Machine Learning Engineer Nanodegree

Stock Price Prediction

Simran Kaur

April 22nd, 2020

I. DEFINITION

Project Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behaviour 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.

The uncertainty in the stock market refrain people from investing in stocks. Thus, there is a need to accurately predict the stock market which can be used in a real-life scenario. [1] DeepAR is an LSTM neural network that can be used to forecast time-series data, accounting for trends and seasonality of the time series in order for the network to learn and give accurate forecasts.[2]

Problem Statement

The aim is to predict the future closing value of a certain stock using the data of the last 60 days.

The solution should include the steps in the order:

1. Obtaining the data from yahoo finance of a particular stock.

2. Data Preprocessing.

3. Using Keras to build RNN with LSTM.

4. Training the Neural Network.

5. Applying an optimizer

6. Making Predictions.

Metric

To evaluate the effectiveness of the model, root mean square error (RMSE) between the actual price and the predicted price will be used. RMSE can provide the average deviation of the prediction from the true value and can be compared with the mean of the true value.

II. ANALYSIS

Data Exploration

I have used Yahoo Finance to procure the dataset from. I have used the Apple stock.

It initially had 6 columns namely:-

1. High
2. Low
3. Open

4. Close
5. Volume
6. Adj Close

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	58.928570	58.428570	58.485714	58.747143	75555200.0	50.994907
2012-01-04	59.240002	58.468571	58.571430	59.062859	65005500.0	51.268970
2012-01-05	59.792858	58.952858	59.278572	59.718571	67817400.0	51.838169
2012-01-06	60.392857	59.888573	59.967144	60.342857	79573200.0	52.380054
2012-01-09	61.107143	60.192856	60.785713	60.247143	98506100.0	52.296970

Fig 1. Initial Dataset Snapshot

It can be seen from the above snapshot that the dataset was indexed by date.
After preprocessing and sorting the data, the final dataset looks like :

	date	high	low	open	close	volume	adj_close	month
0	2012-01-03	58.928570	58.428570	58.485714	58.747143	75555200.0	50.994907	1
1	2012-01-04	59.240002	58.468571	58.571430	59.062859	65005500.0	51.268970	1
2	2012-01-05	59.792858	58.952858	59.278572	59.718571	67817400.0	51.838169	1
3	2012-01-06	60.392857	59.888573	59.967144	60.342857	79573200.0	52.380054	1
4	2012-01-09	61.107143	60.192856	60.785713	60.247143	98506100.0	52.296970	1

Fig 2. Final Dataset Snapshot

Exploratory Visualization

In fig 3., the close price is shown over the years. We can also see that there is an upward trend in general in the stock.

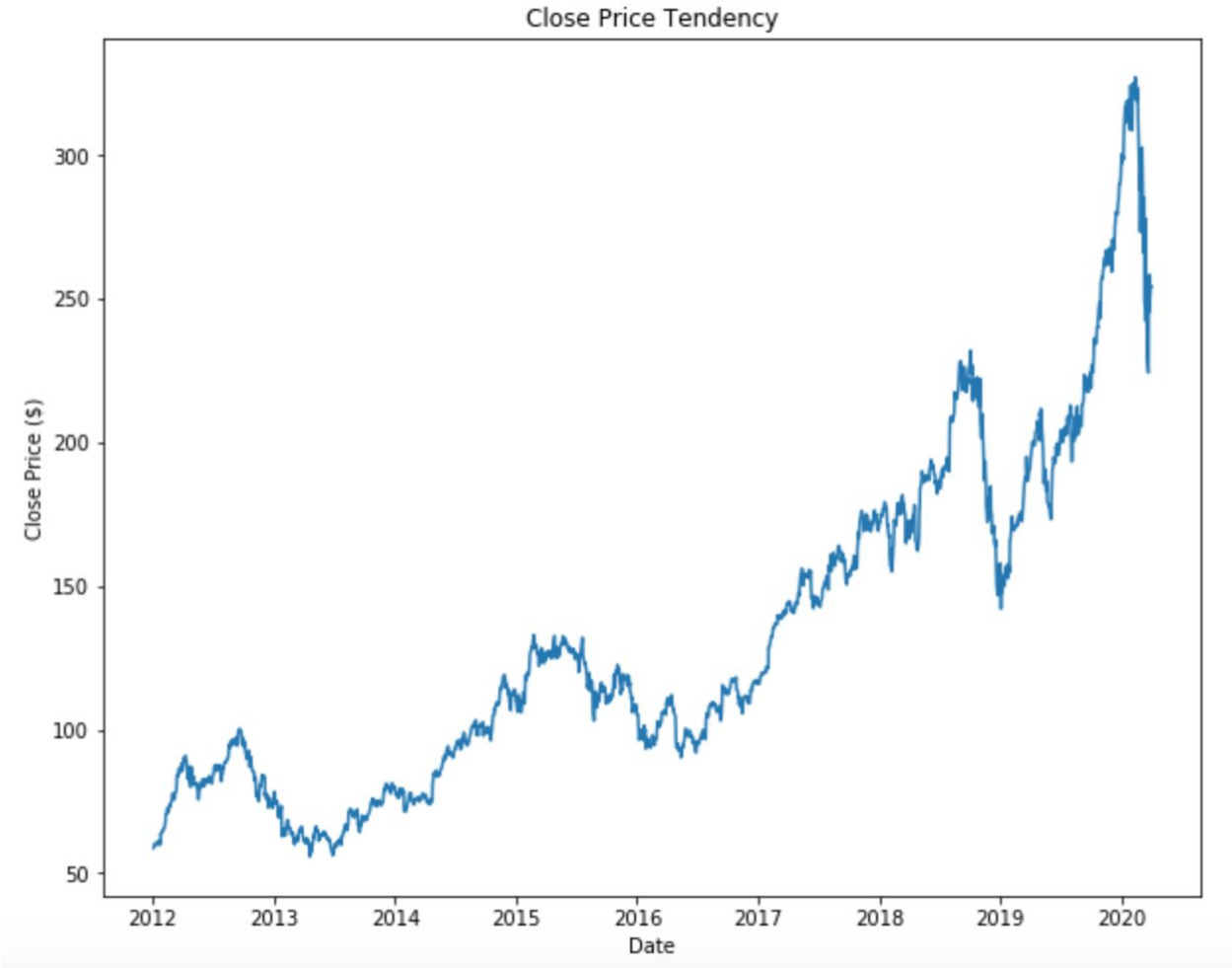

Fig 3. Closing Price History

Algorithms and Techniques

In this project, I used a recurrent neural network to achieve the aim of my project. I used a Long-short term memory (LSTM) to predict the stock prices. I fed the data of the past 60 days to the model, and the predicted outcome is the price of the 61st day.

The network architecture is as follows:

- Two LSTM layers
- One Dense layer
- Two Dropout layers

Benchmark Model

The benchmark model of this project is Linear Regression. The RMSE value of this model is that of 70.672.

Fig 4. Benchmark Model Prediction

III. METHODOLOGY

Data Preprocessing

The stock data was selected from 1st Jan 2012 to 31st March 2020.

The first done to the data as obtained from yahoo finance was to reindex it. It was previously index by date as shown in fig 1. After reindexing and sorting by month, it looked like fig 2.

The prediction was done using the close prices of the stock. A list of the close prices was filtered. Thereafter, the data was split among training and testing sets.

Implementation

The neural network model will be implemented using Keras module. It has two LSTM layers, a drop layer after every LSTM layer. The probability of that is set 1. The loss function is chosen to be mean squared error and the optimizer is chosen to be Adam.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 60, 50)            10400
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
_________________________________________________________________
```

Fig 5. Model summary

Refinement

One of the hyperparameters is the timestep, I tried changing the timestep to smaller or larger than 60. 60 obtains the optimum results.
The second hyperparameter is the number of layers. Having a network with 3 LSTM layer yields the same result like the one with the 2.
The third hyperparameter I tuned was the probability of dropout. Initially, I had it at 0.5 but 1 yielded optimum results.
The fourth hyperparameter is an optimizer. I used ADAGrade as well as Adam but theoretically, Adam yields better convergence hence chose Adam.

IV. RESULTS

Model Evaluation and Validation

The model was trained on a training set and validated.

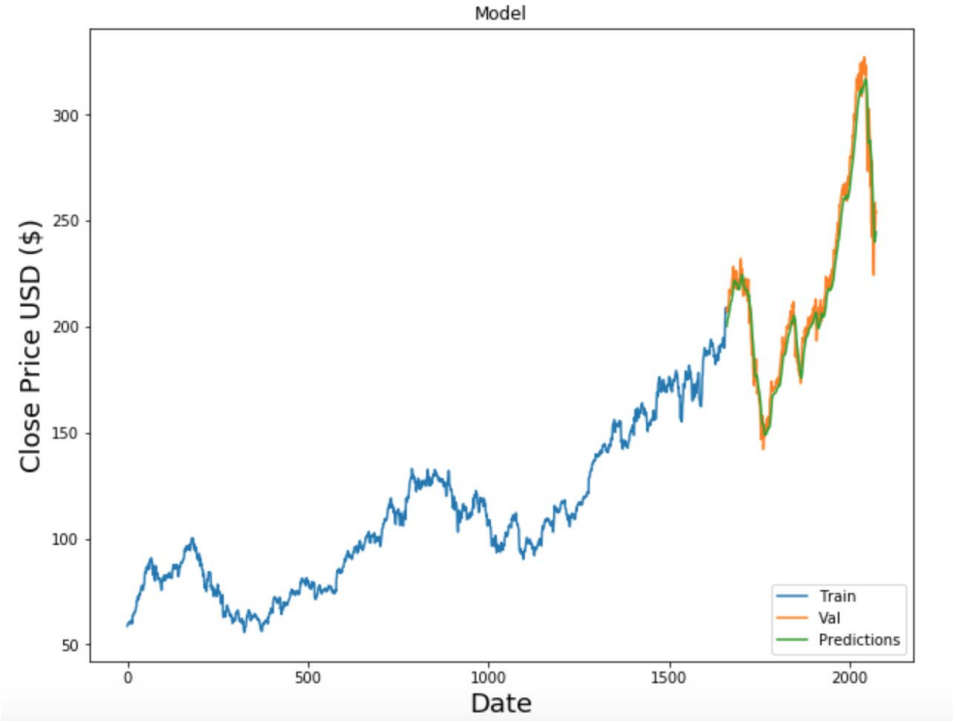

Fig 6. LSTM predictions

The RMSE value is 8.18 obtained by the model which is less than the benchmark model. The lower the RMSE value, the better is the result, having said that stock price prediction isn't a straightforward task. It has many confounding factors determining the price.
The predicted and actual values on the test set can be seen below :

	close	Predictions
1660	207.250000	200.229111
1661	208.880005	201.737610
1662	207.529999	203.055573
1663	208.869995	203.930054
1664	209.750000	204.681213
1665	210.240005	205.380951
1666	213.320007	206.024857
1667	217.580002	206.960388
1668	215.460007	208.438904
1669	215.039993	209.665695
1670	215.050003	210.600845
1671	215.490005	211.294067
1672	216.160004	211.848526
1673	217.940002	212.353561
1674	219.699997	212.999359
1675	222.979996	213.841492
1676	225.029999	215.082993
1677	227.630005	216.590042

Fig 7. Test set predictions

Justification

We can compare the benchmark model and the LSTM model. The LSTM model performed much better than the benchmark model that can be shown by the RMSE value. LSTM model generates its own feature representation in the hidden units and its long-short term memory cell is useful which is why it good at dealing with time-series data, unlike linear regression.

V. CONCLUSION

Reflection

The process of this project can be summed up in the below points.

1. Load data
2. Reindexing and sorting data
3. Splitting the data between training and testing
4. Building the model
5. Building the benchmark model
6. Training the model
7. Predictions

I was new to LSTM and basically Keras framework. I have since found it easy to use.

Improvement

The stock prices cannot be accurately predicted since it may or may not be dependent on a lot of factors. However, an improvement for this

project could be taking into account other real-world features, maybe news, into account to get a more reasonable prediction.