IoT Assignment- 1 Report

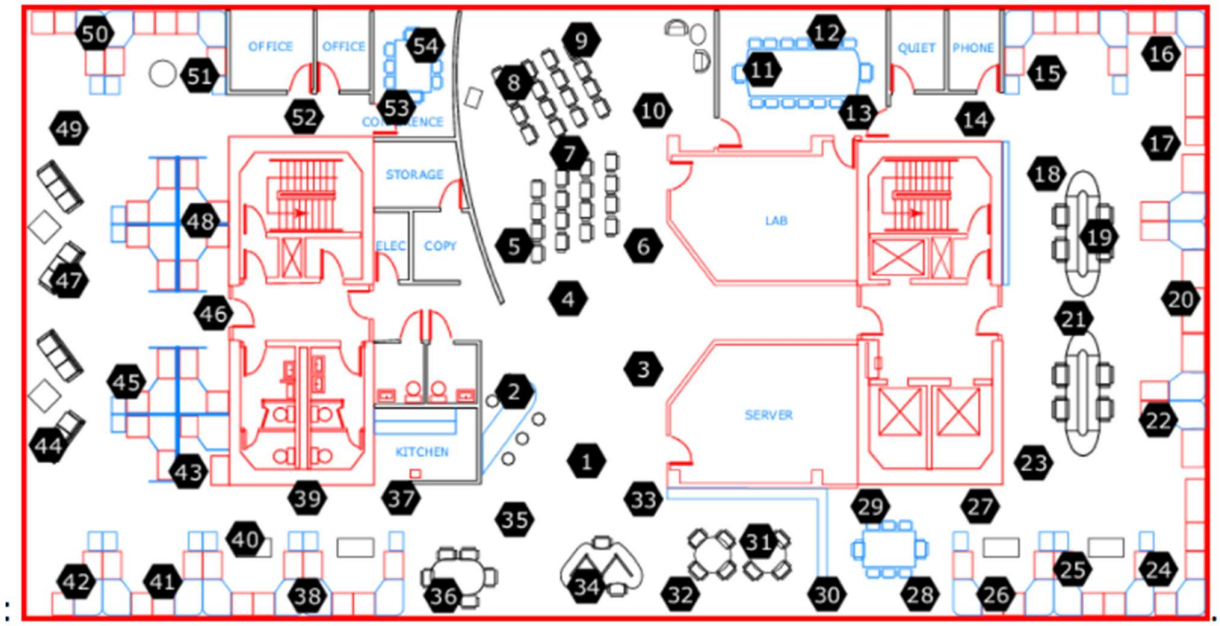
Name: Kalahasti V V Deepesh

Roll Number: IMT2019508

Email ID: deepesh.vv@iiitb.ac.in

Objective: The objective of this lab is to work with available sensor data and to determine the statistical correlation of this data based on its spatial and temporal distance.

Question 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.



Sensor placement

The given dataset contains data collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and April 5th, 2004.

Mica2Dot sensors with weather boards collected timestamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform.

The sensors were arranged in the lab according to the diagram shown above.

The given dataset contains a log of about 2.3 million readings collected from these sensors.

On taking a closer look at the dataset we can see that there is come missing data. In some columns there are NaN values and in others some epochs are missing. The NaN values have been handled by deleting the rows with NaN values.

```
dataset.isna().sum()
```

```
[33]: Date              0
      Time              0
      Epoch             0
      Moteid          526
      Temperature     901
      Humidity        902
      Light         93878
      Voltage         526
      dtype: int64
```

Question 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.

I have chosen sensor 16 for this question. To calculate the temporal correlation, I have used the autocorr function of pandas.

```
auto_correlation_temp = sensor_16_data['Temperature'].autocorr()
auto_correlation_light = sensor_16_data['Light'].autocorr()
auto_correlation_humidity = sensor_16_data['Humidity'].autocorr()
print('Temperature Correlation: ', auto_correlation_temp)
print('Light Correlation: ', auto_correlation_light)
print('Humidity Correlation: ', auto_correlation_humidity)
```

```
Temperature Correlation:  0.9989753895840152
Light Correlation:  0.998792692381538
Humidity Correlation:  0.9988783237943332
```

From the above auto-correlation values, we can see that the temporal correlation is vey high for sensor 16 with the default lag of 1.

