Mini Project

New York Stock Price Prediction

Description:

This notebook demonstrates the future price prediction for different stocks using recurrent neural networks in tensorflow. Recurrent neural networks with basic RNN, LSTM or GRU cells are implemented.

Done By:

Anni Priscilla A (195002010)

Badri MSV (195002017)

Gokulakrishnan S (195002039)

Harshithaa Murali (195002049)

Paarkeshvaran R (195002077)

1. Libraries and settings

In [1]:

```
import numpy as np
import pandas as pd
import math
import sklearn
import sklearn.preprocessing
import datetime
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

# split data in 80%/10%/10% train/validation/test sets
valid_set_size_percentage = 10
test_set_size_percentage = 10
```

2. Analyze the Data

- load stock prices from prices-split-adjusted.csv
- analyze data

In [2]:

```
# import all stock prices
df = pd.read_csv("prices-split-adjusted.csv", index_col = 0)
df.info()
df.head()

# number of different stocks
```

```
print('\nnumber of different stocks: ', len(list(set(df.symbol))))
print(list(set(df.symbol))[:10])
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 851264 entries, 2016-01-05 to 2016-12-30
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   symbol  851264 non-null  object 
 1   open    851264 non-null  float64
 2   close   851264 non-null  float64
 3   low     851264 non-null  float64
 4   high    851264 non-null  float64
 5   volume  851264 non-null  float64
dtypes: float64(5), object(1)
memory usage: 45.5+ MB

number of different stocks:  501
['DLPH', 'JPM', 'TSS', 'GM', 'LUK', 'RHT', 'AIZ', 'DE', 'CME', 'DLR']
```

In [3]:

```
df.tail()
```

Out[3]:

date	symbol	open	close	low	high	volume
2016-12-30	ZBH	103.309998	103.199997	102.849998	103.930000	973800.0
2016-12-30	ZION	43.070000	43.040001	42.689999	43.310001	1938100.0
2016-12-30	ZTS	53.639999	53.529999	53.270000	53.740002	1701200.0
2016-12-30	AIV	44.730000	45.450001	44.410000	45.590000	1380900.0
2016-12-30	FTV	54.200001	53.630001	53.389999	54.480000	705100.0

In [4]:

```
df.describe()
```

Out[4]:

	open	close	low	high	volume
count	851264.000000	851264.000000	851264.000000	851264.000000	8.512640e+05
mean	64.993618	65.011913	64.336541	65.639748	5.415113e+06
std	75.203893	75.201216	74.459518	75.906861	1.249468e+07
min	1.660000	1.590000	1.500000	1.810000	0.000000e+00
25%	31.270000	31.292776	30.940001	31.620001	1.221500e+06
50%	48.459999	48.480000	47.970001	48.959999	2.476250e+06
75%	75.120003	75.139999	74.400002	75.849998	5.222500e+06
max	1584.439941	1578.130005	1549.939941	1600.930054	8.596434e+08

In [5]:

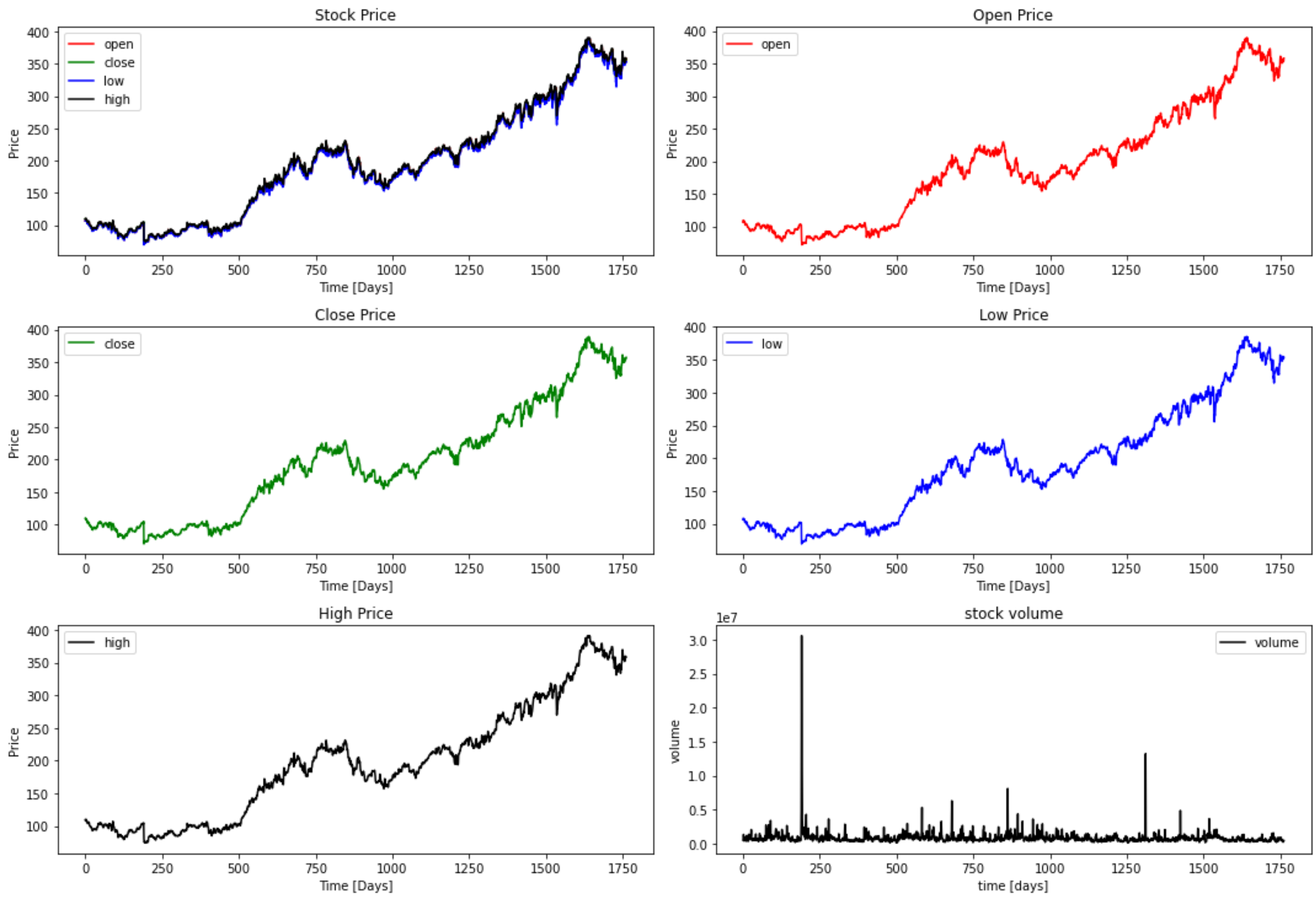
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 851264 entries, 2016-01-05 to 2016-12-30
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   symbol  851264 non-null  object 
```

```
 1   open    851264 non-null  float64
 2   close   851264 non-null  float64
 3   low     851264 non-null  float64
 4   high    851264 non-null  float64
 5   volume  851264 non-null  float64
dtypes: float64(5), object(1)
memory usage: 45.5+ MB
```

In [6]:

```python
plt.figure(figsize=(15, 20));
plt.subplot(6,2,1);
plt.plot(df[df.symbol == 'EQIX'].open.values, color='red', label='open')
plt.plot(df[df.symbol == 'EQIX'].close.values, color='green', label='close')
plt.plot(df[df.symbol == 'EQIX'].low.values, color='blue', label='low')
plt.plot(df[df.symbol == 'EQIX'].high.values, color='black', label='high')
plt.title('Stock Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')
#plt.show()

plt.subplot(6,2,2);
plt.plot(df[df.symbol == 'EQIX'].open.values, color='red', label='open')
plt.title('Open Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,3);
plt.plot(df[df.symbol == 'EQIX'].close.values, color='green', label='close')
plt.title('Close Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,4);
plt.plot(df[df.symbol == 'EQIX'].low.values, color='blue', label='low')
plt.title('Low Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,5);
plt.plot(df[df.symbol == 'EQIX'].high.values, color='black', label='high')
plt.title('High Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,6);
plt.plot(df[df.symbol == 'EQIX'].volume.values, color='black', label='volume')
plt.title('stock volume')
plt.xlabel('time [days]')
plt.ylabel('volume')
plt.legend(loc='best');

plt.tight_layout()
```



3. Manipulate data

- choose a specific stock
- drop feature: volume
- normalize stock data
- create train, validation and test data sets

In [7]:

```
# function for min-max normalization of stock
def normalize_data(df):
    min_max_scaler = sklearn.preprocessing.MinMaxScaler()
    df['open'] = min_max_scaler.fit_transform(df.open.values.reshape(-1,1))
    df['high'] = min_max_scaler.fit_transform(df.high.values.reshape(-1,1))
    df['low'] = min_max_scaler.fit_transform(df.low.values.reshape(-1,1))
    df['close'] = min_max_scaler.fit_transform(df['close'].values.reshape(-1,1))
    return df

# function to create train, validation, test data given stock data and sequence length
def load_data(stock, seq_len):
    data_raw = np.array(stock, dtype="float32") # convert to numpy array
    data = []

