Replicated Financial Data Time Series Forecasting Code

Data Loading

In [1]:
```
import numpy as np
import pandas as pd

data = pd.read_csv("https://raw.githubusercontent.com/ritvik02/Financial-Data-Time-Series-Forecasting-Using-Neural-Networks/main/
```

In [2]:
```
data.head()
```

Out[2]:

	Unnamed: 0	A	AAL	AAP	AAPL	ABBV	ABC	ABMD	ABT	ACN	...	XLNX	XOM	XRAY	XRX	XYL	YUM	ZBH	ZION	ZTS
0	01/01/2015	40.94	53.630	159.28	110.38	65.44	90.16	38.06	45.02	89.31	...	43.290	92.45	53.27	36.5155	38.07	52.3831	113.42	28.510	43.03
1	02/01/2015	40.56	53.910	158.56	109.33	65.89	90.46	37.31	44.90	88.84	...	43.600	92.83	51.93	36.2257	38.08	52.0236	112.59	28.290	43.31
2	05/01/2015	39.80	53.875	156.47	106.25	64.65	89.69	37.07	44.91	87.34	...	42.795	90.29	51.57	35.4353	35.71	50.9666	116.79	27.230	43.05
3	06/01/2015	39.18	53.040	156.36	106.26	64.33	90.18	36.13	44.40	86.71	...	42.180	89.81	50.93	34.9611	35.50	50.3410	115.80	26.190	42.63
4	07/01/2015	39.70	53.010	159.72	107.75	66.93	91.98	37.28	44.76	88.53	...	42.195	90.72	52.25	35.4089	35.78	52.0092	118.68	26.435	43.51

5 rows × 507 columns

In [3]:
```
data.columns
```

Out[3]:
```
Index(['Unnamed: 0', 'A ', 'AAL ', 'AAP ', 'AAPL ', 'ABBV ', 'ABC ', 'ABMD ',
       'ABT ', 'ACN ',
       ...
       'XLNX ', 'XOM ', 'XRAY ', 'XRX ', 'XYL ', 'YUM ', 'ZBH ', 'ZION ',
       'ZTS ', 'Cash'],
      dtype='object', length=507)
```

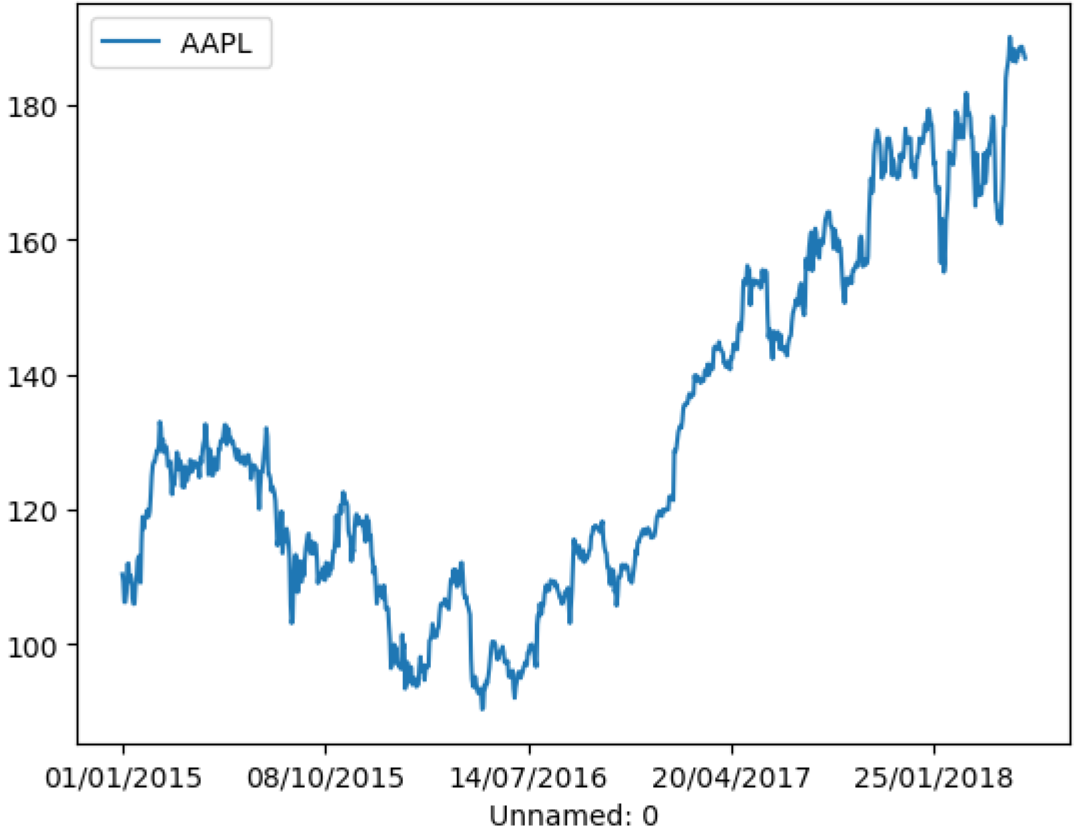
In [4]:
```
# sum of null values in the data
data.isna().sum()
```

Out[4]:
```
Unnamed: 0    0
A             0
AAL           0
AAP           0
AAPL          0
             ..
YUM           0
ZBH           0
ZION          0
ZTS           0
Cash          0
Length: 507, dtype: int64
```

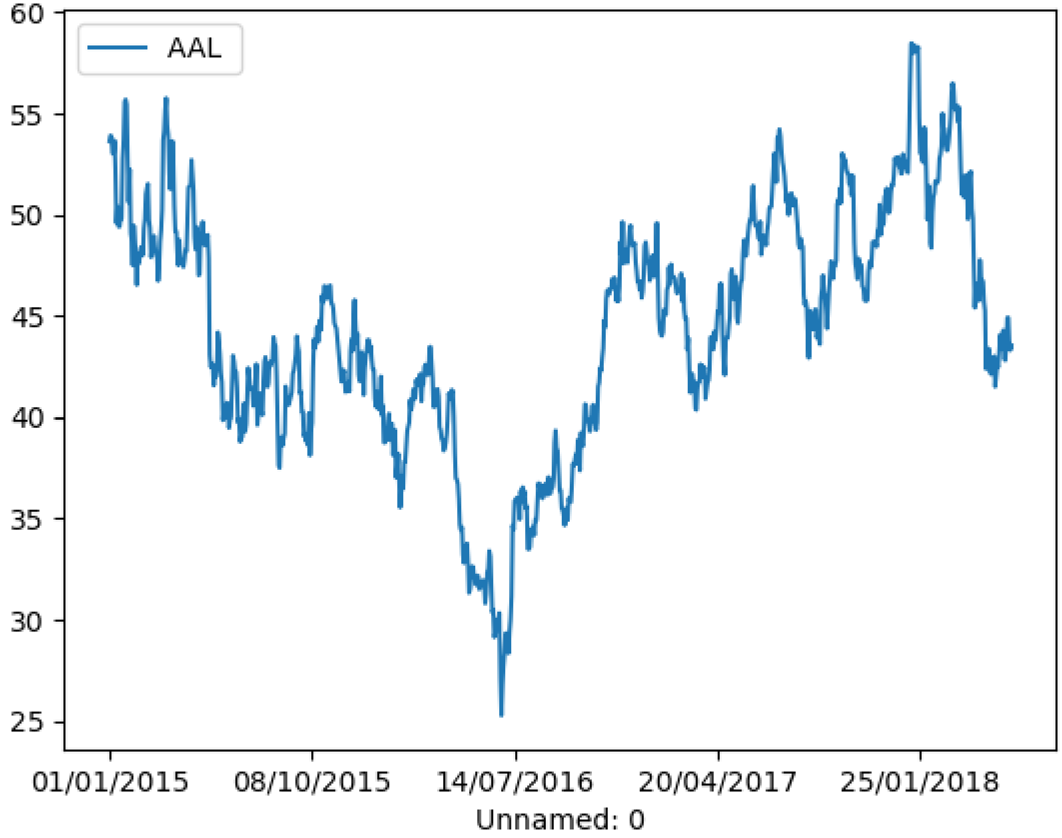
In [5]:
```
# # converting date column from object to datetime
# data['Unnamed: 0'] = pd.to_datetime(data['Unnamed: 0'])
```

Trend Analysis

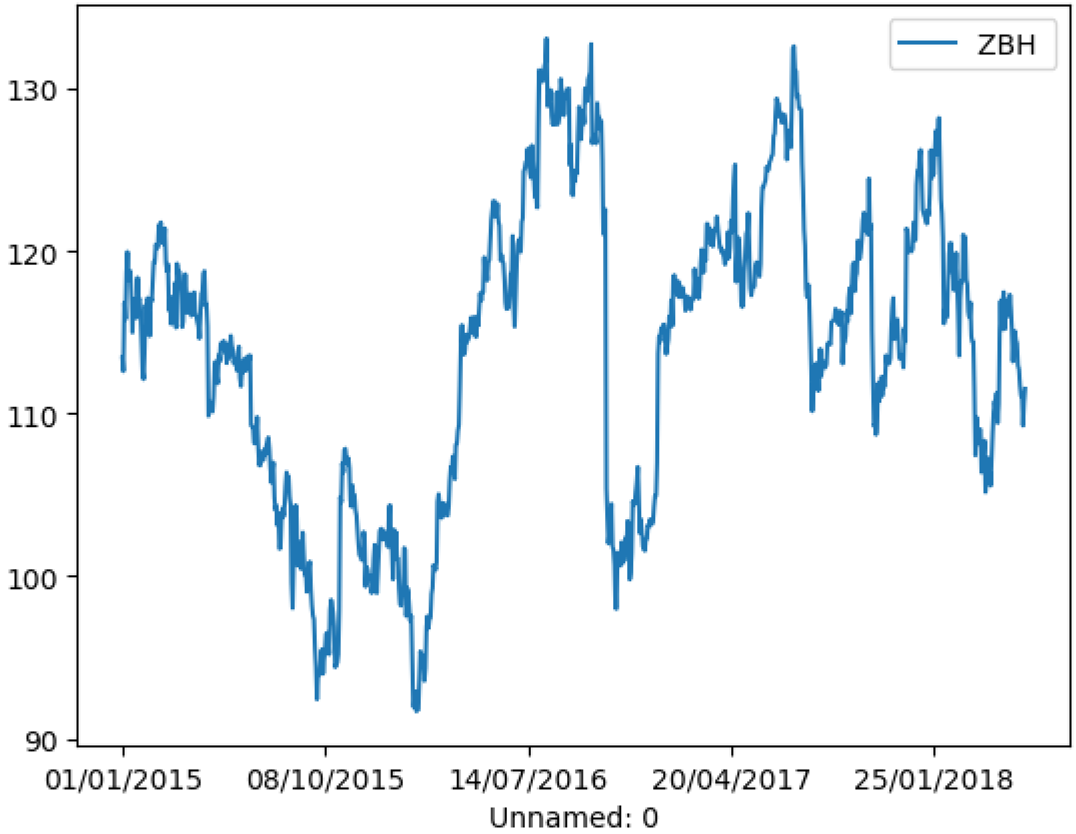
In [6]:
```
import matplotlib.pyplot as plt

# using matplotlib to plot the data
data.plot.line(x = 'Unnamed: 0', y = 'AAPL ')
plt.show()
```



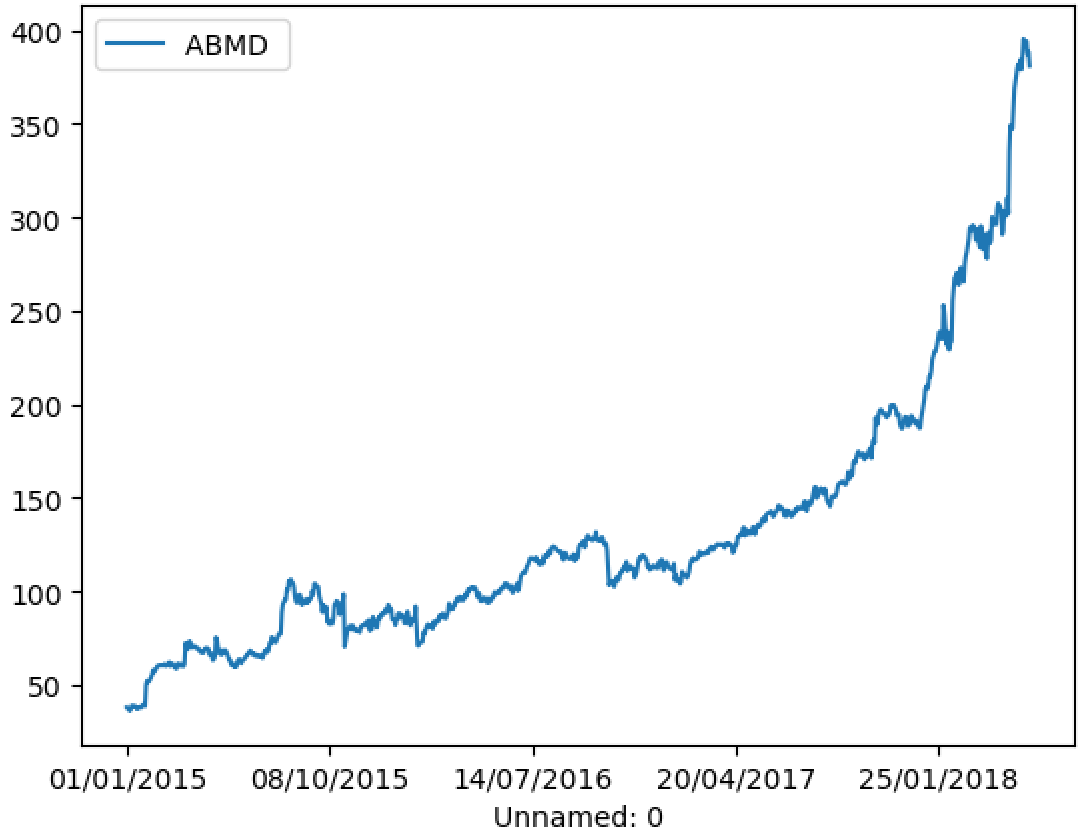
In [7]:
```
# using matplotlib to plot the data
data.plot.line(x = 'Unnamed: 0', y = 'AAL ')
plt.show()
```



In [8]:
```
# using matplotlib to plot the data
data.plot.line(x = 'Unnamed: 0', y = 'ZBH ')
plt.show()
```



In [9]:
```
# using matplotlib to plot the data
data.plot.line(x = 'Unnamed: 0', y = 'ABMD ')
plt.show()
```



In [10]:
```
# using matplotlib to plot the data
data.plot.line(x = 'Unnamed: 0', y = 'YUM ')
plt.show()
```

Selecting stock

In [11]:
```
data_stock = data['AAPL ']
```

In [12]:
```
data_stock.shape
```

Out[12]:
```
(891,)
```

In [13]:
```
891 - 178
```

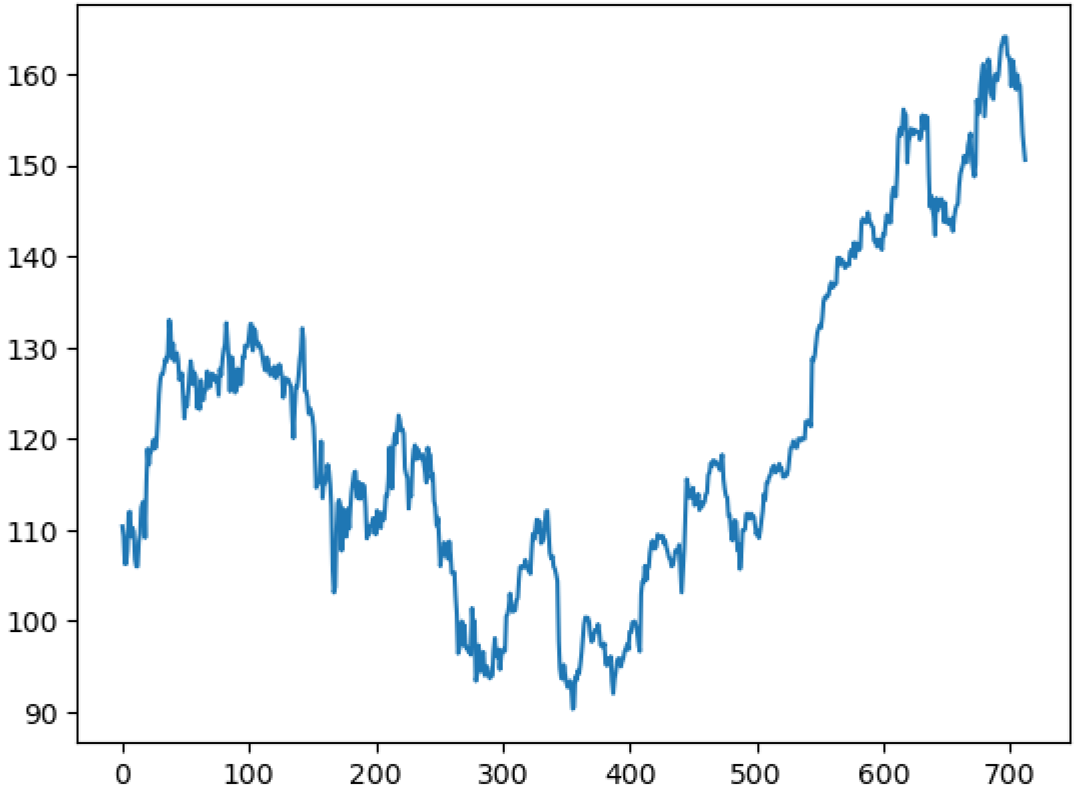
Out[13]:
```
713
```

In [14]:
```
data_stock_np = np.array(data_stock)
```

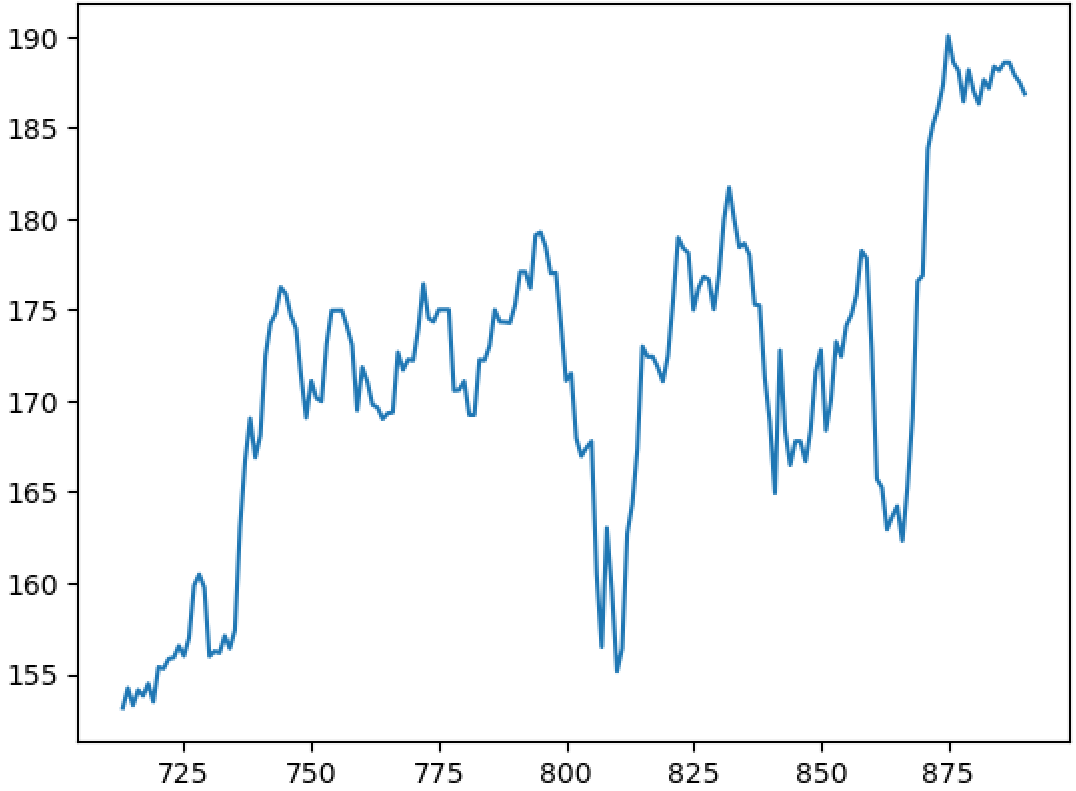
In [15]:
```
data_stock_np[-15]
```

Out[15]:
```
188.59
```

In [16]:
```
data_stock[:713].plot.line(x = 'Unnamed: 0', y = 'AAPL ')
plt.show()
```

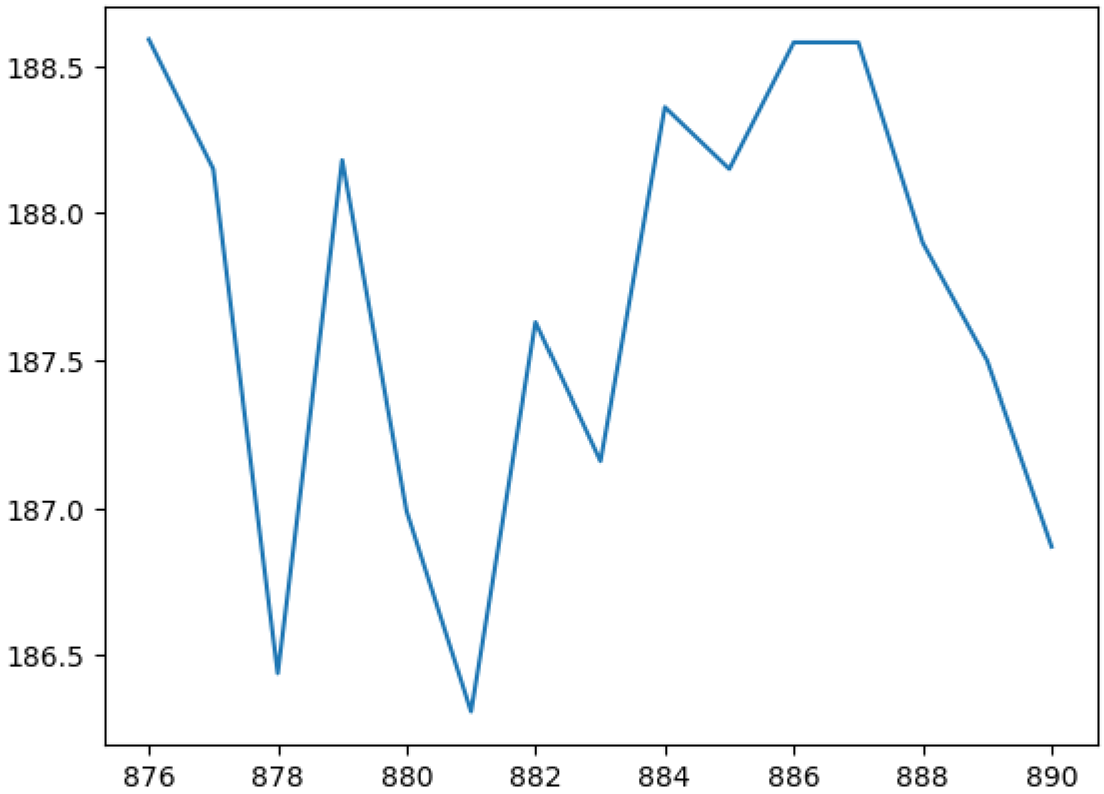


In [17]:
```
data_stock[-178:].plot.line(x = 'Unnamed: 0', y = 'AAPL ')
plt.show()
```



In [18]:

```
data_stock[-15:].plot.line(x = 'Unnamed: 0', y = 'AAPL ')
plt.show()
```



Utility function

In [19]:
```
from pandas import Series
from pandas import concat
from pandas import read_csv
import datetime
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, Bidirectional
from keras.layers import LSTM
from math import sqrt
from matplotlib import pyplot
import numpy
from tensorflow.keras import layers
```

```
# converting the series data to supervised data
def series_data_to_supervised_data(data, lag=1):
        df = pd.DataFrame(data)
        columns = [df.shift(i) for i in range(1, lag+1)]
        columns.append(df)
        df = concat(columns, axis=1)
        df.fillna(0, inplace=True)
        return df

# finding difference between rows
def difference(dataset, interval=1):
        diff = list()
        for i in range(interval, len(dataset)):
                value = dataset[i] - dataset[i - interval]
                diff.append(value)
        return Series(diff)

# inverting the difference value
def inverse_the_difference(history_data, yhat, interval=1):

  return (yhat+history_data[-interval])

# scaling the difference
def scale(train, test):
        # scaler
        scaler = MinMaxScaler(feature_range=(-1, 1))
        scaler = scaler.fit(train)

        train = train.reshape(train.shape[0], train.shape[1])
        train_scaled = scaler.transform(train)

        test = test.reshape(test.shape[0], test.shape[1])
        test_scaled = scaler.transform(test)
        return scaler, train_scaled, test_scaled

# inverse the scaling
def invert_scale(scaler, X, value):
        temp = [x for x in X] + [value]
        array = numpy.array(temp)
        array = array.reshape(1, len(array))
        data_inverted = scaler.inverse_transform(array)
        return data_inverted[0, -1]
```

```
# LSTM model
def fit_lstm(train, batchSize, epoch, neurons):
        X, y = train[:, 0:-1], train[:, -1]
        X = X.reshape(X.shape[0], 1, X.shape[1])
        model = Sequential()
        model.add(LSTM(neurons, batch_input_shape=(batchSize, X.shape[1], X.shape[2]), stateful=True))
        model.add(Dense(1))

        model.compile(loss='mean_squared_error', optimizer='adam')
        for i in range(epoch):
                model.fit(X, y, epochs=1, batch_size=batchSize, verbose=0, shuffle=False)
                model.reset_states()
        return model


# make a one-step forecast
def forecast_lstm(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]

def fit_rnn(train,batchSize,epoch, neurons):
  X, y = train[:, 0:-1], train[:, -1]
  X = X.reshape(X.shape[0], 1, X.shape[1])


  #RNN model

  print("X.shape[1]), ",X.shape[1])
  print("X.shape[2]) ",X.shape[2])

  model = Sequential()
  model.add(layers.SimpleRNN(neurons, batch_input_shape=(batchSize, X.shape[1], X.shape[2])))
  model.add(layers.Dense(1))
  model.compile(loss='mean_squared_error', optimizer='adam')


  for i in range(epoch):
    model.fit(X, y, epochs=1, batch_size=batchSize, verbose=0, shuffle=False )
    model.reset_states()
  return model

# make a one-step forecast
def forecast_lstm(model, batch_size, X):
```

```
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]


# make a one-step forecast
def forecast_rnn(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]

def fit_cnn(train,batchSize,epoch, neurons):
  X, y = train[:, 0:-1], train[:, -1]
  X = X.reshape(X.shape[0], 1, X.shape[1])
  print("X.shape[1]), ",X.shape[1])
  print("X.shape[2]) ",X.shape[2])

  #CNN model
  model = Sequential()
  model.add(layers.Conv1D(neurons,1, batch_input_shape=(batchSize, X.shape[1], X.shape[2])))
  model.add(layers.GlobalMaxPooling1D())
  model.add(layers.Dense(1))
  model.compile(loss='mean_squared_error', optimizer='adam')

   #model.add(layers.Dense(3, activation='sigmoid'))
  for i in range(epoch):
    model.fit(X, y, epochs=1, batch_size=batchSize, verbose=0, shuffle=False )
    model.reset_states()
  return model


# make a one-step forecast
def forecast_cnn(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]



def fit_bilstm(train,batchSize,epoch, neurons):
  X, y = train[:, 0:-1], train[:, -1]
  X = X.reshape(X.shape[0], 1, X.shape[1])
  model = Sequential()
```

```
  print("X.shape[1]), ",X.shape[1])
  print("X.shape[2]) ",X.shape[2])

  # Bi LSTM Model
  model = Sequential()
  model.add(Bidirectional(LSTM(neurons, batch_input_shape=(batchSize, X.shape[1], X.shape[2]), stateful=True)))
  model.add(Dense(1))
  model.compile(loss='mean_squared_error', optimizer='adam')

  #model.add(layers.Dense(3, activation='sigmoid'))
  for i in range(epoch):
    model.fit(X, y, epochs=1, batch_size=batchSize, verbose=0, shuffle=False )
    model.reset_states()
  return model


# make a one-step forecast
def forecast_bilstm(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]
```

LSTM

In [20]:
```
data_stock.shape
```

Out[20]:
```
(891,)
```

In [21]:
```
0.20 * 891
```

Out[21]:
```
178.20000000000002
```

In [22]:
```
# modify data to be stationary
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values
```

```
# split data
train, test = supervised_values[0:-178], supervised_values[-178:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
lstm_model = fit_lstm(train_scaled, 1, 1, 4)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        #print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-178:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-178:])
pyplot.plot(predictions)
pyplot.show()
```

```
712/712 [==============================] - 2s 2ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 28ms/step
```

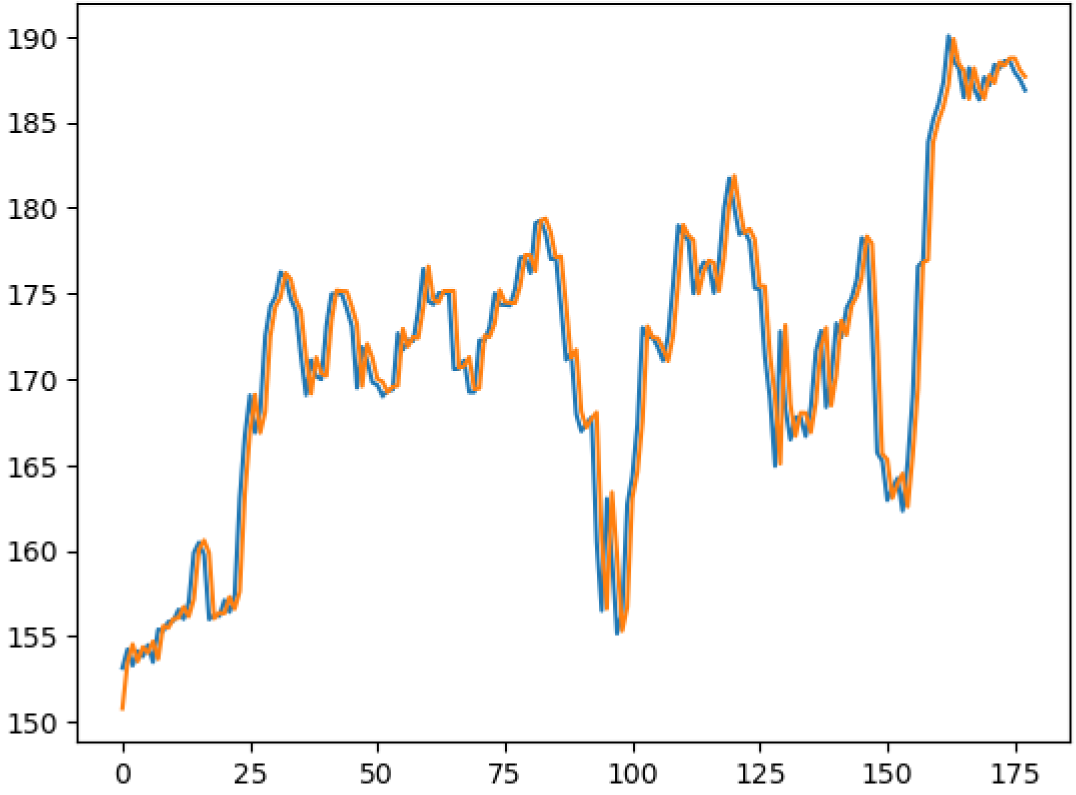
```
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
```

```
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 34ms/step
```

```
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
```

```
Test RMSE: 2.370
```



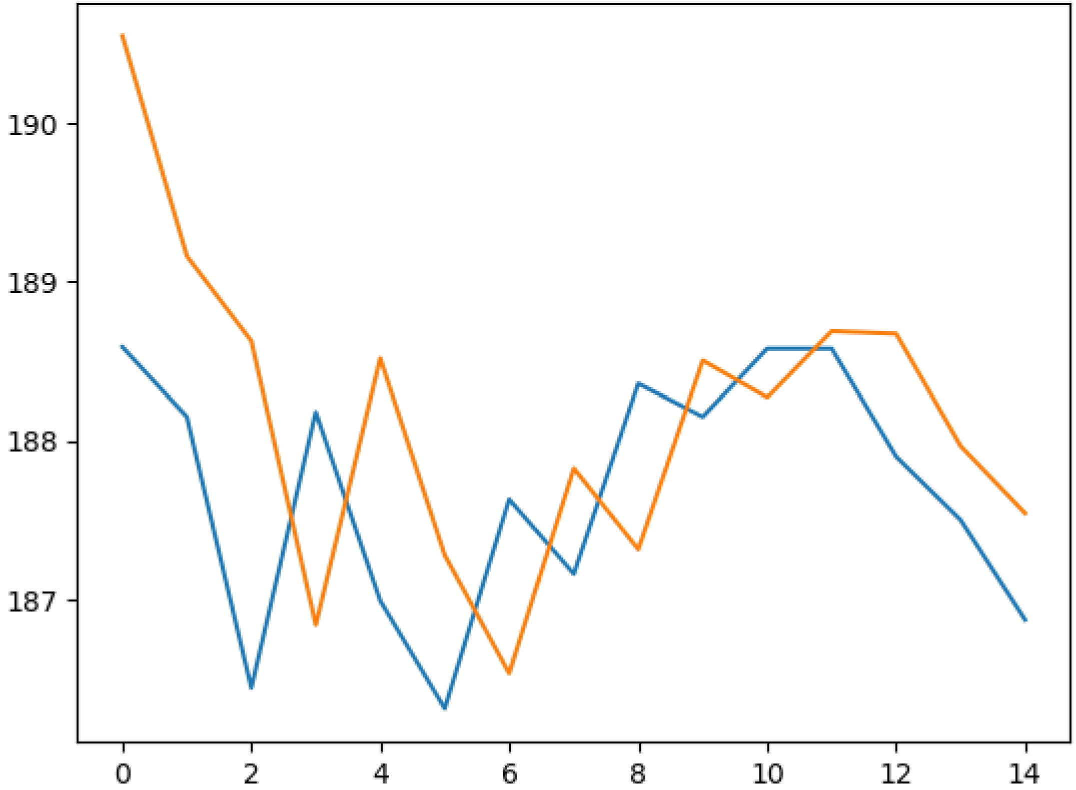
Over 15 test rows

In [23]:
```
# modify data to be stationary
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
```

```
train, test = supervised_values[0:-15], supervised_values[-15:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
lstm_model = fit_lstm(train_scaled, 1, 1, 4)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_lstm(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        #print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-15:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-15:])
pyplot.plot(predictions)
pyplot.show()
```

```
875/875 [==============================] - 2s 2ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
```

```
Test RMSE: 1.125
```



RNN

In [24]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-178], supervised_values[-178:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
lstm_model = fit_rnn(train_scaled, 1, 1, 4)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_rnn(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-178:], predictions))
```

```
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-178:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
712/712 [==============================] - 1s 2ms/step
1/1 [==============================] - 0s 30ms/step
Month=1, Predicted=150.858546, Expected=153.140000
1/1 [==============================] - 0s 31ms/step
Month=2, Predicted=153.036291, Expected=154.230000
1/1 [==============================] - 0s 30ms/step
Month=3, Predicted=154.275633, Expected=153.280000
1/1 [==============================] - 0s 35ms/step
Month=4, Predicted=153.547355, Expected=154.120000
1/1 [==============================] - 0s 35ms/step
Month=5, Predicted=154.192395, Expected=153.810000
1/1 [==============================] - 0s 37ms/step
Month=6, Predicted=154.008013, Expected=154.480000
1/1 [==============================] - 0s 34ms/step
Month=7, Predicted=154.570775, Expected=153.480000
1/1 [==============================] - 0s 32ms/step
Month=8, Predicted=153.752692, Expected=155.390000
1/1 [==============================] - 0s 33ms/step
Month=9, Predicted=155.351098, Expected=155.300000
1/1 [==============================] - 0s 36ms/step
Month=10, Predicted=155.473905, Expected=155.840000
1/1 [==============================] - 0s 31ms/step
Month=11, Predicted=155.944907, Expected=155.900000
1/1 [==============================] - 0s 45ms/step
Month=12, Predicted=156.057445, Expected=156.550000
1/1 [==============================] - 0s 32ms/step
Month=13, Predicted=156.642945, Expected=156.000000
1/1 [==============================] - 0s 36ms/step
Month=14, Predicted=156.224198, Expected=156.990000
1/1 [==============================] - 0s 30ms/step
Month=15, Predicted=157.046295, Expected=159.880000
1/1 [==============================] - 0s 31ms/step
Month=16, Predicted=159.749618, Expected=160.470000
1/1 [==============================] - 0s 31ms/step
Month=17, Predicted=160.569465, Expected=159.760000
1/1 [==============================] - 0s 34ms/step
Month=18, Predicted=160.001548, Expected=155.980000
1/1 [==============================] - 0s 31ms/step
Month=19, Predicted=156.508405, Expected=156.250000
1/1 [==============================] - 0s 37ms/step
Month=20, Predicted=156.384414, Expected=156.170000
1/1 [==============================] - 0s 37ms/step
```

```
Month=21, Predicted=156.342808, Expected=157.100000
1/1 [==============================] - 0s 32ms/step
Month=22, Predicted=157.162721, Expected=156.410000
1/1 [==============================] - 0s 38ms/step
Month=23, Predicted=156.649385, Expected=157.410000
1/1 [==============================] - 0s 38ms/step
Month=24, Predicted=157.465226, Expected=163.050000
1/1 [==============================] - 0s 34ms/step
Month=25, Predicted=162.743349, Expected=166.720000
1/1 [==============================] - 0s 34ms/step
Month=26, Predicted=166.526611, Expected=169.040000
1/1 [==============================] - 0s 33ms/step
Month=27, Predicted=168.961343, Expected=166.890000
1/1 [==============================] - 0s 33ms/step
Month=28, Predicted=167.279906, Expected=168.110000
1/1 [==============================] - 0s 32ms/step
Month=29, Predicted=168.141868, Expected=172.500000
1/1 [==============================] - 0s 33ms/step
Month=30, Predicted=172.257352, Expected=174.250000
1/1 [==============================] - 0s 32ms/step
Month=31, Predicted=174.227117, Expected=174.810000
1/1 [==============================] - 0s 36ms/step
Month=32, Predicted=174.912729, Expected=176.240000
1/1 [==============================] - 0s 37ms/step
Month=33, Predicted=176.249891, Expected=175.880000
1/1 [==============================] - 0s 41ms/step
Month=34, Predicted=176.083480, Expected=174.670000
1/1 [==============================] - 0s 39ms/step
Month=35, Predicted=174.964934, Expected=173.970000
1/1 [==============================] - 0s 35ms/step
Month=36, Predicted=174.210467, Expected=171.340000
1/1 [==============================] - 0s 37ms/step
Month=37, Predicted=171.774628, Expected=169.080000
1/1 [==============================] - 0s 34ms/step
Month=38, Predicted=169.480411, Expected=171.100000
1/1 [==============================] - 0s 34ms/step
Month=39, Predicted=171.050243, Expected=170.150000
1/1 [==============================] - 0s 30ms/step
Month=40, Predicted=170.417355, Expected=169.980000
1/1 [==============================] - 0s 31ms/step
Month=41, Predicted=170.162679, Expected=173.140000
1/1 [==============================] - 0s 34ms/step
Month=42, Predicted=172.986733, Expected=174.960000
1/1 [==============================] - 0s 32ms/step
```