We also perform cross-correlation between the temperature, humidity, and light values for sensor_16.

[39]:

```
sensor_16_temp_humidity = sensor_16_data['Temperature'].corr(sensor_16_data['Humidity'])
sensor_16_temp_light = sensor_16_data['Temperature'].corr(sensor_16_data['Light'])
sensor_16_humidity_light = sensor_16_data['Humidity'].corr(sensor_16_data['Light'])
print('Temperature_Humidity Correlation: ', sensor_16_temp_humidity)
print('Temperature_Light Correlation: ', sensor_16_temp_light)
print('Humidity_Light Correlation: ', sensor_16_humidity_light)
```

```
Temperature_Humidity Correlation:  -0.9643012735514049
Temperature_Light Correlation:  -0.07369896059338749
Humidity_Light Correlation:  0.01416710092659697
```

From the above figures we can see that the correlation is very low and the correlation between temperature and humidity is negative which is to be expected.

Question 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.

I have chosen the sensors 16 and 17 to be the datasets which are close together and the sensors 42 and 24 which are far apart from each other.

Then to compute the spatial correlation I merged the datasets for sensors 16,17 and 42,24 such that only those entries are considered which have the same epoch value.

```
temp_16_17_correlation = dataset_16_17['Temperature_x'].corr(dataset_16_17['Temperature_y'])
light_16_17_correlation = dataset_16_17['Light_x'].corr(dataset_16_17['Light_y'])
humidity_16_17_correlation = dataset_16_17['Humidity_x'].corr(dataset_16_17['Humidity_y'])
print('Temperature Correlation: ', temp_16_17_correlation)
print('Light Correlation: ', light_16_17_correlation)
print('Humidity Correlation: ', humidity_16_17_correlation)
```

```
Temperature Correlation:  0.9854808288677805
Light Correlation:  0.8977025070592072
Humidity Correlation:  0.9845158955393034
```

```
temp_42_24_correlation = dataset_42_24['Temperature_x'].corr(dataset_42_24['Temperature_y'])
light_42_24_correlation = dataset_42_24['Light_x'].corr(dataset_42_24['Light_y'])
humidity_42_24_correlation = dataset_42_24['Humidity_x'].corr(dataset_42_24['Humidity_y'])
print('Temperature Correlation: ', temp_42_24_correlation)
print('Light Correlation: ', light_42_24_correlation)
print('Humidity Correlation: ', humidity_42_24_correlation)
```

```
Temperature Correlation:  0.35311317310672036
Light Correlation:  0.46647112814774605
Humidity Correlation:  0.28009870737365855
```

From the above picture we can see that the correlation values are pretty high for the

sensors which closer together and the values are low for sensors which are farther apart.

This signifies those sensors which are closer to each other will log similar values and sensors which are farther apart will log different values.

Question 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.

For predicting the missing epoch values in sensor 1 data I have used the linear regression prediction model.

First, I found out which epochs were missing, then I used the epochs which were present to train the linear regression model. After the model was trained, I predicted the missing epoch values.

```
predictor = LinearRegression()
max_epoch = max(sensor_1_epoch)
missing_epochs = []
for i in range(1, max_epoch+1):
    if i not in sensor_1_epoch:
        missing_epochs.append(i)
```

+ Code + Markdown

```
#Plots of data

plt.scatter(sensor_1_epoch, sensor_1_temp)
plt.xlabel('Epochs')
plt.ylabel('Temperature')
plt.show()
```

```
plt.scatter(sensor_1_epoch, sensor_1_light)
plt.xlabel('Epochs')
plt.ylabel('Light')
plt.show()
```

[ ]:

```
plt.scatter(sensor_1_epoch, sensor_1_humidity)
plt.xlabel('Epochs')
plt.ylabel('Humidity')
plt.show()
```

[ ]:

```
sensor_1_temp = np.array(sensor_1_temp).reshape(-1,1)
sensor_1_humidity = np.array(sensor_1_humidity).reshape(-1,1)
sensor_1_light = np.array(sensor_1_light).reshape(-1,1)
sensor_1_epoch = np.array(sensor_1_epoch).reshape(-1,1)
missing_epochs = np.array(missing_epochs).reshape(-1,1)
```