    # create all possible sequences of length seq_len
    for index in range(len(data_raw) - seq_len):
        data.append(data_raw[index: index + seq_len])

    data = np.array(data);
    valid_set_size = int(np.round(valid_set_size_percentage/100*data.shape[0]));
    test_set_size = int(np.round(test_set_size_percentage/100*data.shape[0]));
    train_set_size = data.shape[0] - (valid_set_size + test_set_size);
```

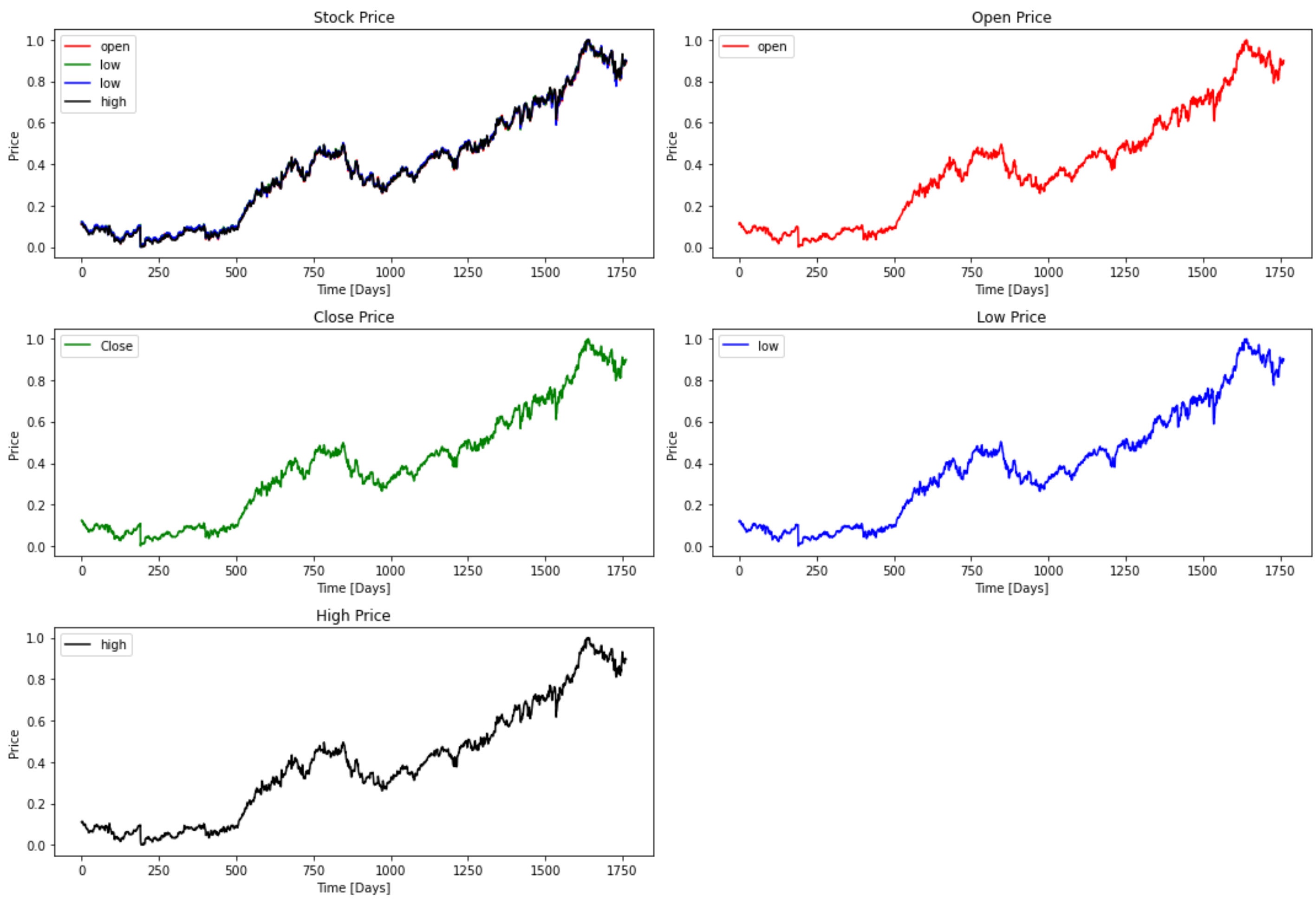
```
    x_train = data[:train_set_size,:-1,:]
    y_train = data[:train_set_size,-1,:]

    x_valid = data[train_set_size:train_set_size+valid_set_size,:-1,:]
    y_valid = data[train_set_size:train_set_size+valid_set_size,-1,:]

    x_test = data[train_set_size+valid_set_size:,:-1,:]
    y_test = data[train_set_size+valid_set_size:,-1,:]

    return [x_train, y_train, x_valid, y_valid, x_test, y_test]

# choose one stock
df_stock = df[df.symbol == 'EQIX'].copy()
df_stock.drop(labels=['symbol'],axis=1,inplace=True)
df_stock.drop(labels=['volume'],axis=1,inplace=True)

cols = list(df_stock.columns.values)
print('df_stock.columns.values = ', cols)

# normalize stock
df_stock_norm = df_stock.copy()
df_stock_norm = normalize_data(df_stock_norm)

# create train, test data
seq_len = 20 # choose sequence length
x_train, y_train, x_valid, y_valid, x_test, y_test = load_data(df_stock_norm, seq_len)
print('x_train.shape = ',x_train.shape)
print('y_train.shape = ', y_train.shape)
print('x_valid.shape = ',x_valid.shape)
print('y_valid.shape = ', y_valid.shape)
print('x_test.shape = ', x_test.shape)
print('y_test.shape = ',y_test.shape)
```

```
df_stock.columns.values =  ['open', 'close', 'low', 'high']
x_train.shape =  (1394, 19, 4)
y_train.shape =  (1394, 4)
x_valid.shape =  (174, 19, 4)
y_valid.shape =  (174, 4)
x_test.shape =  (174, 19, 4)
y_test.shape =  (174, 4)
```

In [8]:

```
plt.figure(figsize=(15, 20));
plt.subplot(6,2,1);
plt.plot(df_stock_norm.open.values, color='red', label='open')
plt.plot(df_stock_norm.close.values, color='green', label='low')
plt.plot(df_stock_norm.low.values, color='blue', label='low')
plt.plot(df_stock_norm.high.values, color='black', label='high')
plt.title('Stock Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')
#plt.show()

plt.subplot(6,2,2);
plt.plot(df_stock_norm.open.values, color='red', label='open')
plt.title('Open Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,3);
plt.plot(df_stock_norm.close.values, color='green', label='Close')
plt.title('Close Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
```

```
plt.legend(loc='best')

plt.subplot(6,2,4);
plt.plot(df_stock_norm.low.values, color='blue', label='low')
plt.title('Low Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.subplot(6,2,5);
plt.plot(df_stock_norm.high.values, color='black', label='high')
plt.title('High Price')
plt.xlabel('Time [Days]')
plt.ylabel('Price')
plt.legend(loc='best')

plt.tight_layout()
```



4. Model and validate data

- RNNs with basic, LSTM, GRU cells

In [9]:

```
## Basic Cell RNN in tensorflow
index_in_epoch = 0;
perm_array  = np.arange(x_train.shape[0])
np.random.shuffle(perm_array)

# function to get the next batch
def get_next_batch(batch_size):
    global index_in_epoch, x_train, perm_array
    start = index_in_epoch
```

```
    index_in_epoch += batch_size

    if index_in_epoch > x_train.shape[0]:
        np.random.shuffle(perm_array) # shuffle permutation array
        start = 0 # start next epoch
        index_in_epoch = batch_size

    end = index_in_epoch
    return x_train[perm_array[start:end]], y_train[perm_array[start:end]]

# parameters
n_steps = seq_len-1
n_inputs = 4
n_neurons = 200
n_outputs = 4
n_layers = 2
batch_size = 100
n_epochs = 50
```

In [10]:

```
RNNcells = [tf.keras.layers.SimpleRNNCell(n_neurons) for _ in range(n_layers)]
rnn = tf.keras.layers.StackedRNNCells(RNNcells, input_shape = (None, n_inputs))
RNNmodel = Sequential()
RNNmodel.add(layers.RNN(rnn))
RNNmodel.add(layers.Dense(n_outputs))
RNNmodel.compile(loss=tf.keras.losses.mean_squared_error, optimizer=tf.keras.optimizers.Ac
RNNmodel.fit(x_train, y_train,validation_data=(x_valid, y_valid), batch_size=batch_size, e
```

```
Epoch 1/50
14/14 [==============================] - 1s 24ms/step - loss: 0.2534 - val_loss: 0.1164
Epoch 2/50
14/14 [==============================] - 0s 15ms/step - loss: 0.0124 - val_loss: 0.0243
Epoch 3/50
14/14 [==============================] - 0s 12ms/step - loss: 0.0031 - val_loss: 0.0304
Epoch 4/50
14/14 [==============================] - 0s 13ms/step - loss: 0.0013 - val_loss: 0.0108
Epoch 5/50
14/14 [==============================] - 0s 14ms/step - loss: 6.6568e-04 - val_loss: 0.006
2
Epoch 6/50
14/14 [==============================] - 0s 15ms/step - loss: 5.3134e-04 - val_loss: 0.004
9
Epoch 7/50
14/14 [==============================] - 0s 12ms/step - loss: 4.9550e-04 - val_loss: 0.004
2
Epoch 8/50
14/14 [==============================] - 0s 14ms/step - loss: 4.5524e-04 - val_loss: 0.004
1
Epoch 9/50
14/14 [==============================] - 0s 14ms/step - loss: 4.2308e-04 - val_loss: 0.003
5
Epoch 10/50
14/14 [==============================] - 0s 13ms/step - loss: 4.1254e-04 - val_loss: 0.004
2
Epoch 11/50
14/14 [==============================] - 0s 14ms/step - loss: 4.0121e-04 - val_loss: 0.004
5
Epoch 12/50
14/14 [==============================] - 0s 16ms/step - loss: 3.8661e-04 - val_loss: 0.002
8
Epoch 13/50
14/14 [==============================] - 0s 14ms/step - loss: 3.9752e-04 - val_loss: 0.003
5
Epoch 14/50
14/14 [==============================] - 0s 13ms/step - loss: 3.6560e-04 - val_loss: 0.003
```

```
3
Epoch 15/50
14/14 [==============================] - 0s 14ms/step - loss: 3.3925e-04 - val_loss: 0.002
1
Epoch 16/50
14/14 [==============================] - 0s 14ms/step - loss: 3.5416e-04 - val_loss: 0.003
2
Epoch 17/50
14/14 [==============================] - 0s 13ms/step - loss: 3.4145e-04 - val_loss: 0.002
4
Epoch 18/50
14/14 [==============================] - 0s 12ms/step - loss: 3.1041e-04 - val_loss: 0.002
9
Epoch 19/50
14/14 [==============================] - 0s 14ms/step - loss: 2.9453e-04 - val_loss: 0.002
9
Epoch 20/50
14/14 [==============================] - 0s 14ms/step - loss: 3.0721e-04 - val_loss: 0.002
0
Epoch 21/50
14/14 [==============================] - 0s 12ms/step - loss: 2.9505e-04 - val_loss: 0.001
5
Epoch 22/50
14/14 [==============================] - 0s 12ms/step - loss: 2.7489e-04 - val_loss: 0.001
9
Epoch 23/50
14/14 [==============================] - 0s 12ms/step - loss: 2.7428e-04 - val_loss: 0.001
7
Epoch 24/50
14/14 [==============================] - 0s 13ms/step - loss: 2.4986e-04 - val_loss: 0.001
6
Epoch 25/50
14/14 [==============================] - 0s 12ms/step - loss: 2.4607e-04 - val_loss: 0.001
8
Epoch 26/50
14/14 [==============================] - 0s 12ms/step - loss: 2.5072e-04 - val_loss: 0.001
3
Epoch 27/50
14/14 [==============================] - 0s 13ms/step - loss: 2.8352e-04 - val_loss: 0.001
1
Epoch 28/50
14/14 [==============================] - 0s 14ms/step - loss: 2.9835e-04 - val_loss: 0.001
3
Epoch 29/50
14/14 [==============================] - 0s 12ms/step - loss: 2.9285e-04 - val_loss: 0.001
1
Epoch 30/50
14/14 [==============================] - 0s 12ms/step - loss: 2.3959e-04 - val_loss: 0.001
4
Epoch 31/50
14/14 [==============================] - 0s 12ms/step - loss: 2.3423e-04 - val_loss: 0.001
4
Epoch 32/50
14/14 [==============================] - 0s 13ms/step - loss: 2.2818e-04 - val_loss: 0.001
3
Epoch 33/50
14/14 [==============================] - 0s 12ms/step - loss: 2.9997e-04 - val_loss: 0.001
2
Epoch 34/50
14/14 [==============================] - 0s 14ms/step - loss: 4.0386e-04 - val_loss: 0.001
5
Epoch 35/50
14/14 [==============================] - 0s 14ms/step - loss: 4.4291e-04 - val_loss: 0.001
5
Epoch 36/50
14/14 [==============================] - 0s 15ms/step - loss: 3.4102e-04 - val_loss: 0.001
```

```
4
Epoch 37/50
14/14 [==============================] - 0s 14ms/step - loss: 2.8595e-04 - val_loss: 0.001
4
Epoch 38/50
14/14 [==============================] - 0s 14ms/step - loss: 2.4480e-04 - val_loss: 0.001
2
Epoch 39/50
14/14 [==============================] - 0s 13ms/step - loss: 3.1918e-04 - val_loss: 0.001
4
Epoch 40/50
14/14 [==============================] - 0s 12ms/step - loss: 3.1493e-04 - val_loss: 8.347
7e-04
Epoch 41/50
14/14 [==============================] - 0s 12ms/step - loss: 2.7492e-04 - val_loss: 0.001
1
Epoch 42/50
14/14 [==============================] - 0s 14ms/step - loss: 2.0584e-04 - val_loss: 9.135
8e-04
Epoch 43/50
14/14 [==============================] - 0s 14ms/step - loss: 2.0360e-04 - val_loss: 7.938
2e-04
Epoch 44/50
14/14 [==============================] - 0s 12ms/step - loss: 2.3141e-04 - val_loss: 0.001
0
Epoch 45/50
14/14 [==============================] - 0s 12ms/step - loss: 3.1216e-04 - val_loss: 0.001
2
Epoch 46/50
14/14 [==============================] - 0s 12ms/step - loss: 2.7916e-04 - val_loss: 0.001
0
Epoch 47/50
14/14 [==============================] - 0s 12ms/step - loss: 2.2374e-04 - val_loss: 6.172
8e-04
Epoch 48/50
14/14 [==============================] - 0s 13ms/step - loss: 2.1808e-04 - val_loss: 8.801
4e-04
Epoch 49/50
14/14 [==============================] - 0s 13ms/step - loss: 2.1725e-04 - val_loss: 0.001
3
Epoch 50/50
14/14 [==============================] - 0s 14ms/step - loss: 1.9219e-04 - val_loss: 7.270
3e-04
```

Out[10]:

```
<keras.callbacks.History at 0x24f44f9a5b0>
```

In [11]:

```python
LSTMcells = [tf.keras.layers.LSTMCell(n_neurons) for _ in range(n_layers)]
lstm = tf.keras.layers.StackedRNNCells(LSTMcells, input_shape = (None, n_inputs))
LSTMmodel = Sequential()
LSTMmodel.add(layers.RNN(lstm))
LSTMmodel.add(layers.Dense(n_outputs))
LSTMmodel.compile(loss=tf.keras.losses.mean_squared_error, optimizer=tf.keras.optimizers.A
LSTMmodel.fit(x_train, y_train,validation_data=(x_valid, y_valid), batch_size=batch_size,
```

```
Epoch 1/50
14/14 [==============================] - 2s 57ms/step - loss: 0.0186 - val_loss: 0.0036
Epoch 2/50
14/14 [==============================] - 1s 39ms/step - loss: 0.0020 - val_loss: 0.0047
Epoch 3/50
14/14 [==============================] - 1s 38ms/step - loss: 7.4356e-04 - val_loss: 0.001
8
Epoch 4/50
14/14 [==============================] - 1s 36ms/step - loss: 4.2694e-04 - val_loss: 0.001
4
Epoch 5/50
```

```
14/14 [==============================] - 1s 48ms/step - loss: 3.4548e-04 - val_loss: 0.001
1
Epoch 6/50
14/14 [==============================] - 1s 44ms/step - loss: 3.5666e-04 - val_loss: 0.001
5
Epoch 7/50
14/14 [==============================] - 1s 43ms/step - loss: 3.4101e-04 - val_loss: 0.001
6
Epoch 8/50
14/14 [==============================] - 1s 45ms/step - loss: 3.2600e-04 - val_loss: 0.001
1
Epoch 9/50
14/14 [==============================] - 1s 38ms/step - loss: 3.1237e-04 - val_loss: 0.001
1
Epoch 10/50
14/14 [==============================] - 1s 38ms/step - loss: 3.1321e-04 - val_loss: 0.001
0
Epoch 11/50
14/14 [==============================] - 1s 44ms/step - loss: 3.1176e-04 - val_loss: 9.615
0e-04
Epoch 12/50
14/14 [==============================] - 1s 45ms/step - loss: 3.0640e-04 - val_loss: 0.001
1
Epoch 13/50
14/14 [==============================] - 1s 42ms/step - loss: 3.0303e-04 - val_loss: 9.232
1e-04
Epoch 14/50
14/14 [==============================] - 1s 44ms/step - loss: 2.9403e-04 - val_loss: 0.001
0
Epoch 15/50
14/14 [==============================] - 1s 39ms/step - loss: 2.9458e-04 - val_loss: 9.224
6e-04
Epoch 16/50
14/14 [==============================] - 1s 41ms/step - loss: 2.8718e-04 - val_loss: 9.555
0e-04
Epoch 17/50
14/14 [==============================] - 1s 46ms/step - loss: 2.9672e-04 - val_loss: 9.870
3e-04
Epoch 18/50
14/14 [==============================] - 1s 43ms/step - loss: 2.8598e-04 - val_loss: 9.117
5e-04
Epoch 19/50
14/14 [==============================] - 1s 42ms/step - loss: 2.7390e-04 - val_loss: 8.914
2e-04
Epoch 20/50
14/14 [==============================] - 1s 41ms/step - loss: 2.7558e-04 - val_loss: 9.599
3e-04
Epoch 21/50
14/14 [==============================] - 1s 39ms/step - loss: 2.7213e-04 - val_loss: 8.948
0e-04
Epoch 22/50
14/14 [==============================] - 1s 39ms/step - loss: 2.6875e-04 - val_loss: 9.017
4e-04
Epoch 23/50
14/14 [==============================] - 1s 43ms/step - loss: 2.6387e-04 - val_loss: 8.708
3e-04
Epoch 24/50
14/14 [==============================] - 1s 42ms/step - loss: 2.8013e-04 - val_loss: 9.695
3e-04
Epoch 25/50
14/14 [==============================] - 1s 46ms/step - loss: 2.8670e-04 - val_loss: 0.001
2
Epoch 26/50
14/14 [==============================] - 1s 43ms/step - loss: 2.7280e-04 - val_loss: 9.137
7e-04
Epoch 27/50
```

