Supplement of Atmos. Meas. Tech., 14, 37–52, 2021
https://doi.org/10.5194/amt-14-37-2021-supplement

Atmospheric Measurement Techniques
Open Access
EGU

Supplement of

Robust statistical calibration and characterization of portable low-cost air quality monitoring sensors to quantify real-time O_3 and NO_2 concentrations in diverse environments

Ravi Sahu et al.

Correspondence to: Sachchida Nand Tripathi (snt@iitk.ac.in)

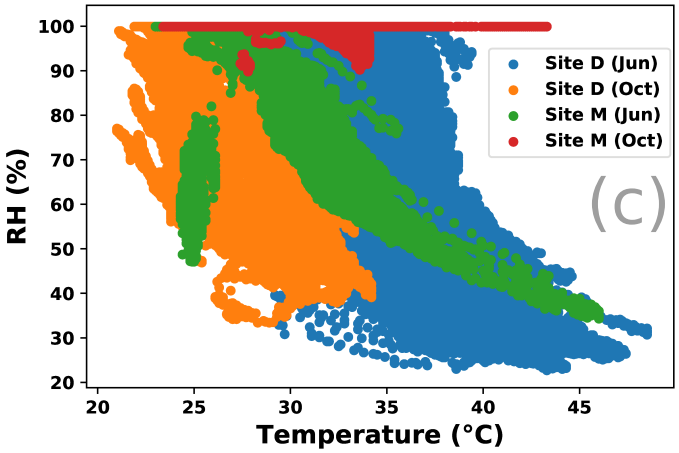

Figure S1. A scatter plot showing variations in RH and T at the two sites across the two deployments. The sites offer substantially diverse weather conditions. Site D exhibits wide variations in RH and T levels during both deployments. Site M exhibits almost uniformly high RH levels during the Oct deployment which coincided with the retreating monsoons.

S1 Details of the Deployment Sites

Here we offer some additional details about the two deployment sites.

About Site D: According to Greenpeace India (Times, 2018) and the Niti Aayog, Govt. of India (Aggarwal, 2018), Faridabad was the second most polluted city in India in 2018. Surrounded by the Aravalli Hills, this is a rapidly growing city and a leading industrial center suffering from heavy air pollution that mask the city and its neighborhoods routinely during the fall and winter seasons. The study site is 5 km away from Delhi and near Delhi-Surajkund Highway. It falls in the Indo-Gangetic Plain, which registered critical levels of ambient air pollution attributable to a combination of multiple ambient sources, the use of biomass and coal for household cooking and heating needs, and the stubble or agricultural residue burning (Chowdhury et al., 2019). The deployment site is affected by vehicular traffic which are likely a dominant source of precursors to O_3 formation (NO_2 and volatile organic compounds) and of nitric oxide that reacts with O_3 to form the pollutant NO_2. The reference monitors were deployed in a laboratory on the first floor of the building with the low-cost AQ monitoring sensors next to its inlets.

About Site M: This site presents relatively lower pollution levels as it is situated within the IIT Bombay campus between the Vihar and Powai lakes, and it is adjacent to the Sanjay Gandhi National park. Less that 5 km to its west side passes the Thane creek (an inlet in the shoreline of the Arabian Sea) that isolates the city of Mumbai from the Indian mainland while the Arabian sea is at around 10 km to its west. The reference monitors were deployed on the rooftop of the building with the low-cost AQ monitoring sensors next to its inlets. All AQ monitoring devices were in a Stevenson box to avoid damage to sensors.

S1.1 Analysis of Raw Data from Deployment Sites

We now take a closer look at raw data from the deployment sites to understand data characteristics better. Our deployment strategy consisted of two sites at geographically diverse locations and experiencing varying air pollution levels, over two extended deployments during months experiencing significant variations in RH and T. This and the swap-out experiment were aimed at covering a wide range of real-world ambient working conditions (Cross et al., 2017). As we saw in the main paper, data from such diverse operating conditions is crucial for proper calibration of these sensors in order to not expect drastic extrapolations from the models during actual deployment.

To illustrate this, refer to Fig. S1 which shows the RH and T ranges observed during the two deployments across the two sites. It is clear that both sites offer extremely diverse meteorological conditions, with only site M offering somewhat uniformly high RH values during the Oct deployment. However, what is also interesting is that the sites also offer RH and T ranges that

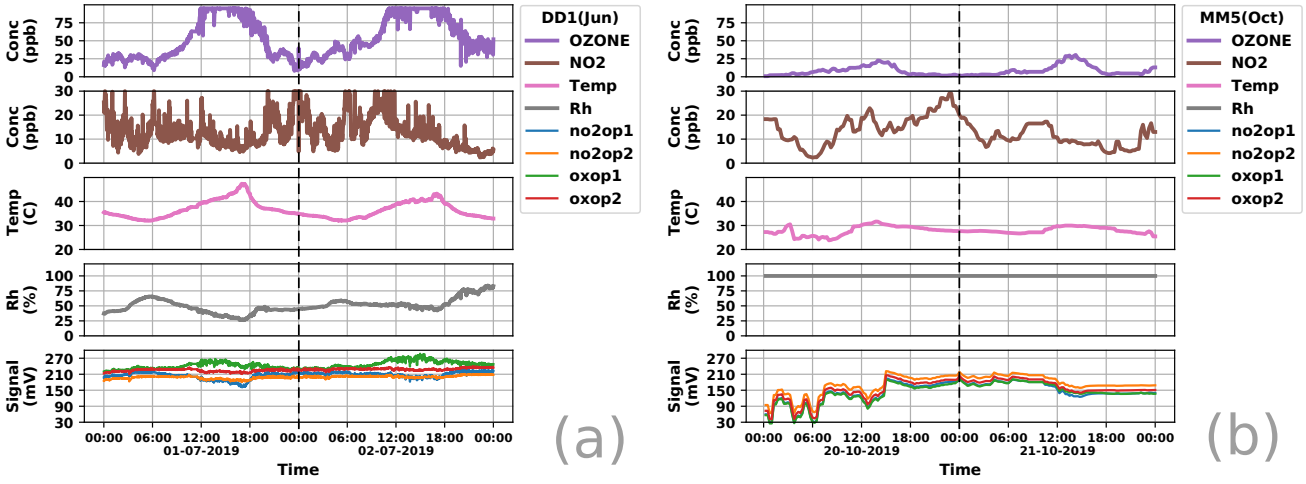

Figure S2. Time series showing the variation in the raw parameters measured using the reference monitors (NO_2 and O_3 concentrations) as well as those measured using the SATVAM LCAQ sensors (RH, T, no2op1, no2op2, oxop1, oxop2). Fig. S2(a) considers a 48 hour period during the Jun deployment (01-02 July 2019) at site D with signal measurements taken from the sensor DD1 whereas Fig. S2(b) considers a 48 hour period during the Oct deployment (20-21 October 2019) at site M with signal measurements taken from the sensor MM5. Values for site D are available at 1 minute intervals while those for site M are averaged over 15-min intervals which explains why the left plot is more granular than those in the right plot. Also notice that site D experiences higher levels of both NO_2 and O_3 as compared to site M.

are often non-overlapping. For instance, Fig. S1 shows that the ranges for site D (Oct) and site M(Jun) experience little overlap, indicating the extreme diversity in this data.

