Stock Price Prediction and Forecasting using Stacked LSTM.

DEEP LEARNING INTERMEDIATE TIME SERIES FORECASTING

This article was published as a part of the Data Science Blogathon

Introduction

Trying to predict how the securities exchange will work is one of the most difficult tasks. There are so many variables involved with the expectation – physical elements versus psychological factors, rational and irrational behaviour, and so on.
All of these factors combine to make share costs unpredictable and difficult to predict with any degree of certainty.
Is it possible to use AI to our advantage in this space?AI approaches will potentially reveal examples and insights we hadn't seen before, and these can be used to make unerringly exact expectations, using features like the most recent declarations of an organization, their quarterly income figures, and so on.
We will work with published information regarding a freely recorded organization's stock costs in this report.
We'll use a combination of AI calculations to forecast this company's future stock price with LSTM.
This article's main purpose is to demonstrate how these calculations are carried out. I'll provide a quick overview of the process and make key connections to revisit the concepts as needed. If you're new to the world of time management, I suggest starting with the articles below.

Table of Contents:

Loading the Data

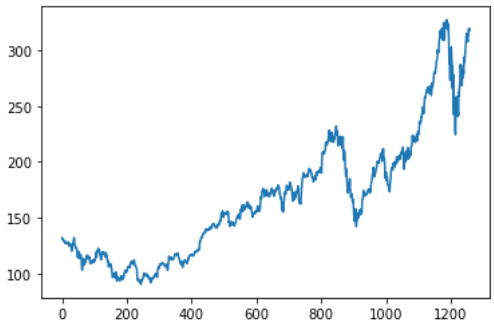
The Apple data is up to 22-05-2020. Let's take the close column for the stock prediction. We can use the same strategy.

Unnamed: 0	symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash
0	AAPL	2015-05-27 00:00:00+00:00	132.045	132.260	130.05	130.34	45833246	121.682558	121.880685	119.844118	120.111360	45833246	0.0
1	AAPL	2015-05-28 00:00:00+00:00	131.780	131.950	131.10	131.86	30733309	121.438354	121.595013	120.811718	121.512076	30733309	0.0
2	AAPL	2015-05-29 00:00:00+00:00	130.280	131.450	129.90	131.23	50884452	120.056069	121.134251	119.705890	120.931516	50884452	0.0
3	AAPL	2015-06-01 00:00:00+00:00	130.535	131.390	130.05	131.20	32112797	120.291057	121.078960	119.844118	120.903870	32112797	0.0
4	AAPL	2015-06-02 00:00:00+00:00	129.960	130.655	129.32	129.86	33667627	119.761181	120.401640	119.171406	119.669029	33667627	0.0

We should reset the index

```
df1=df.reset_index()['close']
```

so that the data will be clear
Let us plot the Close value graph using pyplot
From 2015-2020



Now get into the Solution:

LSTM is very sensitive to the scale of the data, Here the scale of the Close value is in a kind of scale, we should always try to transform the value.

Here we will use min-max scalar to transform the values from 0 to 1.We should reshape so that we can use fit transform.

Code:

```
from sklearn.preprocessing import MinMaxScaler scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

Train and Test Split

Whenever training Timeseries data we should divide the data differently we should train the data with the respective date.

Always remember that in time-series data the one data is dependent on other data. The training size should be 65% of the total length of the data frame, the test size should be the difference between the length of the dataset and the training size.

Code:

```
training_size=int(len(df1)*0.65)                                    test_size=len(df1)-training_size
train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]
```

Train data and Test data is ready.

Data Preprocessing

Now consider the time steps, if I want to predict the price of the stock in a day that how previous data should be considered.

Now the timestep value will be 100. Let's split the data X, Y. In the 0th iteration the first 100 elements goes as your first record and the 101 elements will be put up in the X. The 100 elements will be put up in the Y.

Code:

```
import numpy def create_dataset(dataset, time_step=1): dataX, dataY = [], [] for i in range(len(dataset)-time_step-1): a = dataset[i:(i+time_step), 0] dataX.append(a) dataY.append(dataset[i + time_step, 0]) return numpy.array(dataX), numpy.array(dataY) time_step = 100 X_train, y_train = create_dataset(train_data, time_step) X_test, ytest = create_dataset(test_data, time_step)
```

LSTM

LSTMs are widely used for sequence prediction problems and have proven to be extremely effective. The reason they work so well is that LSTM can store past important information and forget the information that is not.
LSTM has three gates:

- The input gate: The input gate adds information to the cell state,
- The forget gate: It removes the information that is no longer required by the model,
- The output gate: Output Gate at LSTM selects the information to be shown as output.

While Implementing any LSTM, we should always reshape our X train in 3-D, add 1 the reason behind is the time step and the 1 is given to the LSTM.

Code:

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

Then import required modules for the stacked LSTM.

Code:

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

We will be using a sequential model and adding the layers of the LSTM as said, in the above sentence. The first layer should be the time step in 1.

Code:

```
model=Sequential()                                  model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
model.add(LSTM(50,return_sequences=True))           model.add(LSTM(50))           model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

Let's see the summary.

Now the final part is to fit the X_train and the y_train.

Prediction

Predict both the X_train and the X_test, now let's scaler inverse transform because I want to see the root mean square performance.

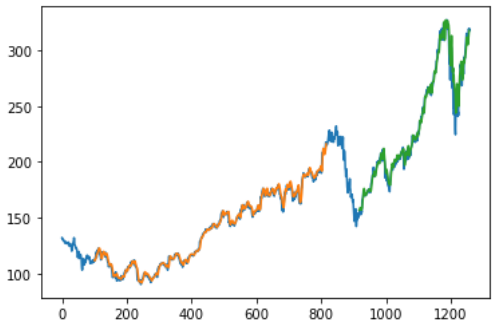
Code:

```
train_predict=model.predict(X_train)                                    test_predict=model.predict(X_test)
train_predict=scaler.inverse_transform(train_predict) test_predict=scaler.inverse_transform(test_predict)
```

Code:

```
import math from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

Here the time step is 100, Whatever the values in train predict and test predict. I got I am just plotting it don't forget we have to inverse the scaler transform.



- Green indicates the Predicted Data
- Blue indicates the Complete Data
- Orange indicates the Train Data

If I consider the last date in the test data as of 22-05-2020, I want to predict the output of 23-05-2020. We need the previous 100 data for that I am taking the data and reshaping it.

Code:

```
x_input=test_data[341:].reshape(1,-1) x_input.shape
```

So, you can predict the prices of preferred stocks using this strategy.

Inference:

Oh my goodness! Various parameters of the LSTM model can be tweaked, such as the number of LSTM layers, the dropout value, and the number of epochs. Are the LSTM projections, however, precise enough to predict whether the stock price will rise or fall? Without a doubt.
As I stated at the outset of this article, stock prices are influenced by company news as well as other factors such as demonetization or company mergers and demergers. In addition, many intangible variables are difficult to predict in advance.

Conclusion:

In this article we have seen how to predict a stock price, this is a simple algorithm. You can make use of auto-ml so that the adding of new data will be easy. Refer a lot of Deep Learning Algorithms, Machine Learning … etc.

Thanks for spending your timing in reading the article.

Article Url - https://www.analyticsvidhya.com/blog/2021/05/stock-price-prediction-and-forecasting-using-stacked-lstm/

SANJAY P