```
14/14 [==============================] - 1s 42ms/step - loss: 2.5139e-04 - val_loss: 9.775
6e-04
Epoch 28/50
14/14 [==============================] - 1s 38ms/step - loss: 2.5032e-04 - val_loss: 9.855
4e-04
Epoch 29/50
14/14 [==============================] - 1s 38ms/step - loss: 2.4550e-04 - val_loss: 8.542
7e-04
Epoch 30/50
14/14 [==============================] - 1s 39ms/step - loss: 2.5724e-04 - val_loss: 8.709
0e-04
Epoch 31/50
14/14 [==============================] - 1s 40ms/step - loss: 2.6006e-04 - val_loss: 8.478
2e-04
Epoch 32/50
14/14 [==============================] - 1s 41ms/step - loss: 2.3364e-04 - val_loss: 8.512
2e-04
Epoch 33/50
14/14 [==============================] - 1s 40ms/step - loss: 2.3124e-04 - val_loss: 8.535
1e-04
Epoch 34/50
14/14 [==============================] - 1s 39ms/step - loss: 2.2746e-04 - val_loss: 8.558
5e-04
Epoch 35/50
14/14 [==============================] - 1s 40ms/step - loss: 2.6065e-04 - val_loss: 0.001
4
Epoch 36/50
14/14 [==============================] - 1s 43ms/step - loss: 2.6439e-04 - val_loss: 9.158
8e-04
Epoch 37/50
14/14 [==============================] - 1s 44ms/step - loss: 2.3936e-04 - val_loss: 8.462
4e-04
Epoch 38/50
14/14 [==============================] - 1s 45ms/step - loss: 2.2822e-04 - val_loss: 8.421
6e-04
Epoch 39/50
14/14 [==============================] - 1s 43ms/step - loss: 2.2241e-04 - val_loss: 0.001
0
Epoch 40/50
14/14 [==============================] - 1s 39ms/step - loss: 2.3205e-04 - val_loss: 8.289
7e-04
Epoch 41/50
14/14 [==============================] - 1s 38ms/step - loss: 2.3372e-04 - val_loss: 8.646
8e-04
Epoch 42/50
14/14 [==============================] - 1s 43ms/step - loss: 2.2165e-04 - val_loss: 7.806
6e-04
Epoch 43/50
14/14 [==============================] - 1s 38ms/step - loss: 2.2464e-04 - val_loss: 7.882
1e-04
Epoch 44/50
14/14 [==============================] - 1s 39ms/step - loss: 2.1855e-04 - val_loss: 8.130
0e-04
Epoch 45/50
14/14 [==============================] - 1s 41ms/step - loss: 2.1038e-04 - val_loss: 7.621
3e-04
Epoch 46/50
14/14 [==============================] - 1s 39ms/step - loss: 2.2608e-04 - val_loss: 9.024
1e-04
Epoch 47/50
14/14 [==============================] - 1s 39ms/step - loss: 2.3826e-04 - val_loss: 8.852
7e-04
Epoch 48/50
14/14 [==============================] - 1s 42ms/step - loss: 2.2422e-04 - val_loss: 7.416
0e-04
Epoch 49/50
```

```
14/14 [==============================] - 1s 38ms/step - loss: 2.1369e-04 - val_loss: 8.039
2e-04
Epoch 50/50
14/14 [==============================] - 1s 38ms/step - loss: 2.2793e-04 - val_loss: 8.038
7e-04
```

Out[11]:

```
<keras.callbacks.History at 0x24f44d7da00>
```

In [12]:

```python
GRUcells = [tf.keras.layers.GRUCell(n_neurons) for _ in range(n_layers)]
gru = tf.keras.layers.StackedRNNCells(GRUcells, input_shape = (None, n_inputs))
GRUmodel = Sequential()
GRUmodel.add(layers.RNN(gru))
GRUmodel.add(layers.Dense(n_outputs))
GRUmodel.compile(loss=tf.keras.losses.mean_squared_error, optimizer=tf.keras.optimizers.Ad
GRUmodel.fit(x_train, y_train,validation_data=(x_valid, y_valid), batch_size=batch_size, e
```

```
Epoch 1/50
14/14 [==============================] - 2s 45ms/step - loss: 0.0188 - val_loss: 0.0193
Epoch 2/50
14/14 [==============================] - 0s 33ms/step - loss: 0.0015 - val_loss: 0.0056
Epoch 3/50
14/14 [==============================] - 0s 35ms/step - loss: 5.4858e-04 - val_loss: 5.986
3e-04
Epoch 4/50
14/14 [==============================] - 1s 38ms/step - loss: 2.0210e-04 - val_loss: 6.872
5e-04
Epoch 5/50
14/14 [==============================] - 0s 32ms/step - loss: 1.5820e-04 - val_loss: 5.272
9e-04
Epoch 6/50
14/14 [==============================] - 0s 32ms/step - loss: 1.4459e-04 - val_loss: 5.425
3e-04
Epoch 7/50
14/14 [==============================] - 0s 33ms/step - loss: 1.4280e-04 - val_loss: 5.430
3e-04
Epoch 8/50
14/14 [==============================] - 0s 33ms/step - loss: 1.3968e-04 - val_loss: 5.113
6e-04
Epoch 9/50
14/14 [==============================] - 0s 33ms/step - loss: 1.3790e-04 - val_loss: 5.005
8e-04
Epoch 10/50
14/14 [==============================] - 0s 32ms/step - loss: 1.3467e-04 - val_loss: 4.852
7e-04
Epoch 11/50
14/14 [==============================] - 0s 33ms/step - loss: 1.3575e-04 - val_loss: 5.213
7e-04
Epoch 12/50
14/14 [==============================] - 0s 32ms/step - loss: 1.3385e-04 - val_loss: 4.762
4e-04
Epoch 13/50
14/14 [==============================] - 0s 32ms/step - loss: 1.3197e-04 - val_loss: 4.698
3e-04
Epoch 14/50
14/14 [==============================] - 0s 34ms/step - loss: 1.2973e-04 - val_loss: 4.662
8e-04
Epoch 15/50
14/14 [==============================] - 0s 35ms/step - loss: 1.2957e-04 - val_loss: 4.805
9e-04
Epoch 16/50
14/14 [==============================] - 0s 33ms/step - loss: 1.2746e-04 - val_loss: 4.548
5e-04
Epoch 17/50
14/14 [==============================] - 0s 32ms/step - loss: 1.2797e-04 - val_loss: 4.418
7e-04
```

```
Epoch 18/50
14/14 [==============================] - 0s 32ms/step - loss: 1.2652e-04 - val_loss: 4.567
6e-04
Epoch 19/50
14/14 [==============================] - 0s 33ms/step - loss: 1.2329e-04 - val_loss: 4.378
6e-04
Epoch 20/50
14/14 [==============================] - 0s 34ms/step - loss: 1.2385e-04 - val_loss: 4.346
3e-04
Epoch 21/50
14/14 [==============================] - 0s 32ms/step - loss: 1.2269e-04 - val_loss: 4.777
5e-04
Epoch 22/50
14/14 [==============================] - 0s 34ms/step - loss: 1.2224e-04 - val_loss: 4.148
8e-04
Epoch 23/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1761e-04 - val_loss: 4.050
5e-04
Epoch 24/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1673e-04 - val_loss: 4.077
6e-04
Epoch 25/50
14/14 [==============================] - 0s 35ms/step - loss: 1.1911e-04 - val_loss: 3.942
8e-04
Epoch 26/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1440e-04 - val_loss: 3.729
5e-04
Epoch 27/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1148e-04 - val_loss: 3.799
0e-04
Epoch 28/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1201e-04 - val_loss: 3.972
4e-04
Epoch 29/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1405e-04 - val_loss: 3.757
4e-04
Epoch 30/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1104e-04 - val_loss: 3.767
1e-04
Epoch 31/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1566e-04 - val_loss: 3.911
8e-04
Epoch 32/50
14/14 [==============================] - 0s 32ms/step - loss: 1.1194e-04 - val_loss: 3.464
4e-04
Epoch 33/50
14/14 [==============================] - 0s 33ms/step - loss: 1.0445e-04 - val_loss: 3.451
6e-04
Epoch 34/50
14/14 [==============================] - 0s 33ms/step - loss: 1.0572e-04 - val_loss: 3.457
0e-04
Epoch 35/50
14/14 [==============================] - 0s 33ms/step - loss: 1.0352e-04 - val_loss: 3.280
1e-04
Epoch 36/50
14/14 [==============================] - 0s 34ms/step - loss: 1.0230e-04 - val_loss: 3.376
2e-04
Epoch 37/50
14/14 [==============================] - 0s 32ms/step - loss: 1.0073e-04 - val_loss: 3.393
2e-04
Epoch 38/50
14/14 [==============================] - 0s 31ms/step - loss: 1.0056e-04 - val_loss: 3.149
9e-04
Epoch 39/50
14/14 [==============================] - 0s 33ms/step - loss: 1.0053e-04 - val_loss: 3.232
8e-04
```

```
Epoch 40/50
14/14 [==============================] - 0s 33ms/step - loss: 9.8292e-05 - val_loss: 3.096
2e-04
Epoch 41/50
14/14 [==============================] - 0s 36ms/step - loss: 9.9300e-05 - val_loss: 3.349
4e-04
Epoch 42/50
14/14 [==============================] - 0s 33ms/step - loss: 1.0062e-04 - val_loss: 3.076
4e-04
Epoch 43/50
14/14 [==============================] - 0s 34ms/step - loss: 9.7102e-05 - val_loss: 3.092
8e-04
Epoch 44/50
14/14 [==============================] - 0s 34ms/step - loss: 9.8307e-05 - val_loss: 3.350
9e-04
Epoch 45/50
14/14 [==============================] - 0s 32ms/step - loss: 1.0839e-04 - val_loss: 2.968
3e-04
Epoch 46/50
14/14 [==============================] - 0s 31ms/step - loss: 1.0412e-04 - val_loss: 3.710
5e-04
Epoch 47/50
14/14 [==============================] - 1s 37ms/step - loss: 9.8820e-05 - val_loss: 3.228
2e-04
Epoch 48/50
14/14 [==============================] - 0s 34ms/step - loss: 9.5387e-05 - val_loss: 2.968
0e-04
Epoch 49/50
14/14 [==============================] - 0s 33ms/step - loss: 9.1806e-05 - val_loss: 2.945
7e-04
Epoch 50/50
14/14 [==============================] - 0s 31ms/step - loss: 9.1184e-05 - val_loss: 2.895
9e-04
```

Out[12]:

