Final Project
Deep Learning and Reinforcement Learning – Google Stock Market

- IBM Machine Learning Profession Certificate
- By Sadhana Jarag

Content

Data Description Section

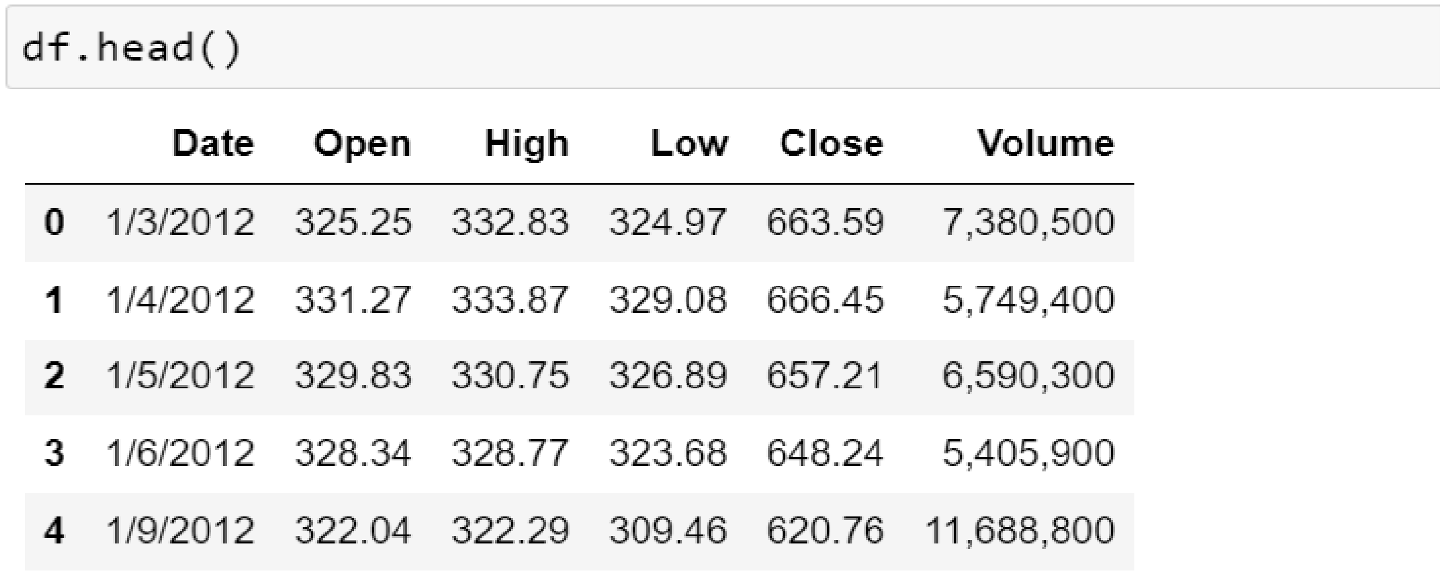
Introduction :

- The term stock market refers to several exchanges in which shares of publicly held companies are bought and sold.
- The role of the stock market is to provide a way for companies to raise capital by selling ownership shares to public investors. At the same time, the stock market allows private investors to buy shares of stock in public companies and become part owners of their businesses.
- **Why You Need to Know About the Stock Market** : The stock market can be extremely volatile on a year-to-year or month-to-month basis, but its long-term performance has been remarkably strong and consistent over time. Not only are these stock market returns remarkably consistent, but they are also higher than the historical returns of other asset classes, such as gold, U.S. Treasury bonds and real estate.

Project Introduction:

- This project we are having the the stock information of year 2012 to 2016.
- In this project we are analyzing the data and try to predict about the future dataset which will help the investors, researchers before investing into the stock market.
- The art of forecasting stock prices has been a difficult task for many of the researchers and analysts
- In fact, investors are highly interested in the research area of stock price prediction. For a good and successful investment, many investors are keen on knowing the future situation of the stock market.
- Good and effective prediction systems for the stock market help traders, investors, and analyst by providing supportive information like the future direction of the stock market.
- In this work, we present a recurrent neural network (RNN) and Long Short-Term Memory (LSTM) approach to predict stock market indices.

