STOCK PRICE PREDICTOR

Using Long-Short Term Memory Networks

Mohit Vernekar
MSc IT A-7

Table of Content

DEFINITION

Project Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process. Can we predict stock prices with machine learning? Investors make educated guesses by analyzing data. They will read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theories is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. We have this idea of a trading floor being filled with adrenaline infuse men with loose ties running around yelling something into a phone but these days they are more likely to see rows of machine learning experts quietly sitting in front of computer screens. In fact, about 70% of all orders on Wall Street are now placed by software, we are now living in the age of the algorithm. This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with timeframes recurrent neural networks (RNNs) come in handy but recent research have shown that LSTM, networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

Problem Statement

The challenge of this project is to accurately predict the future closing value of a given stock across a given period in the future. For this project I will use a Long Short Term Memory networks – usually just called "LSTMs" to predict the closing price of Tata Consumer Products using a dataset of past prices.

GOALS:

1. Explore stock prices.
2. Implement basic model using linear regression.
3. Implement LSTM using Keras library.
4. Compare the results and submit the report.

[1] Long Short-Term Memory networks

ANALYSIS

Data Exploration

The data used in this project is of the Tata Consumer Products from January 1, 2016 to March 5, 2021, this is a series of data points indexed in time order or a time series. My goal was to predict the future closing price after training. For ease of reproducibility and reusability, all data was pulled from Yahoo Finance. The prediction must be made for Closing (Adjusted closing) price of the data. Since Yahoo Finance already adjusts the closing prices for us, we just need to make prediction for "CLOSE" price.

The dataset is of following form:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-01-01	146.050003	148.800003	145.500000	146.399994	136.011261	781067.0
1	2016-01-04	145.899994	147.199997	142.250000	143.699997	133.502853	950484.0
2	2016-01-05	144.000000	149.500000	142.199997	148.350006	137.822906	1502760.0
3	2016-01-06	147.100006	149.800003	144.500000	145.250000	134.942871	834054.0
4	2016-01-07	144.100006	144.699997	139.000000	140.050003	130.111877	1034747.0

Note: I did not observe any abnormality in datasets, i.e., no feature is empty and does not contains any incorrect value as negative values.

[2] Tata Consumer Products

[3] Yahoo Finance

Exploratory Visualization

To visualize the data, I have used matplotlib library. I have plotted Closing stock price of the data with the number of years. Following is the snapshot of the plotted data:

X-axis: Represents Years
Y-axis: Represents Closing Price

Through this data, we can see a dynamic growth of Tata Consumer Products' share prices.

[4] Matplotlib

Algorithms and Techniques

The goal of this project was to study time-series data and explore as many options as possible to accurately predict the Stock Price. Through my research, I came to know about Recurrent Neural Nets (RNN) which are used specifically for sequence and pattern learning. As they are networks with loops in them, allowing information to persist and thus ability to memorize the data accurately. But Recurrent Neural Nets have vanishing Gradient descent problem which does not allow it to learn from past data as was expected. The remedy of this problem was solved in Long-Short Term Memory Networks , usually referred as LSTMs. These are a special kind of RNN, capable of learning long-term dependencies.

In addition to adjusting the architecture of the Neural Network, the following full set of parameters can be tuned to optimize the prediction model:

- Input Parameters
  - Preprocessing and Normalization
- Neural Network Architecture
  - Number of Layers (how many layers of nodes in the model; used 3)
  - Number of Nodes (how many nodes per layer; tested 1,3,8, 16, 32, 64, 100,128)
- Training Parameters
  - Training / Test Split (how much of dataset to train versus test model on; kept constant at 82.95% and 17.05% for LSTM model)
  - Validation Sets (kept constant at 0.05% of training sets)
  - Batch Size (how many time steps to include during a single training step; kept at 32)
  - Optimizer Function (which function to optimize by minimizing error; used “Adam” throughout)
  - Epochs (how many times to run through the training process, kept at 100)

[5] Recurrent Neural Network

METHODOLOGY

Data Pre-processing

Acquiring and pre-processing the data for this project occurred in the following sequence:

1. Downloaded the dataset from Yahoo Finance and saved it as **TATACONSUM.csv** file in the following format:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-01-01	146.050003	148.800003	145.500000	146.399994	136.011261	781067.0
1	2016-01-04	145.899994	147.199997	142.250000	143.699997	133.502853	950484.0
2	2016-01-05	144.000000	149.500000	142.199997	148.350006	137.822906	1502760.0
3	2016-01-06	147.100006	149.800003	144.500000	145.250000	134.942871	834054.0
4	2016-01-07	144.100006	144.699997	139.000000	140.050003	130.111877	1034747.0

2. Sorted the dataset on date time and filtered “Date” and “Close” columns.
3. Analyzed the closing prices from Dataframe.
4. Normalized the new filtered dataset using **MinMaxScaler** helper function from **Scikit-Learn**.
5. Split the dataset into the training and test datasets for linear regression model.
6. Split the dataset into the training (82.95%) and test (17.05%) datasets for LSTM model.

[6] MinMaxScaler
[7] Scikit-Learn

Implementation

Some code implementation insight:

1. Incorporating Timesteps Into Data

```
for i in range(60,len(train_data)):
    x_train_data.append(scaled_data[i-60:i,0])
    y_train_data.append(scaled_data[i,0])
```

Inputted the data in the form of a 3D array to the LSTM model. First, I created data in 60 timesteps before using **NumPy** to convert it into an array. Finally, I converted the data into a 3D array with x_train samples, 60 timesteps, and one feature at each step.

2. Creating the LSTM Model
   - Imported from Keras: Sequential for initializing the neural network, LSTM to add the LSTM layer, Dropout for preventing overfitting with dropout layers, and Dense to add a densely connected neural network layer.

```
from keras.models import Sequential

from keras.layers import LSTM,Dropout,Dense
```

   - Added LSTM layer with 50 units, which is the dimensionality of the output space, set the return_sequences=True for stacking LSTM layers so the consequent LSTM layer has a three-dimensional sequence input, and used input_shape for the shape of the training dataset.

```
lstm_model=Sequential()
lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train_data.shape[1],1)))
lstm_model.add(LSTM(units=50))
lstm_model.add(Dense(1))
```

   - To compile my model, I used the **Adam** optimizer and set the loss as the mean_squared_error. After that, I fitted the model to run for 100

epochs (the epochs are the number of times the learning algorithm will work through the entire training set) with a batch size of 32.

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=100,batch_size=32,verbose=2)
```

3. Making Predictions on the Test Set

```
X_test=[]

for i in range(60,inputs_data.shape[0]):
    X_test.append(inputs_data[i-60:i,0])

X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))

predicted_closing_price=lstm_model.predict(X_test)
predicted_closing_price=scaler.inverse_transform(predicted_closing_price)
```

4. Plotting the Results

   I then used matplotlib to visualize the result of the predicted stock price and the actual stock price.

```
train_data=new_dataset[:900]
valid_data=new_dataset[900:]
valid_data['Predictions']=predicted_closing_price
plt.plot(train_data["Close"])
plt.plot(valid_data[['Close',"Predictions"]])
```

[8] NumPy

[9] Adam Optimizer

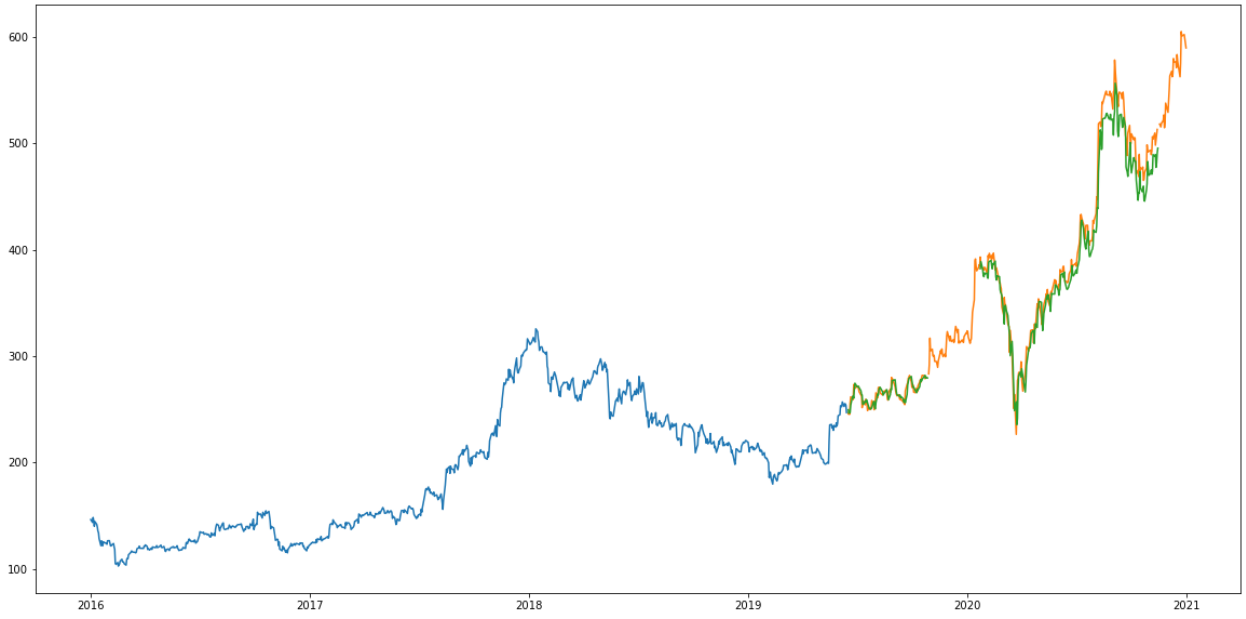
Refinement

For this project I have worked on fine tuning parameters of LSTM to get better predictions. I did the improvement by testing and analysing each parameter and then selecting the final value for each of them.
To improve LSTM, I have done the following:

- Increased batch size from 1 to 32
- Increased epochs from 1 to 100
- Added verbose = 2
- Made prediction with the batch size

Result

The graph is as follows:



As it can be observed, the LSTM has predicted stocks almost similar to actual stocks. Although the predictions are not completely accurate, the model, however, did still indicate overall trends such as going up or down. Thus, LSTMs can be effective in times series forecasting.

[10] Time Series Forecasting

CONCLUSION

Free-Form Visualization

I have already discussed all the important features of the datasets and their visualisation in one of the above sections. But to conclude my report, I would choose my final model visualization, which is the improved version of LSTM by fine tuning parameters.

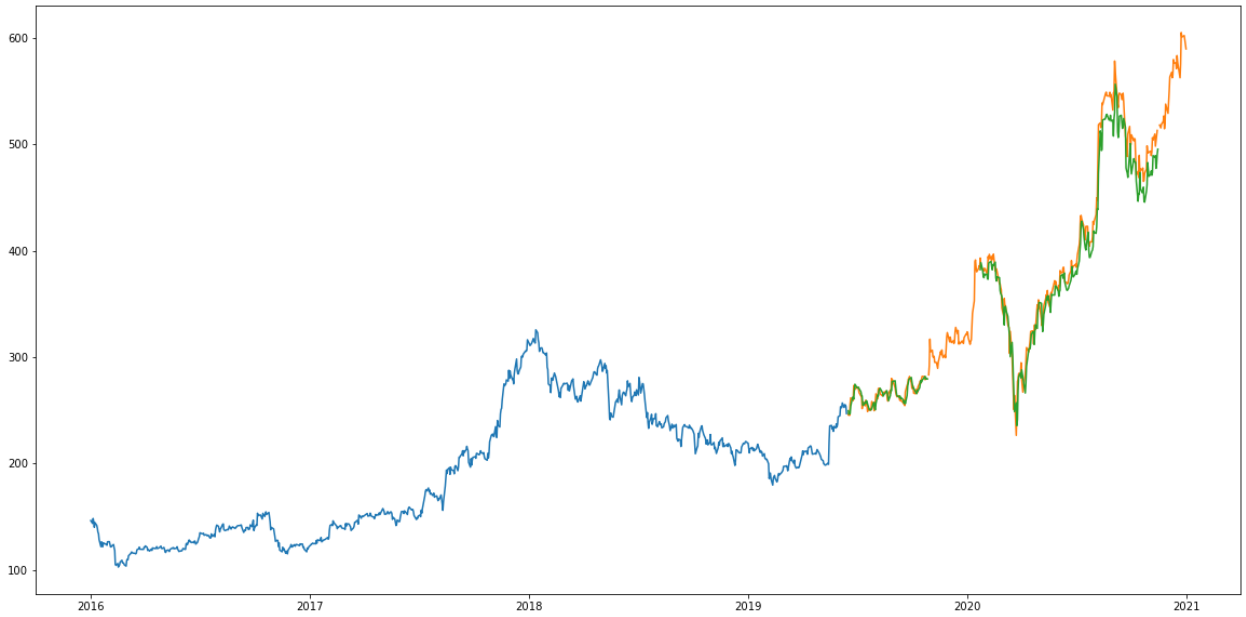


Fig: Plot of Improved Long-Short Term Memory Model

[11] Overfitting and Underfitting in LSTM

Reflection

To recap the process undertaken in this project:

- Set Up Infrastructure
  - Jupyter Notebook
  - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib, Sklearn, Numpy)
- Prepare Dataset
  - Incorporate data of Tata Consumer Products
  - Process the requested data into Pandas Dataframe
  - Develop function for normalizing data
  - Dataset used with an 80/20 split on training and test data
- Develop LSTM Model
  - Set up LSTM model with Keras
  - Improve the LSTM Model
  - Develop, document, and compare results using additional labels for the LSTM model
- Plot Actual and LSTM predicted values
- Analyze and describe results for the report

The final model really exceeded my expectations and have worked remarkably well. I am greatly satisfied with the results. The major problem I faced during the implementation of this project was exploring the data.
It was the toughest task. To convert data from raw format to pre-process data and then to split them into training and test data. All of these steps require a great deal of patience and very precise approach.

Improvement

As there is scope of improvement in each individual, so is the case with this project. This project though predicts closing prices with very minimum error, still there are many things that are lagging in this project. Two of the most important things are:

1. There is no user interaction or interface provided in this project. A UI can be provided where the user can check the value for future dates.
2. The stocks used for this project are only of Tata Consumer Products, I can surely add more companies in the list so as to make this project more comprehensive.

I would definitely like to add these improvements to this project in future.