ABHINAV RANJAN
RA1911003010003
CSE A1 SECTION
SRMIST , KTR

AI LAB EXP 10

DEEP LEARNING BASED SOLUTIONS FOR REAL WORLD PROBLEMS

PROBLEM STATEMENT :

Device a model used to predict the closing share price of stocks after 30 days and plot the same on a graph. Collect the necessary previous data and preprocess it (train and test model) and create a stacked LSTM model. Predict the test data and plot the output.

TOOLS USED : python3 , numpy , tensorflow , keras , google colab

METHODOLOGY AND CONCEPTS USED :

1. TIME SERIES PREDICTION

- Time series analysis can be useful to see how a given asset, security or economic variable changes over time.
- It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period.
- For example, suppose we wanted to analyze a time series of daily closing stock prices for a given stock over a period of one year.
- We would obtain a list of all the closing prices for the stock from each day for the past year and list them in chronological order.
- This would be a one-year daily closing price time series for the stock.
- Delving a bit deeper, we might be interested to know whether the stock's time series shows any seasonality to determine if it goes through peaks and valleys at regular times each year.
- The analysis in this area would require taking the observed prices and correlating them to a chosen season. This can include

traditional calendar seasons, such as summer and winter, or retail seasons, such as holiday seasons.

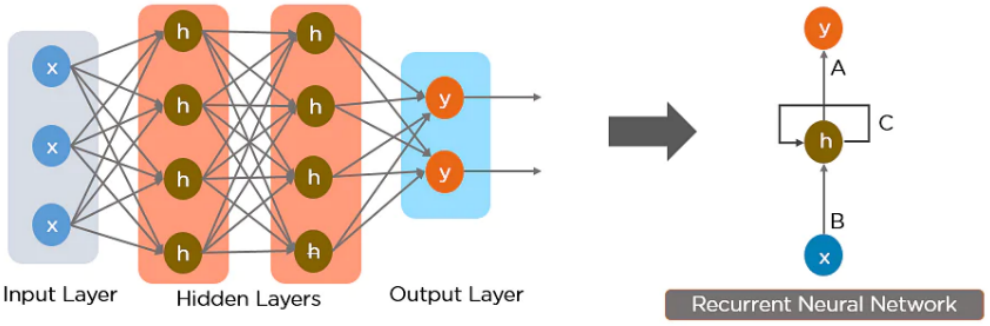
- Alternatively, one can record a stock's share price changes as it relates to an economic variable, such as the unemployment rate.
- By correlating the data points with information relating to the selected economic variable, one can observe patterns in situations exhibiting dependency between the data points and the chosen variable.

2. DATASET AND HOW WE USE IT

- We are using a .csv file which has 1258 records telling about the situation of a company named AAPL in the stock market for over a period of 5 years (from 27th may 2015 to 22nd may 2020).
- It has columns like date , closing price , adjacent close price , adjacent high and low , etc .
- We are going to consider the closing price column and apply MinMaxScaler on it so that the data comes to a scale between 0 to 1 (LSTM are sensitive to the scale of the data and operate better in lower range).
- 65 percent of the rows , that is 817 rows, are put in the train bracket and the remaining 441 rows (35 percent) are put in the test bracket .
- This is the data which goes through the epoch cycle and will help in predicting the stock price after 30 days.

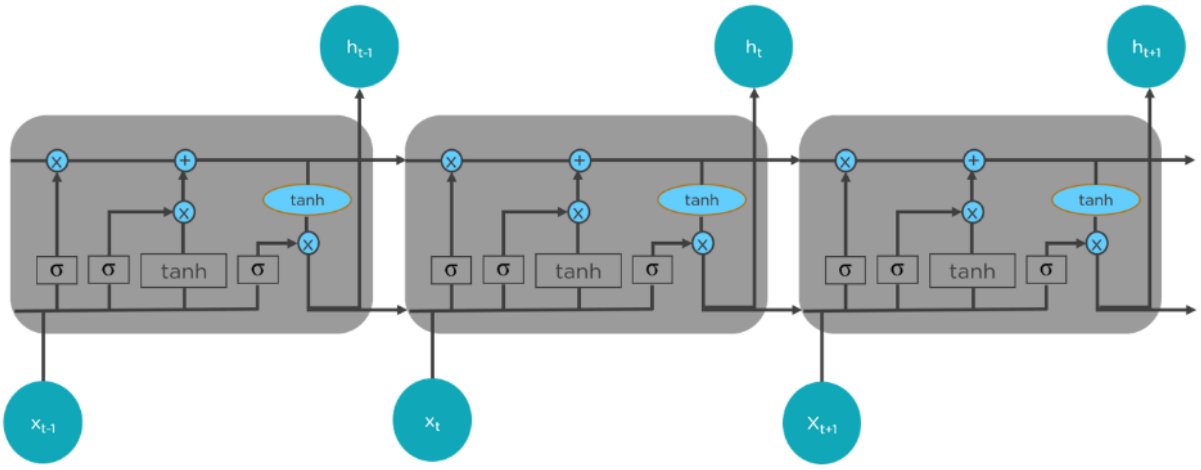
3. RNN (RECURRENT NEURAL NETWORK)

- A Recurrent Neural Network (RNN) is an advanced form of neural networks that has internal memory that makes RNN capable of processing long sequences .
- This makes RNN very suitable for stock price prediction, which involves long historical data.
- RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.



4. LSTM (LONG SHORT TERM MEMORY)

- LSTMs are a type of Recurrent Neural Network(RNN) for learning long-term dependencies. It is commonly used for processing and predicting time-series data.
- LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.
- 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.



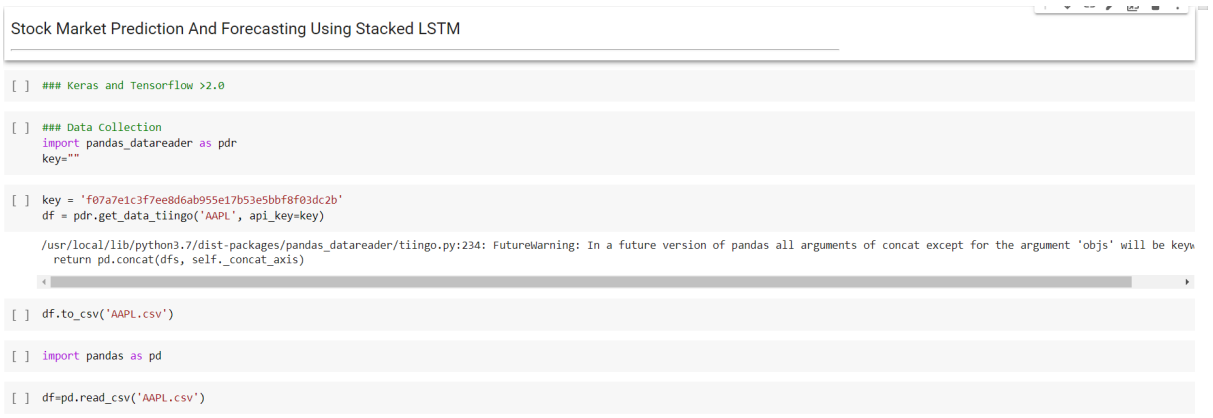
From the image on the top, you can see LSTMs have a chain-like structure. General RNNs have a single neural network layer. LSTMs, on the other hand, have four interacting layers communicating extraordinarily.
LSTMs work in a three-step process.

- 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.

ALGORITHM :

1. Import the data set of the stocks - AAPL
2. Preprocess the data set - train and test
3. Create a stacked LSTM model
4. Predict the test data and plot the output
5. Predict the future 30 days and plot the output

CODE :



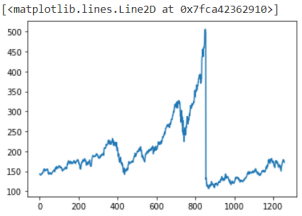
Stock Market Prediction And Forecasting Using Stacked LSTM

```
[ ] ### Keras and Tensorflow >2.0

[ ] ### Data Collection
    import pandas_datareader as pdr
    key=""

[ ] key = 'f07a7e1c3f7ee8d6ab955e17b53e5bbf8f03dc2b'
    df = pdr.get_data_tiingo('AAPL', api_key=key)

    /usr/local/lib/python3.7/dist-packages/pandas_datareader/tiingo.py:234: FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will be keyw
      return pd.concat(dfs, self._concat_axis)

[ ] df.to_csv('AAPL.csv')

[ ] import pandas as pd

[ ] df=pd.read_csv('AAPL.csv')
```

+ Code + Text Copy to Drive Connect Editing

```
[ ] import matplotlib.pyplot as plt
    plt.plot(df1)
```

```
[<matplotlib.lines.Line2D at 0x7fca42362910>]
```




```
[ ] ### LSTM are sensitive to the scale of the data. so we apply MinMax scaler
```

```
[ ] import numpy as np
```

```
[ ] df1
```

```
0       143.17
1       141.63
2       141.80
3       141.05
4       141.83
```

```
[ ] import numpy
    # convert an array of values into a dataset matrix
    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]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
      return numpy.array(dataX), numpy.array(dataY)
```

```
[ ] # reshape into X=t,t+1,t+2,t+3 and Y=t+4
    time_step = 100
    X_train, y_train = create_dataset(train_data, time_step)
    X_test, ytest = create_dataset(test_data, time_step)
```

```
[ ] print(X_train.shape), print(y_train.shape)
```

```
(716, 100)
(716,)
(None, None)
```

```
[ ] print(X_test.shape), print(ytest.shape)
```

```
(340, 100)
(340,)
(None, None)
```

+ Code + Text Copy to Drive Connect Editing

```
[ ] # reshape input to be [samples, time steps, features] which is required for LSTM
    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)
```

```
[ ] ### Create the Stacked LSTM model
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.layers import LSTM
```

```
[ ] 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')
```

```
[ ] model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400

lstm_1 (LSTM)                (None, 100, 50)           20200

lstm_2 (LSTM)                (None, 50)                20200

dense (Dense)                (None, 1)                 51
```

```
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
##Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)
```

```
### Calculate RMSE performance metrics
import math
from sklearn.metrics import mean_squared_error
math.sqrt(mean_squared_error(y_train,train_predict))
```

```
216.8692186892385
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
145.8540982991411
```

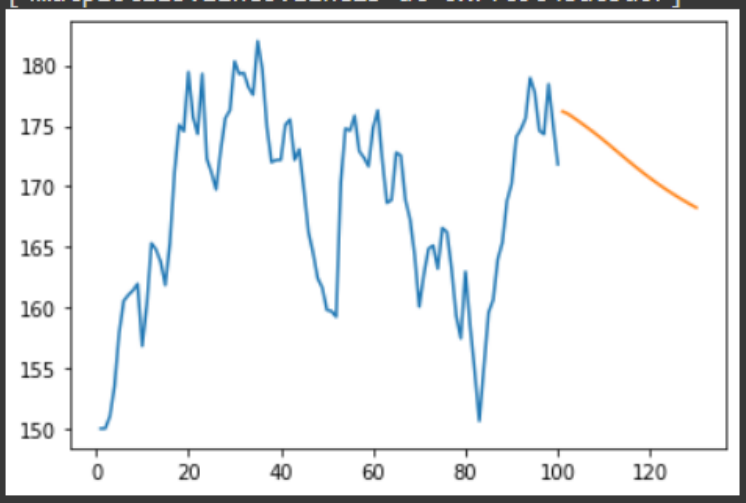
```
### Plotting
# shift train predictions for plotting
look_back=100
trainPredictPlot = numpy.empty_like(df1)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(df1)
```

```
# demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=100
i=0
while(i<30):

    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1
```

OUTPUT :

```
plt.plot(day_new,scaler.inverse_transform(df1[1158:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x7fc9c4bac5d0>]
```



After the training and testing is being done at the end we predict the future 30 days of stock price using the trained model the orange line shows the forecast for apple stocks based on the previous history

RESULT :

Thus we have successfully made a deep learning model to predict the stock prices of a company