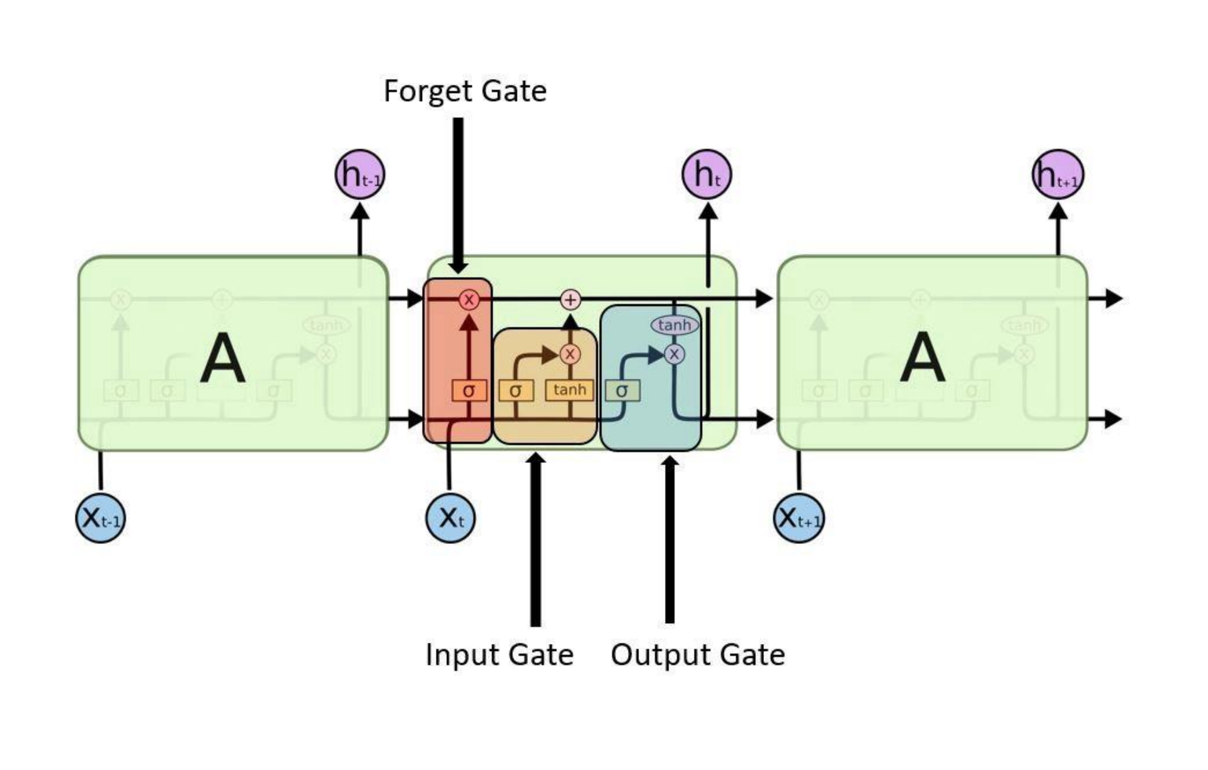
Stock Price Prediction using LSTM

By: Parth Aggarwal

LSTM (a basic introduction)

- Long Short-Term Memory (LSTM) networks are a modified version of Recurrent Neural Networks (RNN).
- This makes it easier to remember past data in memory.
- The vanishing gradient problem of RNN is resolved here.
- LSTM is well-suited to classify, process and predict time series given time lags of unknown duration.
- LSTM trains the model by using back-propagation.



The project

Using a Keras LSTM model to forecast stock trends

Imports

In the first cell, we import:

- NumPy – For making scientific computations
- Pandas – For loading and modifying datasets
- Matplotlib – For plotting graphs

```
[ ]   import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd
```

Loading the data

- We load data of Tesla’s (ticker - TSLA) past stock prices.
- From the data, we select the values of the first and second columns (“Open” and “High” respectively) as our training dataset.
- The “Open” column represents the opening price for shares that day and the “High” column represents the highest price shares reached that day.

```
[ ]
url = r'/content/TSLA train.csv'
dataset_train = pd.read_csv(url)
training_set = dataset_train.iloc[:, 1:2].values
```

Checking the data

- To get a look at the dataset we're using, we can check the head, which shows us the first five rows of our dataset.
- "Open" represents the opening price of the share for that day.
- "High" represents the highest price for the share that day.
- "Low" represents the lowest share price for the day.
- "Close" represents the price shares ended at for the day.

```
[ ]  1 dataset_train.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	01-04-2015	188.699997	192.300003	186.050003	187.589996	187.589996	3794600
1	02-04-2015	190.229996	193.229996	190.000000	191.000000	191.000000	5010400
2	06-04-2015	198.000000	207.750000	197.500000	203.100006	203.100006	12455800
3	07-04-2015	202.509995	205.059998	201.139999	203.250000	203.250000	4347900
4	08-04-2015	208.199997	210.899994	205.869995	207.669998	207.669998	6303100

Data Normalization

- Normalization is changing the values of numeric columns in the dataset to a common scale
  - This helps the performance of our model.

```
[ ]  1 from sklearn.preprocessing import MinMaxScaler
     2 sc = MinMaxScaler(feature_range=(0,1))
     3 training_set_scaled = sc.fit_transform(training_set)
```

- To scale the training dataset we use Scikit-Learn’s MinMaxScaler with numbers between 0 and 1.

Incorporating Timesteps Into Data

- We input our data in the form of a 3D array to the model.
- We create data in 60 timesteps before using NumPy to convert it into an array.
- Then we convert the data into a 3D array with X_train samples, 60 timestamps, and one feature at each step.

```
X_train = []
y_train = []
for i in range(60, 896):
  X_train.append(training_set_scaled[i-60:i, 0])
  y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

More imports

- We need to make a few imports from Keras:
- Sequential for initializing the neural network
- LSTM to add the LSTM layer
- Dropout for preventing overfitting with dropout layers
- Dense to add a densely connected neural network layer.

```
from tensorflow.python.keras import Sequential
from tensorflow.python.keras.layers import LSTM
from tensorflow.python.keras.layers import Dropout
from tensorflow.python.keras.layers import Dense
```

The model

- 50 units is the dimensionality of the output space
- return_sequences = True is necessary for stacking LSTM layers so the consequent LSTM layer has a three-dimensional sequence input
- input_shape is the shape of the training dataset.
- Specifying 0.2 in the Dropout layer means that 20% of the layers will be dropped.
- Now we add the Dense layer that specifies an output of one unit.
- To compile our model we use the Adam optimizer and set the loss as the mean_squared_error.
- After that, we fit the model to run for 50 epochs (the epochs are the number of times the learning algorithm will work through the entire training set) with a batch size of 100.

```
model = Sequential()

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

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

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

model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))

model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(X_train,y_train,epochs=50,batch_size=100)
```

Making predictions

- We now make predictions on the test set.

- We do this by importing the test data set.

  Here, the test set is 30% of the total data set. The rest 70% was used for training the model.

```
url = r'/content/TSLA test.csv'
dataset_test = pd.read_csv(url)
real_stock_price = dataset_test.iloc[:, 1:2].values

url = r'/content/TSLA.csv'
dataset_total = pd.read_csv(url)
```

Modifying the test data set

We need to modify the test set.

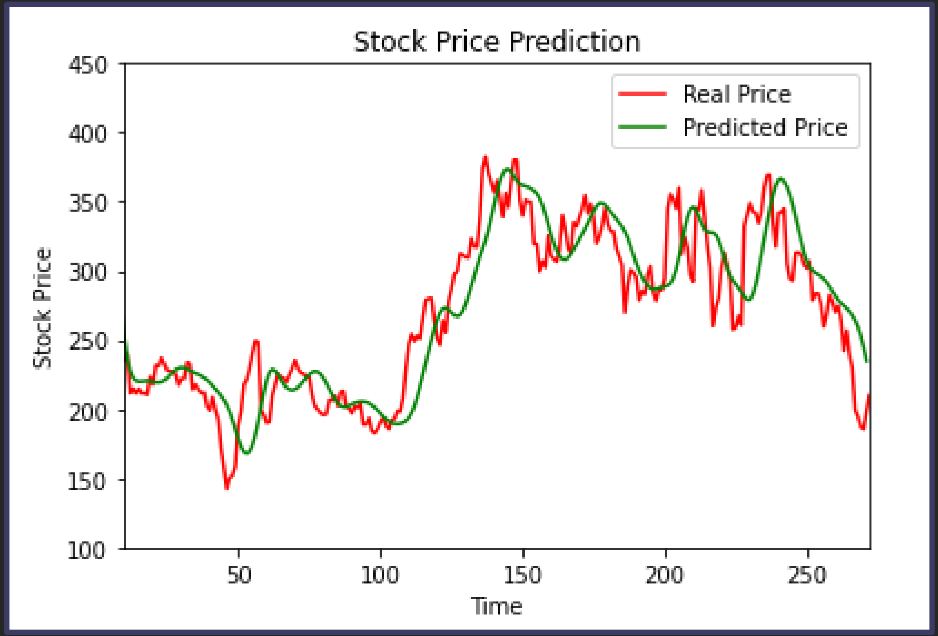
(similar to the procedure used for training set)

- Merge the training set and the test set on the 0 axis.
- Set 60 as the time step again.
- Use MinMaxScaler and reshape data.
- Then, inverse_transform puts the stock prices in a normal readable format.

```
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 = sc.transform(inputs)
X_test = []

for i in range(60, 332):
    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))
predicted_stock_price = model.predict(X_test)

predicted_stock_price =
sc.inverse_transform(predicted_stock_price)
```

Plotting the result

- We use matplotlib to visualize the result of our predicted stock price and the actual stock price.




```
plt.matplotlib.pyplot.xlim(10,272)
plt.matplotlib.pyplot.ylim(100,450)
plt.plot(real_stock_price,
         color = 'red', label = 'Real Price')
plt.plot(predicted_stock_price, color =
         'green', label = 'Predicted Price')
plt.title('Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

Google drive link to the code (data sets included)

https://drive.google.com/open?id=1U8x4Sp0m8R82FEQqICC__05QD46Cmnc1

References

- https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e
- Derrick Mwiti's GitHub profile
- Stack Overflow
- Towards Data Science magazine
- Python.org
- Yahoo finance
- Tableau software
- Google Colaboratory
- Tensorflow.org
- Keras.io
- https://www.youtube.com/channel/UCiT9RITQ9PW6BhXK0y2jaeg/featured

Thank you!