```
# Importing all required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime as dt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import Dense,Dropout,LSTM
from tensorflow.keras.models import Sequential
from google.colab import files
uploaded = files.upload()
```

```
Choose Files file1.csv
• file1.csv(application/vnd.ms-excel) - 55085 bytes, last modified: 5/12/2021 - 100% done
Saving file1.csv to file1.csv
```

```
!pip install tensorflow-gpu==2.0.0-alpha0
```

```
Collecting tensorflow-gpu==2.0.0-alpha0
  Downloading https://files.pythonhosted.org/packages/6a/37/04e98bf98d055a177d5b4ab34
     |████████████████████████████████| 332.5MB 53kB/s
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.7
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: gast>=0.2.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.7/dist-packa
Collecting keras-applications>=1.0.6
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf62
     |████████████████████████████████| 51kB 6.9MB/s
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy<2.0,>=1.14.5 in /usr/local/lib/python3.7/dist-pa
Collecting tf-estimator-nightly<1.14.0.dev2019030116,>=1.14.0.dev2019030115
  Downloading https://files.pythonhosted.org/packages/13/82/f16063b4eed210dc2ab057930
     |████████████████████████████████| 419kB 39.9MB/s
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.7/dist-package
Collecting tb-nightly<1.14.0a20190302,>=1.14.0a20190301
  Downloading https://files.pythonhosted.org/packages/a9/51/aa1d756644bf4624c03844115
     |████████████████████████████████| 3.0MB 38.1MB/s
Requirement already satisfied: google-pasta>=0.1.2 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/loc
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in /u
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Installing collected packages: keras-applications, tf-estimator-nightly, tb-nightly,
Successfully installed keras-applications-1.0.8 tb-nightly-1.14.0a20190301 tensorflow
WARNING: The following packages were previously imported in this runtime:
  [tensorboard,tensorflow]
You must restart the runtime in order to use newly installed versions.
```

RESTART RUNTIME

```
# Read the file
df = pd.read_csv('file1.csv',parse_dates=['Date'])

# Show the file
df.tail(10)
# Last few rows are having NaN so we will have to remove them later
# We have 1442 data points of 60 days.
```

	Unnamed: 0	Date	Time	Price	Unnamed: 4
1439	1439.0	2021-12-05	10:32:21	314187.7489	NaN
1440	1440.0	2021-12-05	11:30:15	316069.7025	NaN
1441	1441.0	2021-12-05	12:31:16	315253.3926	NaN
1442	1442.0	2021-12-05	12:40:41	315508.8576	NaN
1443	NaN	NaT	NaN	NaN	NaN
1444	NaN	NaT	NaN	NaN	NaN
1445	NaN	NaT	NaN	NaN	NaN
1446	NaN	NaT	NaN	NaN	NaN
1447	NaN	NaT	NaN	NaN	NaN
1448	NaN	NaT	NaN	NaN	NaN

```
# Squeeze data in [0,1]
scaler = MinMaxScaler()
price = df.Price.values.reshape(-1, 1)
scaled_price=scaler.fit_transform(price)

# Show data
print(scaled_price)

# Remember that scaled_price is a numpy array
print('\n')

# Remove NaN
scaled_price=scaled_price[~np.isnan(scaled_price)]
scaled_price = scaled_price.reshape(-1, 1)

# Show data
print(scaled_price)
```

```
[[0.07894156]
 [0.07960763]
 [0.08118525]
 ...
 [       nan]
 [       nan]
 [       nan]]


[[0.07894156]
```

```
       [0.07960763]
       [0.08118525]
       ...
       [0.99133343]
       [0.98733219]
       [0.98858438]]


# LSTM data should be 3D - [batch_size, sequence_length, n_features]
# Create functions to arrange data as required
sequence_length=110

# Sequence will take raw data and sequence length and return an numpy array with all data
def sequence(data,sequence_length):
  d = []
  for index in range(len(data) - sequence_length):
        d.append(data[index: index + sequence_length])
  return np.array(d)

def preprocess(data_raw, sequence_length, train_split):
    data = sequence(data_raw, sequence_length)
    num_train = int(train_split * data.shape[0])
    X_train = data[:num_train, :-1, :]
    y_train = data[:num_train, -1, :]
    X_test = data[num_train:, :-1, :]
    y_test = data[num_train:, -1, :]

    return X_train, y_train, X_test, y_test


X_train,y_train, X_test, y_test=preprocess(scaled_price,sequence_length, train_split = 0.9


import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Bidirectional, Dropout, Activation, Dense, LSTM
from tensorflow.python.keras.layers import CuDNNLSTM
from tensorflow.keras.models import Sequential
DROPOUT = 0.2
WINDOW_SIZE = sequence_length - 1
model = keras.Sequential()

model.add(Bidirectional(
  CuDNNLSTM(WINDOW_SIZE, return_sequences=True),
  input_shape=(WINDOW_SIZE, X_train.shape[-1])
))
model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(
  CuDNNLSTM((WINDOW_SIZE * 2), return_sequences=True)
))
model.add(Dropout(rate=DROPOUT))

model.add(Bidirectional(
  CuDNNLSTM(WINDOW_SIZE, return_sequences=False)
))
```

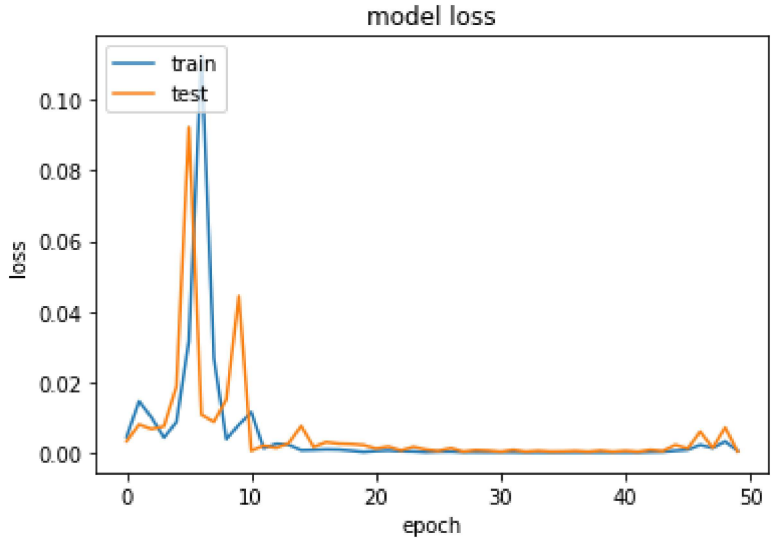
```
''

model.add(Dense(units=1))

model.add(Activation('linear'))


BATCH_SIZE = 64

model.compile(
    loss='mean_squared_error',
    optimizer='adam'
)

history = model.fit(
    X_train,
    y_train,
    epochs=50,
    batch_size=BATCH_SIZE,
    shuffle=False,
    validation_split=0.1
)
```

```
 18/18 [==============================] - 1s 83ms/step - loss: 0.0011 - val_loss:
 Epoch 23/50
 18/18 [==============================] - 1s 84ms/step - loss: 6.2928e-04 - val_lo
 Epoch 24/50
 18/18 [==============================] - 1s 82ms/step - loss: 6.4726e-04 - val_lo
 Epoch 25/50
 18/18 [==============================] - 1s 83ms/step - loss: 2.1180e-04 - val_lo
 Epoch 26/50
 18/18 [==============================] - 1s 83ms/step - loss: 3.7347e-04 - val_lo
 Epoch 27/50
 18/18 [==============================] - 2s 85ms/step - loss: 6.8963e-04 - val_lo
 Epoch 28/50
 18/18 [==============================] - 2s 84ms/step - loss: 2.1697e-04 - val_lo
 Epoch 29/50
 18/18 [==============================] - 1s 84ms/step - loss: 2.1523e-04 - val_lo
 Epoch 30/50
 18/18 [==============================] - 2s 85ms/step - loss: 2.3475e-04 - val_lo
 Epoch 31/50
 18/18 [==============================] - 2s 85ms/step - loss: 2.2617e-04 - val_lo
 Epoch 32/50
 18/18 [==============================] - 1s 83ms/step - loss: 2.5139e-04 - val_lo
 Epoch 33/50
 18/18 [==============================] - 1s 83ms/step - loss: 1.2417e-04 - val_lo
 Epoch 34/50
 18/18 [==============================] - 2s 85ms/step - loss: 1.4509e-04 - val_lo
 Epoch 35/50
 18/18 [==============================] - 2s 84ms/step - loss: 1.1263e-04 - val_lo
 Epoch 36/50
 18/18 [==============================] - 2s 84ms/step - loss: 1.1945e-04 - val_lo
 Epoch 37/50
 18/18 [==============================] - 2s 85ms/step - loss: 1.5568e-04 - val_lo
 Epoch 38/50
 18/18 [==============================] - 2s 84ms/step - loss: 1.4115e-04 - val_lo
 Epoch 39/50
 18/18 [==============================] - 2s 84ms/step - loss: 1.5924e-04 - val_lo
 Epoch 40/50
 18/18 [==============================] - 1s 84ms/step - loss: 1.3706e-04 - val_lo
```

```
Epoch 41/50
18/18 [==============================] - 1s 82ms/step - loss: 1.7264e-04 - val_lo
Epoch 42/50
18/18 [==============================] - 2s 85ms/step - loss: 1.4997e-04 - val_lo
Epoch 43/50
18/18 [==============================] - 2s 85ms/step - loss: 3.6719e-04 - val_lo
Epoch 44/50
18/18 [==============================] - 2s 84ms/step - loss: 4.4828e-04 - val_lo
Epoch 45/50
18/18 [==============================] - 1s 83ms/step - loss: 0.0011 - val_loss:
Epoch 46/50
18/18 [==============================] - 1s 83ms/step - loss: 0.0015 - val_loss:
Epoch 47/50
18/18 [==============================] - 2s 85ms/step - loss: 0.0041 - val_loss:
Epoch 48/50
18/18 [==============================] - 1s 83ms/step - loss: 0.0024 - val_loss:
Epoch 49/50
18/18 [==============================] - 1s 83ms/step - loss: 0.0059 - val_loss:
Epoch 50/50
18/18 [==============================] - 2s 84ms/step - loss: 7.8909e-04 - val_lo
```

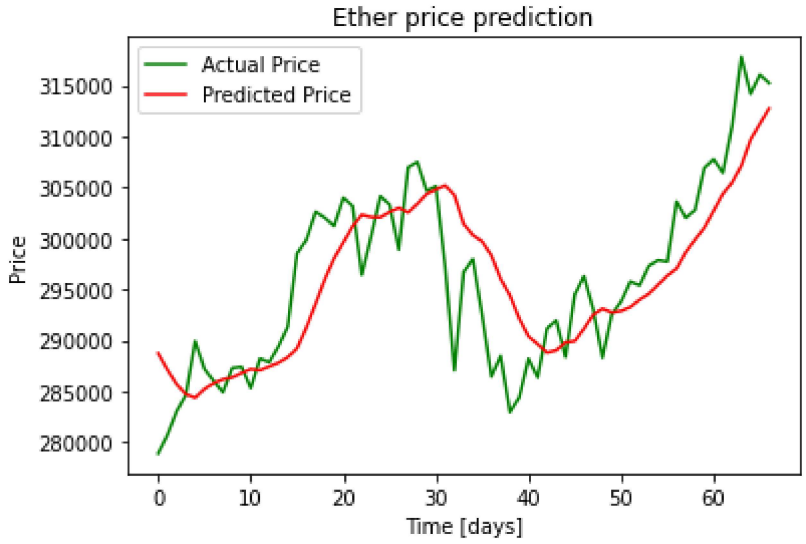
```
# Evaluate loss of model
model.evaluate(X_test, y_test)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
3/3 [==============================] - 0s 36ms/step - loss: 6.7521e-04
```




```
y_hat = model.predict(X_test)

y_test_inverse = scaler.inverse_transform(y_test)
y_hat_inverse = scaler.inverse_transform(y_hat)

plt.plot(y_test_inverse, label="Actual Price", color='green')
```

```
    plt.plot(y_test_inverse, label="Actual Price", color='green')
    plt.plot(y_hat_inverse, label="Predicted Price", color='red')

    plt.title('Ether price prediction')
    plt.xlabel('Time [days]')
    plt.ylabel('Price')
    plt.legend(loc='best')

    plt.show();
```



✓ 0s completed at 4:41 PM