Stock market prediction using LSTM

In [34]:
```
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
```

In [35]:
```
df=pd.read_csv(r"C:\Users\Jaswanth Reddy\Downloads\MINDTREE.NS (1).csv")
df.head()
```

Out[35]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-12-24	848.799988	856.900024	834.150024	837.700012	776.659790	690775.0
1	2018-12-26	839.000000	859.849976	818.000000	854.049988	791.818359	1128860.0
2	2018-12-27	859.349976	872.900024	853.000000	861.849976	799.050049	1251109.0
3	2018-12-28	863.000000	869.000000	858.049988	864.099976	801.136047	420544.0
4	2018-12-31	870.000000	871.700012	861.549988	864.500000	801.506958	317152.0

In [36]:
```
df.isnull().sum()
```

Out[36]:
```
Date         0
Open         2
High         2
Low          2
Close        2
Adj Close    2
Volume       2
dtype: int64
```

In [37]:
```
df=df.dropna()
df.shape
```

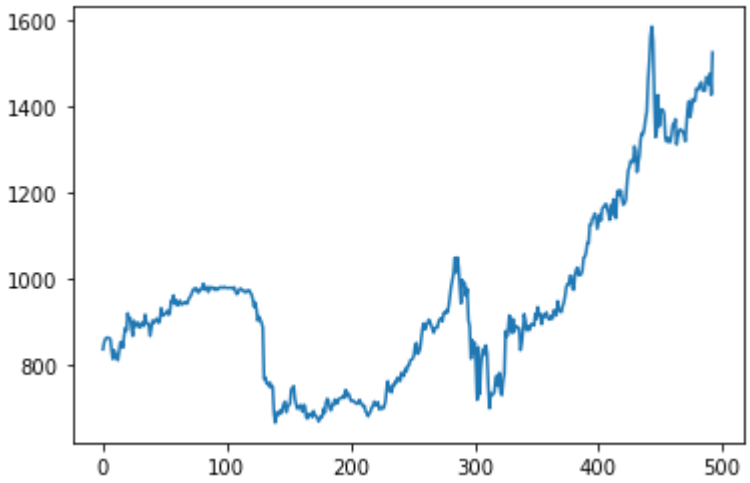
Out[37]:
```
(491, 7)
```

In [108]:
```
df1=df['Close']
df1.head(30)
```

Out[108]:
```
0     837.700012
1     854.049988
2     861.849976
3     864.099976
4     864.500000
5     863.400024
6     860.500000
7     834.799988
8     815.000000
9     835.349976
10    821.200012
11    823.450012
12    812.250000
13    832.000000
14    853.150024
15    854.500000
16    841.200012
17    867.849976
18    887.000000
19    881.750000
20    921.099976
21    906.549988
22    910.400024
23    886.099976
24    868.599976
25    903.599976
26    890.450012
27    894.049988
28    899.650024
29    889.799988
Name: Close, dtype: float64
```

In [39]:
```
import matplotlib.pyplot as plt
plt.plot(df['Close'])
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x18b5067b040>]
```



In [40]:
```
# Preparing dependent and independent data
def prepare_data(timeseries_data, n_features):
    X, y =[],[]
    for i in range(len(timeseries_data)):
        end_ix = i + n_features
        if end_ix > len(timeseries_data)-1:
            break
        seq_x, seq_y = timeseries_data[i:end_ix], timeseries_data[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

In [62]:
```
# define input sequence
timeseries_data = np.array(df1)
# number of time steps
n_steps = 10
# spliting into samples
X, y = prepare_data(timeseries_data, n_steps)
```

In [63]:
```
print(X)
```

```
[[ 837.700012  854.049988  861.849976 ...  834.799988  815.
   835.349976]
 [ 854.049988  861.849976  864.099976 ...  815.        835.349976
   821.200012]
 [ 861.849976  864.099976  864.5      ...  835.349976  821.200012
   823.450012]
 ...
 [1437.949951 1442.699951 1452.300049 ... 1468.349976 1459.5
  1452.099976]
 [1442.699951 1452.300049 1456.       ... 1459.5      1452.099976
  1476.099976]
 [1452.300049 1456.       1437.25     ... 1452.099976 1476.099976
  1427.150024]]
```

In [64]:
```
print(y)
```

```
[ 821.200012  823.450012  812.25      832.        853.150024  854.5
  841.200012  867.849976  887.        881.75      921.099976  906.549988
  910.400024  886.099976  868.599976  903.599976  890.450012  894.049988
  899.650024  889.799988  887.900024  897.299988  899.299988  891.549988
  918.049988  900.299988  897.        895.5       868.599976  887.299988
  903.950012  898.799988  902.75      907.75      907.25      898.700012
  908.799988  932.849976  915.650024  917.099976  920.200012  921.700012
  927.        917.400024  923.900024  950.099976  947.400024  962.5
  943.299988  950.650024  939.        940.75      950.5       943.150024
  941.25      944.950012  946.650024  943.799988  943.5       952.950012
  957.299988  962.200012  970.25      976.650024  975.650024  979.650024
  972.450012  968.5       976.200012  975.400024  979.400024  989.400024
  975.849976  981.25      981.400024  970.200012  981.75      979.950012
  980.5       980.450012  975.349976  979.099976  975.200012  977.049988
  980.349976  981.799988  980.049988  980.099976  981.900024  979.849976
  980.049988  979.599976  980.099976  980.099976  980.099976  977.900024
  981.25      971.799988  965.150024  969.549988  975.5       978.549988
  975.        974.799988  971.099976  969.200012  973.099976  971.549988
  974.5       968.5       964.099976  952.450012  935.950012  945.5
  927.400024  903.650024  912.75      905.5       899.049988  888.549988
  769.25      772.450012  758.5       759.450012  752.150024  759.950012
  748.400024  750.700012  690.650024  667.599976  681.950012  692.599976
  686.549988  695.150024  689.950012  700.450012  711.75      716.299988
  691.950012  704.599976  707.599976  712.299988  745.299988  747.799988
  752.150024  721.450012  708.650024  699.799988  702.200012  708.299988
  700.849976  693.299988  709.049988  706.25      680.700012  677.099976
  687.549988  688.599976  684.299988  680.450012  694.400024  685.5
  682.75      679.650024  670.349976  676.650024  681.799988  681.
  699.950012  690.950012  713.650024  722.900024  718.349976  700.400024
  696.299988  707.849976  707.799988  720.900024  720.049988  711.700012
  721.200012  723.5       724.5       724.900024  730.549988  725.349976
  743.400024  729.400024  735.700012  729.200012  718.450012  718.200012
  718.700012  713.        711.200012  712.799988  720.349976  714.349976
  710.049988  705.099976  707.25      693.799988  689.599976  682.549988
  689.549988  693.25      703.700012  705.049988  716.25      710.150024
  708.200012  714.849976  698.400024  701.5       703.349976  699.849976
  705.25      717.099976  746.799988  763.5       741.299988  749.400024
  737.549988  754.799988  752.849976  763.950012  761.        772.400024
  766.799988  764.400024  783.450012  777.650024  775.799988  792.450012
  786.849976  799.700012  800.150024  811.75      813.75      815.549988
```

```
 824.099976  846.25      852.150024  827.049988  830.849976  839.900024
 864.349976  887.849976  897.599976  883.549988  892.75      897.75
 906.25      900.75      892.75      885.049988  876.        886.549988
 888.099976  887.900024  902.75      907.599976  909.75      902.349976
 921.799988  920.299988  926.950012  922.849976  936.549988  959.900024
 981.950012  997.       1013.599976 1049.5      1015.549988 1050.300049
1029.400024  985.700012  944.        998.099976  989.049988  990.5
 961.599976  977.599976  905.150024  891.099976  816.        859.549988
 831.950012  853.        804.299988  720.5       842.450012  733.900024
 792.049988  822.200012  837.650024  825.650024  846.400024  828.700012
 753.5       701.099976  735.75      732.099976  733.75      739.650024
 771.900024  776.849976  752.650024  783.049988  736.349976  731.5
 765.049988  780.349976  879.650024  867.299988  881.599976  915.950012
 903.400024  875.299988  907.849976  877.700012  891.099976  890.599976
 887.400024  885.650024  835.849976  852.150024  879.549988  918.900024
 904.400024  882.        890.700012  881.799988  891.450012  896.5
 894.599976  892.400024  920.400024  908.349976  935.349976  919.150024
 918.599976  895.950012  909.099976  918.700012  914.        923.
 912.450012  906.849976  907.400024  915.099976  909.049988  928.049988
 924.400024  916.700012  949.099976  930.700012  925.650024  923.700012
 937.400024  945.150024  960.599976  982.049988  988.299988  988.150024
1007.950012 1007.799988  978.150024  975.5      1008.900024 1022.
1026.800049 1009.200012 1013.099976 1011.150024 1021.       1050.099976
1052.050049 1062.75     1083.800049 1083.400024 1126.949951 1125.900024
1140.300049 1142.75     1152.25     1142.199951 1116.75     1133.300049
1149.300049 1135.949951 1163.400024 1166.75     1171.699951 1174.650024
1162.550049 1153.099976 1137.349976 1171.75     1149.349976 1184.699951
1155.75     1141.300049 1194.150024 1205.800049 1194.150024 1205.300049
1186.949951 1172.5      1175.699951 1183.800049 1221.699951 1251.699951
1260.599976 1274.699951 1272.5      1271.199951 1307.699951 1297.849976
1248.150024 1268.900024 1285.75     1323.5      1338.5      1336.300049
1349.449951 1371.949951 1388.199951 1461.150024 1504.25     1562.349976
1585.199951 1551.900024 1424.849976 1329.150024 1341.75     1426.349976
1354.050049 1388.349976 1393.300049 1391.949951 1381.449951 1327.25
1318.650024 1327.599976 1317.849976 1317.800049 1335.25     1357.900024
1356.699951 1370.599976 1312.       1329.650024 1342.099976 1347.599976
1342.099976 1334.300049 1319.300049 1357.449951 1395.849976 1412.150024
1376.099976 1391.599976 1417.300049 1411.599976 1414.300049 1441.699951
1437.949951 1442.699951 1452.300049 1456.       1437.25     1436.75
1437.050049 1468.349976 1459.5      1452.099976 1476.099976 1427.150024
1525.050049]
```

In [65]:
```
X.shape
```

Out[65]:
```
(481, 10)
```

In [66]:
```
y.shape
```

Out[66]:
```
(481,)
```

In [67]:
```
# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))
```

In [126]:
```
# Training model
model = Sequential()
model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(LSTM(100, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mae')
```

In [127]:
```
# fit model
model.fit(X, y, epochs=300, verbose=1)
```

```
16/16 [==============================] - 0s 12ms/step - loss: 31.4750
Epoch 293/300

16/16 [==============================] - 0s 10ms/step - loss: 27.1738
Epoch 294/300
16/16 [==============================] - 0s 9ms/step - loss: 23.5257
Epoch 295/300
16/16 [==============================] - 0s 9ms/step - loss: 24.2602
Epoch 296/300
16/16 [==============================] - 0s 13ms/step - loss: 21.4785
Epoch 297/300
16/16 [==============================] - 0s 13ms/step - loss: 21.8275
Epoch 298/300
16/16 [==============================] - 0s 14ms/step - loss: 30.6590
Epoch 299/300
16/16 [==============================] - 0s 10ms/step - loss: 19.9293
Epoch 300/300
16/16 [==============================] - 0s 16ms/step - loss: 22.4484
```

Out[127]:
```
<tensorflow.python.keras.callbacks.History at 0x18b75847640>
```

In [106]:
```
x_input.shape
```

Out[106]:
```
(1, 10, 1)
```

In [118]:

```
# prediction for next 10 days
x_input = np.array([1442.69, 1437.94,1452.30, 1456.0, 1437.25, 1436.75, 1437.05, 1468.34, 1459.5, 1452.09, 1476.
#x_input=np.array([815.3,821.200012  ,823.450012,  812.25 ,     832.   ,     853.150024 , 854.5,841.200012,  867
temp_input=list(x_input)
lst_output=[]
i=0
while(i<15):
    if(len(temp_input)>3):
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input = x_input.reshape((1, n_steps, n_features))
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.append(yhat[0][0])
        temp_input=temp_input[1:]
        lst_output.append(yhat[0][0]) # storing output values
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps, n_features))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.append(yhat[0][0])
        lst_output.append(yhat[0][0])
        i=i+1

print(lst_output)
```

```
0 day input [1437.94  1452.3   1456.    1437.25  1436.75  1437.05  1468.34  1459.5
 1452.09  1476.099]
0 day output [[1453.7325]]
1 day input [1452.3        1456.         1437.25       1436.75       1437.05
 1468.34       1459.5        1452.09       1476.099      1453.73254395]
1 day output [[1446.1445]]
2 day input [1456.         1437.25       1436.75       1437.05       1468.34
 1459.5        1452.09       1476.099      1453.73254395 1446.14453125]
2 day output [[1445.3861]]
3 day input [1437.25       1436.75       1437.05       1468.34       1459.5
 1452.09       1476.099      1453.73254395 1446.14453125 1445.3861084 ]
3 day output [[1445.1527]]
4 day input [1436.75       1437.05       1468.34       1459.5        1452.09
 1476.099      1453.73254395 1446.14453125 1445.3861084  1445.15270996]
```

```
4 day output [[1446.4323]]
5 day input [1437.05       1468.34       1459.5        1452.09       1476.099
 1453.73254395 1446.14453125 1445.3861084  1445.15270996 1446.43225098]
5 day output [[1446.7557]]
6 day input [1468.34       1459.5        1452.09       1476.099      1453.73254395
 1446.14453125 1445.3861084  1445.15270996 1446.43225098 1446.7557373 ]
6 day output [[1447.4006]]
7 day input [1459.5        1452.09       1476.099      1453.73254395 1446.14453125
 1445.3861084  1445.15270996 1446.43225098 1446.7557373  1447.40063477]
7 day output [[1445.2279]]
8 day input [1452.09       1476.099      1453.73254395 1446.14453125 1445.3861084
 1445.15270996 1446.43225098 1446.7557373  1447.40063477 1445.22790527]
8 day output [[1443.3448]]
9 day input [1476.099      1453.73254395 1446.14453125 1445.3861084  1445.15270996
 1446.43225098 1446.7557373  1447.40063477 1445.22790527 1443.34484863]
9 day output [[1442.185]]
10 day input [1453.7325 1446.1445 1445.3861 1445.1527 1446.4323 1446.7557 1447.4006
 1445.2279 1443.3448 1442.185 ]
10 day output [[1439.3645]]
11 day input [1446.1445 1445.3861 1445.1527 1446.4323 1446.7557 1447.4006 1445.2279
 1443.3448 1442.185  1439.3645]
11 day output [[1438.1947]]
12 day input [1445.3861 1445.1527 1446.4323 1446.7557 1447.4006 1445.2279 1443.3448
 1442.185  1439.3645 1438.1947]
12 day output [[1437.4111]]
13 day input [1445.1527 1446.4323 1446.7557 1447.4006 1445.2279 1443.3448 1442.185
 1439.3645 1438.1947 1437.4111]
13 day output [[1436.6111]]
14 day input [1446.4323 1446.7557 1447.4006 1445.2279 1443.3448 1442.185  1439.3645
 1438.1947 1437.4111 1436.6111]
14 day output [[1435.6958]]
[1453.7325, 1446.1445, 1445.3861, 1445.1527, 1446.4323, 1446.7557, 1447.4006, 1445.2279, 1443.3448, 1442.18
5, 1439.3645, 1438.1947, 1437.4111, 1436.6111, 1435.6958]
```

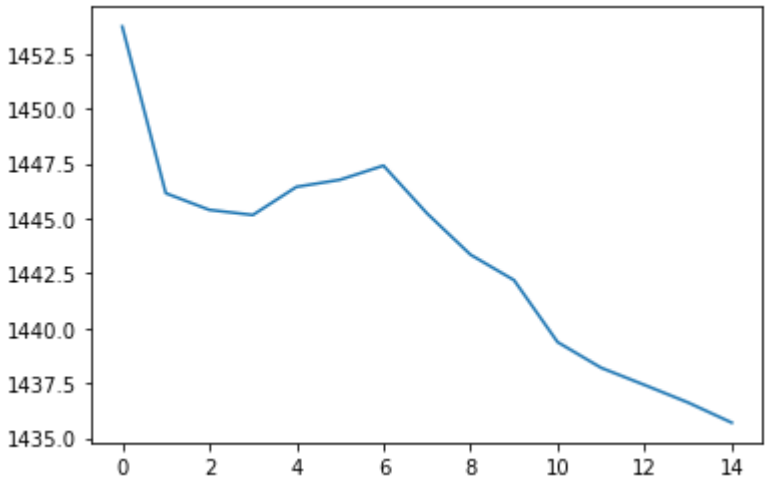
In [109]:
```
df.tail(18)
```

Out[109]:

	Date	Open	High	Low	Close	Adj Close	Volume
475	2020-11-26	1379.900024	1402.000000	1348.650024	1391.599976	1391.599976	759889.0
476	2020-11-27	1410.000000	1426.900024	1383.449951	1417.300049	1417.300049	1354040.0
477	2020-12-01	1417.300049	1423.949951	1391.000000	1411.599976	1411.599976	712194.0
478	2020-12-02	1411.599976	1421.500000	1394.000000	1414.300049	1414.300049	672487.0
479	2020-12-03	1414.000000	1444.500000	1414.000000	1441.699951	1441.699951	1001572.0
480	2020-12-04	1443.000000	1454.099976	1428.099976	1437.949951	1437.949951	734335.0
481	2020-12-07	1440.000000	1466.000000	1433.650024	1442.699951	1442.699951	892569.0
482	2020-12-08	1456.000000	1472.000000	1428.800049	1452.300049	1452.300049	1174551.0
483	2020-12-09	1465.400024	1472.000000	1450.300049	1456.000000	1456.000000	813977.0
484	2020-12-10	1455.000000	1455.000000	1428.150024	1437.250000	1437.250000	631499.0
485	2020-12-11	1444.000000	1451.800049	1422.050049	1436.750000	1436.750000	371319.0
486	2020-12-14	1436.599976	1445.099976	1417.550049	1437.050049	1437.050049	273214.0
487	2020-12-15	1439.449951	1470.000000	1431.000000	1468.349976	1468.349976	695656.0
488	2020-12-16	1475.050049	1484.000000	1455.050049	1459.500000	1459.500000	895448.0
489	2020-12-17	1470.000000	1472.500000	1446.250000	1452.099976	1452.099976	573057.0
490	2020-12-18	1465.000000	1484.800049	1436.050049	1476.099976	1476.099976	1836980.0
491	2020-12-21	1476.099976	1491.750000	1399.699951	1427.150024	1427.150024	1613406.0
492	2020-12-22	1436.000000	1530.000000	1430.550049	1525.050049	1525.050049	2779116.0

In [119]:
```
# Based on the last 10 day results, predicted the further values
plt.plot(lst_output)
```

Out[119]:
```
[<matplotlib.lines.Line2D at 0x18b6e73b3d0>]
```



In [ ]: