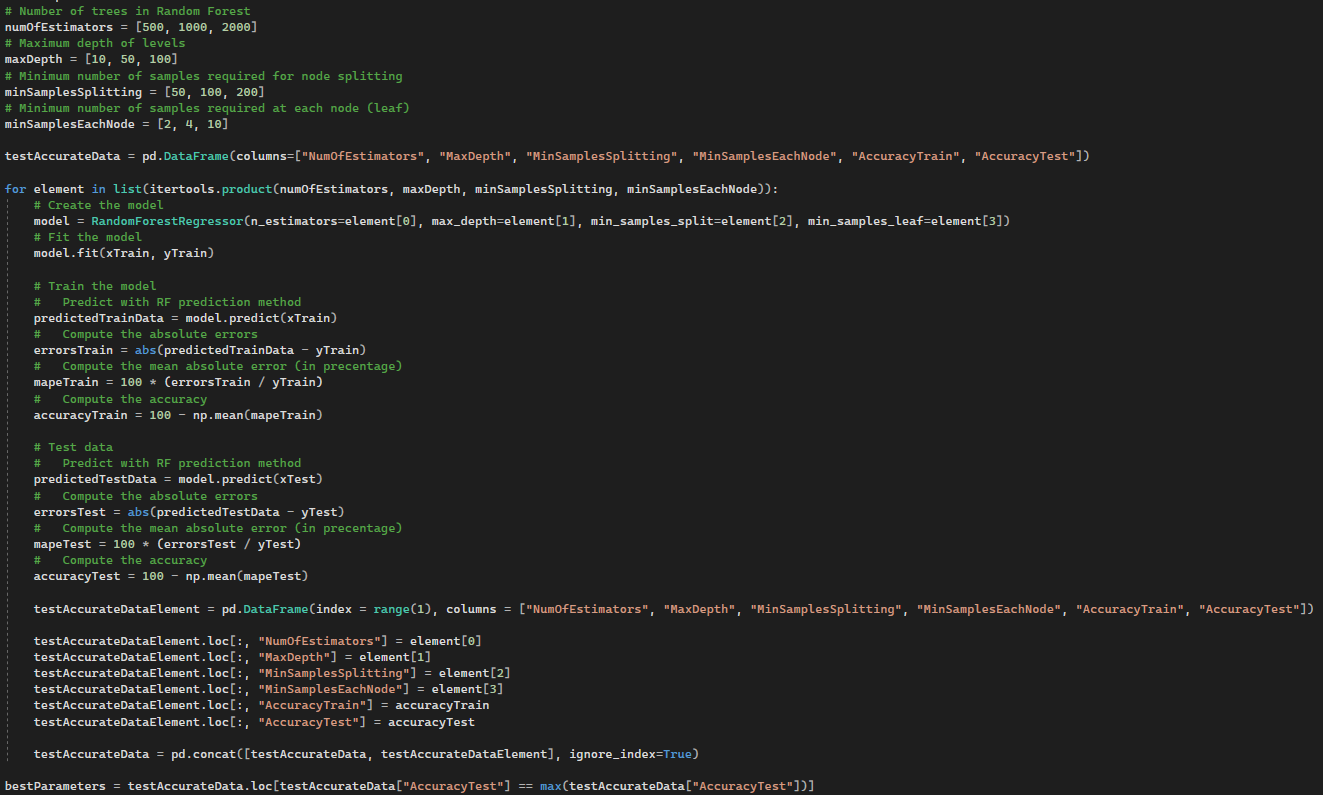
Random forest is an ensemble technique of the Decision Tree Algorithms. The key of this algorithm is that the number of decision trees are created from different bootstrap samples.

**1. “createRTModel()” method**

For data preparation for doing this model, from the original data downloaded from Yahoo Finance, I have taken the columns in “FEATURE\_COLUMNS” for the “xTrain” and “xTest”, and taken “Close” column for the “yTrain” and “yTest”, after having splitted it by the “TRAIN\_RATIO”:



After that, if the model had not been previously saved, I would find the best parameters for the RandomForestRegressor model:



First, I will randomly initialize some possibles values for the four necessary parameters: “numOfEstimators” – number of tree in RT, “maxDepth” – maximum depth of levels in RT, “minSamplesSplitting” – minimum number of samples required for node splitting, “minSampleEachNode” – minimum number of samples needed at each leaf/node of the decision tree.

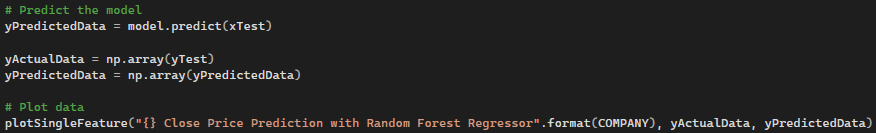
For each combination of hyperparameters, it trains a RandomForestRegressor model on the given training data (“xTrain” and “yTran”) and evaluates its performance on both the training and testing datasets, calculating accuracy scores for both datasets based on mean absolute percentage error (MAPE), before storing the results of those hyperparameters sets in a “pandas” Dataframe named “testAccurateData”

As using the “accuracyTrain” may cause overfitting issue, the method identifies the best set of hyperparameters based on the highest accuracy score on the **testing** data.

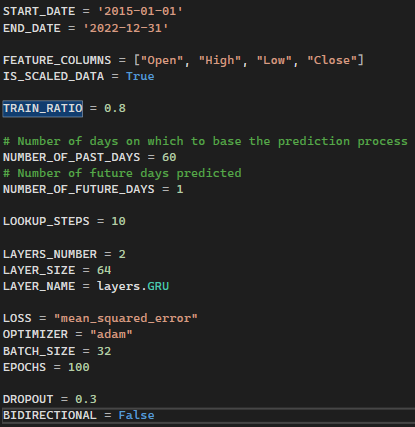
It then creates a new Random Forest Regressor model using the best hyperparameters and trains it on the entire training dataset. The trained model is saved to a file using "joblib".

**2. Test independently**

Whether the model was loaded or trained, after the above stages, the method proceeds to make predictions on the testing dataset (xTest) and plots the actual vs. predicted closing prices using a function called plotSingleFeature:



For testing the model of Random Forest independently, I have used the similar parameter to easily conclude:

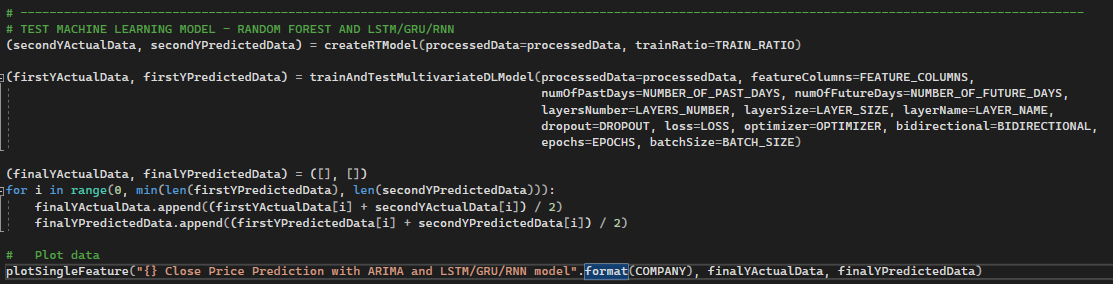


And, the output is shown in the below illustrations:



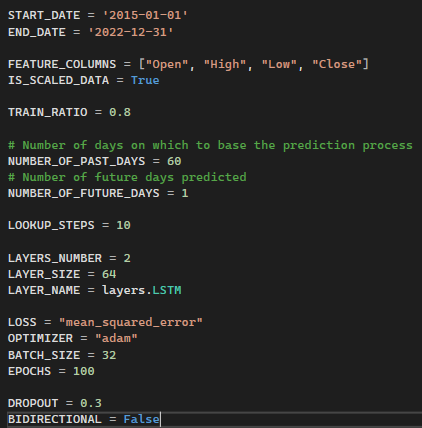
**3. Ensemble with LSTM/GRU/RNN model**

It is similar with ensembling LSTM/GRU model with ARIME mentioned above, I have predicted the Closing price independently using the two types of model, then finalise the output by getting the average values:

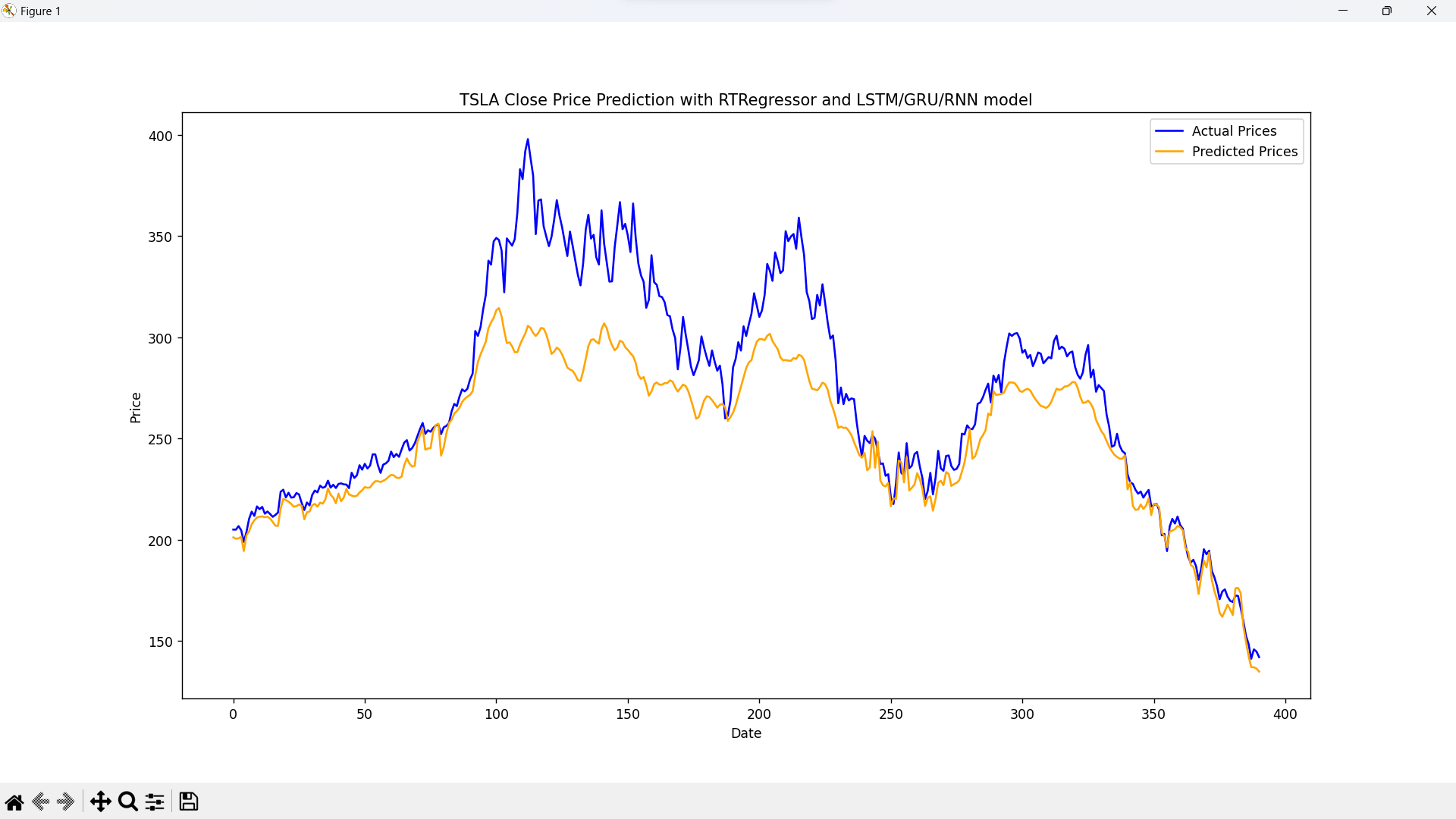


Then, I have remained the global parameters as the above test.

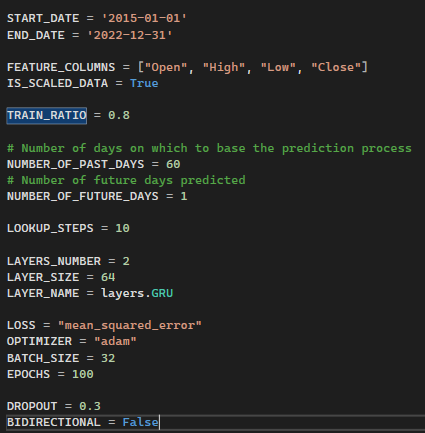
* Ensemble with LSTM model:



