```
Stock Market Prediction Using Machine Learning
```

Dataset Link = https://www.kaggle.com/datasets/vaibhavsxn/google-stock-prices-training-and-test-data?select=Google_Stock_Price_Train.csv (https://www.kaggle.com/datasets/vaibhavsxn/google-stock-prices-training-and-test-data?select=Google_Stock_Price_Train.csv)

In [ ]:

```
```

In [1]:

```
# For Mathimatical Calculation
import numpy as np
# For Visualization
import matplotlib.pyplot as plt
# Data Analysis
import pandas as pd
# To Work With Dates
import datetime
```

In [2]:

```
#to read data set
dataset=pd.read_csv(r"C:\Users\HOME\Downloads\Stock_Price_Train.csv")
print(dataset)
```

```
            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
...          ...     ...     ...     ...     ...         ...
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

[1258 rows x 6 columns]
```

In [3]:

```
dataset.head(10)
```

Out[3]:

	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
5	1/10/2012	313.70	315.72	307.30	621.43	8,824,000
6	1/11/2012	310.59	313.52	309.40	624.25	4,817,800
7	1/12/2012	314.43	315.26	312.08	627.92	3,764,400
8	1/13/2012	311.96	312.30	309.37	623.28	4,631,800
9	1/17/2012	314.81	314.81	311.67	626.86	3,832,800

In [4]:

```
dataset.isna().any()
```

Out[4]:

```
Date      False
Open      False
High      False
Low       False
Close     False
Volume    False
dtype: bool
```

In [5]:

```
dataset.info()
```

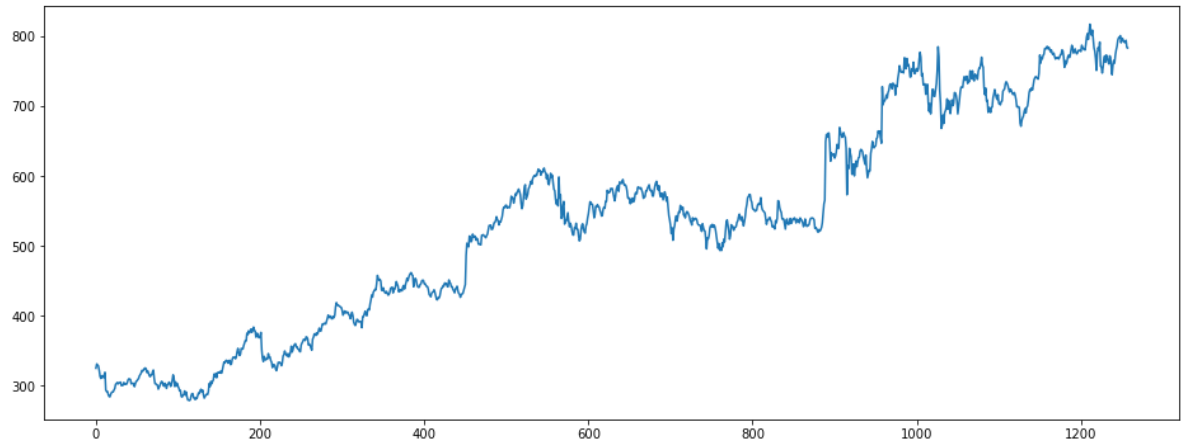
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    1258 non-null   object 
 1   Open    1258 non-null   float64
 2   High    1258 non-null   float64
 3   Low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

In [6]:

```
dataset['Open'].plot(figsize=(16,6))
```

Out[6]:

<AxesSubplot:>



In [7]:

```
#convert column 'a' of a dataframe
dataset['Close']=dataset['Close'].str.replace(',','').astype(float)
```

In [8]:

```
dataset["Volume"]=dataset["Volume"].str.replace(',','').astype(float)
```

In [9]:

```
#7day rolling mean
dataset.rolling (7).mean().head(20)
```

Out[9]:

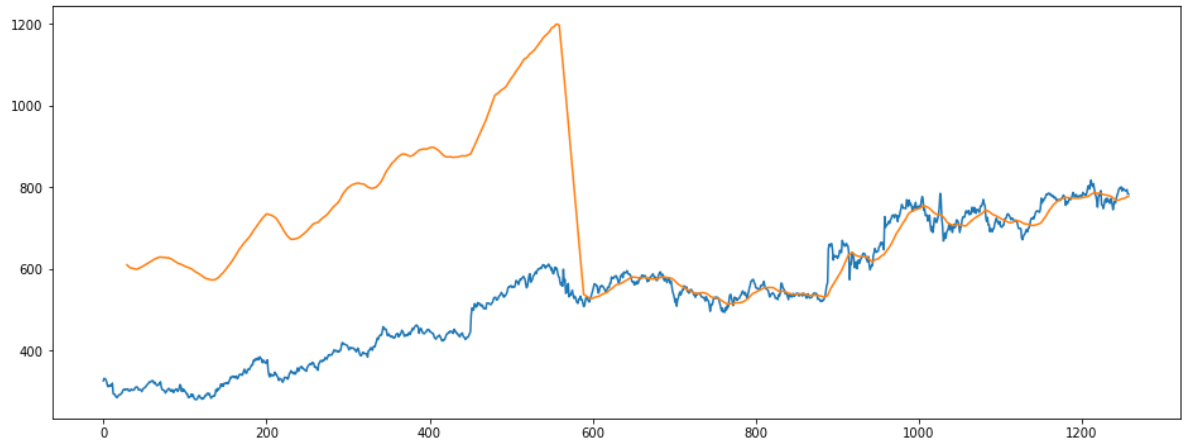
	Open	High	Low	Close	Volume
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN
6	323.002857	325.392857	318.682857	643.132857	7.208100e+06
7	321.457143	322.882857	316.841429	638.037143	6.691514e+06
8	318.698571	319.801429	314.025714	631.870000	6.531857e+06
9	316.552857	317.524286	311.851429	627.534286	6.137929e+06
10	314.238571	315.674286	309.882857	625.097143	6.157657e+06
11	313.847143	315.247143	310.610000	627.534286	6.296086e+06
12	311.055714	312.201429	308.104286	622.242857	8.068629e+06
13	308.387143	309.302857	305.402857	616.481429	8.359129e+06
14	305.192857	306.085714	301.951429	609.541429	8.697700e+06
15	301.724286	302.652857	298.060000	601.634286	9.466400e+06
16	297.454286	298.561429	293.710000	593.017143	9.844071e+06
17	293.480000	294.741429	289.952857	585.475714	1.008950e+07
18	289.001429	290.401429	285.821429	576.660000	8.949586e+06
19	288.465714	289.902857	285.355714	575.821429	6.530857e+06

In [10]:

```
dataset['Open'].plot(figsize=(16,6))
dataset.rolling(window=30).mean()['Close'].plot()
```

Out[10]:

<AxesSubplot:>



In [11]:

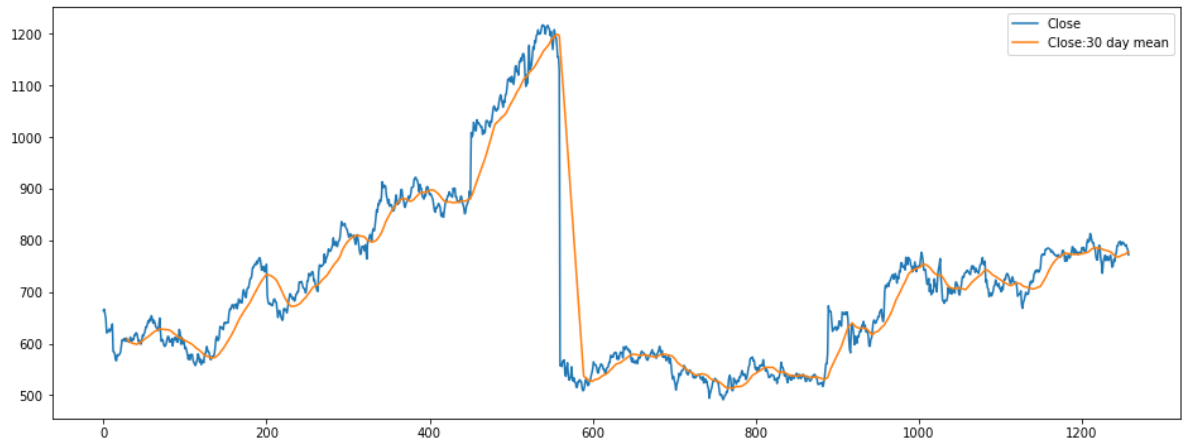
```
dataset['Close:30 day mean']=dataset['Close'].rolling(window=30).mean()
```

