STOCK PRICE PREDICTION – AMERICAN AIRLINES

PART 4: RNN: LSTM with Time-Series Data

Summary of the Core Parameters

- *Data set used: American Airlines Inc.*
- *Percentage of the data used for testing: 10% of the data has been used.*
- *Number of LSTM Layers: Two Layer LSTM*
- *Number of Neurons: 50 Neurons in each LSTM Layer.*
- *Dropout Layers: Yes*
- *The percentage of dropout: 20%*
- *Model Used: LSTM Kera Seqential*
- *Number of epochs: 100*
- *Batch size for training: 32*

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Import Libraries

In [99]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
```

In [100]:
```
# For timeseries RNN LSTM neural networks
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
```

In [101]:
```
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.preprocessing import MinMaxScaler
```

Dataset American Airlines Inc (AAL: Jan 22, 2009 - Jan 22, 2012)

In [102]:
```
pwd
```

Out[102]:
```
'/Users/udaykumar/Downloads'
```

In [103]:
```
cd /Users/udaykumar/Downloads
```

```
/Users/udaykumar/Downloads
```

Load Dataset

In [104]:
```
df_ALL=pd.read_csv('AAL.csv')
```

In [105]:
```
df_ALL.head()
```

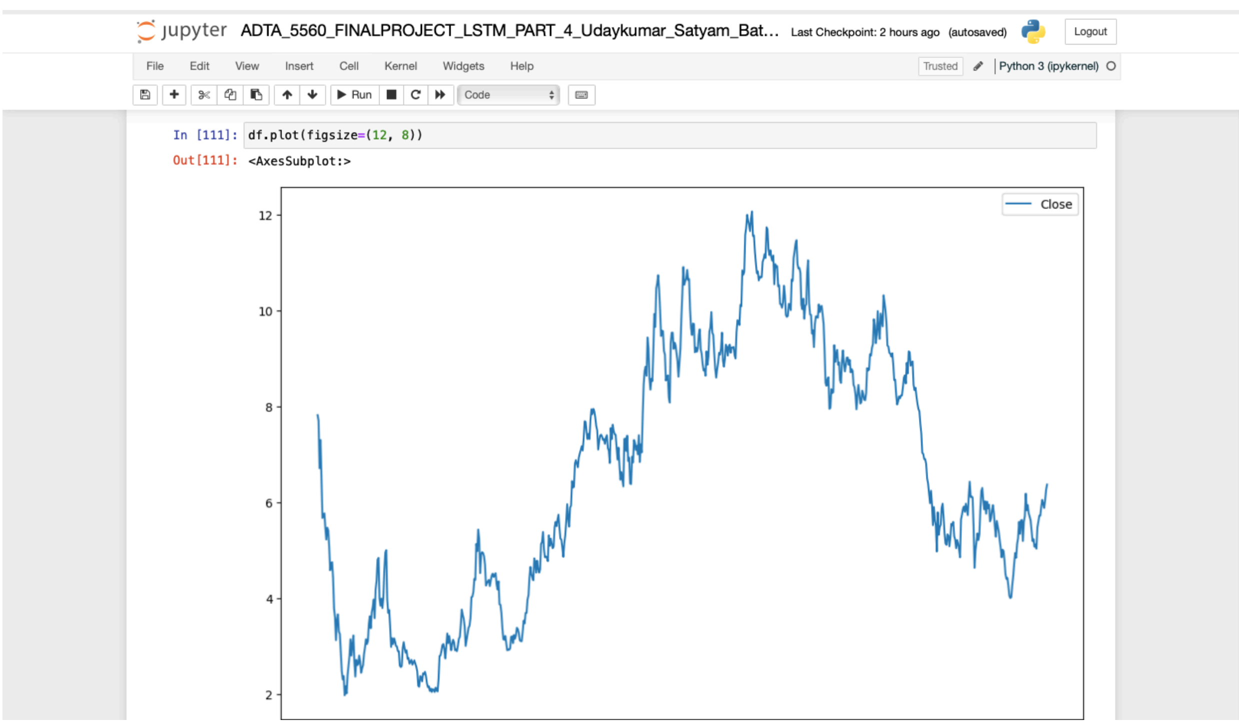
Out[105]:

Date Open High Low Close Adj Close Volume

I have selected the American Airlines Inc company dataset for this project. First, I have imported the necessary libraries for data analysis and visualization such as pandas and NumPy. Next, I have imported the required classes for creating a neural network with fully connected layers, dropout regularization, and LSTM layers.

Then, I imported classes for creating batches of time series data and scaling the data to a specific range. These classes can be used to prepare data for neural network training. I have downloaded the American Airlines dataset for the period of January 22, 2009, to January 22, 2012, from the yahoo finance website.

I also carried out a quick exploratory analysis. A line plot of the "Close" column in the DataFrame df is produced by Df. plot (fig size=(12, 8)) and displayed as a figure with dimensions of 12 inches in width and 8 inches in height. Additional MinMaxscaler is required for data scalability. For the basic RNN, we need data that is in chronological order. It is included in a time sequence. I'll use stock info in this situation. On the stock, a wealth of information is accessible.



jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Brief Exploratory Data Analysis (EDA)

In [106]:
```
df_ALL.shape
```
Out[106]:
```
(755, 7)
```

In [107]:
```
df_ALL.dtypes
```
Out[107]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

In [108]:
```
# Statistics Sumary
df_ALL.describe()
```
Out[108]:

	Open	High	Low	Close	Adj Close	Volume
count	755.000000	755.000000	755.000000	755.000000	755.000000	7.550000e+02
mean	6.618040	6.794053	6.422146	6.596026	6.218348	8.517628e+06
std	2.701919	2.729561	2.675576	2.699536	2.544964	5.097315e+06
min	2.000000	2.090000	1.880000	1.970000	1.857201	1.899200e+06
25%	4.415000	4.555000	4.315000	4.410000	4.157490	5.303200e+06
50%	6.280000	6.450000	6.060000	6.260000	5.901561	7.126600e+06
75%	9.040000	9.210000	8.845000	9.030000	8.512956	1.003475e+07
max	12.240000	12.260000	11.870000	12.070000	11.378889	5.235450e+07

Keep only "Close" (for closing price) and filter out all other attributes.

The shapes can be seen. There are 755 types and 7 functions available. There is a separate value for each characteristic. It is assumed that the capacity is a number. The minimum, maximum, and standard variation are all given in the statistical summary of the numerical traits.

We will deal with duration and previous data sets as we work on a time series. There is input a time sequence of 60 points. The information needs to be divided in half for training and testing purposes. There are 755 data elements in the collection. The data frame's capacity is that. The next stage is to locate a data point's index among the 755 data points that make up the entire collection. We must determine the index - the duration of the testing data to divide the information. The overall index of the data values was 679. To divide the info, we will use it locally. Thus, the index is 679.

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Time Series Dataset: Train / Test