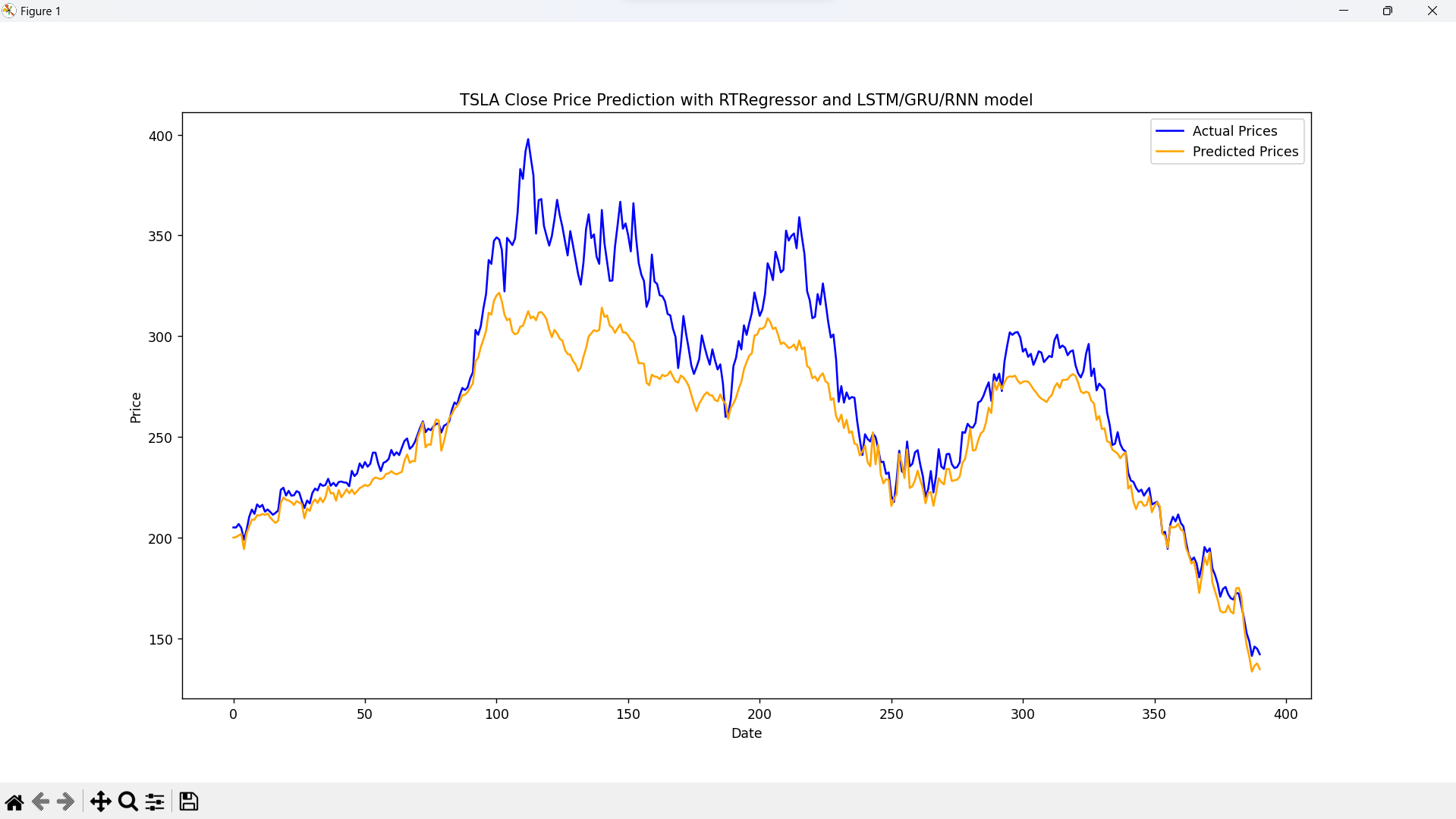
Output:



* Ensemble with GRU model:

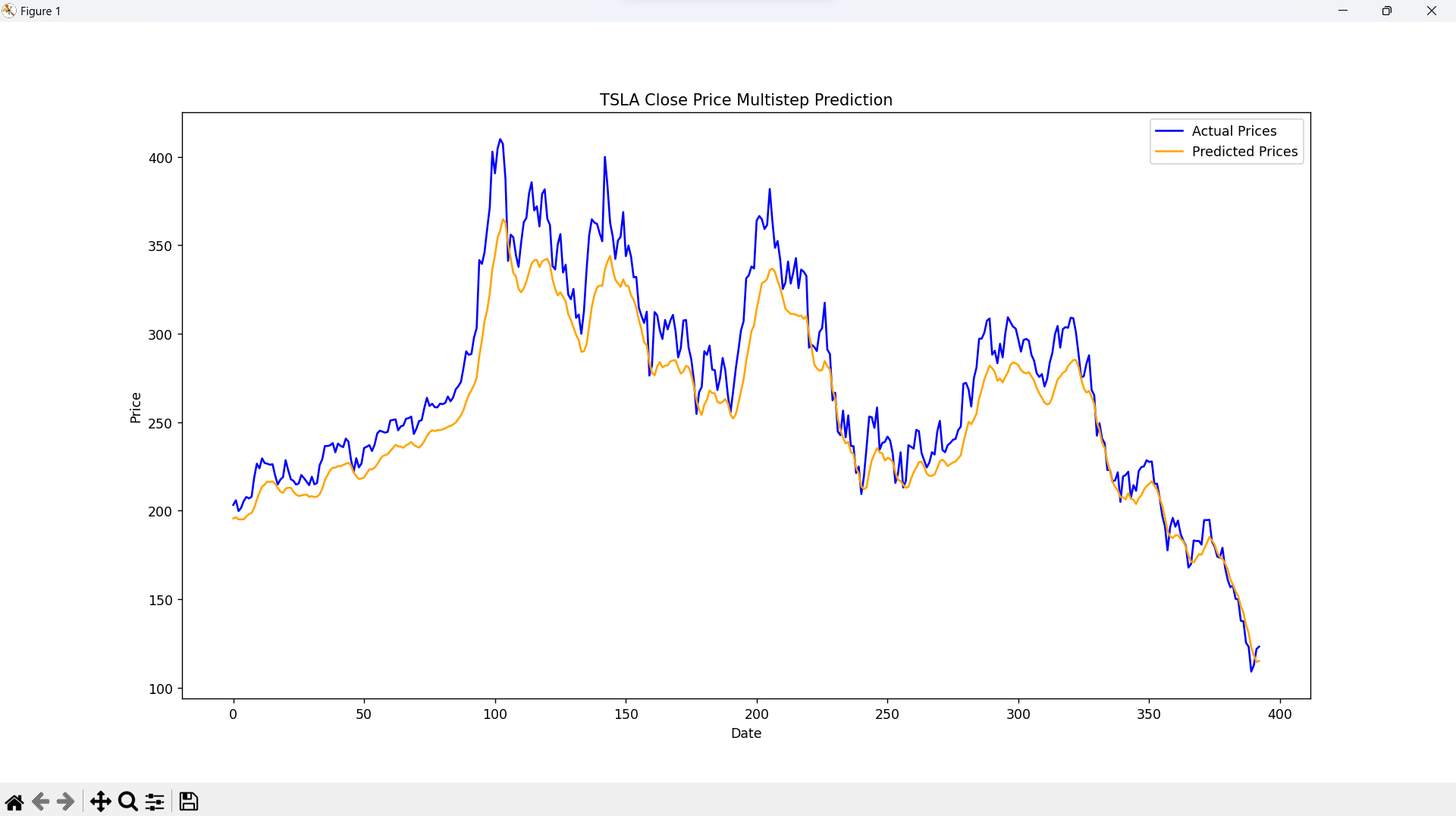


Output:



**4. Summary**

From the result of the tests shown in the above diagrams of graphs, and the previous task’s ones, the Random Forest Regressor shows the great weakness in prediction compared to LSTM/GRU model in the fluctuation events, even with the hyperparameter tuning method to find the good parameters for the model. On the other hand, its predicted data seems to be more accurate during the other periods.



*Output from a prediction with GRU layer (Previous task)*

**CONCLUSION**

With the above testing of ensembling ARIMA and Random Forest Regressor models with LSTM/GRU models, it is clear that the accuracy of the prediction data has increased dramatically when ensembling with multiple ARIMA models, in comparison with using LSTM/GRU layers independently. Whereas the Random Forest performed quite bad in predicting significant fluctuation in the data.

**REFERENCES**

[1] R. Saha, “Forecasting of Stock Market using ARIMA in Python,” Medium, Apr. 27, 2022. <https://medium.com/@raj.saha3382/forecasting-of-stock-market-using-arima-in-python-cd4fe76fc58a>

[2] I. D. Baruah, “Combining Time Series Analysis with Artificial Intelligence: The Future of Forecasting,” Analytics Vidhya, Jul. 21, 2021. <https://medium.com/analytics-vidhya/combining-time-series-analysis-with-artificial-intelligence-the-future-of-forecasting-5196f57db913>