Internet of Things

Assignment 1

Name: Mohd. Rizwan Shaikh
Roll no. IMT2019513

Part 1. Refer to the link provided in the reference section. It provides information on a sensor testbed deployment and data collection from various environmental sensors. Understand the sensor placement in the floor plan, the types and data reading frequency. Download the data file and understand the schema.

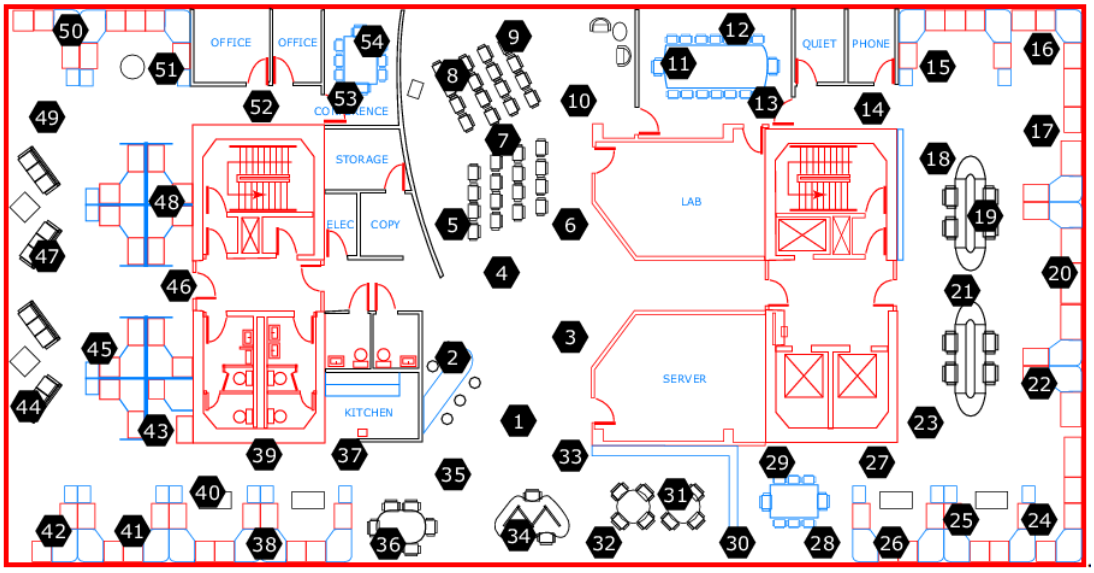

Figure 1: Sensor placement in the floor plan

The dataset used for this assignment consisted data from 54 sensors placed in various locations of on the floor. The data has been collected at a sampling rate of 31 seconds. In the dataset, there are 2313482 datapoints. Each data point gives the following information:

date:yyyy-mm-dd	time:hh:mm:ss.xxx	epoch:int	moteid:int	temperature:real	humidity:real	light:real	voltage:real

The data has been collected from February 28th, 2004 to April 5th, 2004. On having a closer look at the dataset, we realize that some dataset is missing. This is identifiable because of missing epochs in the dataset. Moreover, there are also Nan values in other columns of the dataset. The dataset also gives information about time, humidity, light and voltage recorded by sensors.

This assignment has been solved in Google Colaboratory.

Part 2. Choose a sensor (other than sensor 1) and work with its data set. Find the temporal correlation between the data sets for this sensor.

For this question, I have considered the sensor with MoteId 20.

This sensor had 3 Nan values in temperature, humidity and light columns each. These rows with Nan values were dropped. After dropping the rows, the datapoints were plotted against the epoch number to visualize and have a better understanding of the dataset.

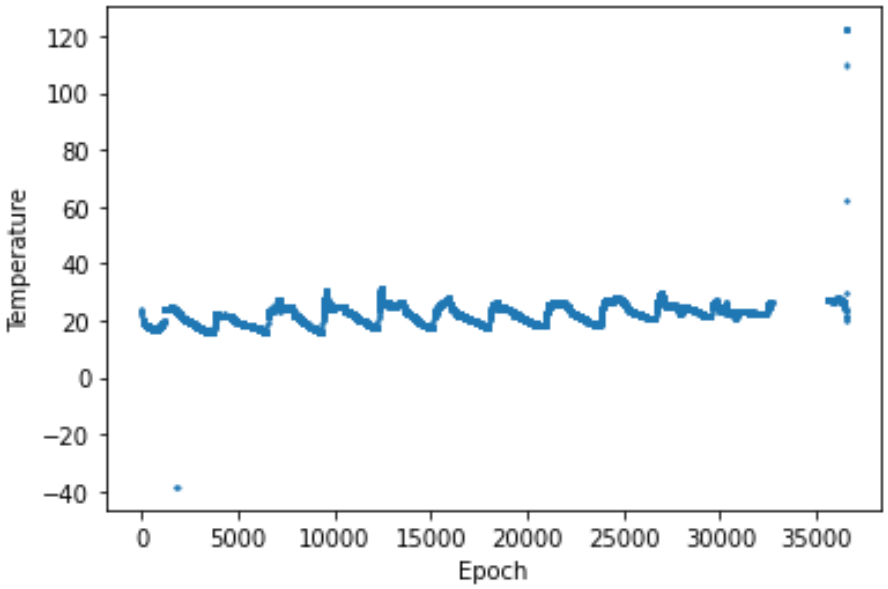

Figure 2: Epoch vs Temperature for sensor 20

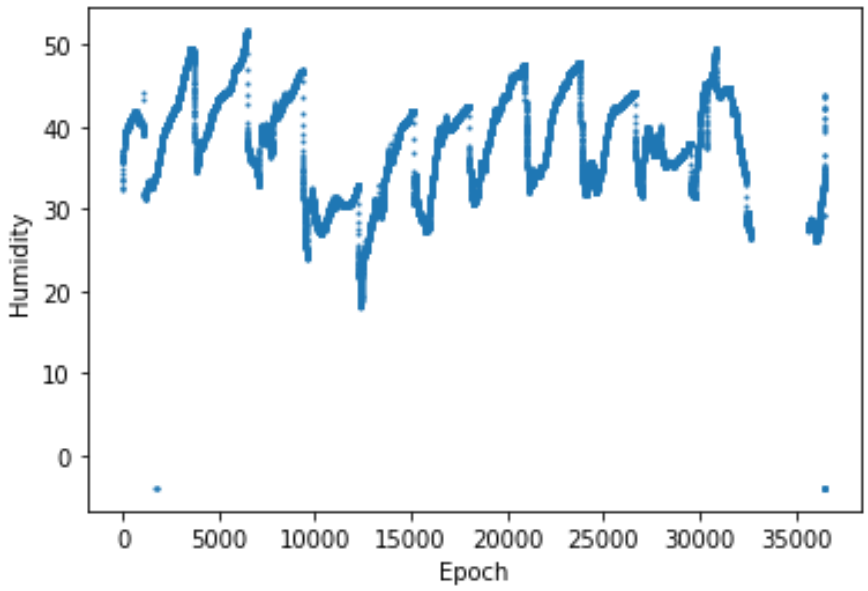

Figure 3: Epoch vs Humidity for sensor 20

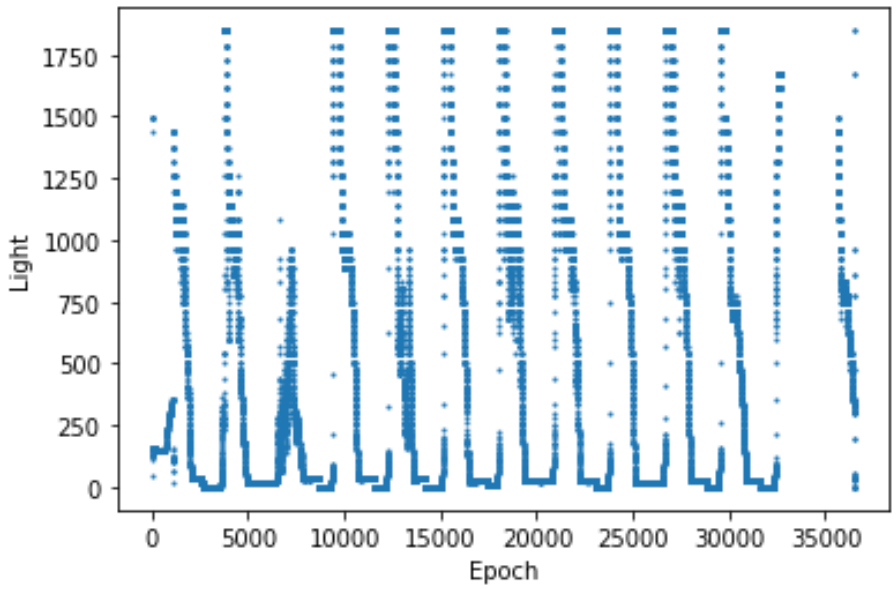

Figure 4: Epoch vs Light for sensor 20

For each of the figures given above (for sensor 20), we find a repeating pattern in the figures. For temperature plot, the repeating pattern is due to the varying temperature over the course of the day. Same is true for humidity and light. In the case of light, the rampant peaking and falling is a result of broad daylight and dark nights.

The *autocorr()* function was used to find temporal correlation between the datasets for the sensor.

The code snippet and the results are shown below:

```
temperature = temporaldata[4]
humidity = temporaldata[5]
light = temporaldata[6]

tempCorr = temperature.autocorr()
humidityCorr = humidity.autocorr()
lightCorr = light.autocorr()
print("Temperature:", tempCorr, ", Humidity:", humidityCorr, ", Light: ", lightCorr)
```

Figure 5: Code to find temporal correlation

- Temporal Correlation in Temperature: 9693772795892756
- Temporal Correlation in Humidity: 0.9942375226538598
- Temporal Correlation in Light: 0.9978655231153823

Here, we observe that the temporal coefficient is high in all the three parameters.

Part 3. Choose any two sensors in proximity of each other. Find the spatial correlation between their data sets pertaining to each sensed parameter. Repeat the same for any two sensors which are located far from each other in the map.

To find the spatial correlation, sensors 16 and 17 and sensors 38 and 54 are used. While the sensors 16 and 17 are in close proximity to each other, the sensors 38 and 54 are located far apart from each other.

To find the spatial correlation, the data from sensors were first stored in separate dataframes. After this, rows with Nan values were dropped. Since such rows were very less in number, dropping the rows has minimal impact on the final output. This was followed by joining the dataframes using the epoch number. The code used is shown below:

```
#spatial1 contains data from sensor 16 and spatial2 contains data from sensor 17
mergedSpatial = pd.merge(spatial1, spatial2, on=2)
mergedSpatial.columns
```

```
Index(['0_x', '1_x',     2, '3_x', '4_x', '5_x', '6_x', '7_x', '0_y', '1_y',
       '3_y', '4_y', '5_y', '6_y', '7_y'],
      dtype='object')
```

The resulting dataframe (mergedSpatial) contained data from both the sensors such that the data is joined using common epoch number.

After this, the spatial correlation was found out using *corr()* function.

The code is as shown below:

```
tempCol1 = mergedSpatial['4_x']
tempCol2 = mergedSpatial['4_y']
HumCol1 = mergedSpatial['5_x']
HumCol2 = mergedSpatial['5_y']
LightCol1 = mergedSpatial['6_x']
LightCol2 = mergedSpatial['6_y']

TemperatureCorr = tempCol1.corr(tempCol2)
HumidityCorr = HumCol1.corr(HumCol2)
LightCorr = LightCol1.corr(LightCol2)
print("Temperature:", TemperatureCorr, "Humidity:", HumidityCorr, "Light:", LightCorr)
```

For sensor 16 and 17 (close proximity to each other), the plots showing the temperatures, humidity and light values as measured by the sensors is shown below:

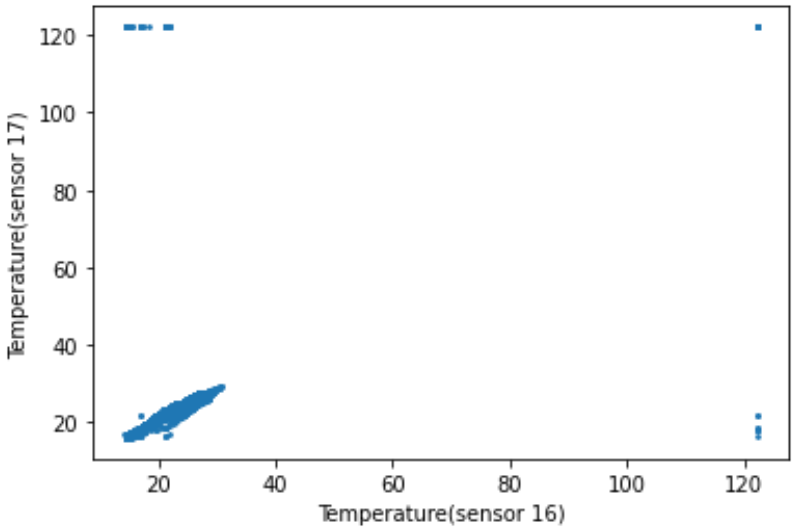

Figure 6: Temperature (sensor 16) vs Temperature (sensor 17)

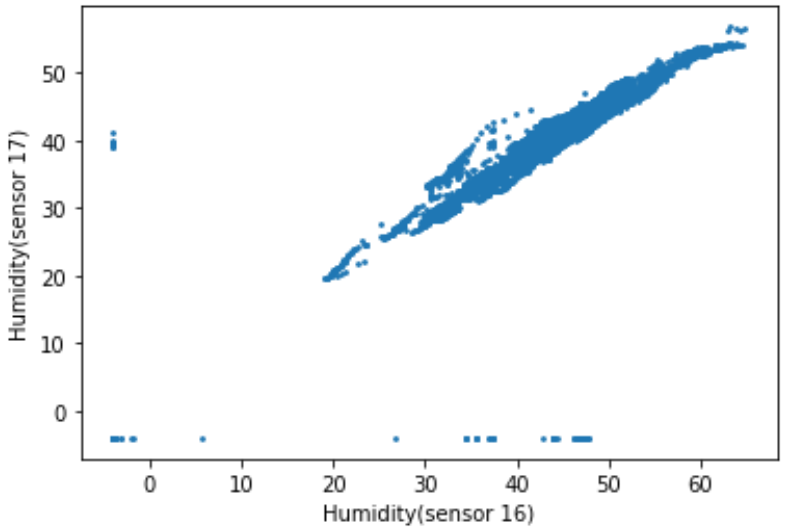

Figure 7: Humidity (sensor 16) vs Humidity (sensor 17)

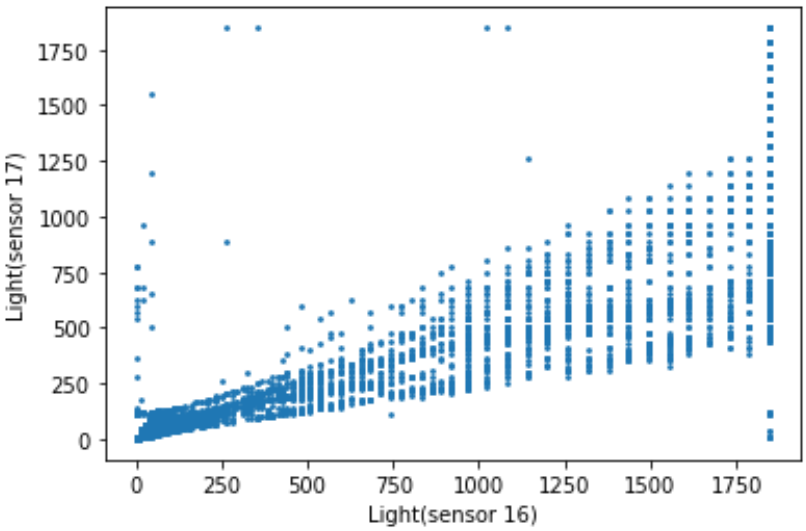

Figure 8: Light (sensor 16) vs Light (sensor 17)

The spatial correlation of dataset of sensor 16 and 17 is as follows:

- Spatial correlation in Temperature: 0.9854808288677805
- Spatial correlation in Humidity: 0.9845158955393034
- Spatial correlation in Light: 0.8977025070592072

For sensors 38 and 54(Located far apart), the plots showing the temperatures, humidity and light values as measured by the sensors is shown below:

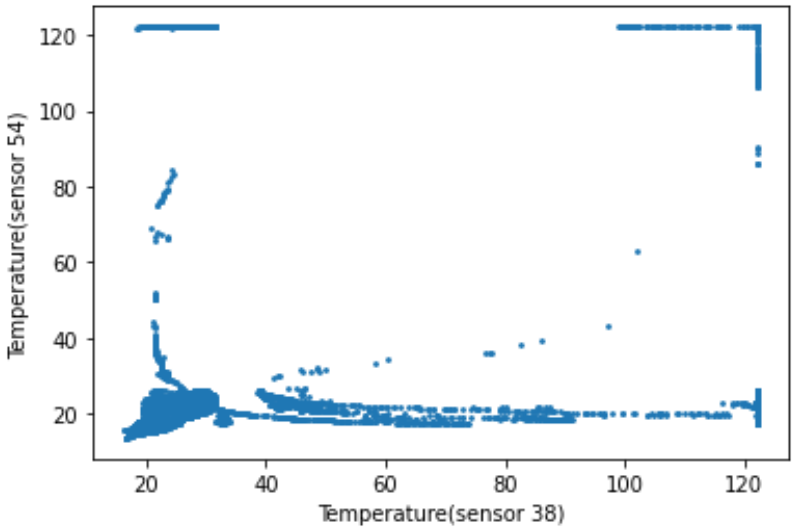

Figure 9: Temperature (sensor 38) vs Temperature (sensor 54)

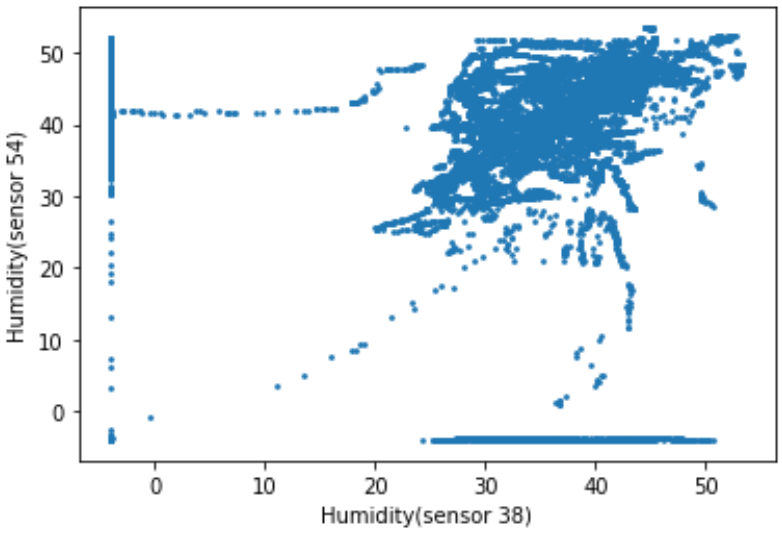

Figure 10: Humidity (sensor 38) vs Humidity (sensor 54)

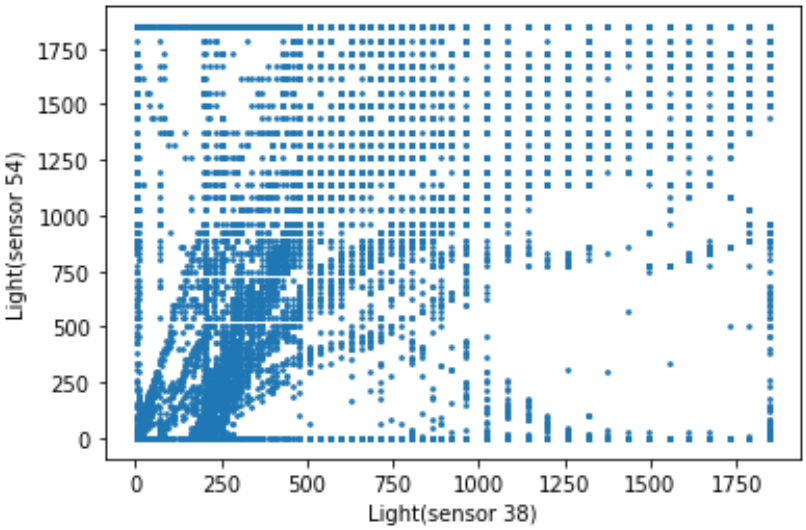

Figure 11: Light (sensor 38) vs Light (sensor 54)

The spatial correlation of dataset of sensor 38 and 54 is as follows:

- Spatial correlation in Temperature: 0.3105463834825988
- Spatial correlation in Humidity: 0.29776444303466604
- Spatial correlation in Light: 0.5125582326239231

Part 4. For sensor 1, save the temperature, relative humidity and light data. Please note the epoch numbers. Complete the missing data by using any prediction technique.

Before proceeding with the prediction, we visualize the data to find some relationship in the data.

The plots of epoch vs temperature, epoch vs humidity and epoch vs light for sensor 1 is shown below:

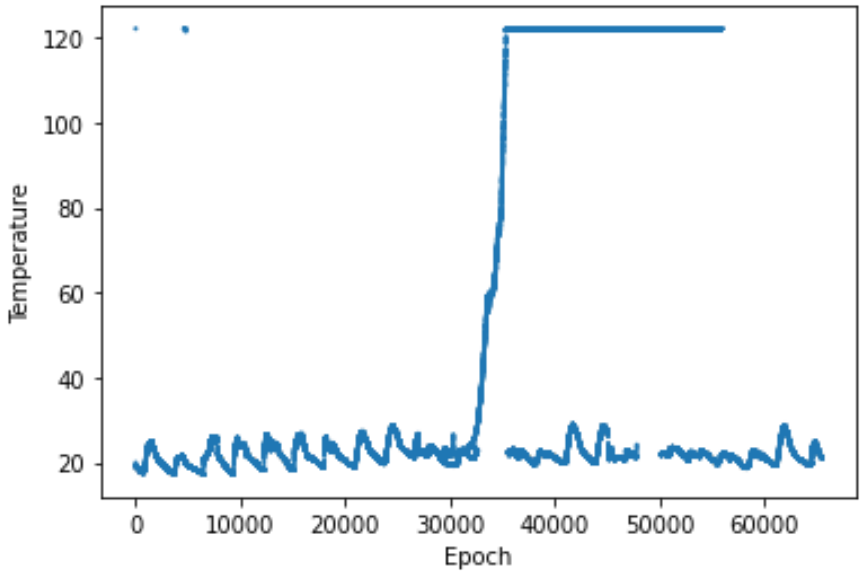

Figure 12: Epoch vs Temperature (sensor 1)

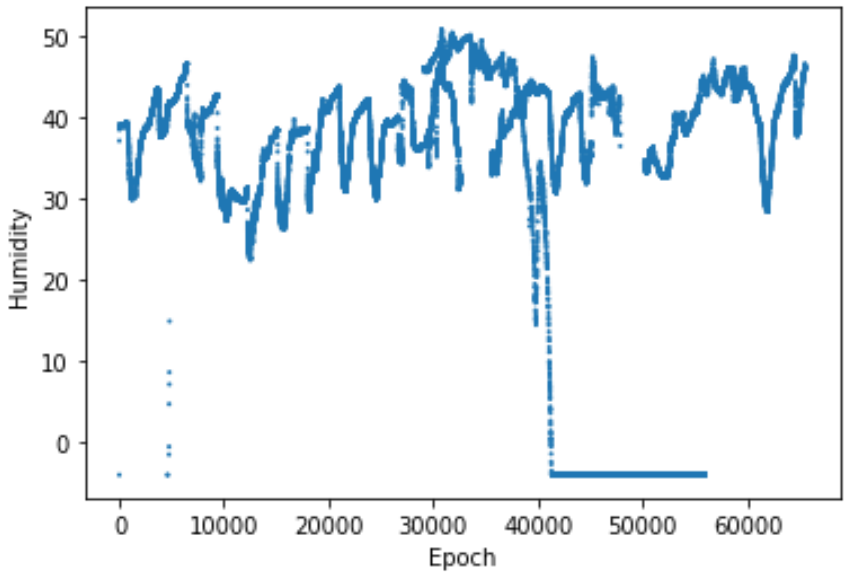

Figure 13: Epoch vs Humidity (sensor 1)