Dataset Description :Part 1



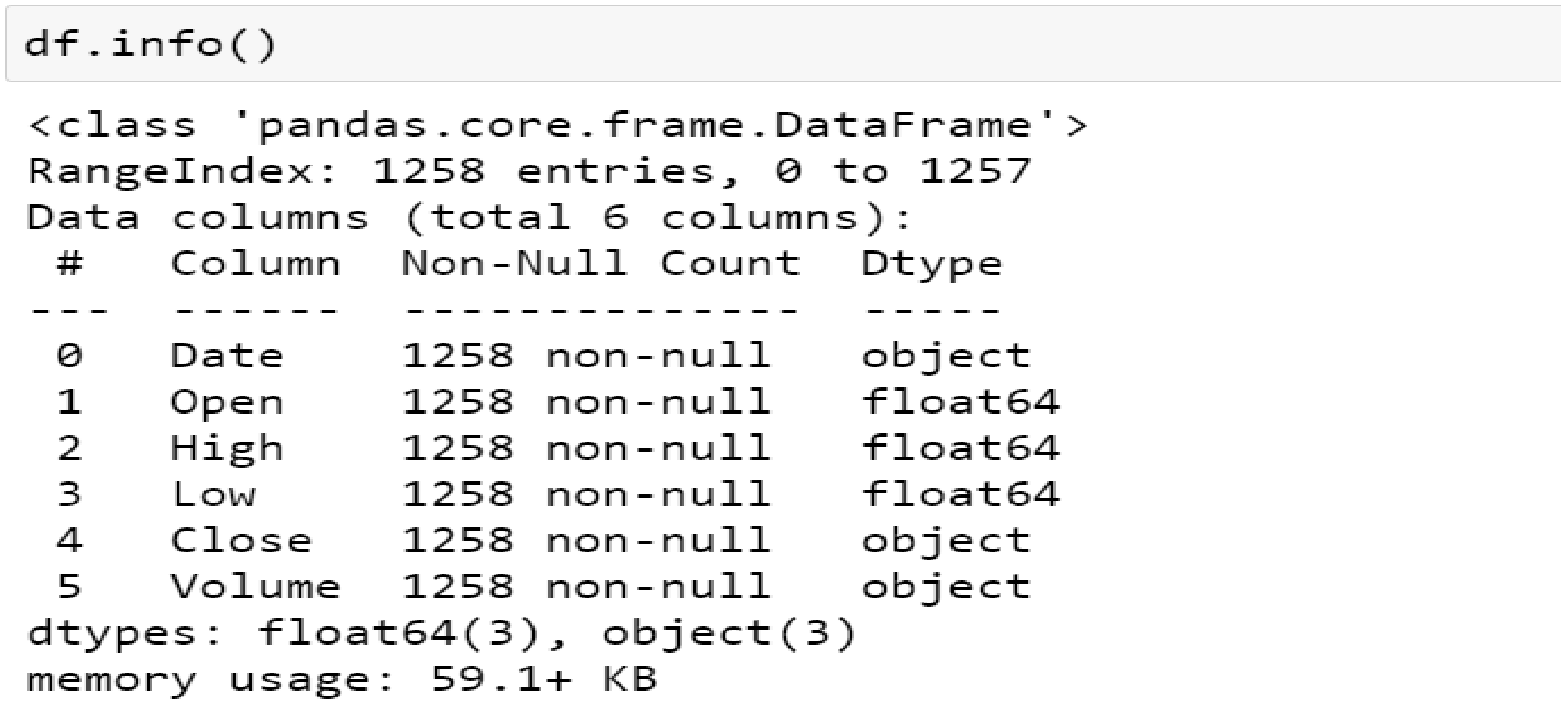
```
df.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

Dataset Description :Part 2

- Attribute Information:
- 1. Date : Date of the Stock market observation
- 2. Open : Open value of stock market on the given date.
- 3. High : High value of the stock market on the given date
- 4. Low :Low value of google stock market on that given date
- 5. Close : Closing value of google stock market on the given date
- 6. Volume : Volume of the Google stock market on the given date .

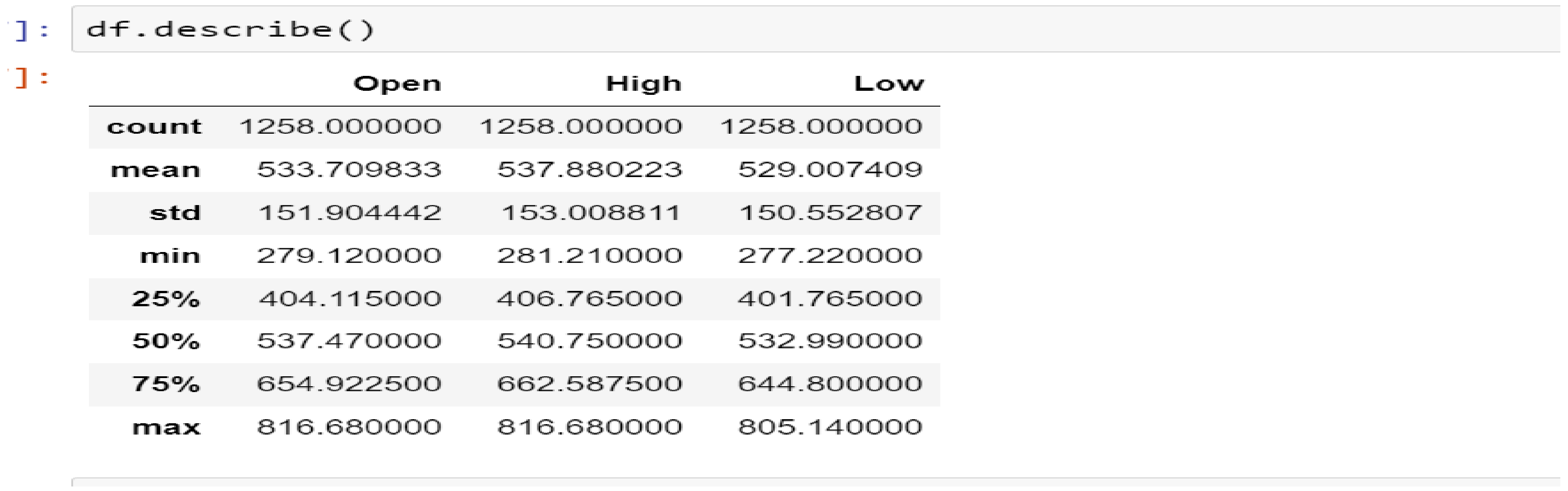
Dataset Description



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    1258 non-null   object 
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object 
 5   Volume  1258 non-null   object 
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

Descriptive Statistics:



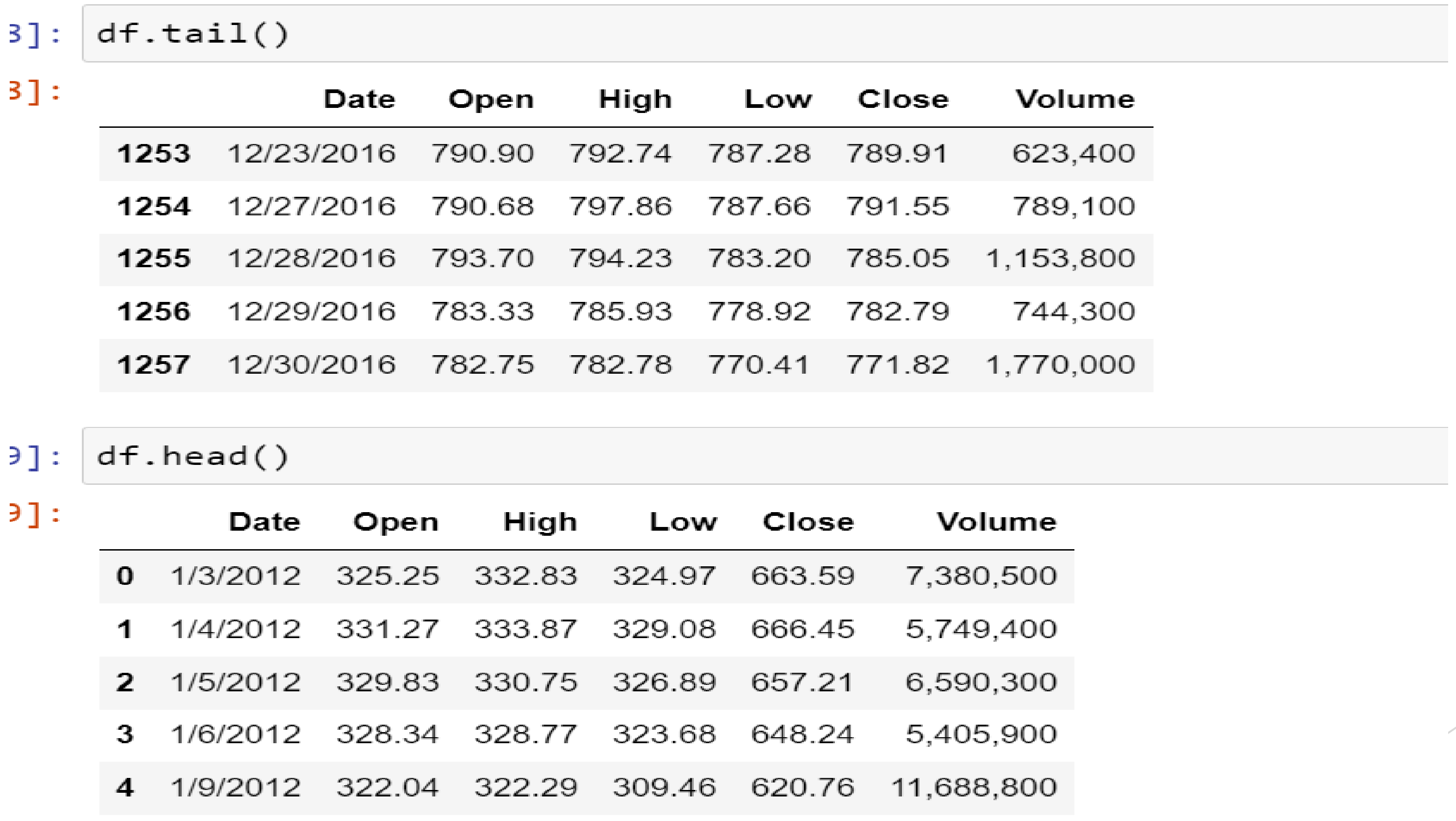
```
]: df.describe()
]:
```

	Open	High	Low
count	1258.000000	1258.000000	1258.000000
mean	533.709833	537.880223	529.007409
std	151.904442	153.008811	150.552807
min	279.120000	281.210000	277.220000
25%	404.115000	406.765000	401.765000
50%	537.470000	540.750000	532.990000
75%	654.922500	662.587500	644.800000
max	816.680000	816.680000	805.140000

Insights :

- Minimum opening of the stock market is 279.1200 and max is 816.68
- Minimum value of the low stock of google is 277.22 and max 805.14
- High value of google stock is 816.68 and min is 281.21

First and last Rows in the Dataset



```
3]: df.tail()
```

3]:

	Date	Open	High	Low	Close	Volume
1253	12/23/2016	790.90	792.74	787.28	789.91	623,400
1254	12/27/2016	790.68	797.86	787.66	791.55	789,100
1255	12/28/2016	793.70	794.23	783.20	785.05	1,153,800
1256	12/29/2016	783.33	785.93	778.92	782.79	744,300
1257	12/30/2016	782.75	782.78	770.41	771.82	1,770,000

```
9]: df.head()
```

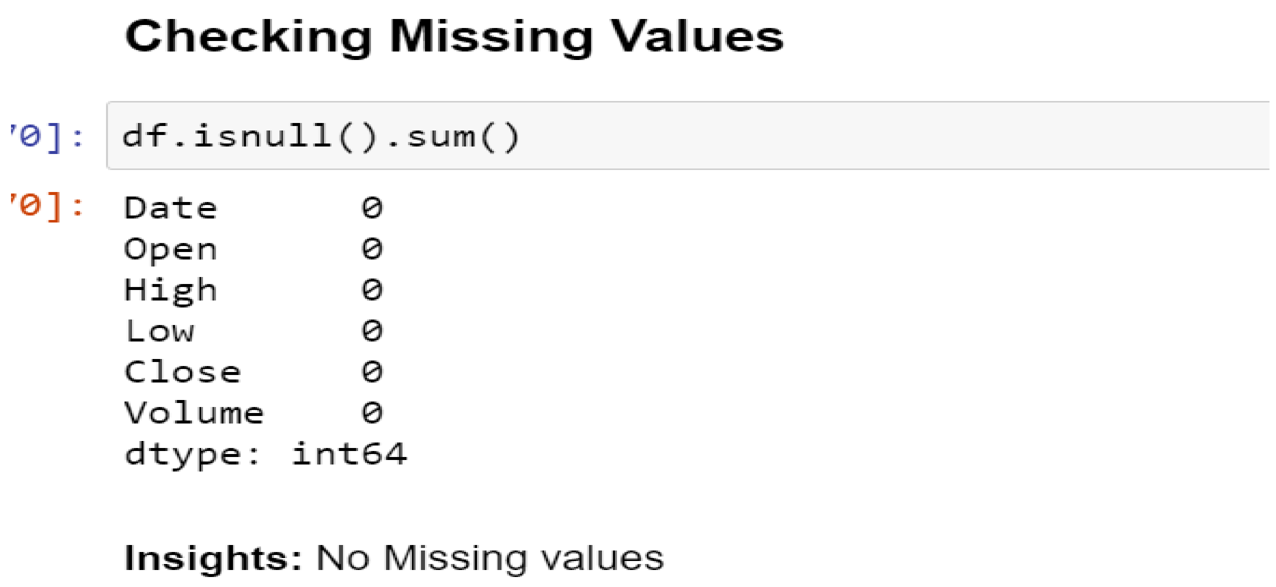
9]:

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

Main Objective of Analysis

- In this section we will go through with some insights of the given data.
- We will try to find out some patterns out of the data or some of good observations wrt to the given datasets.

Checking for the missing values



Checking Missing Values

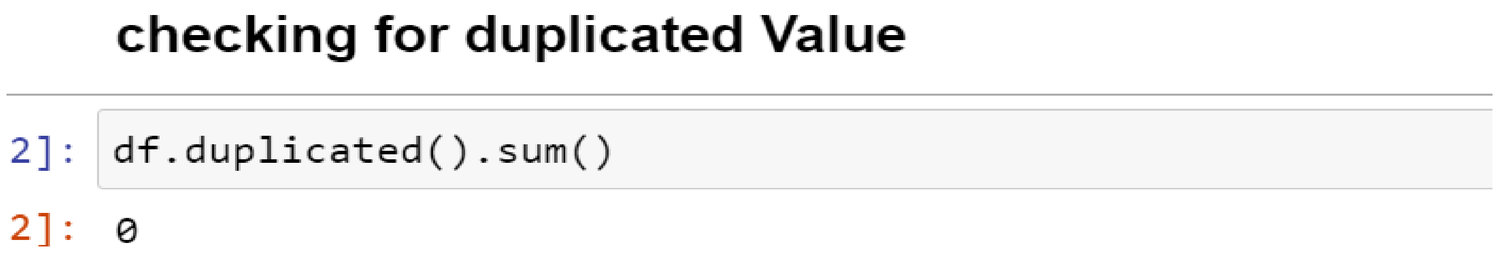
```
'0]: df.isnull().sum()
```

```
'0]: Date      0
     Open      0
     High      0
     Low       0
     Close     0
     Volume    0
     dtype: int64
```

Insights: No Missing values

Checking for duplicated values

Check for duplicate Value: No duplicate Value found



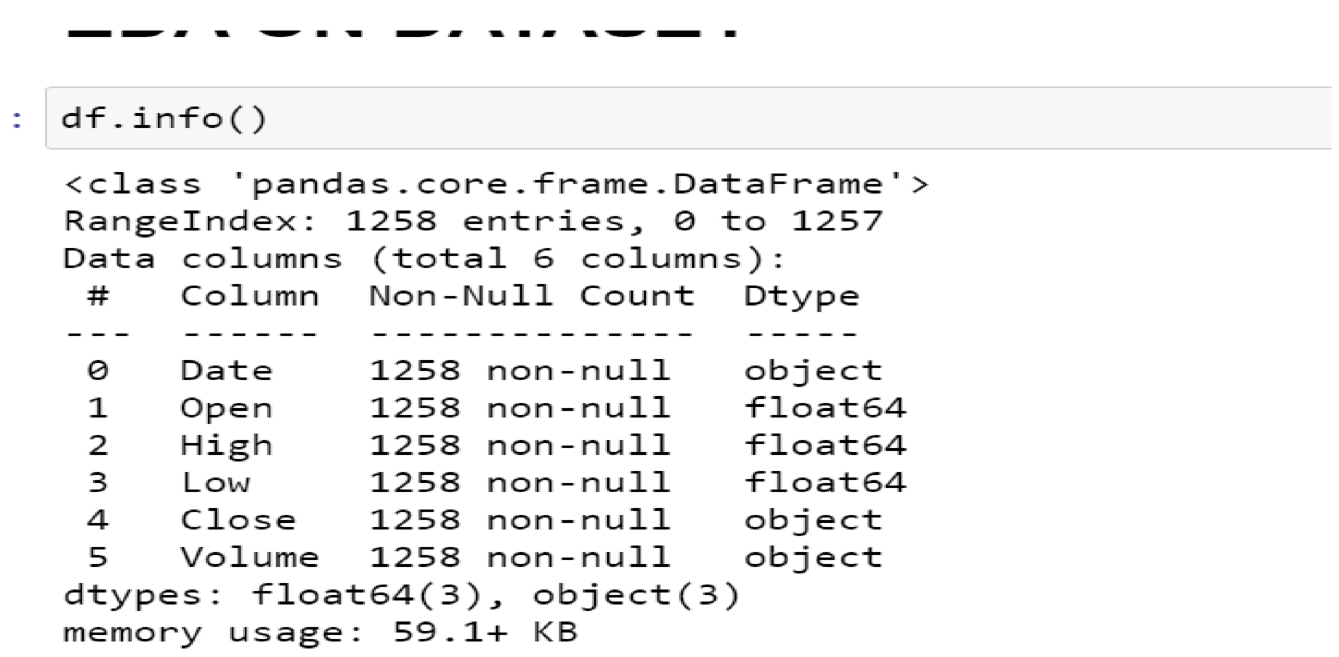
checking for duplicated Value

```
2]: df.duplicated().sum()

2]: 0
```

EDA :Exploratory Data Analysis

EDA :Exploratory Data Analysis

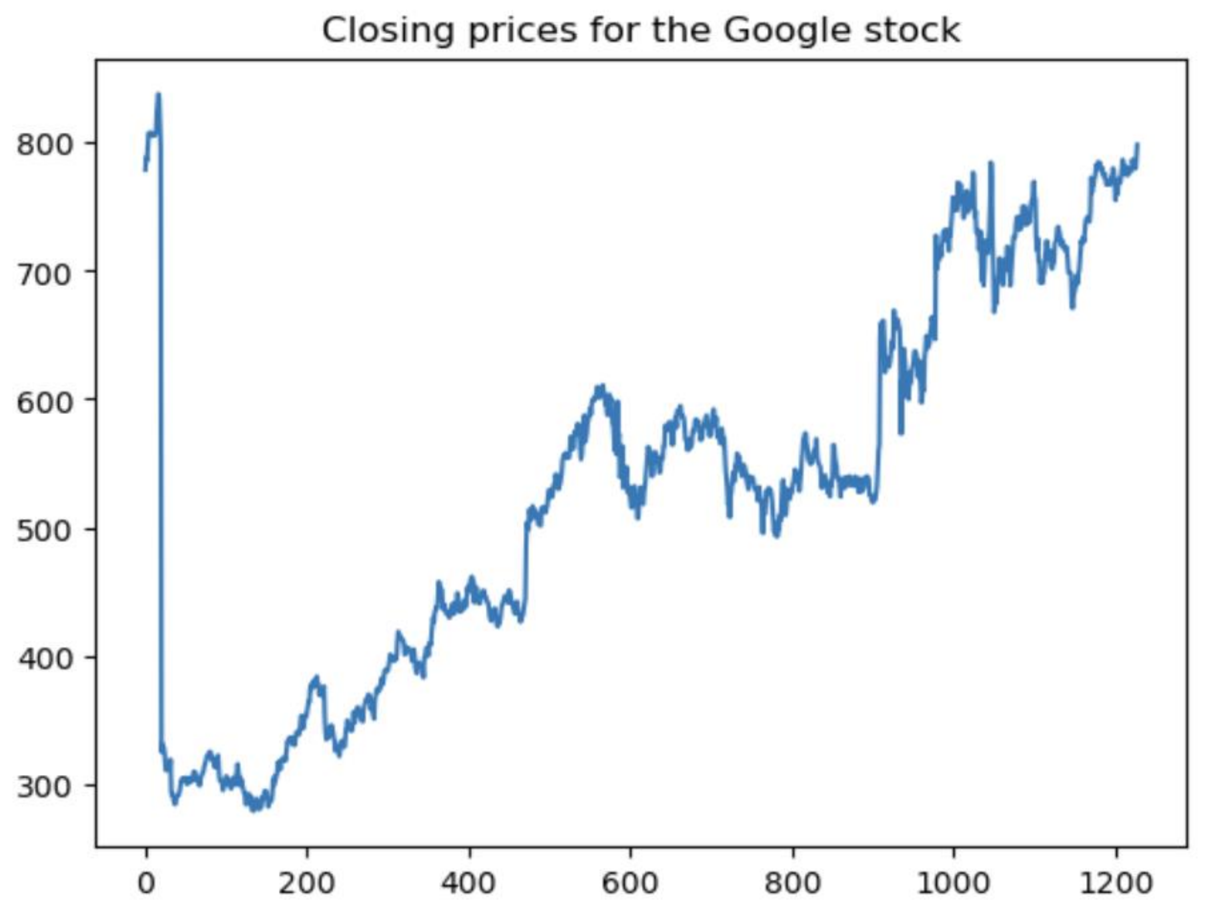


```
: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1258 non-null   object
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low     1258 non-null   float64
 4   Close   1258 non-null   object
 5   Volume  1258 non-null   object
dtypes: float64(3), object(3)
memory usage: 59.1+ KB
```

- Some of the values are string datatype and some are with float.

EDA :Exploratory Data Analysis

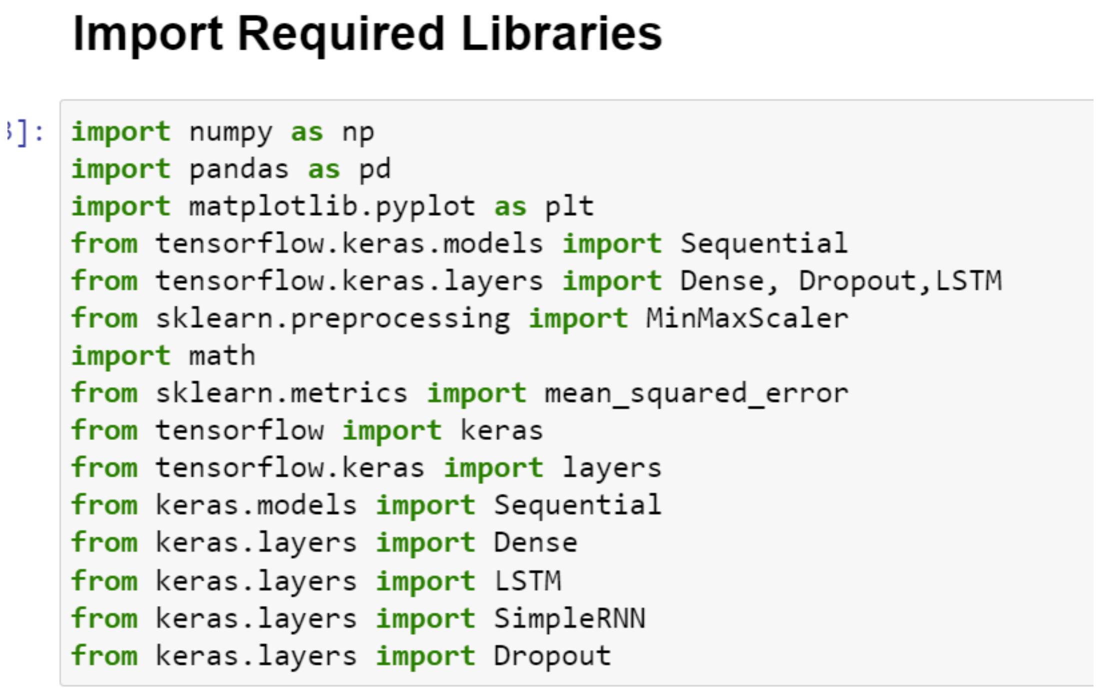


- Insights
- We can see that closing value of google stock is fluctuated over the time

RNN Model

- A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes.
- This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state to process variable length sequences of inputs.
- This makes them applicable to tasks such as unsegmented, connected handwriting recognition[4] or speech recognition.
- Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.

Import Required Libraries



Import Required Libraries

```
]: import numpy as np
   import pandas as pd
   import matplotlib.pyplot as plt
   from tensorflow.keras.models import Sequential
   from tensorflow.keras.layers import Dense, Dropout,LSTM
   from sklearn.preprocessing import MinMaxScaler
   import math
   from sklearn.metrics import mean_squared_error
   from tensorflow import keras
   from tensorflow.keras import layers
   from keras.models import Sequential
   from keras.layers import Dense
   from keras.layers import LSTM
   from keras.layers import SimpleRNN
   from keras.layers import Dropout
```

How to apply RNN Model

- Here I have used 4 hidden layers with each having 50 modes and activation function as ‘tanh’ and SimpleRNN model used for predicting the data.
- For regularization used the dropout technique to avoid the overfitting.
- Epoch are 100 and used batch size is 32

Initialising the RNN

```
regressor = Sequential()
```

Adding the first RNN layer and Dropout for regularization

```
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))
```

Adding the second RNN layer and Dropout for regularization.

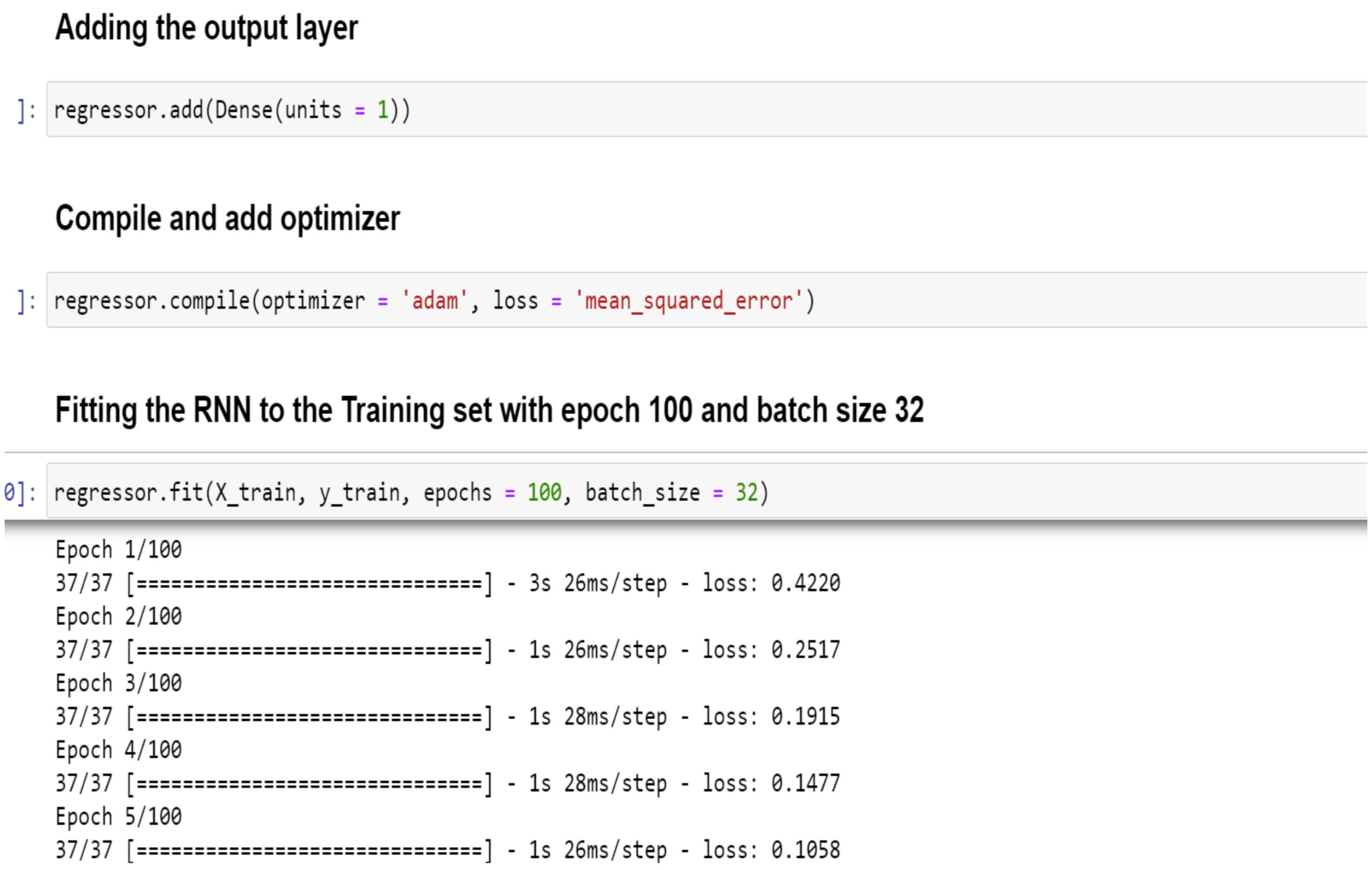
```
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))
```

Adding the third RNN layer and Dropout for regularization.

```
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))
```

Adding the Fourth RNN layer and Dropout for regularization.

```
regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))
```



Adding the output layer

```
]: regressor.add(Dense(units = 1))
```

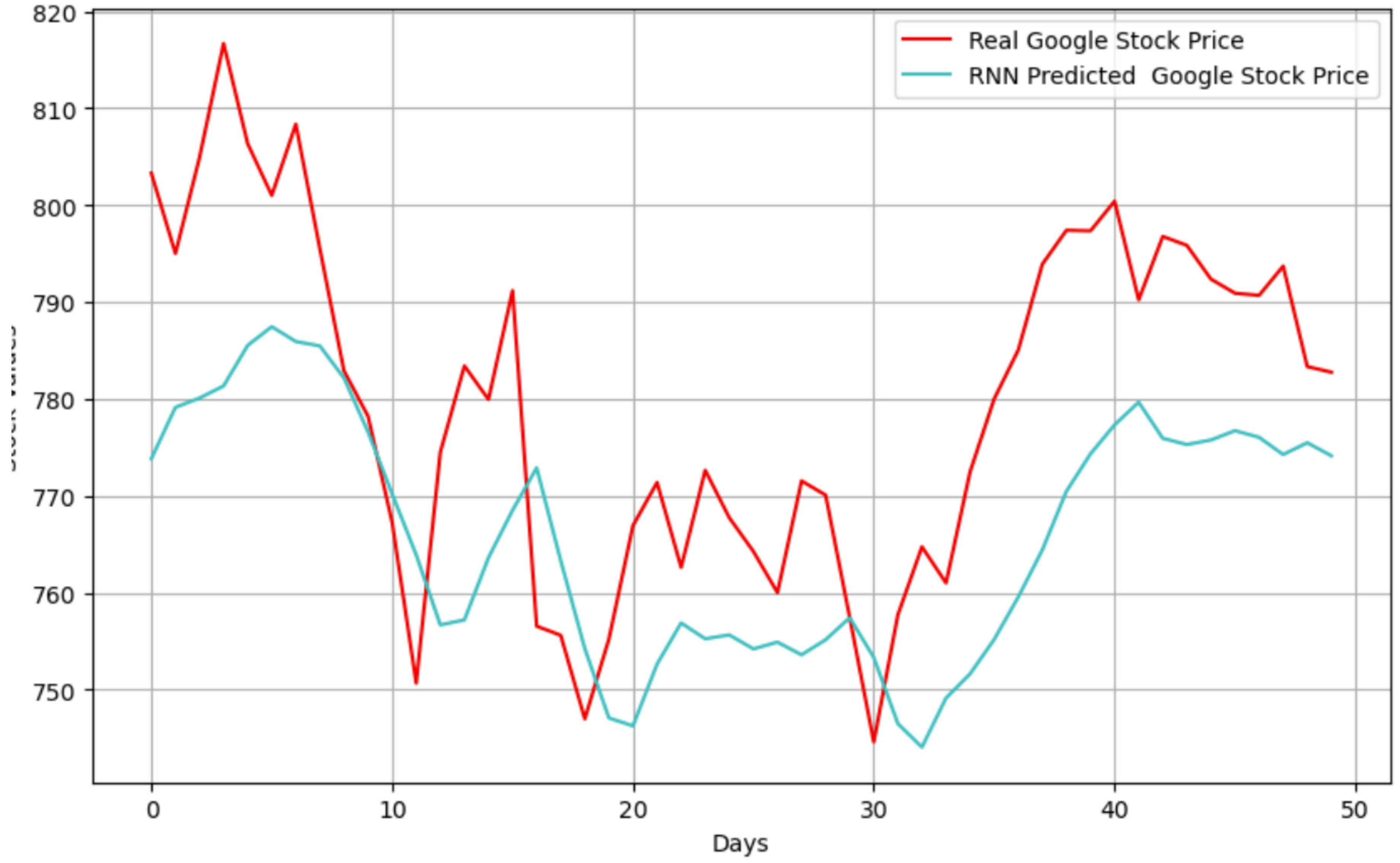
Compile and add optimizer

