Stock price prediction

Abstract

The prediction of stock value is a complex task which needs a robust algorithm background in order to compute the longer term share prices. Stock prices are correlated within the nature of market; hence it will be difficult to predict the costs. The proposed algorithm using the market data to predict the share price using machine learning techniques like recurrent neural network named as Long Short Term Memory, in that process weights are corrected for each data points using stochastic gradient descent. This system will provide accurate outcomes in comparison to currently available stock price predictor algorithms. The network is trained and evaluated with various sizes of input data to urge the graphica outcome

Keywords : Machine Learning, Stock Price Prediction, Long Short-Term Memory, Stock Market, Artificial neural Networks, National Stock Exchange

Introduction

Stock market prediction is the act of trying to determine the future value of company stock or other financial instruments traded on an exchange. The successful prediction of a stock's future price could yield a significant profit. In this application, we used the LSTM network to predict the closing stock price using the past 60-day stock price.For the application, we used the machine learning technique called Long Short Term Memory (LSTM). LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learningUnlike standard feed-forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).LSTM is widely used for the problems of sequence prediction and been very effective

Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) refers to the method of studying and exploring record sets to apprehend their predominant traits, discover patterns, locate outliers, and identify relationships between variables. EDA is normally carried out as a preliminary step before undertaking extra formal statistical analyses or modelling.

Proposed System:

As represented in the previous section getting the historical data from market is mandatory step. Then there is a need to extract the feature which is required for data analysis, then divide it as testing and training data, training the algorithm to predict the price and the final step it to visualize the data. Fig. 1 represents the Architecture of the proposed system.

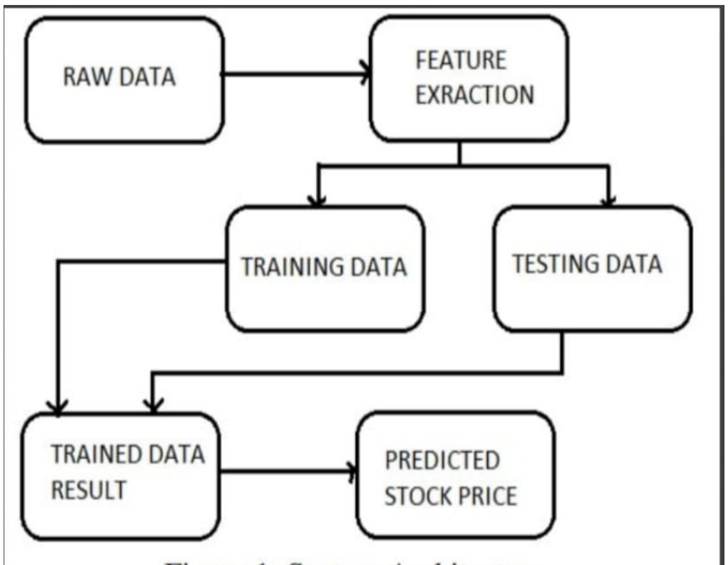


Figure 1: System Architecture

The typical LSTM unit consists of a cell, an info door, an entrance door and a door with a view. The cell collects values over discretionary time intervals, and the three inputs manage the progress of data into and out of the cell. The main advantage of the LSTM is its ability to learn context-specific temporal dependence. Each LSTM unit collects information for either a long or short period of time (hence the name) without explicitly using the activation function within the recurrent components.

A significant certainty to note is that any cell state is uniquely increased by the output of the overlooked entryway, which changes somewhere in the range of 0 and 1. In other words, the overhead door in the LSTM cell is responsible for both the loads and the capacity to initiate the cell state. Subsequently, data from a past cell state can pass through a cell unaltered rather than expanding or decreasing exponentially at each time-step or layer, and loads can meet their ideal quality in a reasonable measure of time .

For example, LSTM is material for undertakings, such as un partitioned, associated penmanship recognition, speech recognition and recognition of peculiarities in arranged traffic or IDS (interruption location frameworks). Algorithm 1: Stock prediction using LSTM Input: Historic stock data Output: prediction of stock price using price variation Step 1: Start. Step 2: Data Preprocessing after getting the historic data from the market for a particular share. Step 3: import the dataset to the data structure and read the open price. Step 4: do a feature scaling on the data so that the data values will vary from 0 and 1. Step 5: Creating a data structure with 60 timestamps and 1 output. Step 6: Building the RNN (Recurrent neural network) for

Implementation of Stock Price Prediction

1. Importing Modules:

First step is to import all the necessary modules in the project.

```
#import the libraries
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import keras.models as sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

For the project, we will be using basic modules like **numpy**, **pandas**, and **matplotlib**. In addition to this, we will be using some submodules of keras to create and build our model properly.
We would also require the math module for basic calculation and preprocessing module of sklearn to handle the data in a better and simpler way.

Data Attributes:

For the project we will be using the all_stocks_5yrs csv file which includes stock data for 5 years and has seven columns which are listed below.

1. **Date** – Format of date is: “yy-mm-dd”
2. **Open** – Price of the stock at open market
3. **High** – Highest price reached in the day
4. **Low** – Lowest price reached in the day
5. **Close** – Price of the stock at the close market
6. **Volume** – Number of shares traded
7. **Name** – The name of the stock ticker

```
#Get the stock quote
import yfinance as yf
df=yf.download('AAPL',start='2012-01-01',end='2019-12-17')
#show the data
df
```

The head function displays first five rows of the dataset.

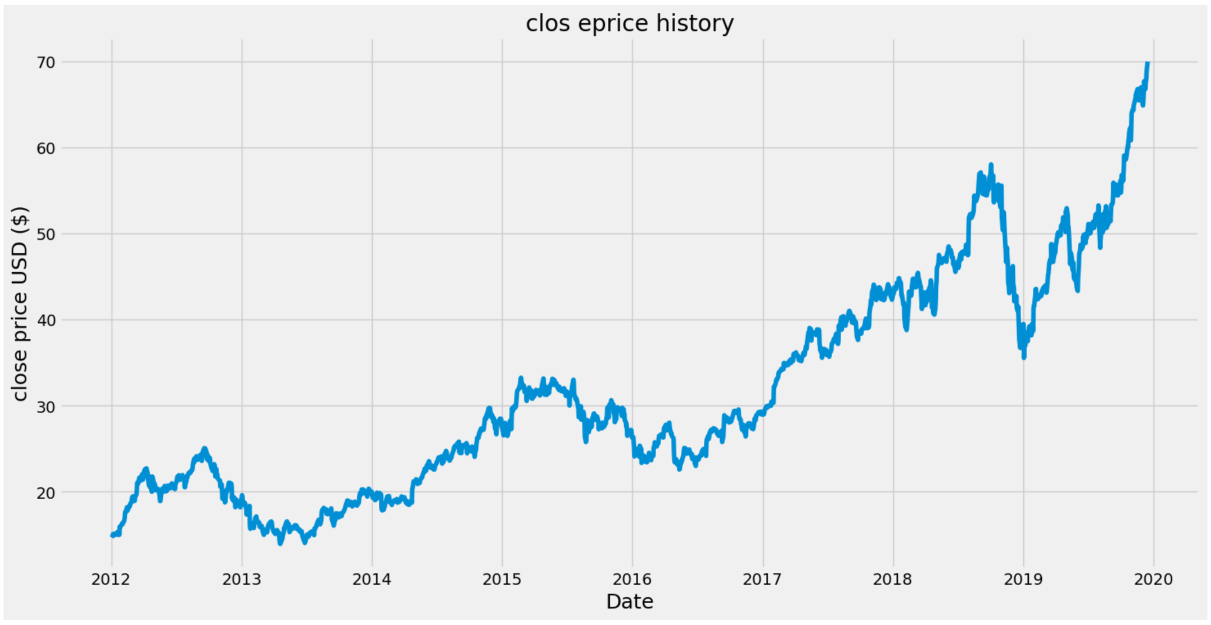
	Open	**High**	**Low**	**Close**	**Adj Close**	**Volume**
Date						
2012-01-03	14.621429	14.732143	14.607143	14.686786	12.449689	302220800
2012-01-04	14.642857	14.810000	14.617143	14.765714	12.516597	260022000
2012-01-05	14.819643	14.948214	14.738214	14.929643	12.655556	271269600
2012-01-06	14.991786	15.098214	14.972143	15.085714	12.787856	318292800
2012-01-09	15.196429	15.276786	15.048214	15.061786	12.767570	394024400
...	...	...	...	...	...	...
2019-12-10	67.150002	67.517502	66.464996	67.120003	65.390457	90420400
2019-12-11	67.202499	67.775002	67.125000	67.692497	65.948196	78756800
2019-12-12	66.945000	68.139999	66.830002	67.864998	66.116241	137310400
2019-12-13	67.864998	68.824997	67.732498	68.787498	67.014992	133587600
2019-12-16	69.250000	70.197502	69.245003	69.964996	68.162140	128186000

2002 rows × 6 columns

Visualizing the data:

To visualize the data we will be first plotting the date vs close market prices for the **AAPLE** stock for all the data points. To make the visualization simpler, we would be plotting the same plot but for only the first 60 data points.

```
#visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('clos eprice history')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('close price USD ($)',fontsize=18)
plt.show()
```



Creating a new Dataframe and Training data:

To make our study easier we will only consider the closing market price and predict the closing market price using Python. The whole train data preparation is shown in the steps below. Comments are added for your reference

```
#Create new dataframe with only the close column
data=df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset=data.values
#Get the number of rows to train the model on
training_data_len=math.ceil(len(dataset)*.8)
training_data_len
#Scale the data
scaler=MinMaxScaler(feature_range=(0,11))
```

```
scaled_data=scaler.fit_transform(dataset)
scaled_data
#Create training data set
#Create the scaled training data set
train_data=scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train=[]
y_train=[]
for i in range(60,len(train_data)):
  x_train.append(train_data[i-60:i, 0])
  y_train.append(train_data[i, 0])
  if i<=61:
    print(x_train)
    print(y_train)
    print()
#Convert the x_train and y_train to numpy arrays
x_train,y_train =np.array(x_train),np.array(y_train)
#Reshape the data
x_train=np.reshape(x_train,(x_train.shape[0], x_train.shape[1], 1))
x_train.shape
```

Here we create a data set to train the data that contains the closing price of 60 days (60 data points) so that we could do the prediction for the 61st closing price.Now the x_train data set will contain a total of 60 values, the first column will contain from the index of 0 to 59 and the second column from the index of 1 to 60, and so on

The y_train data set will contain the 61st value at its first column located at index 60 and for the second column, it will contain the 62nd value located at index 61 and so on.Converting both the independent and dependent train data set as x_train_data and y_train_data respectively, into the NumPy arrays so that they can be used to train the LSTM model.

Also, as the LSTM model is expecting the data in 3-dimensional data set, using reshape() function we will reshape the data in the form of 3-dimension.

Building LSTM Model:

The LSTM model will have two LSTM layers with 50 neurons and two Dense layers, one with 25 neurons and the other with one neuron.

```
#Build the LSTM model
model = Sequential()
model.add(LSTM(50,return_sequences=True, input_shape=
(x_train.shape[1], 1)))
model.add(LSTM(50,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Compiling the Model:

The LSTM model is compiled using the mean squared error (MSE) loss function and the adam optimizer.

```
#Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
#Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Testing the model on testing data:

The code below will get all the rows above the training_data_len from the column of the closing price. Then convert the x_test data set into the NumPy arrays so that they can be used to train the LSTM model.As the LSTM model is expecting the data in 3-dimensional data set, using reshape() function we will reshape the data set in the form of 3-dimension. Using the predict() function, get the predicted values from the model using the test data. And scaler.inverse_transform() function is undoing the scaling.

```
#Creating the testing data set
#Create a new array containing sacled values from index 1543 to 2003
test_data = scaled_data[training_data_len - 60:, :]
#Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
  x_test.append(test_data[i-60:i, 0])
#Convert the data to a numpy array
x_test = np.array(x_test)
#Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
x_test.shape
#Get the models predicted price values
```

```
predictions=model.predict(x_test)
predictions=scaler.inverse_transform(predictions)
```

Error Calculation:

RMSE is the root mean squared error, which helps to measure the accuracy of the model.

```
#Get the root mean squared error(RMSE)
rmse=np.sqrt(np.mean(predictions - y_test)**2)
rmse
```

The lower the value, the better the model performs. The 0 value indicates the model’s predicted values match the actual values from the test data set perfectly.

rmse value we received was 1.9779846096038818which is decent enough.

Predictions:

The final step is to plot and visualize the data. To visualize the data we use these basic functions like title, label, plot as per how we want our graph to look like.

```
#plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['predictions']=predictions
#Visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close price USD ($)',fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','predictions']])
plt.legend(['Train','val','predictions'])
plt.show()
```

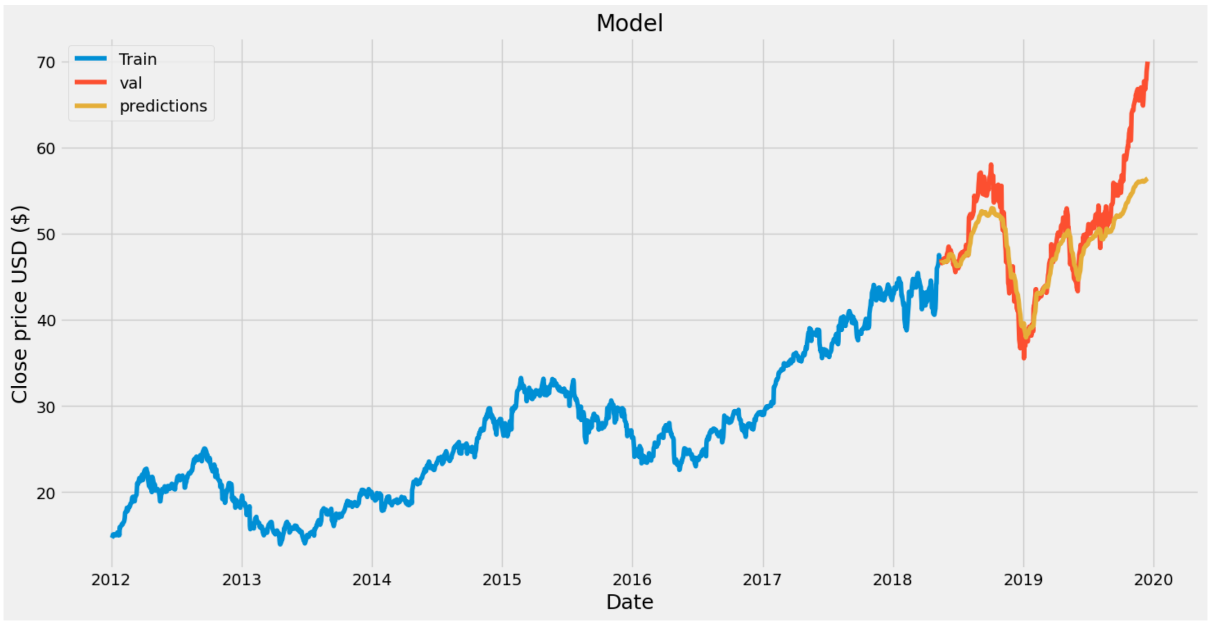
The Actual vs Predicted Values

Final Predictions Vs Actual Values LSTM Stocks

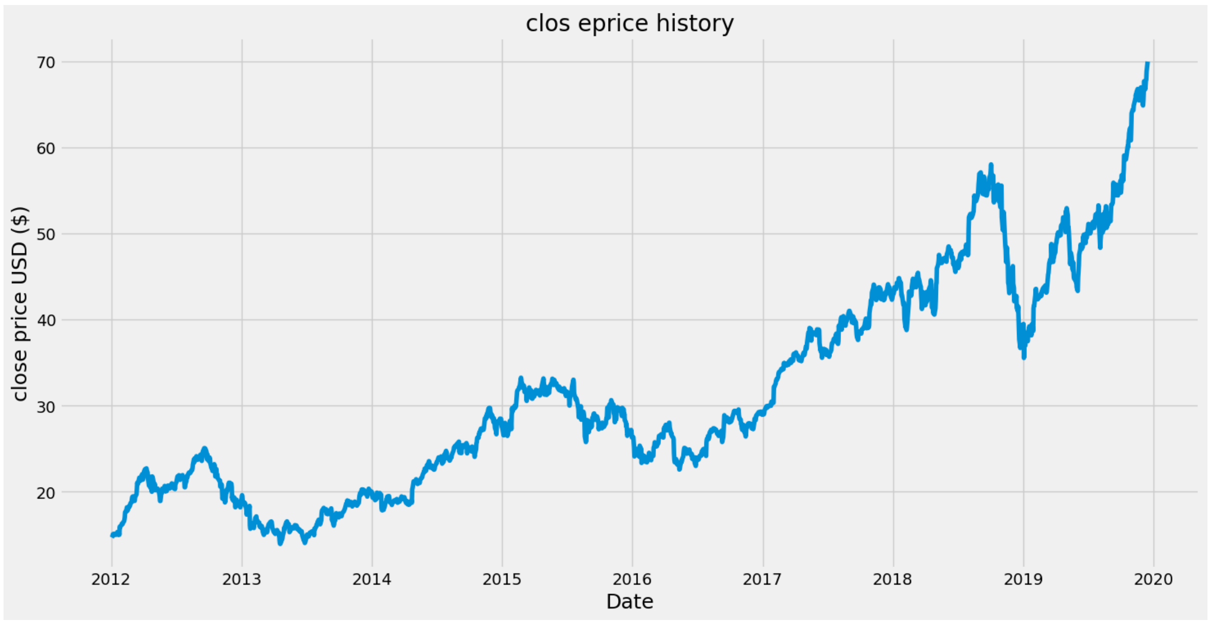
Date	Close	prediction
2018-05-16	47.044998	46.598351
2018-05-17	46.747501	46.661991
2018-05-18	46.577499	46.672195
2018-05-21	46.907501	46.639080
2018-05-22	46.790001	46.649742
...		...
2019-12-10	67.120003	56.174370
2019-12-11	67.692497	56.211159
2019-12-12	67.864998	56.265327
2019-12-13	68.787498	56.319946
2019-12-16	69.964996	56.398315

Results and Discussion:

The implementation of proposed LSTM model using python which predicts the future price of APPLE share based on its historical data. The below visualization figure shows the visualization of APPLE prediction. In our paper the implementation of an algorithm which predicts the stock price of a share for given period of time, the below graph from our algorithm will show the predicted price of APPLE share. In the result shown in the below graph is the plotted form our algorithm outcome by applying 96 LSTM units for achieving the accuracy. The Fig 2 is drawn from original dataset and also shown the result by comparing its correctness with the trained model from algorithm that is defined in the previous section. the “x” axis is share price. The “y” axis is days. The data is slot of 1500 days is shown in the Fig.



The Fig is drawn from original dataset also shown the result by comparing its correctness with the trained model from algorithm which that is defined in the previous section. the “x” axis is share price. The “y” axis is days. The data is slot of 300 days is shown in the Fig . \



Conclusion:

The study of the share is carried out in this paper and it can be carried out for several shares in the future. Prediction could be more reliable if the model trains a greater number of data sets using higher computing capacities, an increased number of layers, and LSTM modules. In future enhancement the inclusion of sentiment analysis from social media to understand what the market thinks about the price variation for a particular share and it can be implement this by adding twitter and Facebook API to our program as Facebook is a leading social media which has lots of market trend information posted by users.