In [12]:

```
dataset[['Close','Close:30 day mean']].plot(figsize=(16,6))
```

Out[12]:

<AxesSubplot:>

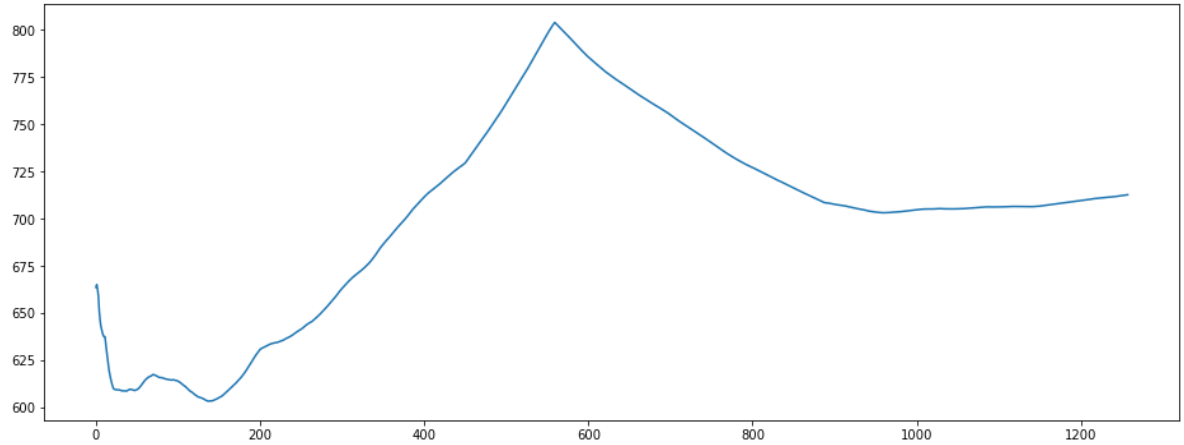


In [13]:

```
# optimal specify a minimum number of periods
dataset['Close'].expanding(min_periods=1).mean().plot(figsize=(16,6))
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
#creating a dataframe
training_set=dataset['Open']
training_set=pd.DataFrame(training_set)
```

In [15]:

```
# data cleaning
dataset.isna().any()
```

Out[15]:

```
Date                 False
Open                 False
High                 False
Low                  False
Close                False
Volume               False
Close:30 day mean     True
dtype: bool
```

In [16]:

```
#feature scaling
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
```

In [17]:

```
#creating a data structure with 60 timesteps and 1 output
x_train=[]
y_train=[]
for i in range(60,1258):
    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)

#reshaping
x_train=np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

In [18]:

```
# Building the RNN

#importing the Keras libraries and packages
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))

#Adding a second LSTM Layer and some Dropout regularisation
regressor.add(LSTM(units=50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM Layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM Layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

#Adding the output Layer

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

In [21]:

```
# compiling the RNN
regressor.compile(optimizer = 'adam',loss ='mean_squared_error')

#fitting the RNN to the training set
regressor.fit(x_train,y_train,epochs=100,batch_size=32)
```

```
Epoch 93/100
38/38 [==============================] - 2s 63ms/step - loss: 0.0015
Epoch 94/100
38/38 [==============================] - 2s 59ms/step - loss: 0.0013
Epoch 95/100
38/38 [==============================] - 2s 61ms/step - loss: 0.0015
Epoch 96/100
38/38 [==============================] - 2s 62ms/step - loss: 0.0015
Epoch 97/100
38/38 [==============================] - 2s 63ms/step - loss: 0.0014
Epoch 98/100
38/38 [==============================] - 2s 61ms/step - loss: 0.0014
Epoch 99/100
38/38 [==============================] - 2s 60ms/step - loss: 0.0014
Epoch 100/100
38/38 [==============================] - 2s 61ms/step - loss: 0.0016
```