```
Month=43, Predicted=174.930076, Expected=174.960000
1/1 [==============================] - 0s 32ms/step
Month=44, Predicted=175.124030, Expected=174.970000
1/1 [==============================] - 0s 36ms/step
Month=45, Predicted=175.132932, Expected=174.090000
1/1 [==============================] - 0s 42ms/step
Month=46, Predicted=174.349859, Expected=173.070000
1/1 [==============================] - 0s 33ms/step
Month=47, Predicted=173.344823, Expected=169.480000
1/1 [==============================] - 0s 39ms/step
Month=48, Predicted=169.994333, Expected=171.850000
1/1 [==============================] - 0s 31ms/step
Month=49, Predicted=171.766633, Expected=171.050000
1/1 [==============================] - 0s 32ms/step
Month=50, Predicted=171.301260, Expected=169.800000
1/1 [==============================] - 0s 35ms/step
Month=51, Predicted=170.099136, Expected=169.640000
1/1 [==============================] - 0s 34ms/step
Month=52, Predicted=169.821582, Expected=169.010000
1/1 [==============================] - 0s 34ms/step
Month=53, Predicted=169.242886, Expected=169.320000
1/1 [==============================] - 0s 31ms/step
Month=54, Predicted=169.450033, Expected=169.370000
1/1 [==============================] - 0s 31ms/step
Month=55, Predicted=169.528542, Expected=172.670000
1/1 [==============================] - 0s 31ms/step
Month=56, Predicted=172.505305, Expected=171.700000
1/1 [==============================] - 0s 32ms/step
Month=57, Predicted=171.969492, Expected=172.270000
1/1 [==============================] - 0s 34ms/step
Month=58, Predicted=172.371641, Expected=172.220000
1/1 [==============================] - 0s 32ms/step
Month=59, Predicted=172.389517, Expected=173.970000
1/1 [==============================] - 0s 32ms/step
Month=60, Predicted=173.947117, Expected=176.420000
1/1 [==============================] - 0s 39ms/step
Month=61, Predicted=176.329162, Expected=174.540000
1/1 [==============================] - 0s 30ms/step
Month=62, Predicted=174.903528, Expected=174.350000
1/1 [==============================] - 0s 33ms/step
Month=63, Predicted=174.534871, Expected=175.010000
1/1 [==============================] - 0s 41ms/step
Month=64, Predicted=175.101860, Expected=175.010000
1/1 [==============================] - 0s 36ms/step
```

```
Month=65, Predicted=175.174030, Expected=175.010000
1/1 [==============================] - 0s 35ms/step
Month=66, Predicted=175.174030, Expected=170.570000
1/1 [==============================] - 0s 33ms/step
Month=67, Predicted=171.142619, Expected=170.600000
1/1 [==============================] - 0s 34ms/step
Month=68, Predicted=170.760737, Expected=171.080000
1/1 [==============================] - 0s 38ms/step
Month=69, Predicted=171.191449, Expected=169.230000
1/1 [==============================] - 0s 35ms/step
Month=70, Predicted=169.590547, Expected=169.230000
1/1 [==============================] - 0s 32ms/step
Month=71, Predicted=169.394030, Expected=172.260000
1/1 [==============================] - 0s 38ms/step
Month=72, Predicted=172.117615, Expected=172.230000
1/1 [==============================] - 0s 29ms/step
Month=73, Predicted=172.397322, Expected=173.030000
1/1 [==============================] - 0s 29ms/step
Month=74, Predicted=173.106708, Expected=175.000000
1/1 [==============================] - 0s 34ms/step
Month=75, Predicted=174.955161, Expected=174.350000
1/1 [==============================] - 0s 37ms/step
Month=76, Predicted=174.585054, Expected=174.330000
1/1 [==============================] - 0s 34ms/step
Month=77, Predicted=174.496225, Expected=174.290000
1/1 [==============================] - 0s 34ms/step
Month=78, Predicted=174.458419, Expected=175.280000
1/1 [==============================] - 0s 32ms/step
Month=79, Predicted=175.336295, Expected=177.090000
1/1 [==============================] - 0s 29ms/step
Month=80, Predicted=177.061079, Expected=177.090000
1/1 [==============================] - 0s 32ms/step
Month=81, Predicted=177.254030, Expected=176.190000
1/1 [==============================] - 0s 31ms/step
Month=82, Predicted=176.452004, Expected=179.100000
1/1 [==============================] - 0s 33ms/step
Month=83, Predicted=178.967885, Expected=179.260000
1/1 [==============================] - 0s 30ms/step
Month=84, Predicted=179.406473, Expected=178.460000
1/1 [==============================] - 0s 32ms/step
Month=85, Predicted=178.711260, Expected=177.000000
1/1 [==============================] - 0s 31ms/step
Month=86, Predicted=177.320993, Expected=177.040000
1/1 [==============================] - 0s 32ms/step
```

```
Month=87, Predicted=177.199640, Expected=174.220000
1/1 [==============================] - 0s 31ms/step
Month=88, Predicted=174.671485, Expected=171.110000
1/1 [==============================] - 0s 31ms/step
Month=89, Predicted=171.586215, Expected=171.510000
1/1 [==============================] - 0s 32ms/step
Month=90, Predicted=171.630186, Expected=167.960000
1/1 [==============================] - 0s 40ms/step
Month=91, Predicted=168.471295, Expected=166.970000
1/1 [==============================] - 0s 36ms/step
Month=92, Predicted=167.241626, Expected=167.430000
1/1 [==============================] - 0s 29ms/step
Month=93, Predicted=167.543631, Expected=167.780000
1/1 [==============================] - 0s 33ms/step
Month=94, Predicted=167.905654, Expected=160.500000
1/1 [==============================] - 0s 31ms/step
Month=95, Predicted=161.180099, Expected=156.490000
1/1 [==============================] - 0s 39ms/step
Month=96, Predicted=157.034645, Expected=163.030000
1/1 [==============================] - 0s 36ms/step
Month=97, Predicted=162.693890, Expected=159.540000
1/1 [==============================] - 0s 33ms/step
Month=98, Predicted=160.046691, Expected=155.150000
1/1 [==============================] - 0s 33ms/step
Month=99, Predicted=155.719528, Expected=156.410000
1/1 [==============================] - 0s 31ms/step
Month=100, Predicted=156.437656, Expected=162.710000
1/1 [==============================] - 0s 33ms/step
Month=101, Predicted=162.380442, Expected=164.340000
1/1 [==============================] - 0s 32ms/step
Month=102, Predicted=164.329297, Expected=167.370000
1/1 [==============================] - 0s 39ms/step
Month=103, Predicted=167.227615, Expected=172.990000
1/1 [==============================] - 0s 31ms/step
Month=104, Predicted=172.684158, Expected=172.430000
1/1 [==============================] - 0s 34ms/step
Month=105, Predicted=172.655285, Expected=172.430000
1/1 [==============================] - 0s 37ms/step
Month=106, Predicted=172.594030, Expected=171.850000
1/1 [==============================] - 0s 33ms/step
Month=107, Predicted=172.077458, Expected=171.070000
1/1 [==============================] - 0s 32ms/step
Month=108, Predicted=171.319105, Expected=172.500000
1/1 [==============================] - 0s 35ms/step
```

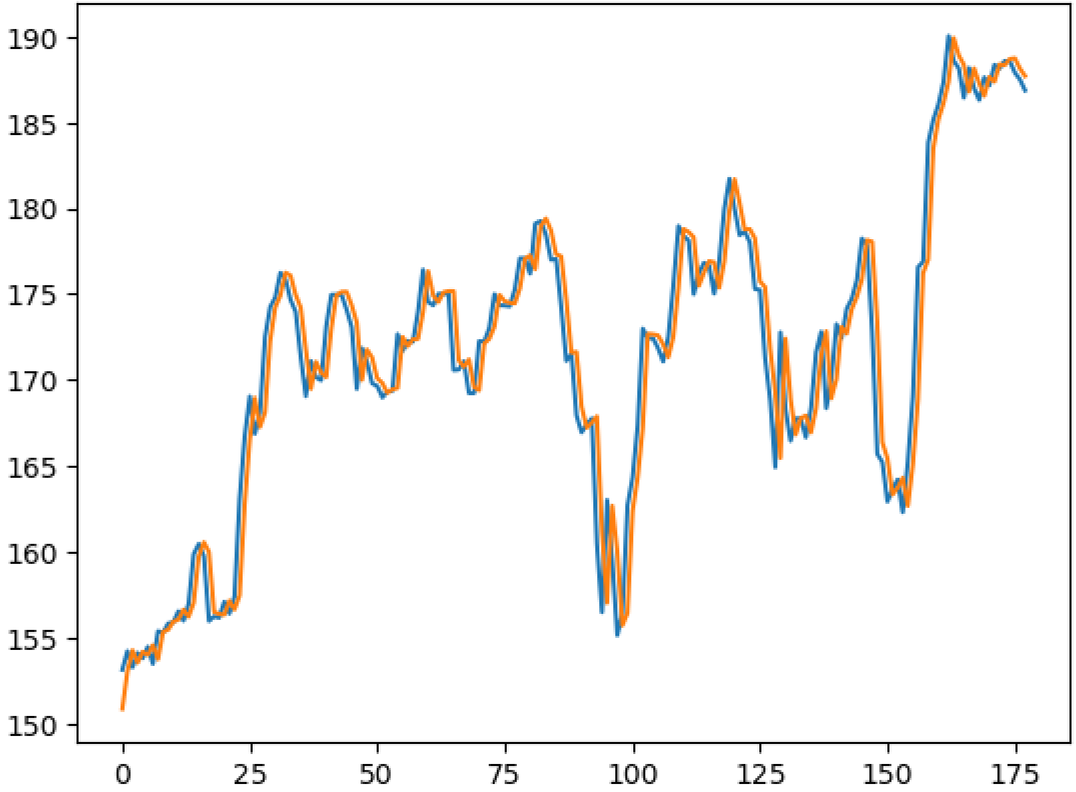
```
Month=109, Predicted=172.509891, Expected=175.500000
1/1 [==============================] - 0s 40ms/step
Month=110, Predicted=175.360162, Expected=178.970000
1/1 [==============================] - 0s 30ms/step
Month=111, Predicted=178.791846, Expected=178.390000
1/1 [==============================] - 0s 34ms/step
Month=112, Predicted=178.617458, Expected=178.120000
1/1 [==============================] - 0s 32ms/step
Month=113, Predicted=178.313635, Expected=175.000000
1/1 [==============================] - 0s 37ms/step
Month=114, Predicted=175.477046, Expected=176.210000
1/1 [==============================] - 0s 46ms/step
Month=115, Predicted=176.242923, Expected=176.820000
1/1 [==============================] - 0s 33ms/step
Month=116, Predicted=176.917290, Expected=176.670000
1/1 [==============================] - 0s 32ms/step
Month=117, Predicted=176.850486, Expected=175.030000
1/1 [==============================] - 0s 31ms/step
Month=118, Predicted=175.369428, Expected=176.940000
1/1 [==============================] - 0s 40ms/step
Month=119, Predicted=176.901098, Expected=179.980000
1/1 [==============================] - 0s 39ms/step
Month=120, Predicted=179.836769, Expected=181.720000
1/1 [==============================] - 0s 39ms/step
Month=121, Predicted=181.698126, Expected=179.970000
1/1 [==============================] - 0s 34ms/step
Month=122, Predicted=180.320545, Expected=178.440000
1/1 [==============================] - 0s 32ms/step
Month=123, Predicted=178.768197, Expected=178.650000
1/1 [==============================] - 0s 28ms/step
Month=124, Predicted=178.790990, Expected=178.020000
1/1 [==============================] - 0s 31ms/step
Month=125, Predicted=178.252886, Expected=175.300000
1/1 [==============================] - 0s 40ms/step
Month=126, Predicted=175.742676, Expected=175.240000
1/1 [==============================] - 0s 38ms/step
Month=127, Predicted=175.410614, Expected=171.270000
1/1 [==============================] - 0s 37ms/step
Month=128, Predicted=171.811884, Expected=168.850000
1/1 [==============================] - 0s 38ms/step
Month=129, Predicted=169.265425, Expected=164.940000
1/1 [==============================] - 0s 29ms/step
Month=130, Predicted=165.477693, Expected=172.770000
1/1 [==============================] - 0s 31ms/step
```

```
Month=131, Predicted=172.413504, Expected=168.340000
1/1 [==============================] - 0s 32ms/step
Month=132, Predicted=168.912004, Expected=166.480000
1/1 [==============================] - 0s 36ms/step
Month=133, Predicted=166.841541, Expected=167.780000
1/1 [==============================] - 0s 39ms/step
Month=134, Predicted=167.803456, Expected=167.780000
1/1 [==============================] - 0s 34ms/step
Month=135, Predicted=167.944030, Expected=166.680000
1/1 [==============================] - 0s 36ms/step
Month=136, Predicted=166.963320, Expected=168.390000
1/1 [==============================] - 0s 33ms/step
Month=137, Predicted=168.371161, Expected=171.610000
1/1 [==============================] - 0s 32ms/step
Month=138, Predicted=171.451798, Expected=172.800000
1/1 [==============================] - 0s 39ms/step
Month=139, Predicted=172.835035, Expected=168.380000
1/1 [==============================] - 0s 37ms/step
Month=140, Predicted=168.951388, Expected=170.050000
1/1 [==============================] - 0s 38ms/step
Month=141, Predicted=170.035222, Expected=173.250000
1/1 [==============================] - 0s 37ms/step
Month=142, Predicted=173.093437, Expected=172.440000
1/1 [==============================] - 0s 32ms/step
Month=143, Predicted=172.692337, Expected=174.140000
1/1 [==============================] - 0s 33ms/step
Month=144, Predicted=174.122175, Expected=174.730000
1/1 [==============================] - 0s 30ms/step
Month=145, Predicted=174.829465, Expected=175.820000
1/1 [==============================] - 0s 32ms/step
Month=146, Predicted=175.865633, Expected=178.240000
1/1 [==============================] - 0s 30ms/step
Month=147, Predicted=178.151954, Expected=177.840000
1/1 [==============================] - 0s 32ms/step
Month=148, Predicted=178.047850, Expected=172.800000
1/1 [==============================] - 0s 30ms/step
Month=149, Predicted=173.406378, Expected=165.720000
1/1 [==============================] - 0s 39ms/step
Month=150, Predicted=166.396619, Expected=165.240000
1/1 [==============================] - 0s 34ms/step
Month=151, Predicted=165.456577, Expected=162.940000
1/1 [==============================] - 0s 31ms/step
Month=152, Predicted=163.344194, Expected=163.650000
1/1 [==============================] - 0s 42ms/step
```

```
Month=153, Predicted=163.736439, Expected=164.220000
1/1 [==============================] - 0s 38ms/step
Month=154, Predicted=164.321641, Expected=162.320000
1/1 [==============================] - 0s 33ms/step
Month=155, Predicted=162.685509, Expected=165.260000
1/1 [==============================] - 0s 38ms/step
Month=156, Predicted=165.125297, Expected=169.100000
1/1 [==============================] - 0s 42ms/step
Month=157, Predicted=168.894183, Expected=176.570000
1/1 [==============================] - 0s 38ms/step
Month=158, Predicted=176.216824, Expected=176.890000
1/1 [==============================] - 0s 35ms/step
Month=159, Predicted=177.018938, Expected=183.830000
1/1 [==============================] - 0s 32ms/step
Month=160, Predicted=183.484976, Expected=185.160000
1/1 [==============================] - 0s 31ms/step
Month=161, Predicted=185.180314, Expected=186.050000
1/1 [==============================] - 0s 33ms/step
Month=162, Predicted=186.117015, Expected=187.360000
1/1 [==============================] - 0s 32ms/step
Month=163, Predicted=187.382408, Expected=190.040000
1/1 [==============================] - 0s 34ms/step
Month=164, Predicted=189.928157, Expected=188.590000
1/1 [==============================] - 0s 36ms/step
Month=165, Predicted=188.909960, Expected=188.150000
1/1 [==============================] - 0s 40ms/step
Month=166, Predicted=188.362216, Expected=186.440000
1/1 [==============================] - 0s 42ms/step
Month=167, Predicted=186.786516, Expected=188.180000
1/1 [==============================] - 0s 34ms/step
Month=168, Predicted=188.158126, Expected=186.990000
1/1 [==============================] - 0s 39ms/step
Month=169, Predicted=187.282829, Expected=186.310000
1/1 [==============================] - 0s 34ms/step
Month=170, Predicted=186.548303, Expected=187.630000
1/1 [==============================] - 0s 33ms/step
Month=171, Predicted=187.651361, Expected=187.160000
1/1 [==============================] - 0s 34ms/step
Month=172, Predicted=187.375487, Expected=188.360000
1/1 [==============================] - 0s 42ms/step
Month=173, Predicted=188.393978, Expected=188.150000
1/1 [==============================] - 0s 32ms/step
Month=174, Predicted=188.337063, Expected=188.580000
1/1 [==============================] - 0s 37ms/step
```

```
Month=175, Predicted=188.696908, Expected=188.580000
1/1 [==============================] - 0s 38ms/step
Month=176, Predicted=188.744030, Expected=187.900000
1/1 [==============================] - 0s 33ms/step
Month=177, Predicted=188.138303, Expected=187.500000
1/1 [==============================] - 0s 34ms/step
Month=178, Predicted=187.707850, Expected=186.870000
```

```
Test RMSE: 2.412
```

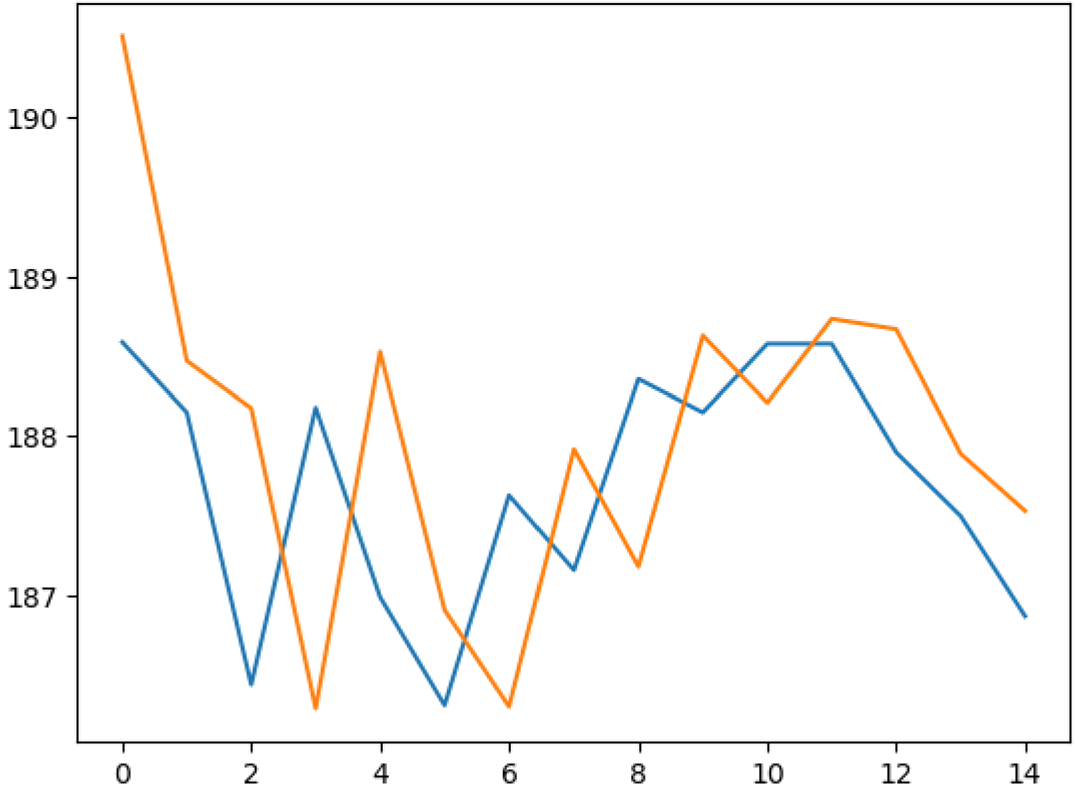


Over 15 test rows

In [25]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
```

```
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-15], supervised_values[-15:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
lstm_model = fit_rnn(train_scaled, 1, 1, 4)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_rnn(lstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-15:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-15:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
875/875 [==============================] - 2s 2ms/step
1/1 [==============================] - 0s 31ms/step
Month=1, Predicted=190.515305, Expected=188.590000
1/1 [==============================] - 0s 30ms/step
Month=2, Predicted=188.473870, Expected=188.150000
1/1 [==============================] - 0s 32ms/step
Month=3, Predicted=188.176493, Expected=186.440000
1/1 [==============================] - 0s 39ms/step
Month=4, Predicted=186.289961, Expected=188.180000
1/1 [==============================] - 0s 37ms/step
Month=5, Predicted=188.530675, Expected=186.990000
1/1 [==============================] - 0s 33ms/step
Month=6, Predicted=186.909095, Expected=186.310000
1/1 [==============================] - 0s 35ms/step
Month=7, Predicted=186.301352, Expected=187.630000
1/1 [==============================] - 0s 30ms/step
Month=8, Predicted=187.920263, Expected=187.160000
1/1 [==============================] - 0s 32ms/step
Month=9, Predicted=187.182069, Expected=188.360000
1/1 [==============================] - 0s 32ms/step
Month=10, Predicted=188.632629, Expected=188.150000
1/1 [==============================] - 0s 34ms/step
Month=11, Predicted=188.210663, Expected=188.580000
1/1 [==============================] - 0s 36ms/step
Month=12, Predicted=188.737146, Expected=188.580000
1/1 [==============================] - 0s 44ms/step
Month=13, Predicted=188.672168, Expected=187.900000
1/1 [==============================] - 0s 34ms/step
Month=14, Predicted=187.891352, Expected=187.500000
1/1 [==============================] - 0s 29ms/step
Month=15, Predicted=187.532405, Expected=186.870000

 Test RMSE: 1.109
```



