ANSHUL VATS

104491909

INTELLIGENT SYSTEMS – COS30018

REPORT TASK B.2 – DATA PROCESSING-1 (PROJECT OPTION-B)

For the task B.2, data processing for the stock prediction model has been done as per required accordingly. The changes and the amendments have been successfully done for the existing python files.

Below are some of the changes that we conducted in the codes for the desired functionality of the code.

1. Parameters.py
   New Concepts Introduced:

   Dynamic Date Handling: Introduce a mechanism for dynamical dates of stock data downloading. This enables users can enter their own start date and end date when working on data processing making the pipeline flexible.
   Configuration Variables: Set and defined other parameters which included sequence length which was equal to N_STEPS and the value of the look up step which was equal to LOOKUP_STEP The model parameters had also been defined. This makes configuration of the model and data processing straightforward as shown by the following configurations.
   Details:

   N_STEPS and LOOKUP_STEP: The first two control the number of prior days the model learns from whereas the latter one determines the number of days for which the model makes predictions in the future.
   SCALE, SHUFFLE, SPLIT_BY_DATE, TEST_SIZE: All of these options allow for scaling feature columns, shuffling data, splitting by date and also defining the size of the test set.
   FEATURE_COLUMNS: Determines which fields are to be selected in the creation of the model so that flexibility is obtained in the selection process.

2. stock_prediction.py
   New Concepts Introduced:

   Date Input Functionality: At the window, included the options of inputting dates that mark the beginning and end of stock data pulled. This feature offers a window of control over the time that is spent analysing the data.
   Data Preprocessing Enhancements: Enhanced data handling concerning missing values and

scaling of the input features in order to optimize the process.
Details:

load_data Function: Altered to accept start and end dates, to search for the data in this interval and for scaling and data cleaning. This gives the function the flexibility to be able to work with a variety of analysis periods as well as produces higher quality data.
shuffle_in_unison Function: Intended to resize the datasets so that they align with each other and feature-target pairs are not divided between two different shuffles.
create_model Function: Enhanced for multiple model configurations such as bidirectional LSTM and the dropout layer. Of course, flexibility in tuning of the model is also helpful when a user would like to make it meet specific requirements.

3. train.py
New Concepts Introduced:

Model Checkpointing and TensorBoard: Included callbacks for model checkpoints and TensorBoard for monitoring training and for the purpose of saving the best model weights. The following are the benefits of these features: It increases the level of monitoring to help manage the models.
Dynamic Model Loading: Add functionality to read weights of the model in case the training was interrupted/ paused and continue training from a certain point as well as evaluate on new data.
Details:

ModelCheckpoint: Helps in saving the weights of the model thereby making sure that the best model is kept safe. This is important since we do not want to over-fit our network and also in preservation of the best state of our model.
TensorBoard: Serves as a way to monitor training statistics and observe problems, in addition to displaying model evolution.
Dynamic Loading: Enables to resume training from a state of the last checkpoint, which is convenient when training for a long time or making gradual changes.
4. test.py
New Concepts Introduced:

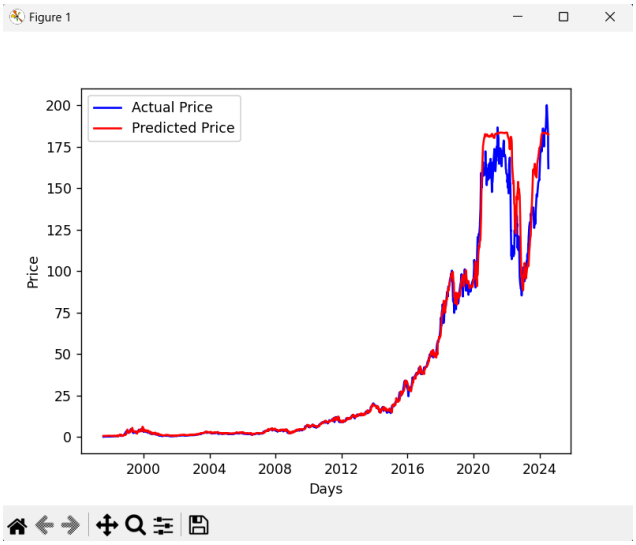
Evaluation and Visualization: Brought in the extra features for training the model on the test data and to make predictions as well as to visualize the predictions in the form of plots. This also gives information concerning the performance as well as the accuracy of the models that have been developed.
Profit Calculation: Put into practice procedures to compute the probable gains or losses arising from trading options depending on the models. This improves the applicability of the predictions as well as the craft of call-centre simulation.
Details:

plot_graph Function: Works to plot the actual stock price against the modeled one, which in its turn can be effective to visually evaluate its efficacy and to define patterns as well as differences.
get_final_df Function: Generates a new DataFrame with true prices, predicted prices and

respective profit margining. This function is important for assessing the financial consequences of the predictions and the efficiency of models in reality.
Each of these enhancements and new concepts enhance the flexibility, performance and generality of the stock prediction system and the improved ability to dissect the model forecasts to support better decision making.

These were some of the total changes we have made to the codes, though the changes have been successfully committed to the github and are also well-commented. In order to complete the tasks, obviously we needed to research and also some of the python syntax to get the concept and how to reach to that particular level of completion of the code. Yet, efforts have been made to complete the task B.2.

Below is the result output's screenshots:



As the graph shows now the plotting in two different lines i.e. blue and red. Representing both the actual price as well as the predicted price.