विश्वजीवनामृतं ज्ञानम्

Atal Bihari Vajpayee Indian Institute of Information Technology and Management Gwalior

Integrated M.Tech IT

Mini Project Report

Stock Price Prediction using Machine Learning(LSTM)

Roll Number :
2019IMT-108 - Uditansh Patel

Under the supervision of
Dr. Kiran Kumar Pattanaik

Contents

NOTE: This project is still work in progress and this report is intended for the intermediate evaluation. Hence, this project report may contain data and insights from past iterations as well. For the current version and detailed work timeline, please do visit the repository on GitHub: **My Project On GitHub**

1 ABSTRACT

In this project we attempt to implement machine learning approach to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to predict the stock prices in order to make more informed and accurate investment decisions. We propose a stock price prediction system that integrates mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades. There are two types of stocks. You may know of intraday trading by the commonly used term "day trading." Interday traders hold securities positions from at least one day to the next and often for several days to weeks or months. LSTMs are very powerful in sequence prediction problems because they're able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

Keywords: LSTM, CNN, ML, DL, Trade Open, Trade Close, Trade Low, Trade High

2 INTRODUCTION

Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behavior, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

3 MOTIVATION

Businesses primarily run over customer's satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an unbiased automated system to classify customer reviews regarding any problem. Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition.

4 FEATURES

There are a lot of methods and tools used for the purpose of stock market prediction. This program could be one such tool. It could be used to predict the stock market trends in order to provide better insight into the volatility of any particular stock market listings. Predicting stock prices is an uncertain task which is modelled using machine learning to predict the return on stocks. The stock market is considered to be very dynamic and complex in nature. An accurate prediction of future prices may lead to a higher yield of profit for investors through stock investments. As per the predictions, investors will be able to pick the stocks that may give a higher return.

5 ABOUT LSTM

Long Short-Term Memory(LSTM - RNN)

Long-Short-Term Memory Recurrent Neural Network belongs to the family of deep learning algorithms. It is a recurrent network because of the feedback connections in its architecture. It has an advantage over traditional neural networks due to its capability to process the entire sequence of data. Its architecture comprises the cell, input gate, output gate and forget gate.

The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. The cell of the model is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell, and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

However, there are some variants of the LSTM model such as Gated Recurrent Units (GRUs) that do not have the output gate. LSTM Networks are popularly used on time-series data for classification, processing, and making predictions. The reason for its popularity in time-series application is that there can be several lags of unknown duration between important events in a time series.

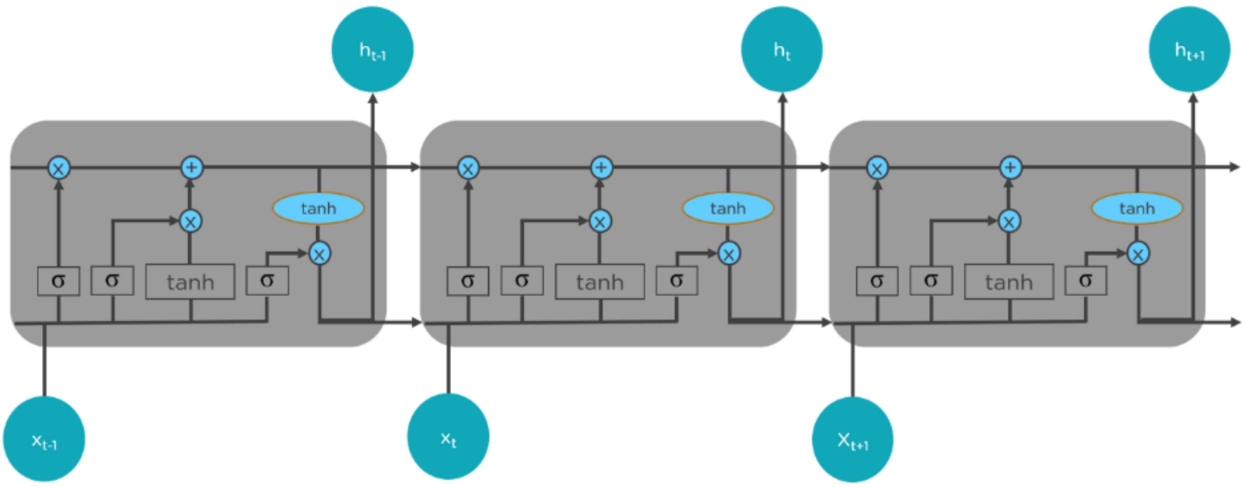
Understanding LSTM - RNN

Understanding Long Short Term Memory Network Here, you will use a Long Short Term Memory Network (LSTM) for building your model to predict the stock prices.

LTSMs are a type of Recurrent Neural Network for learning long-term dependencies. It is commonly used for processing and predicting time-series data.

LSTM

From the image below, you can see LSTMs have a chain-like structure. General RNNs have a single neural network layer. LSTMs, on the other hand, have four interacting layers communicating extraordinarily.



LSTMs work in a three-step process.

The first step in LSTM is to decide which information to be omitted from the cell in that particular time step. It is decided with the help of a sigmoid function. It looks at the previous state (ht-1) and the current input xt and computes the function. There are two functions in the second layer. The first is the sigmoid function, and the second is the tanh function. The sigmoid function decides which values to let through (0 or 1). The tanh function gives the weightage to the values passed, deciding their level of importance from -1 to 1. The third step is to decide what will be the final output. First, you need to run a sigmoid layer which determines what parts of the cell state make it to the output. Then, you must put the cell state through the tanh function to push the values between -1 and 1 and multiply it by the output of the sigmoid gate. With this basic understanding of LSTM, you can dive into the hands-on demonstration part of this tutorial regarding stock price prediction using machine learning.

6 WORKING PROCESS

Google Stock Price Prediction Using LSTM

1. Import the Libraries.

```
#Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

LoadLibraries

2. Load the Training Dataset.

The Google training data has information from 3 Jan 2012 to 30 Dec 2016. There are five columns. The Open column tells the price at which a stock started trading when the market opened on a particular day. The Close column refers to the price of an individual stock when the stock exchange closed the market for the day. The High column depicts the highest price at which a stock traded during a period. The Low column tells the lowest price of the period. Volume is the total amount of trading activity during a period of time.

```
dataset_train = pd.read_csv("Google_Stock_Price_Train.csv")
dataset_train.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

LoadDataset

3. Use the Open Stock Price Column to Train Your Model.

```
training_set = dataset_train.iloc[:,1:2].values

print(training_set)
print(training_set.shape)
```

```
[[325.25]
 [331.27]
 [329.83]
 ...
 [793.7 ]
 [783.33]
 [782.75]]
(1258, 1)
```

OpenPrice

4. Normalizing the Dataset.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range = (0,1))
scaled_training_set = scaler.fit_transform(training_set)

scaled_training_set
```

```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       ...,
       [0.95725128],
       [0.93796041],
       [0.93688146]])
```

NormalizingData

5. Creating X$_{t}rainandy_{t}rainDataStructures$.

```
X_train = []
y_train = []
for i in range(60,1258):
    X_train.append(scaled_training_set[i-60:i, 0])
    y_train.append(scaled_training_set[i, 0])
X_train = np.array(X_train)
y_train = np.array(y_train)
```

TrainingData

```
print(X_train.shape)
print(y_train.shape)
```

```
(1198, 60)
(1198,)
```

ShapeOfData

6. Reshape the Data.

```
X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 1))

X_train.shape
```

```
(1198, 60, 1)
```

ReshapeData.

7. Building the Model by Importing the Crucial Libraries and Adding Different Layers to LSTM.

```
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
```

DeepLearningLibraries

```
regressor = Sequential()

regressor.add(LSTM(units = 50, return_sequences= True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences= True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences= True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units=1))
```

BuildingModel

8. Fitting the Model.

```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
38/38 [==============================] - 11s 114ms/step - loss: 0.1011
Epoch 2/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0061
Epoch 3/100
38/38 [==============================] - 4s 118ms/step - loss: 0.0063
Epoch 4/100
```

FitModel

9. Extracting the Actual Stock Prices of Jan-2017.

```
dataset_test = pd.read_csv("Google_Stock_Price_Test.csv")
actual_stock_price = dataset_test.iloc[:,1:2].values
```

TestData

10. Preparing the Input for the Model.

```
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total)- len(dataset_test)-60:].values

inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)

X_test = []
for i in range(60,80):
    X_test.append(inputs[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))
```

ModelInput

11. Predicting the Values for Jan 2017 Stock Prices.

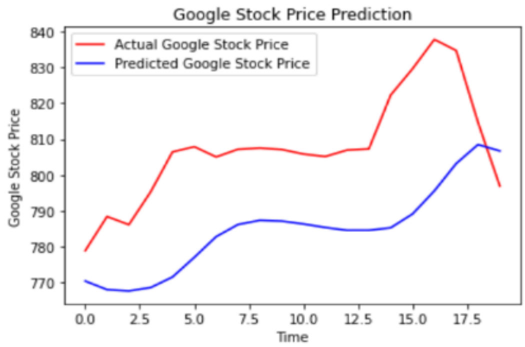
```
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

PredictStocks

12. Plotting the Actual and Predicted Prices for Google Stocks.

```
plt.plot(actual_stock_price, color = 'red', label = 'Actual Google Stock Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Google Stock Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fbf71eb1b90>
```



PlottingData

7 RESULT

As you can see above, the model can predict the trend of the actual stock prices very closely. The accuracy of the model can be enhanced by training with more data and increasing the LSTM layers.

8 FUTURE SCOPE

Traditionally most machine learning (ML) models use as input features some observations (samples / examples) but there is no time dimension in the data. Time-series forecasting models are the models that are capable to predict future values based on previously observed values. Time-series forecasting is widely used for non-stationary data. Non-stationary data are called the data whose statistical properties e.g. the mean and standard deviation are not constant over time but instead, these metrics vary over time. These non-stationary input data (used as input to these models) are usually called time-series. Some examples of time-series include the temperature values over time, stock price over time, price of a house over time etc. So, the input is a signal (time-series) that is defined by observations taken sequentially in time.

9 CONCLUSION

The stock market plays a remarkable role in our daily lives. It is a significant factor in a country's GDP growth. In this project, we aim to solve some of the investor problems by providing a program to gain critical insights into the world of stock market, through stock price prediction using machine learning.

10 REFERENCES

1.https://the-learning-machine.com/article/dl/long-short-term-memory
2.https://www.kaggle.com/amarsharma768/stock-price-prediction-using-lstm/notebook