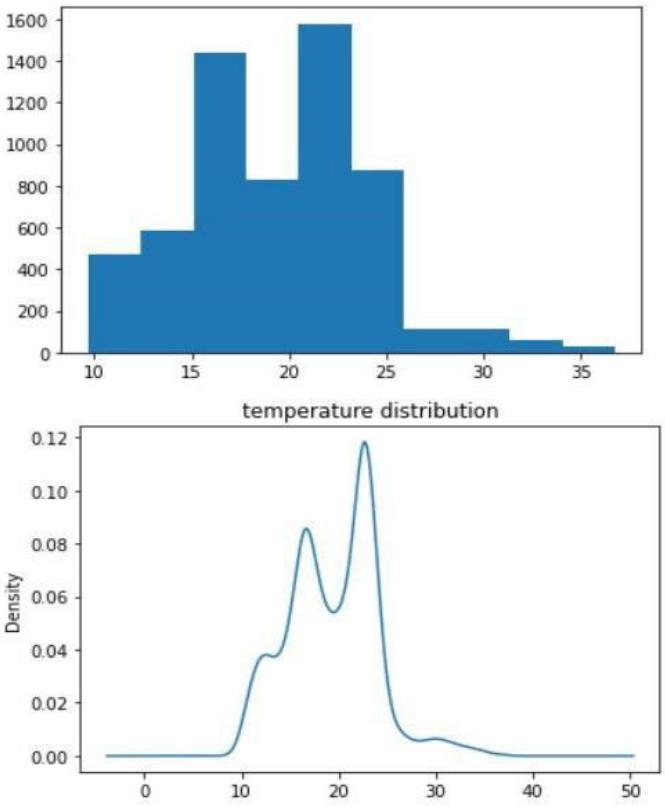
Predicting missing data in sensor variables

The data set has temperature and humidity values of 5 sensors out of which some values are missing and the task is to predict the missing values.

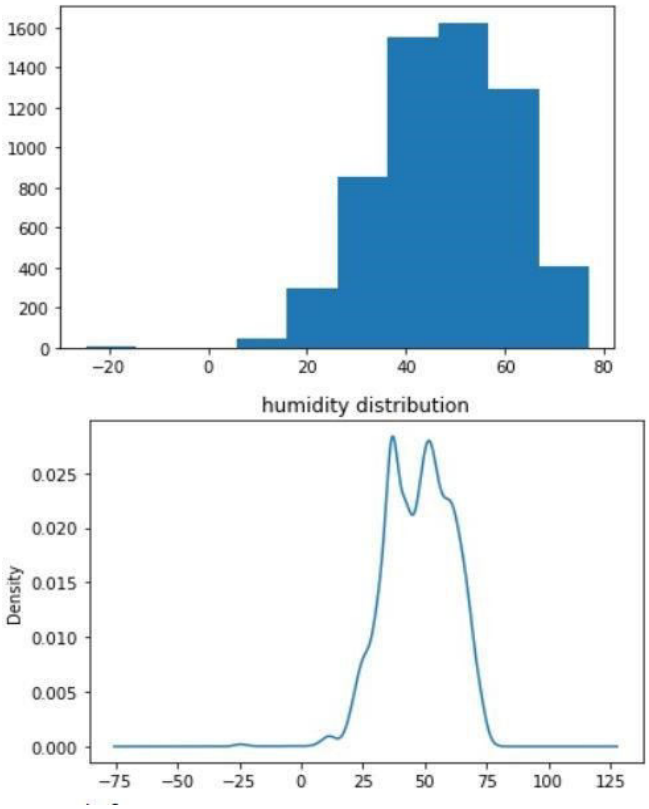
I have created two models – one predicting temperature taking humidity as the input and the other predicting humidity taking temperature as the input. Both the models have common set of input features, namely, latitude, longitude and timestamp.

Data pre-processing:

Data from all the nodes were loaded onto a single pandas dataframe. The timestamp mentioned in the data files were converted to integer datatype in the input feature.
The missing data was separated into another dataframe. Further they were bifurcated according to whether temperature or humidity was missing.
Temperature and humidity plots were done to identify and eliminate any outliers while taking temperature and humidity as the input feature respectively.



Histogram plots and distribution of temperature



Histogram plots and distribution of humidity

The full dataset (containing both temperature and humidity) was then converted to normalized format (wherever required) and divided into two datasets – 80% for training the model and 20% for testing it.
#no normalization for linear regression

Model Selection:

We will first start with the most obvious model i.e., Bayesian Linear Regression.

i. *Bayesian Linear regression:* $y = Xw + \in$

*The data was not normalized for this case.

Initially, the normal equation with L2 regularization was applied to obtain an initial guess for the parameter, w, as $w = (X^T X + \lambda I)^{-1} X^T Y$, where λ is a hyperparameter. Mean and variance of this vector was then calculated directly.

Prior was taken as Gaussian with mean and variance calculated as above. Likelihood was also taken as Gaussian with mean as Xw and variance was taken as $\sigma^2 I$, where σ^2 is also a hyperparameter. This inherently gives us a Gaussian posterior whose mean and variance can directly be calculated (as given in Slide 20, CP218_5.pdf).

Here, we have two hyperparameters to tune: σ^2 and λ.

I have one major assumption here – the initial guess for parameter, w, gives the mean and variance of the Gaussian distribution of the parameter itself. Hence, this model will cause some inherent errors in prediction.

Since I already are inclined and biased towards using Neural Networks, I wanted to explore NN. Here I came across Bayesian Neural Network, which is probabilistic approach with NN.

ii. Bayesian Neural Network:

*No significant data pre-processing was done here. Only the timestamp was converted to integer. The entire data was then normalized. Outliers, however, weren't removed in this case.

In BNN, the weights and biases are also taken as a distribution instead of point estimates as in traditional NN and the posterior is approximated using variational inference. Here I have taken prior distribution of weights and biases as Normal distribution with mean as 0 and variance as 1.

Number of hidden layers is kept as 4, in case of predicting humidity, and 2, in case of predicting temperature. Increasing number of hidden layers, in case of predicting temperature, results in gradient explosion, driving some predictions to NaN. The number of neurons in the hidden layers is a hyperparameter which is taken as 8 in the model.

The approximation to the actual Bayesian posterior is done by variational inference i.e., by minimizing the KL divergence between approximated and original posterior and optimized using Stochastic Gradient Descent (SGD) method. Using Adam Optimization Algorithm, the BNN is encountering the problem of exploding gradients.

Increasing the hidden layers to 5 (for humidity prediction) overfits the model. Also increasing to 3 hidden layers (for temperature prediction) results in overfitting the model. Also, the Kaggle score increased tremendously even though the RMSE over test data decreased.

Hence, I used Gaussian Process Regression as I can assume all possible functions which can fit the dataset along with some noise (with precision as hyperparameter).

iii. Gaussian Process Regression: $y = f(X) + \in$

*Here I have kept the data pre-processing part is kept as same as BNN. The timestamp is converted to integer and the dataset is normalized. Outliers are not eliminated.

*I have eliminated latitude and longitude as the features and I have added index as one of the features.

In GPR, instead of fixing the function to be fitted over the given dataset and estimating the posterior over the parameters to define the fitted function, I take a prior over all possible (Gaussian) functions, f(x), that can fit over the given dataset to estimate the mean as the predicted value and the variance as the uncertainty

associated with the prediction. This is done by using a kernel to calculate the similarity between the training data and test data input whose output is to be predicted.

Here, in my code I have used the Matern kernel. This is a generalisation of the Gaussian kernel. Here I have kept the default values of the hyperparameters (as in the sklearn library):

- smoothness parameter, $\nu = 1.5$ (for once differentiable functions)
- length scale parameter, $\lambda = 1.0$ (by default, this can be modified during optimization step)

As ν increases, the smoothness of the kernel also increases. As a result, $\nu \to \infty$, it reduces to Radial Basis Function or Gaussian kernel, and for $\nu=0.5$, it becomes identical to absolute exponential kernel.

In the GPR, the mean of the prior is taken as zero (by default). I have also considered the noise, ϵ, to be distributed as $N(0, \alpha^{-1}I)$, where α is the precision parameter and hence a hyperparameter (taken as 0.1 here). Also, the n_restarts_optimizer is set as 10 (this specifies the number of times the optimizer restarts to find the maximum of the log-marginal likelihood by sampling log-uniform randomly from the allowed hyperparameter values, λ, of the kernel).

Here, I have arrived at a quite low RMSE, both for test data and Kaggle score.

*In the RMSE evaluation, I have done each Node separately and provided the average value since predicting over all values was taking significant amount of time.

Hyperparameter Tuning:

All hyperparameters were trained using either grid search or trial-and-error method.

*Only for GPR, I haven't used any hyperparameter tuning step as this is done during optimization itself.

Evaluation:

RMSE is considered as metric to evaluate model on test data.

Model Name	RMSE on Test data		Public Score on Kaggle
	Temperature as output	Humidity as output	
Bayesian Linear Regression	8.94	3.74	7.21
Bayesian Neural Network	1.69	3.80	16.94
Gaussian Process Regression	1.91	0.56	1.29