```
]: regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

Fitting the RNN to the Training set with epoch 100 and batch size 32

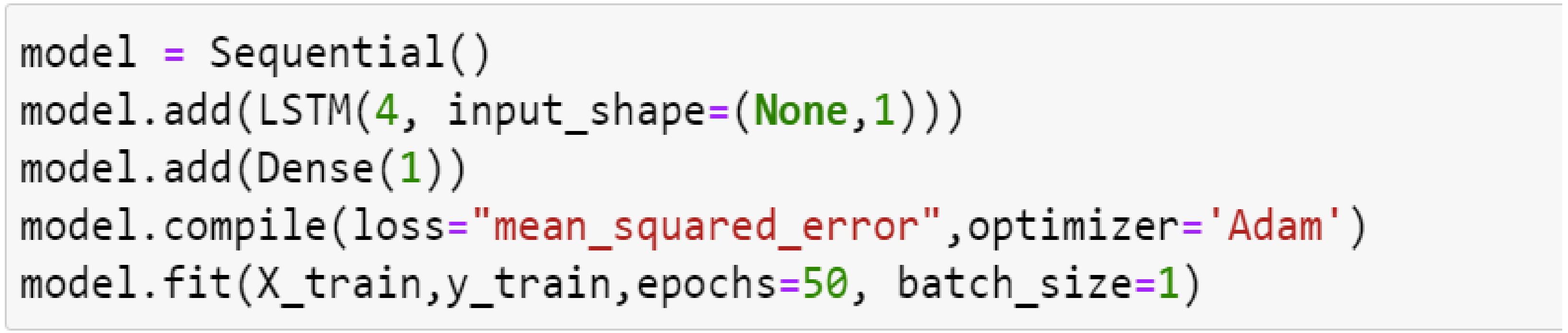
```
0]: regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

```
Epoch 1/100
37/37 [==============================] - 3s 26ms/step - loss: 0.4220
Epoch 2/100
37/37 [==============================] - 1s 26ms/step - loss: 0.2517
Epoch 3/100
37/37 [==============================] - 1s 28ms/step - loss: 0.1915
Epoch 4/100
37/37 [==============================] - 1s 28ms/step - loss: 0.1477
Epoch 5/100
37/37 [==============================] - 1s 26ms/step - loss: 0.1058
```


Real Google Stock Price
RNN Predicted Google Stock Price
820
810
800
790
780
770
760
750
0
10
20
30
40
50
Days
Stock Values

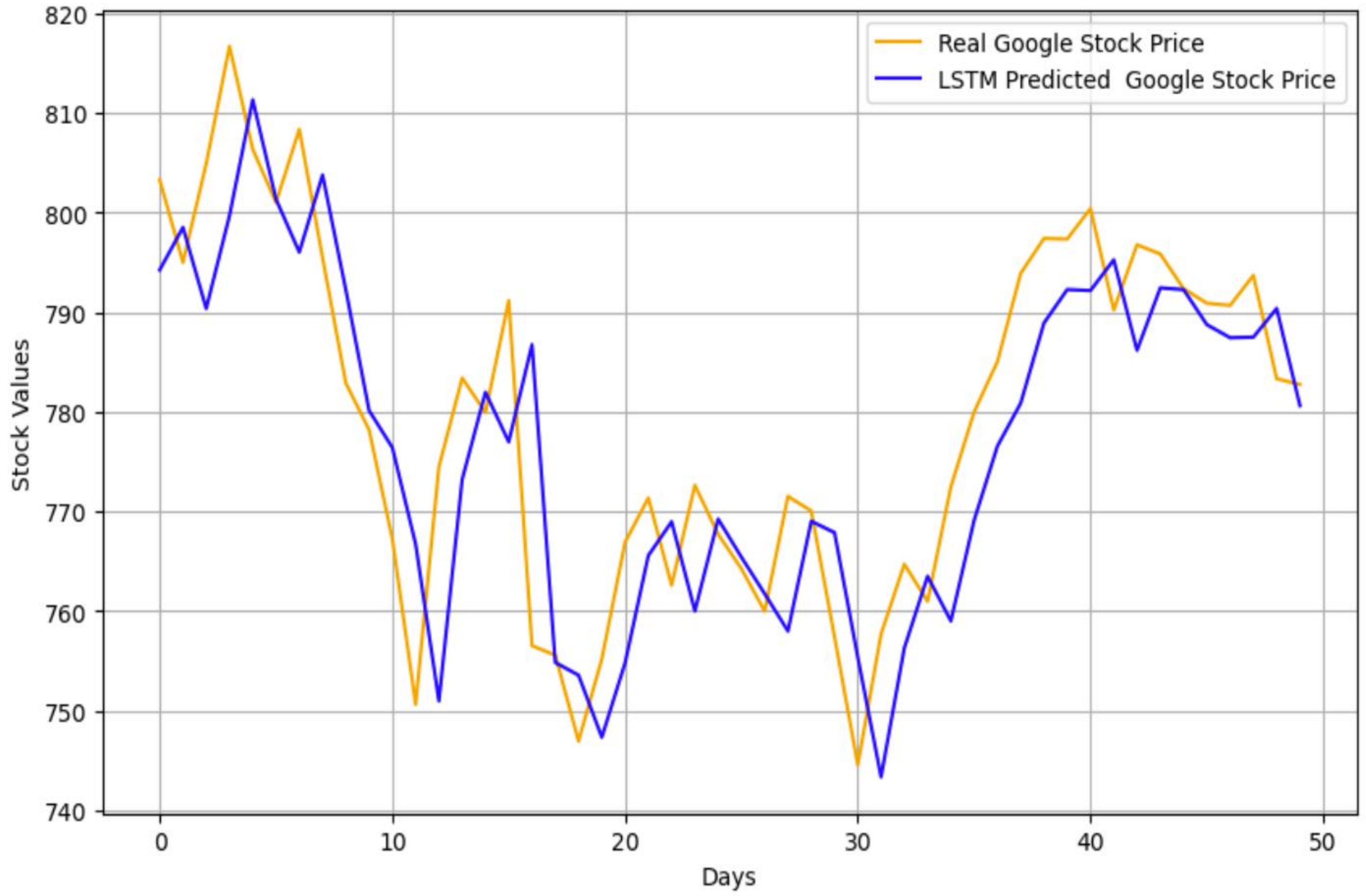
How to apply LSTM

LSTM MODULE



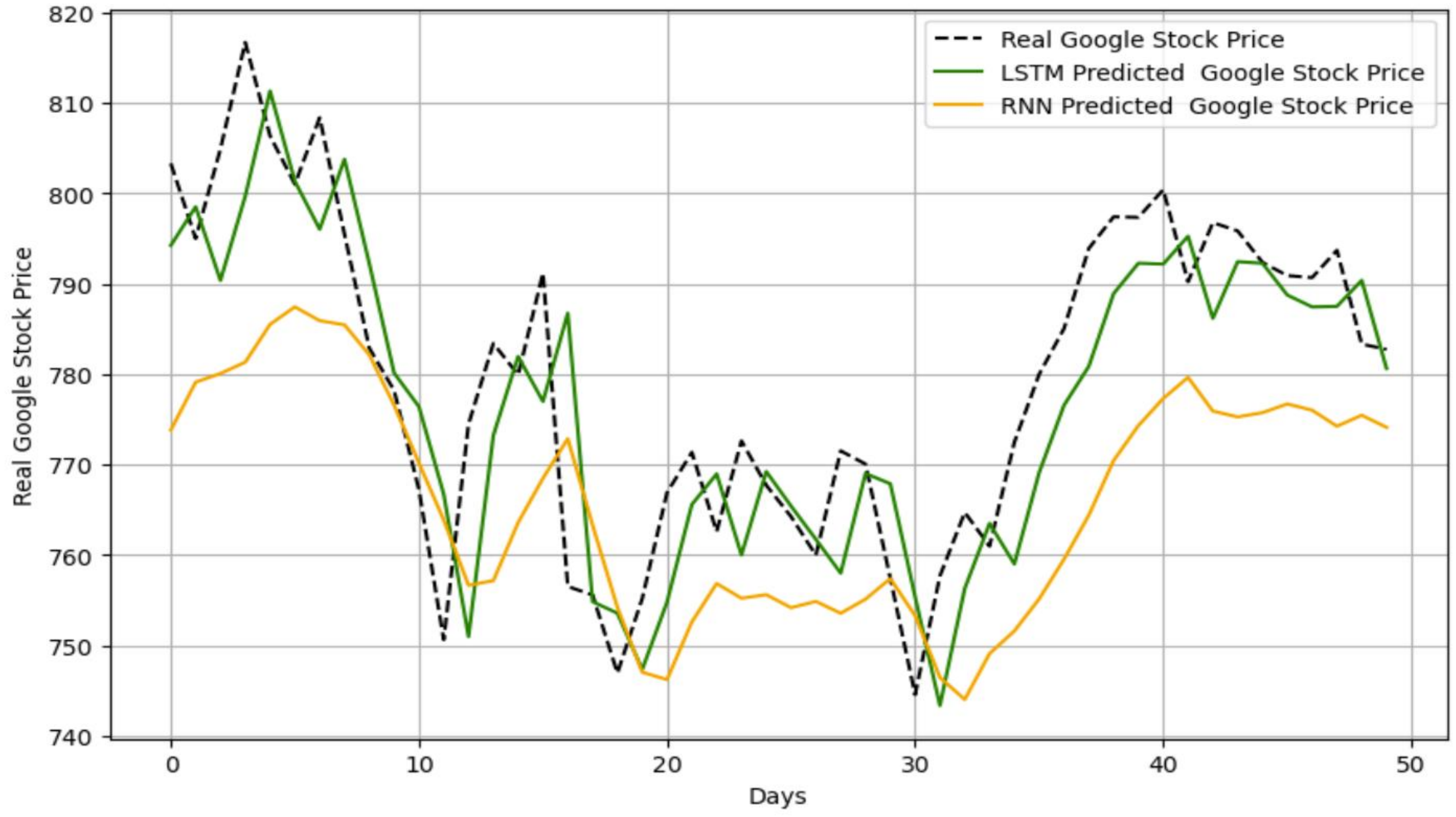
```
model = Sequential()
model.add(LSTM(4, input_shape=(None,1)))
model.add(Dense(1))
model.compile(loss="mean_squared_error",optimizer='Adam')
model.fit(X_train,y_train,epochs=50, batch_size=1)
```

```
plt.show()
```



820
810
800
790
780
770
760
750
740
Stock Values
0
10
20
30
40
50
Days
Real Google Stock Price
LSTM Predicted Google Stock Price

Comparing RNN vs LSTM



Here we can see LSTM is giving better result

Model Comparison, Model Flaws ,Advanced Step and further Suggestion

- <u>**Here is the analysis**</u>
- Here we can see that LSTM is performing well on the data.
- We can use that as model for prediction.
- Model flaws we can say that if data is changes then we have to reanalyzed and rebuild the model with the help of hyperparameter tunning.
- Advanced steps: we can go ahead and use other models and check the performance and decide the model which one is the best one .
- We can also do hyperparameter tunning for RNN and can use optimization technique to improve the model performance.
- Suggestion: we must check with respect to more data, model performance and according to that we have retrain the model again.

Thank You!!!!

Deep Learning and Reinforcement Learning

By Sadhana Jarag