12/14/2020

STOCK PRICE PREDICTOR

Using Long-Short Term Memory Networks

Kartik Sojitra
100723768

Contents

Executive Summery

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're 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. 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 1 S&P 500 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

Metrics

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).

Data Exploration

I will be using the daily prices of the S&P 500 from 1ST December,2010 to 1st Decemcer,2020; this is a series of data points indexed in time order or a time series. My goal will be to predict the closing price for any given date after training. All the necessary data for the project will come from Yahoo Finance.

Data Exploration The data used in this project is of the Alphabet Inc from January 1, 2005 to June 20, 3 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 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 5 prediction for "CLOSE" price.

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.

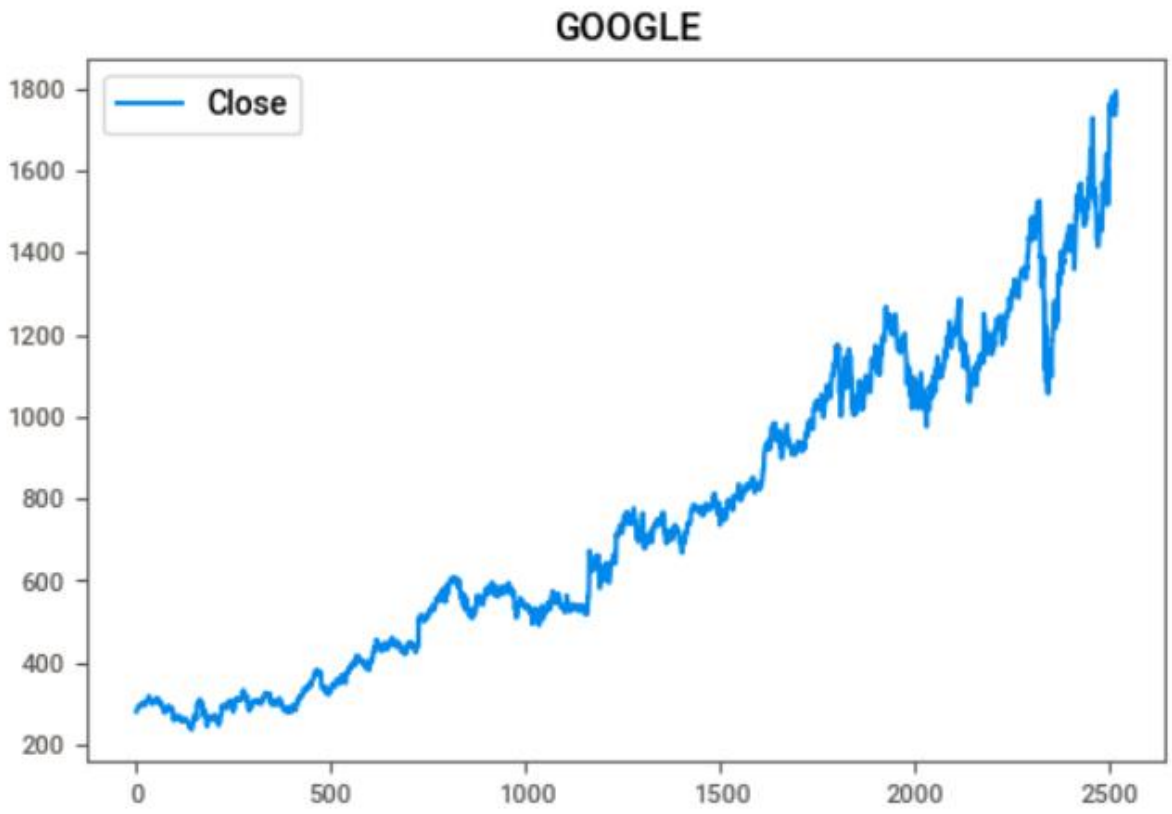
The mean, standard deviation, maximum and minimum of the data was found to be following:

Feature	Open	High	Low	Close	Volume
Mean	752.69	759.65	745.75	752.94	2871046.64
Std	378.10	382.60	374.28	378.60	2352120.73
Max	1790.90	1818.06	1772.43	1793.18	28192500
Min	236.11	239.48	235.63	236.55	7900

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 prices. 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.

Exploratory Visualization

To visualize the data, I have used matplotlib library. I have plotted Closing stock price of 7 the data with the no of items (no of days) available.



Through this data we can see a continuous growth in Alphabet Inc. The major rise in the prices between 1200-1800 might be because of the more users worldwide.

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 8 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 Long-Short Term Memory Networks, usually referred as LSTMs. These are a special kind of RNN, capable of 9 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; kept constant at 82.95% and 17.05% for benchmarks and LSTM model)

Validation Sets (kept constant at 0.05% of training sets)

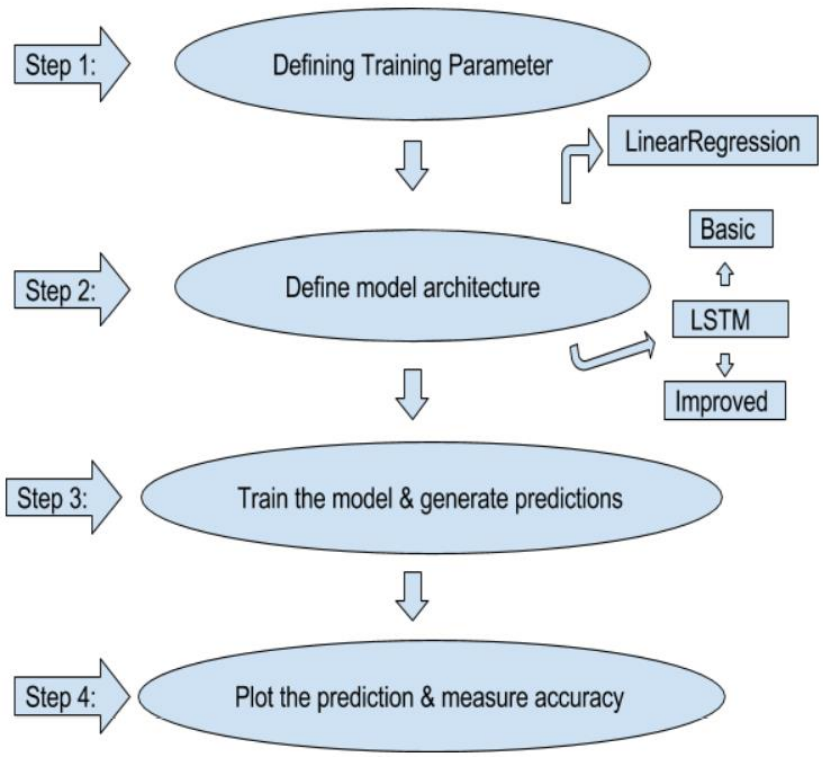
Batch Size (how many time steps to include during a single training step; kept at 1 for basic LSTM model and at 512 for improved LSTM model)

Optimizer Function (which function to optimize by minimizing error; used “Adam” throughout)

Epochs (how many times to run through the training process; kept at 1 for base model and at 20 for improved LSTM)

Implementation

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



Solutions Statement

For this project, according to my research, the best possible solution is to utilize a LSTM Neural Net model capable of learning from time-series data. This project will be programmed in a Jupyter Notebook (iPython) for ease of reproducibility. Using a Keras implementation of the Tensor Flow library, the solution will utilize a LSTM Neural Net model and will be supported by a Pandas DataFrame library for convenient time series data schema. The measures of performance will be based on the predicted stock ticker price in comparison to both the actual price and the benchmark model's predicted price.

Benchmark Model

For this project, I will use a Linear Regression model as its primary benchmark. As one of my goals is to understand the relative performance and implementation differences of machine learning versus deep learning models. This Linear Regressor will be based on the examples presented in Udacity's Machine Learning for Trading course and will be used for error rate comparison MSE and RMSE utilizing the same dataset as the deep learning models.

