Stock Price Prediction

In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

```
df_train = pd.read_csv('stock_price_prediction_train.csv')
```

In [3]:

```
df_train
```

Out[3]:

	Date	Open	High	Low	Close	Volume	Stock Trading
0	2016-12-30	42120	42330	41700	41830	610000	2562802800
1	2016-12-29	43000	43220	42540	42660	448400	1918822700
2	2016-12-28	43940	43970	43270	43270	339900	1478067000
3	2016-12-27	43140	43700	43140	43620	400100	1742799300
4	2016-12-26	43310	43660	43090	43340	358200	1554780300
...	...	...	...	...	...	...	...
1221	2012-01-11	14360	14750	14280	14590	1043400	1519198800
1222	2012-01-10	13890	14390	13860	14390	952300	1353341300
1223	2012-01-06	13990	14030	13790	13850	765500	1063560900

	Date	Open	High	Low	Close	Volume	Stock Trading
							0
1224	2012-01-05	13720	13840	13600	13800	511500	7030811000
1225	2012-01-04	14050	14050	13700	13720	55910	7719804000

1226 rows × 7 columns

In [4]:

```
df_train.tail()
```

Out[4]:

	Date	Open	High	Low	Close	Volume	Stock Trading
1221	2012-01-11	14360	14750	14280	14590	1043400	1519198800
1222	2012-01-10	13890	14390	13860	14390	952300	1353341300
1223	2012-01-06	13990	14030	13790	13850	76550	1063560900
1224	2012-01-05	13720	13840	13600	13800	511500	7030811000
1225	2012-01-04	14050	14050	13700	13720	55910	7719804000

In [5]:

```
df_test = pd.read_csv('stock_price_prediction_test.csv')
```

In [6]:

```
df_test
```

Out[6]:

	Date	Open	High	Low	Close	Volume	Stock Trading
0	2017-01-13	38900	39380	38240	38430	1321200	5119728900
1	2017-01-12	38300	38450	37930	38010	800900	3054035900
2	2017-01-11	38710	38880	38480	38560	545900	2108979800
3	2017-01-10	38620	38850	38150	38690	1196900	4610770300
4	2017-01-06	40500	41030	39720	39720	1435500	5770893400
5	2017-01-05	43250	43330	42470	42590	516800	2212425000
6	2017-01-04	42480	43330	42450	43290	648100	2791672800

In [7]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(df_train[['Open']])
x = scaler.transform(df_train[['Open']])
```

In [8]:

```
import matplotlib.pyplot as plt
```

In [9]:

```
plt.plot(df_train['Date'], df_train['Open'])
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x22929c57910>]
```

In [10]:

```
x
```

Out[10]:

```
array([[0.5937696 ],
       [0.6121681 ],
       [0.63182103],
       ...,
       [0.00564499],
       [0.        ],
       [0.00689944]])
```

In [11]:

```
import numpy as np
```

In [12]:

```
x_temp = []
for i in range (0,1198):
    window = x[i:i+61,0]
    x_temp.append(window)
x_temp = np.array(x_temp)
x_temp.shape
```

```
C:\Users\OM\AppData\Local\Temp/ipykernel_456/977076073.py:5:
VisibleDeprecationWarning: Creating an ndarray from ragged nested
sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays
with different lengths or shapes) is deprecated. If you meant to do
this, you must specify 'dtype=object' when creating the ndarray.
  x_temp = np.array(x_temp)
```

Out[12]:

```
(1198,)
```

In [13]:

```
window
```

Out[13]:

```
array([0.03721514, 0.03721514, 0.04097847, 0.04557809, 0.04327828,
       0.04223291, 0.03930587, 0.03616977, 0.03596069, 0.02843404,
       0.02717959, 0.02822496, 0.02174368, 0.02467071, 0.02885218,
       0.02195275, 0.01902572, 0.02655237, 0.0263433 , 0.02216182,
       0.01923479, 0.01860757, 0.02132553, 0.0167259 , 0.01338072,
       0.00355425, 0.00564499, 0.        , 0.00689944])
```

In [14]:

```
x_temp
```

Out[14]:

```
array([array([0.5937696 , 0.6121681 , 0.63182103, 0.61509513,
0.61864938,
              0.62596697, 0.63098474, 0.61028643, 0.60777755,
0.61948568,
              0.62136734, 0.61844031, 0.61488605, 0.63203011,
0.5849885 ,
              0.58038888, 0.55989964, 0.55676354, 0.5507004 ,
0.55571817,
              0.55989964, 0.55571817, 0.55822705, 0.57578925,
0.58749739,
              0.57662555, 0.54693707, 0.5538365 , 0.54902781,
0.53272005,
              0.53125653, 0.52017562, 0.5019862 , 0.49905917,
0.47585198,
              0.48505122, 0.48400585, 0.46623458, 0.45452645,
0.45431737,
              0.4541083 , 0.45933515, 0.46623458, 0.47104328,
0.48337863,
              0.47627012, 0.47940623, 0.47313402, 0.44302739,
0.44281831,
              0.43424629, 0.43884591, 0.40727577, 0.42400167,
0.41459335,
              0.41522057, 0.41563872, 0.42818315, 0.42713778,
0.40936651,
              0.39891282])
,
       array([0.6121681 , 0.63182103, 0.61509513, 0.61864938,
0.62596697,
              0.63098474, 0.61028643, 0.60777755, 0.61948568,
0.62136734,
              0.61844031, 0.61488605, 0.63203011, 0.5849885 ,
0.58038888,
              0.55989964, 0.55676354, 0.5507004 , 0.55571817,
0.55989964,
              0.55571817, 0.55822705, 0.57578925, 0.58749739,
0.57662555,
              0.54693707, 0.5538365 , 0.54902781, 0.53272005,
0.53125653,
              0.52017562, 0.5019862 , 0.49905917, 0.47585198,
0.48505122,
              0.48400585, 0.46623458, 0.45452645, 0.45431737,
0.4541083 ,
              0.45933515, 0.46623458, 0.47104328, 0.48337863,
0.47627012,
              0.47940623, 0.47313402, 0.44302739, 0.44281831,
0.43424629,
              0.43884591, 0.40727577, 0.42400167, 0.41459335,
0.41522057,
```

```
              0.41563872, 0.42818315, 0.42713778, 0.40936651,
0.39891282,
              0.38281413])
,
       array([0.63182103, 0.61509513, 0.61864938, 0.62596697,
0.63098474,
              0.61028643, 0.60777755, 0.61948568, 0.62136734,
0.61844031,
              0.61488605, 0.63203011, 0.5849885 , 0.58038888,
0.55989964,
              0.55676354, 0.5507004 , 0.55571817, 0.55989964,
0.55571817,
              0.55822705, 0.57578925, 0.58749739, 0.57662555,
0.54693707,
              0.5538365 , 0.54902781, 0.53272005, 0.53125653,
0.52017562,
              0.5019862 , 0.49905917, 0.47585198, 0.48505122,
0.48400585,
              0.46623458, 0.45452645, 0.45431737, 0.4541083 ,
0.45933515,
              0.46623458, 0.47104328, 0.48337863, 0.47627012,
0.47940623,
              0.47313402, 0.44302739, 0.44281831, 0.43424629,
0.43884591,
              0.40727577, 0.42400167, 0.41459335, 0.41522057,
0.41563872,
              0.42818315, 0.42713778, 0.40936651, 0.39891282,
0.38281413,
              0.39514949])
,
       ...,
       array([0.05017771, 0.0430692 , 0.03721514, 0.03721514,
0.04097847,
              0.04557809, 0.04327828, 0.04223291, 0.03930587,
0.03616977,
              0.03596069, 0.02843404, 0.02717959, 0.02822496,
0.02174368,
              0.02467071, 0.02885218, 0.02195275, 0.01902572,
0.02655237,
              0.0263433 , 0.02216182, 0.01923479, 0.01860757,
0.02132553,
              0.0167259 , 0.01338072, 0.00355425, 0.00564499, 0.
,
              0.00689944])
,
       array([0.0430692 , 0.03721514, 0.03721514, 0.04097847,
0.04557809,
              0.04327828, 0.04223291, 0.03930587, 0.03616977,
0.03596069,
              0.02843404, 0.02717959, 0.02822496, 0.02174368,
```

```
0.02467071,
              0.02885218, 0.02195275, 0.01902572, 0.02655237,
0.0263433 ,
              0.02216182, 0.01923479, 0.01860757, 0.02132553,
0.0167259 ,
              0.01338072, 0.00355425, 0.00564499, 0.        ,
0.00689944]),
       array([0.03721514, 0.03721514, 0.04097847, 0.04557809,
0.04327828,
              0.04223291, 0.03930587, 0.03616977, 0.03596069,
0.02843404,
              0.02717959, 0.02822496, 0.02174368, 0.02467071,
0.02885218,
              0.02195275, 0.01902572, 0.02655237, 0.0263433 ,
0.02216182,
              0.01923479, 0.01860757, 0.02132553, 0.0167259 ,
0.01338072,
              0.00355425, 0.00564499, 0.        , 0.00689944])
],
      dtype=object)
```

In [15]:

```
x_train = x_temp[:,:-1]
y_train = x_temp[:,-1]
x_train
y_train
```

```
---------------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_456/1289817907.py in <module>
----> 1 x_train = x_temp[:,:-1]
      2 y_train = x_temp[:,-1]
      3 x_train
      4 y_train

IndexError: too many indices for array: array is 1-dimensional, but 2
were indexed
```

In [ ]:

In [ ]:

```
x_train.shape
```

In [ ]:

```
 x_train=x_train.reshape(-1,)
```

In [ ]:

```
y_train.shape
```

In [ ]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,LSTM,Dropout
```

In [ ]:

```
model = Sequential()
model.add(LSTM(50,input_shape=(60,1)),return_sequence = True)
model.add(Dropout(0.2) # indicate 20% wt we are going to dropout
model.add(LSTM(50,return_sequence = True))
model.add(Dropout(0.2))
model.add(LSTM(50,return_sequence = True))
model.add(Dropout(0.2))
model.add(LSTM(50,return_sequence = False))
model.add(Dense(1))
```

In [ ]:

```
model.compile(optimizer = 'adam',loss =  'mean_squared_error',metric
=['accuracy'])
```

In [ ]:

```
h = model.fit(x_train,y_train,epochs= 50, batch_size = 64)
```

In [ ]:

```
h.history.
```

In [ ]:

```
h.history['loss']
```

In [ ]:

```
len(h.history['loss'])
```

In [ ]:

```
plt.plot(h.history['loss'][2:1])
```

In [ ]:

```
temp = scaler.transform(df_test[['Open']])
```

In [ ]:

```
temp
```

In [ ]:

```
temp[:,0]
```

In [ ]:

```
x_temp
```

In [ ]:

```
len(x[-60:,0])
```

In [ ]:

```
list(temp[:,0])+list(x[-60:,0])
```

In [ ]:

```
test_data = (list(temp[:,0])+list(x[-60:,0]))
```

In [ ]:

```
test_data
```

In [ ]:

```
x_test = []
for i in range(0,60):
    row = test_data[1:60]
```

In [ ]:

```
x_test
```

In [ ]:

```
y_pre = model.predict(x_test.reshape(-1,60))
```

In [ ]:

```
x_test = []
for i in range(0,20):
    row = test_data[1:60]
    x_test.append(row)
x_test = np.array(x_test)
x_test.shape
```

In [ ]:

```
x_test
```

In [ ]:

```
y_pre = moel.predict(x_test.reshape(-1,60,1))
```

In [ ]:

```
y_pre
```

In [ ]:

```
y_pre = scaler.inverse.transform(y_pre)
```

In [ ]:

```
y_pre
```

In [ ]:

```
df_test['Open']
```

In [ ]:

```
plt.plot(df_test['Open'],c='blue')
plt.plot(y_pre,c= 'red')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: