Predicting Bitcoin Price Using RNN

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

After reviewing the dataset, Recode the datetime values because it's preferable to have the dataset sorted by date rather than by minutes. Next, Group the dataset by date and calculate the average price for all the minutes in a day to determine the daily price.

In [2]:
```
# Import the dataset and encode the date
df = pd.read_csv('../input/bitstampUSD_1-min_data_2012-01-01_to_2021-03-31.csv')
df['date'] = pd.to_datetime(df['Timestamp'],unit='s').dt.date
group = df.groupby('date')
Real_Price = group['Weighted_Price'].mean()
```

Split the dataset

Predict the BTC price for a month, so we take the data of last 30 days as the test datset.

In [3]:
```
# split data
prediction_days = 30
df_train= Real_Price[:len(Real_Price)-prediction_days]
df_test= Real_Price[len(Real_Price)-prediction_days:]
```

Process Data

Scale the data and reshape it for using Keras

In [4]:
```
# Data preprocess
training_set = df_train.values
training_set = np.reshape(training_set, (len(training_set), 1))
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
training_set = sc.fit_transform(training_set)
X_train = training_set[0:len(training_set)-1]
y_train = training_set[1:len(training_set)]
X_train = np.reshape(X_train, (len(X_train), 1, 1))
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is req
uired for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Build the model

Build the RNN model using Keras.

In [5]:
```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

# Initialising the RNN
regressor = Sequential()

# Adding the input layer and the LSTM layer
regressor.add(LSTM(units = 4, activation = 'sigmoid', input_shape = (None, 1)))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, batch_size = 5, epochs = 100)
```

```
Epoch 1/100
669/669 [==============================] - 6s 2ms/step - loss: 0.0352
Epoch 2/100
669/669 [==============================] - 2s 2ms/step - loss: 0.0103
Epoch 3/100
669/669 [==============================] - 2s 2ms/step - loss: 0.0081
Epoch 4/100
669/669 [==============================] - 2s 2ms/step - loss: 0.0058
Epoch 5/100
669/669 [==============================] - 2s 2ms/step - loss: 0.0034
Epoch 6/100
669/669 [==============================] - 2s 2ms/step - loss: 0.0015
Epoch 7/100
669/669 [==============================] - 2s 2ms/step - loss: 4.6913e-04
Epoch 8/100
669/669 [==============================] - 2s 2ms/step - loss: 1.1160e-04
Epoch 9/100
669/669 [==============================] - 2s 2ms/step - loss: 5.1264e-05
Epoch 10/100
669/669 [==============================] - 2s 2ms/step - loss: 4.6440e-05
Epoch 11/100
669/669 [==============================] - 2s 2ms/step - loss: 4.5273e-05
Epoch 12/100
669/669 [==============================] - 2s 2ms/step - loss: 4.5480e-05
Epoch 13/100
669/669 [==============================] - 2s 2ms/step - loss: 4.4847e-05
Epoch 14/100
669/669 [==============================] - 2s 2ms/step - loss: 4.3528e-05
Epoch 15/100
669/669 [==============================] - 2s 2ms/step - loss: 4.1909e-05
Epoch 16/100
669/669 [==============================] - 2s 2ms/step - loss: 4.1207e-05
Epoch 17/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0940e-05
Epoch 18/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0752e-05
Epoch 19/100
669/669 [==============================] - 2s 3ms/step - loss: 4.0717e-05
Epoch 20/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0337e-05
Epoch 21/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9552e-05
Epoch 22/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0844e-05
Epoch 23/100
```

```
669/669 [==============================] - 2s 2ms/step - loss: 3.9939e-05
Epoch 24/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9876e-05
Epoch 25/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9179e-05
Epoch 26/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0338e-05
Epoch 27/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0569e-05
Epoch 28/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9479e-05
Epoch 29/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8841e-05
Epoch 30/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9795e-05
Epoch 31/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9998e-05
Epoch 32/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9494e-05
Epoch 33/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9249e-05
Epoch 34/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0394e-05
Epoch 35/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9645e-05
Epoch 36/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0515e-05
Epoch 37/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0430e-05
Epoch 38/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9867e-05
Epoch 39/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9917e-05
Epoch 40/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0472e-05
Epoch 41/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9544e-05
Epoch 42/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0054e-05
Epoch 43/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9989e-05
Epoch 44/100
669/669 [==============================] - 2s 2ms/step - loss: 4.1017e-05
Epoch 45/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0787e-05
```

```
Epoch 46/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0347e-05
Epoch 47/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9619e-05
Epoch 48/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9105e-05
Epoch 49/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0454e-05
Epoch 50/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9601e-05
Epoch 51/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8952e-05
Epoch 52/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9690e-05
Epoch 53/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0771e-05
Epoch 54/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9223e-05
Epoch 55/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9389e-05
Epoch 56/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9578e-05
Epoch 57/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0880e-05
Epoch 58/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9135e-05
Epoch 59/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8757e-05
Epoch 60/100
669/669 [==============================] - 2s 3ms/step - loss: 3.8967e-05
Epoch 61/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0186e-05
Epoch 62/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9487e-05
Epoch 63/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9461e-05
Epoch 64/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9500e-05
Epoch 65/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0316e-05
Epoch 66/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9553e-05
Epoch 67/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9737e-05
Epoch 68/100
```

```
669/669 [==============================] - 2s 2ms/step - loss: 3.9663e-05
Epoch 69/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9864e-05
Epoch 70/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0279e-05
Epoch 71/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9655e-05
Epoch 72/100
669/669 [==============================] - 2s 2ms/step - loss: 4.1011e-05
Epoch 73/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9646e-05
Epoch 74/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9274e-05
Epoch 75/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0448e-05
Epoch 76/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0037e-05
Epoch 77/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9349e-05
Epoch 78/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8425e-05
Epoch 79/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9770e-05
Epoch 80/100
669/669 [==============================] - 2s 3ms/step - loss: 3.8653e-05
Epoch 81/100
669/669 [==============================] - 2s 2ms/step - loss: 4.0982e-05
Epoch 82/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9123e-05
Epoch 83/100
669/669 [==============================] - 2s 2ms/step - loss: 4.1137e-05
Epoch 84/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9863e-05
Epoch 85/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8776e-05
Epoch 86/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8863e-05
Epoch 87/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9386e-05
Epoch 88/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9412e-05
Epoch 89/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9096e-05
Epoch 90/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9569e-05
```

```
Epoch 91/100
669/669 [==============================] - 2s 2ms/step - loss: 3.7886e-05
Epoch 92/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9302e-05
Epoch 93/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8737e-05
Epoch 94/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9093e-05
Epoch 95/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9843e-05
Epoch 96/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9890e-05
Epoch 97/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9667e-05
Epoch 98/100
669/669 [==============================] - 2s 2ms/step - loss: 3.9383e-05
Epoch 99/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8613e-05
Epoch 100/100
669/669 [==============================] - 2s 2ms/step - loss: 3.8867e-05
```

Out[5]:
```
<keras.callbacks.History at 0x7fb440167fd0>
```

Prediction

To predict the price of the next day based on today's price, it's important to consider that numerous influencing factors are at play, and longer-term predictions are prone to greater errors.

In [6]:
```
# Making the predictions
test_set = df_test.values
inputs = np.reshape(test_set, (len(test_set), 1))
inputs = sc.transform(inputs)
inputs = np.reshape(inputs, (len(inputs), 1, 1))
predicted_BTC_price = regressor.predict(inputs)
predicted_BTC_price = sc.inverse_transform(predicted_BTC_price)
```

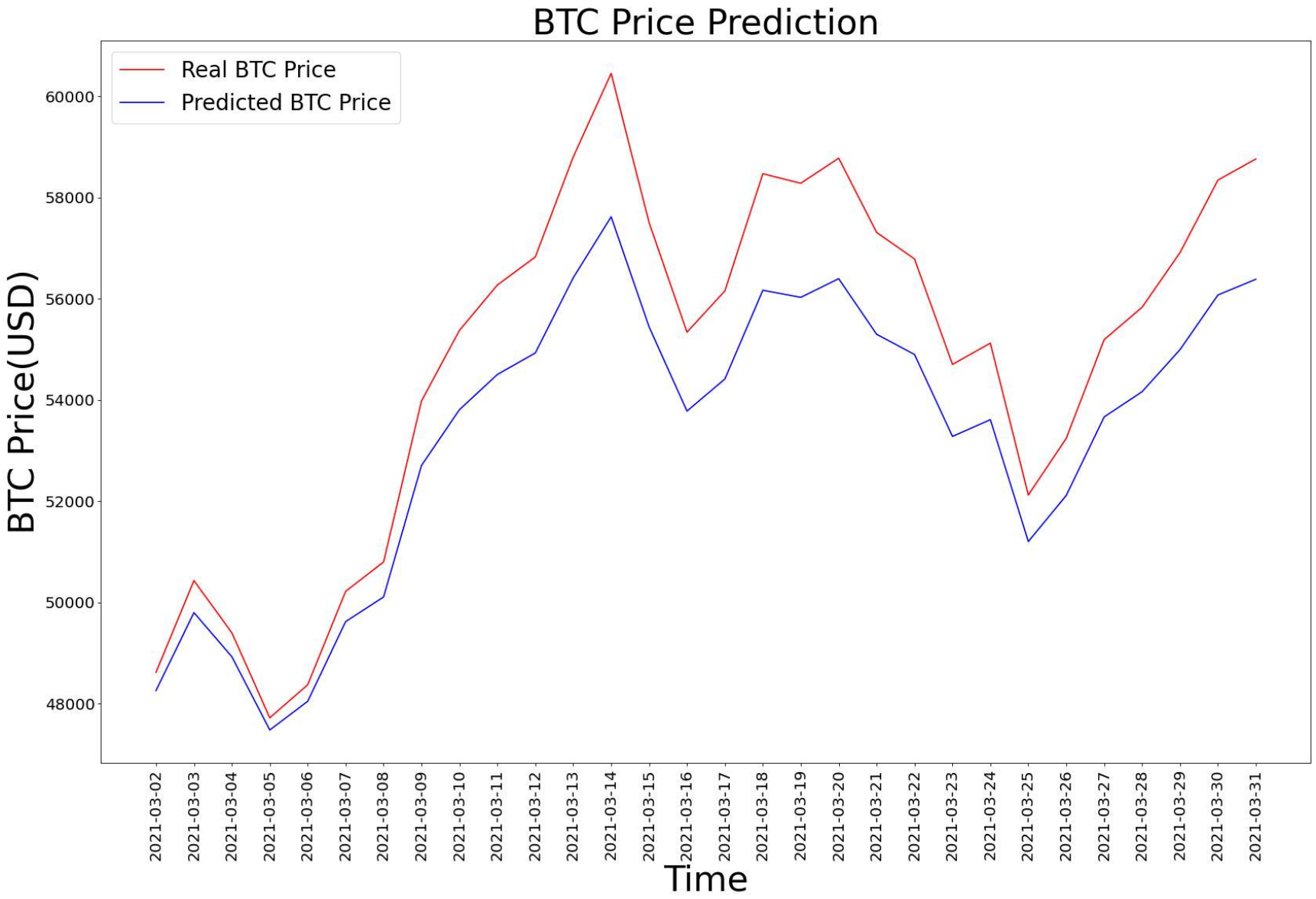
```
1/1 [==============================] - 0s 161ms/step
```

Visualising

Plot both the predicted price and the actual price, and then compare the differences. It's worth noting that the difference tends to increase as the prediction time moves further from the training set. This is precisely why I'm focused on predicting the price for a one-month period.

In [7]:

```
# Visualising the results
plt.figure(figsize=(25,15), dpi=80, facecolor='w', edgecolor='k')
ax = plt.gca()
plt.plot(test_set, color = 'red', label = 'Real BTC Price')
plt.plot(predicted_BTC_price, color = 'blue', label = 'Predicted BTC Price')
plt.title('BTC Price Prediction', fontsize=40)
df_test = df_test.reset_index()
x=df_test.index
labels = df_test['date']
plt.xticks(x, labels, rotation = 'vertical')
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(18)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(18)
plt.xlabel('Time', fontsize=40)
plt.ylabel('BTC Price(USD)', fontsize=40)
plt.legend(loc=2, prop={'size': 25})
plt.show()
```


BTC Price Prediction
Real BTC Price
Predicted BTC Price
BTC Price(USD)
60000
58000
56000
54000
52000
50000
48000
2021-03-02
2021-03-03
2021-03-04
2021-03-05
2021-03-06
2021-03-07
2021-03-08
2021-03-09
2021-03-10
2021-03-11
2021-03-12
2021-03-13
2021-03-14
2021-03-15
2021-03-16
2021-03-17
2021-03-18
2021-03-19
2021-03-20
2021-03-21
2021-03-22
2021-03-23
2021-03-24
2021-03-25
2021-03-26
2021-03-27
2021-03-28
2021-03-29
2021-03-30
2021-03-31
Time