Project Design

This project will be implemented through the Keras/Tensor Flow library using LSTM Neural Networks. Development workflow will follow the below sequence:

1. Set Up Infrastructure
    - iPython Notebook
    - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib, Sklearn, Numpy)
    - Git project organization
2. Prepare Dataset
    - Incorporate data of S&P 500 companies
    - Process the requested data into Pandas
    - Dataframe Develop function for normalizing data
    - Dataset will be used with 80/20 split on training and test data across all models
3. Develop Benchmark Model
    - Set up basic Linear Regression model with Scikit-Learn
    - Calibrate parameters
4. Develop Basic LSTM Model
    - Set up basic LSTM model with Keras utilizing parameters from Benchmark Mode
5. Improve LSTM Model
    - Develop, document, and compare results using additional labels for the LSMT model
6. Document and Visualize Results
    - Plot Actual, Benchmark Predicted Values, and LSTM Predicted Values per time series
    - Analyze and describe results for report

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 8 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 Long-Short Term Memory Networks, usually referred as LSTMs. These are a special kind of RNN, capable of 9 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:

1) Input Parameters

- Preprocessing and Normalization (see Data Preprocessing Section)

2) 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)

3) Training Parameters

- Training / Test Split (how much of dataset to train versus test model on; kept constant at 82.95% and 17.05% for benchmarks and 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 1 for basic LSTM model and at 512 for improved LSTM model)
- Optimizer Function (which function to optimize by minimizing error; used “Adam” throughout)
- Epochs (how many times to run through the training process; kept at 1 for base model and at 20 for improved LSTM)

Data Preprocessing

Acquiring and preprocessing the data for this project occurs in following sequence, much of which has been modularized into the preprocess.py file for importing and use across all notebooks:

- Request the data from the Yahoo Finance Python API
- Normalized the data using MinMaxScaler helper function from Scikit-Learn.
- Stored the normalized data in google_preprocessed.csv file for future reusability.

- Splitting the dataset into the training (68.53%) and test (31.47%) datasets for linear regression model. The split was of following shape:
  x_train (2155, 1)
  y_train (2155, 1)
  x_test (990, 1)
  y_test (990, 1)
- Splitted the dataset into the training (82.95%) and test (17.05%) 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 (446,3)

Implementation

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

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

Step 1: Split into train and test model.

Note: The same set of training and testing data is used for improved LSTM as is used with basic LSTM.

Step 2: Build an improved LSTM model.

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

NOTE: The function uses Keras Long short-term memory 11 library to implement LSTM model.

I have increased the batch_size to 512 from 1 Epochs from 1 to 20 for my improved LSTM model. Also, the function I have add increased the no of nodes in hidden layer to 128 from 100 and have added a drop out of 0.2 to all the layers.

Step 3: We now need to train our model.

I have used here a built-in library function to train the model. Step 4: Now it is time to predict the prices for given test datasets.

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

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.

Refinement

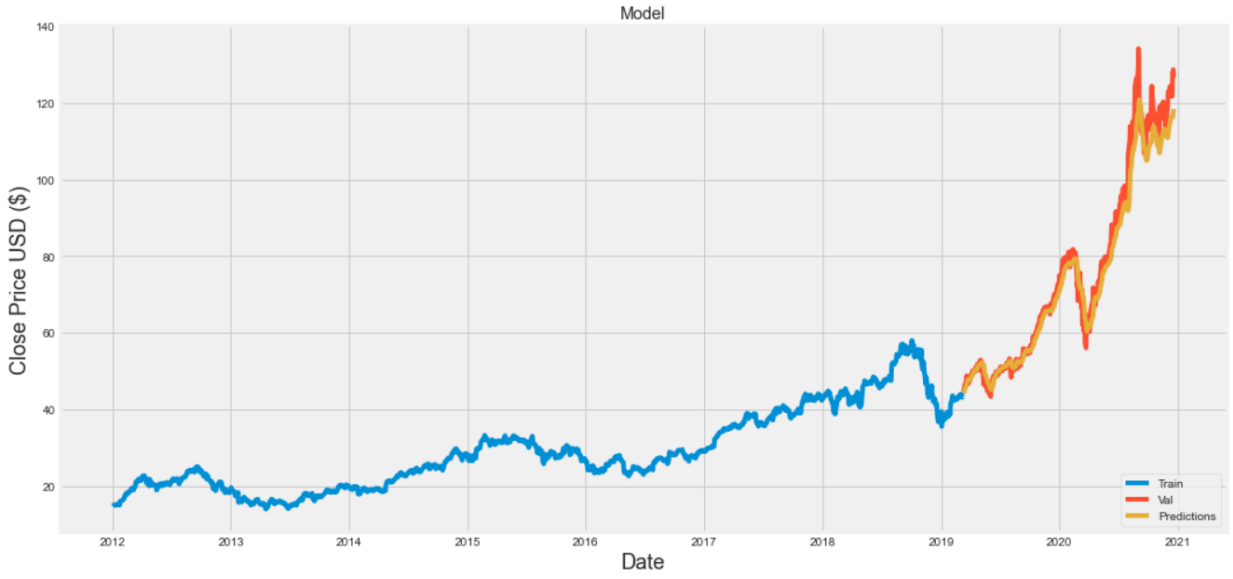
For this project I have worked on fine tuning parameters of LSTM to get better predictions. I did the improvement by testing and analysing each parameter and then selecting the final value for each of them. To improve LSTM, I have done following:

- Increased the number of hidden nodes from 100 to 128.
- Added Dropout of 0.2 at each layer of LSTM
- Increased batch size from 1 to 512
- Increased epochs from 1 to 20
- Added verbose = 2
- Made prediction with the batch size Thus improved my mean squared error, for testing sets, from 0.01153170 MSE to 0.00093063 MSE.

Conclusion

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 improved version of 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 'Aha!' moment for me as I had to poke around a lot. But it was fun working on this project.

RMSE Score: 4.3905



Reflection

To recap, the process 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 Data frame
  - Develop function for normalizing data
  - Dataset used with an 80/20 split on training and test data across all models
- Develop Benchmark Model
  - Set up basic Linear Regression model with Scikit-Learn
  - Calibrate parameters
- Develop Basic LSTM Model
  - Set up basic LSTM model with Keras utilizing parameters from Benchmark Model
- Improve LSTM Model
  - Develop, document, and compare results using additional labels for the LSMT model
  - Document and Visualize Results
- Plot Actual, Benchmark Predicted Values, and 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 to explore a real time series data set. The final model really exceeded my expectation and have worked remarkably well. I am greatly satisfied with these results. The major problem I faced during the implementation of project was exploring the data. It was toughest task. To convert data from raw format to preprocess data and then to split them into training and test data. All these steps require a great deal of patience and very precise approach. Also, I had to work around a lot to successfully use the data for 2 models, i.e., Linear Regression and Long-Short Term Memory, as both have different inputs sizes. I read many research papers to get this final model right and I think it was all worth it.

Improvement

Before starting my journey as Machine Learning Nanodegree Graduate i had no prior experience in python. In the beginning of this course to do everything with python, i had to google it. But now i have not only made 7 projects in python, i have explored many libraries along the ways and can use them very comfortably. This is all because of highly interactive videos and forum provided by Udacity. I am happy and satisfied taking up this course. And as there is scope of improvement

in everyone so is the case with this project. This project though predicts closing prices with very minimum Mean Squared Error, still there are many things that are lagging in this project. Two of most important things are:

- There is no user interaction or interface provided in this project. A UI can be provided where user can check the value for future dates.
- The stocks used for this project are only of Apple Inc, we can surely add more S&P 500 in the list to make this project more comprehensive.

I would like to add these improvements to this project in future and would like to deploy it.