Project Report
on

STOCK PRICE PREDICTOR

Using Long-Short Term Memory Networks

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING JAIPUR

Project By
HARSHIT BHARDWAJ
DEEKSHANT JAIN

Table of Content

DEFINITION

Project Overview

Investment firms, money 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 actually predict stock prices with machine learning? Investors make educated guesses by analyzing data. They'll 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 In fact about 70% of all orders on Wall Street are now placed by software, we're 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 researches have shown that LSTM, networks are the most popular and useful variants of RNNs.

We 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 of time 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 the S&P 500[1] 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.

Model Evaluation

For this project measure of performance will be using the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) calculated as the difference between predicted and actual values of the target stock at adjusted close price and the delta between the performance of the benchmark model (Linear Regression) and our primary model (Deep Learning).

ANALYSIS

Data Exploration

The data used in this project is of the Alphabet I**nc** from January 1, 2005 to June 20, 2017, this is a series of data points indexed in time order or a time series. My goal was to predict the closing price for any given date after training. For ease of reproducibility and reusability, all data was pulled from the Google Finance Python API [4].

The prediction has to be made for Closing (Adjusted closing) price of the data. Since Google Finance already adjusts the closing prices for us [5], we just need to make prediction for "CLOSE" price.

The dataset is of following form :

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

Table: The whole data can be found out in 'Google.csv' in the project root folder

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

We can infer from this dataset that date, high and low values are not important features of the data. As it does not matter at what was the highest prices of the stock for a particular day or what was the lowest trading price. What matters is the opening price of the stock and closing prices of the stock. If at the end of the day

we have higher closing prices than the opening prices that we have some profit otherwise we saw losses. Also volume of share is important as a rising market should see rising volume, i.e, increasing price and decreasing volume show lack of interest, and this is a warning of a potential reversal. A price drop (or rise) on large volume is a stronger signal that something in the stock has fundamentally changed.

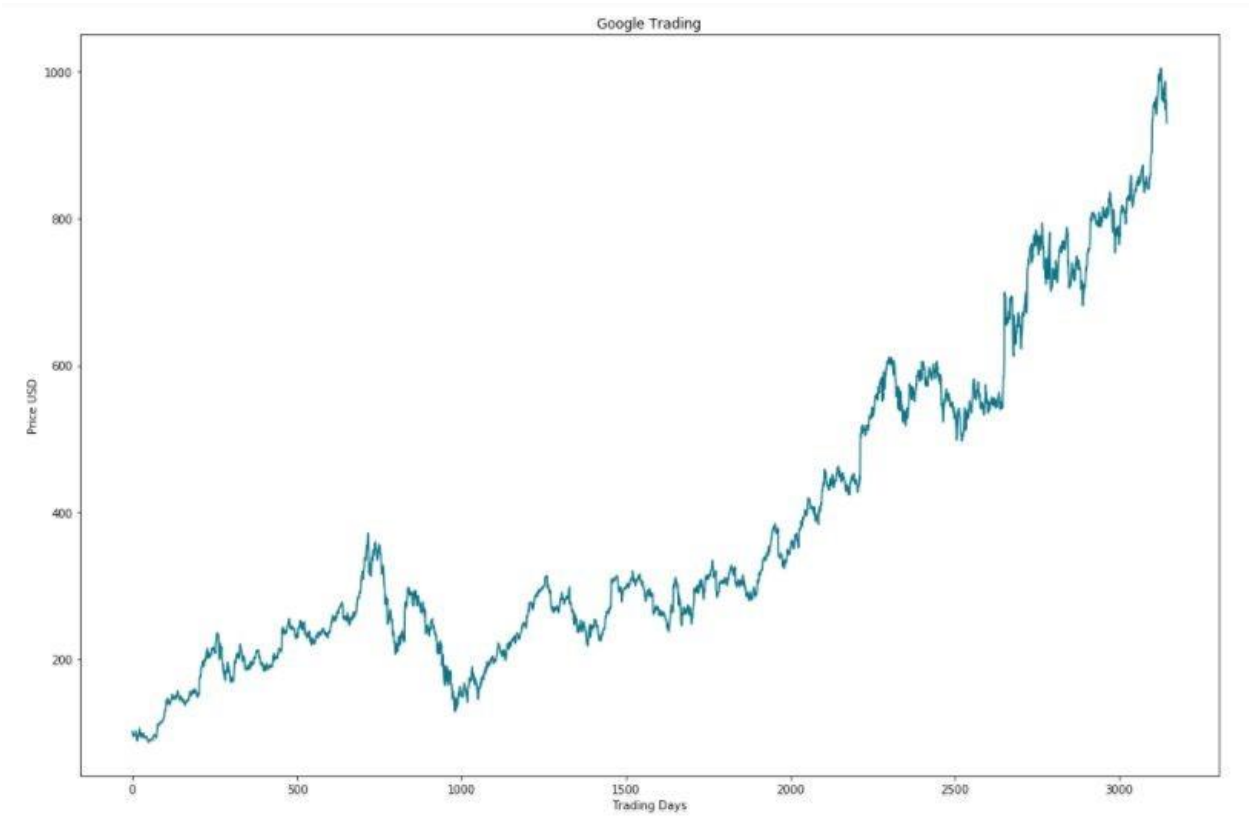
Therefore i have removed Date, High and low features from data set at preprocessing step.

