Imports/Initial Data

we import numpy for making scientific computations, pandas for loading and modifying datasets, and matplotlib for plotting graphs.

In [19]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [20]:

```
url = 'https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NSE-TATAGLOBAL.c
dataset_train = pd.read_csv(url)
training_set = dataset_train.iloc[:, 1:2].values
```

In [21]:

```
dataset_train.head()
```

Out[21]:

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-09-28	234.05	235.95	230.20	233.50	233.75	3069914	7162.35
1	2018-09-27	234.55	236.80	231.10	233.80	233.25	5082859	11859.95
2	2018-09-26	240.00	240.00	232.50	235.00	234.25	2240909	5248.60
3	2018-09-25	233.30	236.75	232.00	236.25	236.10	2349368	5503.90
4	2018-09-24	233.55	239.20	230.75	234.00	233.30	3423509	7999.55

Data Normalization

Normalization is changing the values of numeric columns in the dataset to a common scale, which helps the performance of our model. To scale the training dataset we use Scikit-Learn's MinMaxScaler with numbers between zero and one.

In [4]:

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

Incorporating Timesteps Into Data

We should input our data in the form of a 3D array to the LSTM model. First, we create data in 60 timesteps before using numpy to convert it into an array. Finally, we convert the data into a 3D array with X_train samples, 60 timestamps, and one feature at each step.

In [6]:

```
X_train = []
y_train = []
for i in range(60, 2035):
    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)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

Creating the LSTM Model

Before we can develop the LSTM, we have to make a few imports from Keras: Sequential for initializing the neural network, LSTM to add the LSTM layer, Dropout for preventing overfitting with dropout layers, and Dense to add a densely connected neural network layer.

In [7]:

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

In [8]:

```
model = Sequential()
model.add(LSTM(units=50,return_sequences=True,input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(X_train,y_train,epochs=100,batch_size=32)
```

```
04
Epoch 75/100
62/62 [==============================] - 6s 102ms/step - loss: 7.1993e-
04
Epoch 76/100
62/62 [==============================] - 6s 97ms/step - loss: 7.1372e-0
4
Epoch 77/100
62/62 [==============================] - 7s 105ms/step - loss: 6.7776e-
04
Epoch 78/100
62/62 [==============================] - 6s 97ms/step - loss: 6.9611e-0
4
Epoch 79/100
62/62 [==============================] - 6s 101ms/step - loss: 7.5705e-
04
Epoch 80/100
62/62 [==============================] - 6s 100ms/step - loss: 5.9560e-
04
Epoch 81/100
```

Making Predictions on the Test Set

In [18]:

```
url = 'https://raw.githubusercontent.com/mwitiderrick/stockprice/master/tatatest.csv'
dataset_test = pd.read_csv(url)
real_stock_price = dataset_test.iloc[:, 1:2].values
real_stock_price
```

Out[18]:

```
array([[220.1 ],
       [221.1 ],
       [229.45],
       [230.3 ],
       [237.7 ],
       [237.1 ],
       [229.7 ],
       [226.25],
       [215.  ],
       [215.  ],
       [215.5 ],
       [208.  ],
       [217.  ],
       [223.5 ],
       [230.  ],
       [234.55]])
```

In [22]:

```
dataset_total = pd.concat((dataset_train['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, 76):
    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 = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
predicted_stock_price
```

```
1/1 [==============================] - 0s 71ms/step
```

Out[22]:

```
array([[121.91632],
       [187.00351],
       [221.43755],
       [230.38322],
       [229.75122],
       [234.1822 ],
       [237.79788],
       [235.66179],
       [232.53702],
       [224.67242],
       [221.05115],
       [220.44574],
       [215.72054],
       [218.35175],
       [223.9955 ],
       [229.98834]], dtype=float32)
```

Plotting the Results

In [12]:

```
plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```