In [32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
```

In [33]:
```
df = pd.read_csv("./dataset/data_range.csv")
df.drop(0,inplace=True)
df.head(10)
```

Out[33]:

	date	open	high	low	clos
1	2012-05-18	42.04999923706055	45.0	38.0	38.2299995422363
2	2012-05-21	36.529998779296875	36.65999984741211	33.0	34.0299987792968
3	2012-05-22	32.61000061035156	33.59000015258789	30.940000534057617	31
4	2012-05-23	31.3700008392334	32.5	31.360000610351562	32
5	2012-05-24	32.95000076293945	33.209999084472656	31.770000457763672	33.0299987792968
6	2012-05-25	32.900001525878906	32.95000076293945	31.110000610351562	31.909999847412
7	2012-05-29	31.479999542236328	31.690000534057617	28.649999618530273	28.840000152587
8	2012-05-30	28.700000762939453	29.549999237060547	27.860000610351562	28.1900005340576
9	2012-05-31	28.549999237060547	29.670000076293945	26.829999923706055	29.6000003814697
10	2012-06-01	28.889999389648438	29.149999618530273	27.389999389648438	27.7199993133544

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3331 entries, 1 to 3331
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   date       3331 non-null   object
 1   open       3331 non-null   object
 2   high       3331 non-null   object
 3   low        3331 non-null   object
 4   close      3331 non-null   object
 5   adj_close  3331 non-null   object
 6   volume     3331 non-null   object
dtypes: object(7)
memory usage: 182.3+ KB
```

Convert Data Types

In [35]:
```
df['date'] = pd.to_datetime(df['date'])
df['open'] = pd.to_numeric(df['open'])
df['high'] = pd.to_numeric(df['high'])
df['low'] = pd.to_numeric(df['low'])
df['close'] = pd.to_numeric(df['close'])
df['adj_close'] = pd.to_numeric(df['adj_close'])
df['volume'] = df['volume'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3331 entries, 1 to 3331
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   date       3331 non-null   datetime64[ns]
 1   open       3331 non-null   float64       
 2   high       3331 non-null   float64       
 3   low        3331 non-null   float64       
 4   close      3331 non-null   float64       
 5   adj_close  3331 non-null   float64       
 6   volume     3331 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 182.3 KB
```

In [36]:
```
df.head()
```

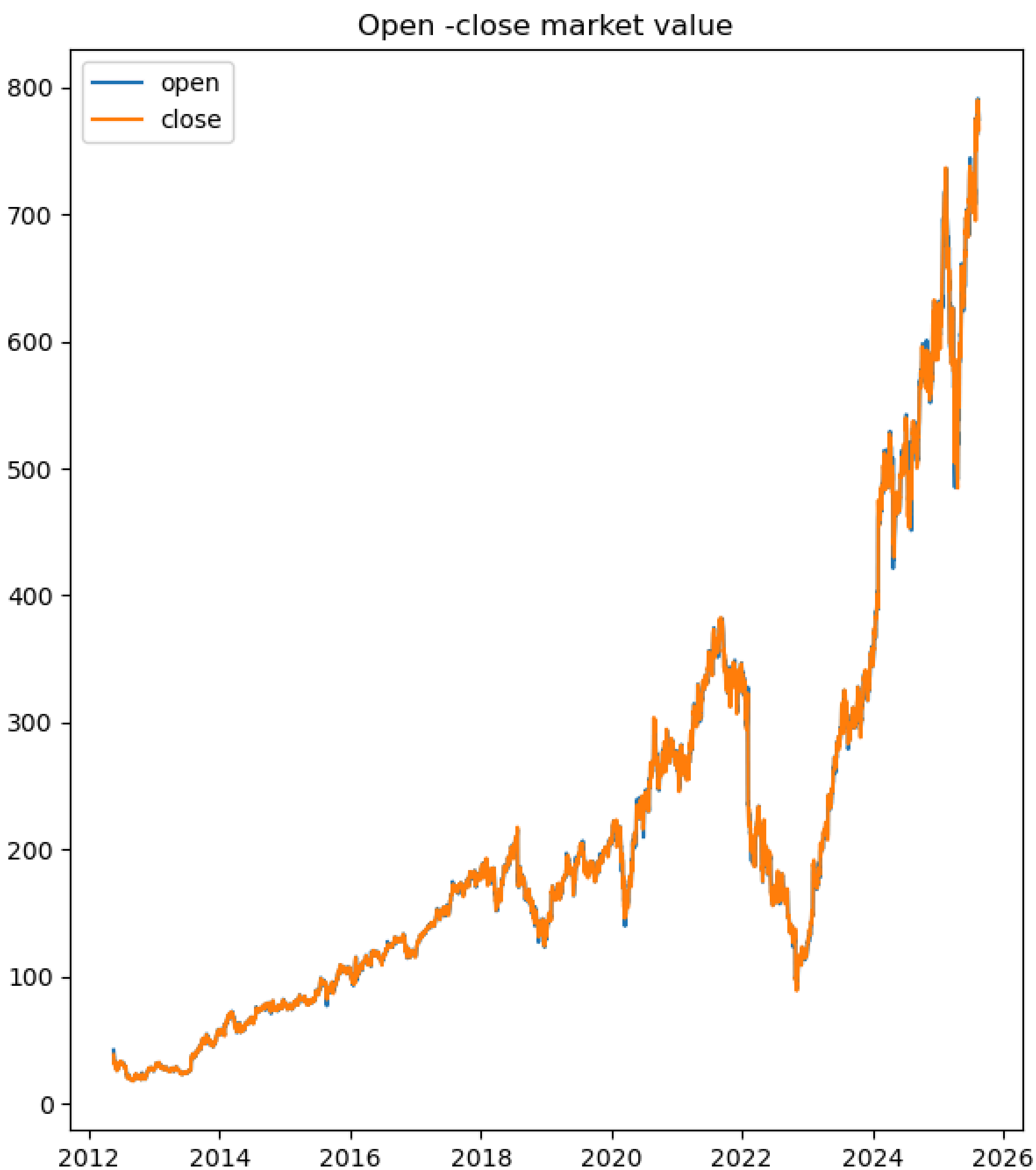
Out[36]:

	date	open	high	low	close	adj_close	volume
1	2012-05-18	42.049999	45.000000	38.000000	38.230000	38.021412	573576400
2	2012-05-21	36.529999	36.660000	33.000000	34.029999	33.844330	168192700
3	2012-05-22	32.610001	33.590000	30.940001	31.000000	30.830860	101786600
4	2012-05-23	31.370001	32.500000	31.360001	32.000000	31.825403	73600000
5	2012-05-24	32.950001	33.209999	31.770000	33.029999	32.849785	50237200

Plot graphs

In [37]:
```
plt.figure(figsize=(7,8))
plt.plot(df['date'],df['open'], label='open')
plt.plot(df['date'],df['close'], label='close')
plt.title("Open -close market value")
plt.legend()
```

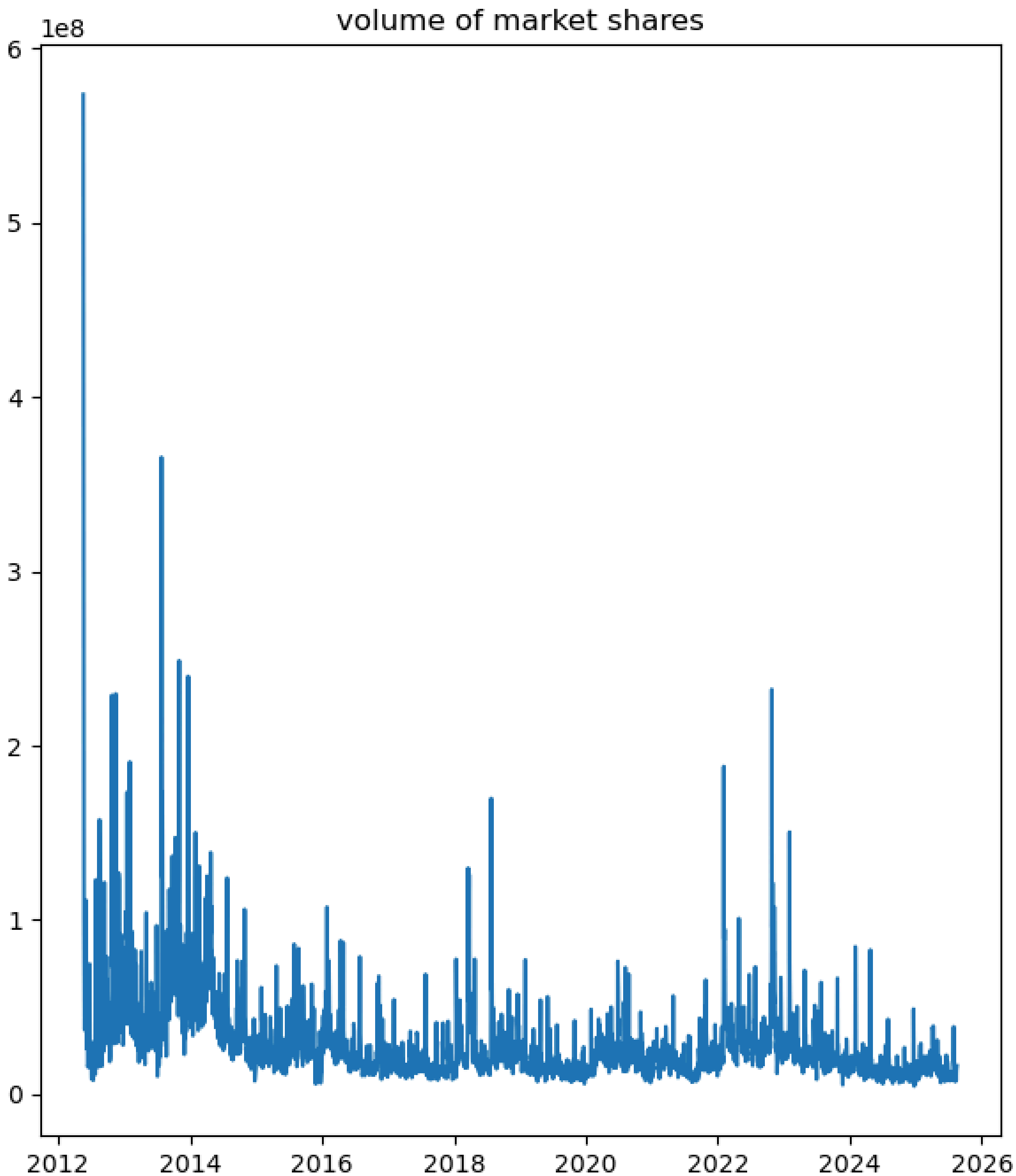
Out[37]: <matplotlib.legend.Legend at 0x170ada96030>



In [38]:
```
plt.figure(figsize=(7,8))
plt.plot(df['date'],df['volume'])
```

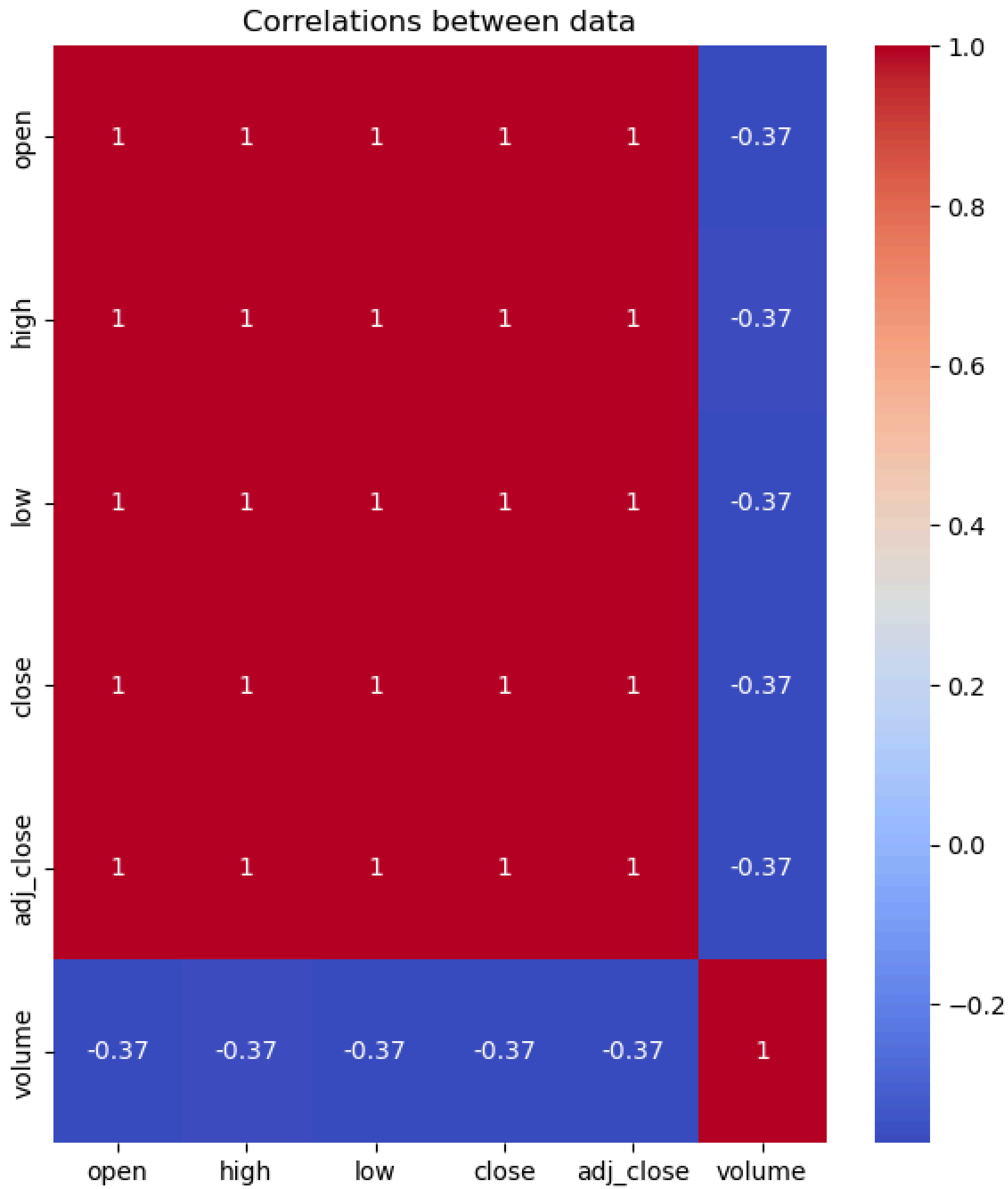
```
plt.title("volume of market shares")
```

Out[38]: Text(0.5, 1.0, 'volume of market shares')



In [39]:
```
numeric = df.drop(columns='date')
```

In [40]:
```
plt.figure(figsize=(7,8))
sns.heatmap(numeric.corr(),annot=True,cmap='coolwarm')
plt.title("Correlations between data")
plt.show()
```



Building the LSTM model

In [41]:
```
stock_close = df[['close']]
```

In [42]:
```
dataset = stock_close.values
```

In [43]:
```
training_len = int(np.ceil(len(dataset)*0.95))
```

In [44]:
```
# preprocessing stages
scaler = StandardScaler()
scaled_training_data = scaler.fit_transform(stock_close.iloc[:training_len,:])
```

In [45]:
```
x_train,y_train=[],[]
```

In [46]:
```
# Create sliding window for stocks (60 days)
for i in range(60,training_len):
    x_train.append(scaled_training_data[i-60:i,0])
    y_train.append(scaled_training_data[i,0])
```

In [83]:
```
for x in range (len(x_train)):
    if len(x_train[x]) != 60:
        print(len(x_train[x]))
```

In [47]:
```
x_train,y_train = np.array(x_train), np.array(y_train)
```

In [48]:
```
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
```

In [49]:
```
#Build the model
model = keras.Sequential()
```

In [50]:
```
model.add(keras.layers.LSTM(64, return_sequences=True,input_shape=(x_train.shape[1]
model.add(keras.layers.LSTM(64, return_sequences=False))
model.add(keras.layers.Dense(128,activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(1))
```

```
C:\Anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:199: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

In [51]:
```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 60, 64)	16,896
lstm_3 (LSTM)	(None, 64)	33,024
dense_2 (Dense)	(None, 128)	8,320
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 58,369 (228.00 KB)

Trainable params: 58,369 (228.00 KB)

Non-trainable params: 0 (0.00 B)

In [52]:
```
model.compile(optimizer='adam',
              loss='mae',
              metrics=[keras.metrics.RootMeanSquaredError()])
```

In [53]:
```
training = model.fit(x_train,y_train,epochs=20,batch_size=32)
```

```
Epoch 1/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 6s 27ms/step - loss: 0.1860 - root_mean_squared_error: 0.
3262
Epoch 2/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 25ms/step - loss: 0.1179 - root_mean_squared_error: 0.
1838
Epoch 3/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - loss: 0.1136 - root_mean_squared_error: 0.
1746
Epoch 4/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - loss: 0.1159 - root_mean_squared_error: 0.
1884
Epoch 5/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - loss: 0.1002 - root_mean_squared_error: 0.
1573
Epoch 6/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - loss: 0.1006 - root_mean_squared_error: 0.
1540
Epoch 7/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - loss: 0.1004 - root_mean_squared_error: 0.
1501
Epoch 8/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 0.1020 - root_mean_squared_error: 0.
1574
Epoch 9/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - loss: 0.0982 - root_mean_squared_error: 0.
1543
Epoch 10/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - loss: 0.1008 - root_mean_squared_error: 0.
1522
Epoch 11/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - loss: 0.0983 - root_mean_squared_error: 0.
1520
Epoch 12/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 28ms/step - loss: 0.0912 - root_mean_squared_error: 0.
1475
Epoch 13/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - loss: 0.0959 - root_mean_squared_error: 0.
1517
Epoch 14/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 0.0906 - root_mean_squared_error: 0.
1422
Epoch 15/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 31ms/step - loss: 0.0973 - root_mean_squared_error: 0.
1556
Epoch 16/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - loss: 0.0895 - root_mean_squared_error: 0.
1381
Epoch 17/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 0.0862 - root_mean_squared_error: 0.
1322
Epoch 18/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 0.0901 - root_mean_squared_error: 0.
1428
Epoch 19/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 35ms/step - loss: 0.0871 - root_mean_squared_error: 0.
```

```
1355
Epoch 20/20
98/98 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - loss: 0.0896 - root_mean_squared_error: 0.
1418
```

In [54]:
```
# Prepare test data
scaled_test_data = scaler.fit_transform(stock_close.iloc[training_len-60:,:])
```

In [59]:
```
x_test,y_test = [],  dataset[training_len:]
```

In [62]:
```
# Create sliding window for stocks (60 days)
for i in range(60,len(scaled_test_data)):
    x_test.append(scaled_test_data[i-60:i,0])
```

In [91]:
```
sum = 0
for i in range(len(y_test)):
    if len(x_test[i]) !=30:
        print(len(x_test[i]))
```

In [85]:
```
len(y_test)
```

Out[85]:
```
166
```

In [80]:
```
len(scaled_test_data)
```

Out[80]:
```
226
```

In [74]:
```
len(x_test[0])
```

Out[74]:
```
30
```

In [96]:
```
x_test = np.array(x_test[:166])
```

In [97]:
```
x_test = np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

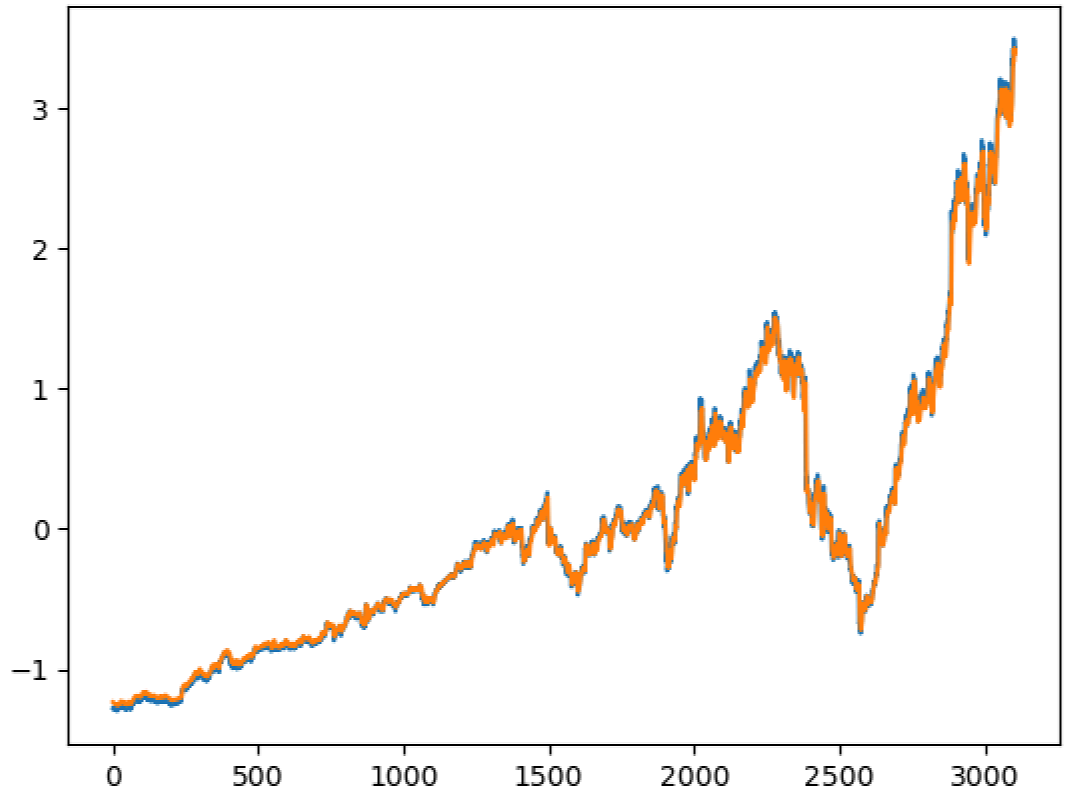
Predictions

In [98]:
```
predictions = model.predict(x_train)
```

```
98/98 ━━━━━━━━━━━━━━━━━━━━ 4s 32ms/step
```

In [100…
```
plt.plot(y_train)
plt.plot(predictions)
```

Out[100…
```
[<matplotlib.lines.Line2D at 0x170b4a1b920>]
```



In [ ]: