

# Internship Report

## INFOSYS Stock Price Prediction(RNN)

Name: Ankita Gupta

Course: ML1119

Duration: 2 Weeks

Problem Statement: Importing necessary libraries,

---

Import Dataset from the given link, Perform exploratory data analysis and Creating Training and Test Data.

Github link: <https://github.com/Ankita30-ui/Car-Sales-Purchase-Prediction>

Dataset used

---

The data source used for this project is train.xlsx and test.xlsx.

Data Set Link: <https://www.kaggle.com/rohanrao/nifty50-stock-market-data?select=INFY.csv>

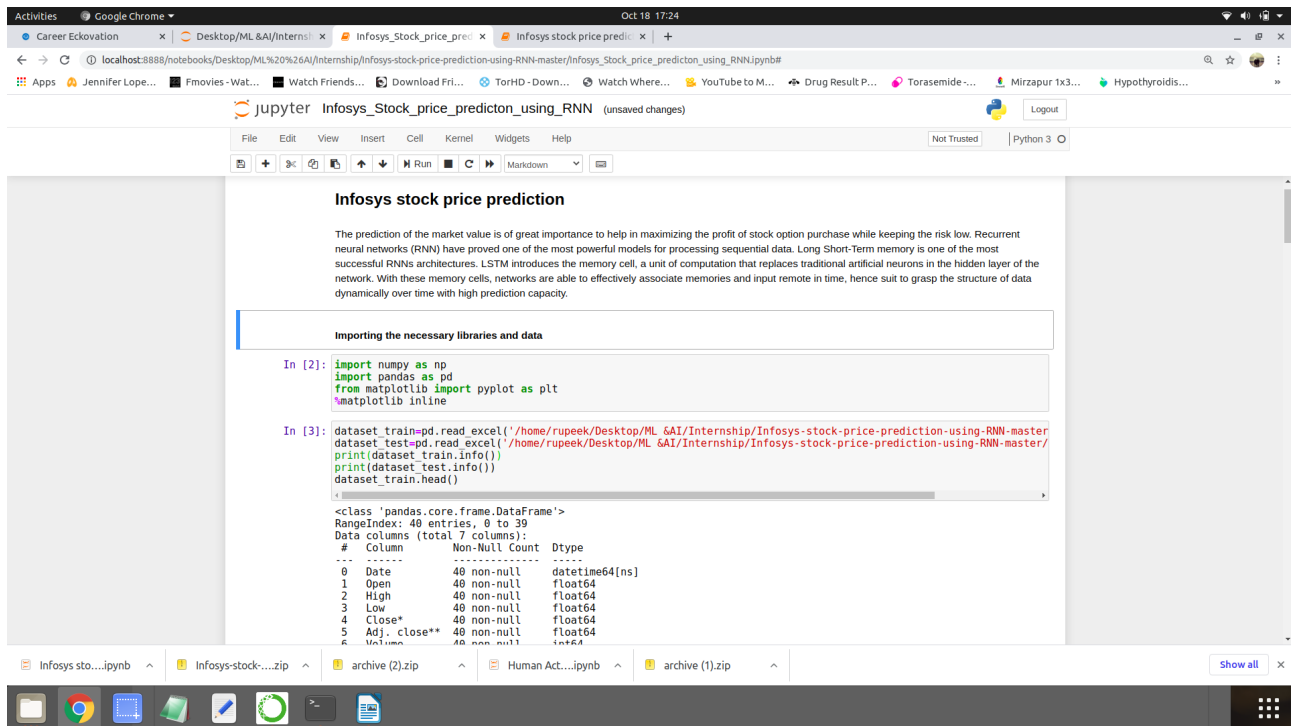
Applying algorithms

---

RNN

---

## 1.Importing the libraries:



**Infosys stock price prediction**

The prediction of the market value is of great importance to help in maximizing the profit of stock option purchase while keeping the risk low. Recurrent neural networks (RNN) have proved one of the most powerful models for processing sequential data. Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

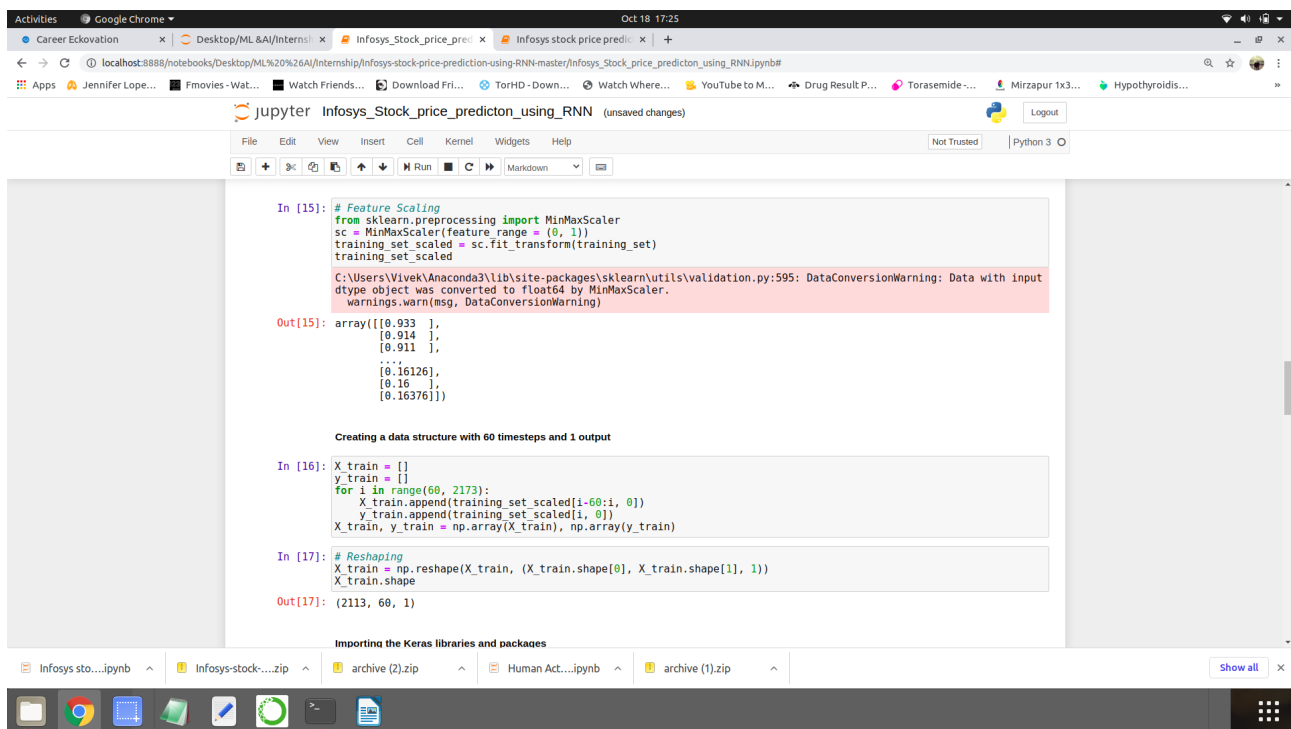
**Importing the necessary libraries and data**

```
In [2]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

In [3]: dataset_train=pd.read_excel('/home/rupeek/Desktop/ML &AI/Internship/Infosys-stock-price-prediction-using-RNN-master/Infosys_Stock_price_prediction_using_RNN.ipynb#')
dataset_test=pd.read_excel('/home/rupeek/Desktop/ML &AI/Internship/Infosys-stock-price-prediction-using-RNN-master/Infosys_Stock_price_prediction_using_RNN.ipynb#')
print(dataset_train.info())
print(dataset_test.info())
dataset_train.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   Date        40 non-null     datetime64[ns]
 1   Open        40 non-null     float64
 2   High        40 non-null     float64
 3   Low         40 non-null     float64
 4   Close*      40 non-null     float64
 5   Adj. close** 40 non-null     float64
 6   Volume      40 non-null     float64
```

## 2. Featureselection:



```
In [15]: # Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
training_set_scaled

C:\Users\Vivek\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype object was converted to float64 by MinMaxScaler.
warnings.warn(msg, DataConversionWarning)
```

```
Out[15]: array([[0.933 ],
 [0.914 ],
 [0.911 ],
 ...,
 [0.16126],
 [0.16 ],
 [0.16376]])
```

**Creating a data structure with 60 time steps and 1 output**

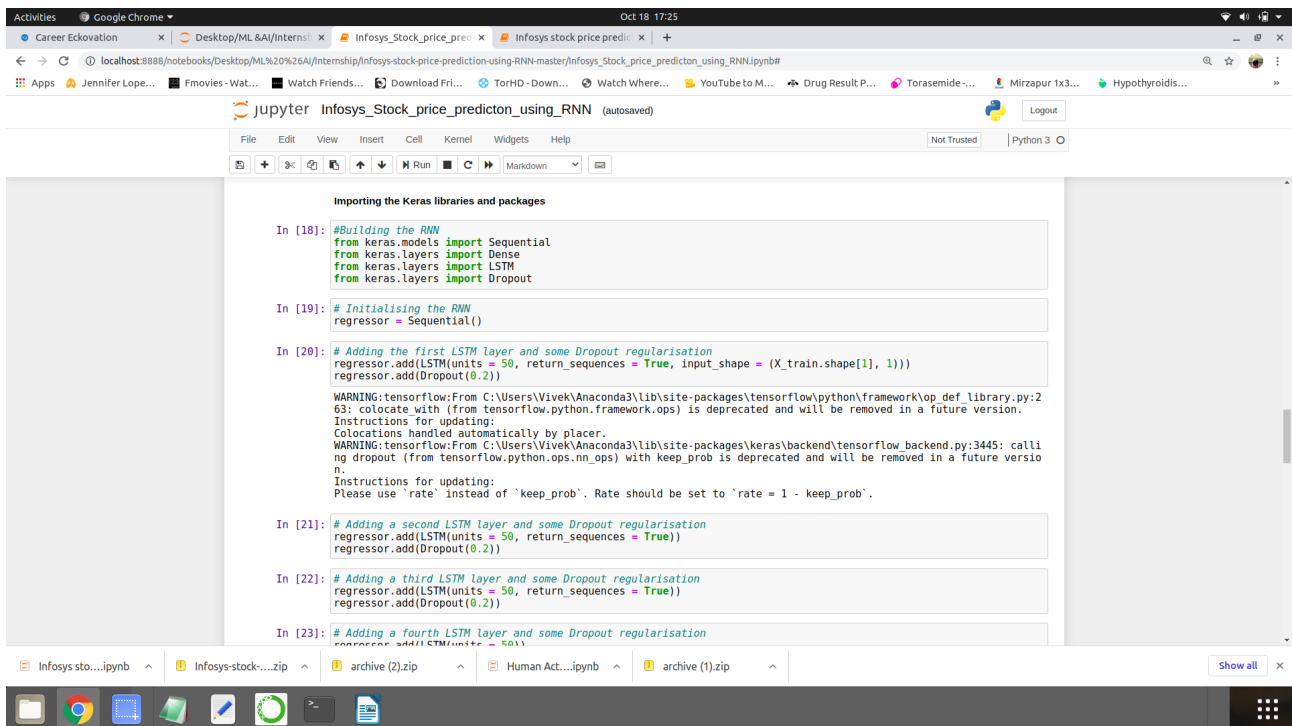
```
In [16]: X_train = []
y_train = []
for i in range(60, 2173):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

```
In [17]: # Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape

Out[17]: (2113, 60, 1)
```

**Importing the Keras libraries and packages**

### 3. Building and initialising RNN



The screenshot shows a Jupyter Notebook titled 'Infosys\_Stock\_price\_predicton\_using\_RNN'. The code is as follows:

```
In [18]: #Building the RNN
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

In [19]: # Initialising the RNN
regressor = Sequential()

In [20]: # Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

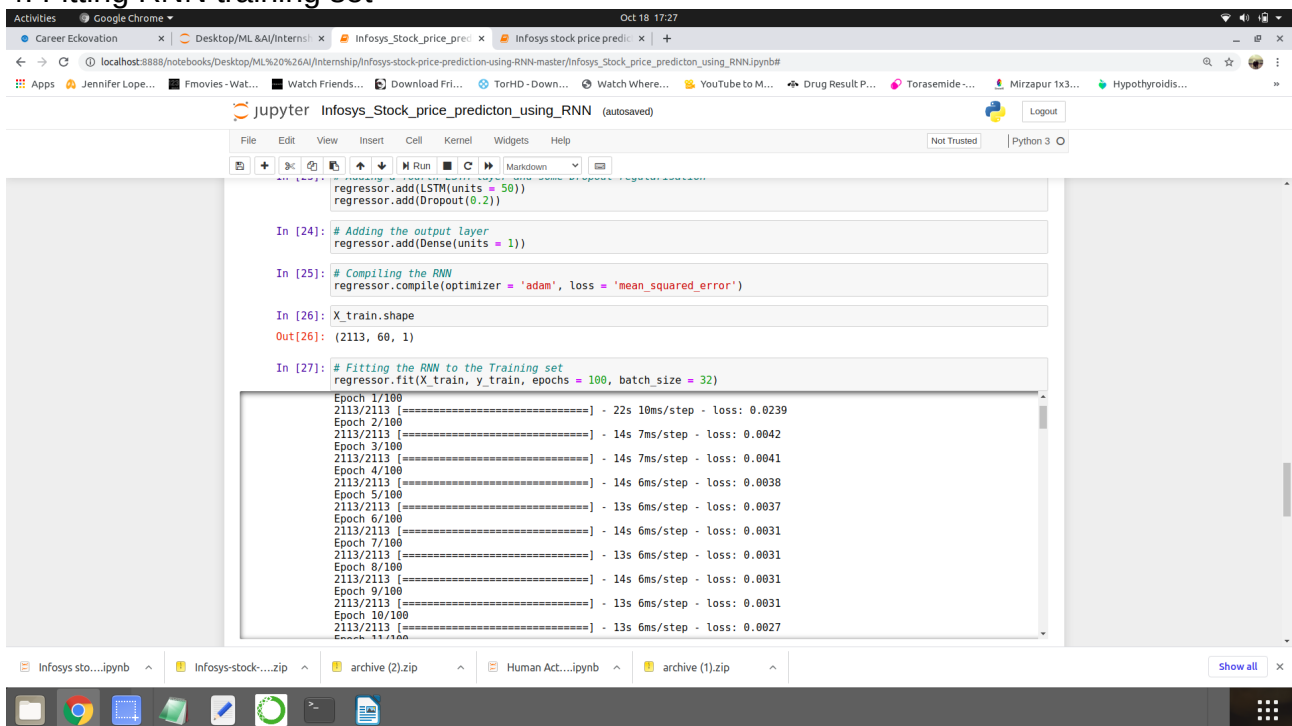
WARNING:tensorflow:From C:\Users\Vivek\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\Vivek\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.

In [21]: # Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

In [22]: # Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

In [23]: # Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
```

### 4. Fitting RNN training set



The screenshot shows the continuation of the Jupyter Notebook. The code is as follows:

```
In [24]: # Adding the output layer
regressor.add(Dense(units = 1))

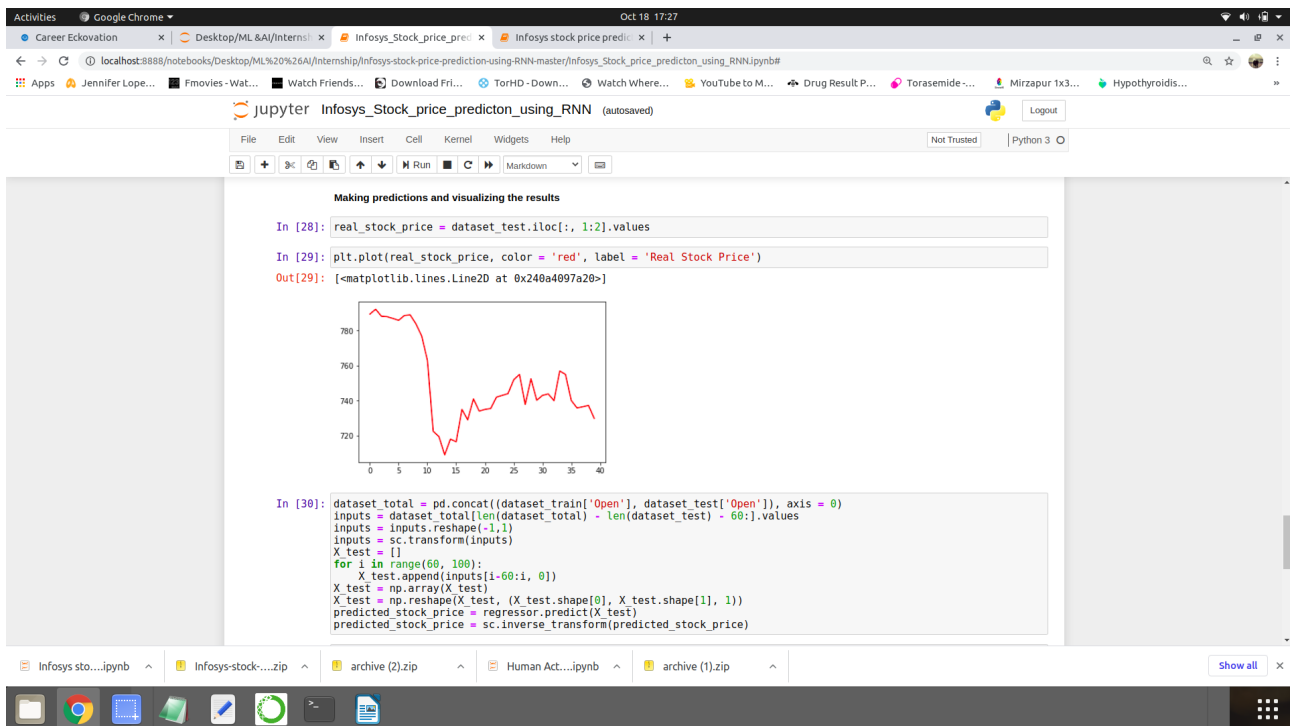
In [25]: # Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

In [26]: X_train.shape
Out[26]: (2113, 60, 1)

In [27]: # Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)

Epoch 1/100
2113/2113 [=====] - 22s 10ms/step - loss: 0.0239
Epoch 2/100
2113/2113 [=====] - 14s 7ms/step - loss: 0.0042
Epoch 3/100
2113/2113 [=====] - 14s 7ms/step - loss: 0.0041
Epoch 4/100
2113/2113 [=====] - 14s 6ms/step - loss: 0.0038
Epoch 5/100
2113/2113 [=====] - 13s 6ms/step - loss: 0.0037
Epoch 6/100
2113/2113 [=====] - 14s 6ms/step - loss: 0.0031
Epoch 7/100
2113/2113 [=====] - 13s 6ms/step - loss: 0.0031
Epoch 8/100
2113/2113 [=====] - 14s 6ms/step - loss: 0.0031
Epoch 9/100
2113/2113 [=====] - 13s 6ms/step - loss: 0.0031
Epoch 10/100
2113/2113 [=====] - 13s 6ms/step - loss: 0.0027
```

## 5. Making predictions and visualizing the results:



## 6. Final output:

