Report : Stock Market Prediction using LSTM Model
By- Swagata Naskar (19EC39038)

Long-term short-term memory (LSTM) is a special type of Recurrent Neural Network (RNN) that can retain important information over time using memory cells. This LSTM property makes it an excellent algorithm for learning relational patterns and can help generate solutions such as translation, sales timelines, plotting, inaccuracies, next-word suggestions, etc. To pull the data for any stock we can use a library named 'nsepy'

- **Preparing the data**

The LSTM model will require data input in the form of X Vs y. Where X will represent the price of last 10 days and y will represent the price of 11 days. By looking at many such examples over the past 2 years, LSTM will be able to learn price movements. Therefore, when we go through the last ten days of the price, it will be able to predict the closing price of the stock tomorrow. Since LSTM is an algorithm based on neural networks, sorting or organizing the data is a requirement for fast and efficient fitting.

- **Preparing the data for LSTM**

- **Splitting the data into training and testing**

Keeping last few days of data to test the learnings of the model and rest for training the model. Here I am choosing Last 5 days as testing.

- **Visualizing the input and output data for LSTM**

Printing some sample input and output values to help you visualize how the LSTM model will learn the prices. You can see the input is a 3D array of the last 10 prices and the output is a 1D array of the next price.

- **Creating the Deep Learning LSTM model**

Look at the use of the LSTM function instead of Dense to define the hidden layers. The output layer has one neuron as we are predicting the next day price, if you want to predict for multiple days, then change the input data and neurons equal to the number of days of forecast.
In the below code snippet I have used three hidden LSTM layers and one output layer. You can choose more layers if you don't get accuracy for your data. Similarly you can increase or decrease the number of neurons in the hidden layer. Just keep in mind, the more neurons and layers you use, the slower the model becomes. Because now there are many more computations to be done.
Each layer has some hyperparameters which needs to be tuned. Take a look at some of the important hyperparameters of LSTM below units=10: This means we are creating a layer with ten neurons in it. Each of these five neurons will be receiving the values of inputs.

input_shape = (TimeSteps, TotalFeatures): The input expected by LSTM is in 3D format. Our training data has a shape of (420, 10, 1) this is in the form of (number of samples, time steps, number of features). This means we have 420 examples to learn in training data, each example looks back 10-steps in time like what was the stock price yesterday, the day before yesterday so on till last 10 days. This is known as Time steps. The last number '1' represents the number of features. Here we are using just one column 'Closing Stock Price' hence its equal to '1'

kernel_initializer='uniform': When the Neurons start their computation, some algorithm has to decide the value for each weight. This parameter specifies that. You can choose different values for it like 'normal' or 'glorot_uniform'.

activation='relu': This specifies the activation function for the calculations inside each neuron. You can choose values like 'relu', 'tanh', 'sigmoid', etc.

return_sequences=True: LSTMs backpropagate thru time, hence they return the values of the output from each time step to the next hidden layer. This keeps the expected input of the next hidden layer in the 3D format. This parameter is False for the last hidden layer because now it does not have to return a 3D output to the final Dense layer.
optimizer='adam': This parameter helps to find the optimum values of each weight in the neural network. 'adam' is one of the most useful optimizers, another one is 'rmsprop'

batch_size=10: This specifies how many rows will be passed to the Network in one go after which the SSE calculation will begin and the neural network will start adjusting its weights based on the errors.
When all the rows are passed in the batches of 10 rows each as specified in this parameter, then we call that 1-epoch. Or one full data cycle. This is also known as mini-batch gradient descent. A small value of batch_size will make the LSTM look at the data slowly, like 2 rows at a time or 4 rows at a time which could lead to overfitting, as compared to a large value like 20 or 50 rows at a time, which will make the LSTM look at the data fast which could lead to underfitting. Hence a proper value must be chosen using hyperparameter tuning.

Epochs=10: The same activity of adjusting weights continues for 10 times, as specified by this parameter. In simple terms, the LSTM looks at the full training data 10 times and adjusts its weights.

- **Measuring the accuracy of the model on testing data**

Now using the trained model, we are checking if the predicted prices for the last 5 days are close to the actual prices or not.
Notice the inverse transform of the predictions. Since we normalized the data before the model training, the predictions on testing data will also be normalized, hence the inverse transformation will bring the values to the original scale. Then only we must calculate the percentage accuracy.

- **Visualizing the predictions for full data**

Plotting the training and testing data both to see how good the LSTM model has fitted.

- **How to predict the stock price for tomorrow**

If you want to predict the price for tomorrow, all you have to do is to pass the last 10 day's prices to the model in 3D format as it was used in the training. The below snippet shows you how to take the last 10 prices manually and do a single prediction for the next price.

- **Conclusion**

This prediction is only short-term. The moment you try to predict for multiple days like the next 30-days or 60 days, this fails miserably. Not because our LSTM model is bad, but, because Stock markets are highly volatile. So, don't bet your money simply on this model! Do some research and then use this model as a complementary tool for analysis!