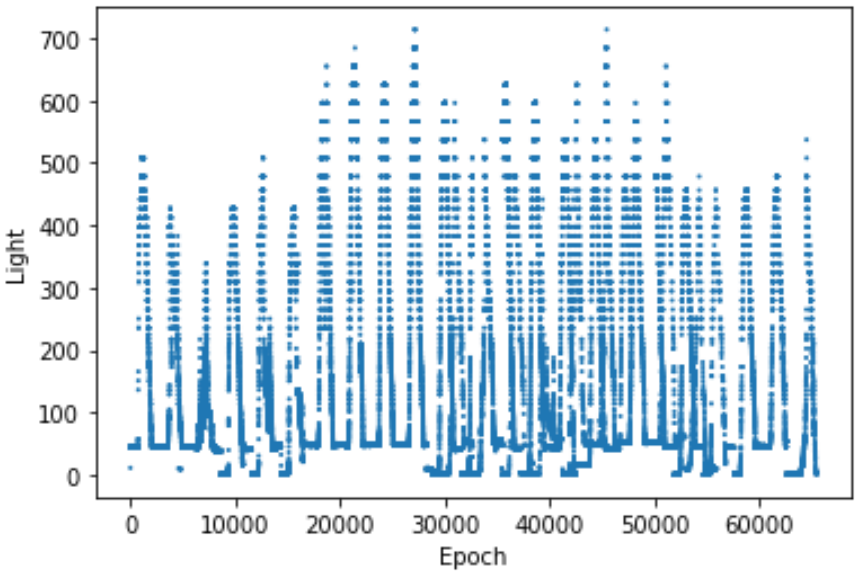

Figure 14: Epoch vs light (sensor 1)

For predicting the missing values, a basic machine learning model is trained. Before training, all the missing epochs are stored in a list and are used as features for prediction.

A simple Linear Regression model is implemented and trained. The input for training is epoch and temperature, epoch and humidity and epoch and light for predicting temperature, humidity and light respectively. The code shown below is used for training the models:

```
TempModel = LinearRegression()
TempModel.fit(np.array(sensor1[2]).reshape(-1, 1), np.array(sensor1[4]).reshape(-1,1))

HumidityModel = LinearRegression()
HumidityModel.fit(np.array(sensor1[2]).reshape(-1, 1), np.array(sensor1[5]).reshape(-1,1))

LightModel = LinearRegression()
LightModel.fit(np.array(sensor1[2]).reshape(-1, 1), np.array(sensor1[6]).reshape(-1,1))
```

The prediction was done as shown below:

```
t = TempModel.predict(np.array(missing).reshape(-1,1))
h = HumidityModel.predict(np.array(missing).reshape(-1,1))
l = LightModel.predict(np.array(missing).reshape(-1,1))
```

Later on, the predicted data was stored in a dataframe.

The predicted values are shown below:

	Epoch	Temperature	Humidity	Light
0	1	16.528929	39.258476	151.928872
1	4	16.530743	39.258013	151.929308
2	5	16.531348	39.257859	151.929453
3	6	16.531953	39.257704	151.929599
4	7	16.532558	39.257550	151.929744
5	8	16.533163	39.257395	151.929889
6	9	16.533768	39.257241	151.930034
7	10	16.534373	39.257087	151.930180
8	12	16.535583	39.256778	151.930470
9	13	16.536188	39.256624	151.930615
10	14	16.536792	39.256469	151.930761

Figure 15: Prediction of missing data

Part 5. Elucidate briefly different techniques (min 4) for predicting missing data. (Read from the available literature and write based on your understanding. Do not copy paste. Cite the references.)

1. Filling with mean/median/mode:

   One of the most basic and commonly used method is to fill the missing value with mean/median/mode. This method works well when the categorical data is skewed. In such a case, it can be assumed safely that the missing value belongs to the majority class and hence can be filled with mode value.

2. Predict the missing value using machine learning:

   In this method, a model is implemented and trained. This model is trained on other input features to predict the missing values. For this method to work, it is necessary that the other features are correlated to the column with missing values.

3. Using K nearest neighbours:

   In this method, 'k' samples are identified that are spatially close or identical in space. Then this 'k' samples are used to estimate the value of missing data points. There are various models that does this work such as KNNImputer.

4. Forward and Backward filling:

   Forward filling means fill missing values with previous data. Backward filling means fill missing values with next data point. This method is useful in datasets with high temporal correlation. We can use the previous and next values of the data to come up with a good guess of current value.

References:

1. Y. Riouali, L. Benhlima and S. Bah, "A benchmark for spatial and temporal correlation based data prediction in wireless sensor networks," 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA), 2015, pp. 1-7, doi: 10.1109/SITA.2015.7358441.