Out[21]:

```
<keras.callbacks.History at 0x15f23db1d00>
```

In [22]:

```
# part 3 - making the predictions and visualising the results

# getting the real stock price of 2017
dataset_test=pd.read_csv(r"C:\Users\HOME\Downloads\Stock_Price_Train.csv",index_col="Date",
```

In [23]:

```
real_stock_price=dataset_test.iloc[:,1:2].values
```

In [24]:

```
dataset_test.head()
```

Out[24]:

Date	Open	High	Low	Close	Volume
2012-01-03	325.25	332.83	324.97	663.59	7,380,500
2012-01-04	331.27	333.87	329.08	666.45	5,749,400
2012-01-05	329.83	330.75	326.89	657.21	6,590,300
2012-01-06	328.34	328.77	323.68	648.24	5,405,900
2012-01-09	322.04	322.29	309.46	620.76	11,688,800

In [25]:

```
dataset_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1258 entries, 2012-01-03 to 2016-12-30
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Open    1258 non-null   float64
 1   High    1258 non-null   float64
 2   Low     1258 non-null   float64
 3   Close   1258 non-null   object 
 4   Volume  1258 non-null   object 
dtypes: float64(3), object(2)
memory usage: 59.0+ KB
```

In [26]:

```
dataset_test["Volume"]=dataset_test["Volume"].str.replace(',','').astype(float)
```

In [27]:

```
test_set=dataset_test['Open']
test_set=pd.DataFrame(test_set)
```

In [28]:

```
test_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1258 entries, 2012-01-03 to 2016-12-30
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Open    1258 non-null   float64
dtypes: float64(1)
memory usage: 19.7 KB
```

In [29]:

```
# Getting the predicted stock price of 2017
dataset_total=pd.concat((dataset['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(60,80):
    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 = regressor.predict(x_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
C:\Users\HOME\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but MinMaxScaler was fitted with featur
e names
  warnings.warn(

1/1 [==============================] - 1s 1s/step
```

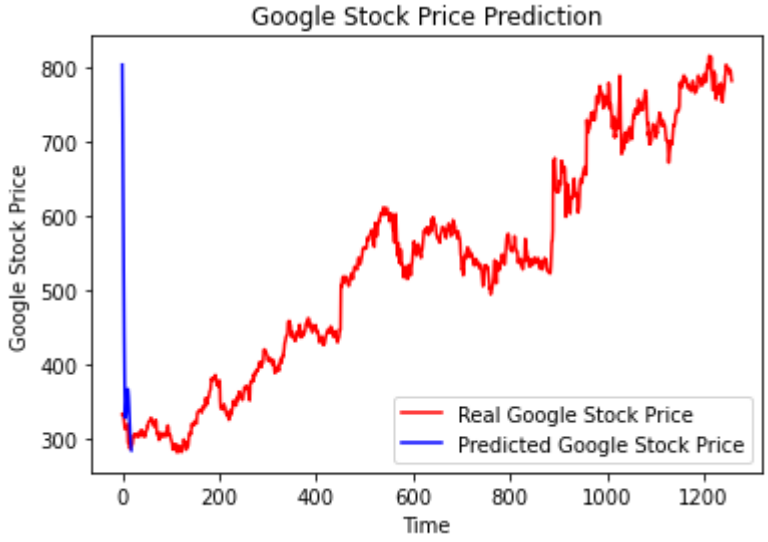
In [30]:

```
predicted_stock_price=pd.DataFrame(predicted_stock_price)
predicted_stock_price.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       20 non-null     float32
dtypes: float32(1)
memory usage: 208.0 bytes
```

In [31]:

```
# Visualising the result
plt.plot(real_stock_price, color = 'red', label = "Real Google Stock Price")
plt.plot(predicted_stock_price,color = "blue",label = "Predicted Google Stock Price")
plt.title("Google Stock Price Prediction")
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]: