```
#Importing Important Libraries
```

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

```
df = pd.read_csv(r'/content/BAJAJ_AUTO__EQ__NSE__NSE__MINUTE.csv')
```

```
df
```

	timestamp	open	high	low	close	volume
0	2017-01-02 09:15:00+05:30	2640.15	2654.30	2617.55	2627.00	2235.0
1	2017-01-02 09:16:00+05:30	2627.00	2646.10	2612.35	2612.35	2806.0
2	2017-01-02 09:17:00+05:30	2614.45	2614.45	2591.30	2596.00	7443.0
3	2017-01-02 09:18:00+05:30	2596.00	2596.00	2587.75	2590.80	3289.0
4	2017-01-02 09:19:00+05:30	2593.00	2596.95	2584.00	2589.95	4862.0
...	...	...	...	...	...	...
170336	2018-11-02 12:56:00+05:30	2686.35	2688.60	2685.20	2687.95	705.0
170337	2018-11-02 12:57:00+05:30	2687.55	2688.40	2686.25	2687.55	915.0
170338	2018-11-02 12:58:00+05:30	2688.00	2688.40	2686.05	2688.40	421.0
170339	2018-11-02 12:59:00+05:30	2688.40	2688.60	2686.35	2686.35	212.0
170340	2018-11-02 13:00:00+0	NaN	NaN	NaN	NaN	NaN

170341 rows × 6 columns

```
#Droping all the null values
```

```
df.dropna(inplace=True)
```

```
#Checking the top 10 values from the dataset
```

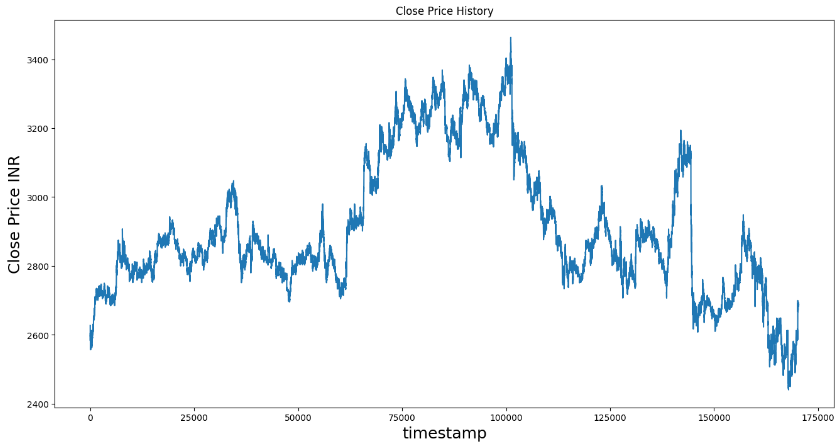
```
df.head(10)
```

	timestamp	open	high	low	close	volume
0	2017-01-02 09:15:00+05:30	2640.15	2654.30	2617.55	2627.00	2235.0
1	2017-01-02 09:16:00+05:30	2627.00	2646.10	2612.35	2612.35	2806.0
2	2017-01-02 09:17:00+05:30	2614.45	2614.45	2591.30	2596.00	7443.0
3	2017-01-02 09:18:00+05:30	2596.00	2596.00	2587.75	2590.80	3289.0
4	2017-01-02 09:19:00+05:30	2593.00	2596.95	2584.00	2589.95	4862.0
5	2017-01-02 09:20:00+05:30	2587.95	2589.35	2583.00	2583.15	2551.0
6	2017-01-02 09:21:00+05:30	2583.00	2596.95	2581.00	2591.40	4544.0
7	2017-01-02 09:22:00+05:30	2591.40	2599.00	2591.40	2599.00	2404.0
8	2017-01-02 09:23:00+05:30	2599.00	2602.80	2598.85	2600.00	2241.0
9	2017-01-02 09:24:00+05:30	2600.00	2603.20	2598.65	2603.20	1145.0

```
#Ploting Close Price History using matplotlib
```

```
import seaborn as sns
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['close'])
```

```
#ax=sns.lineplot(data=df, x='timestamp',y='close', color="blue");
plt.xlabel('timestamp',fontsize=18)
plt.ylabel('Close Price INR',fontsize=18)
plt.show()
```



```
#Converting data to a numpy array
```

```
data = df.filter(['close'])
dataset = data.values
training_data_len = math.ceil( len(dataset) *.8)
```

```
dataset
```

```
     array([[2627.  ],
            [2612.35],
            [2596.  ],
            ...,
            [2687.55],
            [2688.4 ],
            [2686.35]])
```

```
#Transforming the dataset array to range between 0 and 1
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

```
train_data = scaled_data[0:training_data_len  , : ]
x_train=[]
y_train = []
for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i,0])
    y_train.append(train_data[i,0])
```

```
#Spliting data for training and testing
```

```
x_train, y_train = np.array(x_train), np.array(y_train)
```

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

```
#Building a LSTM Model for Stock Market Prediction
```

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True,input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=25))
model.add(Dense(units=1))
```

```
#Using adam optimizer and mean_squared_error as the loss function
```

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
model.fit(x_train, y_train, batch_size=64, epochs=1)
```

```
     2129/2129 [==============================] - 176s 81ms/step - loss: 5.2746e-04
     <keras.callbacks.History at 0x7f6fc1be37c0>
```

```
test_data = scaled_data[training_data_len - 60: , : ]#Create the x_test and y_test data sets
x_test = []
y_test =  dataset[training_data_len : , : ]
for i in range(60,len(test_data)):
    x_test.append(test_data[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))
```

```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
     1065/1065 [==============================] - 26s 22ms/step
```

```
#Finding the root mean squared error
```

```
rmse=np.sqrt(np.mean(((predictions- y_test)**2)))
rmse
```

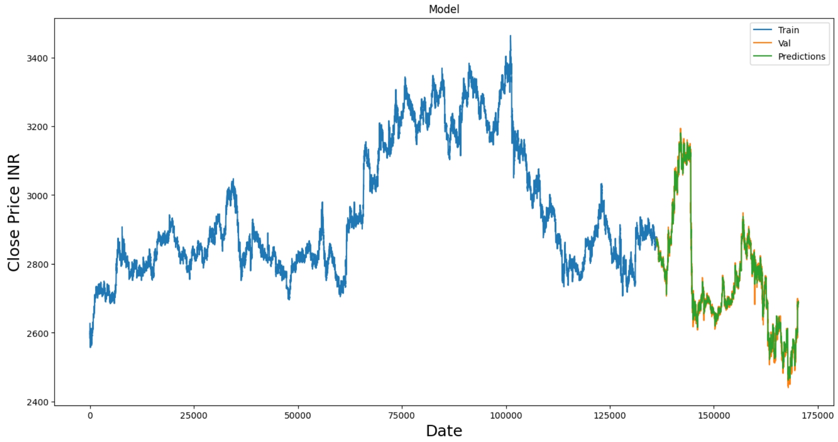
```
     6.8522945093682015
```

```
#Plotting the predicted values
```

```
train = data[:training_data_len]
display = data[training_data_len:]
display['Predictions'] = predictions#Visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price INR', fontsize=18)
plt.plot(train['close'])
plt.plot(display['close'])
plt.plot(display['Predictions'])
plt.legend(['Train', 'Val', 'Predictions'], loc='upper right')
plt.show()
```

```
<ipython-input-33-d1d3e758b1fa>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  display['Predictions'] = predictions#Visualize the data
```



Colab paid products - Cancel contracts here

✓ 1s completed at 16:18