We also present in Fig. S2, time series over 48 hour periods from two deployments at the two sites. The reference data for the site D Jun deployment indicates that O_3 levels exhibit a diurnal trend with a midday peak mainly at around 1500 hrs, while NO_2 levels tend to peak usually in the morning and in the evening to midnight, suggesting nearby roadways could be a predominant source of pollution. Site M on the other hand presents far lower O_3 levels. Ambient RH and T values were observed to vary inversely to each other at site D in both deployments and site M during the Jun deployment. However, site M experienced a near continuous 100% RH level during the Oct deployment. The sensor voltages (no2op1, no2op2, oxop1, oxop2) can be seen to have good correlation in the plots.

Fig. S2 shows that the two sites and deployments exhibit significant diversity with respect to absolute concentrations. The reference NO_2 levels from site M (available at 15 minute intervals) ranged from 0.01-44.13 ppb in the Jun deployment and from 0.01-58.44 ppb in the Oct deployment, respectively. At the same time, the reference NO_2 levels from site D ranged from 0.70-65.49 ppb and from 0.86-159.55 ppb during the Jun and Oct deployments, respectively. Similarly, reference O_3 levels also differ significantly across the sites with site M levels ranging from 0.70-65.49 ppb and 0.86-160.41 ppb during the Jun and Oct deployments respectively and those for site D ranging from 0.70-141.47 ppb and from 0.80-180.00 ppb for the same deployments. In general, Site D experienced higher concentration levels, as well as peaks, than site M. Furthermore, concentration levels were found to go up for both sites during the Oct deployment as compared to the Jun deployment. Such diversity in concentration levels are expected to empower calibration models to offer accurate predictions across wide ranges of operating conditions.

As deployments experienced several cloudy days, peaks of observed O_3 levels are not consistent throughout the deployments. Such influence of meteorological parameters on pollutant levels is well recognized in past literature (Gaur et al., 2014; Tiwari et al., 2015; Simmhan et al., 2019) with effects such as scavenging of PM and gaseous pollutants that occur due to rain that may result in lower concentration peaks of $PM_{2.5}$ levels (not considered in this study) and lower mixing ratio of NO_2, or higher range of concentrations of same pollutants during winter, being observed.

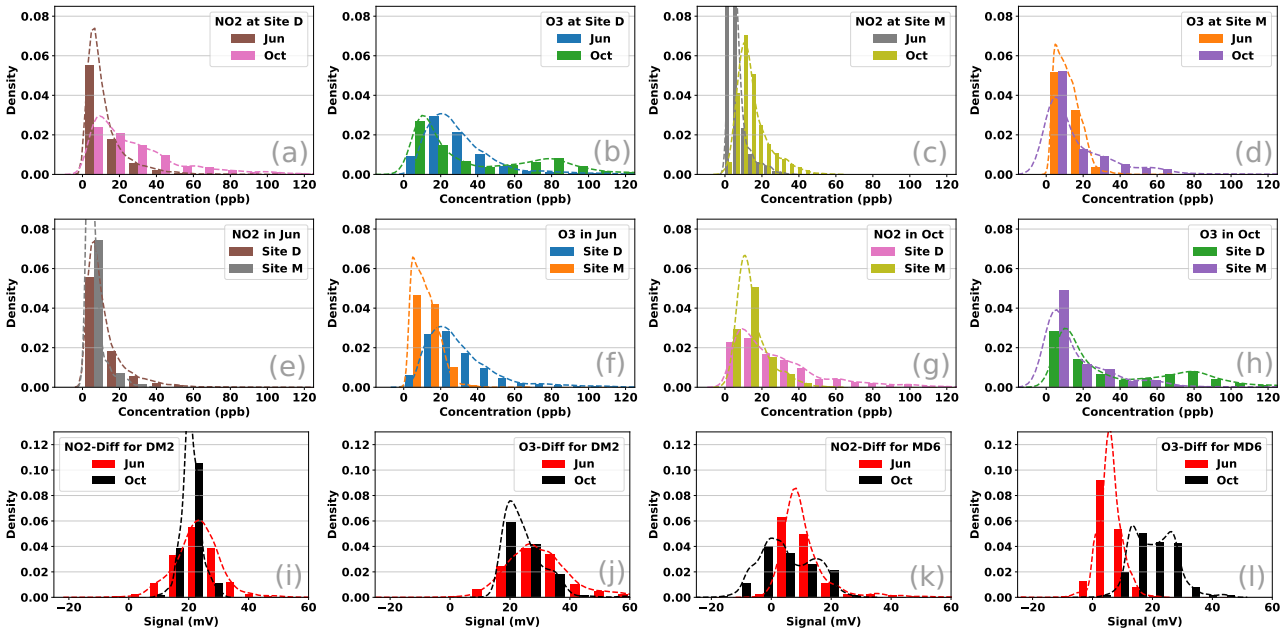

Figure S3. Normalized frequency histograms and distributions for various data series. All figures show histograms to indicate the frequency of occurrence of various values (as indicated by the bars) as well as fit a kernel density estimator to show a smoothed distribution of those values (as indicated by the dotted lines). Figures in the first row i.e. Fig. S3(a,b,c,d) demonstrate how the air quality levels (reference values of NO_2 and O_3) varied at the same site across the two seasons. Figures in the second row i.e. Fig. S3(e,f,g,h) demonstrate how the air quality levels varied in the same season across the two sites. Figures in the third row i.e. Fig. S3(i,j,k,l) explore cross site variations in the sensor voltage values (no2diff and oxdiff) recorded by the sensors DM2 and MD6 in the two seasons. Recall that both sensors participated in the swap-out experiment.

In order to better understand global trends in cross-site and cross-deployment variations, Fig. S3 plots histograms indicating the statistical distribution of reference values as well as sensor voltage readings for various sites and deployments. Fig. S3 (a,b,c,d) tell us that for both sites, the Oct deployment offers larger concentration levels as compared to the Jun deployment. On the other hand, Fig. S3 (e,f,g,h) tell us that site D experiences appreciably greater levels for both NO_2 and O_3. This is understandable since site M is located in a coastal city whereas site D is situated at a more arid location.

Fig. S3 (i,j,k,l) show us that the distribution of the voltage differentials (no2diff, oxdiff) can differ significantly when the same sensor is relocated to a different site during a different season. It is notable that the sensor readings exhibit concentration and are often well approximated by a Gaussian distribution. However, reference values do not exhibit these trends and tend to exhibit heavier tails and multiple modes.

S2 Revisiting Notation and Error Metrics

We revisit the notation and then proceed to give details of the error metrics used in our experiments.

Notation. For every time-stamp t, the vector $\boldsymbol{x}^t \in \mathbb{R}^8$ denotes the 8-dimensional vector of signals recorded by the LCAQ sensors for that time-stamp, namely (RH, T, no2op1, no2op2, oxop1, oxop2, no2diff, oxdiff), while the vector $\boldsymbol{y}^t \in \mathbb{R}^2$ will denote the 2-tuple of the reference values of O_3 and NO_2 for that time step. However, this notation is unnecessarily cumbersome since we will build separate calibration models for O_3 and NO_2. Thus, to simplify the notation, we will instead use $y^t \in \mathbb{R}$ to denote the reference value of the gas being considered (either O_3 or NO_2). The goal of calibration will then be to learn a real valued function $f : \mathbb{R}^8 \to \mathbb{R}$ such that $f(\boldsymbol{x}^t) \approx y^t$ for all time-stamps t (the exact error being measured using metrics such as

MAE, MAPE, etc described in Sect S2). Thus, we will learn two functions, say f_{NO_2} and f_{O_3} to calibrate for NO_2 and O_3 concentrations respectively. Since several of our calibration algorithms will involve the use of some statistical estimation or machine learning algorithm, we will let N (resp. n) denote the number of training (resp. testing) points for a given dataset and split thereof. Thus, we will let $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^N$ denote the training set for that dataset and split with $\boldsymbol{x}^t \in \mathbb{R}^8$ and $y^t \in \mathbb{R}$.

Error Metrics: calibration performance was measured using four popular metrics, mean averaged error (MAE), mean absolute percentage error (MAPE), root mean squared error (RMSE), and the coefficient of determination (R^2) (see below). Here n denotes the number of test points for a given dataset and split thereof, the variable t runs over all time-stamps in the testing set, y^t denotes the reference value (either O_3 or NO_2) at the t-th time-stamp, $\hat{y}^t$ denotes the corresponding value predicted by the calibration model, and $\bar{y}$ denotes the mean reference value i.e. $\bar{y} = \frac{1}{n}\sum_{t=1}^n y^t$.

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} |y^t - \hat{y}^t|$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n} \frac{|y^t - \hat{y}^t|}{y^t} \times 100\%$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} (y^t - \hat{y}^t)^2}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{t=1}^n (y^t - \hat{y}^t)^2}{\sum_{t=1}^n (y^t - \bar{y})^2}$$

S3 Description of Baseline Calibration Algorithms

Below we describe the baseline parametric and non-parametric calibration algorithms used in our study.

S3.1 Parametric Calibration Models

We first consider parametric calibration models that use an affine function to perform calibration i.e. f is of the form $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ where $\boldsymbol{w} \in \mathbb{R}^8$ is the *model vector* and $b \in \mathbb{R}$ is the *bias* term. The model and bias are usually inferred from data. We describe several such calibration models below.

S3.1.1 Alphasense Models

The manufacturers of the Ox-B421 and NO_2-B42F sensors used in the SATVAM devices themselves provides four different calibration algorithms which are described below. The four models are meant to reflect different operating conditions for these sensors, based on the ambient temperature range. In the following, $\alpha, \beta, \gamma, \alpha', \beta'$ are constants unique to individual units i.e. two sensors placed in two different SATVAM devices would have different values for these constants. These constants are provided by the manufacturer, based on their factory calibration.

$$p_1(a,b) = ((a-\alpha) + 0.6 \cdot (b-\beta))/\gamma$$
$$p_2(a,b) = ((a-\alpha) + \alpha'/\beta' \cdot (b-\beta))/\gamma$$
$$p_3(a,b) = ((a-\alpha) + (b-\beta) - (\alpha'-\beta'))/\gamma$$
$$p_4(a,b) = (a-\alpha)/\gamma$$

The AS1, AS2, AS3 and AS4 models: The above present the four different calibration models and can be directly used to calibrate for NO_2 concentrations as follows. For any $k \in 1 \ldots 4$, the k-th calibration model for NO_2 is proposed to be

$$f^k_{\mathrm{NO}_2} = p_k(\text{no2op1}, \text{no2op2})$$

Note that the model disregards RH and T information and uses only the no2op1, no2op2 voltage values to perform calibration for NO_2. Now, it turns out that the O_3 sensor is sensitive to the sum of NO_2 and O_3 concentrations. To account for this, the k-th calibration model for O_3 is proposed to be

$$\begin{aligned} f^k_{O_3} &= p_k(\text{oxop1, oxop2}) - f^k_{NO_2}(\text{no2op1, no2op2}) \\ &= p_k(\text{oxop1, oxop2}) - p_k(\text{no2op1, no2op2}) \end{aligned}$$

In our experiments, none of the four Alphasense calibration models offered satisfactory performance. We note that similar equations were recently used by (Chatzidiakou et al., 2019) for calibration of NO_x, O_3 and CO concentrations although the constants in their models are learnt from data from actual deployment rather than factory deployment.

S3.1.2 Linear Models

We implemented linear regression techniques to fully explore the calibration power of predictive power of affine functions. We recall that Alphasense calibration models are essentially affine functions of the feature vector $\boldsymbol{x} \in \mathbb{R}^8$.

Least Squares (LS): The standard least squares formulation seeks to learn a model vector and bias value that minimizes the (squared) RMSE error by solving the following optimization problem over training data:

$$(\boldsymbol{w}^{\text{LS}}, b^{\text{LS}}) = \arg\min_{\substack{\boldsymbol{w}\in\mathbb{R}^8 \\ b\in R}} \sum_{t=1}^{N} ((\boldsymbol{w}^\top \boldsymbol{x}^t + b) - y^t)^2$$

Least Squares on reduced features (LS(MIN)): To assess the effect of the RH and T features as well as the augmented features oxdiff and no2diff, we performed least squares regression on a reduced feature set which did not contain the augmented features and RH and T features.

Sparse linear regression (LASSO): The LASSO formulation (Tibshirani, 1996) seeks to learn a sparse model i.e. a model vector $\boldsymbol{w}$ such that one or more coordinates of $\boldsymbol{w}$ are zero. LASSO can be effective at *feature selection* i.e. indicating which of the 8 input features are most relevant for a particular calibration task. To do so, LASSO solves the following regularized optimization problem over training data:

$$(\boldsymbol{w}^{\text{LASSO}}, b^{\text{LASSO}}) = \arg\min_{\substack{\boldsymbol{w}\in\mathbb{R}^8 \\ b\in R}} \lambda \cdot \|\boldsymbol{w}\|_1 + \sum_{t=1}^{N} ((\boldsymbol{w}^\top \boldsymbol{x}^t + b) - y^t)^2,$$

where $\|\boldsymbol{w}\|_1 = \sum_{j=1}^{8} |\boldsymbol{w}_j|$ and $|\cdot|$ denotes the absolute value operator. The *regularization parameter* λ was tuned over a fine grid spanning five orders of magnitude $[0.0001, 0.0002, \ldots, 10, 20, 50]$ using held-out validation.

S3.2 Nonparametric Calibration Models

We also consider several baseline calibration models based on non-parametric regression techniques, including standard models such as regression trees, and several variants of kernel regression.

S3.2.1 Regression Trees (RT)

Regression trees are a form of *space partitioning* data structure that recursively subdivide the feature space (in our case $\mathbb{R}^8$) into small regions. Initially, all training data points reside at the *root* node of the tree which represents the entire feature space $\mathbb{R}^8$. Then, a simple rule based on a single feature, for instance whether no2diff = no2op2 - no2op1 ≤ 17.5 or not (see Fig. S4 (top) for this example) is used to *split* this node into two *child* nodes. Thus, training data points at this node that satisfy this

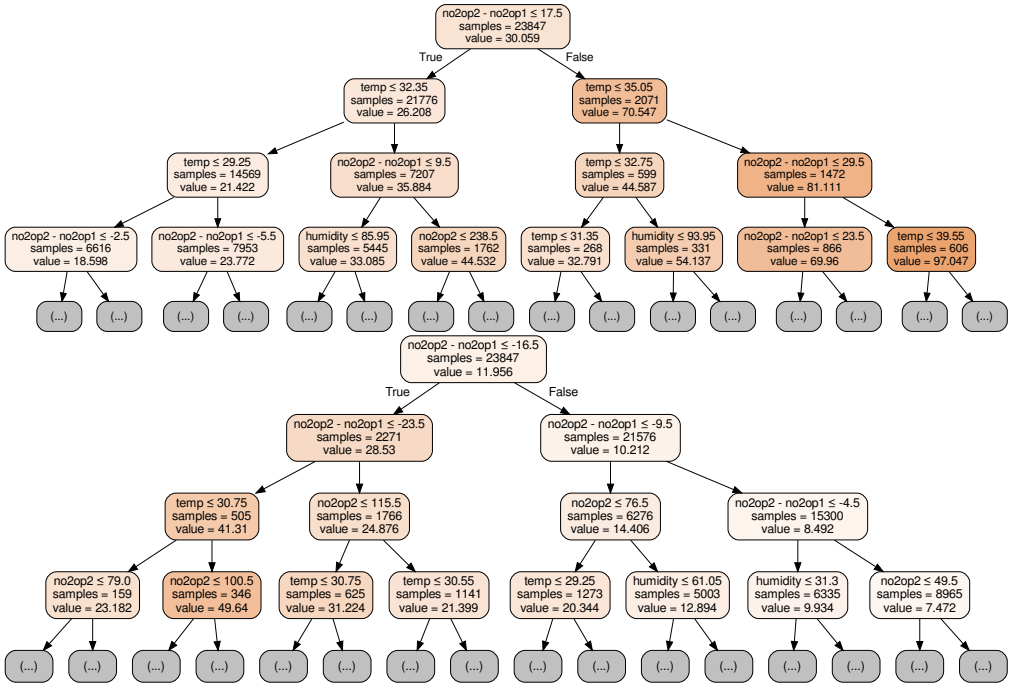

Figure S4. The first three levels of regression trees learnt for O_3 (top) and NO_2 (bottom) calibration for the DD1(Jun) dataset. Each internal node in the tree describes what rule was used to split that node, the number of training data points that reached that node, and the average value of the true (i.e. reference) concentrations of the gas (O_3 or NO_2) within the training data points that reached that node. The shade of color of each node is indicative of the magnitude of the average reference value of training data points at that node. Notice the diversity of the splitting rules used at various nodes in terms of using RH, T as well as electrode potential values (e.g. no2op1, oxop2 etc) to perform the splits.

rule go to one child and data points that do not satisfy this rule go to the other child. Once the nodes are small enough, i.e. they contain fewer training data points than a set threshold, the average reference value of the training data points at that node is used to perform prediction on all (testing) data points that reach that node.

The splitting rules at various nodes are learnt using an exhaustive search to ensure that the child nodes getting created as a result of that rule are as *pure* as possible. In our case, the purity of a node was measured in terms of variance. Specifically, let $\{i_1, \ldots, t_s\} \subseteq [N]$ be the s training data points at a certain node. Then the purity of this node is measured as $\frac{1}{s}\sum_{k=1}^{s}(y^{t_k} - \bar{y})^2$ where $\bar{y} = \frac{1}{s}\sum_{k=1}^{s} y^{t_k}$ is the average reference value of data points at that node.

A standard implementation of a regression tree was used with nodes being asked to be split into two children till the number of training data points at a node fell below a threshold *min_data points*. This threshold was tuned across a fine grid of $[2, 4, 6, 8, 10, 15, 20]$ using held-out validation. Fig. S4 gives examples of actual regression trees learnt on the DD1(Jun) dataset for NO_2 and O_3 calibration.

S3.2.2 Classical $k-$NN Regression Variants

The k-nearest neighbor algorithm is a *local* proximity-based learning algorithm that makes predictions on test data points based on which are the training data points that most resemble the test data point. Resemblance is usually calculated using a metric such as the Euclidean metric. We implement several k-nearest neighbor variants including our proposed variants that use metric learning which we will describe later. Algorithm S1 gives a unified pseudo code that describes all these variants.

Algorithm S1 Variants of KNN based calibration

Require: training data points $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^N$, neighborhood size k, weighing rule, choice of metric
Ensure: based on weighing rule and choice of metric, a prediction from the KNN, KNN-D, KNN(ML) or KNN-D(ML) models
`{KNN, KNN-D use Euclidean metric. KNN(ML), KNN-D(ML) use a learnt metric.}`
if metric == Euclidean **then**
$\boldsymbol{\Sigma} \leftarrow I_8$ `{The 8 × 8 Identity matrix}`
else if metric == learnt **then**
$\boldsymbol{\Sigma} \leftarrow$ use training data points to learn a Mahalanobis metric using the technique from (Weinberger and Tesauro, 2007)
end if
Receive feature vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^8$ for a test data point
Find the k training data points (say $\boldsymbol{x}^{i_1}, \ldots, \boldsymbol{x}^{i_k}$) that are closest to $\tilde{\boldsymbol{x}}$ in terms of the learnt Mahalanobis distance $d^{\text{Maha}}(\cdot, \cdot; \boldsymbol{\Sigma})$
`{KNN, KNN(ML) use uniform weighing. KNN-D, KNN-D(ML) use distance weighing.}`
if weighing rule == uniform **then**
$$\hat{y} = \frac{1}{k} \sum_{l=1}^{k} y^{t_l}$$
else if weighing rule == distance weighted **then**
For all $l = 1 \ldots k$, let $\alpha^l = (d^{\text{Maha}}(\tilde{\boldsymbol{x}}, \boldsymbol{x}^{i_l}; \boldsymbol{\Sigma}))^{-1}$
$$\hat{y} = \frac{\sum_{l=1}^{k} \alpha^l \cdot y^{t_l}}{\sum_{l=1}^{k} \alpha^l}$$
end if
return Calibrated value $\hat{y}$ for the test data point

KNN with Euclidean Distance (KNN): The vanilla k-nearest algorithm (KNN) predicts on a test data point, the average reference value in the k nearest training data points. The Euclidean distance was used to compute neighbors and the neighborhood size k was tuned over the fine grid $[2, 4, 6, 8, 10, 15, 20]$ using held-out validation. Standard implementation of kd-trees were used to accelerate the process of discovering the nearest neighbors for a test data point.

Distance weighted KNN (KNN-D): We also implemented a *distance-weighted* version of this algorithm wherein closest neighbors for particular test data point are weighted according to their Euclidean distance to the test point with closer points getting more weightage. We found this to favorably improve calibration performance.

S3.2.3 Classical Kernel Regression Variants

In statistics and machine learning, the notion of a *kernel* refers to a function that assigns a similarity value to two vectors (Murphy, 2012). Thus, in our setting a kernel would be a function of the form $K : \mathbb{R}^8 \times \mathbb{R}^8 \rightarrow \mathbb{R}$ which, when given two vectors $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathbb{R}^8$, assigns a value $K(\boldsymbol{x}^1, \boldsymbol{x}^2) \in \mathbb{R}$ denoting how similar are these vectors. A popularly used kernel is the *Gaussian* kernel (aka the *RBF* kernel) that calculates this similarity as $K(\boldsymbol{x}^1, \boldsymbol{x}^2) = \exp(-\gamma \cdot \|\boldsymbol{x}^1 - \boldsymbol{x}^2\|_2^2)$ where $\|\cdot\|_2$ denotes the Euclidean norm and γ is a *bandwidth* parameter that controls the scale at which similarity values go down. Note that even kernels, and by extension kernel regression algorithms, make use of notions of distance, e.g. the Euclidean distance to perform computations. This observation will come in handy later. The Nadaraya-Watson estimator and kernel ridge regression are two popular forms of kernel regression algorithms. A closely related cousin is Gaussian-process regression. Below we describe the Nadaraya-Watson estimator as it is useful in the developement of our proposed technique. Kernel ridge regression is described in the Supporting Information document.

Nadaraya-Watson (NW): Given a training set $\{(\boldsymbol{x}^t, y^t)\}_{t=1}^N$, the NW estimator (Nadaraya, 1964; Watson, 1964) makes a prediction on a new (testing) data point $\boldsymbol{x} \in \mathbb{R}^8$ as follows

$$f^{\text{NW}}(\boldsymbol{x}) = \frac{\sum_{t=1}^{N} y^t \cdot K(\boldsymbol{x}^t, \boldsymbol{x})}{\sum_{t=1}^{N} K(\boldsymbol{x}^t, \boldsymbol{x})}$$

The intent of this estimator is clear – the final prediction is a weighted sum of reference values y^t in the training set with the weight of a training data point $t \in [N]$ being proportional to $K(\boldsymbol{x}^t, \boldsymbol{x})$ i.e. how similar is that training data point to the test data point. Notice also the similarity between NW and KNN-D in the way they make predictions. NW almost behaves like a "smoothed" version of KNN-D by performing weighing using kernel values instead of inverse Euclidean distances and considering all training data points instead of just the neighbors. This observation will also be useful later. Apart from the Nadaraya-Watson method, we also consider kernel ridge regression and its accelerated version as baseline methods which are described below.

Kernel Ridge Regression (KRR): The KRR algorithm generalizes NW to learn a parameter value $\{\alpha^t\}_{t=1}^N, \alpha^t \in \mathbb{R}$ for every training data point. These values are intended to denote the relative importance of various training data points in offering an accurate prediction. Let $\boldsymbol{\alpha} = [\alpha^1, \ldots, \alpha^N] \in \mathbb{R}^N$ denote the vector of these values. Given a feature vector $\boldsymbol{x} \in \mathbb{R}^8$, KRR makes a prediction as follows:

$$f^{\text{KRR}}(\boldsymbol{x}; \boldsymbol{\alpha}) = \sum_{t=1}^{N} \alpha^t \cdot K(\boldsymbol{x}^t, \boldsymbol{x}),$$

These parameters learnt so as to minimize the (regularized) RMSE error on the training set as shown below. Let $G \in \mathbb{R}^{N \times N}$ denote the *Gram* matrix of kernel values among the training points i.e. $G_{ij} = K(\boldsymbol{x}^i, \boldsymbol{x}^j)$.

$$\boldsymbol{\alpha}^{\text{KRR}} = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \lambda \cdot \boldsymbol{\alpha}^\top G \boldsymbol{\alpha} + \sum_{t=1}^{N} (f^{\text{KRR}}(\boldsymbol{x}; \boldsymbol{\alpha}) - y^t)^2,$$

The regularization hyperparameter λ was tuned over a fine grid spanning 5 orders of magnitude $[0.0001, 0.0002, \ldots, 10, 20, 50]$ using held-out validation. The bandwidth parameter of the Gaussian kernel γ was tuned to the inverse of a certain percentile of the pairwise Euclidean distances between the training feature vectors. This percentile was tuned over the fine grid $[0.1, 0.2, \ldots, 0.9]$ using held-out validation. This range is popularly held as a reasonable range within which a near-optimal bandwidth value can be discovered (Caputo et al., 2002).

Nystroem Method (NYS): Despite its representational power, KRR is known to be slow at training and prediction. To speed up prediction times, which must happen in real time, we implemented the Nystroem method (Williams and Seeger, 2001) which is a scalable kernel approximation technique. We omit a detailed description of this method for sake of brevity.

S3.2.4 Metric Learning for Calibration (NW(ML), KNN(ML), KNN-D(ML))

As mentioned in the main paper, regression and calibration algorithms using KNN-style algorithms pose a challenge to metric learning. However, we make use to two earlier observations to overcome this problem. Specifically, we note that

1. Kernels such as the Gaussian kernel do internally use a Euclidean distance term to compute similarity.
2. There do exist techniques Weinberger and Tesauro (2007) to learn a good Mahalanobis metric to replace the Euclidean distance within the expression for the Gaussian kernel, when being used alongwith the NW algorithm
3. The NW algorithm resembles KNN-style algorithms and hence a metric suitable for NW should be well suited for KNN algorithms as well.

The modification required to execute NW(ML) with a learnt metric is straightforward – we simply start using an alternate kernel given by

$$K^{\text{Maha}}(\boldsymbol{x}^1, \boldsymbol{x}^2; \boldsymbol{\Sigma}) = \exp(-(d^{\text{Maha}}(\boldsymbol{x}^1, \boldsymbol{x}^2; \boldsymbol{\Sigma}))^2)$$

We note that this alternate kernel does not require an explicit bandwidth parameter since any such parameter can be absorbed into the matrix $\boldsymbol{\Sigma}$ itself. The method proposed by Weinberger and Tesauro (2007) learns this metric by attempting to minimize the leave-one-out RMSE over the training data points. We call the variants of NW, KNN and KNN-D when used with a learnt metric, respectively NW(ML), KNN(ML) and KNN-D(ML). Algorithm S1 presents pseudo-code for these variants.

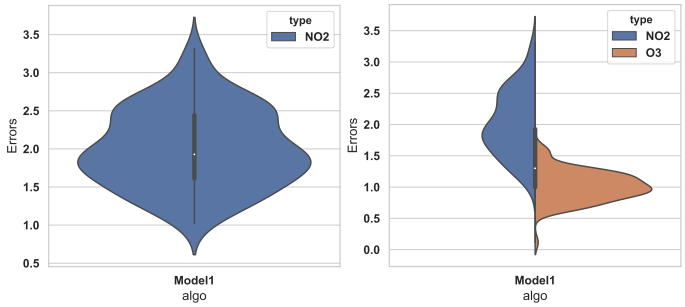

Figure S5. (Interpreting violin plots) Two violin plots based on synthetic error data. The left figure offers a *symmetric* violin plot on a single data source (NO_2 calibration in this synthetic example). The right figure offers a *split* violin plot that considers two data sources together (NO_2 and O_3 calibration in this synthetic example).

S4 Supplementary Results

We first present a helpful discussion on interpreting violin plots and then, for sake of convenience to the reader, revisit the discussion on how to interpret the statistical tests.

S4.1 Interpreting Violin Plots

Fig. S5 shows two sample violin plots based on synthetic error data (i.e. the data does not correspond to any actual model). Violin plots display numeric data by showing a rotated kernel density plot to show the distribution of the data (see Fig. S5 (left)). The white dot in middle represents the median of the data. The thick vertical line in the middle represents the inter-quartile range between the 0.25 and 0.75 quartiles, commonly known as Q1 and Q3. The thinner vertical line in the middle represents *Tukey's fences* and the upper and lower adjacent values, calculated as Q3 + 1.5Δ and Q1 - 1.5Δ respectively where Δ = Q3 - Q1 is the *interquartile range*. Violin plots may also be *split* to simultaneously display two sources of data for ease of comparison (see Fig. S5 (right)). For split violin plots, the median, quartiles, and Tukey's fences were calculated on the combined data from the two sources by our *seaborn* plotting library.

S4.2 Interpreting Two-sample Tests

As mentioned earlier, we used the paired Wilcoxon signed ranked test to compare two algorithms on the same dataset. Given that there are 12 datasets and 10 splits for each dataset, for ease of comprehension, we provide globally averaged statistics of wins scored by an algorithm over another. For example, say we wish to compare KNN-D(ML) and NW(ML) as done in Tab S5. We perform the test for each individual dataset and split. For each test, we either get a win for NW(ML) (in which case NW(ML) gets a +1 score and KNN-D(ML) gets 0), or a win for KNN-D(ML) (in which case KNN-D(ML) gets a +1 score and NW(ML) gets 0) or else the null hypothesis is not refuted (in which case both get 0). The average of these scores is then shown. For example, in Tab S5 (left), row 3 column 5 records a value of 0.02 implying that in 2% of these tests, NW(ML) won over KNN-D(ML) in case of O_3 calibration, whereas row 5 column 3 records a value of 0.62 implying that in 62% of the tests, KNN-D(ML) won over NW(ML). In the balance (1 - 0.02 - 0.62 = 0.36) i.e. 36% of the tests, neither algorithm could be declared a winner.

S4.3 The Effect of Metric Learning

The main paper discussed the need for metric learning in order to place appropriate emphasis on various features, such as RH and T that are known to hugely influence calibration, when calculating distances between data points in algorithms such as KNN or KRR. To assess whether metric learning is indeed discovering such emphasis, we try to understand the action of a Mahalanobis metric better. Recall that a Mahalanobis metric is characterized by a positive semi-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{8\times 8}$ and

Table S1. The linear transformation $\mathbf{\Sigma}^{\frac{1}{2}}$ learnt for NO_2 calibration on the dataset DD1(Jun). Note the large emphasis the transformation places on RH and T and no2op1, no2op2, no2diff, increasing their importance while calculating the Mahalanobis distance and placing relatively less importance on the oxop1, oxop2 and oxdiff features which is understandable since this metric was learnt for NO_2 calibration.

	T	RH	no2op1	no2op2	oxop1	oxop2	no2diff	oxdiff
T	**10.19**	3.29	-1.95	-2.12	3.73	4.29	-0.66	-1.44
RH	3.52	**13.22**	1.43	1.46	-2.32	-2.60	-0.25	0.49
no2op1	-0.17	-0.69	**6.92**	6.20	-3.65	-3.93	0.27	-0.12
no2op2	-0.27	-0.81	5.66	**6.96**	-2.94	-3.20	0.51	0.11
oxop1	1.27	-0.19	1.94	2.11	**1.51**	0.50	0.74	0.27
oxop2	0.89	-0.81	3.34	3.58	-0.86	**0.03**	0.86	0.24
no2diff	-0.74	-0.68	-4.01	-3.94	6.89	7.12	**2.82**	1.88
oxdiff	2.71	3.45	-7.03	-7.36	7.95	8.54	-0.32	**1.32**

calculates the distance between any two points as follows

$$d^{\text{Maha}}(\boldsymbol{x}^1, \boldsymbol{x}^2; \mathbf{\Sigma}) = \sqrt{(\boldsymbol{x}^1 - \boldsymbol{x}^2)^\top \mathbf{\Sigma} (\boldsymbol{x}^1 - \boldsymbol{x}^2)}$$

Let $\mathbf{\Sigma}^{\frac{1}{2}}$ denote the Cholesky decomposition of $\mathbf{\Sigma}$ such that $(\mathbf{\Sigma}^{\frac{1}{2}})^\top \mathbf{\Sigma}^{\frac{1}{2}} = \mathbf{\Sigma}^{\frac{1}{2}}$ and denote $\boldsymbol{v}^i = \mathbf{\Sigma}^{\frac{1}{2}} \boldsymbol{x}^i$ for $i = 1, 2$. Then it is not difficult to see that $d^{\text{Maha}}(\boldsymbol{x}^1, \boldsymbol{x}^2; \mathbf{\Sigma}) = \|\boldsymbol{v}^1 - \boldsymbol{v}^2\|_2$ i.e. the Mahalanobis distance between $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ is simply the Euclidean distance between the transformed vectors $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$. Thus, the transformation $\mathbf{\Sigma}^{\frac{1}{2}}$ is crucial in reorganizing features of $\boldsymbol{x}$ so so that the resulting distances, when used by the kNN algorithm, give better performance. Tab S1 shows the linear transformation $\mathbf{\Sigma}^{\frac{1}{2}}$ corresponding to the Mahalanobis metric learnt by NW(ML) for NO_2 calibration on the DD1(Jun) dataset which gives us in some sense optimal reorganization of features found by the metric learning technique. In particular, note that it places heavy emphasis on the RH and T features. This means that the optimal Mahalanobis metric identifies that a high importance should be placed on RH and T features when computing distances for use by kNN. We point out the following aspects of the matrix by concentrating on the diagonal entries.

1. The diagonal entries corresponding to no2op1, no2op2 and no2diff have much higher values that those for oxop1, oxop2 and oxdiff. This makes sense since this metric was being learnt for NO_2 calibration.

2. The diagonal entries corresponding to RH and T are by far the largest. This implies that the method did find it crucial to put more emphasis on these two features while calculating distances.

S4.4 Detailed Calibration Results

Here we present detailed outcomes of the calibration studies comparing algorithms within certain families e.g. Alphasense, linear, non-parametric etc. We recall that the main paper only included a summary of these results.

S4.4.1 Alphasense Family of Models

We evaluated the four Alphasense algorithms described in Sect. S3.1.1 on all datasets. Since there is no training required for these models, we directly applied them to the test data for all the splits. All four algorithms exhibit extremely poor performance across all metrics on all datasets, offering extremely high MAE and low R^2 values. Two-sample tests confirmed this by declaring all four AS algorithms as losers when compared to any other algorithm (e.g. LS or KNN). This was true for every split of every dataset. However AS3 was the better among the four algorithms and Table S2 presents various error metrics for this variant on two datasets across the two sites and deployments to illustrate the performance level of these algorithms. With the error scales being so huge, violin plots for these algorithms fail to convey useful information and are omitted.

Table S2. Performance of AS3 on data collected by the DD1 and DM2 sensor across the Jun and Oct deployments. All AS models offered extremely poor calibration performance on all datasets. Note the negative R^2 values and the extremely large MAE, MAPE and RMSE values across sites and deployments.

	DD1(Jun)		DM2(Oct)	
	O_3	NO_2	O_3	NO_2
MAE	251.8±0.27	107.7±0.11	13.6±1.0	354.6±2.3
RMSE	253.6±0.27	108.2±0.12	22.7±2.2	356.0±2.3
MAPE	1341.9±120.9	1954.6±51.8	142.4±6.5	8785.4±4607.4
R^2	-178.7±3.9	-95.5±3.5	-0.12±0.0	-1617.7±104.6

Previous studies (Lewis and Edwards, 2016; Jiao et al., 2016; Simmhan et al., 2019) corroborate this poor performance of the AS calibration algorithms. It was suggested that this is likely due to changing levels of confounding effects at the study sites (Kumar et al., 2015; Mijling et al., 2018; Simmhan et al., 2019; Chatzidiakou et al., 2019). In particular, Chatzidiakou et al. (2019) reported that the electrochemical sensors lose sensitivity at higher temperatures in field. Models that do take into account RH and T as explicit features were found to perform much better. Given the extremely poor performance of these models, we do not consider them in our analyses anymore.

S4.4.2 Linear Parametric Family of Models

Among the linear parametric algorithms LS(MIN), LASSO and LS, we found LS to offer the best performance. Tab S3 shows that the paired Wilcoxon test awarded LS a win over the other two algorithms a majority of the time over the 12 datasets and 10 splits of each dataset (see Table S3). A visual inspection of the distribution of absolute errors offered by the three algorithms (see Fig. S6) confirm that the larger number of features (augmented as well as RH and T) offered to the LS algorithm do offer it an advantage. This also confirms that including RH and T as features during the calibration process is essential.

Table S3. Results of the pairwise Wilcoxon signed rank tests on linear parametric models (see Sect. S4.2 for a key). LS wins over both LS(MIN) and LASSO a majority of the times and is not defeated by any of them more than marginally often. The overall ranking of the algorithms in terms of performance is indicated to be LS > LS(MIN) > LASSO

	O_3				NO_2		
	LS(MIN)	LASSO	LS		LS(MIN)	LASSO	LS
LS(MIN)	0	0.83	0.05	LS(MIN)	0	0.61	0.07
LASSO	0.01	0	0.01	LASSO	0.23	0	0
LS	0.65	0.91	0	LS	0.76	0.79	0

S4.4.3 Kernel Regression Family of Models

We confirmed that the Nystroem method does indeed offer competitive calibration performance as compared to kernel ridge regression (KRR). In around 47% of the tests, KRR was found to beat the Nystroem method whereas in around 35% of the tests, Nystroem beat KRR. 18% of the tests were inconclusive in declaring a winner. The violin plots for the two algorithms (see Fig. S7) can be used to visually confirm that the algorithms do indeed offer comparable performance. However, as Tab S4 shows, the prediction time offered by the Nystroem method can be more than $4\times$ faster in terms of prediction time than KRR. This highlights the utility of the Nystroem method as an accurate but accelerated approximation for KRR.

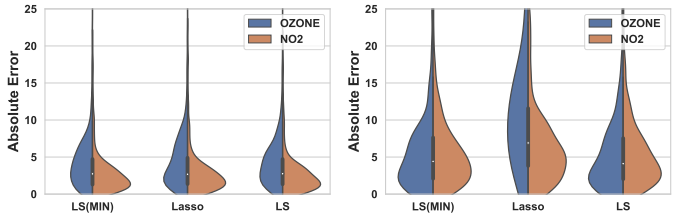

Figure S6. The violin plots on the left and right depict the distribution of absolute errors incurred by various linear parametric calibration models on respectively, the MM5(Jun) and MM5(Oct) datasets. LS offers visibly superior performance on the MM5(Oct) dataset.

Table S4. Prediction time speedups offered by the Nystroem method over the KRR algorithm on both sites and deployments. Notice that the speedup is generally higher for larger datasets.

			Test time per data point (ms)		
		# Train Data Points	KRR	NYS	Speedup
Site D	Jun	23626.5±1299.1	0.39±0.24	0.07±0.042	**4.4**×
	Oct	6682.8±1013.7	0.09±0.08	0.02±0.011	**3.9**×
Site M	Jun	1647.5±229.3	0.14±0.02	0.02±0.007	**5.9**×
	Oct	742.7±173.0	0.009±0.006	0.005±0.002	**1.5**×

S4.5 KNN and Metric Learning Family of Models

Among the KNN family of algorithms, the distance weighted KNN algorithm that uses a learnt metric i.e. KNN-D(ML) was found to offer the best accuracies across all datasets and splits. Table S5 reports the results of the paired Wilcoxon tests comparing all algorithms. In general, KNN-D(ML) was awarded a win over other variants a majority of the time. Some other trends evident from this analysis are the following

1. using a learnt metric always improves performance (KNN-D(ML) wins over KNN-D 76% and 70% of the time w.r.t O_3 and NO_2 calibration, and KNN(ML) wins over KNN 76% and 70% of the time as well).

2. distance-weighing always improves performance (KNN-D(ML) wins over KNN(ML) 81% and 86% of the time, and KNN-D wins over KNN 79% and 85% of the time).

Tab S6 additionally presents various error metrics for these algorithms on the DD1(Jun) and MM5(Oct) datasets. We avoid presenting a violin plot in this case since the plots are not readily discernible.

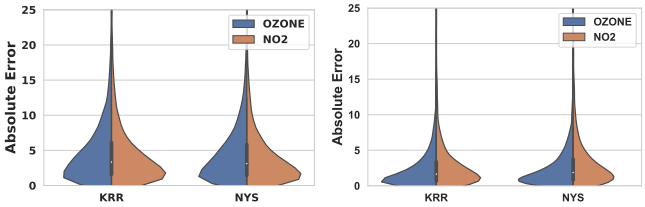

Figure S7. The violin plots on the left and right depict the distribution of absolute errors incurred by KRR and the Nystroem method on respectively, the DD1(Jun) and DD1(Oct) datasets. Both algorithms can be seen to offer comparable performance on both datasets.

Table S5. Results of the pairwise Wilcoxon signed rank tests on KNN and metric learning models (see Sect. S4.2 for a key). KNN-D(ML) beats every other algorithm a large fraction of the time and is scarcely ever beaten. The overall ranking of the algorithms is indicated to be KNN-D(ML) > KNN(ML) > KNN-D > NW(ML) > KNN although, as Tab S6 indicates, in terms of error metrics, KNN(ML), KNN-D and NW(ML) are competitive as well.

O_3					
	KNN	KNN-D	NW(ML)	KNN(ML)	KNN-D(ML)
KNN	0	0.01	0.46	0.01	0
KNN-D	0.79	0	0.5	0.03	0
NW(ML)	0.29	0.25	0	0.12	0.02
KNN(ML)	0.76	0.64	0.59	0	0
KNN-D(ML)	0.86	0.76	0.62	0.81	0

NO_2					
	KNN	KNN-D	NW(ML)	KNN(ML)	KNN-D(ML)
KNN	0	0	0.3	0	0
KNN-D	0.85	0	0.34	0.03	0.01
NW(ML)	0.34	0.27	0	0.18	0.06
KNN(ML)	0.7	0.58	0.56	0	0
KNN-D(ML)	0.81	0.7	0.58	0.86	0

Table S6. A comparison of various KNN and metric learning algorithms on the DD1(Jun) and MM5(Oct) datasets with respect to the MAE and R^2 metrics. The best algorithms in terms of mean statistics are highlighted in bold.

O_3				
	DD1(Jun)		MM5(Oct)	
	MAE	R^2	MAE	R^2
KNN	3.88±0.04	0.909±0.005	3.14±0.19	0.936±0.01
KNN-D	3.82±0.03	0.911±0.004	3.06±0.18	0.940±0.01
NW(ML)	4.23±0.06	0.895±0.005	**2.90±0.27**	**0.943±0.03**
KNN(ML)	3.57±0.05	0.921±0.003	3.02±0.28	0.939±0.03
KNN-D(ML)	**3.52±0.04**	**0.923±0.003**	2.98±0.27	**0.943±0.03**

NO_2				
	DD1(Jun)		MM5(Oct)	
	MAE	R^2	MAE	R^2
KNN	3.19±0.02	0.757±0.01	2.94±0.21	0.729±0.03
KNN-D	3.13±0.02	0.761±0.01	2.84±0.20	0.744±0.03
NW(ML)	3.49±0.07	0.717±0.02	**2.75±0.22**	**0.751±0.04**
KNN(ML)	2.74±0.06	0.808±0.02	2.87±0.26	0.738±0.04
KNN-D(ML)	**2.67±0.05**	**0.819±0.01**	2.79±0.26	**0.751±0.04**

Competing interests. Author Ronak Sutaria is the CEO of Respirer Living Sciences Pvt. Ltd. which builds and deploys low-cost sensor based air quality monitors with trade-name 'Atmos - Realtime Air Quality'. Ronak Sutaria's involvement was primarily in the development of the air quality sensor monitors and the big data enabled application programming interfaces to access the temporal data, but not in the data analysis. Author Brijesh Mishra, subsequent to the work presented in this paper, has joined the Respirer Living Sciences team. The authors declare no other competing interests.

Acknowledgements. This research has been supported under the Research Initiative for Real-time River Water and Air Quality Monitoring program funded by the Department of Science and Technology, Government of India, and Intel® and administered by the Indo-United States Science and Technology Forum (IUSSTF).

References

Aggarwal, M.: NITI Aayog's action plan for air pollution: Old wine in a new bottle?, News article at Mongabay-India, https://india.mongabay.com/2018/07/niti-aayogs-action-plan-for-air-pollution-old-wine-in-a-new-bottle/. Accessed 01 April 2020., 2018.

Caputo, B., Sim, K., Furesjo, F., and Smola, A.: Appearance-based object recognition using SVMs: which kernel should I use?, in: NIPS workshop on Statistical Methods for Computational Experiments in Visual Processing and Computer Vision, 2002.

Chatzidiakou, L., Krause, A., Popoola, O. A., Di Antonio, A., Kellaway, M., Han, Y., Squires, F. A., Wang, T., Zhang, H., Wang, Q., et al.: Characterising low-cost sensors in highly portable platforms to quantify personal exposure in diverse environments, Atmospheric Measurement Techniques, 12, 4643, 2019.

Chowdhury, S., Dey, S., Guttikunda, S., Pillarisetti, A., Smith, K. R., and Di Girolamo, L.: Indian annual ambient air quality standard is achievable by completely mitigating emissions from household sources, Proceedings of the National Academy of Sciences, 116, 10 711–10 716, 2019.

Cross, E. S., Williams, L. R., Lewis, D. K., Magoon, G. R., Onasch, T. B., Kaminsky, M. L., Worsnop, D. R., and Jayne, J. T.: Use of electrochemical sensors for measurement of air pollution: correcting interference response and validating measurements, Atmospheric Measurement Techniques, 10, 3575–3588, 2017.

Gaur, A., Tripathi, S., Kanawade, V., Tare, V., and Shukla, S.: Four-year measurements of trace gases (SO_2, NO_x, CO, and O_3) at an urban location, Kanpur, in Northern India, Journal of Atmospheric Chemistry, 71, 283–301, 2014.

Jiao, W., Hagler, G., Williams, R., Sharpe, R., Brown, R., Garver, D., Judge, R., Caudill, M., Rickard, J., Davis, M., et al.: Community Air Sensor Network (CAIRSENSE) project: evaluation of low-cost sensor performance in a suburban environment in the southeastern United States, Atmospheric Measurement Techniques, 9, 5281–5292, 2016.

Kumar, P., Morawska, L., Martani, C., Biskos, G., Neophytou, M., Di Sabatino, S., Bell, M., Norford, L., and Britter, R.: The rise of low-cost sensing for managing air pollution in cities, Environment International, 75, 199–205, 2015.

Lewis, A. and Edwards, P.: Validate personal air-pollution sensors, Nature, 535, 29–31, 2016.

Mijling, B., Jiang, Q., de Jonge, D., and Bocconi, S.: Field calibration of electrochemical NO2 sensors in a citizen science context, Atmospheric Measurement Techniques, 11, 1297–1312, 2018.

Murphy, K. P.: Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.

Nadaraya, E. A.: On Estimating Regression, Theory of Probability and Its Applications, 9, 141–142, 1964.

Simmhan, Y., Nair, S., Monga, S., Sahu, R., Dixit, K., Sutaria, R., Mishra, B., Sharma, A., SVR, A., Hegde, M., Zele, R., and Tripathi, S. N.: SATVAM: Toward an IoT Cyber-infrastructure for Low-cost Urban Air Quality Monitoring, in: 15th IEEE International Conference on e-Science (eScience 2019), 2019.

Tibshirani, R.: Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society: Series B (Methodological), 58, 267–288, 1996.

Times, H.: Faridabad second most polluted city in country, Gurgaon 12th, Newspaper article, https://www.hindustantimes.com/gurgaon/faridabad-second-most-polluted-city-in-country-gurgaon-12th/story-MJoViOnWDehKKw1QEEeLlO.html. Accessed on 01 April 2020., 2018.

Tiwari, S., Bisht, D., Srivastava, A., and Gustafsson, Ö.: Simultaneous measurements of black carbon and $PM_{2.5}$, CO, and NO x variability at a locally polluted urban location in India, Natural Hazards, 75, 813–829, 2015.

Watson, G. S.: Smooth regression analysis, Sankhyā: The Indian Journal of Statistics, Series A, 26, 359–372, 1964.

Weinberger, K. Q. and Tesauro, G.: Metric Learning for Kernel Regression, in: 11th International Conference on Artificial Intelligence and Statistics (AISTATS), 2007.

Williams, C. and Seeger, M.: Using the Nystroem method to speed up kernel machines, in: Neural Information Processing Systems (NIPS), 2001.