**SANS**

# Homework 4: Calibration of sensors in uncontrolled environments in Air Pollution Sensor Monitoring Networks.

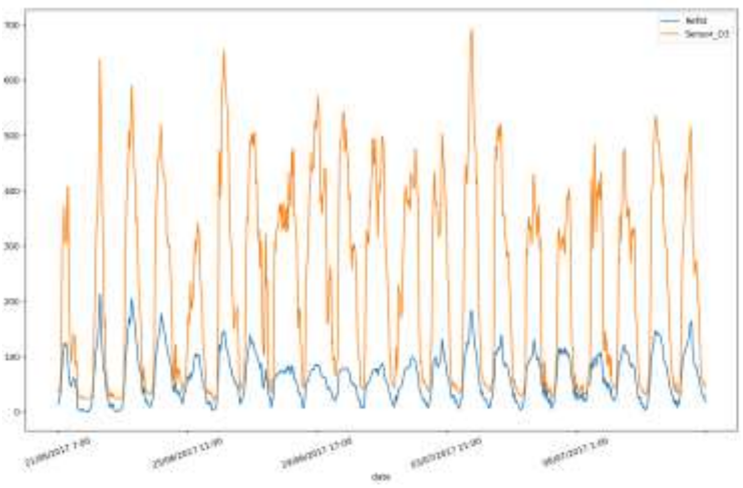
**Oriol Martínez & Imanol Rojas**

## Introduction

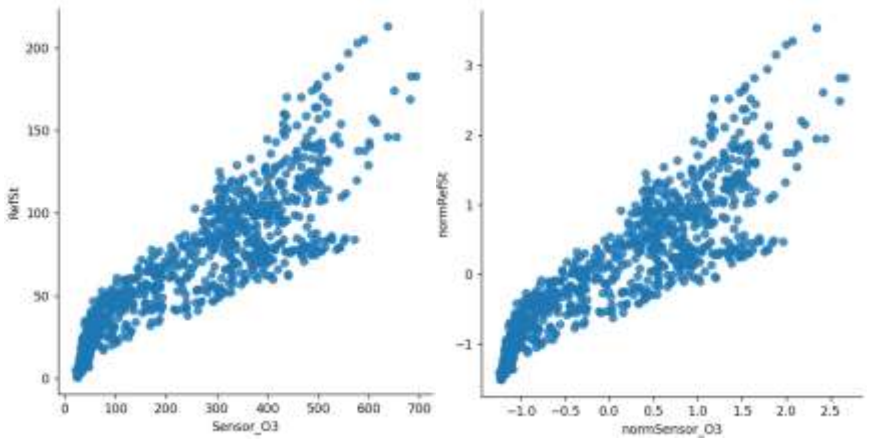
The objective of this project is to calibrate an air pollution sensor in an air pollution monitoring sensor network. We will take the data samples of one node that accommodates three sensors: a MIC2614 O3 sensor, a temperature sensor and a relative humidity sensor. The data set contains one thousand samples. This report shows the results of the main calibration.

## Project realization part 1: Observation of the data

1. If we plot the values of the ozone sensor and the reference station vs the date, we obtain the next plot. The plot shows that both have almost the same shape but with a difference in scale.



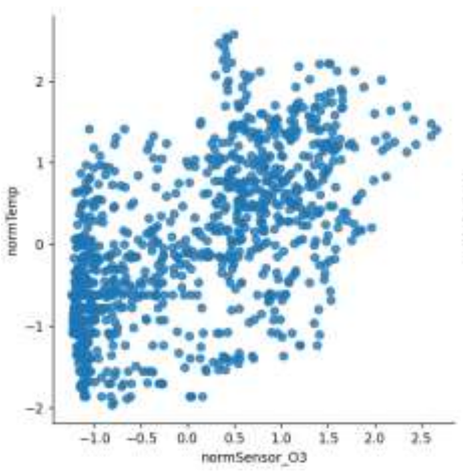
2. The next scatter plots show the relation between the values of the ozone sensor vs the values of the reference station. The one on the right is done with the raw values of the dataset, the one on the left is made with the normalized values of Sensor_O3 and RefSt. The difference in terms of the pattern is inexistent because normalizing is just changing the scale.

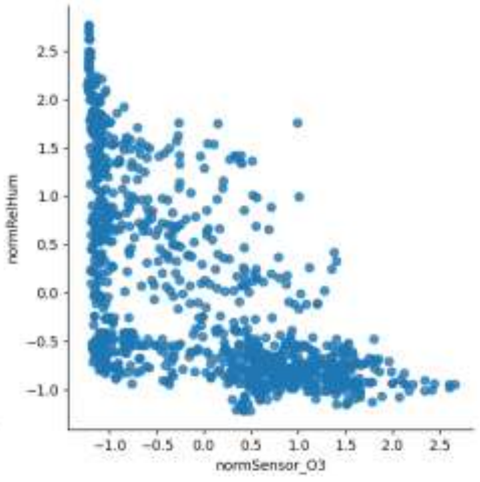


The patterns of both plots are the same and show linear dependence between the two sets of values.

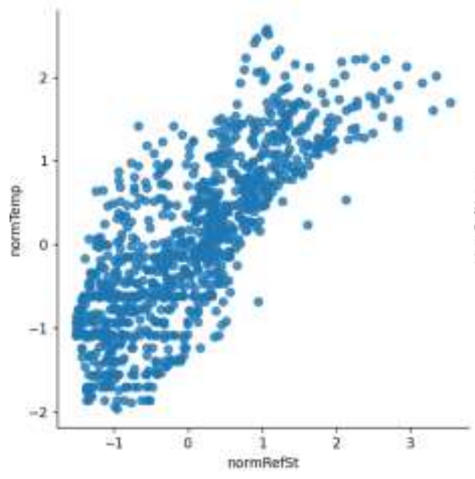
3. The next two scatter plots analyze the dependence of the Sensor_O3 values with the Temp and with the RelHum values. The same is done for the reference station. All the data used is normalized.

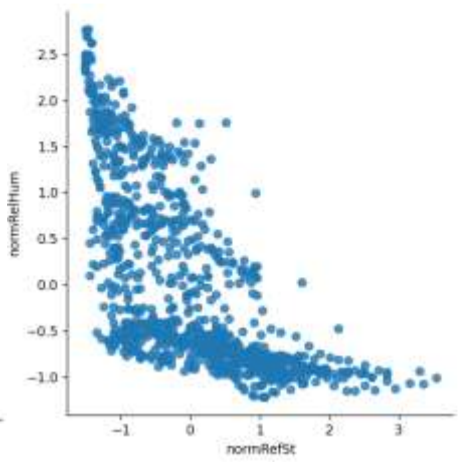
Graphs related to Sensor_O3 values vs Temp and RelHum values respectively.





Graphs related to the RefSt values vs Temp and RelHum values respectively.





When comparing the plots with respect to Sensor_O3 and with respect to RefSt, we see that the shape is similar but not the same due to the lack of calibration. Furthermore, both variables Temp and RelHum show on the plots that there is a correlation between them and the reference data RefSt, meaning that the combination of Temp, RelHum and Sensor_O3 is a good base for making predictions and obtain results like true measurements RefSt.

## Project realization part 2: calibration using multiple linear regression (frequentist framework).

Using the library sklearn we are going to search for the Ordinary Least Squares (multiple linear regression model in a frequentist framework) that fits the data. The linear regression model has the following structure:

$$\hat{y}_{\mathrm{RefSt}_j} = \theta_0 + \theta_1 x_{\mathrm{Sensor\_O3}} + \theta_1 x_{\mathrm{Temp}_j} + \theta_1 x_{\mathrm{RelHum}_j}$$

The model is calculated and then plotted in a scatter plot. This scatter plot is done by plotting the test set through the model (predictions) in the y axis vs the reference station raw data of the test set in the x axis. Then a regression line is added. Finally, the result is analyzed with metrics ( $R^2$ , RMSE, MAE).

In multiple linear regression (frequentist approach) we don't have any prior information or assumption/hypothesis. The model is found using Ordinary Least Squares and creating models searching for the one with least square error. In order to implement it in python first, we create two sets from the data that we have (shuffling the data first). We will end up having 30% of the data in a test set and 70% of the data in the training set.

```
# Create data with all the CSV
data = pd.read_csv('datos-17001.csv', sep = ';', index_col = 0,
parse_dates = False)

# Create the dataframe
df = pd.DataFrame({'RefSt': data["RefSt"], 'Sensor_O3': data["Sen-
sor_O3"], 'Temp': data["Temp"], 'RelHum': data["RelHum"]})

# Splitting the dataset into Y: RefSt and X: Sensor_O3, Temp and RelHum
X = df[['Sensor_O3', 'Temp', 'RelHum']]
Y = df['RefSt']

# Shuffling the data and assigning 30% to the test set and 70% to the
train set
X_tr, X_te, Y_tr, Y_te = train_test_split(X, Y, test_size = 0.3, ran-
dom_state = 1, shuffle = True)
```

Then the model is created, we use the data of the training set to find the coefficients that make RefSt fit with the model using the Ordinary Least Squares method.

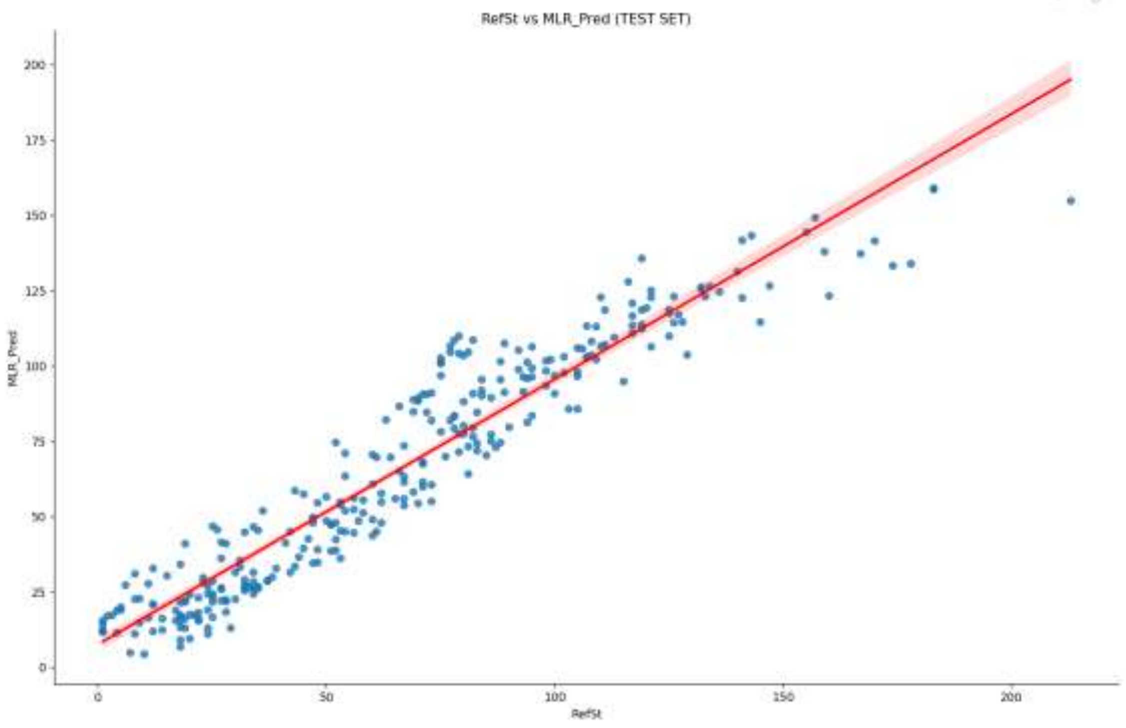
```
df_tr = pd.DataFrame({'RefSt': Y_tr, 'Sensor_O3': X_tr["Sensor_O3"],
'Temp': X_tr["Temp"], 'RelHum': X_tr["RelHum"]})
df_te = pd.DataFrame({'RefSt': Y_te, 'Sensor_O3': X_te["Sensor_O3"],
'Temp': X_te["Temp"], 'RelHum': X_te["RelHum"]})

lr = LinearRegression()
lr.fit(X_tr, Y_tr)
```

This will give us the coefficients that create the model with the lowest number of the sum squared errors in relation to the training data. The obtained coefficients are the following:

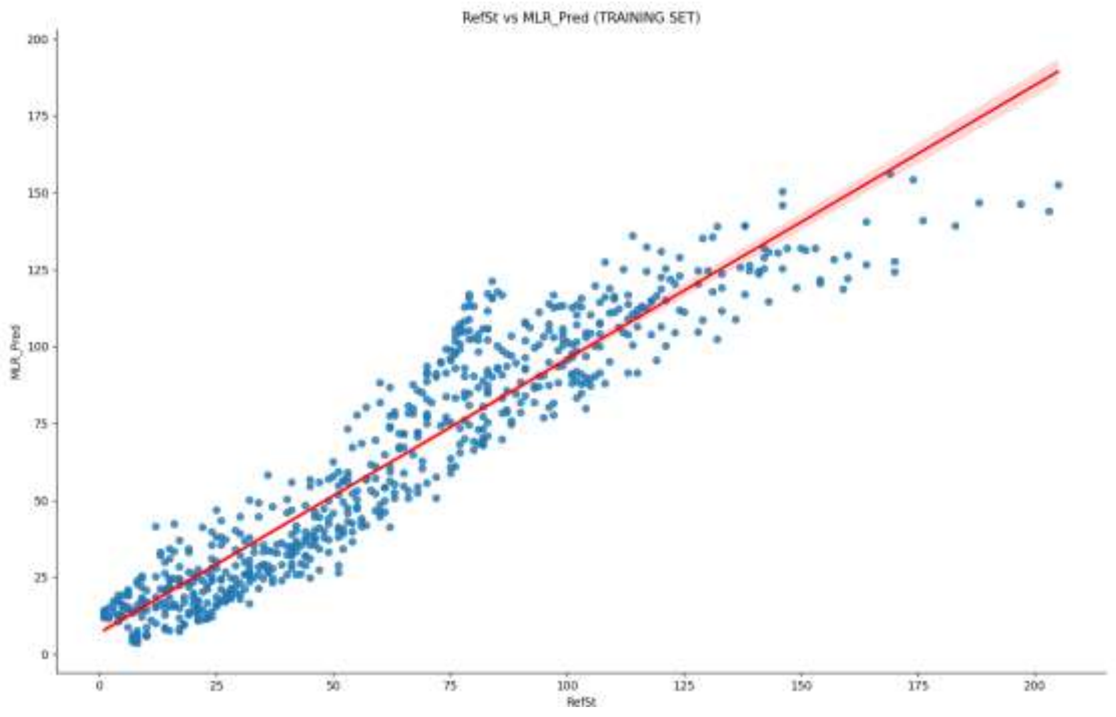
- Intercept = -30.170172077980972
- Coefficient 1 = 0.15704393891215682
- Coefficient 2 = 2.418522936269852
- Coefficient 3 = -0.06497555353321659

Now, the test set is used in order to make predictions. This is done by substituting the sensor values in the model (using the coefficients that we obtained) in order to obtain a reference station value (prediction). The result is plotted in a scatterplot vs the reference station values of the test set that we already had. The result is the following:



As we can see, the model fits the data quite good. The perfect model would have all the points in the regression line (the prediction would be equal to the test data).

If we do the same but using the training set for the values of the parameters and using the coefficients that we obtained, we get the following plot:



This plot shows the difference in terms of the number of data points between the test set and the training set. The graph of the training set is more crowded. Also, we can conclude that the model fits with the reference data quite good.

To know how well fitted the model is if we use the test set to predict we can use the following metrics. These metrics compare the prediction with the reference data that we have.

- $R^2$ = 0.9090705167941918
- RMSE = 12.75765069841883
- MAE = 9.763563481282393

Using the training set to predict, the metrics are the following:

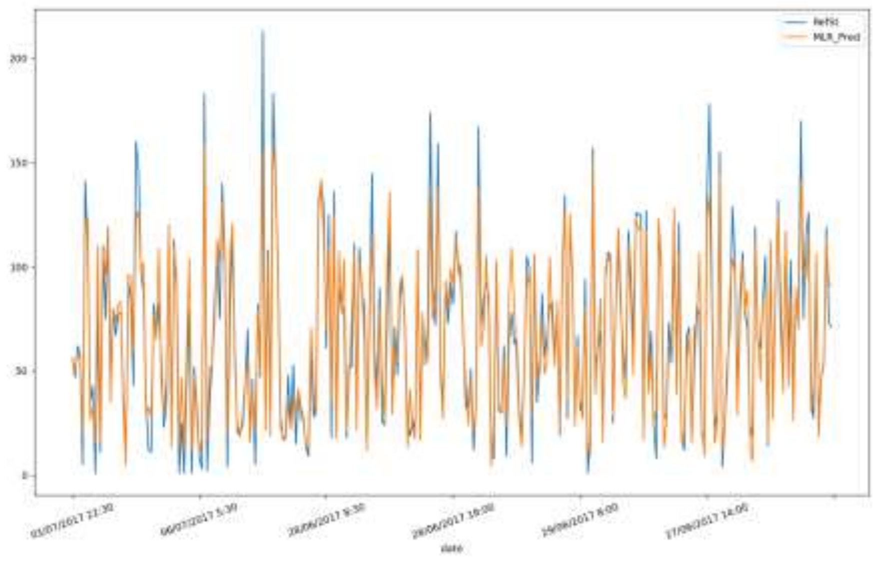
- $R^2$ = 0.8897170074178326
- RMSE = 13.859392289631563
- MAE = 10.875506152115802

The determination coefficient tells us the model quality for predicting the data and also the proportion of the variation of the results. In this case we se that the coefficient is around 90% in both cases. This means that the model fits the data. Ç

The RMSE is the mean of the squared root of sum of the errors (residuals) squared. This metric penalizes high errors due that the errors are squared. In this case the value of the error is small so we can predict data with low amounts of errors.

The MAE is the mean absolute error. measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between the prediction and the actual observation where all individual differences have the same weight. RMSE and MAE values must be close.

If we plot the predicted data using the training set and the reference station data from the training set with respect of time, we have the following plot:



As we can see, both lines follow the same path. The difference between the predicted data and the real data is minimal so we can say that the model predicts the data correctly.

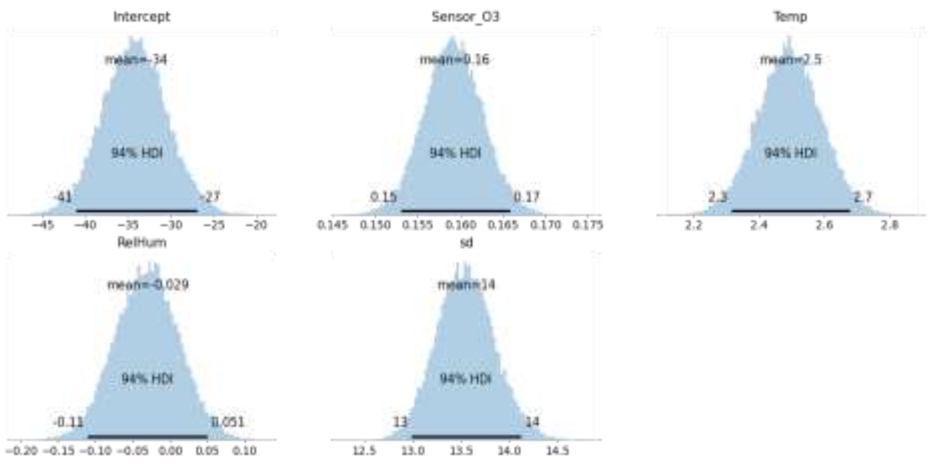
## Project realization part 3: calibration using multiple linear regression (Bayesian framework)

In this part we are going to repeat the calibration but with Bayesian approach. In the Bayesian framework we assume that the output follows a normal distribution of mean $\theta_0 + \theta_1 x_{\text{Sensor\_O3}} + \theta_1 x_{\text{Temp}_j} + \theta_1 x_{\text{RelHum}_j}$ and variance $\sigma_j^2$.

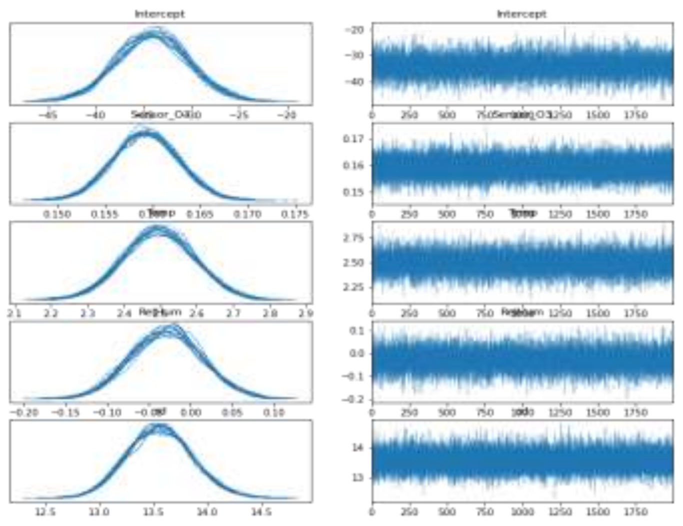
For doing this in python:

```
basic_model =  pm.Model()
with basic_model:
    pm.glm.GLM.from_formula("RefSt ~ Sensor_O3 + Temp + RelHum", data,
family=pm.families.Normal(), offset=1.0)
    start= pm.find_MAP()
    step=pm.NUTS(scaling=start)
    trace= pm.sample(draws=100, step=step, start=start, progressbar=True)
```

If we plot the histogram of the posterior parameters me obtain the following:



If we plot the simulation samples, we obtain the following:


Intercept
Sensor_O3
Temp
RelHum
sd