```
from google.colab import files

files.upload()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense


data = pd.read_csv("household_power_consumption.txt", sep=';', parse_dates={'datetime': ['Date', 'Time']}, infer_datetime_format=True, lo

print(data.head())
print(data.dtypes)
print(data.shape)
```

```
<ipython-input-4-c64bb364e323>:1: FutureWarning: Support for nested sequences for 'parse_dates' in pd.read_csv is deprecated. Combi
  data = pd.read_csv("household_power_consumption.txt", sep=';', parse_dates={'datetime': ['Date', 'Time']}, infer_datetime_format=T
<ipython-input-4-c64bb364e323>:1: FutureWarning: The argument 'infer_datetime_format' is deprecated and will be removed in a future
  data = pd.read_csv("household_power_consumption.txt", sep=';', parse_dates={'datetime': ['Date', 'Time']}, infer_datetime_format=T
                     Global_active_power  Global_reactive_power  Voltage  \
datetime
2006-12-16 17:24:00                4.216                  0.418   234.84
2006-12-16 17:25:00                5.360                  0.436   233.63
2006-12-16 17:26:00                5.374                  0.498   233.29
2006-12-16 17:27:00                5.388                  0.502   233.74
2006-12-16 17:28:00                3.666                  0.528   235.68

                     Global_intensity  Sub_metering_1  Sub_metering_2  \
datetime
2006-12-16 17:24:00              18.4             0.0             1.0
2006-12-16 17:25:00              23.0             0.0             1.0
2006-12-16 17:26:00              23.0             0.0             2.0
2006-12-16 17:27:00              23.0             0.0             1.0
2006-12-16 17:28:00              15.8             0.0             1.0

                     Sub_metering_3
datetime
2006-12-16 17:24:00            17.0
2006-12-16 17:25:00            16.0
2006-12-16 17:26:00            17.0
2006-12-16 17:27:00            17.0
2006-12-16 17:28:00            17.0
Global_active_power      float64
Global_reactive_power    float64
Voltage                  float64
Global_intensity         float64
Sub_metering_1           float64
Sub_metering_2           float64
Sub_metering_3           float64
dtype: object
(2075259, 7)
<ipython-input-4-c64bb364e323>:1: UserWarning: Parsing dates in %d/%m/%Y %H:%M:%S format when dayfirst=False (the default) was spec
  data = pd.read_csv("household_power_consumption.txt", sep=';', parse_dates={'datetime': ['Date', 'Time']}, infer_datetime_format=T
```

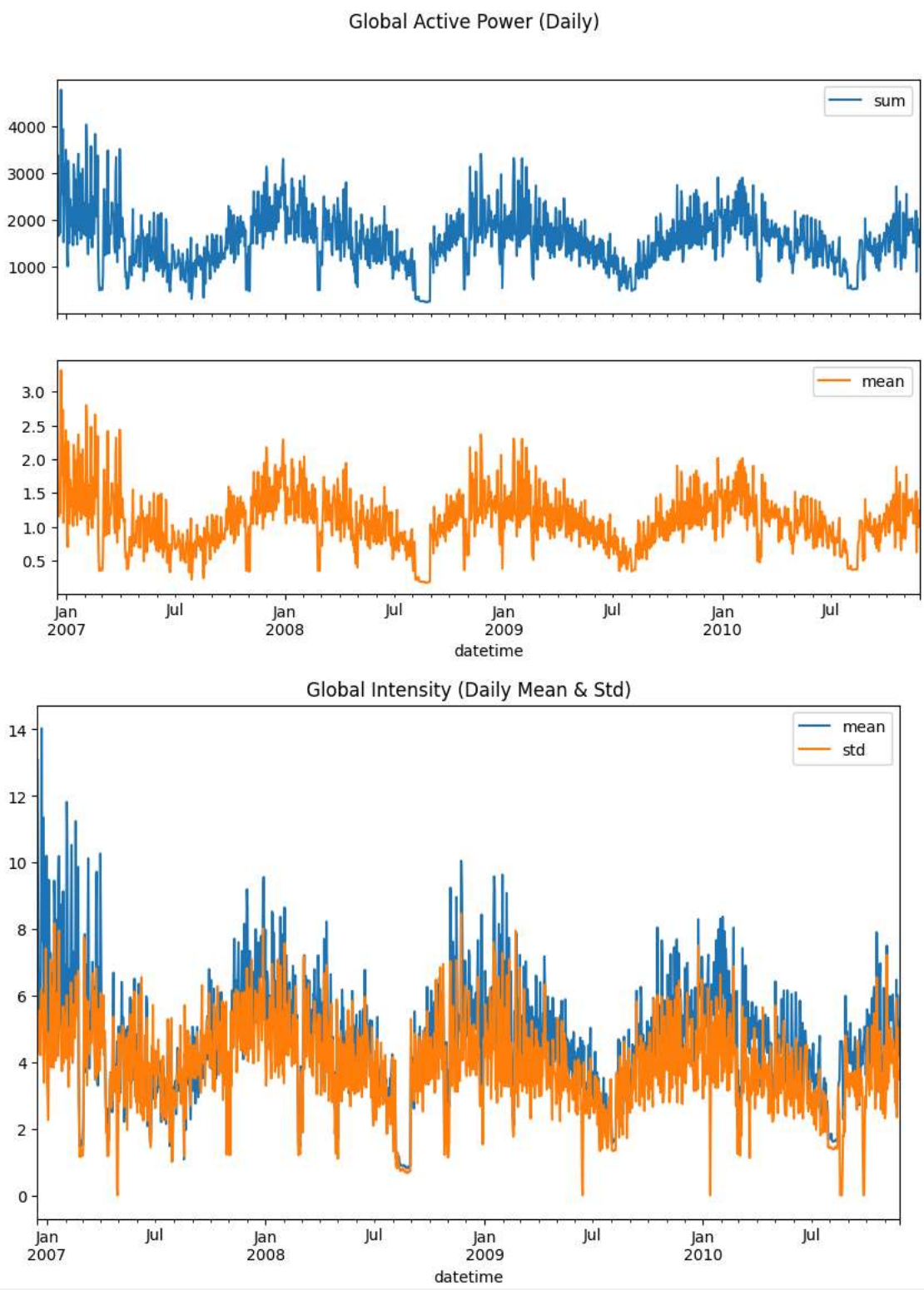
```
data = data.astype(float)
data.fillna(data.mean(), inplace=True)
print(data.isnull().sum())
```

```
Global_active_power      0
Global_reactive_power    0
Voltage                  0
Global_intensity         0
Sub_metering_1           0
Sub_metering_2           0
Sub_metering_3           0
dtype: int64
```

```
data_resampled = data['Global_active_power'].resample('D').agg(['sum', 'mean'])
data_resampled.plot(subplots=True, figsize=(10, 6), title='Global Active Power (Daily)')
plt.show()

data_resampled_intensity = data['Global_intensity'].resample('D').agg(['mean', 'std'])
```

```
data_resampled_intensity.plot(y=['mean', 'std'], figsize=(10, 6), title='Global Intensity (Daily Mean & Std)')
plt.show()
```




```
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
train_size = int(len(data_scaled) * 0.8)
train, test = data_scaled[:train_size], data_scaled[train_size:]

def create_sequences(data, seq_length=10):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length, 0])
    return np.array(X), np.array(y)

seq_length = 10
X_train, y_train = create_sequences(train, seq_length)
X_test, y_test = create_sequences(test, seq_length)
```

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], X_train.shape[2]))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], X_test.shape[2]))


model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(seq_length, data.shape[1])),
    LSTM(50, return_sequences=False),
    Dense(25),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argur
  super().__init__(**kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	11,600
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 25)	1,275
dense_1 (Dense)	(None, 1)	26

Total params: 33,101 (129.30 KB)
Trainable params: 33,101 (129.30 KB)
Non-trainable params: 0 (0.00 B)

```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5, batch_size=32)

plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss over epochs')
plt.show()
```

```
Epoch 1/5
51882/51882 ━━━━━━━━━━━━━━━━━━━━ 342s 7ms/step - loss: 5.7881e-04 - val_loss: 3.7725e-04
Epoch 2/5
51882/51882 ━━━━━━━━━━━━━━━━━━━━ 388s 7ms/step - loss: 5.2237e-04 - val_loss: 3.2890e-04
Epoch 3/5
51882/51882 ━━━━━━━━━━━━━━━━━━━━ 387s 7ms/step - loss: 4.7751e-04 - val_loss: 3.2080e-04
Epoch 4/5
51882/51882 ━━━━━━━━━━━━━━━━━━━━ 383s 7ms/step - loss: 4.6204e-04 - val_loss: 3.0840e-04
Epoch 5/5
51882/51882 ━━━━━━━━━━━━━━━━━━━━ 352s 7ms/step - loss: 4.4051e-04 - val_loss: 3.1128e-04
```