chemosensors

MDPI

Article

Kullback–Leibler Importance Estimation Procedure to Improve Gas Quantification in an Electronic Nose

Daniel Alejandro Piracoca Gordillo [1], Maria Camila Cardenas Castellanos [1], David Nicolás Torres Barrera [1], Jaime Alberto Escobar Gomez [2], Juan Felipe Nieto Sanchez [1] and Jersson X. Leon-Medina [1,3,*]

[1] Department of Mechatronics Engineering, Universidad de San Buenaventura, Bogotá 110141, Colombia
[2] Department of Aeronautic Engineering, Universidad de San Buenaventura, Bogotá 110141, Colombia
[3] Control, Modeling, Identification and Applications (CoDAlab), Department of Mathematics, Escola d'Enginyeria de Barcelona Est (EEBE), Campus Diagonal-Besòs (CDB), Universitat Politècnica de Catalunya (UPC), Eduard Maristany 16, 08019 Barcelona, Spain
* Correspondence: jersson.xavier.leon@upc.edu

Abstract: An electronic nose sensor array can classify and quantify different types of gases; however, the sensor can alter its measurement capability over time. The main problem presented during the measurements of the sensors is related to the variation of the data acquired for long periods due to changes in the chemosensory response, thus affecting the correct functioning of the implemented measuring system. This research presents an approach to improve gas quantification through the implementation of machine learning regression techniques in an array of nose-type electronic sensors. The implemented methodology uses a domain adaptation approach with the Kullback–Leibler importance estimation procedure (KLIEP) to improve the performance of the gas quantification electronic nose array. This approach is validated using a three-year dataset measured by a 16-electronic-nose-sensor array. The R2 regression error obtained for each of the gases fits the resulting dataset's measured values with good precision.

Keywords: electronic nose; quantification; Kullback-–Leibler importance estimation procedure; KLIEP; domain adaptation; regression; machine learning; sensor array

Citation: Piracoca Gordillo, D.A.; Cardenas Castellanos, M.C.; Torres Barrera, D.N.; Escobar Gomez, J.A.; Nieto Sanchez, J.F.; Leon-Medina, J.X. Kullback–Leibler Importance Estimation Procedure to Improve Gas Quantification in an Electronic Nose. *Chemosensors* **2022**, *10*, 538. https://doi.org/10.3390/chemosensors10120538

Academic Editor: Jose V. Ros-Lis

Received: 1 November 2022
Accepted: 12 December 2022
Published: 15 December 2022

1. Introduction

Electronic nose sensor arrays have emerged as a compact, economical and versatile solution to analyze different analytes [1,2]. Electronic nose sensor arrays process information similar to how human senses transmit information about smell to the brain [3]. These electronic nose devices are composed of a sensor array, an electronic data acquisition system, and a pattern recognition unit [4]. The development of better electronic noses is a current trend and can be performed in each of its three components. This study focuses on the improvement of the pattern recognition unit.

Over time, electronic nose sensors have stopped work correctly due to variations in their chemical properties [5]. This time-passing phenomenon is known as drift and has been treated with different approaches in the literature. For example, the research done by Vergara et al. [6] obtained a 36-month dataset from an electronic nose made up of 16 sensors tagged by the manufacturer as TGS2600, TGS2602, TGS2610, and TGS2620 and placed into a test chamber, as is denoted in the research by Vergara et al. The authors successfully used an ensemble machine learning method to classify six different gases (ammonia, ethanol, acetaldehyde, toluene, acetone, and ethylene) with 54% accuracy. The dataset mentioned above has concentration data in parts per million (ppm) of the six gases. The solution of quantification problems in electronic noses allows detecting the concentration of the analytes with low error. These kinds of organic compounds have toxic

capabilities, and determining their concentration with low error is challenging due to the sensors' drift effect [7].

Pattern recognition and artificial intelligence techniques have been used in electronic noses to quantify analytes with high low error [8]. Several machine learning algorithms for regression have been used for processing electronic nose data. Some of them are: partial least square regression (PLSR) [9–11], multilayer perceptron neural networks (MLP-NN) [12,13], support-vector regression (SVR) [14], random forest regression [15,16], long-short term memory (LSTM) [17,18], reservoir computing (RC) [19], and Gaussian process (GP) [20], among others. In the following, a detailed explanation of some important studies related to the use of machine learning regression algorithms for quantification tasks using electronic noses are discussed. In 2016, Fonollosa et al. [21] developed an electronic nose sensor array to quantify methane, ethylene, carbon monoxide and ethanol. Support-vector regression (SVR) was used as the regressor algorithm to determine the concentration of the four gases. The results indicated third quartile errors in ppm for each gas according to the following values: ethanol, 9.4 ppm; ethylene, 5.5 ppm; methane, 14 ppm; and carbon monoxide, 27 ppm. In other work, a fuzzy ART-based concentration estimator was used to quantify hydrogen sulfide, ammonia and their mixture with an electronic nose [22]. The results of the estimated concentrations were measured using the root-mean-square error (RMSE) reaching values of 1.7835 ppm, 0.0227 ppm, and 1.1859/0.0090 ppm for ammonia, hydrogen sulfide and their mixtures, respectively. An electronic nose [23] made with only four sensors was developed to quantify six kinds of gases including ammonia, nitrogen dioxide, toluene, carbon monoxide, formaldehyde and benzene. A multilayer perceptron (MLP) neural network with the multiple multiple inputs single output (MMISO) structure was used to determine the best concentration. The results were expressed in terms of mean square error of prediction (MSEP), with the particle swarm optimization bacterial chemotaxis-back propagation (PSOBC-BP) algorithm being the one that found less MSEP, with 3.327% on average. A metal oxide (MOx)-decorated graphene-based electronic nose sensor array was developed to quantify HCHO and NH_3 in [24] . The signals were processed by a back-propagation neural network (BP-NN) finding a mean absolute error (MAE) of 0.372 ppm and 0.274 ppm for HCHO and NH_3, respectively.

However, the quantification studies in electronic noses mentioned above have yet to consider the passage of time in the sensors of an electronic nose. This study develops a signal processing methodology based on a domain adaptation approach called the Kullback–Leibler importance estimation procedure (KLIEP) [25] to treat the drift problem for gas quantification. Domain adaptation methods are an alternative to traditional machine learning methods that use a transfer learning approach [26]. There are two different data domains in domain adaptation: first, the source domain, and second, the target domain [27]. The source and target domains are part of the electronic nose data grouped in different batches over time. The data distribution of each domain varies over time [28]. Feature, instance, and parameter-based domain adaptation methods offer a solution when only labeled data are in the source domain [29]. The instance-based methods reweight labeled training data to correct the difference between source and target distributions. The KLIEP method is instance-based. The use of the KLIEP method as a domain adaptation procedure is, to the best of the authors' knowledge, presented for the first time to solve the electronic nose data quantification over time problem.

At first, the introduction describes the behavior of electronic nose sensors, emphasizing their main characteristics for proper functioning. The second section presents a selection of the dataset used in the research, their components, and the conditions in which the electronic nose-type sensor array was exposed. The third section compares the best machine learning regressor between XGBoost, simple linear regression and AdaBoost. These regressors are evaluated based on their behavior with the different batches in the dataset. Tuning parameters are also performed, obtaining the R2 error by training the regressor with 80% of the data from batch 1. Then, the fourth section applies the Kullback—Leibler importance estimation procedure (KLIEP) method of domain adaptation in addition to the

selected regressor to improve the performance of the response for the R2 error to quantify different gases. Finally, the last section shows the conclusion and further future works on this research topic.

2. Materials and Methods

2.1. Dataset

The research done by Vergara et al. [6] concerns different variants of drift found in electronic nose-type sensor arrays. They mainly found the actual drift or first-order drift, which is due to the processes of chemical and physical interaction of chemical analytes in the gas phase that occurs in the microstructure of the detection film. The second-order drift, or drift of the measurement system, is produced by external and uncontrollable alterations of the experimental operating system, such as temperature [6].

In their research, Alexander Vergara and his colleagues decided to make and implement an electronic nose sensor in order to control external alterations that can present a second-order drift and focused only on the compensation of a first-order drift, which evidences the use of these sensors for a long period of time.

The electronic nose sensor array was composed of four kind of sensors from the Figaro Company, TGS2620, TGS2602, TGS2600 and TGS2610; four of each one of these sensors composed the sensor array for a total of 16 sensors. A sampling frequency of 100 Hz was used during each measure. The time spent for the measures was approximately 300 s. As a result, each sensor acquired a resistance measurement of 300 Hz $\times$ 100 s = 30,000 datapoints. The experiments comprised adsorption and desorption stages; between these stages, there was a steady state of the signal. For these reasons, a feature extraction procedure was performed on the signal. Only three features in the adsorption stage, two features in the steady state stage and finally three features in the desorption stage were considered. A total of 8 features composed the feature vector for each sensor in the electronic nose. The electronic nose sensor array had 16 sensors; due to this, a total of 16 $\times$ 8 = 128 features were acquired in each experiment. For a detailed description of the mathematical procedure used in the feature extraction, the reader is referred to [1].

The dataset of Vergara et al. includes measures of the following six gases:

- Ammonia;
- Acetaldehyde;
- Acetone;
- Ethylene;
- Ethanol;
- Toluene.

These gases were dosed in different concentrations and used as the response variable in this research. Table 1 shows the amount of gas measured in parts per million (ppm).

The database was constructed by arranging the measurements in different processes during a set period of time. It is important to note that the sampling process has no particular order to ensure the quantification of these gases.

Odor concentrations in parts per million (PPM) of the samples collected for the six gases can be seen in Tables 1 and 2, showing the number of samples taken for each gas per month of study:

Table 2 describes the 36 months of the dataset consisting of 13,910 samples. It demonstrates the drift of the sensors over time between the last months (months 25 to 29 and months 31 to 35), where there are missing data due to severe contamination attached to the detection layer since there was no temperature control. This dataset is publicly available and can be found in [30].

Table 1. Analytes and concentrations in the dataset [6].

Analytes	Concentrations in ppm
Ammonia	50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 260, 270, 275, 280, 290, 300, 350, 400, 450, 500, 600, 700, 750, 800, 900, 950, 1000
Acetaldehyde	5, 10, 13, 20, 25, 30, 35, 40, 45, 50, 60, 70, 75, 80, 90, 100, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 300, 500
Acetone	12, 25, 38, 50, 60, 62, 70, 75, 80, 88, 90, 100, 110, 120, 125, 130, 140, 150, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 260, 270, 275, 280, 290, 300, 350, 400, 450, 500, 1000
Ethylene	10, 20, 25, 30, 35, 40, 50, 60, 70, 75, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 300
Ethanol	10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100, 110, 120, 125, 130, 140, 150, 160, 170, 175, 180, 190, 200, 210, 220, 225, 230, 240, 250, 275, 500, 600
Toluene	10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100

Table 2. Samples taken month by month in the electronic nose dataset for each gas [6].

Samples Taken Month by Month							
Identification	**Ammonia**	**Acetaldehyde**	**Acetone**	**Ethylene**	**Ethanol**	**Toluene**	**Total**
month1	76	0	0	88	84	0	248
month2	7	30	70	10	6	74	197
month3	0	0	7	140	70	0	217
month4	0	4	0	170	82	5	261
month8	0	0	0	20	0	0	20
month9	0	0	0	4	11	0	15
month10	100	105	525	0	1	0	731
month11	0	0	0	146	360	0	506
month12	0	192	0	334	0	0	526
month13	216	48	275	10	5	0	554
month14	0	18	0	43	52	0	113
month15	12	12	12	0	12	0	48
month16	20	46	63	40	28	0	197
month17	0	0	0	20	0	0	20
month18	0	0	0	3	0	0	3
month19	110	29	140	100	264	9	652
month20	0	0	466	451	250	458	1625
month21	360	744	630	662	649	568	3613
month22	25	15	123	0	0	0	163
month23	15	18	20	30	30	18	131
month24	0	25	28	0	0	1	54
month30	100	50	50	55	61	100	416
month36	600	600	600	600	600	600	3600

Table 3 presents samples from each gas for each month arranged in 10 different batches for the study. The idea was to obtain data on all six gases from each batch. However, by rearranging these data in batch 3, batch 4, and batch 5, it was shown that there were no toluene samples. Table 3 also specifies the months that integrate each batch.

2.2. Kullback–Leibler Importance Estimation Procedure (KLIEP)

KLIEP is an instance-based method for domain adaptation. The purpose of the algorithm is to correct the difference between input distributions of source and target domains. It is executed by finding a source instance reweighting that minimizes the Kullback–Leibler divergence between source and target distributions. It is not the purpose to present a detailed mathematical formulation and description of the KLIEP method,

but the source instance weights are given by the following Equation (1), and the description of the KLIEP method is shown in [31]. For more information about the KLIEP method, one can refer to the following literature [25].

$$w(x) = \sum_{x_i \in X_T} \alpha_i K(x, x_i) \quad (1)$$

where:

x, x_i are input instances.
X_T are the target input data.
α_i are the basis functions coefficients.
$K(x, x_i) = \exp\left(-\gamma \|x - x_i\|^2\right)$ for instance if kernel = "rbf".

Table 3. Number of measurements and batch information of the electronic nose dataset with drift [8].

		Samples Taken Batch by Batch						
Batch ID	**Month**	**Acetaldehyde**	**Ethanol**	**Toluene**	**Ammonia**	**Ethylene**	**Acetone**	**Total**
Batch 1	1,2	98	83	74	70	30	90	445
Batch 2	3,4,8,9,10	334	100	5	532	109	164	1244
Batch 3	11,12,13	490	216	0	275	240	365	1586
Batch 4	14,15	43	12	0	12	30	64	161
Batch 5	16	40	20	0	63	46	28	197
Batch 6	17,18,19,20	574	110	467	606	29	514	2300
Batch 7	21	662	360	568	630	744	649	3613
Batch 8	22,23	30	40	18	143	33	30	294
Batch 9	24,30	55	100	101	78	75	61	470
Batch 10	36	600	600	600	600	600	600	3600

The KLIEP algorithm aims to find an optimal α_i approached by finding the local maximum according to the following optimization problem solved with the gradient ascent algorithm:

$$\max_{\alpha_i} \sum_{x_j \in X_T} \log\left(\sum_{x_i \in X_T} \alpha_i K(x_j, x_i)\right) \quad (2)$$

Subject to:

$$\sum_{x_j \in X_s} \sum_{x_i \in X_T} \alpha_i K(x_j, x_i) = n_s \quad (3)$$

where:

X_s is the source input data of size n_s.

Besides, a cross-validation procedure (LCV) is added to select the appropriate parameters for the kernel function K, known as the parameter γ of the Gaussian kernel. It allows for choosing the best parameter using cross-validation on the J score as follows:

$$J = \frac{1}{|X|} \sum_{x \in X} \log(w(x)) \quad (4)$$

Finally, an estimator is fitted over the regressor by using the reweighted labeled source instances. The KLIEP method was originally introduced for unsupervised domain adaptation but could be widened to a supervised domain by simply adding labeled target data to the training set.

2.3. Methodology for Regressor Selection

The programming environment used for the data treatment was python; specifically, the scikitlearn [32], pandas [33], matplotlib [34] and adapt [35] libraries were used.

The steps or processes that describe the gas quantification methodology for the regressor selection are shown in the following Figure 1:

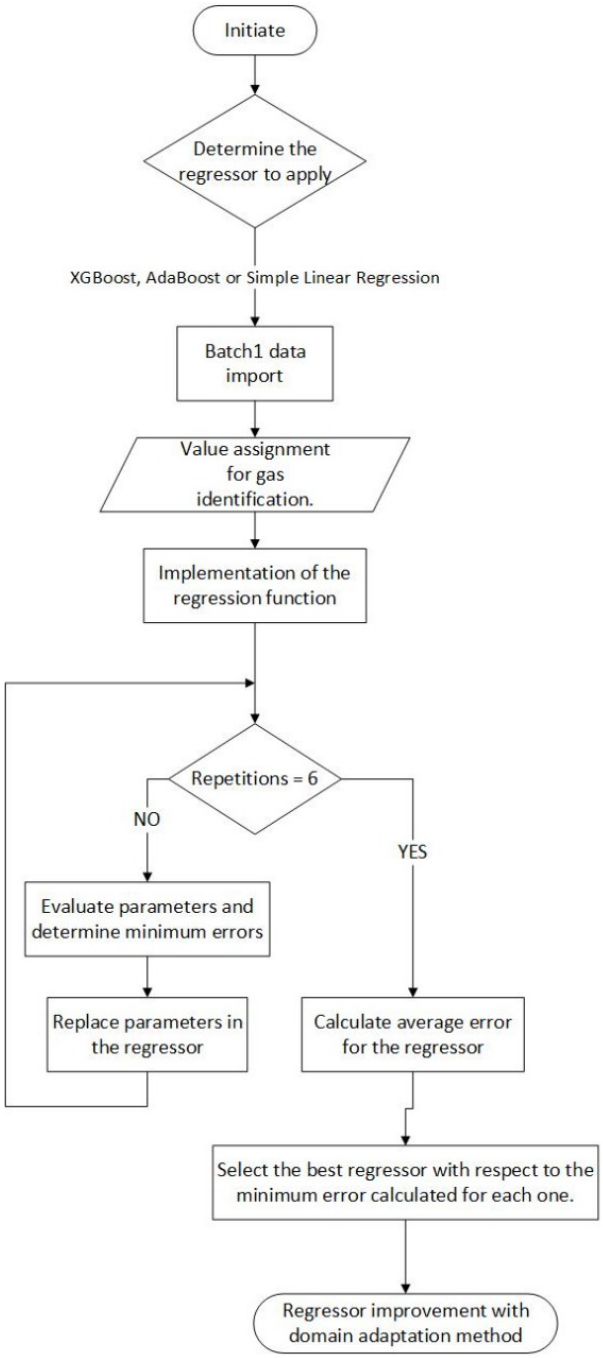

Figure 1. Methodology for selection of the regressor.

1. **Determine the regressor:** For this research, three regressors were implemented: extreme gradient boosting (XGBoost), simple linear regression and AdaBoost.
2. **Import batch data to study:** The dataset had ten batches, as described in Table 3. To choose and define which is the best regressor, it was decided to work with batch 1 by taking the first samples obtained in the research by Vergara et al., where the electronic nose sensor array takes the first defined data.
3. **Assignment of variables to recognize each gas:** It was necessary to assign a variable for each gas that allows the system to validate the behavior of the regressor for each gas.
4. **Implementation of the regression function:** The comparison of the chosen R2 error with the different parameters of the regressor model algorithm allowed determining if the value was close to the model, validating the selection of the regressor function.

5. **Replacing the values in the regressor function:** Once the corresponding error for each parameter was known, they were substituted in the applied regressor function, and in this way, the error was obtained for each gas in the dataset for the regressor.
6. **Evaluation of the errors:** Steps 4 and 5 were repeated for each gas in the dataset using the same model to determine the regression error for each gas according to the dataset.
7. **Minimum error applied to each regressor:** the objective was to evaluate the error applied to each regressor for the six gases to obtain the average error for the evaluated regressor.
8. **Selection of the regressor:** Steps 1 to 7 were carried out for all the regressors to compare the minimum average error to select the regressor that best fit the dataset.

2.4. Extreme Gradient Boosting (XGBoost) Regressor

The parameters of the XGBoost regressor function were:

- *n_estimators*
- *max_depth*
- *eta*
- *subsample*
- *colsample_bytree*

In consequence, to obtain the optimal values for the XGBoost regressor function, it is necessary to develop an iterative process over each parameter, selecting an R2 error value closer to 1.

The R2 error ratio is defined as the sum of the square regression (SSR) that represents the total variation of all the predicted values and the mean value on the regression over the sum of the square total (SST) that represents the total variation of the actual values and its mean value [36], as follows:

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} \quad (5)$$

The *n_estimators* parameter is evaluated at first by modifying its value according to the magnitude error, obtaining a number of estimators between 50 and 100 as the closest value, as shown in Figure 2:

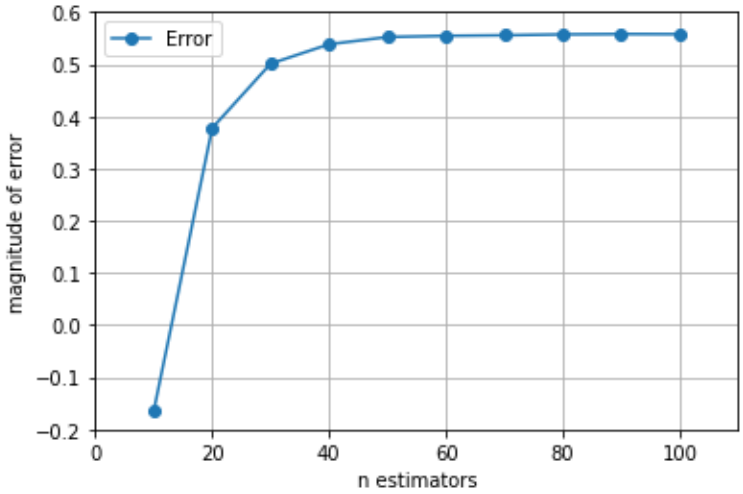

Figure 2. Errors of the n_estimators parameter for acetone gas.

Then, the previous process is repeated for the other parameters, resulting the following values represented in Figure 3. Max_depth parameter is shown in Figure 3a), the variation of eta in Figure 3b), sample_bytree parameter in Figure 3c), and finally the subsample parameter in Figure 3d).

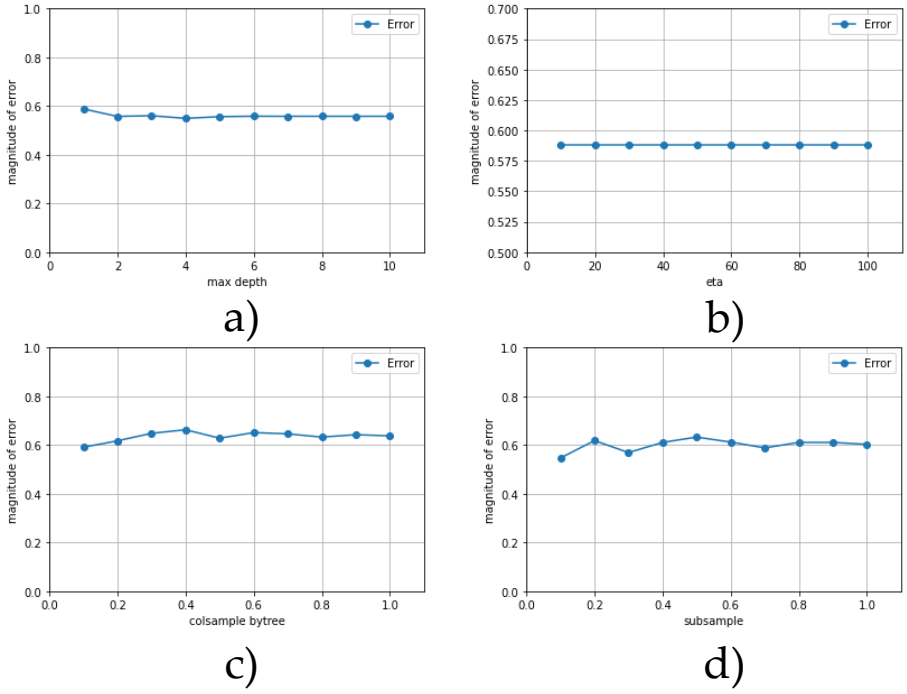

Figure 3. XGboost parameter behaviors: (**a**) max-depth, (**b**) eta, (**c**) colsample by tree, and (**d**) subsample.

Subsequently, the XGBoost regressor function was rewritten with the values of the previously determined parameters to obtain the error for the first gas presented as follows:

XGBRegressor(n_estimators = 90, max_depth = 1, eta = 10, subsample = 0.5, colsample_bytree = 0.4) (6)

With these parameters, the R2 error was 0.6627. To ensure an accurate result for the first gas, 80% of the data from batch 1 were used for training and the other 20% were used for testing, resulting in the following tracking trajectory for the regressor in Figure 4.

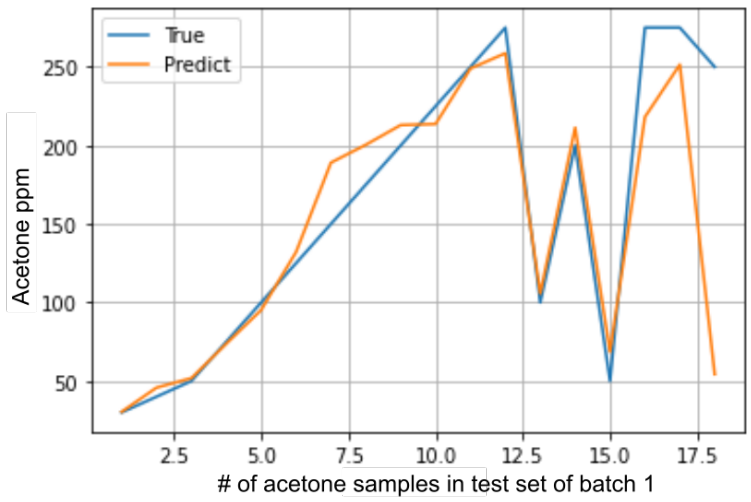

Figure 4. True vs. predicted behavior of the XGBoost regressor for acetone in the test set of batch 1 (18 measurements).

Once the error for the first gas (acetone) was determined, the same process was repeated for the other five gases. The results for the XGBoost regressor are presented in Table 4.

Table 4. Error for each gas with XGBoost.

Gases/Regressor	XGBoost
Acetone	0.662757
Acetaldehyde	0.960751
Ethanol	0.467232
Ethylene	0.9824
Ammonia	0.988174
Toluene	0.994229

After calculating the R2 error for each of the gases (in Table 4), the errors were averaged to evaluate the performance of the XGBoost regressor with the batch 1 dataset.

$$Error_{XGBoostAverage} = 0.842590 \quad (7)$$

2.5. AdaBoost Regressor

The parameters for the AdaBoost regressor function were:

- *max_depth*
- *n_estimators*
- *random_state*

Next, applying the same methodology for the AdaBoost regressor, the R2 error was calculated for each parameter. Figure 5a) presents the behavior for the parameter max_depth; Figure 5b) presents the behavior for the parameter n_estimators, and Figure 5c) shows the behavior for the parameter random_state.

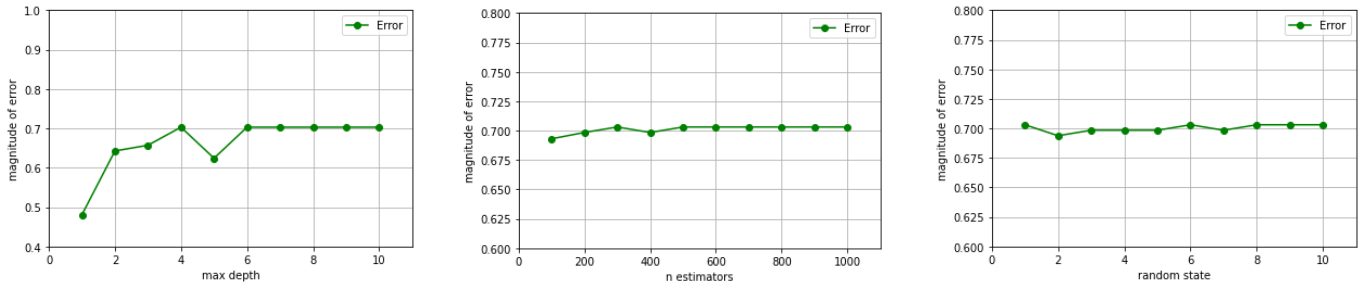

Figure 5. Adaboost regressor parameter tuning versus R2 error for (a) max-depth, (**b**) n-estimators and (**c**) random state.

The AdaBoost regressor function was rewritten with the values of the previously determined parameters to obtain the actual error for the first gas in batch 1, as follows:

AdaBoostRegressor(DecisionTreeRegressor(max_depth = 10), n_estimators = 800, random_state = 1) (8)

With these parameters, the R2 error for gas #1 was 0.7030. As before, the system was trained and tested using 80% and 20%, respectively, of the dataset from batch 1 for the first gas (acetone), resulting in the following tracking trajectory in Figure 6:

Once the R2 error of gas #1 was determined, the same process was repeated for the other five gases.

The results for the AdaBoost regressor are presented in Table 5.

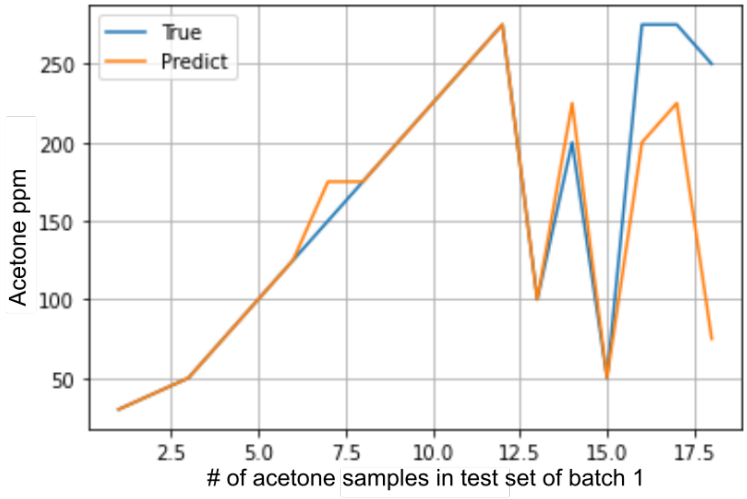

Figure 6. True vs. predicted behavior of the AdaBoost regressor for acetone in the test set of batch 1 (18 measurements).

Table 5. Error for each gas with AdaBoost.

Gases/Regressor	AdaBoost
Acetone	0.703059
Acetaldehyde	0.985634
Ethanol	0.680716
Ethylene	0.955653
Ammonia	0.984298
Toluene	0.992297

After calculating the R2 error for each of the gases (in Table 5), the errors were averaged to evaluate the performance of the AdaBoost regressor with the batch 1 dataset.

$$Error_{AdaBoost\,Average} = 0.883609 \quad (9)$$

2.6. Simple Linear Regression

Finally, the R2 error was evaluated directly with a comparison projected by a linear regressor for batch 1, as is shown in Figure 7.

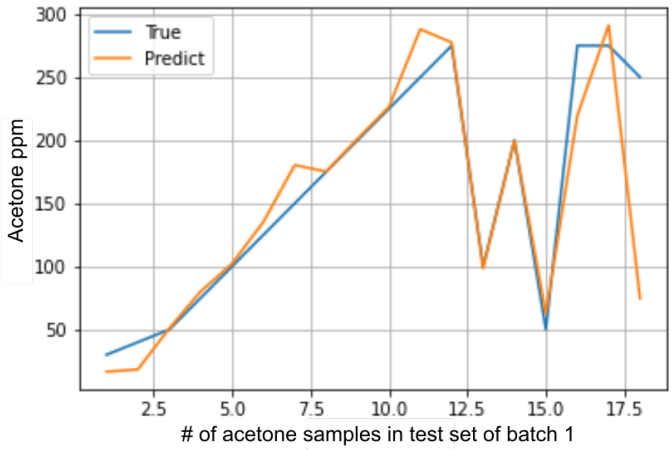

Figure 7. True vs. predicted behavior of the simple linear regressor for acetone in the test set of batch 1 (18 measurements).

The value of the R2 error for the first gas was 0.7223.

As before, repeating the process for the other five gases, the results in Table 6 are obtained for all gases with the simple linear regressor.

Table 6. Error for each gas with the simple linear regressor.

Gases/Regressor	Simple Linear Regressor
Acetone	0.722352
Acetaldehyde	0.982838
Ethanol	0.11206
Ethylene	0.9362
Ammonia	0.874889
Toluene	0.938725

The average between the R2 errors in Table 6 was calculated to evaluate the performance of the simple linear regressor error for batch 1, as follows:

$$Error_{SLRAverage} = 0.761177 \quad (10)$$

2.7. Best Regressor for the Dataset

After the performance of the three previous regressors was evaluated on the batch 1 dataset, the AdaBoost regressor was selected as the regressor that best fit the dataset because its average error was 0.8836, which was more accurate than XGBoost and Simple linear regressor, as shown in the following Table 7.

Table 7. Comparison of average errors of the regressors.

	Best Regressor for the Dataset		
Gases/Regressor	**XGBoost**	**Simple Linear Regression**	**AdaBoost**
Acetone	0.662757	0.722352	0.703059
Acetaldehyde	0.960751	0.982838	0.985634
Ethanol	0.467232	0.11206	0.680716
Ethylene	0.9824	0.9362	0.955653
Ammonia	0.988174	0.874889	0.984298
Toluene	0.994229	0.938725	992297
Average	0.842590	0.761177	0.883609

3. Results

To verify the performance of the AdaBoost regressor on the entire dataset, the following experiment evaluated the regressor's operation on the other gases, as shown in Figure 8, in which the prediction algorithm was divided into the training and testing data. This process was used to find a pattern that helps the model to predict new results with the following methodology:

3.1. First Experiment: n Batch Training and n+1 Batch Testing

The first experiment consisted of training the algorithm using the AdaBoost regressor. First, the algorithm was trained on the data from $Batch_n$, and its performance was tested on $batch_n + 1$. Therefore, the size of the dataset in each batch was a significant parameter since fewer data during training leads to a larger error, such as for batch 5, which has a total of 197 samples.

For example, for the first gas (acetone), the R2 error for the algorithm was 0.9396, which used training data from batch 6 and test data from batch 7, which showed that it has good tracking as its error was close to 1. Figure 9 shows precise signal tracking.

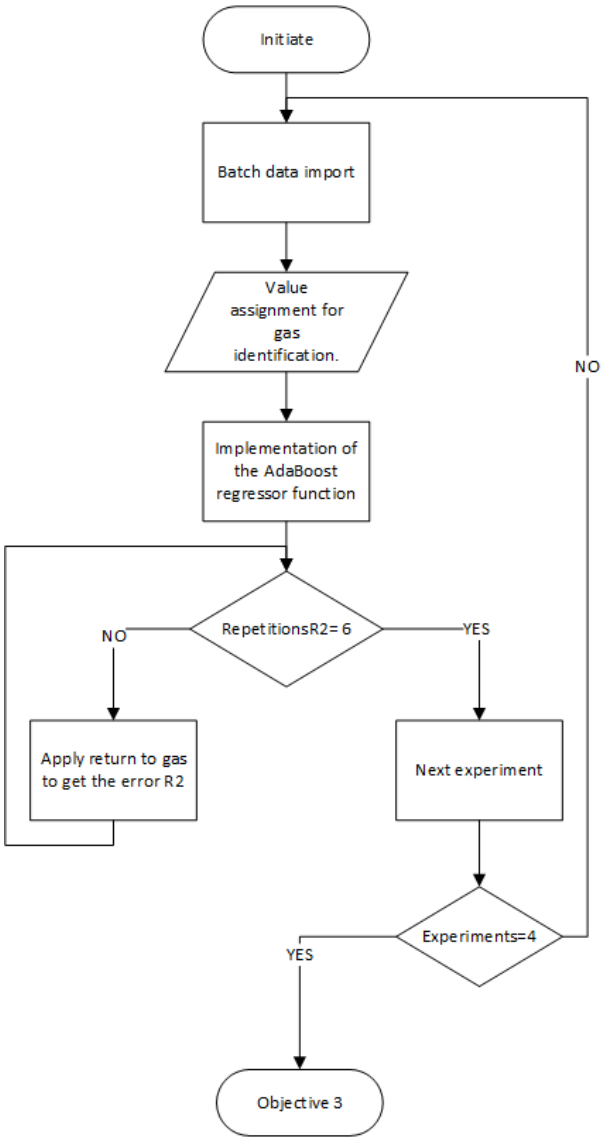

Figure 8. Methodology for checking the operation of the AdaBoost regressor under different conditions.

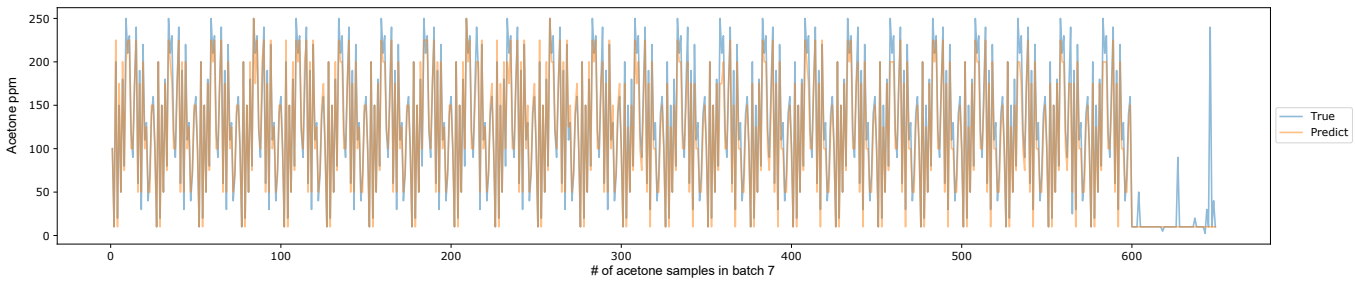

Figure 9. True vs. predicted behavior of AdaBoost regressor for acetone in batch 7 for experiment 1.

Table 8 presents the results for the entire process for the other batches and gases.

Table 8 shows the results of the evaluated error for the trained dataset in each of the six gases. However, the undefined ("-"), zero, and negative values represent a result where there were not enough samples for the training algorithm to perform a proper test over the next batch. This happened with the toluene gas, in which there was not enough data in batches 3 to 5 to train and test, which is why a second experiment was planned to ensure sufficient data of all the gases in each of the batches for a better training algorithm and test

error evaluation. In the case of batch 10, it is important to point out that although it has negative data because the sensors were off for six months, it should be considered because it is essential to reduce the drift of the sensors over time.

Table 8. *Batch_n* training and testing errors in $Batch_{n+1}$.

AdaBoost									
Gases/Batch	**1 → 2**	**2 → 3**	**3 → 4**	**4 → 5**	**5 → 6**	**6 → 7**	**7 → 8**	**8 → 9**	**9 → 10**
Acetone	−0.130729	0.150858	−0.506272	0.983796	−0.486567	0.939646	0	1	−1.92532
Acetaldehyde	0.842822	0.686795	0	−0.183673	0.57755	0.507228	0	1	−1.97581
Ethanol	0.1675	0.563065	0	1	−0.061294	0.0813233	−5.45995	−46.0663	−1.44975
Ethylene	0.63787	0.775758	0.395175	−4.4625	−0.0845528	0.617038	−11.141	0	−0.103226
Ammonia	0.837509	0.91124	0	−0.495356	−0.823205	0.795848	0.495545	0	−1.53419
Toluene	0	-	-	-	-	0.319257	0	1	−1.40275

3.2. Second Experiment: Training Batches 1–5 and Testing Batches 6–9

Due to the lack of data, experiment 1 was undesired, so the second experiment was trained in batches 1–5 and then tested in batches 6–9. This experiment allowed the algorithm to train with more data, verifying the regressor's behavior.

When programming and using the proposed methodology, the R2 error corresponding to the first gas (acetone) was 0.7824. Because it is not so close to 1, this error demonstrates that the AdaBoost regressor allows the proper tracking of data.

Figure 10 shows the tracking trajectory.

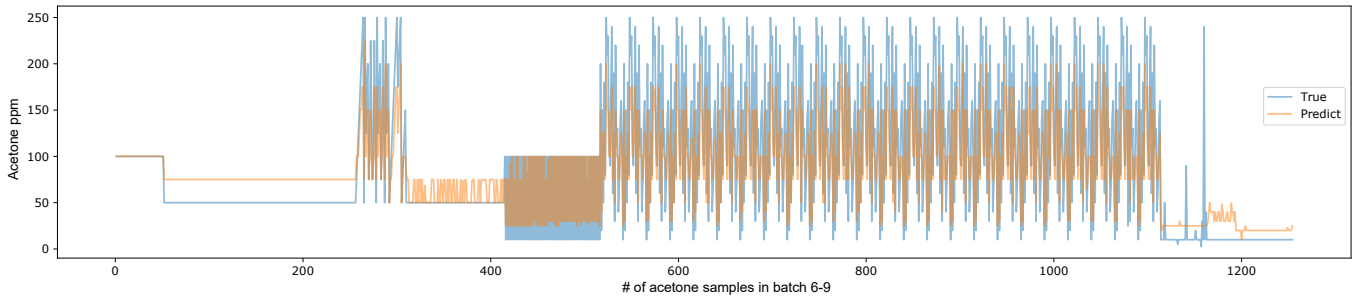

Figure 10. True vs. predicted behavior of AdaBoost regressor for acetone in batches 6–9 for Experiment 2.

Table 9 shows the results obtained for the present experiment. The algorithm's performance for the data implemented is much better than in Experiment 1 because the R2 errors obtained for each gas, since there are no negative errors and all gases could be evaluated, show that there is no problem with a lack of data in the gas samples. This experiment shows that the AdaBoost regressor has a correct function in the database with large volumes of data.

Table 9. R2 errors in $Batch_{1-5}$ training and testing in $Batch_{6-9}$.

EXPERIMENT #2	
Gases	**AdaBoost**
Acetone	0.782454
Acetaldehyde	0.670189
Ethanol	0.423060
Ethylene	0.570609
Ammonia	0.768746
Toluene	0.165012
Average	0.563345

3.3. Third Experiment: Training Batches 6–9 and Testing Batch 10

The third experiment consisted of training with batches 6–9, corresponding to nine months of recorded data, and testing it in batch 10, corresponding to one month of recorded data. The main objective was to obtain more training data, which leads to more accurate tracking. However, it should be mentioned that the data of batch 10 were much more imprecise due to the fact that there was an erroneous acquisition of the data, showing a much more significant error than in the previous experiment. For example, the R2 error for the first gas (acetone) was 0.7944, as shown in Figure 11, presenting the following tracking trajectory.

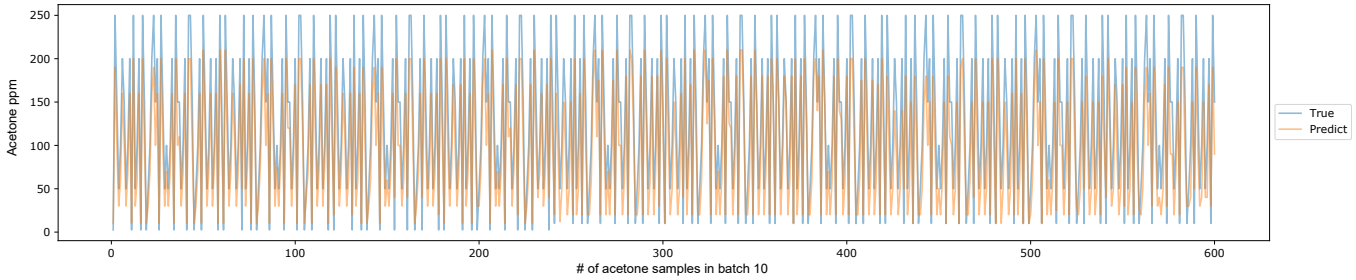

Figure 11. True vs. predicted behavior of AdaBoost regressor for acetone in batch 10 for experiment 3.

Table 10 shows the R2 errors obtained for the present experiment, but it is important to note that the behavior of the first experiment improved because the evaluated data increased, although the data for ethanol and ammonia are presented as uncertain values because of their negative values due to the conditions of batch 10 exposed in the research by Vergara et al.

Table 10. R2 errors in $Batch_{6-9}$ training and testing in $Batch_{10}$.

EXPERIMENT #3	
Gases	**AdaBoost**
Acetone	0.794446
Acetaldehyde	0.183383
Ethanol	−0.703551
Ethylene	0.235817
Ammonia	−0.586564
Toluene	0.099474
Average	0.003834

3.4. Fourth Experiment: Training Batch 1 and Testing the Other Batches

The last experiment involved training the algorithm using batch 1 and then testing it on the different batches of the dataset. In this way, it was possible to verify that it was necessary to have more training data by attaining proper tracking with a minor error.

Once programming was applied and executed, the R2 error corresponding to the first gas (acetone) in batch 2 was −0.1307. This error can be seen in Figure 12 and Table 11 in the first row of columns 1 → 2. Figure 12 shows the tracking trajectory R2 for AdaBoost in the fourth experiment, and Table 11 presents the values obtained for the entire dataset.

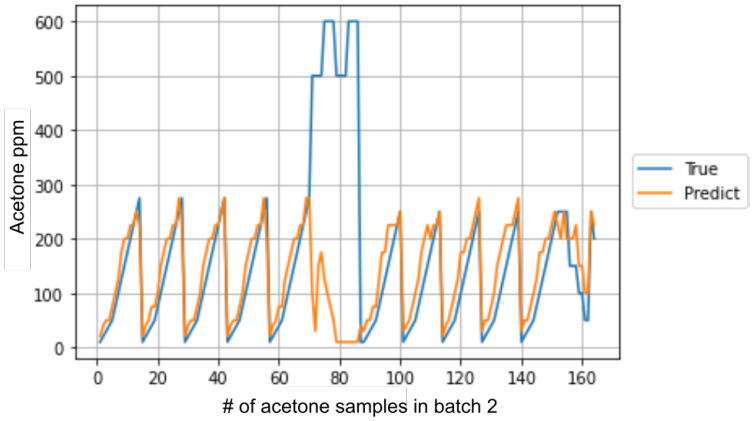

Figure 12. True vs. predicted behavior of AdaBoost regressor for acetone in batch 2 (164 measurements).

Table 11. R2 errors in $Batch_1$ training and testing in $Batch_{1+n}$.

EXPERIMENT #4									
Gases/Batch	**1 → 2**	**1 → 3**	**1 → 4**	**1 → 5**	**1 → 6**	**1 → 7**	**1 → 8**	**1 → 9**	**1 → 10**
Acetone	−0.130729	0.246876	−0.122286	−0.377315	−2.66535	0.170476	0	0	0.572529
Acetaldehyde	0.842822	−0.294713	0	0.7813	−0.098906	0.508916	0	0	0.583842
Ethanol	0.1675	−4.76936	0	0	−15.835	−2.12151	−24.41040	−2145.3	0.635629
Ethylene	0.63787	0.606361	0.402112	−3.3788	0.554814	0.745456	−2.81817	0	−0.127318
Ammonia	0.837509	0.814116	0	0.596261	0.686284	0.698325	0.108429	0	−0.150167
Toluene	0	-	-	-	0.54694	0.258964	0	0	0.521434

Table 8 shows an improvement with respect to the previous experiment, although there are still negative errors, zero errors, and errors that could not be calculated (-).

Negative and zero R2 errors are due to underfitting because of the lack of data when calculating the error.

3.5. Discussion

Comparing the performance between the results:

- From the previous tables, it can be added that the problem of the negative values is caused because of underfitting.
- In some cases, the results improved for some batches, but in other cases, the error increased. This variation is due to the number of samples evaluated in which a lower number of samples represents a lower presicion for the proposed model.
- This experiment verifies the AdaBoost regressor's performance on the dataset. In the end, it shows good monitoring and improvement of the results based on the R2 error for the entire dataset.

4. AdaBoost Regressor with KLIEP Domain Adaptation Methodology

After determining the regressor, it was necessary to apply the domain adaptation method (DA). DA is a method to ensure the proper function of a model over multiple source distributions when trained over a different but related distribution. Combining AdaBoost and DA is expected to reduce the drift error.

Mathelin et al. [35] explained the domain adaptation method and divided it into three main strategies:

1. Feature-based containment methods that perform feature transformation;
2. The instance-based methods with the implementation of reweighting techniques;
3. Parameter-based proposal methods for adapting pre-trained models to novel observations.

ADAPT is a python library that implements different DA methods, including those that are function-based, instance-based, and parameter-based [35]. Mathelin et al. ex-

plained how the DA method's objective is to use the instance-based module of the ADAPT library to correct sample bias and assume that the distribution source and target share the same support in the input space. This method reweights the source instances to correct the difference between the source (source domain S) and destination (target domain T) distributions [25].

The process of the DA instance-based method is shown in Figure 13.

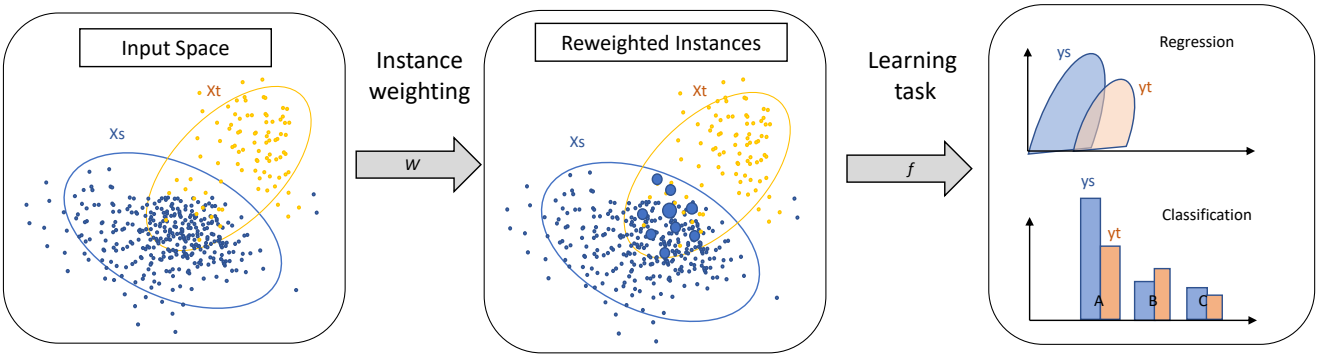

Figure 13. Strategy based on DA instances method. [35].

In this study, the KLIEP algorithm (Kullback–Leibler importance estimation) was used for the instance-based method. This algorithm aims to correct the difference between input distributions of the source and destination domains, finding a reweighting of source instances that minimizes the Kullback–Leibler divergence between the source (source—S) and destination (target—T) distributions [31].

The proposed methodology to implement the KLIEP based DA method with AdaBoost is shown in Figure 14.

The methodology was similar to the validation of the AdaBoost regressor. In this case, the domain adaptation method was also applied in addition to the regressor.

The experiments performed for this section were the same ones performed in the Adaboost regressor behavior validation section to compare the results obtained in each experiment, which is a way to verify the behavior of AD.

4.1. n Batch Training and n+1 Batch Testing with DA

To validate the regressor, it was trained using $batch_n$ and evaluated on $batch_{n+1}$. The same process of implementing the database and recognizing the gases by the program was carried out. Subsequently, it was necessary to implement the AdaBoost regressor and apply the KLIEP algorithm, as shown in Figure 13.

The R2 error corresponding to the first gas was obtained in $1 \rightarrow 2$, which was 0.9988, as shown in Figure 15.

This process was repeated for each gas and batch. Table 12 show the R2 errors for each gas and batch.

Table 12. R2 errors in $Batch_1$ training and testing in $Batch_{1+n}$.

DOMAIN ADAPTATION EXPERIMENT 1									
Gases/Batch	**1→ 2**	**2→ 3**	**3→ 4**	**4→ 5**	**5→ 6**	**6→ 7**	**7→ 8**	**8→ 9**	**9→ 10**
Acetone	0.998806	0.955239	0.918737	1	0.856410	0.968663	0.956009	1	1
Acetaldehyde	0.970722	0.910951	0.926854	1	1	0.908055	0.747359	1	1
Ethanol	0.997943	1	0.991445	1	1	0.942600	0.340262	0.971570	1
Ethylene	0.995803	1	0.998587	1	1	1	0.872420	1	1
Ammonia	0.986907	0.999214	0.991036	1	1	0.948215	0.947587	0.457177	1
Toluene	0.954738	-	-	-	-	0.978532	0.040563	1	1

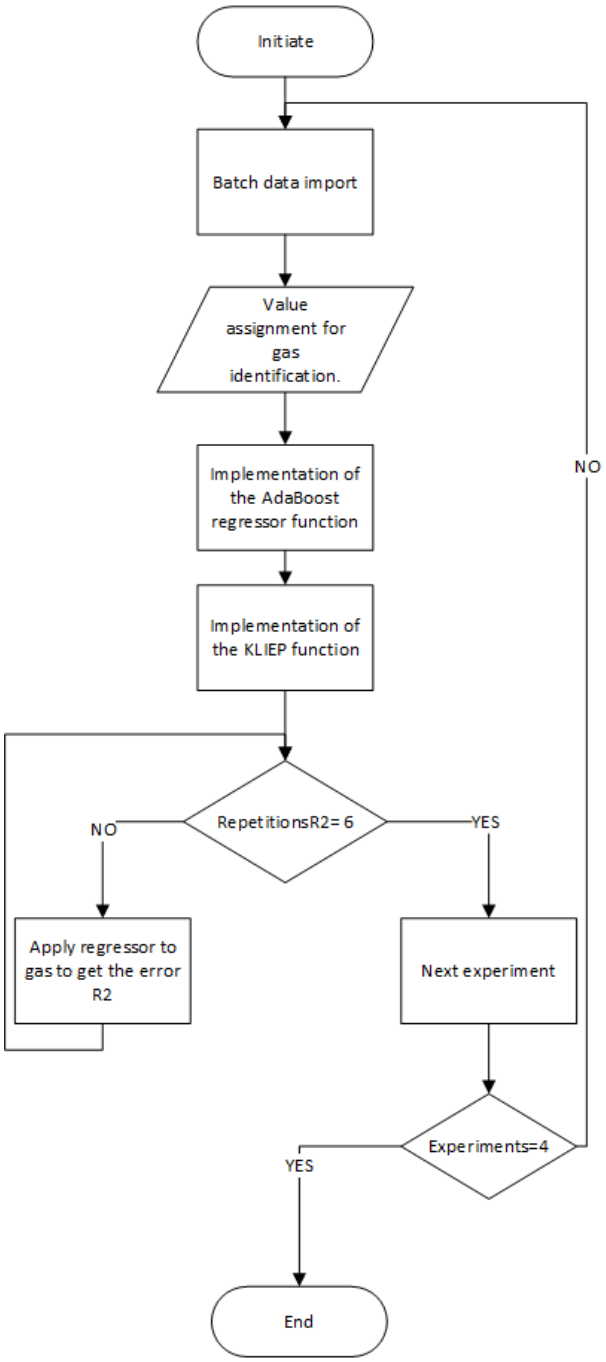

Figure 14. Methodology to assess the operation of the AdaBoost regressor with the KLIEP DA method under different conditions.

4.2. Training Batches 1–5 and Testing Batches 6–9 with DA

The following test involves training the regressor in matches 1 to 5 and testing using batches 6 to 9; this method shows an R2 error of 0.46. Figure 16 shows the signal tracking for the first gas.

Table 13 shows the R2 errors for the missing gases.

4.3. Training Batches 6–9 and Testing Batch 10 with DA

The final test involved training the regressor using batches 6 to 9 and testing with batch 10. The R2 error for acetone was 0.97. This result shows a decrease in sensor drift. The R2 error for acetone is shown in Figure 17.

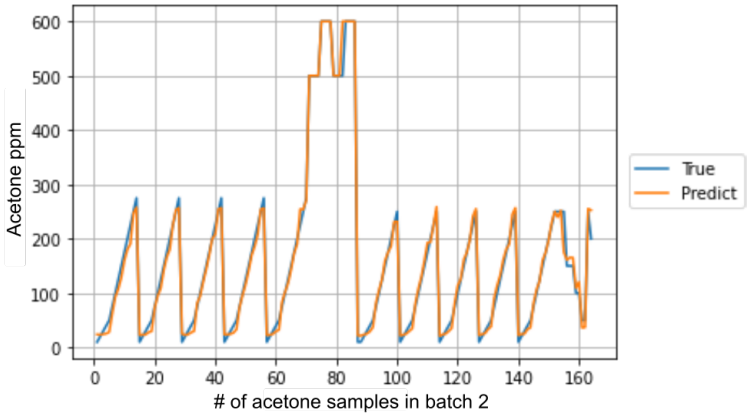

Figure 15. True vs. predicted behavior of AdaBoost regressor with KLIEP method for acetone in batch 2 (164 measurements).

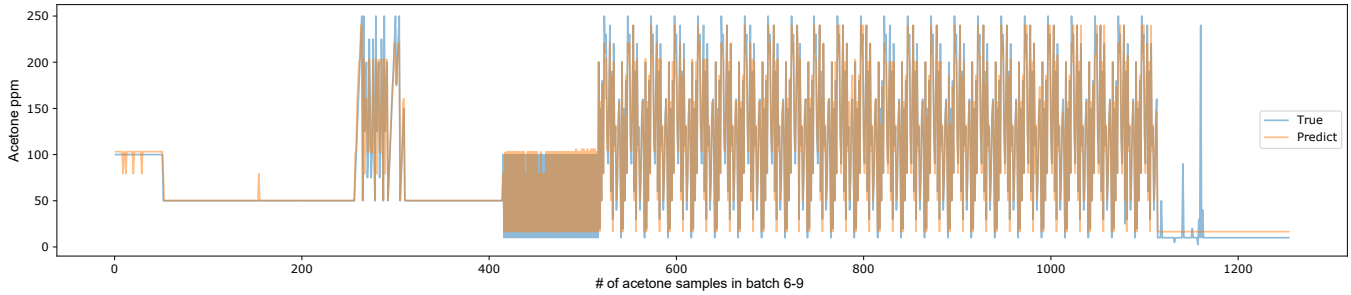

Figure 16. True vs. predicted behavior of AdaBoost regressor with KLIEP method for acetone in batches 6–9 for Experiment 2.

Table 13. R2 errors in application of the AdaBoost regressor with KLIEP for testing in $Batch_{6-9}$.

DOMAIN ADAPTATION EXPERIMENT 2	
Gases	**AdaBoost/DA**
Acetone	0.466610
Acetaldehyde	0.922668
Ethanol	0.980563
Ethylene	0.974383
Ammonia	0.892407
Toluene	0.956007
Average	0.865440

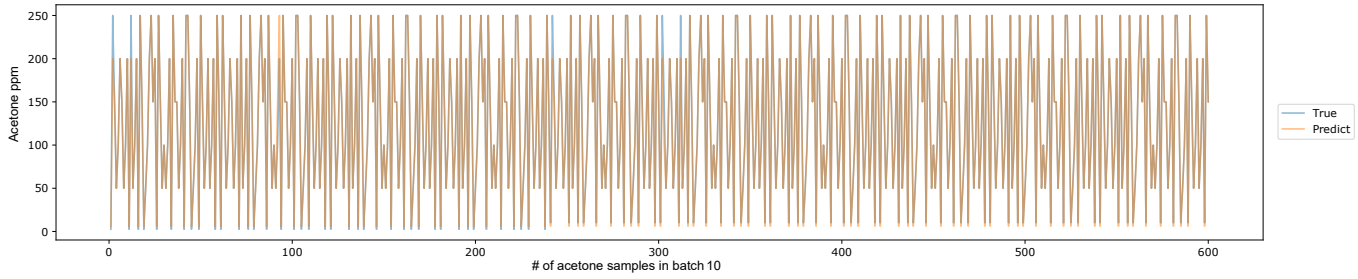

Figure 17. True vs. predicted behavior of Adaboost regressor with KLIEP method for acetone in batch 10 for experiment 3.

Table 14 shows the R2 error when repeating the process for each gas.

Table 14. R2 errors in the application of the AdaBoost regressor with KLIEP for testing in $Batch_{10}$.

DOMAIN ADAPTATION EXPERIMENT 3	
Gases	**AdaBoost/DA**
Acetone	0.968071
Acetaldehyde	0.821917
Ethanol	0.528604
Ethylene	0.907779
Ammonia	0.905870
Toluene	0.632908
Average	0.794192

From the application of AdaBoost with KLIEP DA, it is possible to conclude an improvement in the drift, which is supported by R2 errors being closer to 1.

- From the first experiment using DA, it can be seen that there are no negative errors, as shown in Tables 12 and 8. Several errors of 1% were obtained, which is the best-calculated error, with only six errors less than 88%. This percentage demonstrates the proper functioning of the applied domain adaptation KLIEP algorithm. This behavior was constant among the batches.
- The validation of the correct functioning of the second experiment with AD was carried out directly by comparing Table 13 with Table 9. There was an improvement in the errors calculated with the regressor. In Table 9, the average error was 56.3%, but Table 13 shows an average error of 86.5%. These results show how the instancebased domain adaptation model helped improve the accuracy of the AdaBoost regressor.
- To verify the methods of the third experiment, the comparison of Table 14 was carried out with Table 10. The difference in the average R2 error was 79.4% and 0.003%, respectively. Additionally, no negative error was obtained.

5. Conclusions

This study developed a methodology to improve gas quantification in an electronic nose sensor array. The methodology was based on the application of the KLIEP domain adaptation procedure using an AdaBoost regressor as an estimator. The methodology was satisfactorily validated using a 13,910-measurement dataset of an electronic nose sensor array taken for 36 months. The results obtained in the experiments show a high average value of R2 error. The results also shows that the more data there are in the source domain, the better the behavior of the R2 error is in the target domain.

The gas quantification was addressed in this study through a machine learning data processing approach. First, it was necessary to select a regressor; AdaBoost showed the best R2 error compared with XGBoost and simple linear regression. Second, the AdaBoost estimator was paired with the KLIEP method. According to the results, after applying the developed methodology based on AdaBoost and KLIEP, it improved the average R2 error obtained in different test sets. This can be evidenced in experiment 3, where batches 6–9 composed the source data, and batch 10 composed the target data, yielding an average R2 error value of 0.794 ppm. In contrast, when only using the AdaBoost estimator, the average R2 error was 0.00383 ppm. Due to the combination of the KLIEP with the AdaBoost methods, it was possible to extend the useful life of an electronic nose, and the implementation of the developed methodology can reduce costs by preventing sensors from recalibrating.

As future work, different regressor estimators and domain adaptation methods will be tested to find their behavior related to R2 error. Additionally, to accomplish online monitoring, an ETL process (extract, transform and load) in a web cloud server can be used to store the information in a database so that the experiments are constantly updated, generating more training data. Furthermore, the developed methodology will be tested with new data from another electronic nose sensor array.

Author Contributions: All authors contributed to the development of this work; specifically, their contributions are as follows: Conceptualization, J.X.L.-M. and D.A.P.G.; Methodology, D.A.P.G., M.C.C.C and J.X.L.-M.; Software, D.A.P.G.; Validation, D.N.T.B., J.A.E.G. and J.F.N.S.; Data Curation, D.A.P.G.; Writing—Original Draft Preparation, D.A.P.G., M.C.C.C. and J.X.L.-M.; Writing—Review & Editing, D.N.T.B., J.A.E.G. and J.F.N.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universidad de San Buenaventura sede Bogota with the financing of the research project ID = FI-016-006, entitled: Sistema de Navegación Inercial y Visual para Robot Móviles en Espacios Confinados. and The APC was funded by Universidad de San Buenaventura sede Bogota.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank to Universidad de San Buenaventura sede Bogota for the financing of the research project ID = FI-016-006 entitled: Sistema de Navegación Inercial y Visual para Robot Móviles en Espacios Confinados.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

KLIEP	Kullback–Leibler importance estimation procedure
PPM	Parts per million
LCV	Likelihood cross validation
DA	Domain adaptation
XGBoost	Extreme gradient boosting

References

1. Leon-Medina, J.X.; Parés, N.; Anaya, M.; Tibaduiza, D.A.; Pozo, F. Data Classification Methodology for Electronic Noses Using Uniform Manifold Approximation and Projection and Extreme Learning Machine. *Mathematics* **2021**, *10*, 29. [CrossRef]
2. Rodriguez-Lujan, I.; Fonollosa, J.; Vergara, A.; Homer, M.; Huerta, R. On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemom. Intell. Lab. Syst.* **2014**, *130*, 123–134. [CrossRef]
3. Del Valle, M. Sensor arrays and electronic tongue systems. *Int. J. Electrochem.* **2012**, *2012*. [CrossRef]
4. Ye, Z.; Liu, Y.; Li, Q. Recent progress in smart electronic nose technologies enabled with machine learning methods. *Sensors* **2021**, *21*, 7620. [CrossRef] [PubMed]
5. Zhang, L.; Zhang, D. Domain adaptation extreme learning machines for drift compensation in E-nose systems. *IEEE Trans. Instrum. Meas.* **2014**, *64*, 1790–1801. [CrossRef]
6. Vergara, A.; Vembu, S.; Ayhan, T.; Ryan, M.A.; Homer, M.L.; Huerta, R. Chemical gas sensor drift compensation using classifier ensembles. *Sens. Actuators B Chem.* **2012**, *166*, 320–329. [CrossRef]
7. Dadkhah, M.; Tulliani, J.M. Nanostructured Metal Oxide Semiconductors towards Greenhouse Gas Detection. *Chemosensors* **2022**, *10*, 57. [CrossRef]
8. Leon-Medina, J.X.; Pineda-Muñoz, W.A.; Burgos, D.A.T. Joint distribution adaptation for drift correction in electronic nose type sensor arrays. *IEEE Access* **2020**, *8*, 134413–134421. [CrossRef]
9. Dong, W.; Zhao, J.; Hu, R.; Dong, Y.; Tan, L. Differentiation of Chinese robusta coffees according to species, using a combined electronic nose and tongue, with the aid of chemometrics. *Food Chem.* **2017**, *229*, 743–751. [CrossRef]
10. Gu, D.C.; Liu, W.; Yan, Y.; Wei, W.; Gan, J.h.; Lu, Y.; Jiang, Z.L.; Wang, X.C.; Xu, C.H. A novel method for rapid quantitative evaluating formaldehyde in squid based on electronic nose. *LWT* **2019**, *101*, 382–388. [CrossRef]
11. Blanco-Rodríguez, A.; Camara, V.F.; Campo, F.; Becherán, L.; Durán, A.; Vieira, V.D.; de Melo, H.; Garcia-Ramirez, A.R. Development of an electronic nose to characterize odours emitted from different stages in a wastewater treatment plant. *Water Res.* **2018**, *134*, 92–100. [CrossRef]
12. Kiani, S.; Minaei, S.; Ghasemi-Varnamkhasti, M.; Ayyari, M. An original approach for the quantitative characterization of saffron aroma strength using electronic nose. *Int. J. Food Prop.* **2017**, *20*, S673–S683. [CrossRef]
13. Viejo, C.G.; Fuentes, S.; Godbole, A.; Widdicombe, B.; Unnithan, R.R. Development of a low-cost e-nose to assess aroma profiles: An artificial intelligence application to assess beer quality. *Sens. Actuators B Chem.* **2020**, *308*, 127688. [CrossRef]

14. Han, F.; Huang, X.; Teye, E.; Gu, F.; Gu, H. Nondestructive detection of fish freshness during its preservation by combining electronic nose and electronic tongue techniques in conjunction with chemometric analysis. *Anal. Methods* **2014**, *6*, 529–536. [CrossRef]
15. Xu, M.; Wang, J.; Zhu, L. The qualitative and quantitative assessment of tea quality based on E-nose, E-tongue and E-eye combined with chemometrics. *Food Chem.* **2019**, *289*, 482–489. [CrossRef]
16. Du, D.; Wang, J.; Wang, B.; Zhu, L.; Hong, X. Ripeness prediction of postharvest kiwifruit using a MOS e-nose combined with chemometrics. *Sensors* **2019**, *19*, 419. [CrossRef]
17. Liu, H.; Li, Q.; Gu, Y. A multi-task learning framework for gas detection and concentration estimation. *Neurocomputing* **2020**, *416*, 28–37. [CrossRef]
18. Bakiler, H.; Güney, S. Estimation of Concentration Values of Different Gases Based on Long Short-Term Memory by Using Electronic Nose. *Biomed. Signal Process. Control* **2021**, *69*, 102908. [CrossRef]
19. Fonollosa, J.; Sheik, S.; Huerta, R.; Marco, S. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sens. Actuators B Chem.* **2015**, *215*, 618–629. [CrossRef]
20. Monroy, J.G.; Lilienthal, A.J.; Blanco, J.L.; Gonzalez-Jimenez, J.; Trincavelli, M. Probabilistic gas quantification with MOX sensors in Open Sampling Systems - A Gaussian Process approach. *Sens. Actuators B Chem.* **2013**, *188*, 298–312. [CrossRef]
21. Fonollosa, J.; Fernandez, L.; Gutiérrez-Gálvez, A.; Huerta, R.; Marco, S. Calibration transfer and drift counteraction in chemical sensor arrays using Direct Standardization. *Sens. Actuators B Chem.* **2016**, *236*, 1044–1053. [CrossRef]
22. Cho, J.H.; Kim, Y.W.; Na, K.J.; Jeon, G.J. Wireless electronic nose system for real-time quantitative analysis of gas mixtures using micro-gas sensor array and neuro-fuzzy network. *Sens. Actuators B Chem.* **2008**, *134*, 104–111. [CrossRef]
23. Zhang, L.; Tian, F. Performance study of multilayer perceptrons in a low-cost electronic nose. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 1670–1679. [CrossRef]
24. Zhang, D.; Liu, J.; Jiang, C.; Liu, A.; Xia, B. Quantitative detection of formaldehyde and ammonia gas via metal oxide-modified graphene-based sensor array combining with neural network model. *Sens. Actuators B Chem.* **2017**, *240*, 55–65. [CrossRef]
25. Sugiyama, M.; Nakajima, S.; Kashima, H.; Buenau, P.; Kawanabe, M. Direct importance estimation with model selection and its application to covariate shift adaptation. *Adv. Neural Inf. Process. Syst.* **2007**, *20*, 1–8.
26. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [CrossRef]
27. Pan, S.J.; Tsang, I.W.; Kwok, J.T.; Yang, Q. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Netw.* **2010**, *22*, 199–210. [CrossRef]
28. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
29. Xu, F.; Yu, J.; Xia, R. Instance-based domain adaptation via multiclustering logistic approximation. *IEEE Intell. Syst.* **2018**, *33*, 78–88. [CrossRef]
30. UCI-Machine-Learning-Repository. Gas Sensor Array Drift Dataset at Different Concentrations Data Set. 2013. Available online: https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations (accessed on 5 October 2022).
31. de Mathelin, A.; Deheeger, F.; Richard, G.; Mougeot, M.; Vayatis, N. Adapt Instance Based KLIEP. 2020. Available online: https://adapt-python.github.io/adapt/generated/adapt.instance_based.KLIEP.html (accessed on 5 October 2022).
32. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
33. Wes McKinney. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; pp. 56–61. [CrossRef]
34. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
35. De Mathelin, A.; Deheeger, F.; Richard, G.; Mougeot, M.; Vayatis, N. Adapt: Awesome domain adaptation python toolbox. *arXiv* **2021**, arXiv:2107.03049.
36. Kumar, A. Mean Squared Error or r-Squared. 2022. Available online: https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/ (accessed on 5 October 2022).