DS620 Machine Learning and Deep Learning

HOS10A – RNN

05/29/2021 Developed by Minh Nguyen

06/01/2021 Reviewed by Shanshan Yu

School of Technology & Computing @City University of Seattle (CityU)

Learning objectives

- RNN
- LSTM

Resources

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc.
- Loukas, S. (2020, July 31). Time-Series Forecasting: Predicting Stock Prices Using An LSTM Model. Medium. https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f

Introduction

Last week, we learned about CNN a type of neural network that specializes in processing images. This week, we will learn that image processing is not the only subdomain of Deep Learning. Another popular type of data in Deep Learning is sequential data. Sequential data are data that is organized in a sequential manner, which includes written text, audio clips, video clips, time-series data, etc. Recurrent Neural Network is a popular algorithm used for sequence modeling.

I. Recurrent Neural Network

A Recurrent Neural Network is very similar to the Feed Forward Neural Network but the signal can also flow backward. This allows it to learn the trends and inclination inside the data. RNNs are mostly used in the field of Natural Language Processing. RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are used in time series predictions as well.

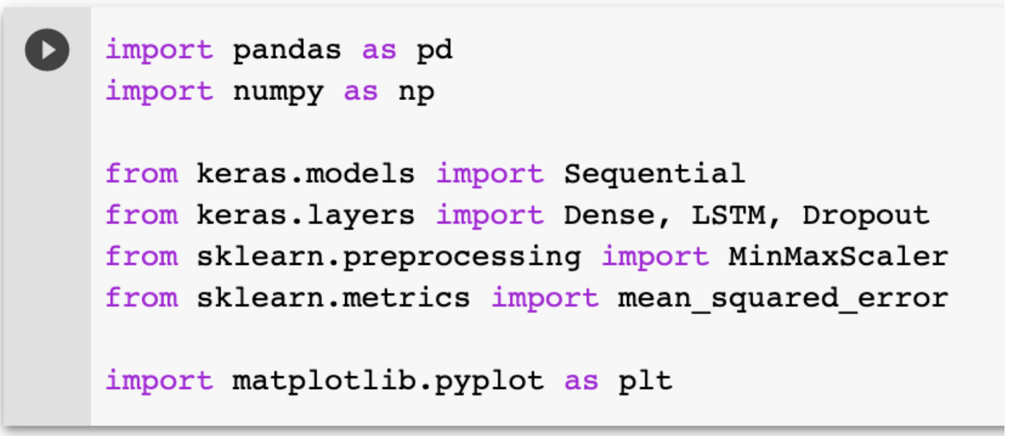
II. Long Short-Term Memory

Traditional RNNs are not good at capturing long-range dependencies. The reason is mainly due to the vanishing gradient problem that we discussed in module 7. Long Short-Term Memory is a modification to the RNN hidden layer that is used to tackle this problem. LSTM has enabled RNNs to remember its inputs over a long period of time. In this exercise, we will use LSTM cells to create a RNN model that can predict stock price. Please be advised that the following code only serves as an example of how sequential data can be processed with RNN. Building a model that can predict the real-world stock market is a complicated task. The stock market, in general, has a very different statistical characteristic; past performance is not a sufficient predictor for future returns.

III. Practice

Preparing development environment

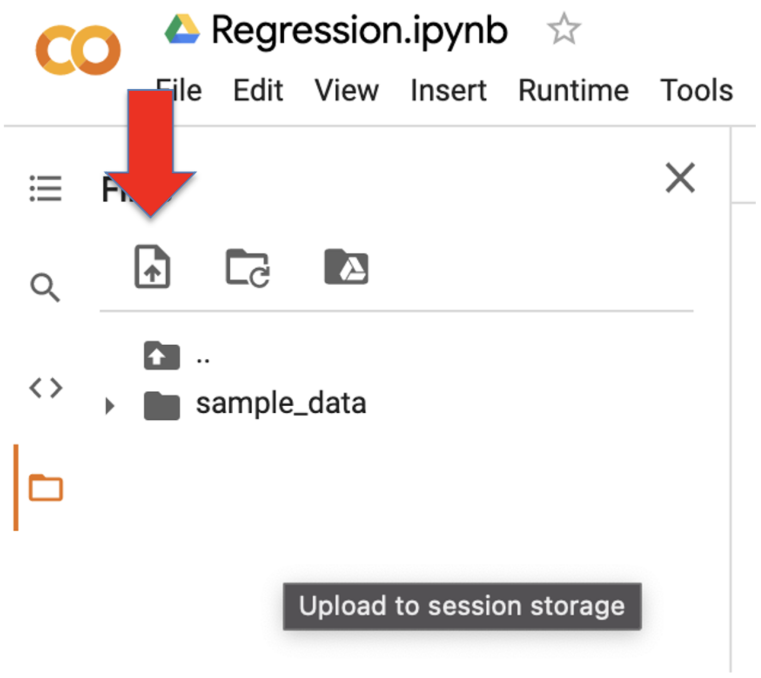
1. From Google Colab, create a new notebook, name it "LSTM.ipynb"
2. Type the following codes to import libraries.



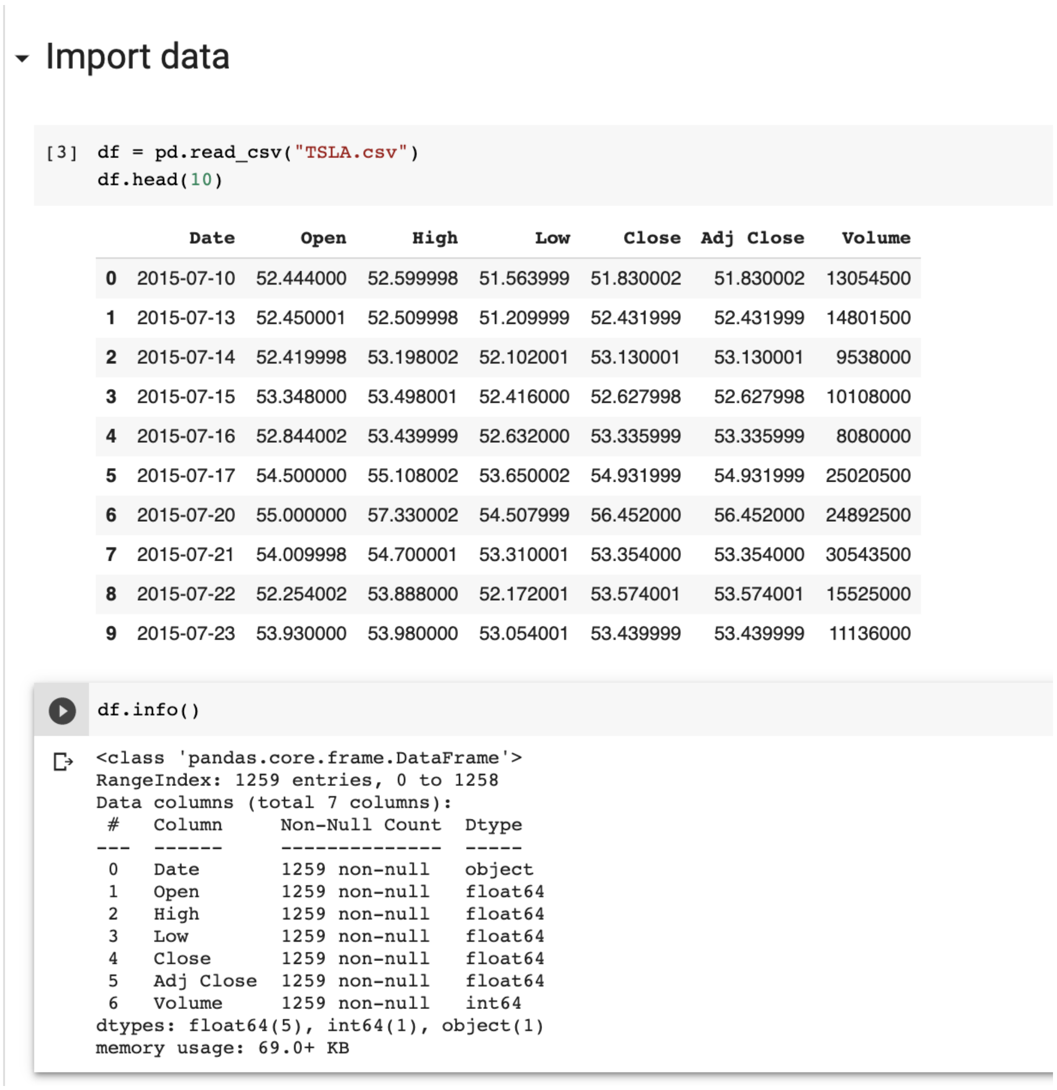
```
import pandas as pd
import numpy as np

from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

import matplotlib.pyplot as plt
```

Get the data

1. Upload the TSLA dataset to Google Colab



2. Run the following code to import the data and view the data information. In this exercise, we will only be working with one column. The ‘Open’ column



Import data

```
[3] df = pd.read_csv("TSLA.csv")
    df.head(10)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-07-10	52.444000	52.599998	51.563999	51.830002	51.830002	13054500
1	2015-07-13	52.450001	52.509998	51.209999	52.431999	52.431999	14801500
2	2015-07-14	52.419998	53.198002	52.102001	53.130001	53.130001	9538000
3	2015-07-15	53.348000	53.498001	52.416000	52.627998	52.627998	10108000
4	2015-07-16	52.844002	53.439999	52.632000	53.335999	53.335999	8080000
5	2015-07-17	54.500000	55.108002	53.650002	54.931999	54.931999	25020500
6	2015-07-20	55.000000	57.330002	54.507999	56.452000	56.452000	24892500
7	2015-07-21	54.009998	54.700001	53.310001	53.354000	53.354000	30543500
8	2015-07-22	52.254002	53.888000	52.172001	53.574001	53.574001	15525000
9	2015-07-23	53.930000	53.980000	53.054001	53.439999	53.439999	11136000

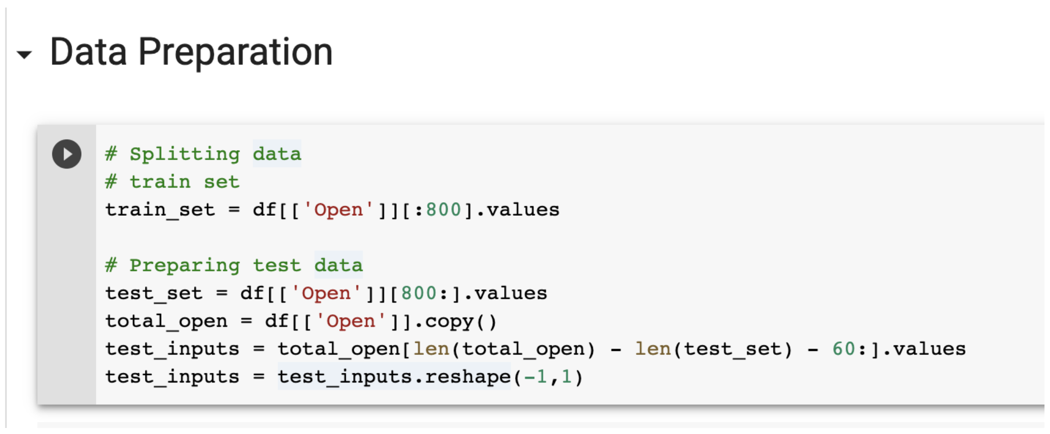
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1259 non-null   object
 1   Open       1259 non-null   float64
 2   High       1259 non-null   float64
 3   Low        1259 non-null   float64
 4   Close      1259 non-null   float64
 5   Adj Close  1259 non-null   float64
 6   Volume     1259 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 69.0+ KB
```

Data Preparation

As mentioned previously, RNN is specialized in training with sequence, we need to format the data in a way that the input is a chain of sequences.

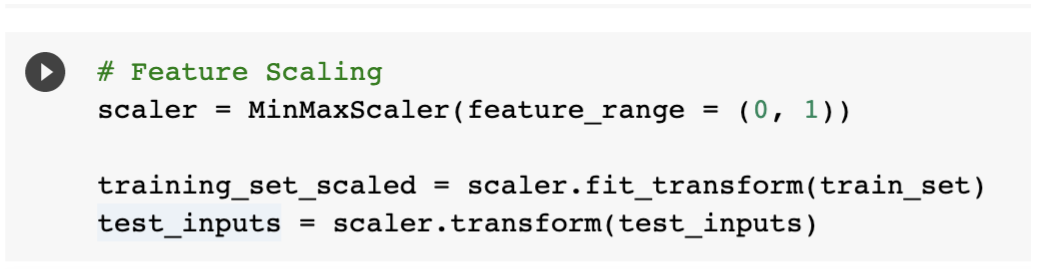
1. First, we will split the dataset. Since this data is sequential, we are not resampling them.



Data Preparation

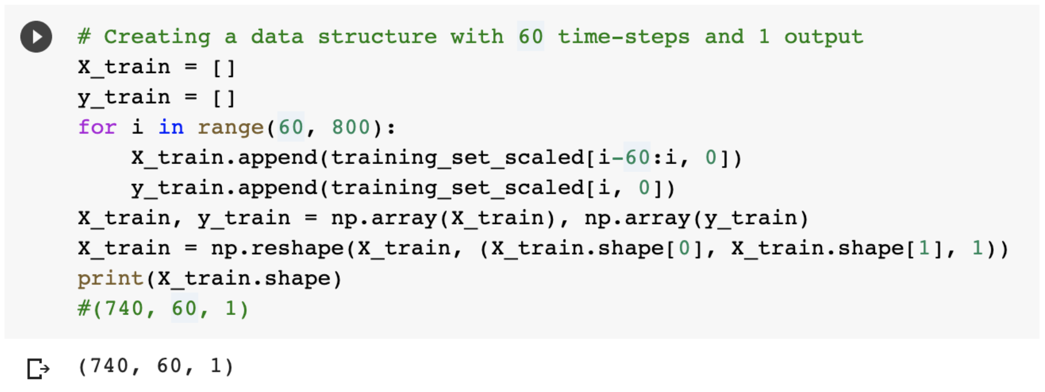
```
# Splitting data
# train set
train_set = df[['Open']][:800].values

# Preparing test data
test_set = df[['Open']][800:].values
total_open = df[['Open']].copy()
test_inputs = total_open[len(total_open) - len(test_set) - 60:].values
test_inputs = test_inputs.reshape(-1,1)
```

2. Next, we will rescale both data set.



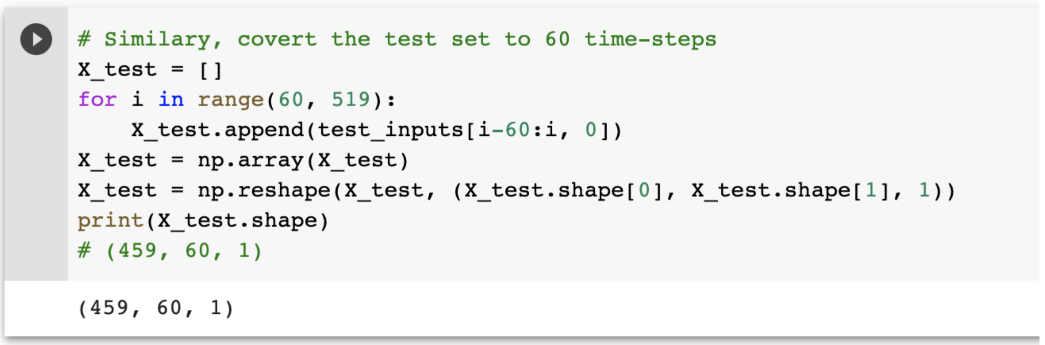
```
# Feature Scaling
scaler = MinMaxScaler(feature_range = (0, 1))

training_set_scaled = scaler.fit_transform(train_set)
test_inputs = scaler.transform(test_inputs)
```

3. We will format the both the input data set as sequences of 60 consecutive days while the label is going to be the entire column.



```
# Creating a data structure with 60 time-steps and 1 output
X_train = []
y_train = []
for i in range(60, 800):
    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))
print(X_train.shape)
#(740, 60, 1)
```

```
(740, 60, 1)
```

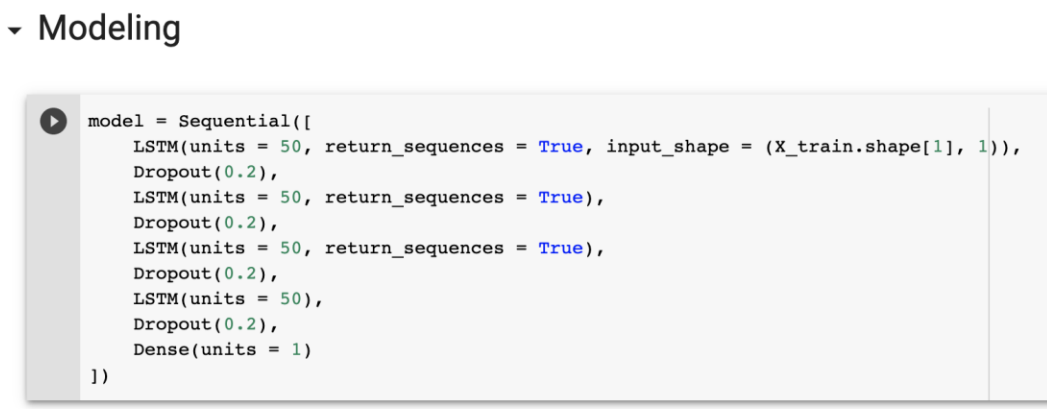


```
# Similary, covert the test set to 60 time-steps
X_test = []
for i in range(60, 519):
    X_test.append(test_inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
print(X_test.shape)
# (459, 60, 1)
```

```
(459, 60, 1)
```

Modeling

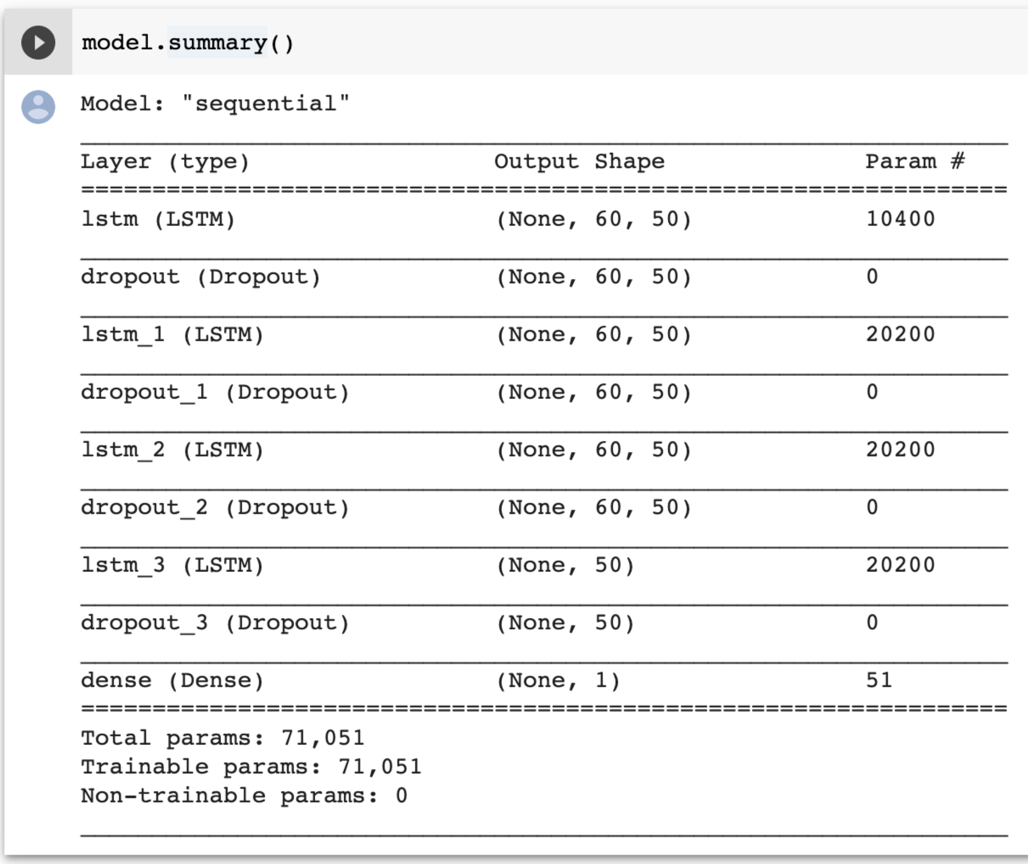
1. Run the following code to build a network with LSTM layers



▾ Modeling

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

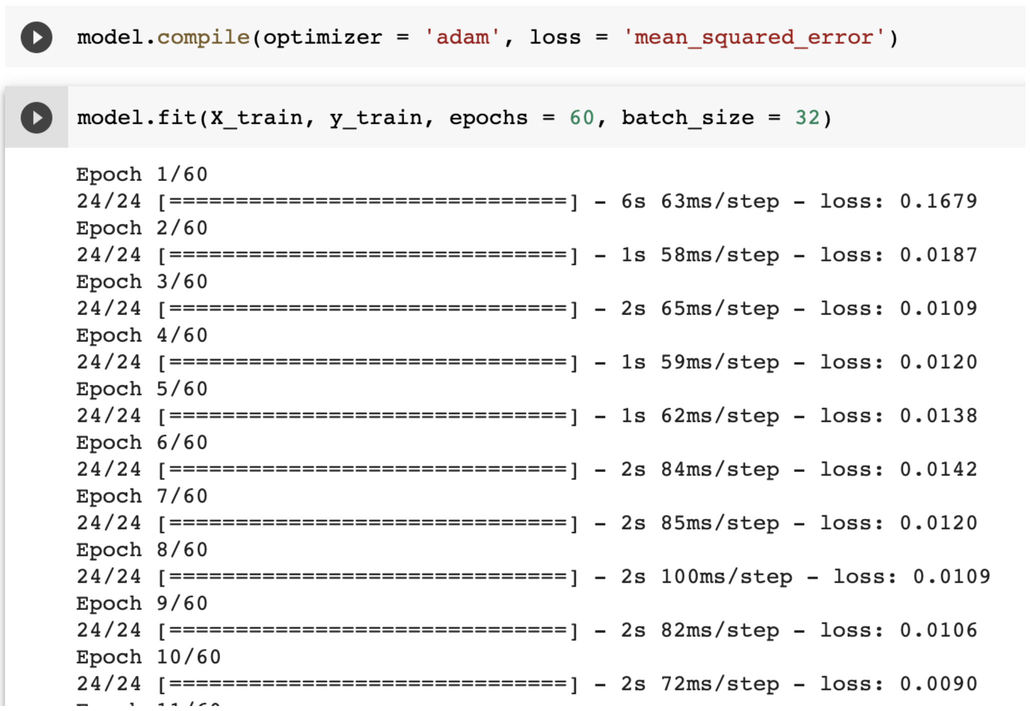
2. Run the following code to see the layers within this model



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 60, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 60, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 50)            20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 50)            20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 50)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 50)                20200
_________________________________________________________________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

3. Run the following code to compile your model and train it.


```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

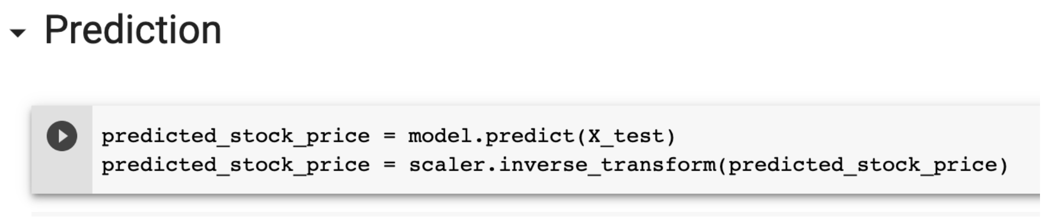
```
model.fit(X_train, y_train, epochs = 60, batch_size = 32)
```

```
Epoch 1/60
24/24 [==============================] - 6s 63ms/step - loss: 0.1679
Epoch 2/60
24/24 [==============================] - 1s 58ms/step - loss: 0.0187
Epoch 3/60
24/24 [==============================] - 2s 65ms/step - loss: 0.0109
Epoch 4/60
24/24 [==============================] - 1s 59ms/step - loss: 0.0120
Epoch 5/60
24/24 [==============================] - 1s 62ms/step - loss: 0.0138
Epoch 6/60
24/24 [==============================] - 2s 84ms/step - loss: 0.0142
Epoch 7/60
24/24 [==============================] - 2s 85ms/step - loss: 0.0120
Epoch 8/60
24/24 [==============================] - 2s 100ms/step - loss: 0.0109
Epoch 9/60
24/24 [==============================] - 2s 82ms/step - loss: 0.0106
Epoch 10/60
24/24 [==============================] - 2s 72ms/step - loss: 0.0090
```

Prediction

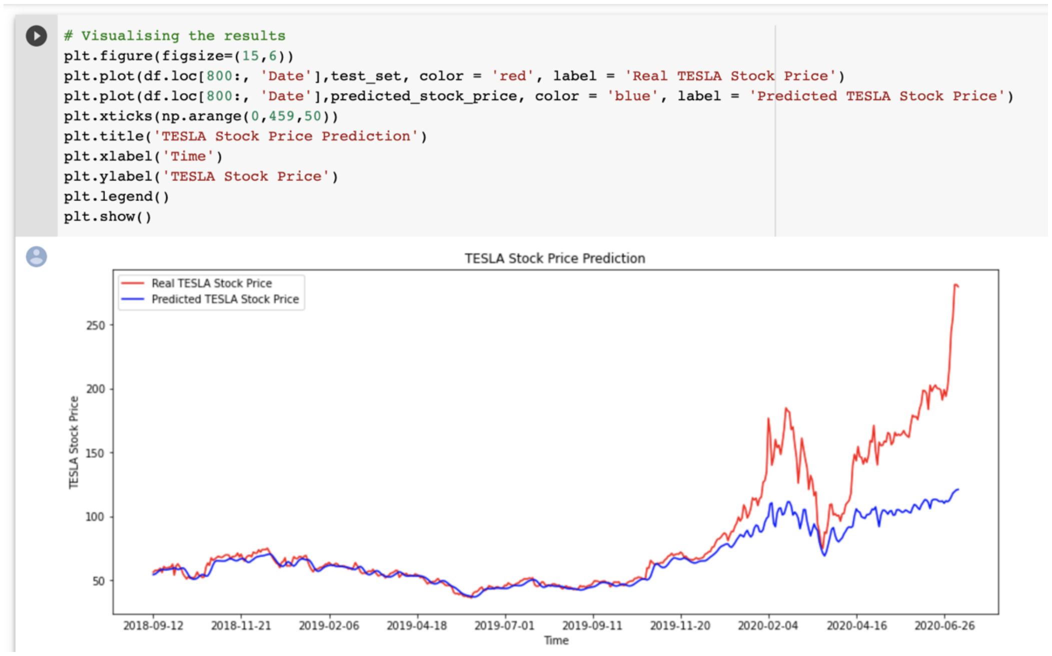
With the model fully trained, it's time to evaluate it. However, for this exercise, we will visualize the predicted sequence and compare it visually with the true sequence.

1. Run the following code to run prediction and rescaled the predicted stock price.


Prediction

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

2. Run the following code to visualize both sequences.



```
# Visualising the results
plt.figure(figsize=(15,6))
plt.plot(df.loc[800:, 'Date'],test_set, color = 'red', label = 'Real TESLA Stock Price')
plt.plot(df.loc[800:, 'Date'],predicted_stock_price, color = 'blue', label = 'Predicted TESLA Stock Price')
plt.xticks(np.arange(0,459,50))
plt.title('TESLA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TESLA Stock Price')
plt.legend()
plt.show()
```

We can see that the model did a good job at predicting the price up to 2019. After this point int time, the model did successfully predict the up and down of the price but it didn't predict the correct magnitude of the change in price.

Push Your Work to GitHub

Download the notebook from Colab:

File -> LSTM.ipynb

Move the downloaded file into your **Module10** working folder.

Open terminal and Navigate to the GitHub folder of this week HOS.

Make sure the assignment files on the subfolder Module10 of hos10a_YouGithubUserName folder, enter the following command to upload your work:

>>>> git add .

>>>> git commit -m "Submission for HOS10"

>>>> git push origin master