```
<keras.callbacks.History at 0x24f45813d30>
```

5. Predictions

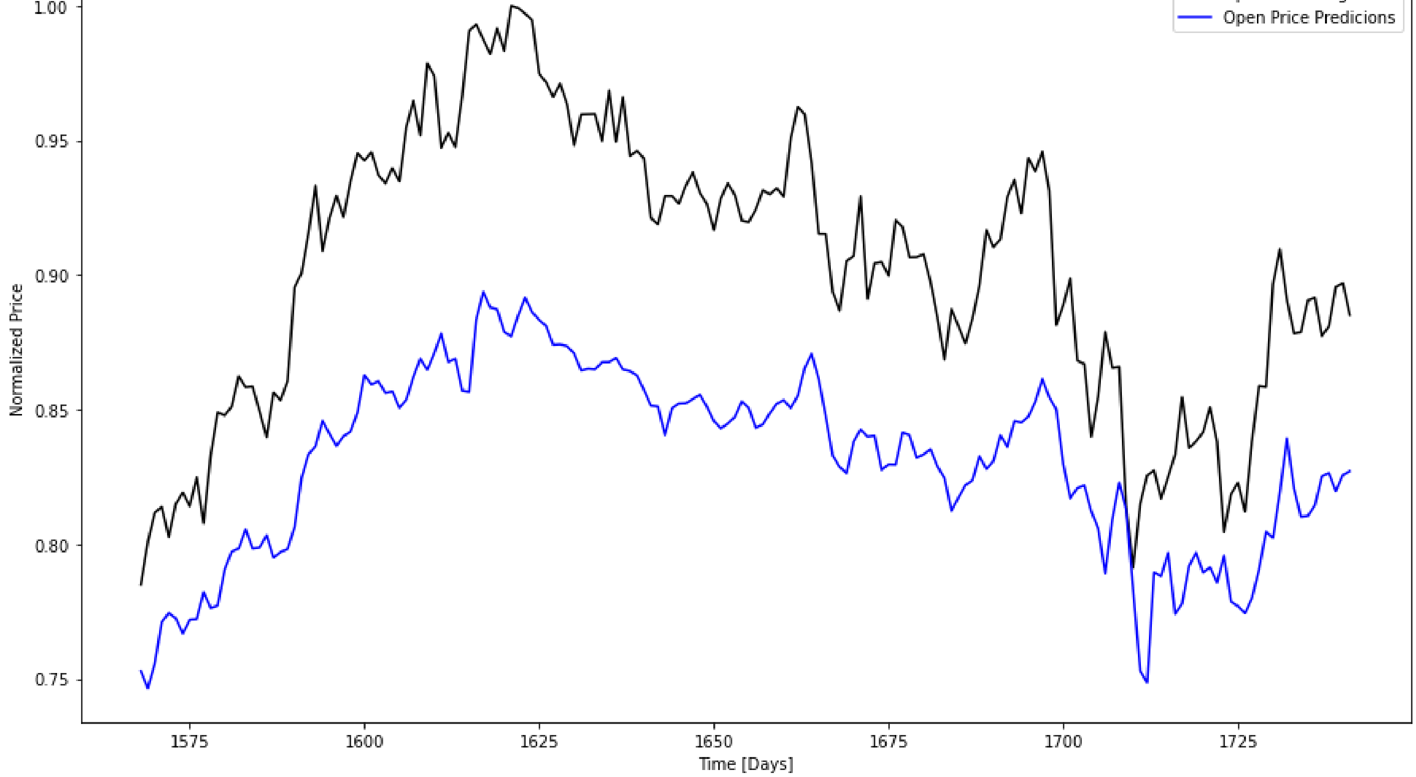
In [13]:

```
# RNN Model Prediction
final = RNNmodel.predict(x_test)
plt.figure(figsize=(15, 40));

# Open Prices Comparison
plt.subplot(4,1,1)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,0],color="black", label="Open Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,0],color="blue", label="Open Price Predicions"
)
plt.title('Future Open Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Close Prices Comparison
```

```python
plt.subplot(4,1,2)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,1],color="black", label="Close Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,1],color="blue", label="Close Price Predicions"
)
plt.title('Future Close Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Low Prices Comparison
plt.subplot(4,1,3)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,2],color="black", label="Low Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,2],color="blue", label="Low Price Predicions"
)
plt.title('Future Low Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# High Prices Comparison
plt.subplot(4,1,4)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,3],color="black", label="High Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,3],color="blue", label="High Price Predicions"
)
plt.title('Future High Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')
```

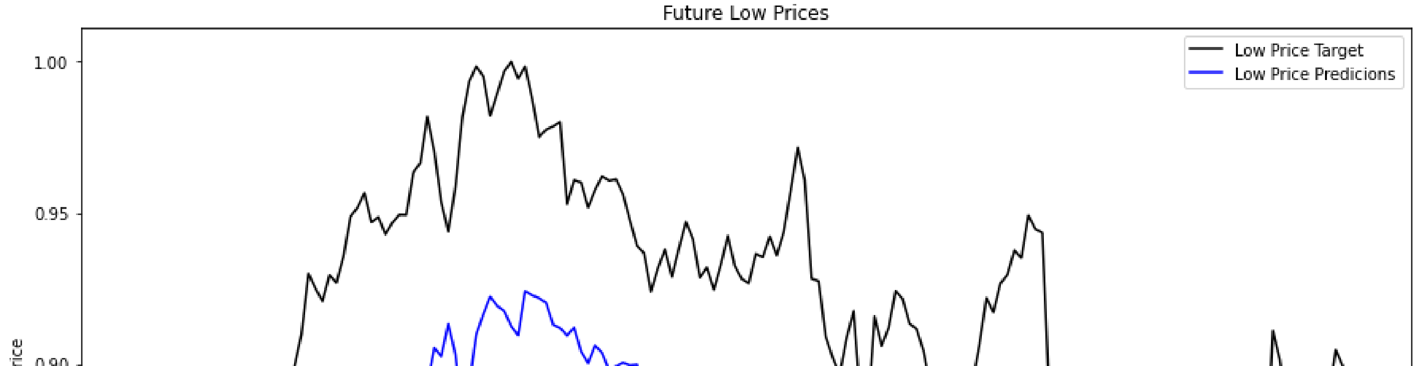
Out[13]:

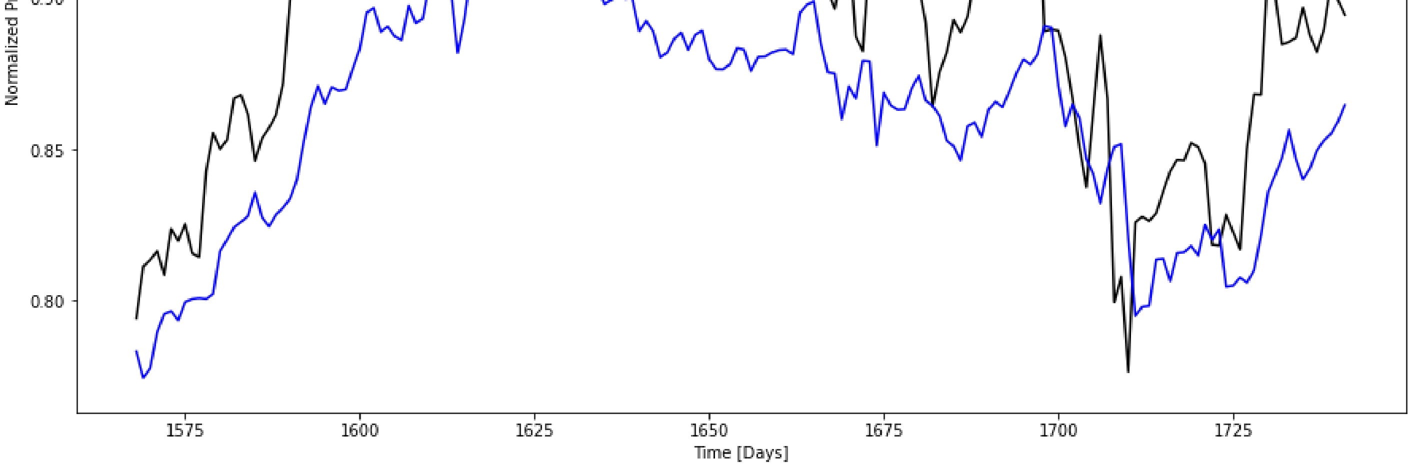
```
<matplotlib.legend.Legend at 0x24f4a100f70>
```

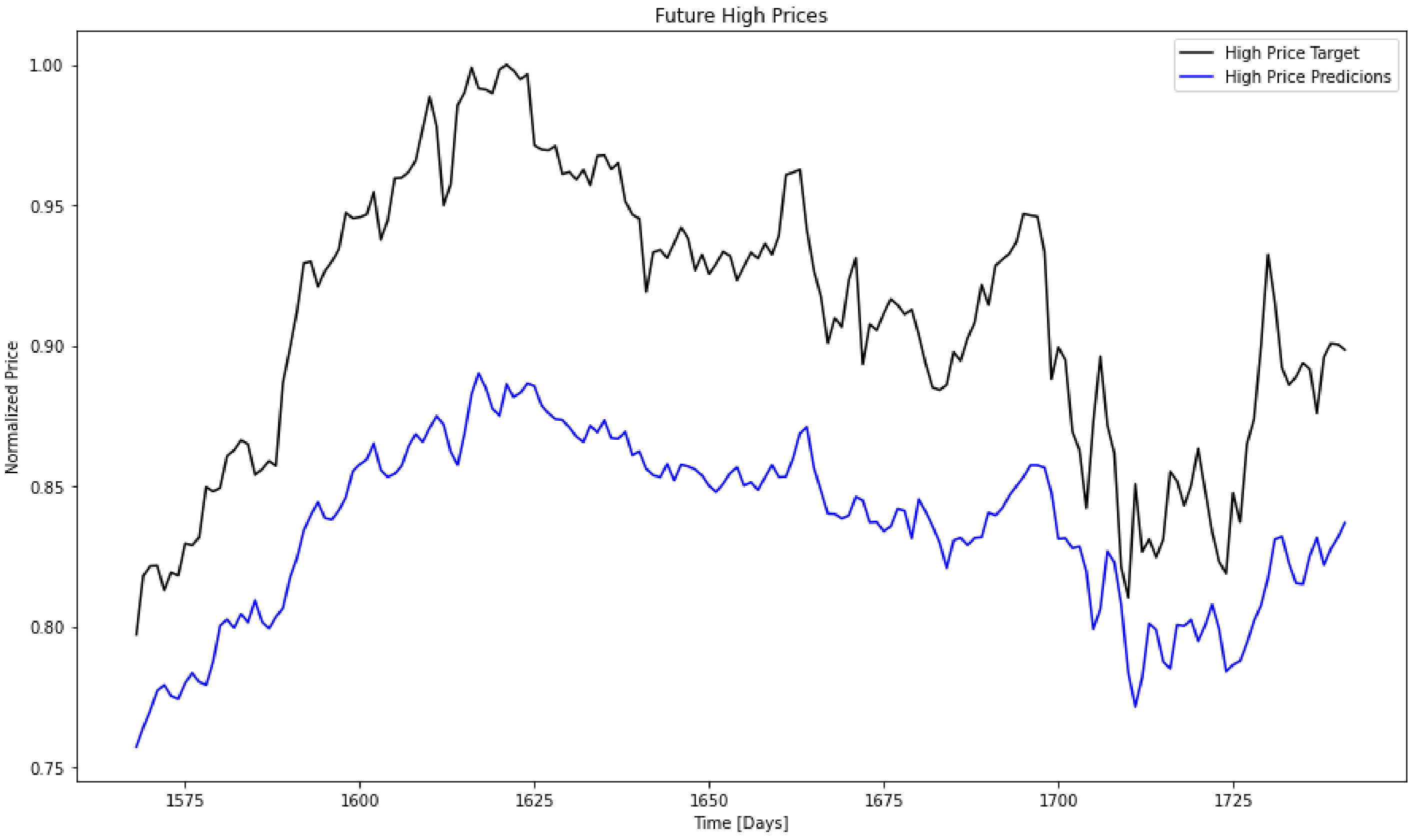


Future Close Prices

Future Low Prices







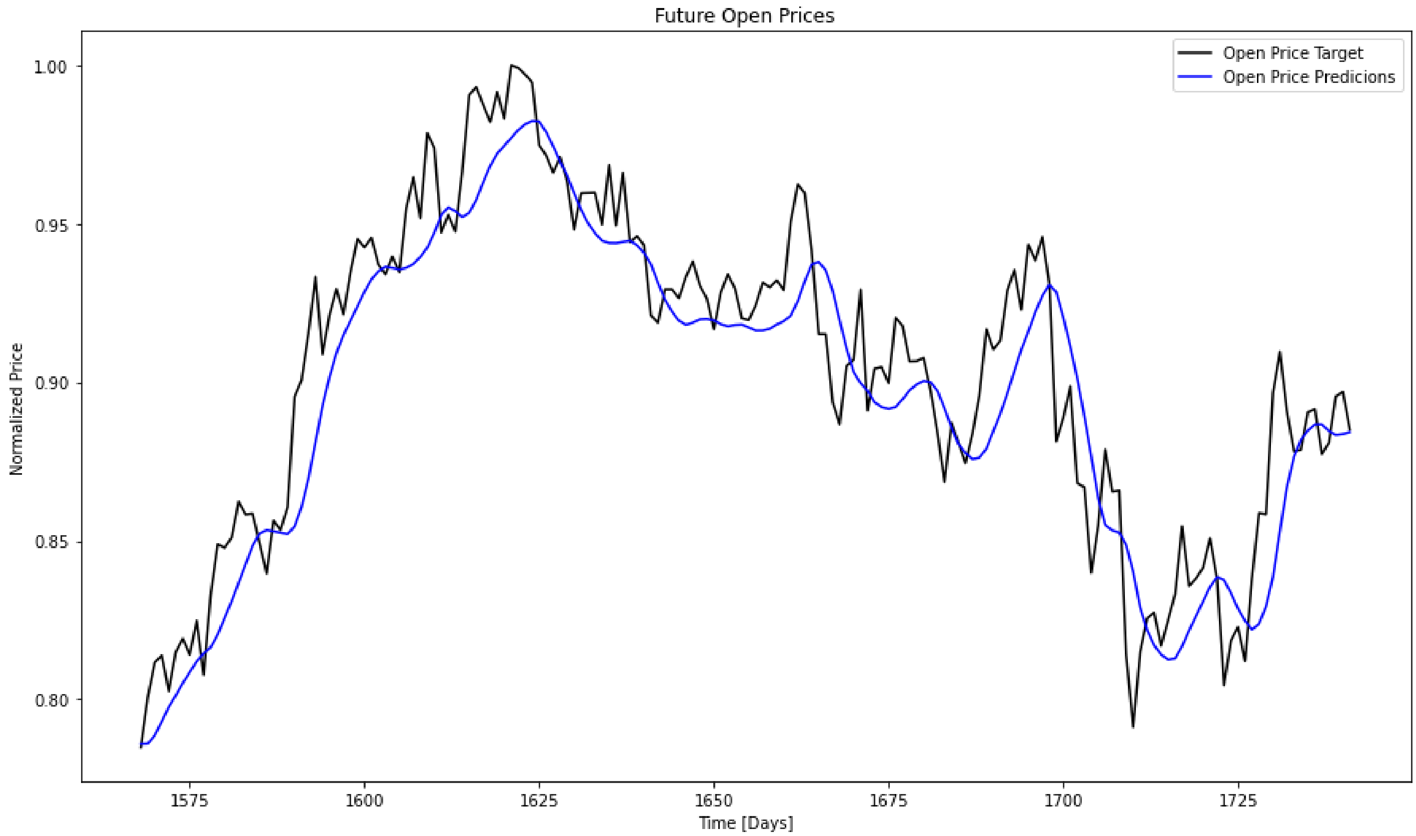
In [14]:

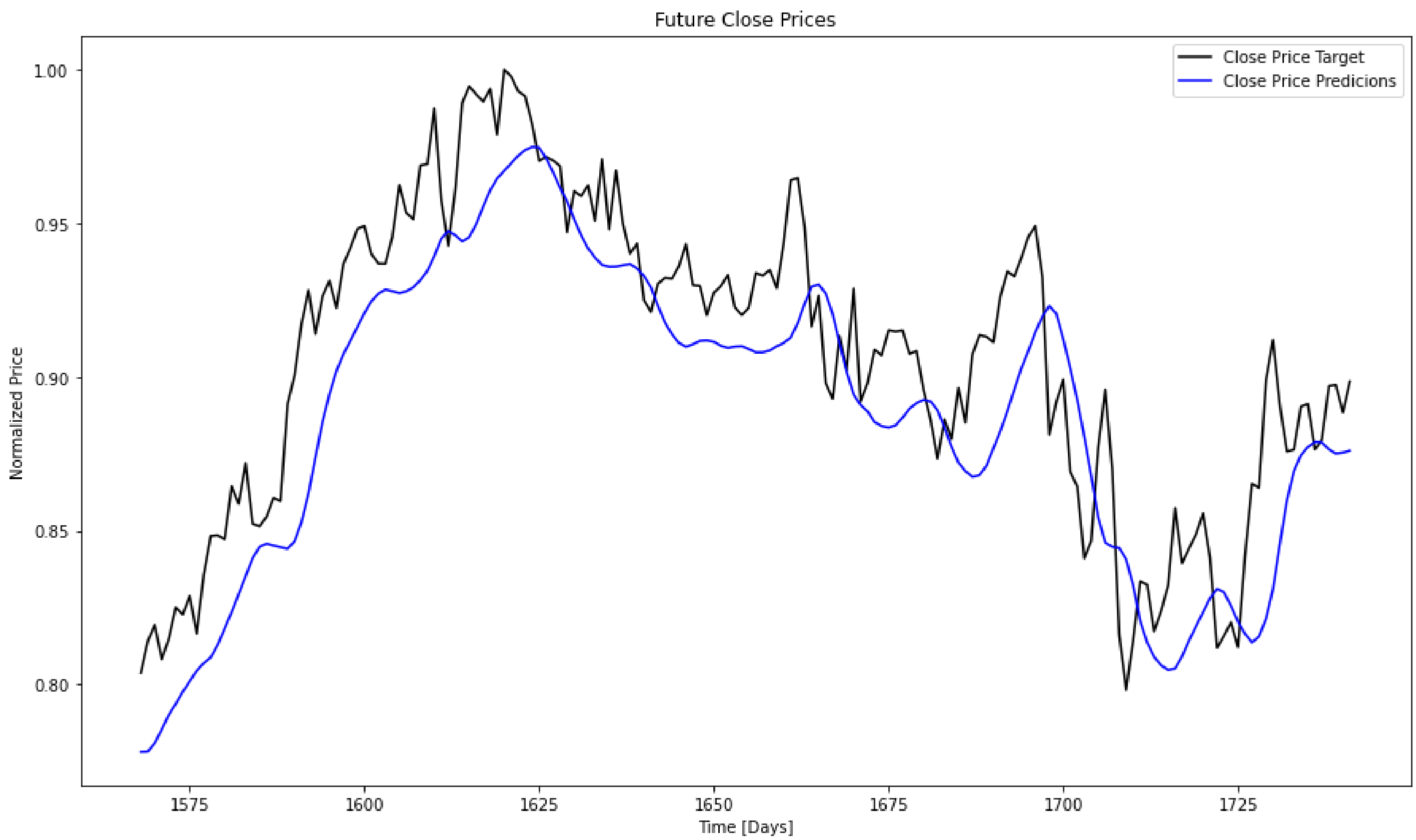
```
# LSTM Model Prediction
final = LSTMmodel.predict(x_test)
plt.figure(figsize=(15, 40));

# Open Prices Comparison
plt.subplot(4,1,1)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,0],color="black", label="Open Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,0],color="blue", label="Open Price Predicions"
)
plt.title('Future Open Prices')
```

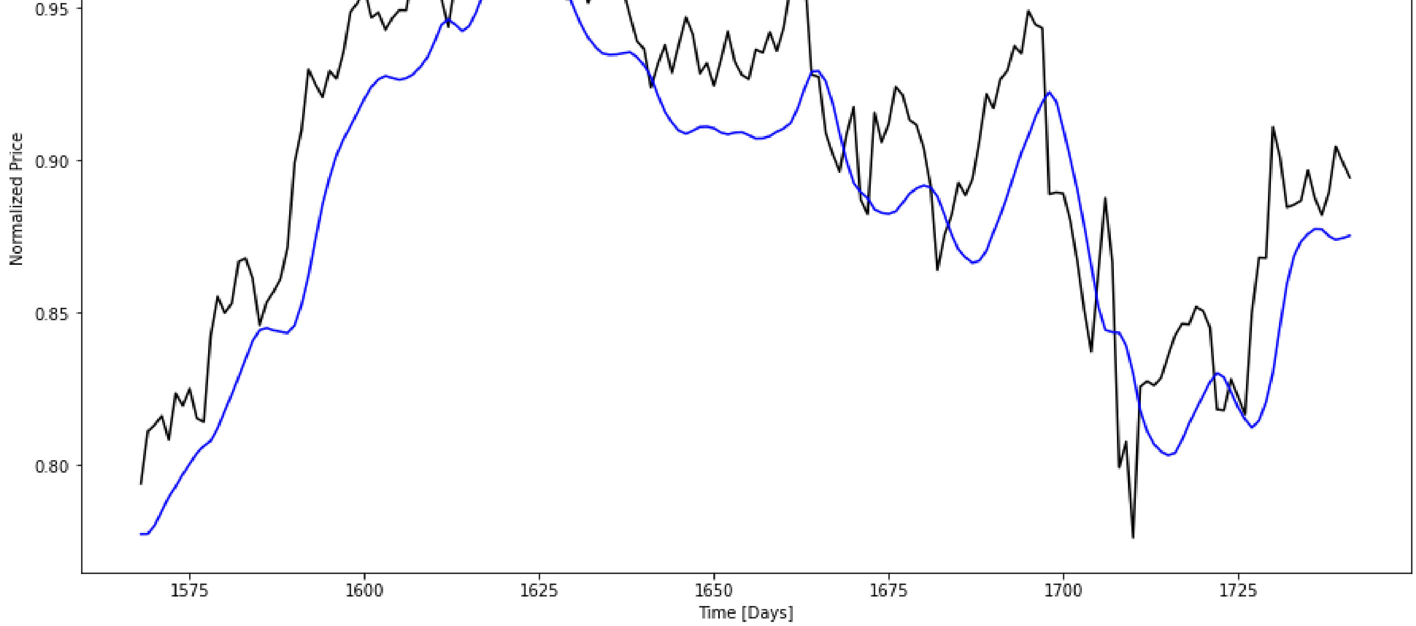
```
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Close Prices Comparison
plt.subplot(4,1,2)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,1],color="black", label="Close Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,1],color="blue", label="Close Price Predicions"
)
plt.title('Future Close Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Low Prices Comparison
plt.subplot(4,1,3)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,2],color="black", label="Low Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,2],color="blue", label="Low Price Predicions"
)
plt.title('Future Low Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# High Prices Comparison
plt.subplot(4,1,4)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,3],color="black", label="High Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,3],color="blue", label="High Price Predicions"
)
plt.title('Future High Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')
```

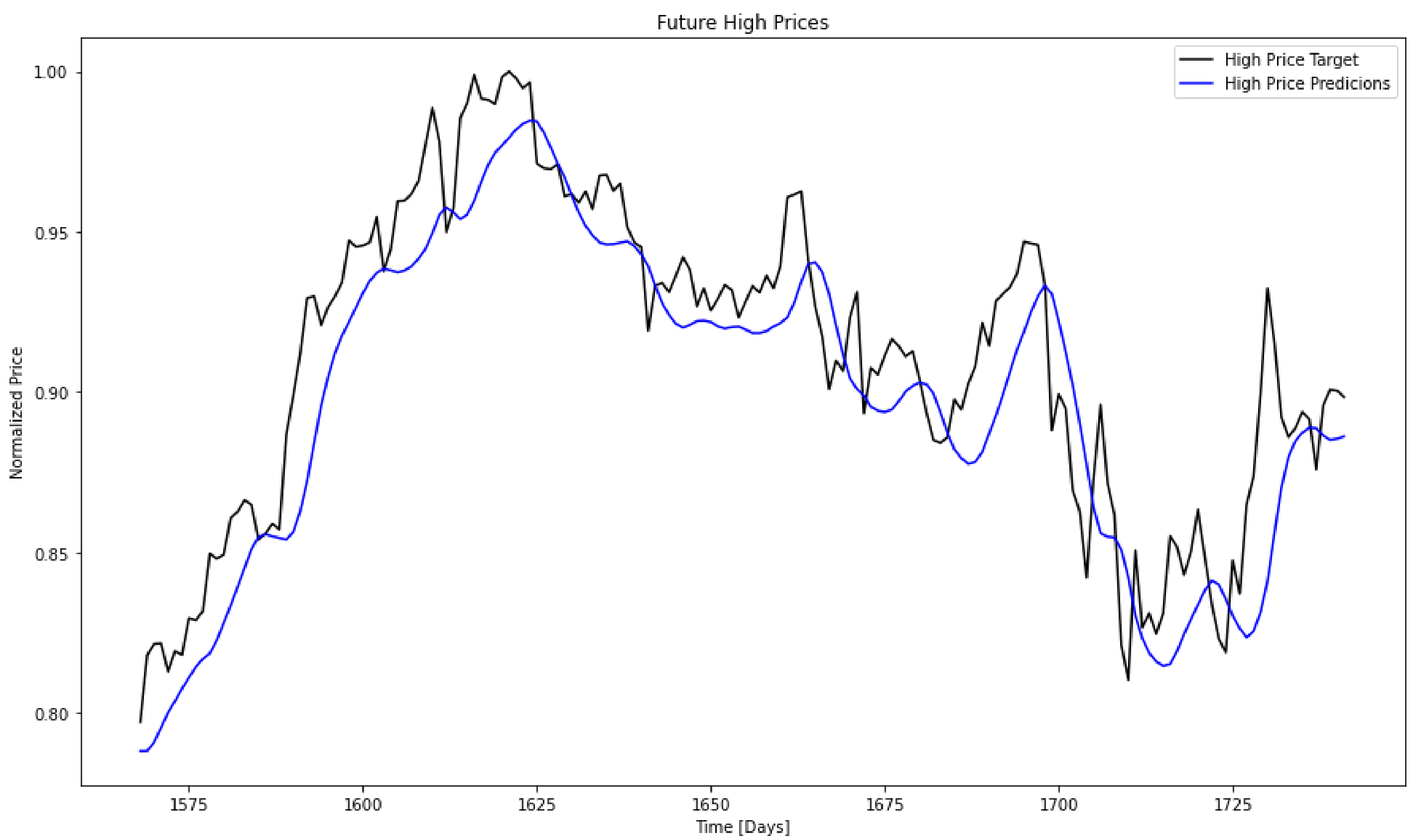
Out[14]:

```
<matplotlib.legend.Legend at 0x24f4c982070>
```









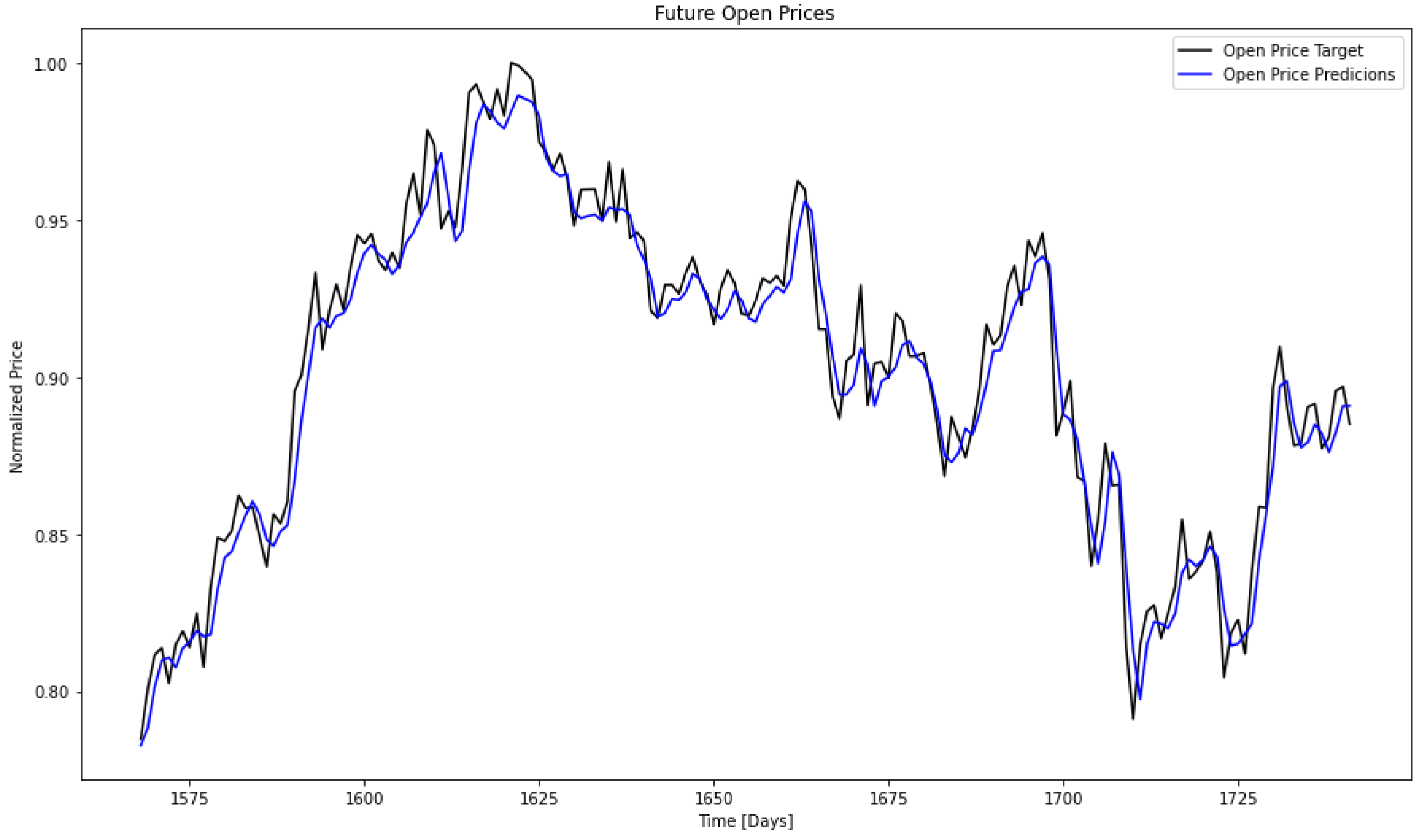
In [15]:

```
# GRU Model Prediction
final = GRUmodel.predict(x_test)
plt.figure(figsize=(15, 40));

# Open Prices Comparison
plt.subplot(4,1,1)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,0],color="black", label="Open Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
```

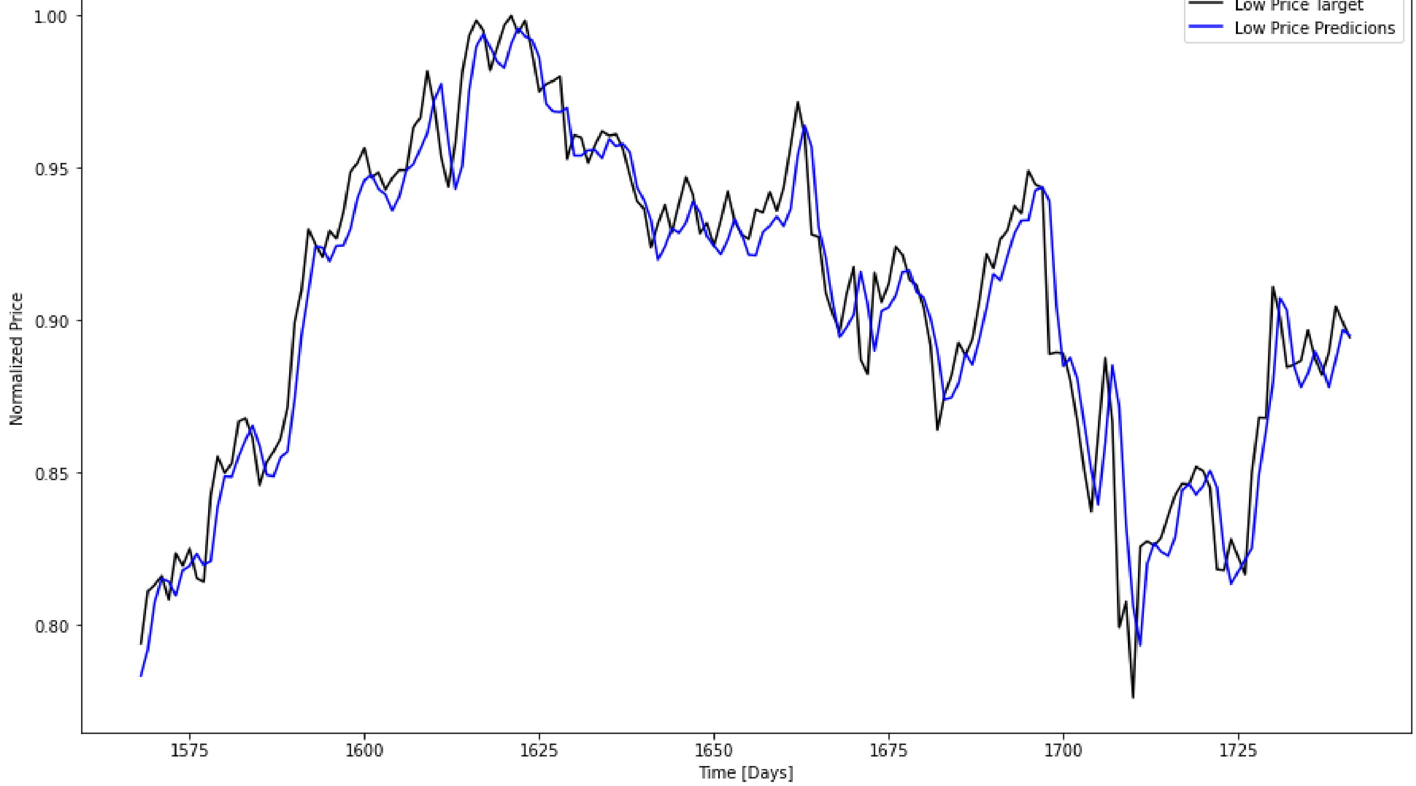
```
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,0],color="blue", label="Open Price Predicions"
)
plt.title('Future Open Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Close Prices Comparison
plt.subplot(4,1,2)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,1],color="black", label="Close Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,1],color="blue", label="Close Price Predicions"
)
plt.title('Future Close Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# Low Prices Comparison
plt.subplot(4,1,3)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,2],color="black", label="Low Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,2],color="blue", label="Low Price Predicions"
)
plt.title('Future Low Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')

# High Prices Comparison
plt.subplot(4,1,4)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    y_test[:,3],color="black", label="High Price Target"
)
plt.plot(
    np.arange(
        y_train.shape[0] + y_valid.shape[0],
        y_train.shape[0] + y_test.shape[0] + y_test.shape[0],
        ),
    final[:,3],color="blue", label="High Price Predicions"
```

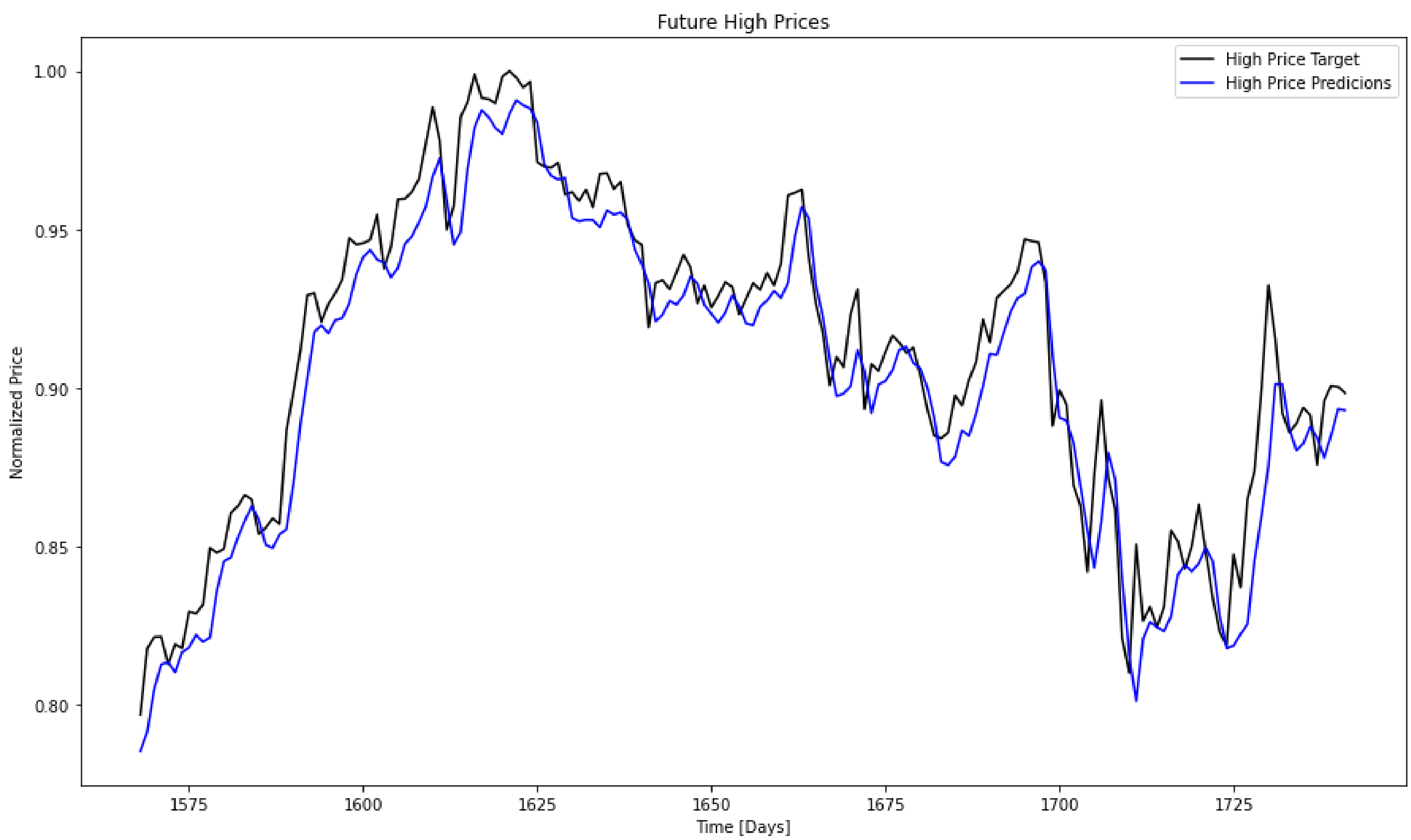
```
)
plt.title('Future High Prices')
plt.xlabel('Time [Days]')
plt.ylabel('Normalized Price')
plt.legend(loc='best')
```

Out[15]: `<matplotlib.legend.Legend at 0x24f4cb99bb0>`



Future Low Prices


Low Price Target
Low Price Predicions
Normalized Price
Time [Days]
1.00
0.95
0.90
0.85
0.80
1575
1600
1625
1650
1675
1700
1725


Future High Prices
High Price Target
High Price Predicions
Normalized Price
Time [Days]
1.00
0.95
0.90
0.85
0.80
1575
1600
1625
1650
1675
1700
1725