CNN Model

In [26]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-178], supervised_values[-178:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
```

```
cnn_model = fit_cnn(train_scaled, 1, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
cnn_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_cnn(cnn_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-178:], predictions))
print('Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-178:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
712/712 [==============================] - 1s 2ms/step
1/1 [==============================] - 0s 33ms/step
Month=1, Predicted=150.999398, Expected=153.140000
1/1 [==============================] - 0s 30ms/step
Month=2, Predicted=152.803142, Expected=154.230000
1/1 [==============================] - 0s 31ms/step
Month=3, Predicted=154.193240, Expected=153.280000
1/1 [==============================] - 0s 32ms/step
Month=4, Predicted=153.651373, Expected=154.120000
1/1 [==============================] - 0s 31ms/step
Month=5, Predicted=154.133256, Expected=153.810000
1/1 [==============================] - 0s 33ms/step
Month=6, Predicted=154.053331, Expected=154.480000
1/1 [==============================] - 0s 33ms/step
Month=7, Predicted=154.527267, Expected=153.480000
1/1 [==============================] - 0s 33ms/step
Month=8, Predicted=153.861376, Expected=155.390000
1/1 [==============================] - 0s 30ms/step
Month=9, Predicted=155.189186, Expected=155.300000
1/1 [==============================] - 0s 28ms/step
Month=10, Predicted=155.499317, Expected=155.840000
1/1 [==============================] - 0s 36ms/step
Month=11, Predicted=155.913275, Expected=155.900000
1/1 [==============================] - 0s 31ms/step
Month=12, Predicted=156.069307, Expected=156.550000
1/1 [==============================] - 0s 32ms/step
Month=13, Predicted=156.601268, Expected=156.000000
1/1 [==============================] - 0s 37ms/step
Month=14, Predicted=156.291347, Expected=156.990000
1/1 [==============================] - 0s 33ms/step
Month=15, Predicted=156.973246, Expected=159.880000
1/1 [==============================] - 0s 30ms/step
Month=16, Predicted=159.483122, Expected=160.470000
1/1 [==============================] - 0s 34ms/step
Month=17, Predicted=160.533272, Expected=159.760000
1/1 [==============================] - 0s 31ms/step
Month=18, Predicted=160.083357, Expected=155.980000
1/1 [==============================] - 0s 32ms/step
Month=19, Predicted=156.917557, Expected=156.250000
1/1 [==============================] - 0s 31ms/step
Month=20, Predicted=156.377293, Expected=156.170000
1/1 [==============================] - 0s 32ms/step
```

```
Month=21, Predicted=156.367316, Expected=157.100000
1/1 [==============================] - 0s 35ms/step
Month=22, Predicted=157.095250, Expected=156.410000
1/1 [==============================] - 0s 33ms/step
Month=23, Predicted=156.729356, Expected=157.410000
1/1 [==============================] - 0s 40ms/step
Month=24, Predicted=157.391245, Expected=163.050000
1/1 [==============================] - 0s 33ms/step
Month=25, Predicted=162.102943, Expected=166.720000
1/1 [==============================] - 0s 34ms/step
Month=26, Predicted=166.167072, Expected=169.040000
1/1 [==============================] - 0s 35ms/step
Month=27, Predicted=168.757160, Expected=166.890000
1/1 [==============================] - 0s 31ms/step
Month=28, Predicted=167.501451, Expected=168.110000
1/1 [==============================] - 0s 35ms/step
Month=29, Predicted=168.047231, Expected=172.500000
1/1 [==============================] - 0s 35ms/step
Month=30, Predicted=171.803025, Expected=174.250000
1/1 [==============================] - 0s 33ms/step
Month=31, Predicted=174.081197, Expected=174.810000
1/1 [==============================] - 0s 36ms/step
Month=32, Predicted=174.879274, Expected=176.240000
1/1 [==============================] - 0s 39ms/step
Month=33, Predicted=176.135218, Expected=175.880000
1/1 [==============================] - 0s 38ms/step
Month=34, Predicted=176.133334, Expected=174.670000
1/1 [==============================] - 0s 33ms/step
Month=35, Predicted=175.093389, Expected=173.970000
1/1 [==============================] - 0s 40ms/step
Month=36, Predicted=174.291356, Expected=171.340000
1/1 [==============================] - 0s 31ms/step
Month=37, Predicted=172.047482, Expected=169.080000
1/1 [==============================] - 0s 32ms/step
Month=38, Predicted=169.713458, Expected=171.100000
1/1 [==============================] - 0s 34ms/step
Month=39, Predicted=170.877179, Expected=170.150000
1/1 [==============================] - 0s 37ms/step
Month=40, Predicted=170.521373, Expected=169.980000
1/1 [==============================] - 0s 33ms/step
Month=41, Predicted=170.195322, Expected=173.140000
1/1 [==============================] - 0s 32ms/step
Month=42, Predicted=172.689105, Expected=174.960000
1/1 [==============================] - 0s 33ms/step
```

```
Month=43, Predicted=174.777192, Expected=174.960000
1/1 [==============================] - 0s 32ms/step
Month=44, Predicted=175.141311, Expected=174.970000
1/1 [==============================] - 0s 33ms/step
Month=45, Predicted=175.149310, Expected=174.090000
1/1 [==============================] - 0s 31ms/step
Month=46, Predicted=174.447368, Expected=173.070000
1/1 [==============================] - 0s 34ms/step
Month=47, Predicted=173.455377, Expected=169.480000
1/1 [==============================] - 0s 41ms/step
Month=48, Predicted=170.379545, Expected=171.850000
1/1 [==============================] - 0s 32ms/step
Month=49, Predicted=171.557156, Expected=171.050000
1/1 [==============================] - 0s 36ms/step
Month=50, Predicted=171.391363, Expected=169.800000
1/1 [==============================] - 0s 39ms/step
Month=51, Predicted=170.231392, Expected=169.640000
1/1 [==============================] - 0s 32ms/step
Month=52, Predicted=169.853321, Expected=169.010000
1/1 [==============================] - 0s 30ms/step
Month=53, Predicted=169.317352, Expected=169.320000
1/1 [==============================] - 0s 33ms/step
Month=54, Predicted=169.439290, Expected=169.370000
1/1 [==============================] - 0s 30ms/step
Month=55, Predicted=169.541307, Expected=172.670000
1/1 [==============================] - 0s 30ms/step
Month=56, Predicted=172.191096, Expected=171.700000
1/1 [==============================] - 0s 31ms/step
Month=57, Predicted=172.075374, Expected=172.270000
1/1 [==============================] - 0s 34ms/step
Month=58, Predicted=172.337274, Expected=172.220000
1/1 [==============================] - 0s 33ms/step
Month=59, Predicted=172.411314, Expected=173.970000
1/1 [==============================] - 0s 29ms/step
Month=60, Predicted=173.801197, Expected=176.420000
1/1 [==============================] - 0s 29ms/step
Month=61, Predicted=176.111151, Expected=174.540000
1/1 [==============================] - 0s 30ms/step
Month=62, Predicted=175.097433, Expected=174.350000
1/1 [==============================] - 0s 32ms/step
Month=63, Predicted=174.569323, Expected=175.010000
1/1 [==============================] - 0s 33ms/step
Month=64, Predicted=175.059268, Expected=175.010000
1/1 [==============================] - 0s 29ms/step
```

```
Month=65, Predicted=175.191311, Expected=175.010000
1/1 [==============================] - 0s 30ms/step
Month=66, Predicted=175.191311, Expected=170.570000
1/1 [==============================] - 0s 34ms/step
Month=67, Predicted=171.639600, Expected=170.600000
1/1 [==============================] - 0s 31ms/step
Month=68, Predicted=170.775309, Expected=171.080000
1/1 [==============================] - 0s 30ms/step
Month=69, Predicted=171.165279, Expected=169.230000
1/1 [==============================] - 0s 32ms/step
Month=70, Predicted=169.781431, Expected=169.230000
1/1 [==============================] - 0s 33ms/step
Month=71, Predicted=169.411311, Expected=172.260000
1/1 [==============================] - 0s 32ms/step
Month=72, Predicted=171.835113, Expected=172.230000
1/1 [==============================] - 0s 33ms/step
Month=73, Predicted=172.417313, Expected=173.030000
1/1 [==============================] - 0s 33ms/step
Month=74, Predicted=173.051259, Expected=175.000000
1/1 [==============================] - 0s 34ms/step
Month=75, Predicted=174.787182, Expected=174.350000
1/1 [==============================] - 0s 31ms/step
Month=76, Predicted=174.661353, Expected=174.330000
1/1 [==============================] - 0s 30ms/step
Month=77, Predicted=174.515312, Expected=174.290000
1/1 [==============================] - 0s 33ms/step
Month=78, Predicted=174.479313, Expected=175.280000
1/1 [==============================] - 0s 31ms/step
Month=79, Predicted=175.263246, Expected=177.090000
1/1 [==============================] - 0s 32ms/step
Month=80, Predicted=176.909193, Expected=177.090000
1/1 [==============================] - 0s 30ms/step
Month=81, Predicted=177.271311, Expected=176.190000
1/1 [==============================] - 0s 31ms/step
Month=82, Predicted=176.551369, Expected=179.100000
1/1 [==============================] - 0s 33ms/step
Month=83, Predicted=178.699121, Expected=179.260000
1/1 [==============================] - 0s 30ms/step
Month=84, Predicted=179.409300, Expected=178.460000
1/1 [==============================] - 0s 29ms/step
Month=85, Predicted=178.801363, Expected=177.000000
1/1 [==============================] - 0s 46ms/step
Month=86, Predicted=177.473406, Expected=177.040000
1/1 [==============================] - 0s 31ms/step
```

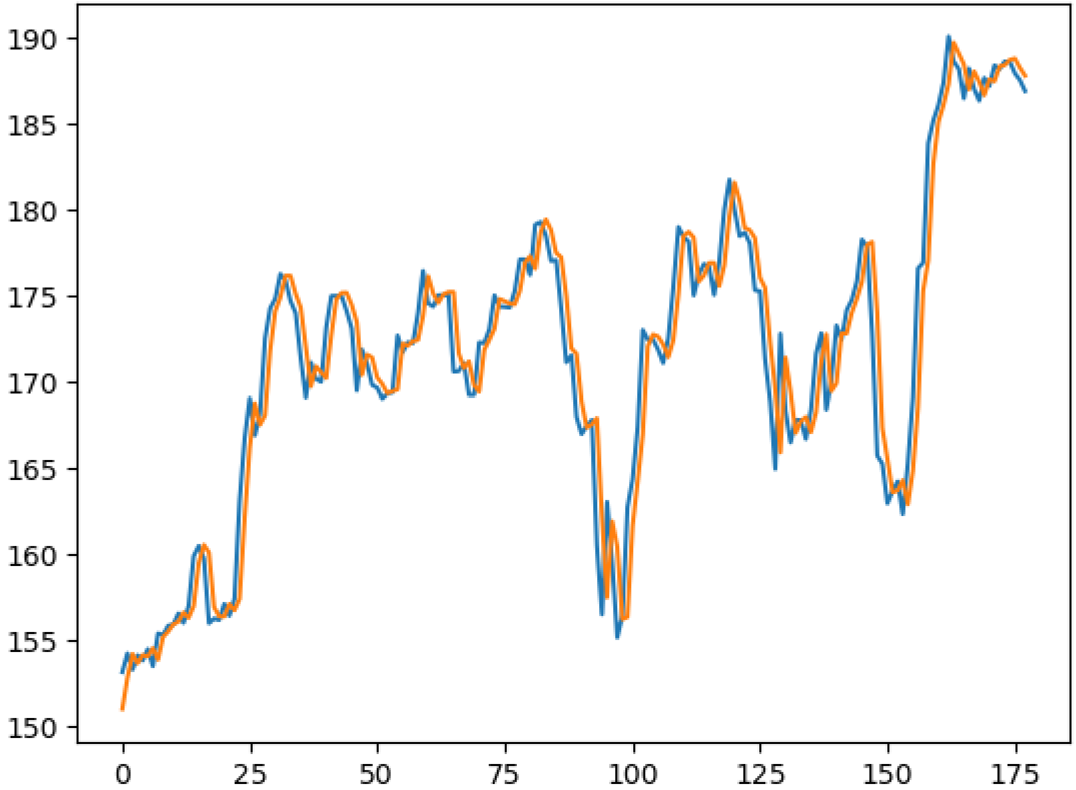
```
Month=87, Predicted=177.213308, Expected=174.220000
1/1 [==============================] - 0s 32ms/step
Month=88, Predicted=174.965494, Expected=171.110000
1/1 [==============================] - 0s 30ms/step
Month=89, Predicted=171.913513, Expected=171.510000
1/1 [==============================] - 0s 33ms/step
Month=90, Predicted=171.611285, Expected=167.960000
1/1 [==============================] - 0s 30ms/step
Month=91, Predicted=168.851542, Expected=166.970000
1/1 [==============================] - 0s 31ms/step
Month=92, Predicted=167.349375, Expected=167.430000
1/1 [==============================] - 0s 31ms/step
Month=93, Predicted=167.519281, Expected=167.780000
1/1 [==============================] - 0s 32ms/step
Month=94, Predicted=167.891288, Expected=160.500000
1/1 [==============================] - 0s 33ms/step
Month=95, Predicted=162.137785, Expected=156.490000
1/1 [==============================] - 0s 30ms/step
Month=96, Predicted=157.473572, Expected=163.030000
1/1 [==============================] - 0s 33ms/step
Month=97, Predicted=161.902884, Expected=159.540000
1/1 [==============================] - 0s 32ms/step
Month=98, Predicted=160.419538, Expected=155.150000
1/1 [==============================] - 0s 36ms/step
Month=99, Predicted=156.209597, Expected=156.410000
1/1 [==============================] - 0s 30ms/step
Month=100, Predicted=156.339229, Expected=162.710000
1/1 [==============================] - 0s 29ms/step
Month=101, Predicted=161.630900, Expected=164.340000
1/1 [==============================] - 0s 30ms/step
Month=102, Predicted=164.195204, Expected=167.370000
1/1 [==============================] - 0s 29ms/step
Month=103, Predicted=166.945113, Expected=172.990000
1/1 [==============================] - 0s 29ms/step
Month=104, Predicted=172.046944, Expected=172.430000
1/1 [==============================] - 0s 28ms/step
Month=105, Predicted=172.723347, Expected=172.430000
1/1 [==============================] - 0s 32ms/step
Month=106, Predicted=172.611311, Expected=171.850000
1/1 [==============================] - 0s 32ms/step
Month=107, Predicted=172.147348, Expected=171.070000
1/1 [==============================] - 0s 31ms/step
Month=108, Predicted=171.407361, Expected=172.500000
1/1 [==============================] - 0s 31ms/step
```

```
Month=109, Predicted=172.395218, Expected=175.500000
1/1 [==============================] - 0s 33ms/step
Month=110, Predicted=175.081115, Expected=178.970000
1/1 [==============================] - 0s 31ms/step
Month=111, Predicted=178.457085, Expected=178.390000
1/1 [==============================] - 0s 34ms/step
Month=112, Predicted=178.687348, Expected=178.120000
1/1 [==============================] - 0s 31ms/step
Month=113, Predicted=178.355328, Expected=175.000000
1/1 [==============================] - 0s 36ms/step
Month=114, Predicted=175.805514, Expected=176.210000
1/1 [==============================] - 0s 34ms/step
Month=115, Predicted=176.149232, Expected=176.820000
1/1 [==============================] - 0s 29ms/step
Month=116, Predicted=176.879271, Expected=176.670000
1/1 [==============================] - 0s 29ms/step
Month=117, Predicted=176.881320, Expected=175.030000
1/1 [==============================] - 0s 32ms/step
Month=118, Predicted=175.539418, Expected=176.940000
1/1 [==============================] - 0s 32ms/step
Month=119, Predicted=176.739186, Expected=179.980000
1/1 [==============================] - 0s 33ms/step
Month=120, Predicted=179.553113, Expected=181.720000
1/1 [==============================] - 0s 30ms/step
Month=121, Predicted=181.553197, Expected=179.970000
1/1 [==============================] - 0s 32ms/step
Month=122, Predicted=180.501425, Expected=178.440000
1/1 [==============================] - 0s 31ms/step
Month=123, Predicted=178.927410, Expected=178.650000
1/1 [==============================] - 0s 31ms/step
Month=124, Predicted=178.789297, Expected=178.020000
1/1 [==============================] - 0s 34ms/step
Month=125, Predicted=178.327352, Expected=175.300000
1/1 [==============================] - 0s 32ms/step
Month=126, Predicted=176.025488, Expected=175.240000
1/1 [==============================] - 0s 33ms/step
Month=127, Predicted=175.433315, Expected=171.270000
1/1 [==============================] - 0s 34ms/step
Month=128, Predicted=172.245569, Expected=168.850000
1/1 [==============================] - 0s 35ms/step
Month=129, Predicted=169.515468, Expected=164.940000
1/1 [==============================] - 0s 34ms/step
Month=130, Predicted=165.903565, Expected=172.770000
1/1 [==============================] - 0s 36ms/step
```

```
Month=131, Predicted=171.384800, Expected=168.340000
1/1 [==============================] - 0s 30ms/step
Month=132, Predicted=169.407599, Expected=166.480000
1/1 [==============================] - 0s 32ms/step
Month=133, Predicted=167.033432, Expected=167.780000
1/1 [==============================] - 0s 33ms/step
Month=134, Predicted=167.701226, Expected=167.780000
1/1 [==============================] - 0s 35ms/step
Month=135, Predicted=167.961311, Expected=166.680000
1/1 [==============================] - 0s 39ms/step
Month=136, Predicted=167.081382, Expected=168.390000
1/1 [==============================] - 0s 40ms/step
Month=137, Predicted=168.229199, Expected=171.610000
1/1 [==============================] - 0s 25ms/step
Month=138, Predicted=171.147101, Expected=172.800000
1/1 [==============================] - 0s 29ms/step
Month=139, Predicted=172.743233, Expected=168.380000
1/1 [==============================] - 0s 29ms/step
Month=140, Predicted=169.445599, Expected=170.050000
1/1 [==============================] - 0s 30ms/step
Month=141, Predicted=169.897202, Expected=173.250000
1/1 [==============================] - 0s 33ms/step
Month=142, Predicted=172.791102, Expected=172.440000
1/1 [==============================] - 0s 32ms/step
Month=143, Predicted=172.783363, Expected=174.140000
1/1 [==============================] - 0s 33ms/step
Month=144, Predicted=173.981200, Expected=174.730000
1/1 [==============================] - 0s 32ms/step
Month=145, Predicted=174.793272, Expected=175.820000
1/1 [==============================] - 0s 32ms/step
Month=146, Predicted=175.783240, Expected=178.240000
1/1 [==============================] - 0s 33ms/step
Month=147, Predicted=177.937153, Expected=177.840000
1/1 [==============================] - 0s 34ms/step
Month=148, Predicted=178.101337, Expected=172.800000
1/1 [==============================] - 0s 33ms/step
Month=149, Predicted=173.989639, Expected=165.720000
1/1 [==============================] - 0s 34ms/step
Month=150, Predicted=167.317772, Expected=165.240000
1/1 [==============================] - 0s 30ms/step
Month=151, Predicted=165.517342, Expected=162.940000
1/1 [==============================] - 0s 29ms/step
Month=152, Predicted=163.581461, Expected=163.650000
1/1 [==============================] - 0s 28ms/step
```

```
Month=153, Predicted=163.689264, Expected=164.220000
1/1 [==============================] - 0s 31ms/step
Month=154, Predicted=164.287274, Expected=162.320000
1/1 [==============================] - 0s 32ms/step
Month=155, Predicted=162.881434, Expected=165.260000
1/1 [==============================] - 0s 39ms/step
Month=156, Predicted=164.853119, Expected=169.100000
1/1 [==============================] - 0s 35ms/step
Month=157, Predicted=168.513061, Expected=176.570000
1/1 [==============================] - 0s 31ms/step
Month=158, Predicted=175.256824, Expected=176.890000
1/1 [==============================] - 0s 30ms/step
Month=159, Predicted=177.007290, Expected=183.830000
1/1 [==============================] - 0s 30ms/step
Month=160, Predicted=182.622859, Expected=185.160000
1/1 [==============================] - 0s 31ms/step
Month=161, Predicted=185.075224, Expected=186.050000
1/1 [==============================] - 0s 31ms/step
Month=162, Predicted=186.053253, Expected=187.360000
1/1 [==============================] - 0s 30ms/step
Month=163, Predicted=187.279225, Expected=190.040000
1/1 [==============================] - 0s 36ms/step
Month=164, Predicted=189.685136, Expected=188.590000
1/1 [==============================] - 0s 27ms/step
Month=165, Predicted=189.061405, Expected=188.150000
1/1 [==============================] - 0s 31ms/step
Month=166, Predicted=188.419339, Expected=186.440000
1/1 [==============================] - 0s 32ms/step
Month=167, Predicted=186.963422, Expected=188.180000
1/1 [==============================] - 0s 39ms/step
Month=168, Predicted=188.013197, Expected=186.990000
1/1 [==============================] - 0s 36ms/step
Month=169, Predicted=187.409388, Expected=186.310000
1/1 [==============================] - 0s 34ms/step
Month=170, Predicted=186.627355, Expected=187.630000
1/1 [==============================] - 0s 35ms/step
Month=171, Predicted=187.547225, Expected=187.160000
1/1 [==============================] - 0s 35ms/step
Month=172, Predicted=187.435341, Expected=188.360000
1/1 [==============================] - 0s 37ms/step
Month=173, Predicted=188.301232, Expected=188.150000
1/1 [==============================] - 0s 37ms/step
Month=174, Predicted=188.373324, Expected=188.580000
1/1 [==============================] - 0s 45ms/step
```

```
Month=175, Predicted=188.675283, Expected=188.580000
1/1 [==============================] - 0s 35ms/step
Month=176, Predicted=188.761311, Expected=187.900000
1/1 [==============================] - 0s 36ms/step
Month=177, Predicted=188.217355, Expected=187.500000
1/1 [==============================] - 0s 32ms/step
Month=178, Predicted=187.761337, Expected=186.870000
Test RMSE: 2.478
```

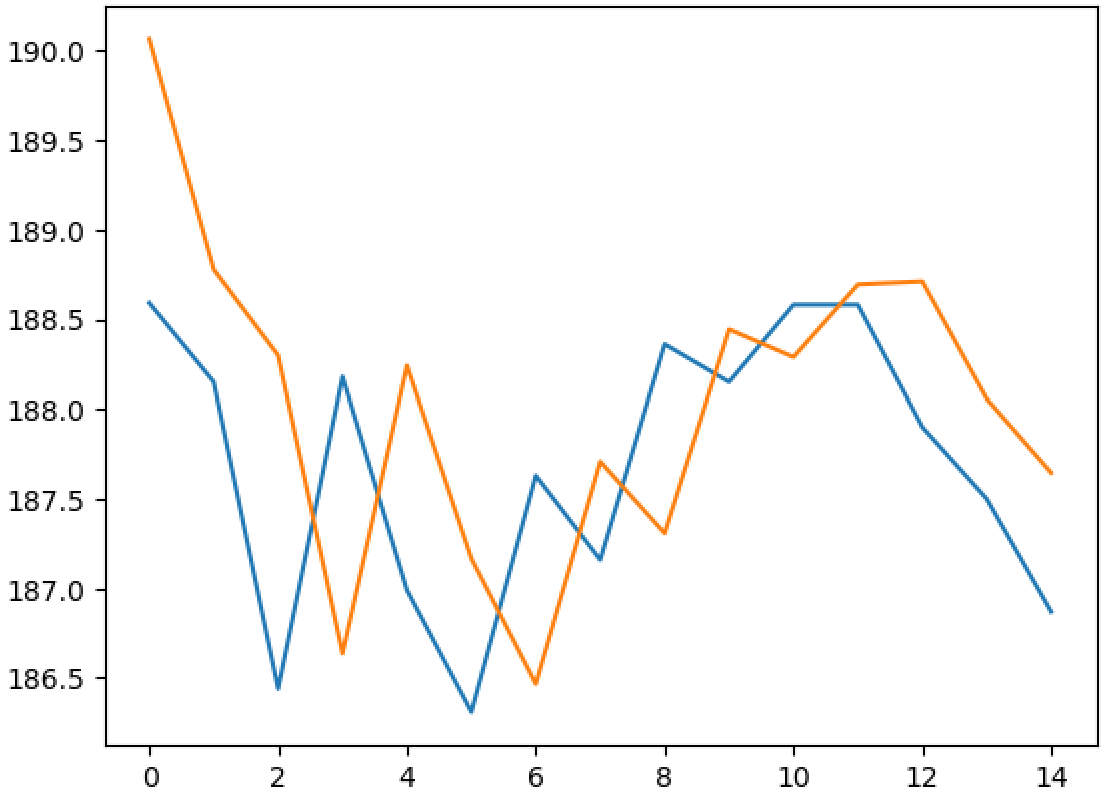


Over 15 test rows

In [27]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values
```

```
# split data
train, test = supervised_values[0:-15], supervised_values[-15:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
cnn_model = fit_cnn(train_scaled, 1, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
cnn_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_cnn(cnn_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-15:], predictions))
print('Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-15:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
875/875 [==============================] - 1s 2ms/step
1/1 [==============================] - 0s 37ms/step
Month=1, Predicted=190.064803, Expected=188.590000
1/1 [==============================] - 0s 32ms/step
Month=2, Predicted=188.776842, Expected=188.150000
1/1 [==============================] - 0s 30ms/step
Month=3, Predicted=188.297215, Expected=186.440000
1/1 [==============================] - 0s 27ms/step
Month=4, Predicted=186.637043, Expected=188.180000
1/1 [==============================] - 0s 29ms/step
Month=5, Predicted=188.241683, Expected=186.990000
1/1 [==============================] - 0s 32ms/step
Month=6, Predicted=187.166641, Expected=186.310000
1/1 [==============================] - 0s 32ms/step
Month=7, Predicted=186.466631, Expected=187.630000
1/1 [==============================] - 0s 33ms/step
Month=8, Predicted=187.708162, Expected=187.160000
1/1 [==============================] - 0s 34ms/step
Month=9, Predicted=187.308392, Expected=188.360000
1/1 [==============================] - 0s 34ms/step
Month=10, Predicted=188.442870, Expected=188.150000
1/1 [==============================] - 0s 30ms/step
Month=11, Predicted=188.288191, Expected=188.580000
1/1 [==============================] - 0s 30ms/step
Month=12, Predicted=188.693081, Expected=188.580000
1/1 [==============================] - 0s 31ms/step
Month=13, Predicted=188.709952, Expected=187.900000
1/1 [==============================] - 0s 32ms/step
Month=14, Predicted=188.056631, Expected=187.500000
1/1 [==============================] - 0s 32ms/step
Month=15, Predicted=187.645645, Expected=186.870000
Test RMSE: 1.007
```



Bi-directional LSTM

In [28]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-178], supervised_values[-178:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)
```

```
# fit the model
bilstm_model = fit_bilstm(train_scaled, 1, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
bilstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_bilstm(bilstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-178:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-178:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
712/712 [==============================] - 2s 2ms/step
1/1 [==============================] - 0s 27ms/step
Month=1, Predicted=150.857409, Expected=153.140000
1/1 [==============================] - 0s 32ms/step
Month=2, Predicted=153.294014, Expected=154.230000
1/1 [==============================] - 0s 36ms/step
Month=3, Predicted=154.347763, Expected=153.280000
1/1 [==============================] - 0s 30ms/step
Month=4, Predicted=153.464256, Expected=154.120000
1/1 [==============================] - 0s 34ms/step
Month=5, Predicted=154.264055, Expected=153.810000
1/1 [==============================] - 0s 33ms/step
Month=6, Predicted=153.980908, Expected=154.480000
1/1 [==============================] - 0s 31ms/step
Month=7, Predicted=154.623137, Expected=153.480000
1/1 [==============================] - 0s 31ms/step
Month=8, Predicted=153.676675, Expected=155.390000
1/1 [==============================] - 0s 34ms/step
Month=9, Predicted=155.492840, Expected=155.300000
1/1 [==============================] - 0s 30ms/step
Month=10, Predicted=155.443962, Expected=155.840000
1/1 [==============================] - 0s 31ms/step
Month=11, Predicted=155.973014, Expected=155.900000
1/1 [==============================] - 0s 28ms/step
Month=12, Predicted=156.048947, Expected=156.550000
1/1 [==============================] - 0s 29ms/step
Month=13, Predicted=156.681157, Expected=156.000000
1/1 [==============================] - 0s 31ms/step
Month=14, Predicted=156.173226, Expected=156.990000
1/1 [==============================] - 0s 37ms/step
Month=15, Predicted=157.120026, Expected=159.880000
1/1 [==============================] - 0s 32ms/step
Month=16, Predicted=159.888880, Expected=160.470000
1/1 [==============================] - 0s 33ms/step
Month=17, Predicted=160.543346, Expected=159.760000
1/1 [==============================] - 0s 33ms/step
Month=18, Predicted=159.910294, Expected=155.980000
1/1 [==============================] - 0s 29ms/step
Month=19, Predicted=156.259854, Expected=156.250000
1/1 [==============================] - 0s 32ms/step
Month=20, Predicted=156.498853, Expected=156.170000
1/1 [==============================] - 0s 32ms/step
```

```
Month=21, Predicted=156.386530, Expected=157.100000
1/1 [==============================] - 0s 37ms/step
Month=22, Predicted=157.258985, Expected=156.410000
1/1 [==============================] - 0s 29ms/step
Month=23, Predicted=156.603077, Expected=157.410000
1/1 [==============================] - 0s 34ms/step
Month=24, Predicted=157.552376, Expected=163.050000
1/1 [==============================] - 0s 32ms/step
Month=25, Predicted=162.902787, Expected=166.720000
1/1 [==============================] - 0s 32ms/step
Month=26, Predicted=166.564416, Expected=169.040000
1/1 [==============================] - 0s 30ms/step
Month=27, Predicted=168.945421, Expected=166.890000
1/1 [==============================] - 0s 40ms/step
Month=28, Predicted=167.034422, Expected=168.110000
1/1 [==============================] - 0s 31ms/step
Month=29, Predicted=168.208416, Expected=172.500000
1/1 [==============================] - 0s 30ms/step
Month=30, Predicted=172.400253, Expected=174.250000
1/1 [==============================] - 0s 28ms/step
Month=31, Predicted=174.220973, Expected=174.810000
1/1 [==============================] - 0s 31ms/step
Month=32, Predicted=174.856552, Expected=176.240000
1/1 [==============================] - 0s 28ms/step
Month=33, Predicted=176.273849, Expected=175.880000
1/1 [==============================] - 0s 33ms/step
Month=34, Predicted=175.998398, Expected=174.670000
1/1 [==============================] - 0s 31ms/step
Month=35, Predicted=174.857930, Expected=173.970000
1/1 [==============================] - 0s 32ms/step
Month=36, Predicted=174.177477, Expected=171.340000
1/1 [==============================] - 0s 30ms/step
Month=37, Predicted=171.617155, Expected=169.080000
1/1 [==============================] - 0s 32ms/step
Month=38, Predicted=169.392377, Expected=171.100000
1/1 [==============================] - 0s 30ms/step
Month=39, Predicted=171.292491, Expected=170.150000
1/1 [==============================] - 0s 30ms/step
Month=40, Predicted=170.367778, Expected=169.980000
1/1 [==============================] - 0s 30ms/step
Month=41, Predicted=170.186188, Expected=173.140000
1/1 [==============================] - 0s 32ms/step
Month=42, Predicted=173.184697, Expected=174.960000
1/1 [==============================] - 0s 31ms/step
```

```
Month=43, Predicted=174.988923, Expected=174.960000
1/1 [==============================] - 0s 30ms/step
Month=44, Predicted=175.064707, Expected=174.970000
1/1 [==============================] - 0s 32ms/step
Month=45, Predicted=175.104008, Expected=174.090000
1/1 [==============================] - 0s 30ms/step
Month=46, Predicted=174.275814, Expected=173.070000
1/1 [==============================] - 0s 33ms/step
Month=47, Predicted=173.288043, Expected=169.480000
1/1 [==============================] - 0s 29ms/step
Month=48, Predicted=169.785756, Expected=171.850000
1/1 [==============================] - 0s 30ms/step
Month=49, Predicted=172.037224, Expected=171.050000
1/1 [==============================] - 0s 31ms/step
Month=50, Predicted=171.259969, Expected=169.800000
1/1 [==============================] - 0s 31ms/step
Month=51, Predicted=170.039409, Expected=169.640000
1/1 [==============================] - 0s 33ms/step
Month=52, Predicted=169.859869, Expected=169.010000
1/1 [==============================] - 0s 32ms/step
Month=53, Predicted=169.232598, Expected=169.320000
1/1 [==============================] - 0s 36ms/step
Month=54, Predicted=169.509856, Expected=169.370000
1/1 [==============================] - 0s 29ms/step
Month=55, Predicted=169.550200, Expected=172.670000
1/1 [==============================] - 0s 31ms/step
Month=56, Predicted=172.688598, Expected=171.700000
1/1 [==============================] - 0s 31ms/step
Month=57, Predicted=171.850851, Expected=172.270000
1/1 [==============================] - 0s 31ms/step
Month=58, Predicted=172.404471, Expected=172.220000
1/1 [==============================] - 0s 28ms/step
Month=59, Predicted=172.374318, Expected=173.970000
1/1 [==============================] - 0s 37ms/step
Month=60, Predicted=174.052254, Expected=176.420000
1/1 [==============================] - 0s 31ms/step
Month=61, Predicted=176.427225, Expected=174.540000
1/1 [==============================] - 0s 30ms/step
Month=62, Predicted=174.716400, Expected=174.350000
1/1 [==============================] - 0s 30ms/step
Month=63, Predicted=174.531984, Expected=175.010000
1/1 [==============================] - 0s 29ms/step
Month=64, Predicted=175.159703, Expected=175.010000
1/1 [==============================] - 0s 31ms/step
```

```
Month=65, Predicted=175.170174, Expected=175.010000
1/1 [==============================] - 0s 30ms/step
Month=66, Predicted=175.175282, Expected=170.570000
1/1 [==============================] - 0s 29ms/step
Month=67, Predicted=170.870743, Expected=170.600000
1/1 [==============================] - 0s 34ms/step
Month=68, Predicted=170.881010, Expected=171.080000
1/1 [==============================] - 0s 29ms/step
Month=69, Predicted=171.294810, Expected=169.230000
1/1 [==============================] - 0s 30ms/step
Month=70, Predicted=169.488699, Expected=169.230000
1/1 [==============================] - 0s 33ms/step
Month=71, Predicted=169.459085, Expected=172.260000
1/1 [==============================] - 0s 30ms/step
Month=72, Predicted=172.327214, Expected=172.230000
1/1 [==============================] - 0s 32ms/step
Month=73, Predicted=172.359743, Expected=173.030000
1/1 [==============================] - 0s 32ms/step
Month=74, Predicted=173.143125, Expected=175.000000
1/1 [==============================] - 0s 32ms/step
Month=75, Predicted=175.047658, Expected=174.350000
1/1 [==============================] - 0s 31ms/step
Month=76, Predicted=174.492548, Expected=174.330000
1/1 [==============================] - 0s 34ms/step
Month=77, Predicted=174.485196, Expected=174.290000
1/1 [==============================] - 0s 30ms/step
Month=78, Predicted=174.454049, Expected=175.280000
1/1 [==============================] - 0s 33ms/step
Month=79, Predicted=175.404470, Expected=177.090000
1/1 [==============================] - 0s 30ms/step
Month=80, Predicted=177.152998, Expected=177.090000
1/1 [==============================] - 0s 29ms/step
Month=81, Predicted=177.210481, Expected=176.190000
1/1 [==============================] - 0s 29ms/step
Month=82, Predicted=176.369682, Expected=179.100000
1/1 [==============================] - 0s 29ms/step
Month=83, Predicted=179.139579, Expected=179.260000
1/1 [==============================] - 0s 29ms/step
Month=84, Predicted=179.367320, Expected=178.460000
1/1 [==============================] - 0s 28ms/step
Month=85, Predicted=178.629350, Expected=177.000000
1/1 [==============================] - 0s 30ms/step
Month=86, Predicted=177.223998, Expected=177.040000
1/1 [==============================] - 0s 30ms/step
```

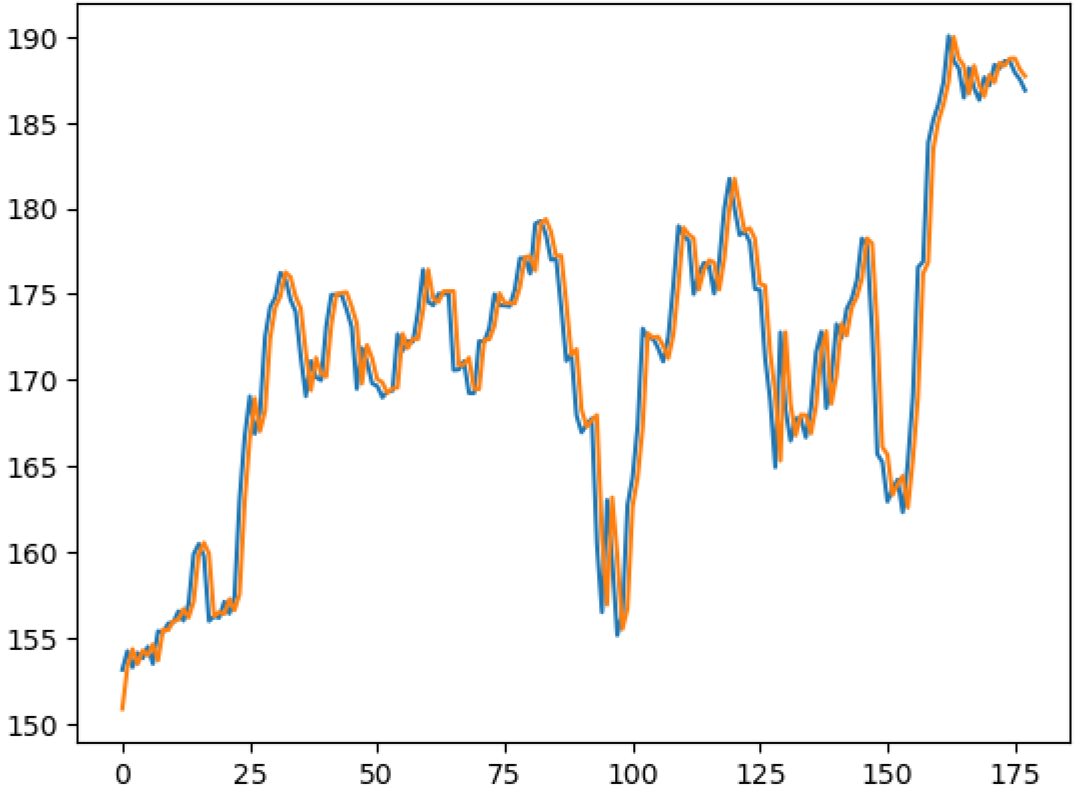
```
Month=87, Predicted=177.243153, Expected=174.220000
1/1 [==============================] - 0s 30ms/step
Month=88, Predicted=174.500069, Expected=171.110000
1/1 [==============================] - 0s 30ms/step
Month=89, Predicted=171.443784, Expected=171.510000
1/1 [==============================] - 0s 32ms/step
Month=90, Predicted=171.790146, Expected=167.960000
1/1 [==============================] - 0s 30ms/step
Month=91, Predicted=168.287831, Expected=166.970000
1/1 [==============================] - 0s 35ms/step
Month=92, Predicted=167.291957, Expected=167.430000
1/1 [==============================] - 0s 35ms/step
Month=93, Predicted=167.677827, Expected=167.780000
1/1 [==============================] - 0s 31ms/step
Month=94, Predicted=167.980007, Expected=160.500000
1/1 [==============================] - 0s 30ms/step
Month=95, Predicted=160.849476, Expected=156.490000
1/1 [==============================] - 0s 33ms/step
Month=96, Predicted=156.928884, Expected=163.030000
1/1 [==============================] - 0s 29ms/step
Month=97, Predicted=163.181504, Expected=159.540000
1/1 [==============================] - 0s 31ms/step
Month=98, Predicted=159.818890, Expected=155.150000
1/1 [==============================] - 0s 32ms/step
Month=99, Predicted=155.507983, Expected=156.410000
1/1 [==============================] - 0s 33ms/step
Month=100, Predicted=156.701788, Expected=162.710000
1/1 [==============================] - 0s 31ms/step
Month=101, Predicted=162.633589, Expected=164.340000
1/1 [==============================] - 0s 30ms/step
Month=102, Predicted=164.344390, Expected=167.370000
1/1 [==============================] - 0s 32ms/step
Month=103, Predicted=167.297394, Expected=172.990000
1/1 [==============================] - 0s 31ms/step
Month=104, Predicted=172.712787, Expected=172.430000
1/1 [==============================] - 0s 32ms/step
Month=105, Predicted=172.468535, Expected=172.430000
1/1 [==============================] - 0s 39ms/step
Month=106, Predicted=172.522765, Expected=171.850000
1/1 [==============================] - 0s 29ms/step
Month=107, Predicted=172.000477, Expected=171.070000
1/1 [==============================] - 0s 29ms/step
Month=108, Predicted=171.258632, Expected=172.500000
1/1 [==============================] - 0s 29ms/step
```

```
Month=109, Predicted=172.619448, Expected=175.500000
1/1 [==============================] - 0s 30ms/step
Month=110, Predicted=175.496652, Expected=178.970000
1/1 [==============================] - 0s 32ms/step
Month=111, Predicted=178.875880, Expected=178.390000
1/1 [==============================] - 0s 30ms/step
Month=112, Predicted=178.477982, Expected=178.120000
1/1 [==============================] - 0s 32ms/step
Month=113, Predicted=178.253578, Expected=175.000000
1/1 [==============================] - 0s 30ms/step
Month=114, Predicted=175.255838, Expected=176.210000
1/1 [==============================] - 0s 31ms/step
Month=115, Predicted=176.398309, Expected=176.820000
1/1 [==============================] - 0s 29ms/step
Month=116, Predicted=176.974337, Expected=176.670000
1/1 [==============================] - 0s 30ms/step
Month=117, Predicted=176.838589, Expected=175.030000
1/1 [==============================] - 0s 30ms/step
Month=118, Predicted=175.259959, Expected=176.940000
1/1 [==============================] - 0s 34ms/step
Month=119, Predicted=177.068683, Expected=179.980000
1/1 [==============================] - 0s 32ms/step
Month=120, Predicted=179.981884, Expected=181.720000
1/1 [==============================] - 0s 30ms/step
Month=121, Predicted=181.731142, Expected=179.970000
1/1 [==============================] - 0s 29ms/step
Month=122, Predicted=180.139007, Expected=178.440000
1/1 [==============================] - 0s 30ms/step
Month=123, Predicted=178.665813, Expected=178.650000
1/1 [==============================] - 0s 31ms/step
Month=124, Predicted=178.847940, Expected=178.020000
1/1 [==============================] - 0s 31ms/step
Month=125, Predicted=178.229799, Expected=175.300000
1/1 [==============================] - 0s 35ms/step
Month=126, Predicted=175.581429, Expected=175.240000
1/1 [==============================] - 0s 33ms/step
Month=127, Predicted=175.492423, Expected=171.270000
1/1 [==============================] - 0s 30ms/step
Month=128, Predicted=171.595922, Expected=168.850000
1/1 [==============================] - 0s 31ms/step
Month=129, Predicted=169.207736, Expected=164.940000
1/1 [==============================] - 0s 40ms/step
Month=130, Predicted=165.324708, Expected=172.770000
1/1 [==============================] - 0s 29ms/step
```

```
Month=131, Predicted=172.780585, Expected=168.340000
1/1 [==============================] - 0s 31ms/step
Month=132, Predicted=168.610899, Expected=166.480000
1/1 [==============================] - 0s 30ms/step
Month=133, Predicted=166.790699, Expected=167.780000
1/1 [==============================] - 0s 31ms/step
Month=134, Predicted=167.996385, Expected=167.780000
1/1 [==============================] - 0s 33ms/step
Month=135, Predicted=167.974424, Expected=166.680000
1/1 [==============================] - 0s 29ms/step
Month=136, Predicted=166.905396, Expected=168.390000
1/1 [==============================] - 0s 30ms/step
Month=137, Predicted=168.522705, Expected=171.610000
1/1 [==============================] - 0s 31ms/step
Month=138, Predicted=171.604019, Expected=172.800000
1/1 [==============================] - 0s 39ms/step
Month=139, Predicted=172.838551, Expected=168.380000
1/1 [==============================] - 0s 36ms/step
Month=140, Predicted=168.638097, Expected=170.050000
1/1 [==============================] - 0s 29ms/step
Month=141, Predicted=170.230960, Expected=173.250000
1/1 [==============================] - 0s 29ms/step
Month=142, Predicted=173.273689, Expected=172.440000
1/1 [==============================] - 0s 32ms/step
Month=143, Predicted=172.584551, Expected=174.140000
1/1 [==============================] - 0s 33ms/step
Month=144, Predicted=174.217135, Expected=174.730000
1/1 [==============================] - 0s 31ms/step
Month=145, Predicted=174.829354, Expected=175.820000
1/1 [==============================] - 0s 30ms/step
Month=146, Predicted=175.902950, Expected=178.240000
1/1 [==============================] - 0s 29ms/step
Month=147, Predicted=178.245992, Expected=177.840000
1/1 [==============================] - 0s 32ms/step
Month=148, Predicted=177.954893, Expected=172.800000
1/1 [==============================] - 0s 32ms/step
Month=149, Predicted=173.091757, Expected=165.720000
1/1 [==============================] - 0s 30ms/step
Month=150, Predicted=166.113403, Expected=165.240000
1/1 [==============================] - 0s 30ms/step
Month=151, Predicted=165.675971, Expected=162.940000
1/1 [==============================] - 0s 35ms/step
Month=152, Predicted=163.317747, Expected=163.650000
1/1 [==============================] - 0s 29ms/step
```

```
Month=153, Predicted=163.942840, Expected=164.220000
1/1 [==============================] - 0s 31ms/step
Month=154, Predicted=164.434906, Expected=162.320000
1/1 [==============================] - 0s 31ms/step
Month=155, Predicted=162.580738, Expected=165.260000
1/1 [==============================] - 0s 30ms/step
Month=156, Predicted=165.364548, Expected=169.100000
1/1 [==============================] - 0s 30ms/step
Month=157, Predicted=169.045175, Expected=176.570000
1/1 [==============================] - 0s 31ms/step
Month=158, Predicted=176.180966, Expected=176.890000
1/1 [==============================] - 0s 33ms/step
Month=159, Predicted=176.865097, Expected=183.830000
1/1 [==============================] - 0s 30ms/step
Month=160, Predicted=183.460986, Expected=185.160000
1/1 [==============================] - 0s 30ms/step
Month=161, Predicted=185.071049, Expected=186.050000
1/1 [==============================] - 0s 31ms/step
Month=162, Predicted=186.042804, Expected=187.360000
1/1 [==============================] - 0s 32ms/step
Month=163, Predicted=187.369960, Expected=190.040000
1/1 [==============================] - 0s 33ms/step
Month=164, Predicted=189.986419, Expected=188.590000
1/1 [==============================] - 0s 35ms/step
Month=165, Predicted=188.722281, Expected=188.150000
1/1 [==============================] - 0s 30ms/step
Month=166, Predicted=188.313389, Expected=186.440000
1/1 [==============================] - 0s 30ms/step
Month=167, Predicted=186.667396, Expected=188.180000
1/1 [==============================] - 0s 33ms/step
Month=168, Predicted=188.315112, Expected=186.990000
1/1 [==============================] - 0s 30ms/step
Month=169, Predicted=187.189592, Expected=186.310000
1/1 [==============================] - 0s 29ms/step
Month=170, Predicted=186.523953, Expected=187.630000
1/1 [==============================] - 0s 29ms/step
Month=171, Predicted=187.771310, Expected=187.160000
1/1 [==============================] - 0s 31ms/step
Month=172, Predicted=187.335930, Expected=188.360000
1/1 [==============================] - 0s 29ms/step
Month=173, Predicted=188.482099, Expected=188.150000
1/1 [==============================] - 0s 31ms/step
Month=174, Predicted=188.306083, Expected=188.580000
1/1 [==============================] - 0s 31ms/step
```

```
Month=175, Predicted=188.724659, Expected=188.580000
1/1 [==============================] - 0s 33ms/step
Month=176, Predicted=188.737415, Expected=187.900000
1/1 [==============================] - 0s 30ms/step
Month=177, Predicted=188.090766, Expected=187.500000
1/1 [==============================] - 0s 32ms/step
Month=178, Predicted=187.698435, Expected=186.870000

Test RMSE: 2.392
```

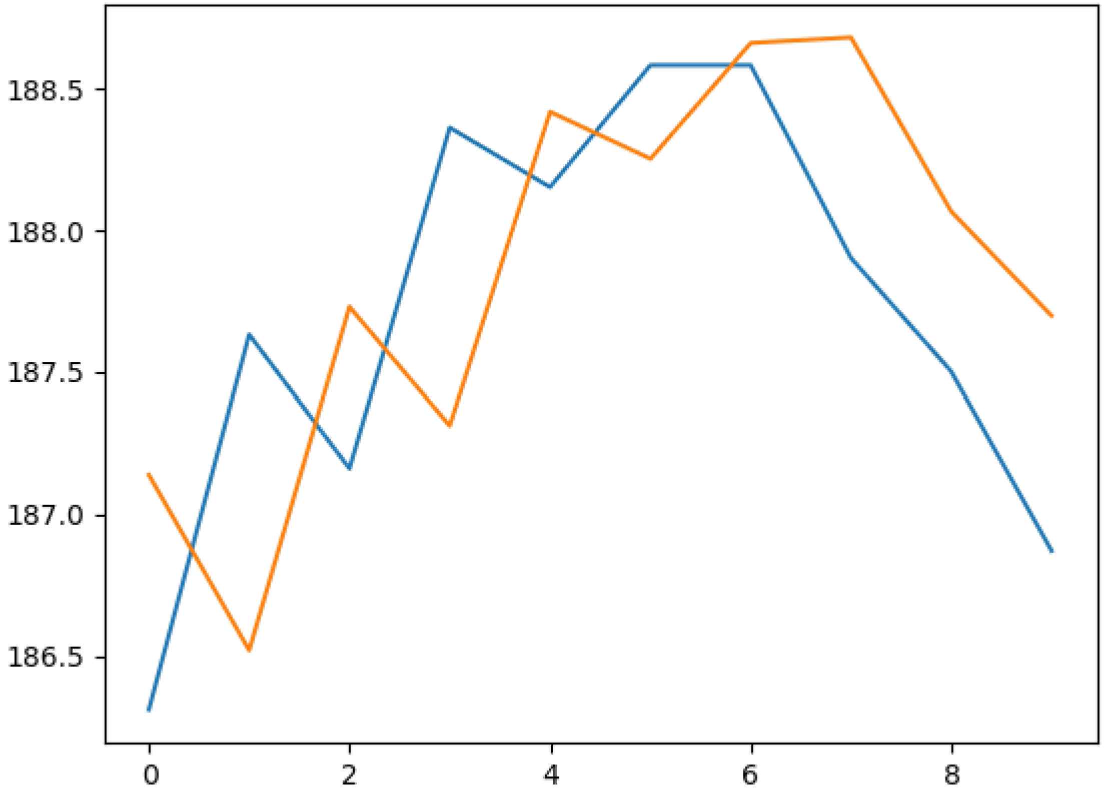


Over 15 test rows

In [29]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
```

```
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-10], supervised_values[-10:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
bilstm_model = fit_bilstm(train_scaled, 1, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
bilstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_bilstm(bilstm_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-10:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-10:])
pyplot.plot(predictions)
pyplot.show()
```

```
X.shape[1]),  1
X.shape[2])  1
880/880 [==============================] - 3s 2ms/step
1/1 [==============================] - 0s 29ms/step
Month=1, Predicted=187.137314, Expected=186.310000
1/1 [==============================] - 0s 28ms/step
Month=2, Predicted=186.518677, Expected=187.630000
1/1 [==============================] - 0s 27ms/step
Month=3, Predicted=187.728263, Expected=187.160000
1/1 [==============================] - 0s 31ms/step
Month=4, Predicted=187.309943, Expected=188.360000
1/1 [==============================] - 0s 30ms/step
Month=5, Predicted=188.415160, Expected=188.150000
1/1 [==============================] - 0s 28ms/step
Month=6, Predicted=188.249773, Expected=188.580000
1/1 [==============================] - 0s 31ms/step
Month=7, Predicted=188.658451, Expected=188.580000
1/1 [==============================] - 0s 31ms/step
Month=8, Predicted=188.678007, Expected=187.900000
1/1 [==============================] - 0s 30ms/step
Month=9, Predicted=188.064169, Expected=187.500000
1/1 [==============================] - 0s 27ms/step
Month=10, Predicted=187.696476, Expected=186.870000

 Test RMSE: 0.717
```



Arima Model

In [30]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in c:\users\jaypa\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: numpy>=1.17 in c:\users\jaypa\anaconda3\lib\site-packages (from statsmodels) (1.22.4)
Requirement already satisfied: scipy>=1.3 in c:\users\jaypa\anaconda3\lib\site-packages (from statsmodels) (1.9.1)
Requirement already satisfied: pandas>=0.25 in c:\users\jaypa\anaconda3\lib\site-packages (from statsmodels) (1.4.4)
Requirement already satisfied: patsy>=0.5.2 in c:\users\jaypa\anaconda3\lib\site-packages (from statsmodels) (0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\users\jaypa\anaconda3\lib\site-packages (from statsmodels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\jaypa\anaconda3\lib\site-packages (from packaging>=21.3->sta
tsmodels) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\jaypa\anaconda3\lib\site-packages (from pandas>=0.25->statsmodels) (202
2.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\jaypa\anaconda3\lib\site-packages (from pandas>=0.25->statsmod
els) (2.8.2)
Requirement already satisfied: six in c:\users\jaypa\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [31]:

```python
import numpy as np
data_stock_np = np.array(data_stock)
```

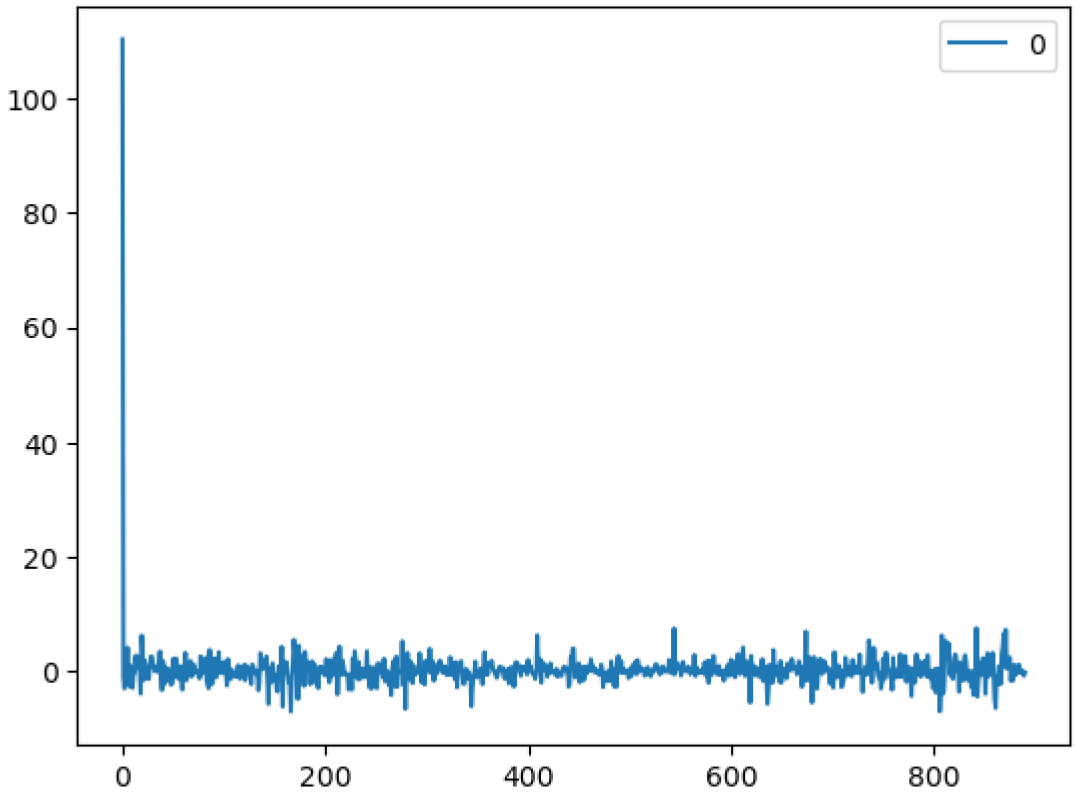
In [32]:

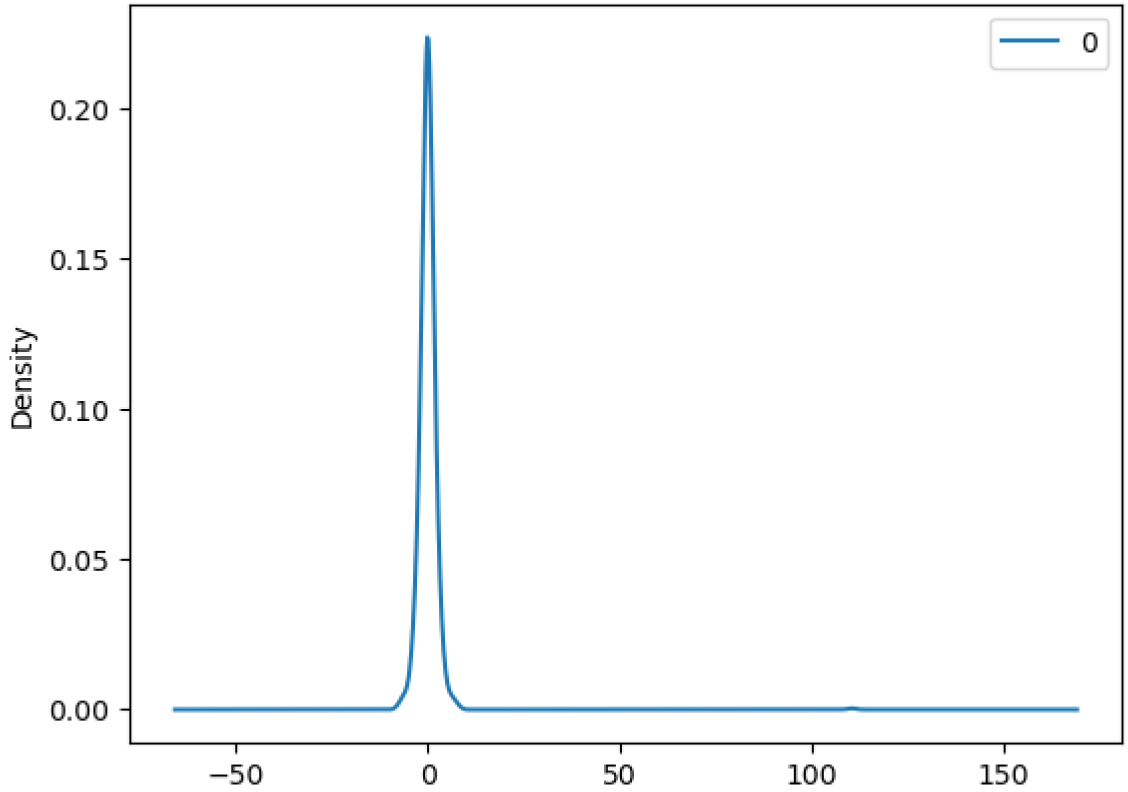
```python
from statsmodels.tsa.arima.model import ARIMA
import statsmodels.api as sm
import numpy as np
from matplotlib import pyplot

#data = data[['Date']]
data = np.asarray(data_stock_np)
```

In [33]:

```python
model = ARIMA(data, order=(5,1,5))

model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
# density plot of residuals
residuals.plot(kind='kde')
pyplot.show()
# summary stats of residuals
print(residuals.describe())
```

```
C:\Users\jaypa\anaconda3\lib\site-packages\statsmodels\base\model.py:604: ConvergenceWarning: Maximum Likelihood optimization fa
iled to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  891
Model:                 ARIMA(5, 1, 5)   Log Likelihood               -1798.906
Date:                Fri, 15 Dec 2023   AIC                           3619.812
Time:                        21:27:53   BIC                           3672.515
Sample:                             0   HQIC                          3639.955
                                - 891
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.3001      0.683     -1.903      0.057      -2.639       0.039
ar.L2         -0.8280      1.516     -0.546      0.585      -3.800       2.144
ar.L3          0.5055      1.932      0.262      0.794      -3.282       4.293
ar.L4          0.8613      1.358      0.634      0.526      -1.801       3.524
ar.L5          0.7209      0.574      1.257      0.209      -0.404       1.845
ma.L1          1.3449      0.687      1.957      0.050      -0.002       2.692
ma.L2          0.8590      1.558      0.551      0.581      -2.194       3.912
ma.L3         -0.5324      1.999     -0.266      0.790      -4.451       3.386
ma.L4         -0.8748      1.394     -0.627      0.530      -3.607       1.858
ma.L5         -0.7011      0.578     -1.213      0.225      -1.834       0.432
sigma2         3.3431      0.111     30.180      0.000       3.126       3.560
===================================================================================
Ljung-Box (L1) (Q):                   0.46   Jarque-Bera (JB):               236.46
Prob(Q):                              0.50   Prob(JB):                         0.00
Heteroskedasticity (H):               1.18   Skew:                             0.03
Prob(H) (two-sided):                  0.15   Kurtosis:                         5.52
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```


0
100
80
60
40
20
0
0
200
400
600
800




```
              0
count  891.000000
mean     0.205665
std      4.121053
min     -7.087373
25%     -0.734104
50%      0.052547
75%      0.965916
max    110.380000
```

In [34]:
```
from math import sqrt
from sklearn.metrics import mean_squared_error
X = data
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
```

```
for t in range(len(test)):
        model = ARIMA(history, order=(5,1,0))
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))
# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)

pyplot.plot(predictions, color='red')
pyplot.show()
```

```
predicted=143.608176, expected=144.770000
predicted=144.784362, expected=144.020000
predicted=143.977548, expected=143.660000
predicted=143.644291, expected=143.340000
predicted=143.377365, expected=143.170000
predicted=143.160002, expected=141.630000
predicted=141.657881, expected=141.800000
predicted=141.878979, expected=141.050000
predicted=141.114360, expected=141.050000
predicted=141.078864, expected=141.830000
predicted=141.928846, expected=141.200000
predicted=141.160754, expected=140.680000
predicted=140.688279, expected=142.440000
predicted=142.505570, expected=142.270000
predicted=142.200036, expected=143.640000
predicted=143.610734, expected=144.530000
predicted=144.512529, expected=143.680000
predicted=143.517691, expected=143.790000
predicted=143.783120, expected=143.650000
predicted=143.626870, expected=146.580000
predicted=146.588935, expected=147.510000
predicted=147.453348, expected=147.060000
predicted=146.890607, expected=146.530000
predicted=146.490825, expected=148.960000
predicted=148.934495, expected=153.010000
predicted=152.993657, expected=153.990000
predicted=153.768625, expected=153.260000
predicted=153.048311, expected=153.950000
predicted=153.905016, expected=156.100000
predicted=156.071392, expected=155.700000
predicted=155.553400, expected=155.470000
predicted=155.415367, expected=150.250000
predicted=150.164348, expected=152.540000
predicted=152.728483, expected=153.060000
predicted=153.238782, expected=153.990000
predicted=153.823913, expected=153.800000
predicted=153.861755, expected=153.340000
predicted=153.256333, expected=153.870000
predicted=153.904007, expected=153.610000
predicted=153.591962, expected=153.610000
predicted=153.592076, expected=153.670000
predicted=153.696654, expected=152.760000
predicted=152.732906, expected=153.180000
predicted=153.220893, expected=155.450000
```

```
predicted=155.514709, expected=153.930000
predicted=153.793858, expected=154.450000
predicted=154.421143, expected=155.370000
predicted=155.446025, expected=154.990000
predicted=154.873265, expected=148.980000
predicted=148.910361, expected=145.420000
predicted=145.591930, expected=146.590000
predicted=147.055563, expected=145.160000
predicted=145.168058, expected=144.290000
predicted=144.359509, expected=142.270000
predicted=142.397389, expected=146.340000
predicted=146.487528, expected=145.010000
predicted=144.937566, expected=145.870000
predicted=145.711761, expected=145.630000
predicted=145.806805, expected=146.280000
predicted=146.117969, expected=145.820000
predicted=145.857524, expected=143.730000
predicted=143.652456, expected=145.830000
predicted=145.983037, expected=143.680000
predicted=143.653894, expected=144.020000
predicted=143.962008, expected=143.500000
predicted=143.684130, expected=143.500000
predicted=143.403557, expected=144.090000
predicted=144.177328, expected=142.730000
predicted=142.676737, expected=144.180000
predicted=144.222570, expected=145.060000
predicted=145.095783, expected=145.530000
predicted=145.394309, expected=145.740000
predicted=145.743037, expected=147.770000
predicted=147.751484, expected=149.040000
predicted=148.970783, expected=149.560000
predicted=149.426942, expected=150.080000
predicted=150.056443, expected=151.020000
predicted=150.982990, expected=150.340000
predicted=150.263560, expected=150.270000
predicted=150.250632, expected=152.090000
predicted=152.158403, expected=152.740000
predicted=152.655729, expected=153.460000
predicted=153.380697, expected=150.560000
predicted=150.516177, expected=149.500000
predicted=149.529135, expected=148.730000
predicted=148.894920, expected=150.050000
predicted=150.061674, expected=157.140000
predicted=157.298472, expected=155.570000
```

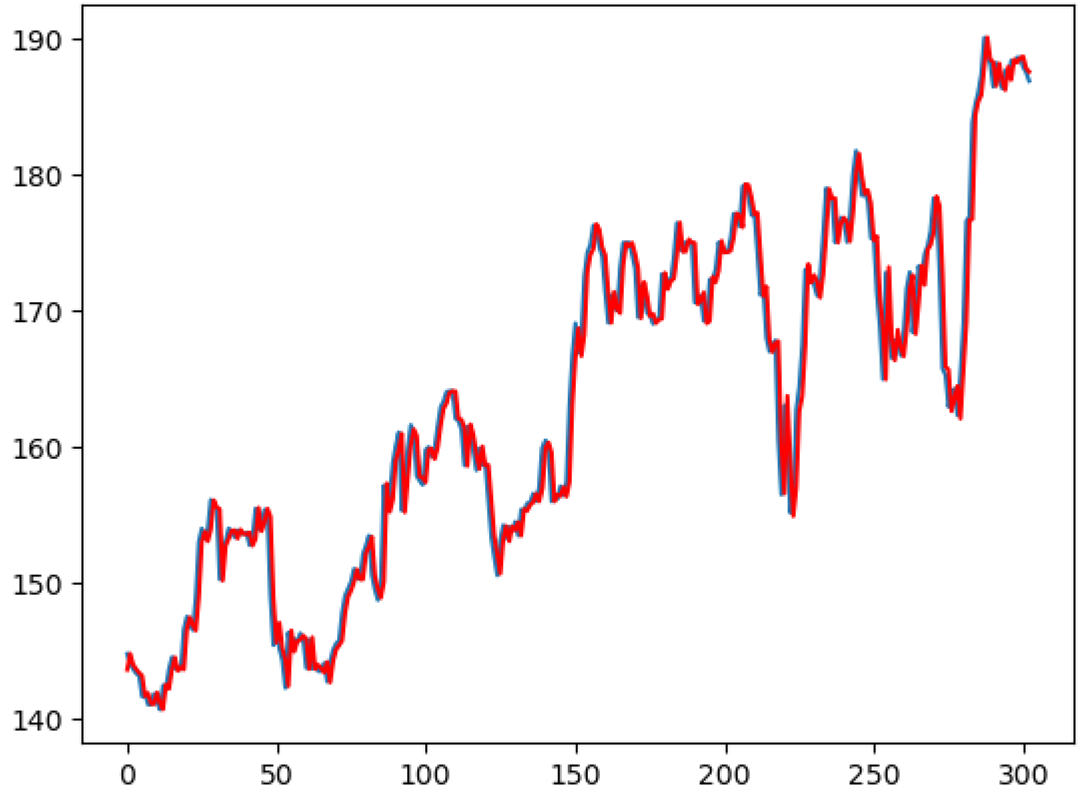
```
predicted=155.223972, expected=156.390000
predicted=156.113198, expected=158.810000
predicted=158.964885, expected=160.080000
predicted=159.844606, expected=161.060000
predicted=160.962914, expected=155.320000
predicted=155.190061, expected=157.480000
predicted=157.657067, expected=159.850000
predicted=160.060587, expected=161.600000
predicted=161.292528, expected=160.950000
predicted=160.808186, expected=157.860000
predicted=157.822865, expected=157.500000
predicted=157.709911, expected=157.210000
predicted=157.389418, expected=159.780000
predicted=159.779478, expected=159.980000
predicted=159.860220, expected=159.270000
predicted=159.097755, expected=159.860000
predicted=159.924039, expected=161.470000
predicted=161.526823, expected=162.910000
predicted=162.812421, expected=163.350000
predicted=163.206146, expected=164.000000
predicted=163.944490, expected=164.050000
predicted=164.042793, expected=164.050000
predicted=164.034563, expected=162.080000
predicted=162.070361, expected=161.910000
predicted=161.996565, expected=161.260000
predicted=161.372041, expected=158.630000
predicted=158.595545, expected=161.500000
predicted=161.650501, expected=160.860000
predicted=160.870034, expected=159.650000
predicted=159.468123, expected=158.280000
predicted=158.377098, expected=159.880000
predicted=160.028153, expected=158.670000
predicted=158.641016, expected=158.730000
predicted=158.659503, expected=156.070000
predicted=156.142117, expected=153.390000
predicted=153.457393, expected=151.890000
predicted=152.110045, expected=150.550000
predicted=150.671360, expected=153.140000
predicted=153.233350, expected=154.230000
predicted=154.173851, expected=153.280000
predicted=153.043985, expected=154.120000
predicted=154.138635, expected=153.810000
predicted=153.864596, expected=154.480000
predicted=154.448764, expected=153.480000
```

```
predicted=153.465203, expected=155.390000
predicted=155.412248, expected=155.300000
predicted=155.291327, expected=155.840000
predicted=155.729683, expected=155.900000
predicted=155.910698, expected=156.550000
predicted=156.537241, expected=156.000000
predicted=155.975711, expected=156.990000
predicted=156.988816, expected=159.880000
predicted=159.918674, expected=160.470000
predicted=160.308911, expected=159.760000
predicted=159.582725, expected=155.980000
predicted=155.980134, expected=156.250000
predicted=156.479871, expected=156.170000
predicted=156.378974, expected=157.100000
predicted=157.027632, expected=156.410000
predicted=156.343657, expected=157.410000
predicted=157.391133, expected=163.050000
predicted=163.150562, expected=166.720000
predicted=166.524634, expected=169.040000
predicted=168.673134, expected=166.890000
predicted=166.635780, expected=168.110000
predicted=168.208157, expected=172.500000
predicted=172.845619, expected=174.250000
predicted=174.062989, expected=174.810000
predicted=174.504634, expected=176.240000
predicted=176.342418, expected=175.880000
predicted=175.942899, expected=174.670000
predicted=174.617099, expected=173.970000
predicted=174.061660, expected=171.340000
predicted=171.356851, expected=169.080000
predicted=169.078673, expected=171.100000
predicted=171.347921, expected=170.150000
predicted=170.083702, expected=169.980000
predicted=169.792521, expected=173.140000
predicted=173.315315, expected=174.960000
predicted=174.918953, expected=174.960000
predicted=174.723959, expected=174.970000
predicted=174.961691, expected=174.090000
predicted=174.140242, expected=173.070000
predicted=173.082228, expected=169.480000
predicted=169.424716, expected=171.850000
predicted=172.085613, expected=171.050000
predicted=171.095977, expected=169.800000
predicted=169.574254, expected=169.640000
```

```
predicted=169.748975, expected=169.010000
predicted=169.056409, expected=169.320000
predicted=169.322226, expected=169.370000
predicted=169.376246, expected=172.670000
predicted=172.745127, expected=171.700000
predicted=171.558394, expected=172.270000
predicted=172.159592, expected=172.220000
predicted=172.327775, expected=173.970000
predicted=173.994698, expected=176.420000
predicted=176.450407, expected=174.540000
predicted=174.318845, expected=174.350000
predicted=174.329806, expected=175.010000
predicted=175.210343, expected=175.010000
predicted=174.977154, expected=175.010000
predicted=174.951830, expected=170.570000
predicted=170.448733, expected=170.600000
predicted=170.761967, expected=171.080000
predicted=171.313344, expected=169.230000
predicted=169.044262, expected=169.230000
predicted=169.205224, expected=172.260000
predicted=172.446143, expected=172.230000
predicted=172.073558, expected=173.030000
predicted=172.901721, expected=175.000000
predicted=175.105550, expected=174.350000
predicted=174.247722, expected=174.330000
predicted=174.289005, expected=174.290000
predicted=174.368797, expected=175.280000
predicted=175.312432, expected=177.090000
predicted=177.103726, expected=177.090000
predicted=176.980840, expected=176.190000
predicted=176.108623, expected=179.100000
predicted=179.273490, expected=179.260000
predicted=179.210073, expected=178.460000
predicted=178.292349, expected=177.000000
predicted=177.045540, expected=177.040000
predicted=177.160714, expected=174.220000
predicted=174.174416, expected=171.110000
predicted=171.064688, expected=171.510000
predicted=171.741778, expected=167.960000
predicted=167.865063, expected=166.970000
predicted=166.932906, expected=167.430000
predicted=167.582725, expected=167.780000
predicted=167.718477, expected=160.500000
predicted=160.176407, expected=156.490000
```

```
predicted=156.565135, expected=163.030000
predicted=163.706095, expected=159.540000
predicted=158.937960, expected=155.150000
predicted=154.902993, expected=156.410000
predicted=157.028934, expected=162.710000
predicted=162.616797, expected=164.340000
predicted=163.693683, expected=167.370000
predicted=167.388306, expected=172.990000
predicted=173.401282, expected=172.430000
predicted=172.037684, expected=172.430000
predicted=172.581878, expected=171.850000
predicted=172.073463, expected=171.070000
predicted=170.930845, expected=172.500000
predicted=172.617247, expected=175.500000
predicted=175.522977, expected=178.970000
predicted=178.929378, expected=178.390000
predicted=178.231151, expected=178.120000
predicted=178.247289, expected=175.000000
predicted=174.979272, expected=176.210000
predicted=176.357550, expected=176.820000
predicted=176.808474, expected=176.670000
predicted=176.468943, expected=175.030000
predicted=175.068518, expected=176.940000
predicted=177.131069, expected=179.980000
predicted=180.009557, expected=181.720000
predicted=181.523586, expected=179.970000
predicted=179.870471, expected=178.440000
predicted=178.571190, expected=178.650000
predicted=178.818452, expected=178.020000
predicted=177.878024, expected=175.300000
predicted=175.152800, expected=175.240000
predicted=175.451087, expected=171.270000
predicted=171.071466, expected=168.850000
predicted=168.813734, expected=164.940000
predicted=164.920258, expected=172.770000
predicted=173.170128, expected=168.340000
predicted=167.830634, expected=166.480000
predicted=166.317813, expected=167.780000
predicted=168.565124, expected=167.780000
predicted=167.291035, expected=166.680000
predicted=166.618044, expected=168.390000
predicted=168.633016, expected=171.610000
predicted=171.596896, expected=172.800000
predicted=172.575615, expected=168.380000
```

```
predicted=168.257332, expected=170.050000
predicted=170.512532, expected=173.250000
predicted=173.285095, expected=172.440000
predicted=171.863264, expected=174.140000
predicted=174.443163, expected=174.730000
predicted=174.807786, expected=175.820000
predicted=175.669052, expected=178.240000
predicted=178.376362, expected=177.840000
predicted=177.671460, expected=172.800000
predicted=172.696843, expected=165.720000
predicted=165.849475, expected=165.240000
predicted=165.639208, expected=162.940000
predicted=162.622756, expected=163.650000
predicted=163.560757, expected=164.220000
predicted=164.431583, expected=162.320000
predicted=162.074659, expected=165.260000
predicted=165.596173, expected=169.100000
predicted=169.127261, expected=176.570000
predicted=176.627510, expected=176.890000
predicted=176.693903, expected=183.830000
predicted=184.347149, expected=185.160000
predicted=185.366108, expected=186.050000
predicted=185.830058, expected=187.360000
predicted=187.876054, expected=190.040000
predicted=190.065010, expected=188.590000
predicted=188.440431, expected=188.150000
predicted=188.244308, expected=186.440000
predicted=186.500263, expected=188.180000
predicted=188.147654, expected=186.990000
predicted=186.880177, expected=186.310000
predicted=186.195659, expected=187.630000
predicted=187.911454, expected=187.160000
predicted=186.946631, expected=188.360000
predicted=188.434007, expected=188.150000
predicted=188.215764, expected=188.580000
predicted=188.537432, expected=188.580000
predicted=188.667736, expected=187.900000
predicted=187.808694, expected=187.500000
predicted=187.535881, expected=186.870000
Test RMSE: 2.146
```

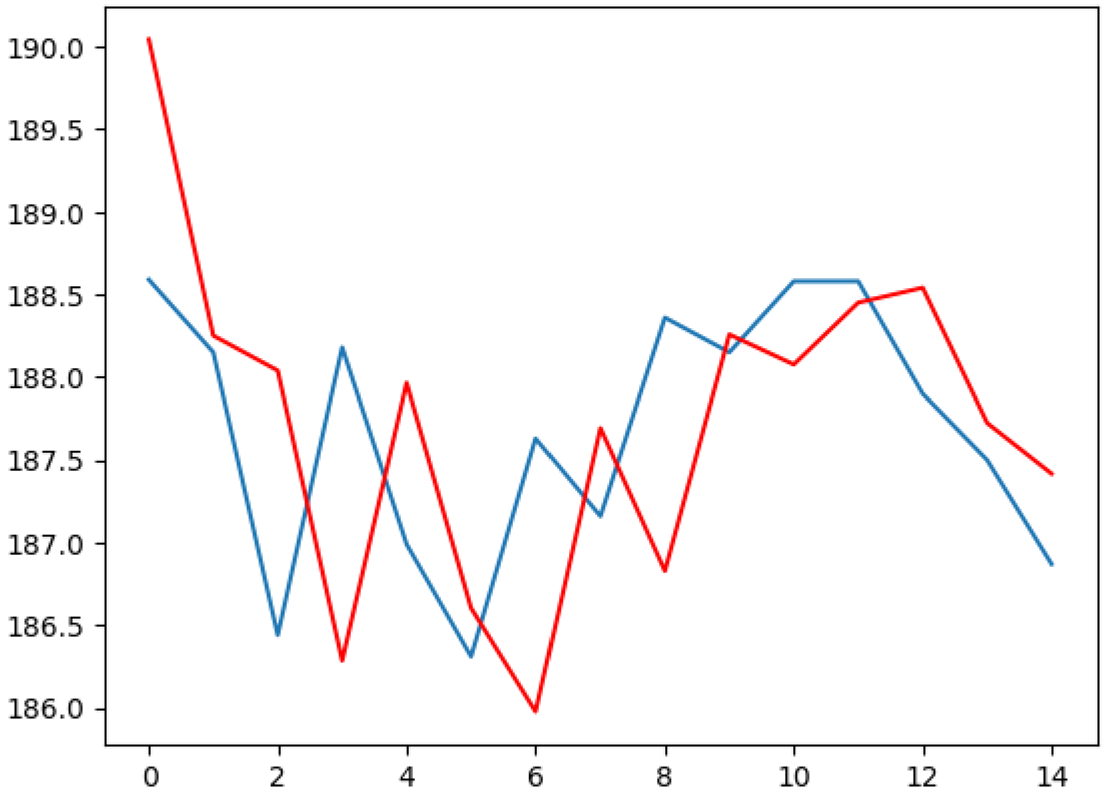


In [35]:

```
from math import sqrt
from sklearn.metrics import mean_squared_error
X = data
size = 876
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
        model = ARIMA(history, order=(5,1,1))
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))
```

```
# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)

pyplot.plot(predictions, color='red')
pyplot.show()
```

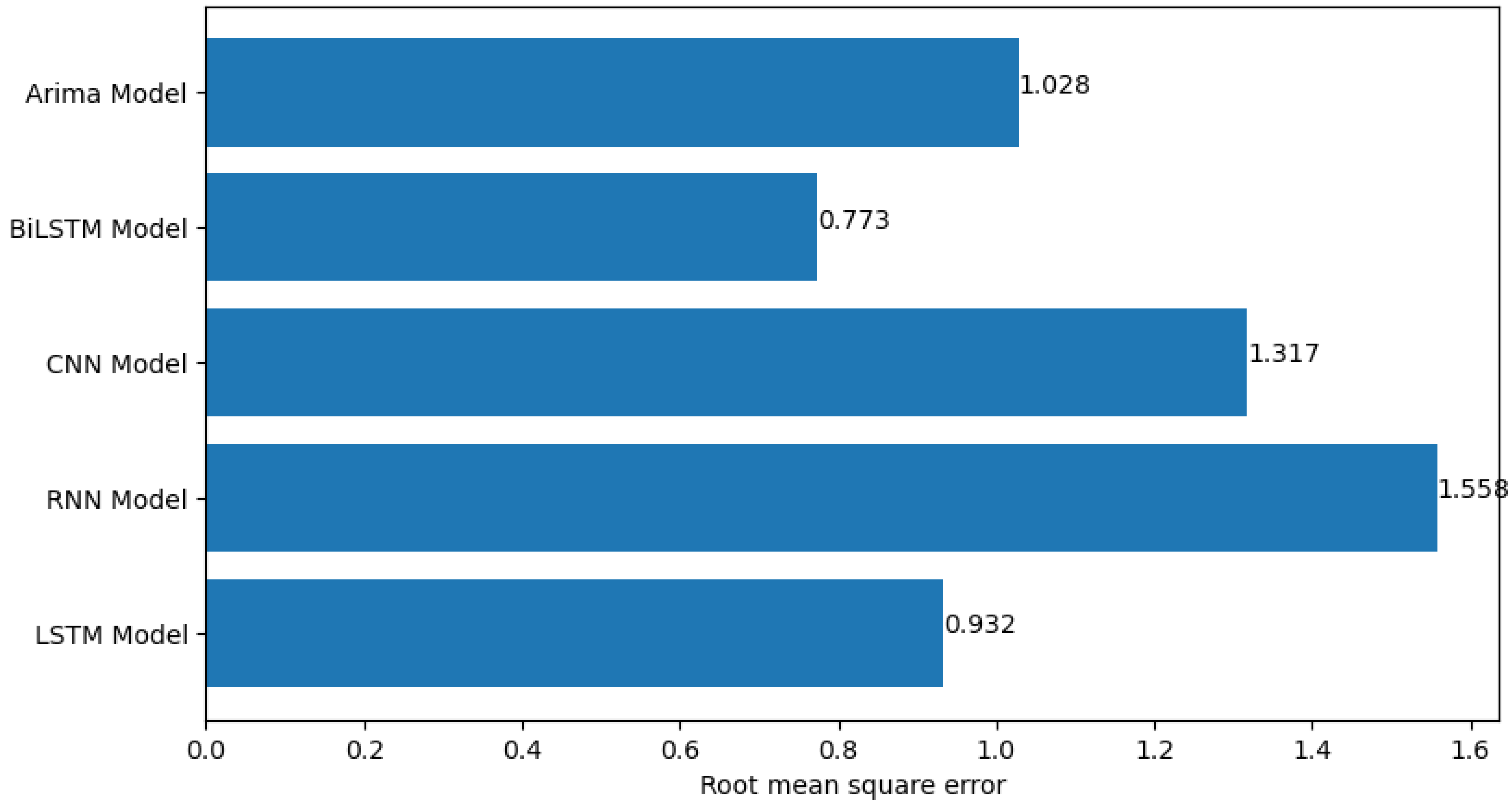
```
predicted=190.044783, expected=188.590000
predicted=188.250956, expected=188.150000
predicted=188.039589, expected=186.440000
predicted=186.285071, expected=188.180000
predicted=187.965689, expected=186.990000
predicted=186.603641, expected=186.310000
predicted=185.976972, expected=187.630000
predicted=187.693149, expected=187.160000
predicted=186.828252, expected=188.360000
predicted=188.257974, expected=188.150000
predicted=188.076107, expected=188.580000
predicted=188.450625, expected=188.580000
predicted=188.540922, expected=187.900000
predicted=187.723895, expected=187.500000
predicted=187.417436, expected=186.870000
Test RMSE: 1.024
```



Make results for comparision

In [36]:
```
plt.figure(figsize=(9, 5))

# models
Models_used = ["LSTM Model", "RNN Model", "CNN Model","BiLSTM Model","Arima Model"]

# metrics of the model in list format
rmse_of_models = [0.932,1.558, 1.317 ,0.773,1.028]

plt.barh(Models_used, rmse_of_models)
plt.xlabel("Root mean square error")
for index, value in enumerate(rmse_of_models):
    plt.text(value, index,
             str(value))
```

```
plt.show()
```



Further Implementation

Gated Recurrent Unit (GRU) Neural Network

- **In addition to the models discussed in the original research paper, we implemented the Gated Recurrent Unit (GRU) model as part of this project.**

- **The GRU model is known for its efficiency and accuracy in handling sequential data, making it a suitable candidate for time series forecasting.**

In [37]:
```
from tensorflow.keras.layers import GRU
```

In [38]:
```
def fit_gru(train, batchSize, epoch):
        X, y = train[:, 0:-1], train[:, -1]
        X = X.reshape(X.shape[0], 1, X.shape[1])
        model = Sequential()
        model.add(GRU(units=64, activation='tanh', batch_input_shape=(batchSize, X.shape[1], X.shape[2])))
        model.add(Dense(1))
        model.compile(loss='mean_squared_error', optimizer='adam')
        for i in range(epoch):
                model.fit(X, y, epochs=1, batch_size=batchSize, verbose=0, shuffle=False)
                model.reset_states()
        return model
```

In [39]:
```
def forecast_gru(model, batch_size, X):
        X = X.reshape(1, 1, len(X))
        yhat = model.predict(X, batch_size=batch_size)
        return yhat[0,0]
```

In [40]:
```
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-178], supervised_values[-178:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
gru_model = fit_gru(train_scaled, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
gru_model.predict(train_reshaped, batch_size=1)
```

```
# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_gru(gru_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        #print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-178:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-178:])
pyplot.plot(predictions)
pyplot.show()
```

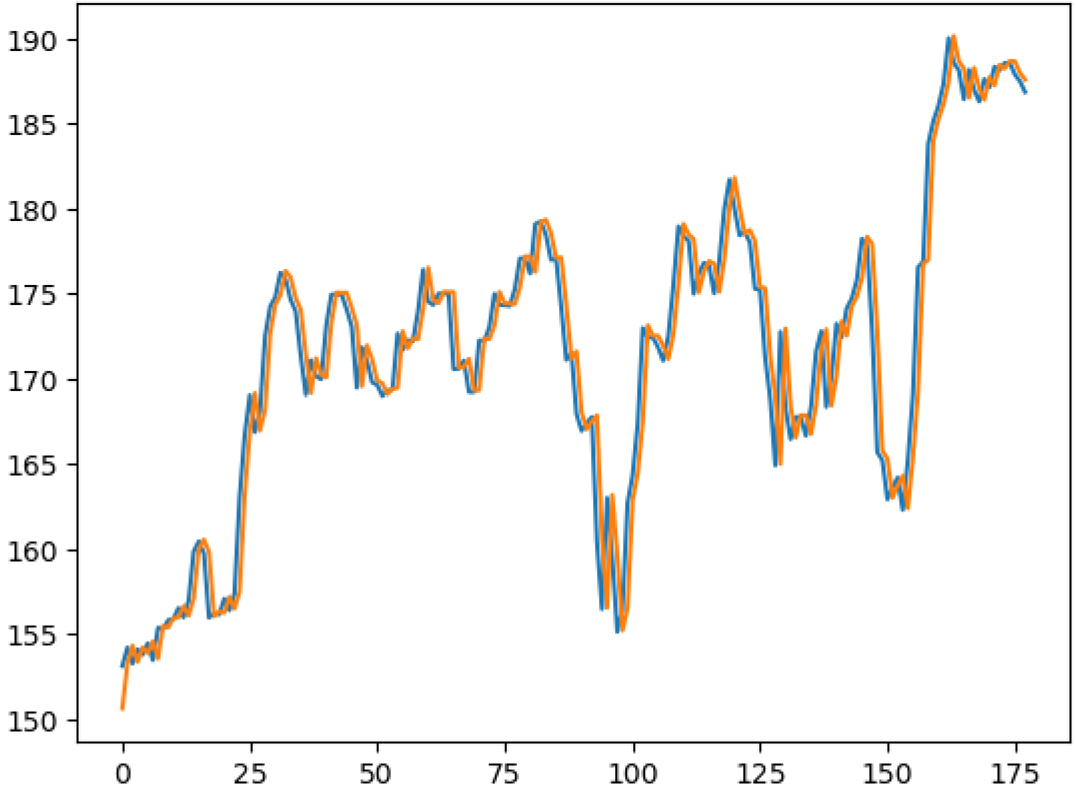
```
712/712 [==============================] - 1s 708us/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
```

```
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 14ms/step

 Test RMSE: 2.371
```



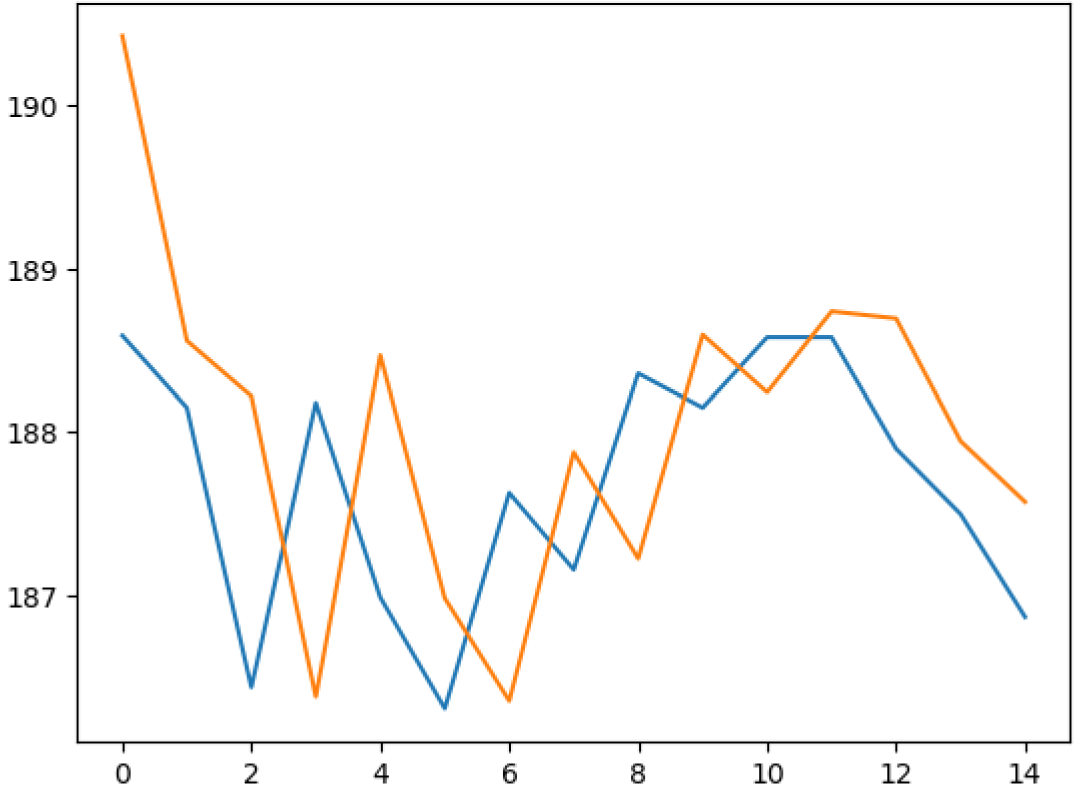
In [41]:

```
# modify data to be stationary
raw_values = np.array(data_stock)
diff_values = difference(raw_values, 1)
# modify data to be supervised
supervised = series_data_to_supervised_data(diff_values, 1)
supervised_values = supervised.values

# split data
train, test = supervised_values[0:-15], supervised_values[-15:]

# modify the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)
```

```
# fit the model
gru_model = fit_gru(train_scaled, 1, 1)

# forecast
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
gru_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
        # make one-step forecast
        X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
        yhat = forecast_gru(gru_model, 1, X)
        # invert scaling
        yhat = invert_scale(scaler, X, yhat)

        yhat = inverse_the_difference(raw_values, yhat, len(test_scaled)-i+1)
        # store forecast
        predictions.append(yhat)
        expected = raw_values[len(train) + i + 1]
        #print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))


##print("X and y data : ", X ," y = ",y )
#print("yhat = ", yhat)
# report performance
rmse = sqrt(mean_squared_error(raw_values[-15:], predictions))
print('\n Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-15:])
pyplot.plot(predictions)
pyplot.show()
```

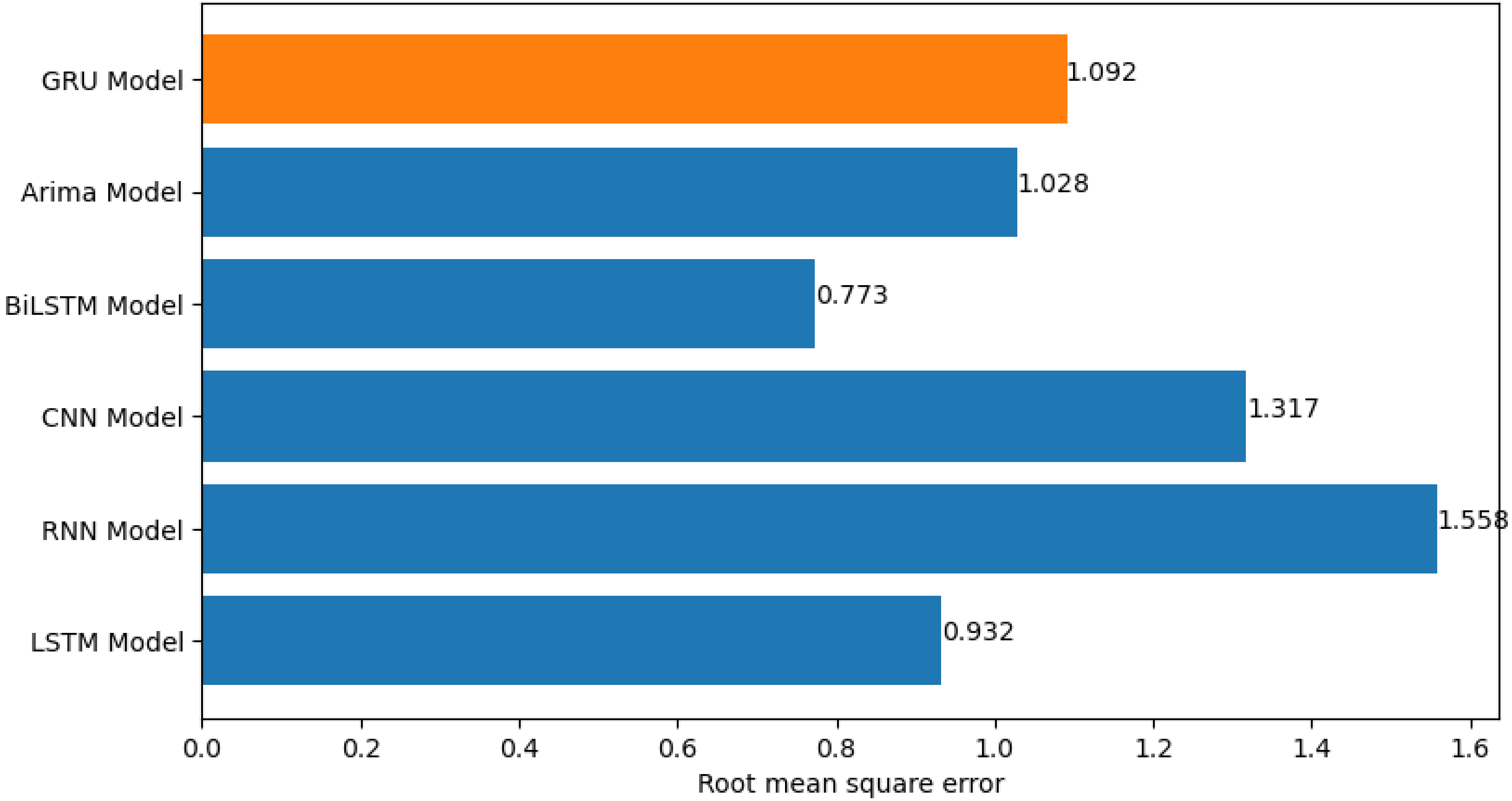
```
875/875 [==============================] - 1s 707us/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
```

```
Test RMSE: 1.085
```



Model Comparison

In [42]:
```
from matplotlib import pyplot
pyplot.figure(figsize=(9, 5))

# models
Models_used = ["LSTM Model", "RNN Model", "CNN Model","BiLSTM Model","Arima Model"]

# metrics of the model in list format
rmse_of_models = [0.932,1.558, 1.317 ,0.773,1.028]
pyplot.barh(Models_used, rmse_of_models)
pyplot.barh("GRU Model", 1.092)
pyplot.xlabel("Root mean square error")
for index, value in enumerate(rmse_of_models):
    pyplot.text(value, index,
             str(value))
pyplot.text(1.088, 5, str(1.092))

pyplot.show()
```



References:

[1]: R. Khandelwal, P. Marfatia, S. Shah, V. Joshi, P. Kamath and K. Chavan, "Financial Data Time Series Forecasting Using Neural Networks and a Comparative Study," 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, 2022, pp. 1-6, doi: 10.1109/ICONAT53423.2022.9725845.

[2]: R. Khandelwal (2022, January 28). Financial-Data-Time-Series-Forecasting-Using-Neural-Networks. https://github.com/ritvik02/Financial-Data-Time-Series-Forecasting-Using-Neural-Networks

Research Paper Link: **https://github.com/ritvik02/Financial-Data-Time-Series-Forecasting-Using-Neural-Networks/blob/main/Paper.pdf**

Source Link: https://ieeexplore.ieee.org/document/9725845

In [ ]:

```

```