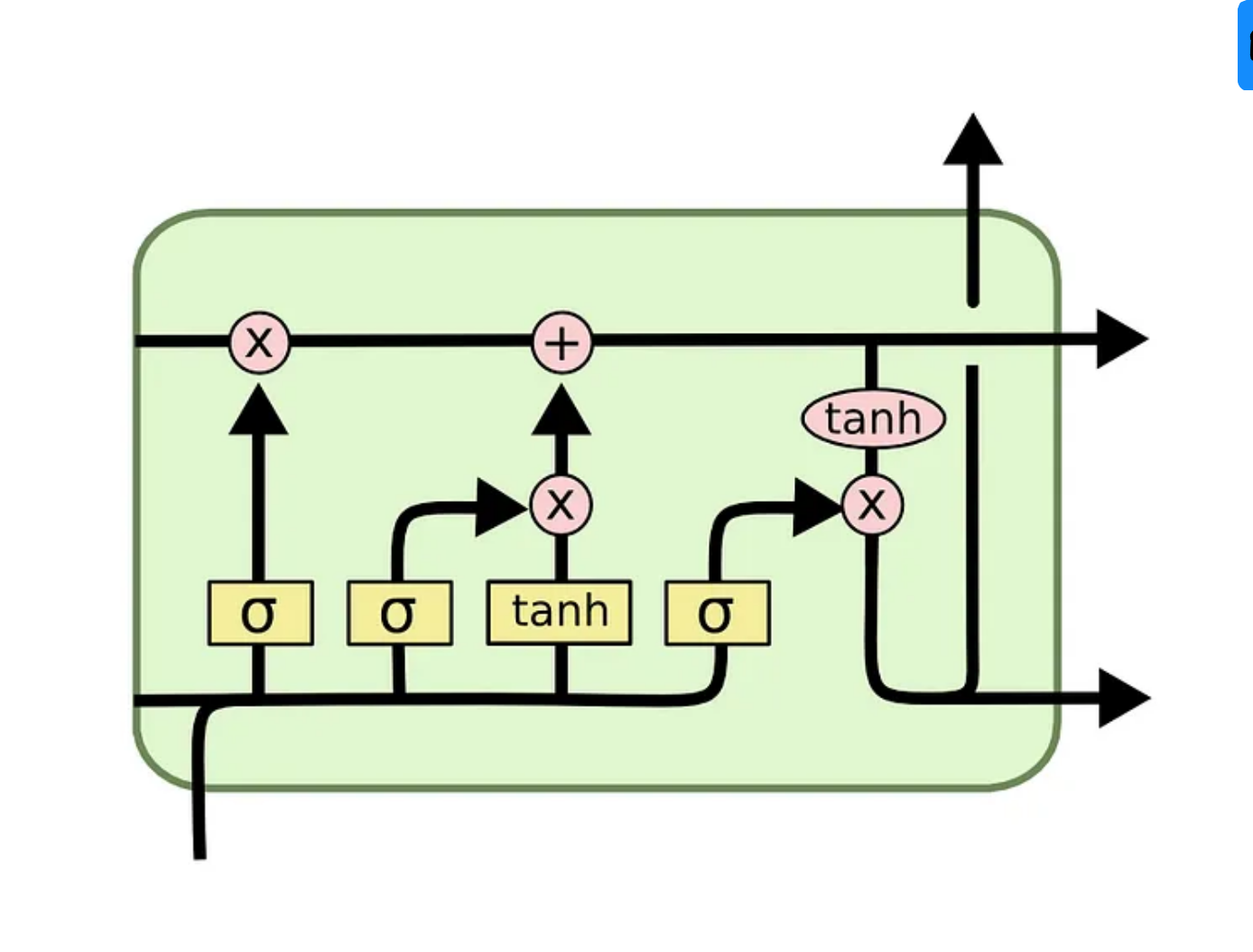
***Stock Prediction***

As financial institutions begin to embrace artificial intelligence, machine learning is increasingly utilized to help make trading decisions. Although there is an abundance of stock data for machine learning models to train on, a high noise to signal ratio and the multitude of factors that affect stock prices are among the several reasons that predicting the market difficult.

At the same time, these models don’t need to reach high levels of accuracy because even 60% accuracy can deliver solid returns. One method for predicting stock prices is using a long short-term memory neural network (LSTM) for times series forecasting.

***LSTM: A Brief Explanation***

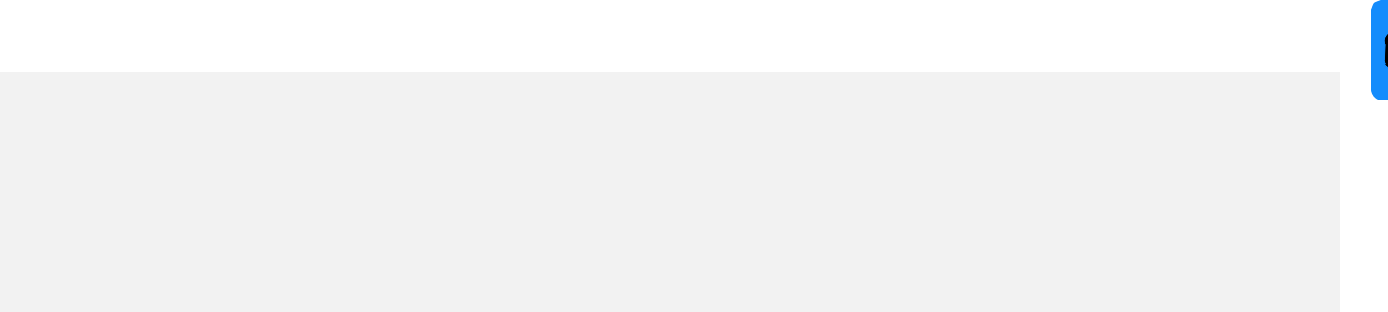


*LSTM diagram* [*(source)*](https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs are an improved version of recurrent neural networks (RNNs). RNNs are analogous to human learning. When humans think, we don’t start our thinking from scratch each second. For example, in the sentence “Bob plays basketball”, we know that Bob is the person who plays basketball because we retain information about past words while reading sentences. Similarly, RNNs are networks with loops in them, which allow them to use past information before arriving at a final output. However, RNNs can only connect recent previous information and cannot connect information as the time gap grows. This is where LSTMs come into play; LSTMs are a type of RNN that remember information over long periods of time, making them better suited for predicting stock prices. For a technical explanation of LSTMs click [here](https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21).

***Imports/Initial Data***

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

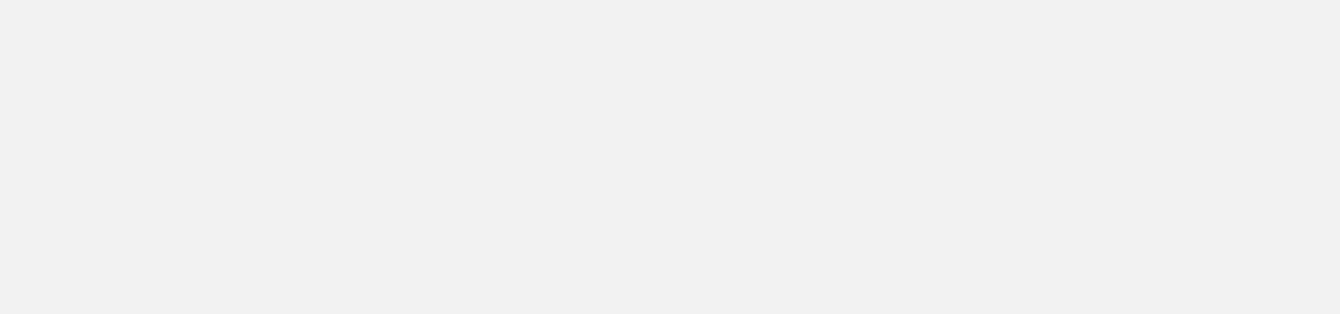


import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

After making the necessary imports, we load data on Tata Global Beverage’s past stock prices. From the data, we select the values of the first and second columns (“Open” and “High” respectively) as our training dataset. The “Open” column represents the opening price for shares that day and the “High” column represents the highest price shares reached that day.



url =

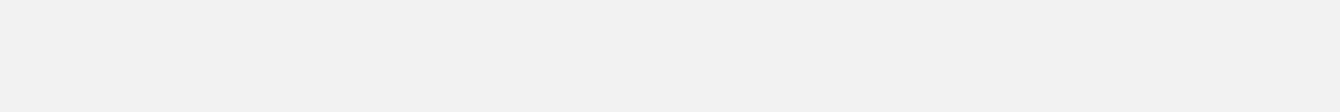
'https://raw.githubusercontent.com/mwitiderrick/stockprice/master/NS

E-TATAGLOBAL.csv'

dataset\_train = pd.read\_csv(url)

training\_set = dataset\_train.iloc[:, 1:2].values

To get a look at the dataset we’re using, we can check the head, which shows us the first five rows of our dataset.



dataset\_train.head()

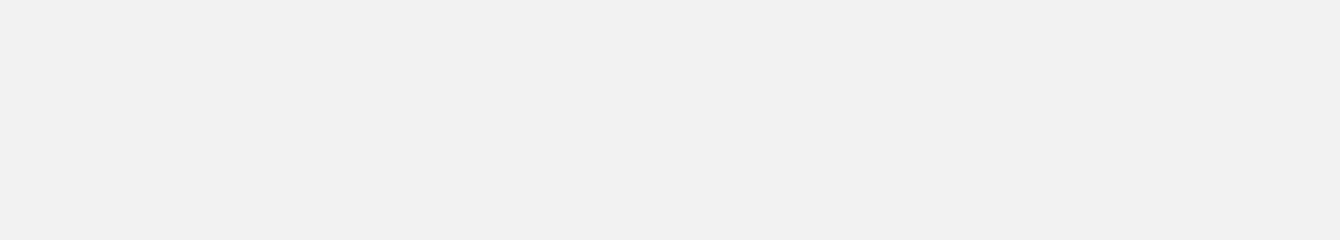


“Low” represents the lowest share price for the day, “Last” represents the price at which the last transaction for a share went through. “Close” represents the price shares ended at for the day.



***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.



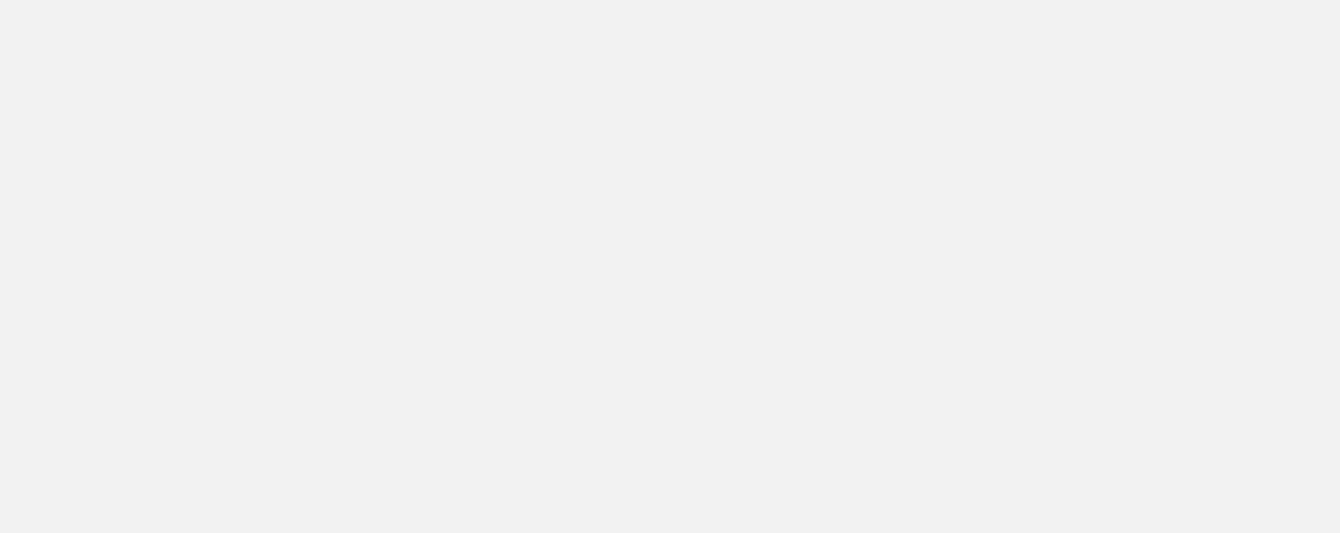
from sklearn.preprocessing import MinMaxScaler

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.



X\_train = []

y\_train = []

for i in range(60, 2035):

X\_train.append(training\_set\_scaled[i-60:i, 0])

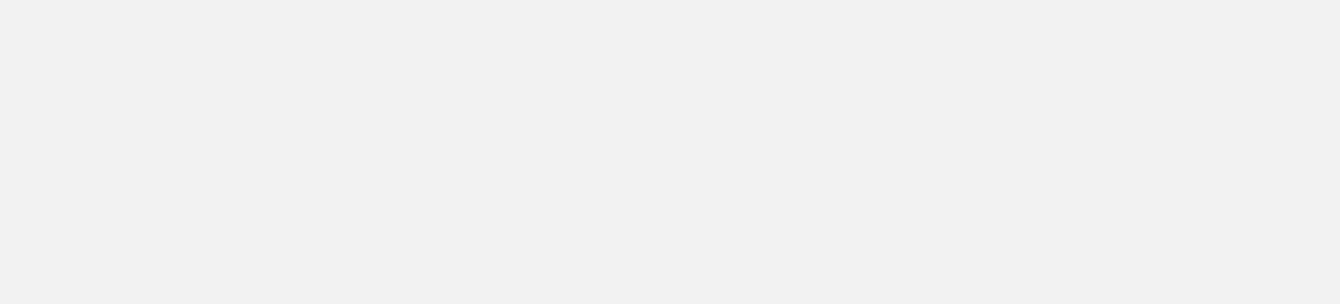
y\_train.append(training\_set\_scaled[i, 0])

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.



from keras.models import Sequential

from keras.layers import LSTM

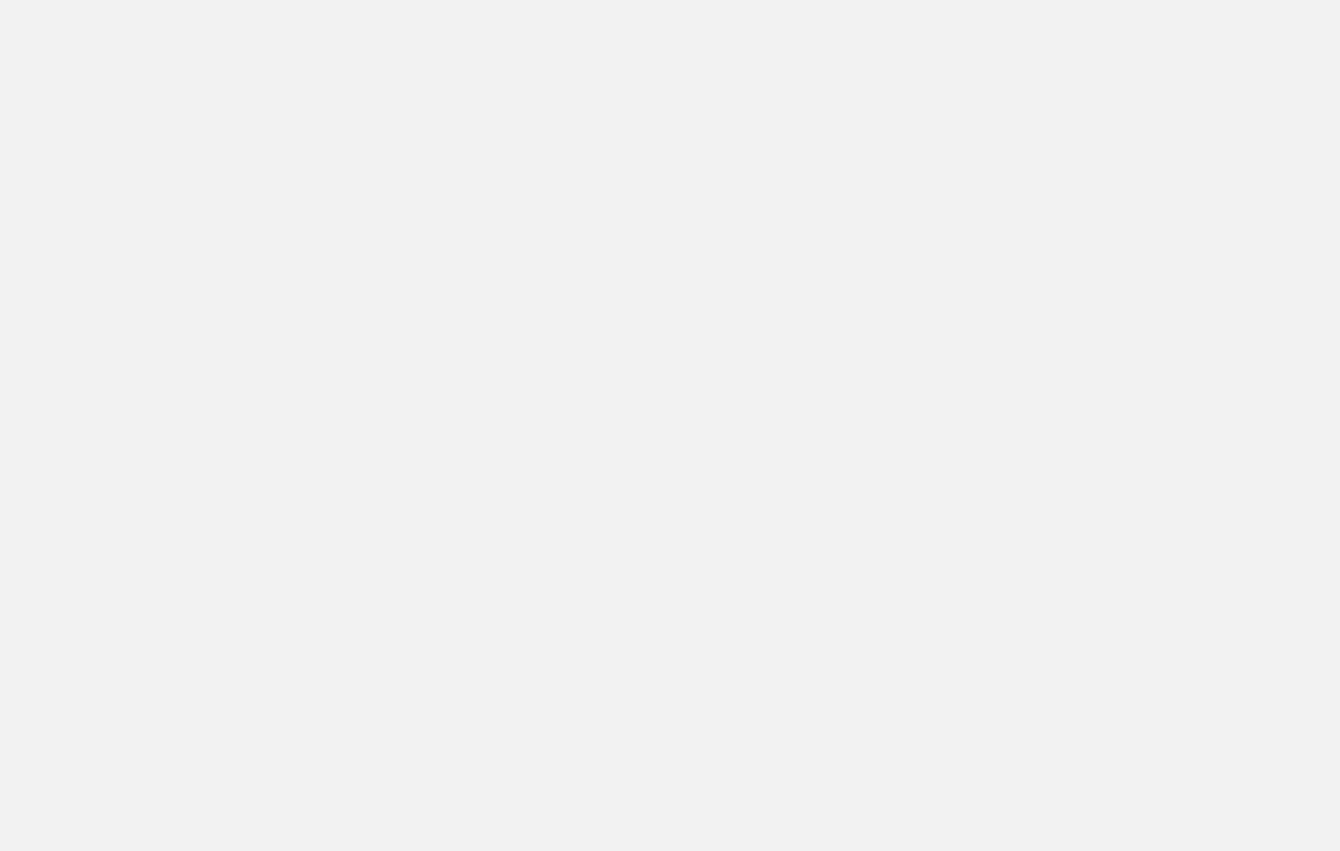


from keras.layers import Dropout

from keras.layers import Dense

The LSTM layer is added with the following arguments: 50 units is the dimensionality of the output space, return\_sequences=True is necessary for stacking LSTM layers so the consequent LSTM layer has a three-dimensional sequence input, and input\_shape is the shape of the training dataset.

Specifying 0.2 in the Dropout layer means that 20% of the layers will be dropped. Following the LSTM and Dropout layers, we add the Dense layer that specifies an output of one unit. To compile our model we use the Adam optimizer and set the loss as the mean\_squared\_error. After that, we fit the model to run for 100 epochs (the epochs are the number of times the learning algorithm will work through the entire training set) with a batch size of 32.



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))

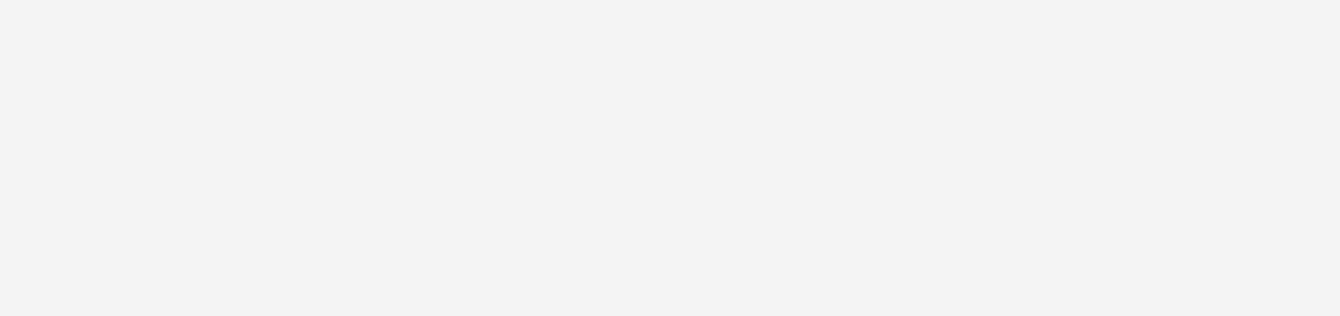
model.add(Dense(units=1))

model.compile(optimizer='adam',loss='mean\_squared\_error')

model.fit(X\_train,y\_train,epochs=100,batch\_size=32)

***Making Predictions on the Test Set***

We start off by importing the test set



url =

'https://raw.githubusercontent.com/mwitiderrick/stockprice/master/ta

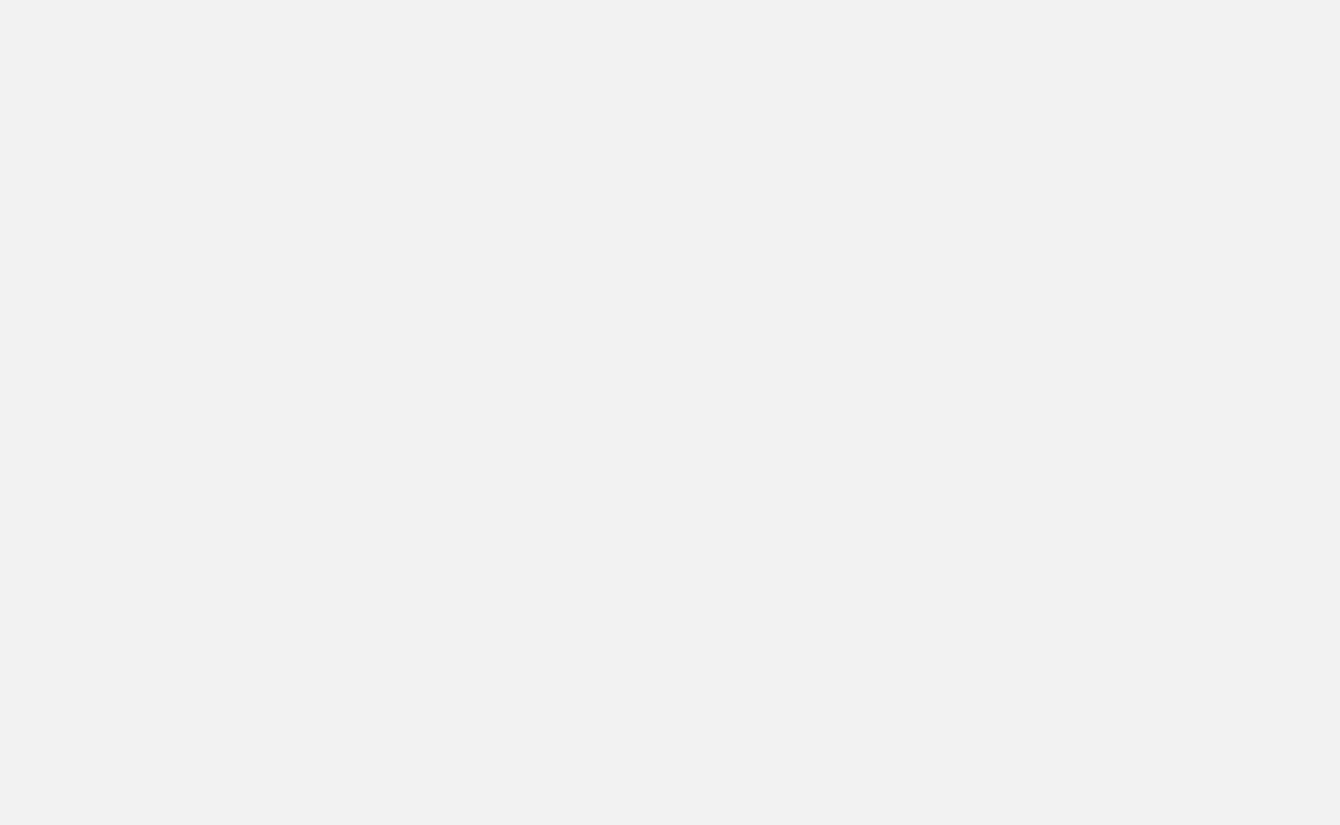


tatest.csv'

dataset\_test = pd.read\_csv(url)

real\_stock\_price = dataset\_test.iloc[:, 1:2].values

Before predicting future stock prices, we have to modify the test set (notice similarities to the edits we made to the training set): merge the training set and the test set on the 0 axis, set 60 as the time step again, use MinMaxScaler, and reshape data. Then, inverse\_transform puts the stock prices in a normal readable format.



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)

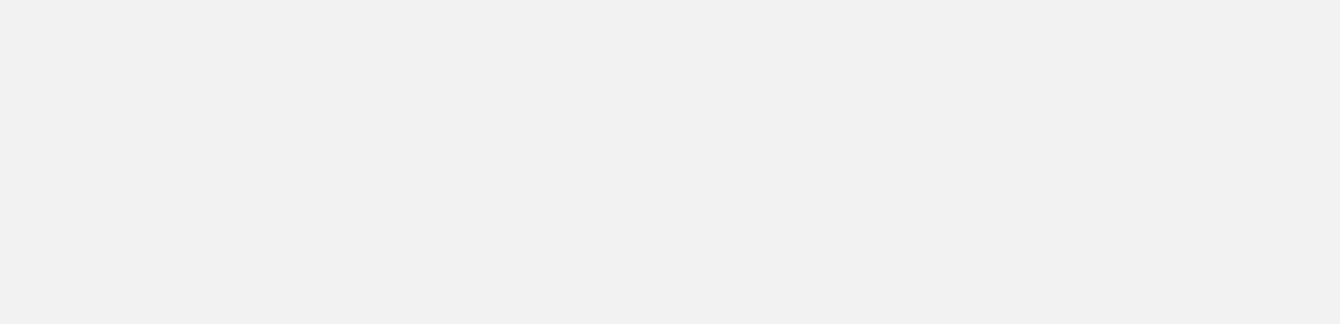
X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_stock\_price = model.predict(X\_test)

predicted\_stock\_price = sc.inverse\_transform(predicted\_stock\_price)

***Plotting the Results***

After all these steps, we can use matplotlib to visualize the result of our predicted stock price and the actual stock price.

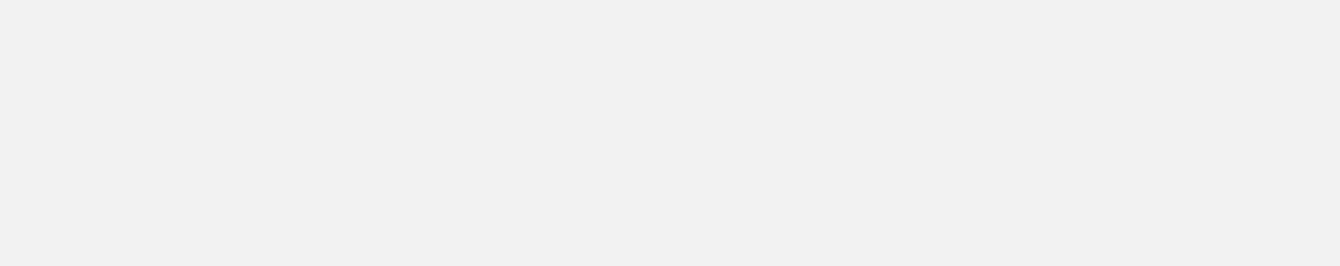


plt.plot(real\_stock\_price, color = 'black', label = 'TATA Stock Price')

plt.plot(predicted\_stock\_price, color = 'green', label = 'Predicted TATA Stock Price')

plt.title('TATA Stock Price Prediction')

plt.xlabel('Time')

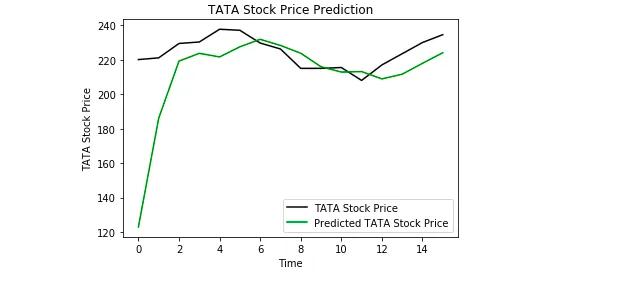


plt.ylabel('TATA Stock Price')



plt.legend()

plt.show()



While the exact price points from our predicted price weren’t always close to the actual price, our model did still indicate overall trends such as going up or down. This project teaches us the LSTMs can be somewhat effective in times series forecasting.



[*Follow*](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F4b91022e00a0&operation=register&redirect=https%3A%2F%2Ftowardsdatascience.com%2Fpredicting-stock-prices-using-a-keras-lstm-model-4225457f0233&user=Roshan+Adusumilli&userId=4b91022e00a0&source=post_page-4b91022e00a0----4225457f0233---------------------follow_profile-----------)