Prediction of Stock Prices using LSTMs

SAISRINIVAS MAMUNURU
2020A7PS1014G

Stock Market

- A stock market is a public market for the trading of company stocks. Stock market allows us to buy and sell units of stocks (ownership) of a company. If the company's profits go up, then we own some of the profits and if they go down, then lose profits with them.
- If more sellers than buyers, stock prices tend to fall. Conversely, when. more buyers than sellers, stock prices tend to rise

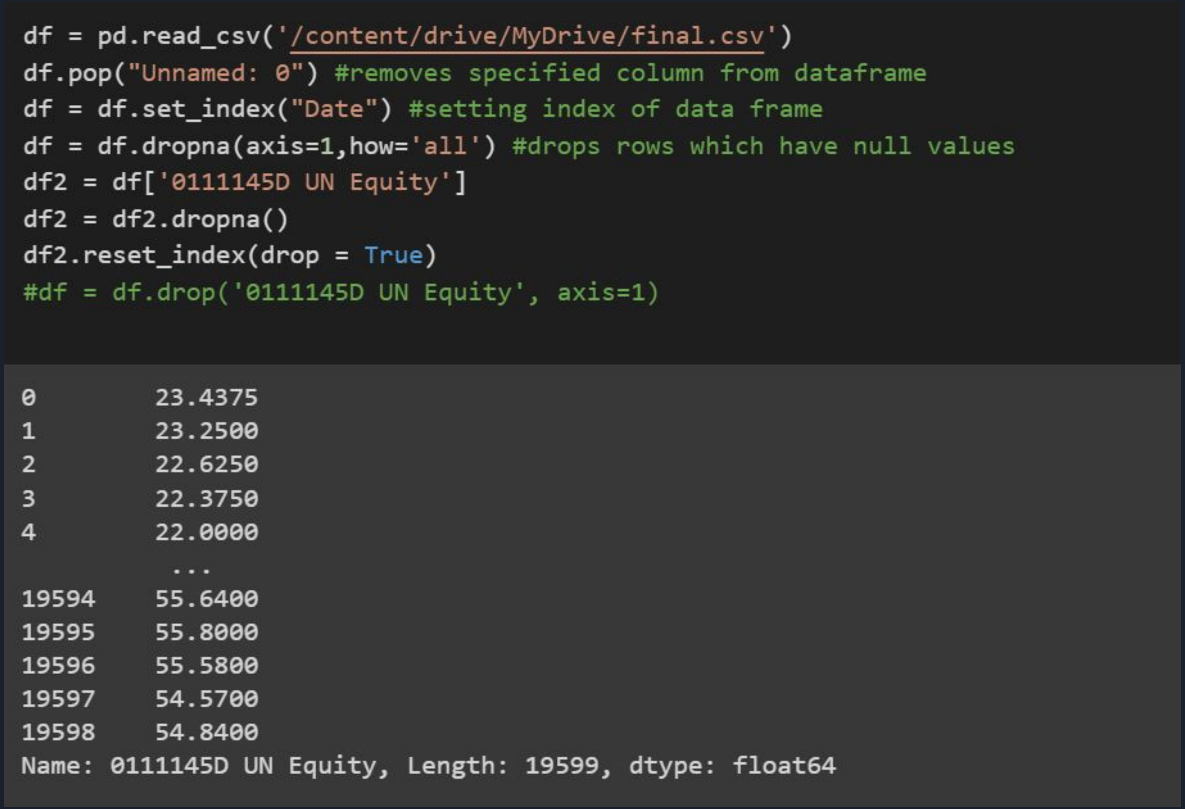
Problem Statement:

- Use both LSTM networks as training methodologies to analyze their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the UN Equity from January 1993 till December 2018 for intraday trading.

Tools used:

- **Pandas** : Pandas is a Python library that provides tools for data manipulation and analysis, including functions for reading, manipulating, and summarizing data in a variety of formats.
- **Numpy** : NumPy is a Python library for working with large, multi-dimensional arrays and matrices of numerical data, providing functions for performing mathematical operations on these
- **Matplotlib**: is a Python library for creating static, animated, and interactive visualizations in Python.

Data Set

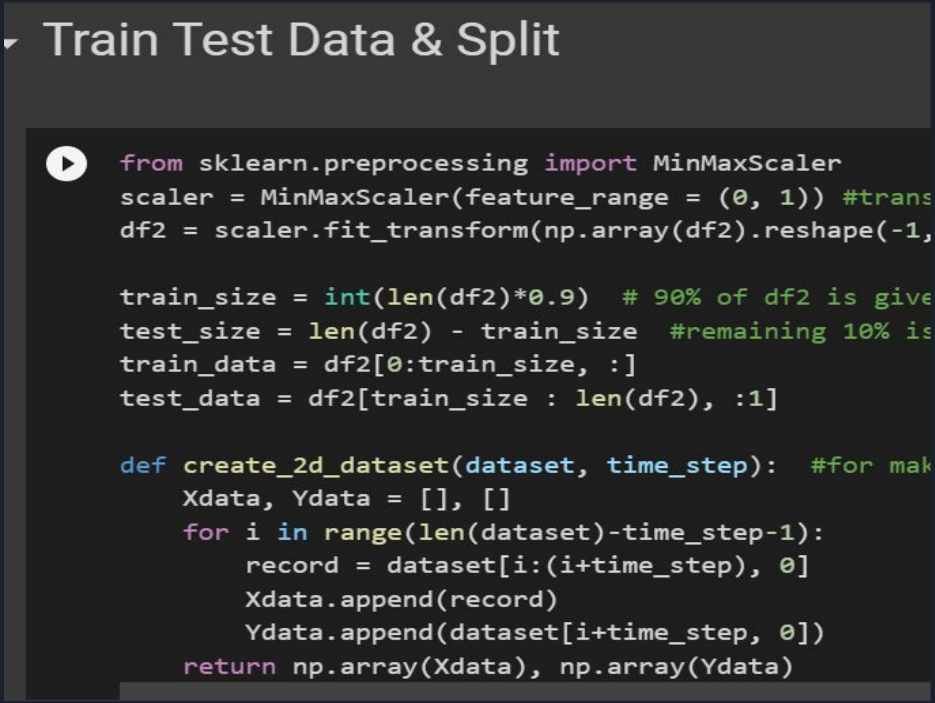


```
df = pd.read_csv('/content/drive/MyDrive/final.csv')
df.pop("Unnamed: 0") #removes specified column from dataframe
df = df.set_index("Date") #setting index of data frame
df = df.dropna(axis=1,how='all') #drops rows which have null values
df2 = df['0111145D UN Equity']
df2 = df2.dropna()
df2.reset_index(drop = True)
#df = df.drop('0111145D UN Equity', axis=1)
```

```
0        23.4375
1        23.2500
2        22.6250
3        22.3750
4        22.0000
          ...
19594    55.6400
19595    55.8000
19596    55.5800
19597    54.5700
19598    54.8400
Name: 0111145D UN Equity, Length: 19599, dtype: float64
```

Represents the data set used and setting the data into data frames.

Data Preprocessing:

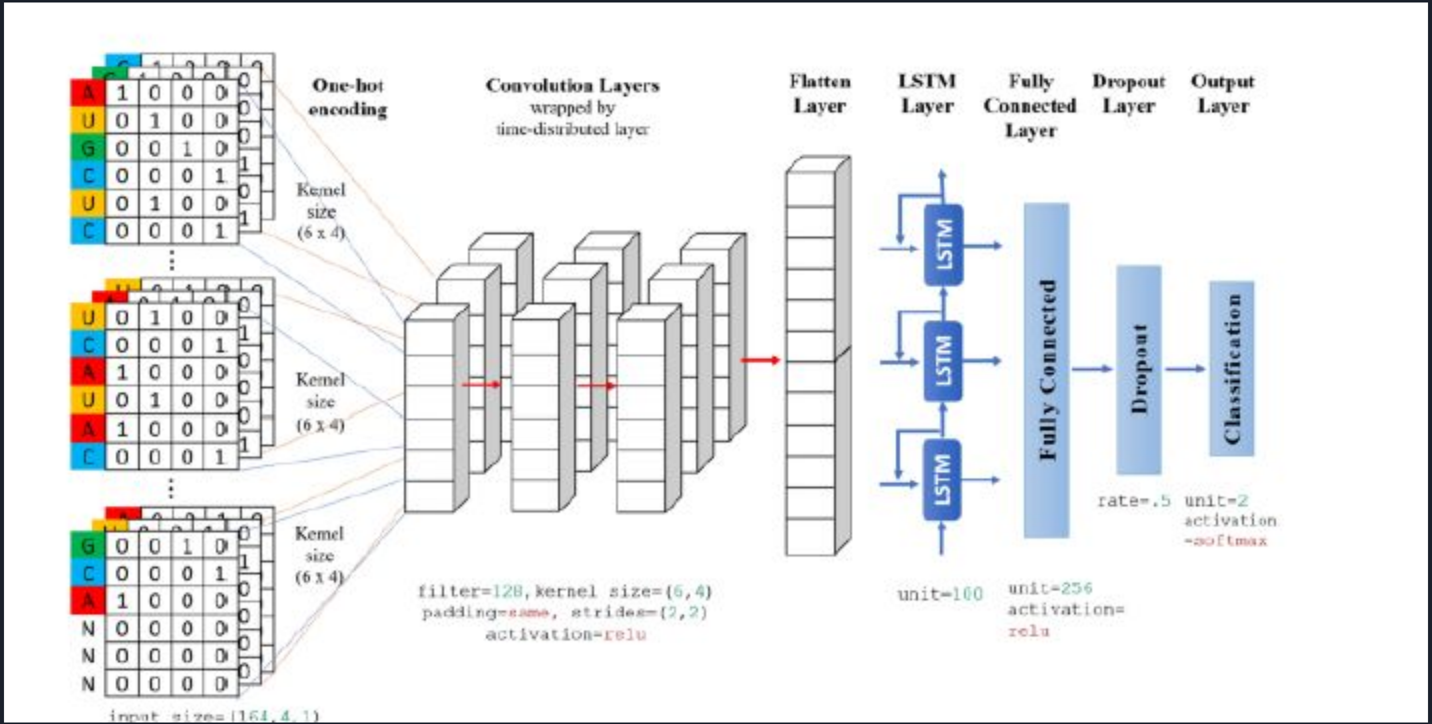


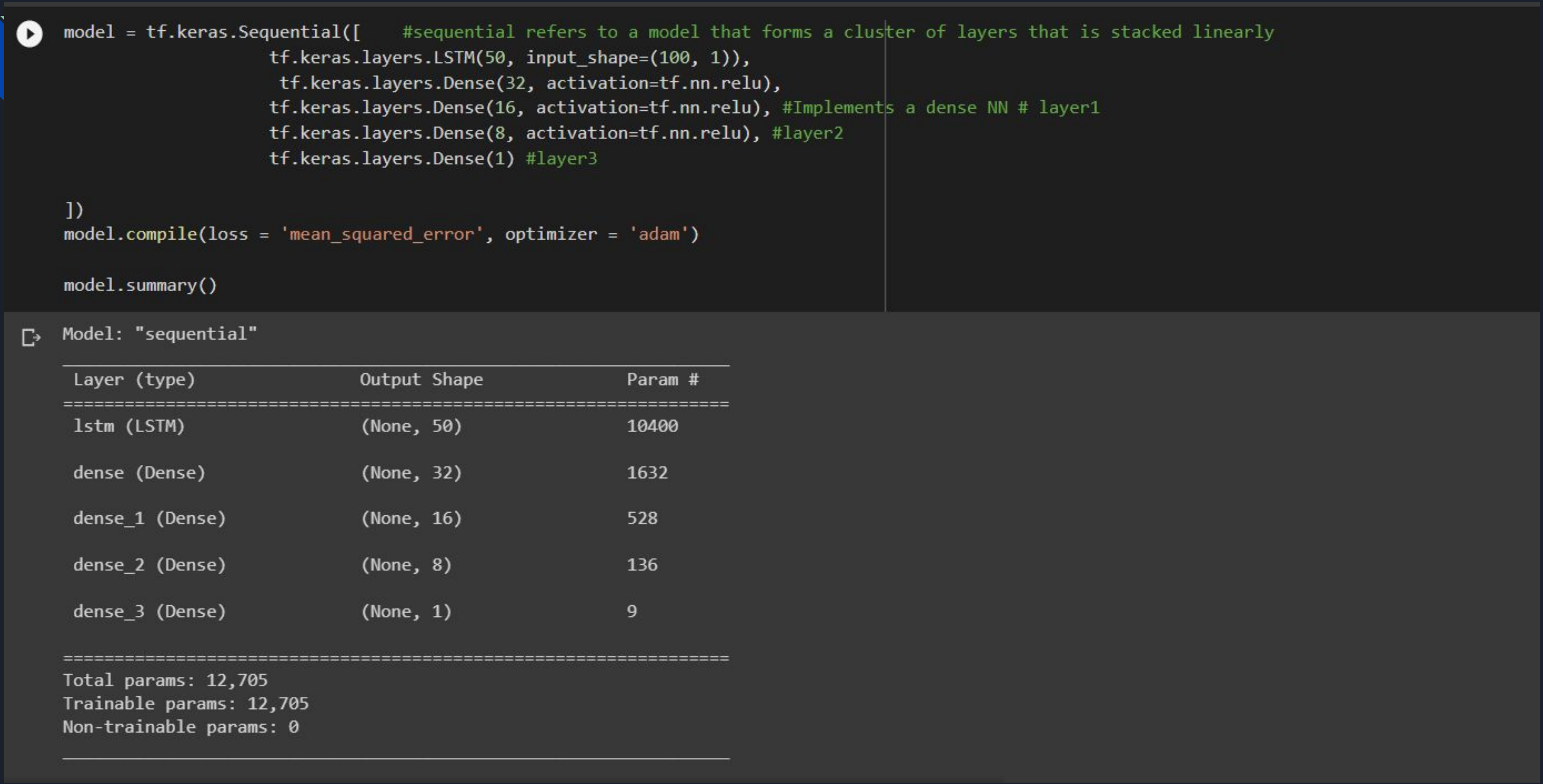
Train Test Data & Split

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0, 1)) #trans
df2 = scaler.fit_transform(np.array(df2).reshape(-1,

train_size = int(len(df2)*0.9)  # 90% of df2 is give
test_size = len(df2) - train_size  #remaining 10% is
train_data = df2[0:train_size, :]
test_data = df2[train_size : len(df2), :1]

def create_2d_dataset(dataset, time_step):  #for mak
    Xdata, Ydata = [], []
    for i in range(len(dataset)-time_step-1):
        record = dataset[i:(i+time_step), 0]
        Xdata.append(record)
        Ydata.append(dataset[i+time_step, 0])
    return np.array(Xdata), np.array(Ydata)
```

Model Used: CNN-LSTM

- CNN-LSTM (Convolutional Neural Network-Long Short-Term Memory) is a type of neural network that combines the strengths of both CNN and LSTM networks.
- A Convolutional Neural Network is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.
- By combining the strengths of both CNNs and LSTMs, a CNN-LSTM network can be used for tasks such as image captioning, where the input is an image and the output is a description of the image in natural language. The CNN portion of the network can extract features from the image,while the LSTM portion can generate the appropriate language description.

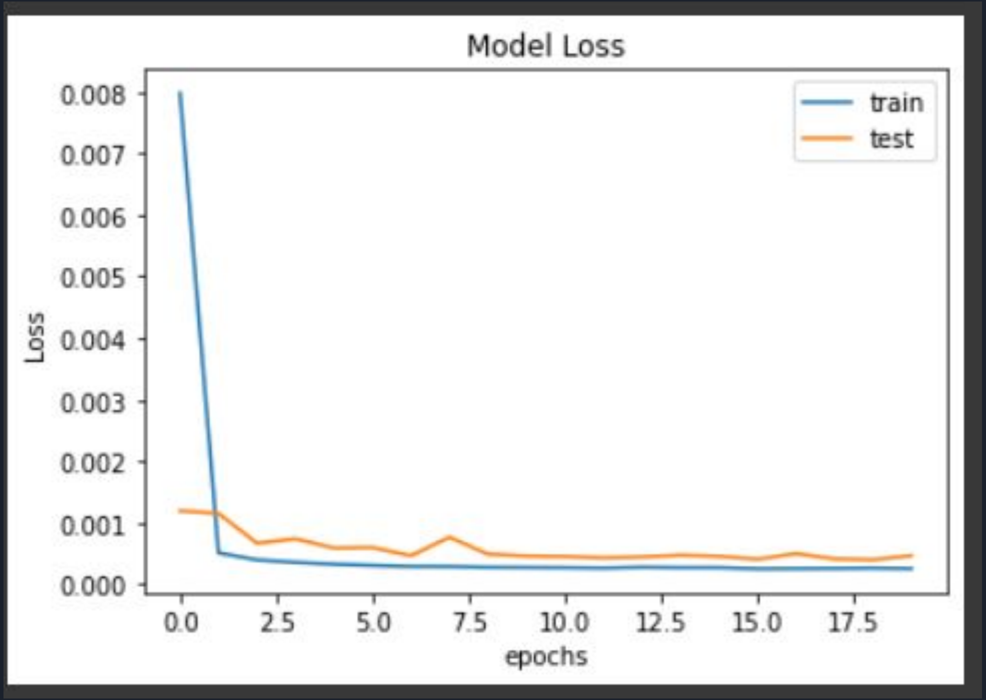

A 1 0 0 0
U 0 1 0 0
G 0 0 1 0
C 0 0 0 1
U 0 1 0 0
C 0 0 0 1
U 0 1 0 0
C 0 0 0 1
A 1 0 0 0
U 0 1 0 0
A 1 0 0 0
C 0 0 0 1
G 0 0 1 0
C 0 0 0 1
A 1 0 0 0
N 0 0 0 0
N 0 0 0 0
N 0 0 0 0
input size=(164,4,1)
One-hot encoding
Kernel size (6 x 4)
Kernel size (6 x 4)
Kernel size (6 x 4)
Convolution Layers
wrapped by
time-distributed layer
filter=128, kernel size=(6,4)
padding=same, strides=(2,2)
activation=relu
Flatten Layer
LSTM Layer
LSTM
LSTM
LSTM
unit=100
Fully Connected Layer
Fully Connected
unit=256
activation=
relu
Dropout Layer
Dropout
rate=.5
Output Layer
Classification
unit=2
activation
=softmax



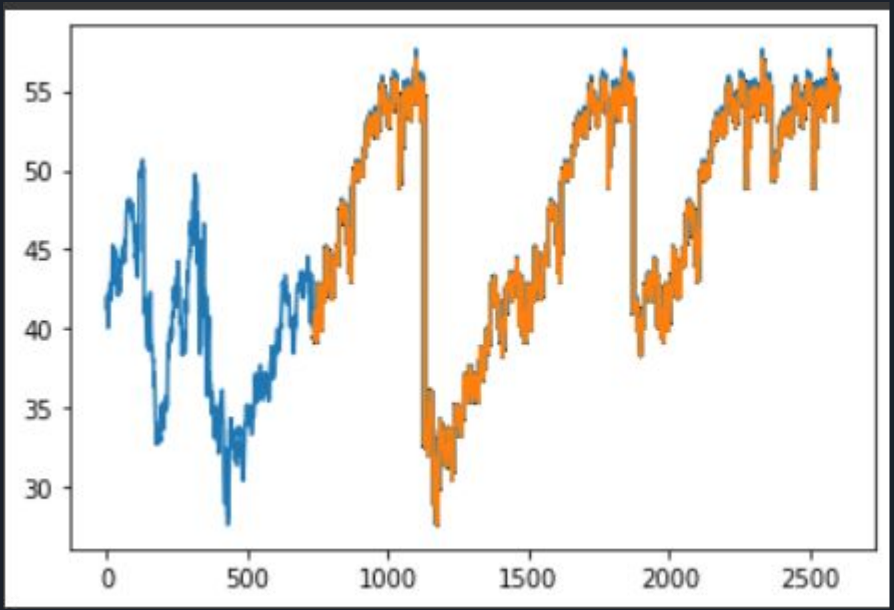
```
model = tf.keras.Sequential([    #sequential refers to a model that forms a cluster of layers that is stacked linearly
                    tf.keras.layers.LSTM(50, input_shape=(100, 1)),
                     tf.keras.layers.Dense(32, activation=tf.nn.relu),
                    tf.keras.layers.Dense(16, activation=tf.nn.relu), #Implements a dense NN # layer1
                    tf.keras.layers.Dense(8, activation=tf.nn.relu), #layer2
                    tf.keras.layers.Dense(1) #layer3

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

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 lstm (LSTM)                 (None, 50)                10400     
                                                                 
 dense (Dense)               (None, 32)                1632      
                                                                 
 dense_1 (Dense)             (None, 16)                528       
                                                                 
 dense_2 (Dense)             (None, 8)                 136       
                                                                 
 dense_3 (Dense)             (None, 1)                 9         
                                                                 
=================================================================
Total params: 12,705
Trainable params: 12,705
Non-trainable params: 0
_________________________________________________________________
```

Performance



Model Plot and Result:



- Orange represents the prediction we have made.
- We can observe it is quite close to the blue line.
- Accuracy obtained is around 90+ % .

References:

- https://scholar.google.co.in/scholar?q=Forecasting+directional+movements+of+stock+prices+for+intraday+trading+using+LSTM+and+random+forests&hl=en&as_sdt=0&as_vis=1&oi=scholart
- https://www.sciencedirect.com/science/article/pii/S1062940822000572

Thank You