[ ]:

```
#Predicting temperature

reg = predictor.fit(sensor_1_epoch, sensor_1_temp)
predicted_temperature = predictor.predict(missing_epochs)
```

```
#Predicting light

reg = predictor.fit(sensor_1_epoch, sensor_1_light)
predicted_light = predictor.predict(missing_epochs)
```

+ Code + Markdown

```
#Predicting humidity

reg = predictor.fit(sensor_1_epoch, sensor_1_humidity)
predicted_humidity = predictor.predict(missing_epochs)
```

```
concatenated = np.concatenate((missing_epochs, predicted_temperature,predicted_light, predicted_humidity),axis=1)
```

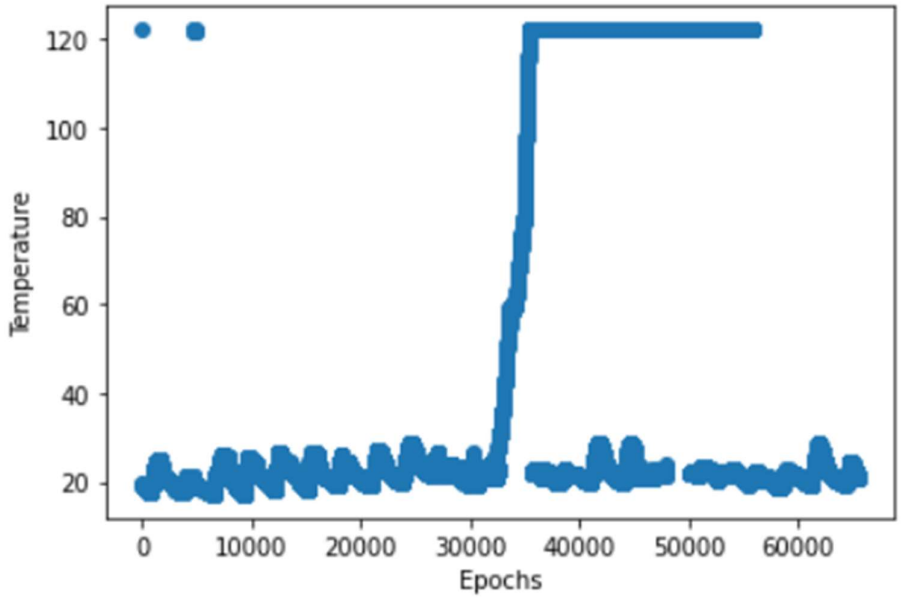
```
missing_values_dataframe = pd.DataFrame(concatenated, columns = ['missing_epochs', 'predicted_temperature','predicted_light'
missing_values_dataframe.head()
```

Code snippets for prediction

Plots for obsereved values:

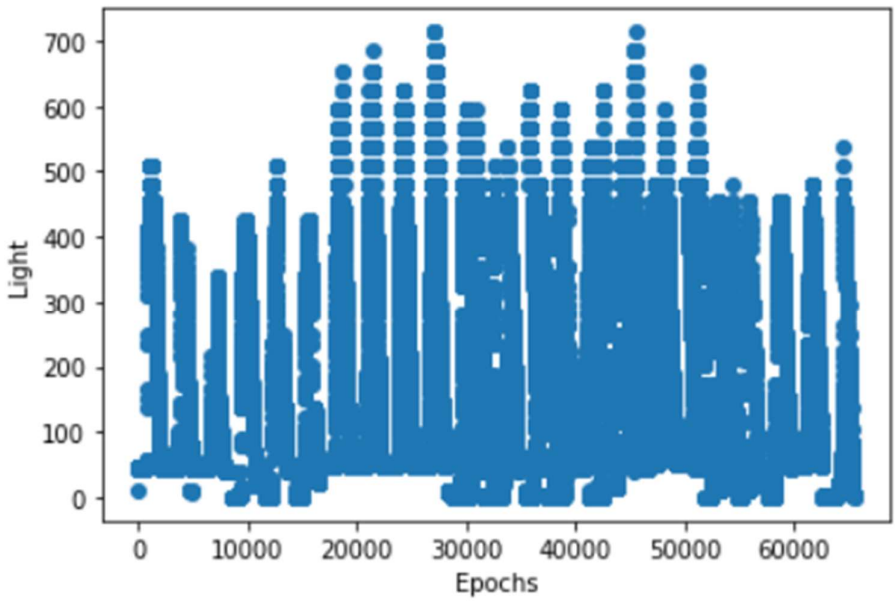
```
#Plots of data

plt.scatter(sensor_1_epoch, sensor_1_temp)
plt.xlabel('Epochs')
plt.ylabel('Temperature')
plt.show()
```



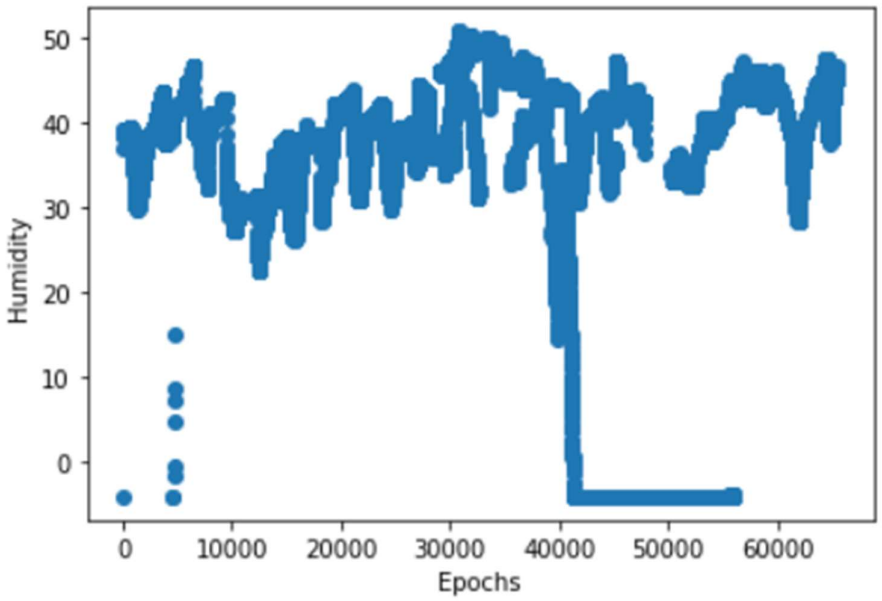
Epoch vs temperature

```
plt.scatter(sensor_1_epoch, sensor_1_light)
plt.xlabel('Epochs')
plt.ylabel('Light')
plt.show()
```



Light vs Epochs

```
plt.scatter(sensor_1_epoch, sensor_1_humidity)
plt.xlabel('Epochs')
plt.ylabel('Humidity')
plt.show()
```



Humidity vs Epochs

Obtained missing values after prediction:

[26]:

	missing_epochs	predicted_temperature	predicted_light	predicted_humidity
0	1.0	16.528929	151.928872	39.258476
1	4.0	16.530743	151.929308	39.258013
2	5.0	16.531348	151.929453	39.257859
3	6.0	16.531953	151.929599	39.257704
4	7.0	16.532558	151.929744	39.257550

Predicted values

Question 5: Elucidate briefly different techniques (min 4) for predicting missing data.

Methods for predicting missing data:

1. Mean imputation: We calculate the mean of the observed data and then replace the missing values with the mean. This method is very simple and straightforward and requires a very low amount of computation. However, it has a lot of disadvantages and is worse than almost all other prediction methods.

2. Replace with most common category: When the missing values are from categorical columns, then the missing values can be replaced with the most common category. This method is also very easy to implement but comes with a lot of issues.

3. Regression imputation: The predicted value is obtained by using a regression model such as linear regression or polynomial regression. We look at observed variables, derive a relation between them and then try to predict the missing variables. This preserves the relationship among the variables involved in the imputation model.

4. Hot deck imputation: Look at other sample point in the sample space who similar values on other variables and then randomly choose one of their values for the missing value. This limits our

choices to only similar sample points in the sample space and can help provide more accurate results.

References:

1. https://www.theanalysisfactor.com/seven-ways-to-make-up-data-common-methods-to-imputing-missing-data/

2. https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e

3. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html