Data Visualization

To visualize the data i have used matplotlib library. I have plotted Closing stock

price of the data with the no of items (no of days) available.

Following is the snapshot of the plotted data :



X-axis: Represents Tradings Days
Y-axis: Represents Closing Price In USD

Through this data we can see a continuous growth in Alphabet Inc. The major fall in the prices between 600-1000 might be because of the Global Financial Crisis of 2008-2009.

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 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 memorise 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 LSTM, 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 (see Data Preprocessing Section)
- 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)
- Optimizer Function (which function to optimize by minimizing error; used "Adam" throughout)
- Epochs (how many times to run through the training process; kept at 2 for base)

METHODOLOGY

Data Preprocessing

Acquiring and preprocessing the data for this project occurs in following sequence:

- Request the data from the Google Finance Python API and save it in google.csv file in the following format.

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

- Remove unimportant features(date, high and low) from the acquired data and reversed the order of data, i.e., from january 03, 2005 to june 30, 2005

Item	Open	Close	Volume
0	98.80	101.46	15860692
1	100.77	97.35	13762396
2	96.82	96.85	8239545
3	97.72	94.37	10389803

- Normalised the data using MinMaxScaler helper function from Scikit-Learn.

Item	Open	Close	Volume
0	0.012051	0.015141	0.377248
1	0.014198	0.010658	0.325644
2	0.009894	0.010112	0.189820
3	0.010874	0.007407	0.242701

- Stored the normalised data in google_preprocessed.csv file for future reusability.

- Splitted the dataset into the training and test datasets for LSTM model. The Split was of following shape: x_train (2589, 50, 3) y_train (2589,) x_test (446, 50, 3) y_test (446,)

Implementation

Once the data has been downloaded and preprocessed, the implementation process occurs consistently through models as follow:

We have thoroughly specified all the steps to build, train and test model and its predictions in the notebook itself.

Some code implementation insight:

LSTM model :

Step 1 : Split into train and test model :

Step 2 : Build an LSTM- model :

```
# # build basic lstm model
model = lstm.build_basic_model(input_dim = X_train.shape[-1],output_dim = unroll_length, return_sequences=True)

# # Compile the model
start = time.time()
model.compile(loss='mean_squared_error', optimizer='adam')
print('compilation time : ', time.time() - start)
```

Here I am calling a function defined in 'lstm.py' which builds the lstm model for the project.

Step 3: We now need to train our model.

```
model.fit(X_train,y_train,epochs=2,validation_split=0.05)
```

I have used here a built in library function to train the model.

Step 4: Now it's time to predict the prices for given test datasets.

```
predictions = model.predict(X_test)
```

I have used a built-in function to predict the outcomes of the model.

Step 5: Finally calculate the test score and plot the results of improved LSTM model.

```
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))
```

```
Train Score: 0.00021457 MSE (0.01464835 RMSE)
Test Score: 0.00088821 MSE (0.02980290 RMSE)
```

The predicted plot difference can be seen as follows:

Fig : Plot For Adjusted Close and Predicted Close Prices for basic LSTM model(epochs=2)

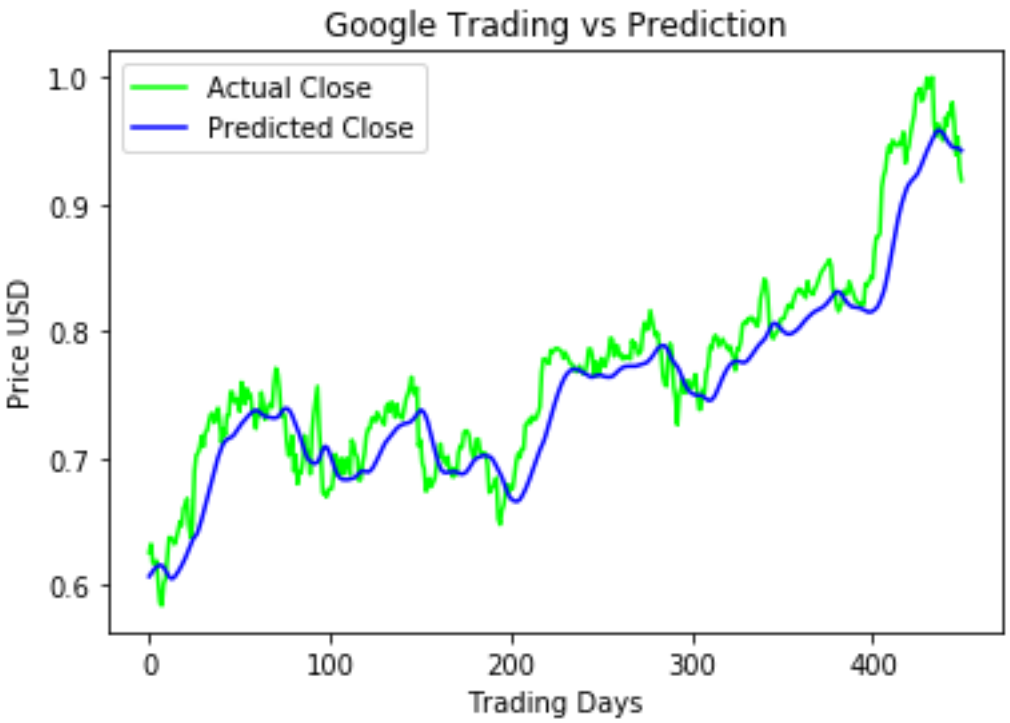
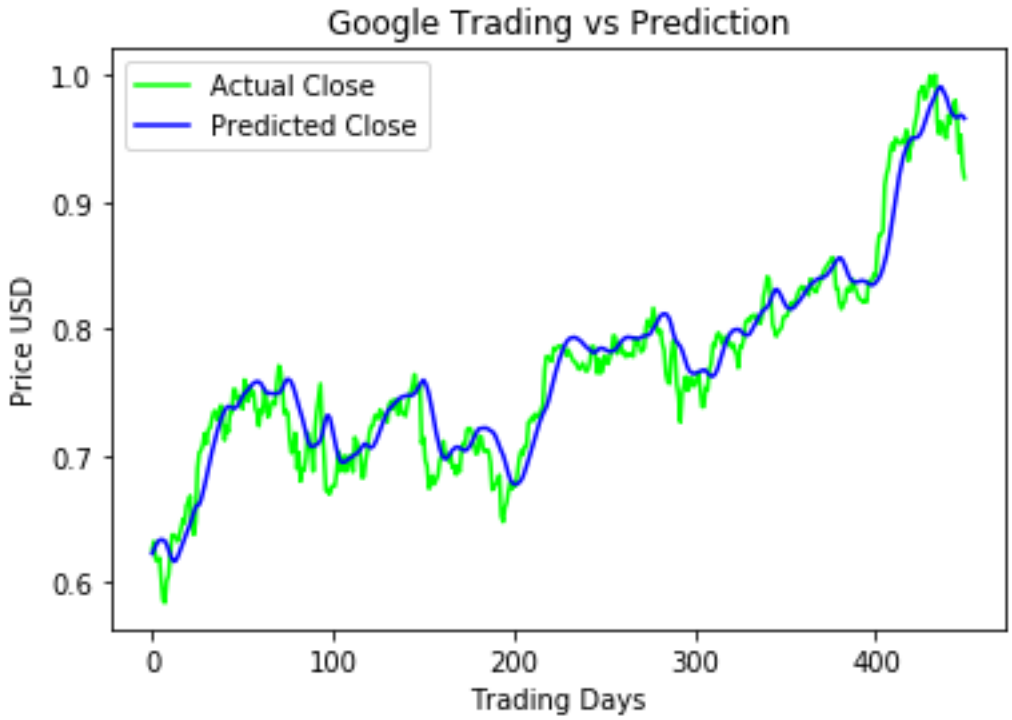


Fig : Plot For Adjusted Close and Predicted Close Prices for basic LSTM model(epochs=5)



RESULT

Model Evaluation and Validation
With each model i have refined and fined tune my predictions and have reduced mean squared error significantly.

- For my model using basic Long-Short Term memory model(epochs=10):
    - Train Score: 0.00016393 MSE (0.01464835 RMSE)
    - Test Score: 0.00044312 MSE (0.02980290 RMSE)

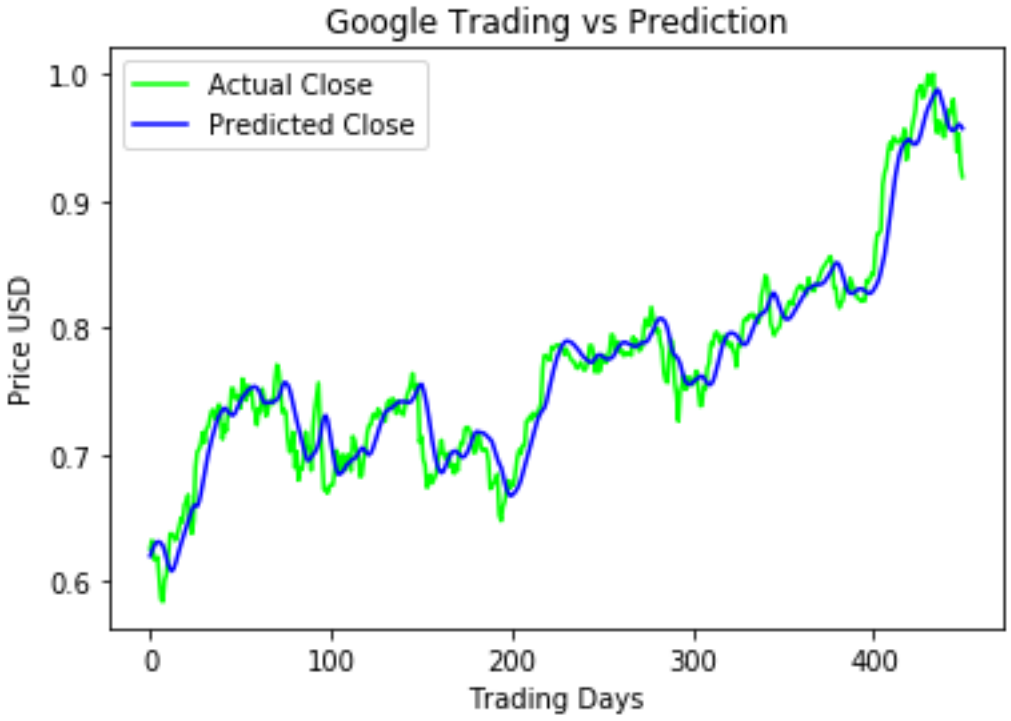


Fig: Plot of Long-Short Term Memory model (epochs=10)

CONCLUSION

There are several Model which suits to our problem

- **Support Vector Machine (SVM)**
- **Random Forest**
- **Multi-layer perceptron**

Free-Form Visualization
To conclude my report i would choose my final model visualization, which is LSTM by fine tuning parameters. As i was very impressed on seeing how close i have gotten to the actual data, with a mean square error of just 0.0009. It was an nice moment for me as i had to poke around a lot (really ALOT !! :P). But it was fun working on this project.

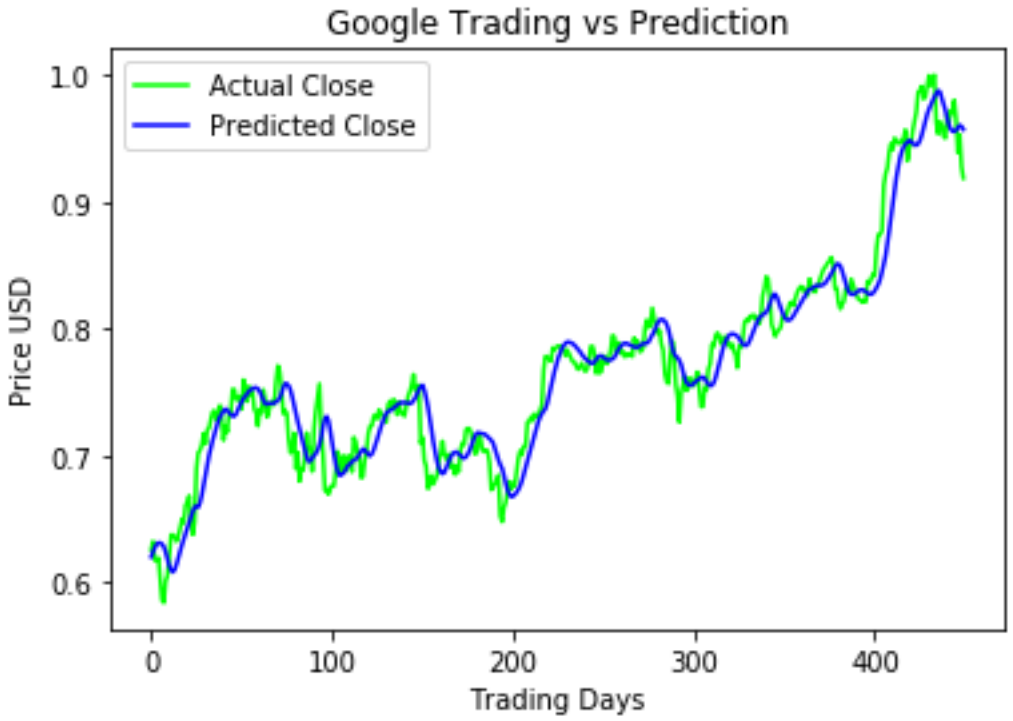


Fig: Plot of Long-Short Term Memory Model (epochs=10)

Technologies Used

The technologies undertaken in this project:

- Set Up Infrastructure
    - iPython Notebook
    - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib, Sklearn, Numpy)
    - Git project organization
- Prepare Dataset
    - Incorporate data of Alphabet Inc company
    - Process the requested data into Pandas Dataframe
    - Develop function for normalizing data
- Develop Basic LSTM Model
    - Set up basic LSTM model with Keras utilizing parameters from Benchmark Model
- Plot LSTM Predicted Values per time series
- Analyze and describe results for report.

I started this project with the hope to learn a completely new algorithm, i.e, Long-Short Term Memory and also to explore a real time series data sets. The final model really exceeded my expectation and have worked remarkably well. I am greatly satisfied with these results.