```
In [114]: len(df)
Out[114]: 755

In [115]: test_precent = 0.1

In [116]: len(df)*test_precent
Out[116]: 75.5
```

Split Data --> Train / Test

```
In [117]: test_length = np.round(len(df)*test_precent)
          test_length
Out[117]: 76.0

In [118]: split_index = int(len(df) - test_length)
          split_index
Out[118]: 679

In [119]: data_train = df.iloc[: split_index]
          data_test = df.iloc[split_index - length60 :]

In [120]: data_train.head(5)
Out[120]:
```

	Close
0	7.82
1	7.70

Now it is necessary to display the training's most recent five records as well as the data needed to capture five records. Zero is the initial number and 4 is the end index. The final number is 755.

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Normalize Data Scale it into the[0, 1]

```
In [124]: scaler = MinMaxScaler()

In [125]: scaler.fit(data_train)
Out[125]: MinMaxScaler()

In [126]: normalized_train = scaler.transform(data_train)
          normalized_test = scaler.transform(data_test)
```

Create Timeseries Generator for training

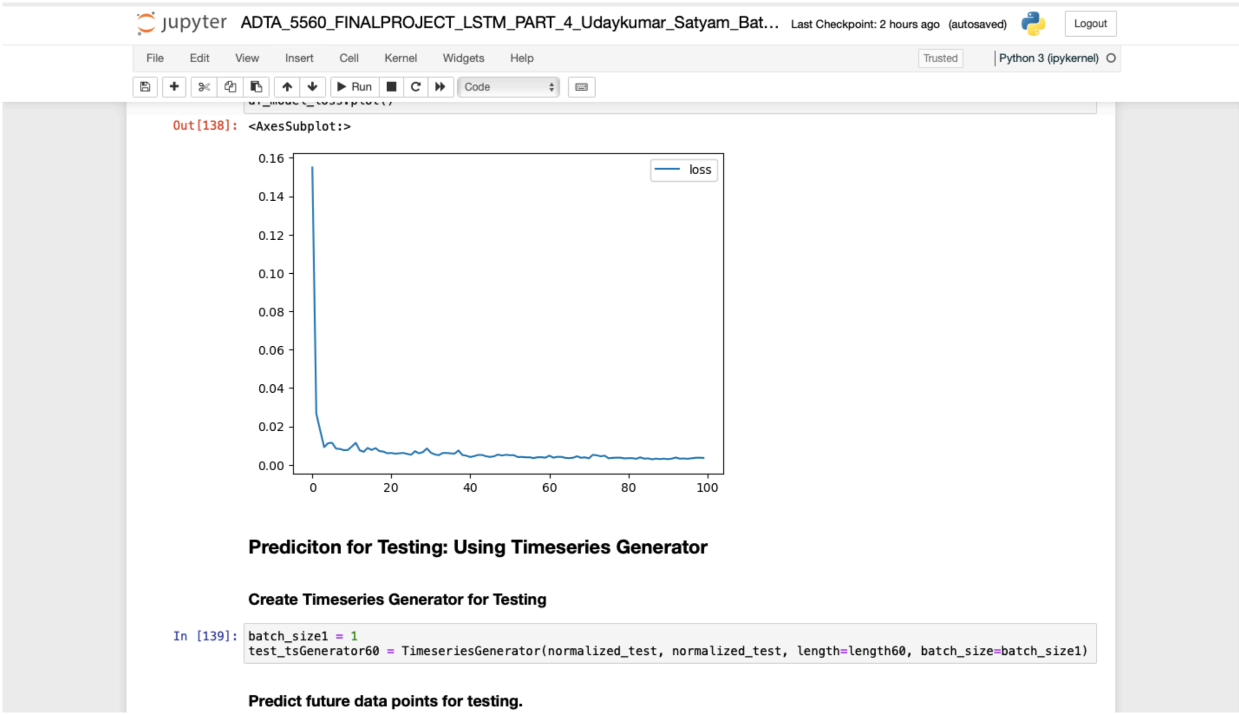
```
In [127]: ch_size32 = 32
          in_tsGenerator60 = TimeseriesGenerator(normalized_train, normalized_train, length=length60, batch_size=batch_size32)

In [128]: len(normalized_train)
Out[128]: 679

In [129]: len(train_tsGenerator60)
Out[129]: 20

In [130]: x,y = train_tsGenerator60[0]

In [131]: # print(x)

In [132]: # print(y)
```

Build. Train. and Test Model

The information now needs to be converted into a 0–1 number. The modal's efficiency can be improved. The Minmax scaler class and the utility will be used to complete it. Therefore, we will build up the model and normalize the data if the training collection of data matches the model. To teach the mode, we must develop a time series generator.

Despite the possibility of group processing, data is typically provided to the modal. Batch correction refers to sample correction. The incoming time series pattern in this instance serves as the sample. There are a total of 32 examples in the lot. A time series sequence input is this example. Batch creation directly using Keras is not a helpful tool.

The data elements' score is 60 points. Data points show a spectrum of 1 to 60. This enables the number at index 61 to be predicted. The overall number of time series input patterns ranges from 0 to 679. There are 60 data bits per cycle. There are 6 total groups, or 679/20. A lot has 20 examples in it. This series foretells the value that will come after it at index 60.



How many elements there will be specified in the project's stages. The number of traits or factors in the dataset is referred to as the number of features, and the final feature is 1. Before utilizing the simple RNN layer offered by Keras, we will first build an LSTM layer. This LSM neural network contains. I'll also employ a fresh layer failure. We use LSTM to match periods series. The result is generated using only this one data series. The stratum beneath receives the product.

Using keras, the neural networks will be built one at a time. A separate tier must be set for the return sequence to move into. Since the data is three-dimensional, figuring out its shape is the next step. We only bring up the two-dimensional, though. We must specify the variety of group amounts.

jupyter ADTA_5560_UdayKumarSatyam_Bathula_Assignment_4 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help   Trusted   Python 3 (ipykernel)

Compile Model

In [37]:
```
model.compile(optimizer='adam', loss='mse')
model.summary()
```

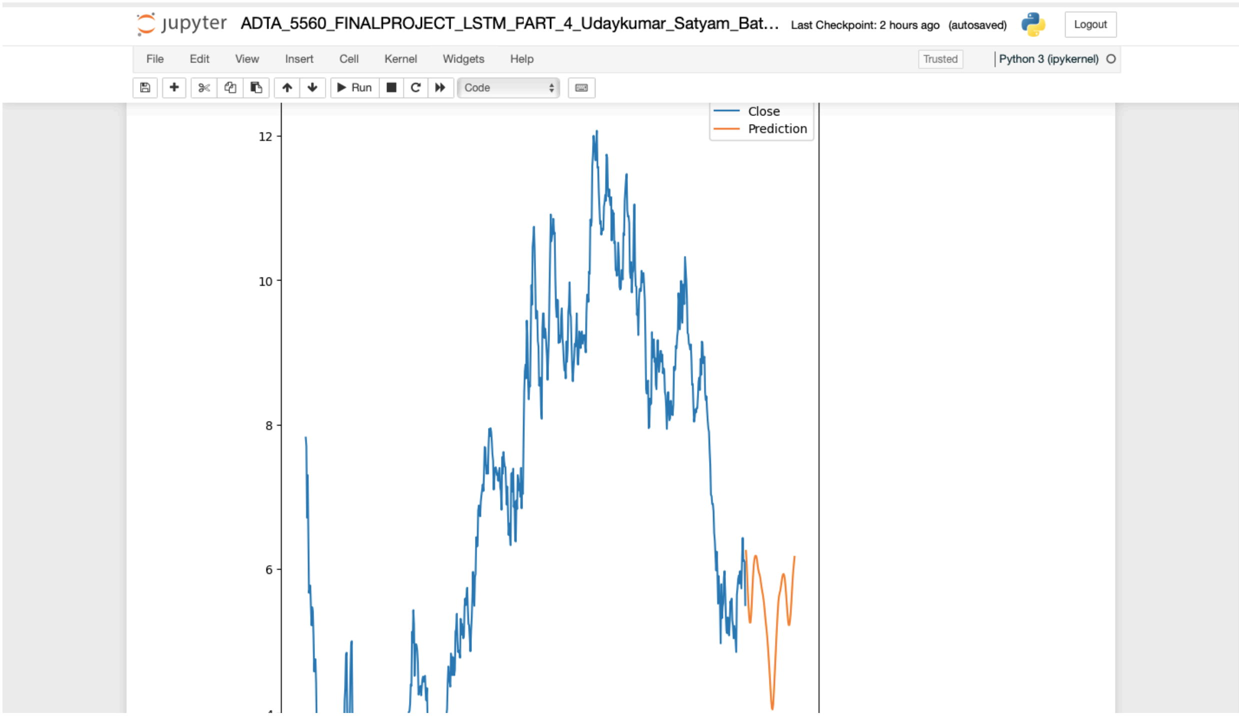
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 dropout (Dropout)           (None, 60, 50)            0

 lstm_1 (LSTM)               (None, 60, 50)            20200

 dropout_1 (Dropout)         (None, 60, 50)            0

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

In [38]:
```
model.fit_generator(train_tsGenerator60, epochs=100)
```

```
/var/folders/jp/xnmsl60d1bsf55jdzg5kjf5h0000gn/T/ipykernel_1283/1152258115.py:1: UserWarning: `Model.fit_generato
r` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(train_tsGenerator60, epochs=100)

Epoch 1/100
6/6 [==============================] - 12s 107ms/step - loss: 0.2408
Epoch 2/100
6/6 [==============================] - 1s 93ms/step - loss: 0.1151
Epoch 3/100
6/6 [==============================] - 1s 103ms/step - loss: 0.0418
Epoch 4/100
6/6 [==============================] - 1s 123ms/step - loss: 0.0368
```

It will be required to construct the mode. There are numerous varieties of optimizers. MSE will be employed for the last procedure. The model's six sections, which have two dropout degrees, were constructed one after the other. Its levels are all interconnected. To complement the option, we will provide another utility. Create the input sequence and the groups necessary to match the mode by using a straightforward RNN neural network.

The standard fit tool will be used. There will be 100 epoch ids displayed. The input sequence must support 6 groups for each stage. The six stages are being practiced. The input sequence will be run in 6 groups during the initial cycle, and the loss is 0.1550. Loss shrinks in scope.



The blue line displays the training data, and the orange line displays the expected number that the mode will produce.

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Timeseries Forecasting With LSTM

Overview

Timeseries Forecasting:

Forecast into the "Future" or unknown range.

Should employ all the available data.

i.e., not split the data into training/testing

Preprocess full input dataset

In [148]:
```
#Still use Minmax Scaler to normalize the full input dataset

full_scaler = MinMaxScaler()
normalized_full_data = full_scaler.fit_transform(df)
```

Create Timeseries Generator for Forecasting.

In [149]:
```
# Number of time steps of the input time series
# still use length60:
length60
```

Out[149]: 60

Time series projection into the future or unclear range. Clearing all the available info is necessary.
keeping the data separated into training and testing.

Using the forecast generator, we will construct, create, and fit an LSTM model for predicting with 100 epochs. For six stages, we will teach the data. There is a decline of 0.0013 for each stage. The original evaluation batch is typically 60 in duration. The current batch is used to predict and simulate the current outlook. Using the forecast add and an inverse transform, we will anticipate the capacity. The adjusted data will be converted back to true numbers using the inverse change.

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Build, Compile and Fit LSTM Model for Forecasting

In [151]:
```
# Train/Fit LSTM model
# In forecasting, we don't valiate, only FORECAST
# So, not need EarlyStop: Not need validation_generator

model.fit_generator(forecast_tsGenerator, epochs = 100)
```

```
Epoch 1/100
/var/folders/jp/xnmsl60d1bsf55jdzg5kjf5h0000gn/T/ipykernel_1257/1181600291.py:5: UserWarning: `Model.fit_generato
r` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(forecast_tsGenerator, epochs = 100)
22/22 [==============================] - 2s 93ms/step - loss: 0.0042
Epoch 2/100
22/22 [==============================] - 2s 86ms/step - loss: 0.0029
Epoch 3/100
22/22 [==============================] - 2s 79ms/step - loss: 0.0030
Epoch 4/100
22/22 [==============================] - 2s 79ms/step - loss: 0.0030
Epoch 5/100
22/22 [==============================] - 2s 74ms/step - loss: 0.0033
Epoch 6/100
22/22 [==============================] - 2s 74ms/step - loss: 0.0033
Epoch 7/100
22/22 [==============================] - 2s 74ms/step - loss: 0.0029
Epoch 8/100
```

Timeseries Forecasting Using LSTM Neural Network

In [152]:
```
forecast = []
# Set periods with the time span of forecasting
# Let's forecast for the next 102 business days
periods = 102

first_eval_batch = normalized_full_data[-length60:]
current_batch = first_eval_batch.reshape((1, length60, n_features))
```

Creating a new time stamp index with pandas

In [155]:
```
#calculate forecast index

forecast_index = np.arange(755,857,step=1)
```

In [156]:
```
forecast_df=pd.DataFrame(data=forecast,index=forecast_index,columns=['Forecast'])
```

In [157]:
```
forecast_df
```

Out[157]:

	Forecast
755	6.009705
756	5.995834
757	5.935212
758	5.857952
759	5.776410
...	...
852	3.262450
853	3.234271
854	3.211408
855	3.192760
856	3.177310

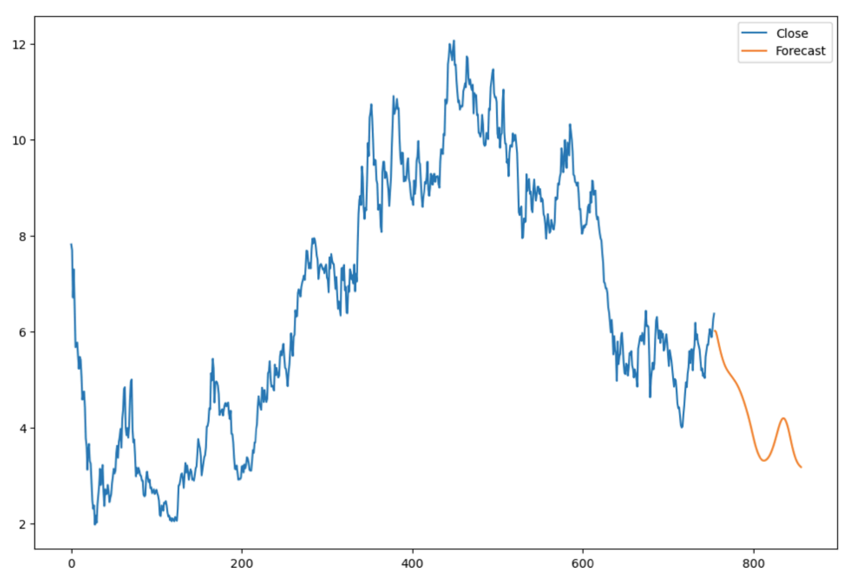
102 rows × 1 columns

In [158]:
```
df.plot()
forecast_df.plot()
```

Out[158]: <AxesSubplot:>

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Out[159]: <AxesSubplot:>



From now on, we will only have the forecast. Below are the numbers that the modal projected. I'll make a fresh box with the final price in it.

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
3.66071895, 3.58199752, 3.50976572, 3.44543421, 3.38855419,
3.33911588, 3.29718263, 3.26245045, 3.2342705 , 3.21140824,
3.19276017, 3.17730981])
```

In [166]:
```
df_JAN_JUN_2012['Forecast'] = forecast_df['Forecast'].values
```

In [167]:
```
df_JAN_JUN_2012
```

Out[167]:

	Close	Forecast
0	6.26	6.009705
1	6.41	5.995834
2	7.52	5.935212
3	7.85	5.857952
4	8.18	5.776410
...	...	...
97	11.36	3.262450
98	11.43	3.234271
99	11.97	3.211408
100	12.48	3.192760
101	12.03	3.177310

102 rows × 2 columns

In [168]:
```
df_JAN_JUN_2012.plot()
```

Out[168]: <AxesSubplot:>

jupyter ADTA_5560_FINALPROJECT_LSTM_PART_4_Udaykumar_Satyam_Bat... Last Checkpoint: 2 hours ago (autosaved) Logout

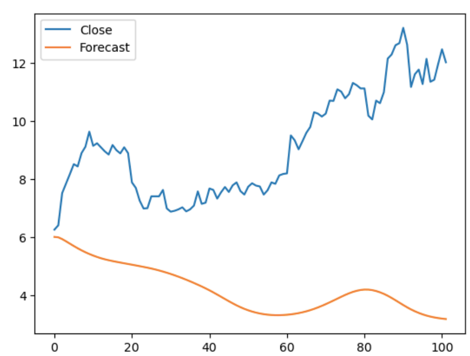
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

101	12.03	3.177310

102 rows × 2 columns

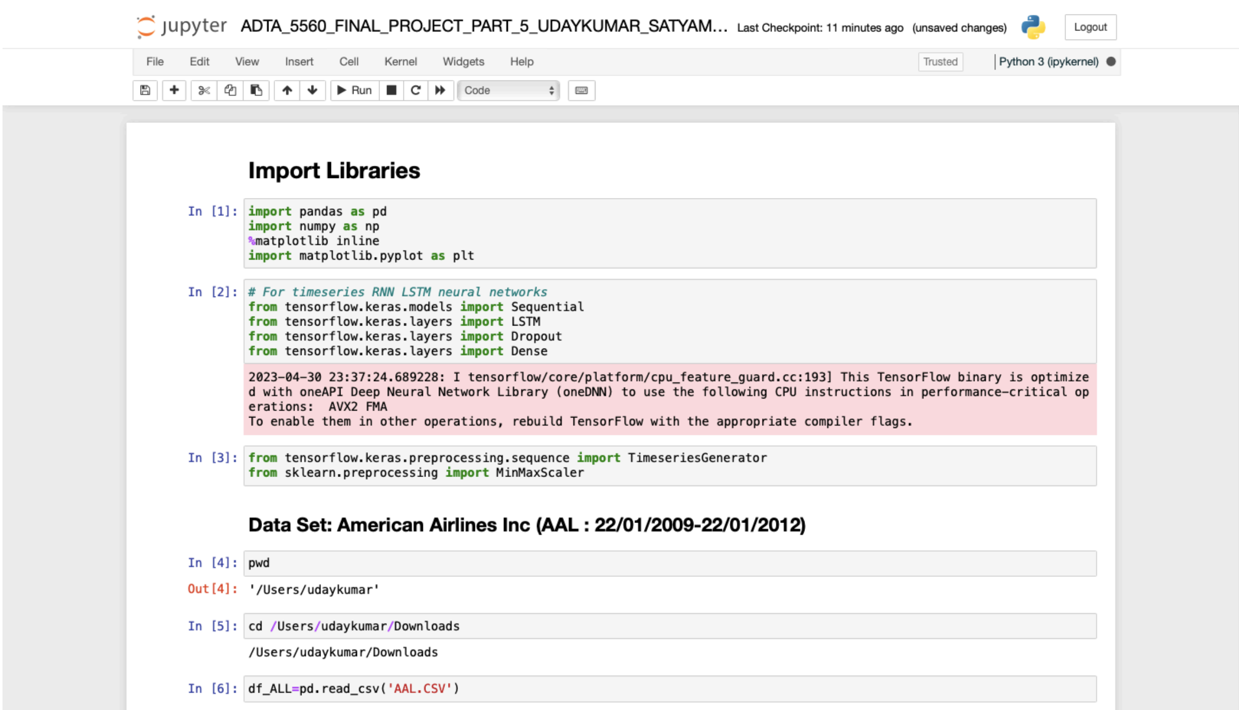
In [168]:
```
df_JAN_JUN_2012.plot()
```

Out[168]: <AxesSubplot:>



The final price of the American Airlines shares will be provided. The prediction is represented by an orange line. If we try it, it fails to predict the actual one. during the option test. The information for the new ending price will come from the datasets. The new value's entry pattern serves as the foundation for the modal. The 102 most recent data values will be the only ones included in the new input number. Accurate predicting values can be created, then anticipated.

PART 5: Redesign the Neural Network.



For this part I have used the same dataset 'American Airlines Inc' for the period January 22, 2009, to January 22, 2012, from the yahoo finance website. First, I have imported the necessary libraries for data analysis and visualization such as pandas and NumPy. Next, I have imported the required classes for creating a neural network with fully connected layers, dropout regularization, and LSTM.

jupyter ADTA_5560_FINAL_PROJECT_PART_5_UDAYKUMAR_SATYAM_BAT... Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

4 2009-01-29 7.15 7.75 6.33 6.47 6.099537 9766200

Brief Exploratory Data Analysis (EDA)

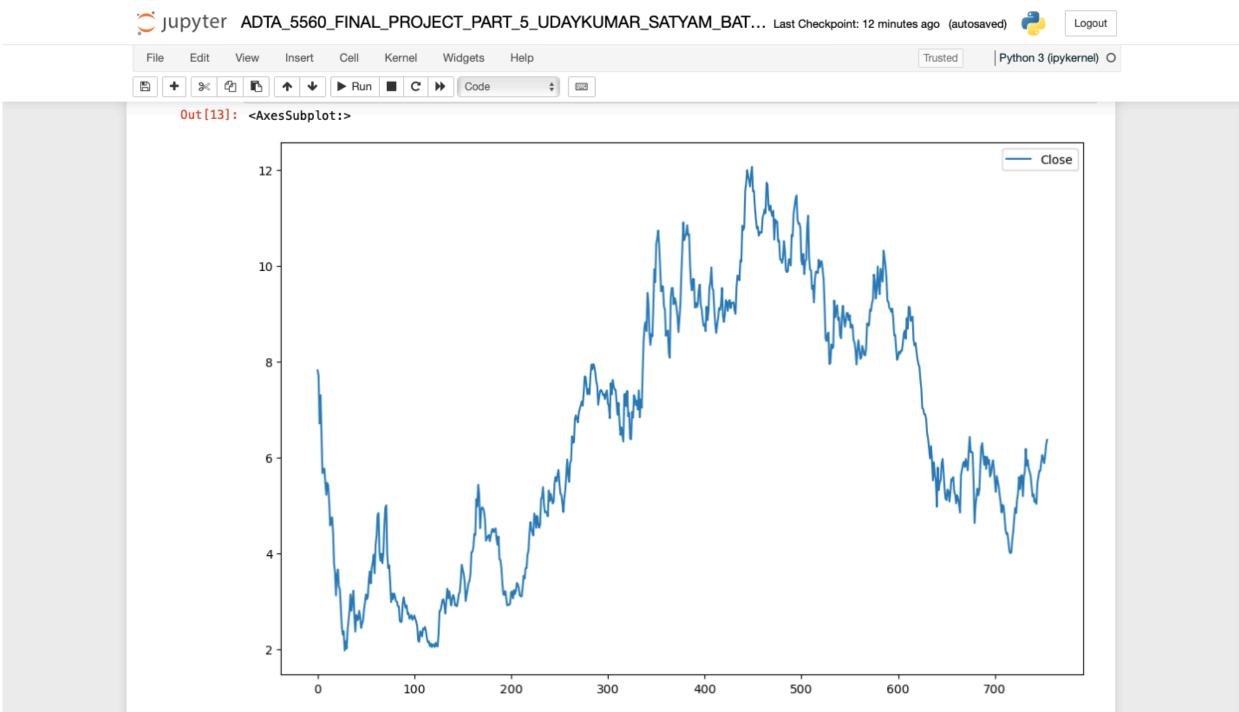
In [8]:
```
df_ALL.shape
```
Out[8]:
```
(755, 7)
```

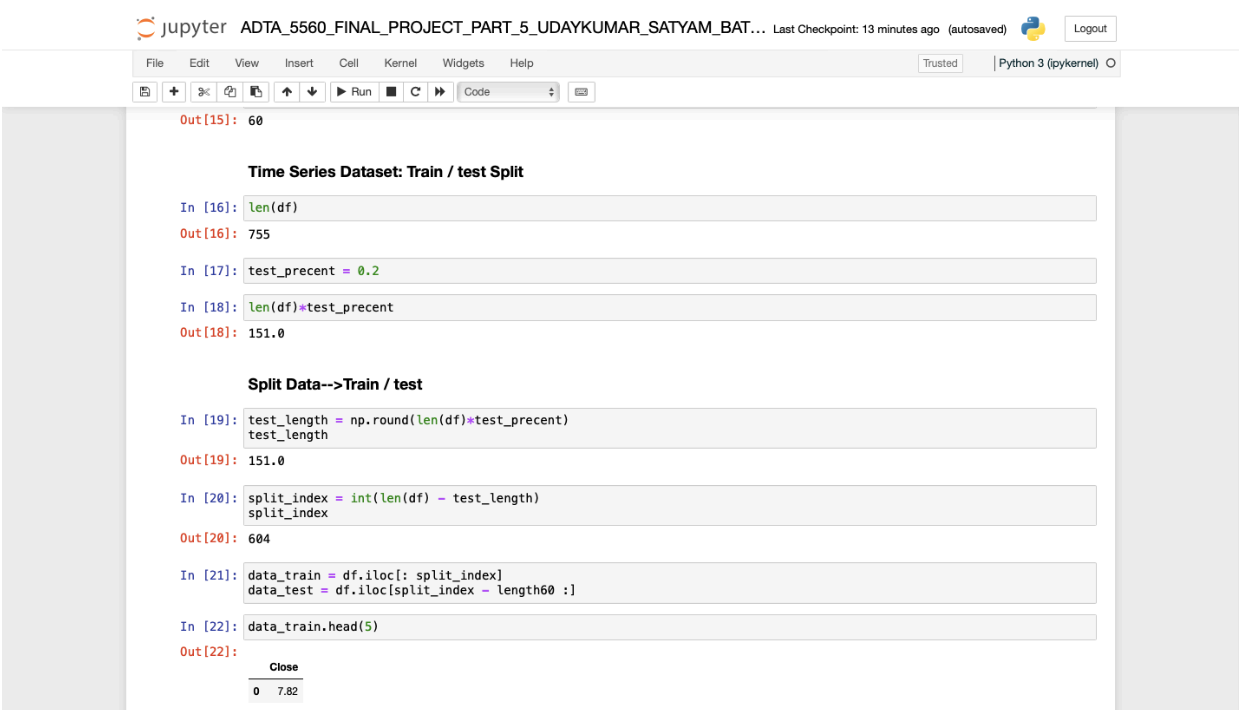
In [9]:
```
df_ALL.dtypes
```
Out[9]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
Volume         int64
dtype: object
```

In [10]:
```
# Statistics Sumary
df_ALL.describe()
```
Out[10]:

	Open	High	Low	Close	Adj Close	Volume
count	755.000000	755.000000	755.000000	755.000000	755.000000	7.550000e+02
mean	6.618040	6.794053	6.422146	6.596026	6.218348	8.517628e+06
std	2.701919	2.729561	2.675576	2.699536	2.544964	5.097315e+06
min	2.000000	2.090000	1.880000	1.970000	1.857201	1.899200e+06
25%	4.415000	4.555000	4.315000	4.410000	4.157490	5.303200e+06
50%	6.280000	6.450000	6.060000	6.260000	5.901561	7.126600e+06
75%	9.040000	9.210000	8.845000	9.030000	8.512956	1.003475e+07
max	12.240000	12.260000	11.870000	12.070000	11.378889	5.235450e+07

I also carried out a quick exploratory analysis. A line plot of the "Close" column in the DataFrame df is produced by Df. plot (fig size= (12, 8)) and displayed as a figure with dimensions of 12 inches in width and 8 inches in height. Additional MinMaxscaler is required for data scalability. For the basic RNN, we need data that is in chronological order. It is included in a time sequence. I'll use stock info in this situation. On the stock, a wealth of information is accessible.






```
Out[15]: 60
```

Time Series Dataset: Train / test Split

```
In [16]: len(df)
Out[16]: 755

In [17]: test_precent = 0.2

In [18]: len(df)*test_precent
Out[18]: 151.0
```

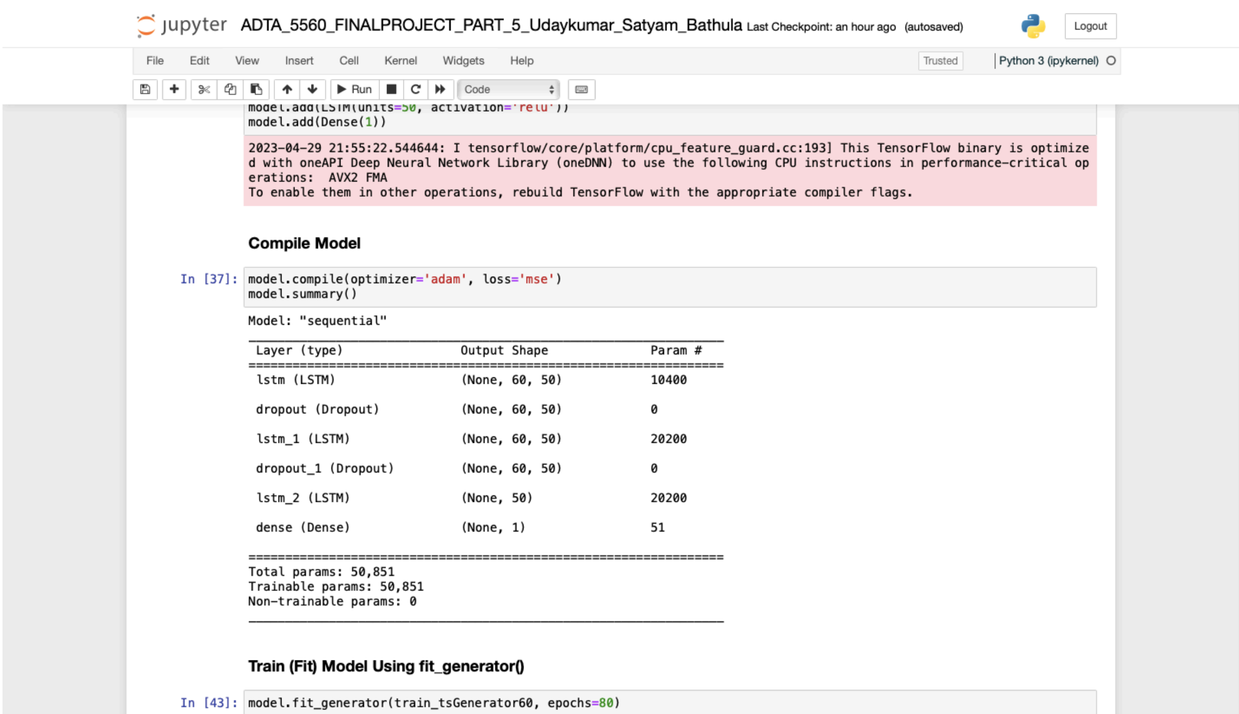
Split Data-->Train / test

```
In [19]: test_length = np.round(len(df)*test_precent)
         test_length
Out[19]: 151.0

In [20]: split_index = int(len(df) - test_length)
         split_index
Out[20]: 604

In [21]: data_train = df.iloc[: split_index]
         data_test = df.iloc[split_index - length60 :]

In [22]: data_train.head(5)
Out[22]:
```

	Close
0	7.82
1	7.70

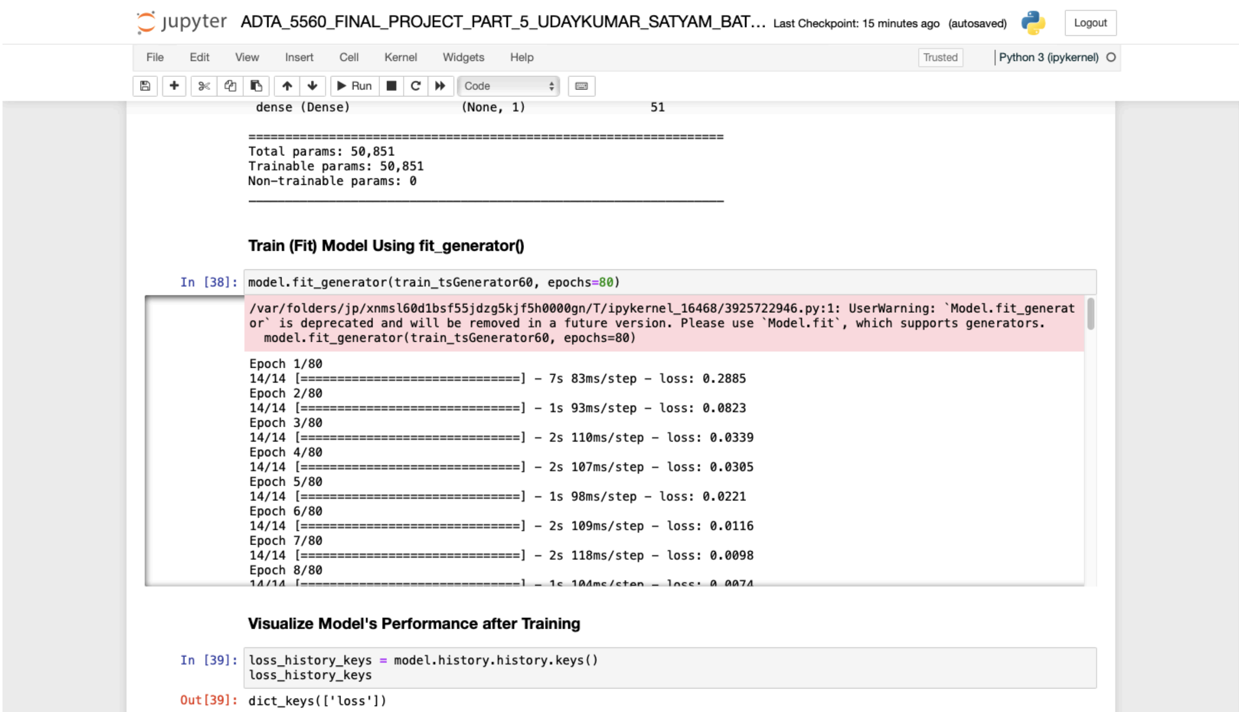
How many elements there will be specified in the project's stages. The number of traits or factors in the dataset is referred to as the number of features, and the final feature is 1. Before utilizing the simple RNN layer offered by Keras, we will first build an LSTM layer. This LSM neural network contains. I'll also employ a fresh layer failure. We use LSTM to match periods series. The result is generated using only this one data series. The stratum beneath receives the product.

Using keras, the neural networks will be built one at a time. A separate tier must be set for the return sequence to move into. Since the data is three-dimensional, figuring out its shape is the next step. We only bring up the two-dimensional, though. We must specify the variety of group amounts.



It will be required to construct the mode. There are numerous varieties of optimizers. MSE will be employed for the last procedure. The model's six sections, which have two dropout degrees, were constructed one after the other. Its levels are all interconnected. To complement the option, we will provide another utility. Create the input sequence and the groups necessary to match the mode by using a straightforward RNN neural network.

The standard fit tool will be used. There will be 80 epoch ids displayed. The input sequence must support 14 groups for each stage. The six stages are being practiced. The input sequence will be run in 4 groups during the initial cycle. Loss shrinks in scope.



Time series projection into the future or unclear range. Clearing all the available info is necessary.
keeping the data separated into training and testing.

Using the forecast generator, we will construct, create, and fit an LSTM model for predicting with 20 epochs. For six stages, we will teach the data. There is a decline of 0.0005 for each stage. The original evaluation batch is typically 40 in duration. The current batch is used to predict and simulate the current outlook. Using the forecast add and an inverse transform, we will anticipate the capacity. The adjusted data will be converted back to true numbers using the inverse change.

jupyter ADTA_5560_FINAL_PROJECT_PART_5_UDAYKUMAR_SATYAM... Last Checkpoint: 21 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Build, Compile, and Fit LSTM model for Forecasting

In [53]:
```
# Train/Fit LSTM model
# In forecasting, we don't validate, only FORECAST
# So, not need EarlyStop: Not need validation_generator

model.fit_generator(forecast_tsGenerator, epochs = 20)
```

```
Epoch 1/20
 1/18 [>.............................] - ETA: 2s - loss: 0.0030
```

```
/var/folders/jp/xnmsl60d1bsf55jdzg5kjf5h0000gn/T/ipykernel_16468/164812513.py:5: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(forecast_tsGenerator, epochs = 20)
```

```
18/18 [==============================] - 2s 96ms/step - loss: 0.0045
Epoch 2/20
18/18 [==============================] - 2s 86ms/step - loss: 0.0039
Epoch 3/20
18/18 [==============================] - 2s 127ms/step - loss: 0.0047
Epoch 4/20
18/18 [==============================] - 2s 100ms/step - loss: 0.0040
Epoch 5/20
18/18 [==============================] - 2s 98ms/step - loss: 0.0038
Epoch 6/20
18/18 [==============================] - 1s 81ms/step - loss: 0.0032
Epoch 7/20
18/18 [==============================] - 1s 80ms/step - loss: 0.0034
Epoch 8/20
18/18 [==============================] - 2s 91ms/step - loss: 0.0038
Epoch 9/20
18/18 [==============================] - 2s 89ms/step - loss: 0.0034
Epoch 10/20
18/18 [==============================] - 1s 78ms/step - loss: 0.0035
Epoch 11/20
18/18 [==============================] - 1s 80ms/step - loss: 0.0037
Epoch 12/20
18/18 [==============================] - 1s 81ms/step - loss: 0.0033
Epoch 13/20
18/18 [==============================] - 2s 100ms/step - loss: 0.0034
```

To boost the amount of time-series data available for training and testing while also increasing the model's accuracy in the Long-Short-Term Memory (LSTM) algorithm. It is an essential tactic for getting more accurate results. It consists of four layers that work together to create both the cell's output and its state. These two elements are subsequently transferred to the following concealed layer.

In contrast to RNNs, which only have one tanh layer, LSTMs contain three logistic sigmoid gates and one layer. 50 neurons have been considered for the LSTM layer. A time series input sequence makes up this sample. By using Keras, manually creating batches is not a practical tool. The data points have a sixty-point index. Data points show a range from 1 to 60. Based on this, the value at index 61 may be predicted. Each batch has 35 samples. To forecast the subsequent value at index sixty, this sequence is employed. For time series, we utilized 80 epochs, while for forecasting, we used 20 epochs.

jupyter ADTA_5560_FINAL_PROJECT_PART_5_UDAYKUMAR_SATYAM_BATH… Last Checkpoint: an hour ago (autosaved)

Logout

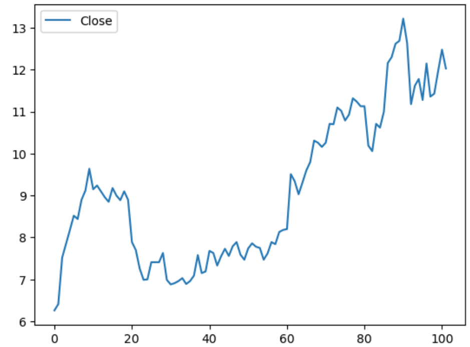
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [66]:
```
df_JAN_JUN_2012.plot()
```

Out[66]: <AxesSubplot:>



In [67]:
```
forecast_df['Forecast'].values
```

Out[67]:
```
array([5.92143912, 5.96748091, 5.97616276, 5.95506783, 5.91431889,
       5.86226425, 5.80472022, 5.74537347, 5.68648516, 5.62934868,
       5.57465635, 5.52274981, 5.47375849, 5.42766523, 5.38435173,
       5.34360821, 5.30515803, 5.26867912, 5.2338178 , 5.20020595,
       5.16747754, 5.13528432, 5.10288852, 5.06983443, 5.03608833,
       5.00150514, 4.96592021, 4.92918695, 4.89123011, 4.85203435,
       4.81162554, 4.77006208, 4.72737647, 4.68284963, 4.63690597,
       4.58978749, 4.54164771, 4.49259814, 4.44273903, 4.39211958,
       4.34034713, 4.28720573, 4.23240474, 4.17484279, 4.11450876,
```

jupyter ADTA_5560_FINAL_PROJECT_PART_5_UDAYKUMAR_SATYAM_BATH… Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
       2.23452827, 2.22123615, 2.21082502, 2.20281626, 2.19680076,
       2.19242778, 2.18941756, 2.18750034, 2.18645127, 2.18607781,
       2.1862165 , 2.18673312, 2.18751575, 2.18847523, 2.1895446 ,
       2.19067197, 2.19180489])
```

In [68]:
```
df_JAN_JUN_2012['Forecast'] = forecast_df['Forecast'].values
```

In [69]:
```
df_JAN_JUN_2012
```

Out[69]:

	Close	Forecast
0	6.26	5.921439
1	6.41	5.967481
2	7.52	5.976163
3	7.85	5.955068
4	8.18	5.914319
...	...	...
97	11.36	2.187516
98	11.43	2.188475
99	11.97	2.189545
100	12.48	2.190672
101	12.03	2.191805

102 rows × 2 columns

In [70]:
```
df_JAN_JUN_2012.plot()
```

Out[70]: <AxesSubplot:>

jupyter ADTA_5560_FINAL_PROJECT_PART_5_UDAYKUMAR_SATYAM_BAT... Last Checkpoint: 44 minutes ago (autosaved) Logout

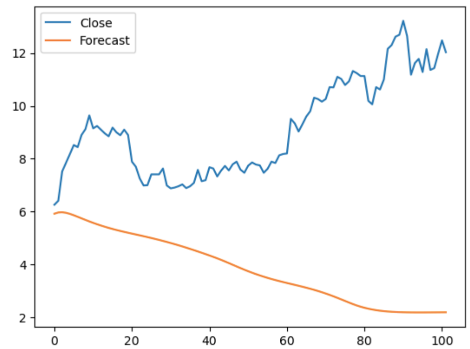
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

99	11.97	2.189545
100	12.48	2.190672
101	12.03	2.191805

102 rows × 2 columns

In [70]:
```
df_JAN_JUN_2012.plot()
```

Out[70]: <AxesSubplot:>



In [ ]:

- *Dataset used: American Airlines Inc*
- *Percentage for testing: 20%*
- *Number of layers of LSTM: 2*
- *Number of neurons in LSTM layer: 50*
- *Batch size for training: 40*
- *Dropout: Yes*
- *Percentage to dropout: 20%*
- *Batch Size: 1*
- *Number of epochs: 80*
- *Model Used: LSTM Kera Sequential*