Received: April 21, 2023. Revised: June 10, 2023.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Data Augmentation Technique Using Two Step SMOTE for Electronic-nose Signal in Breath Ketone Level Detection

Dhiza Wahyu Firmansyah[1] **Riyanarto Sarno[1]***

[1]*Departement of Informatics, Faculty of Intelligent Electrical and Informatics Technology, Institut Technology Sepuluh Nopember (ITS) Sukolilo, Surabaya, Indonesia*
* Corresponding author's Email: riyanarto@if.its.ac.id

Abstract: Breath acetone concentrations were found to be correlated with blood ketone levels. Based on this evidence, predicting blood ketone levels using breath analysis and machine learning (ML) becomes possible. Nevertheless, a good ML model requires a large amount of training data. Under certain conditions, it is difficult to collect large amounts of data such as during the Covid-19 pandemic. To overcome this problem, we propose an augmentation technique to extend the number of training datasets using two step synthetic minority oversampling (SMOTE). The first step was to increase the amount of training data by combining it with synthetic data, while the second step was to balance the data at each ketone level. The strategy for using SMOTE with regression was further explained since this study aims to predict ketone levels with numerical output values and SMOTE is typically used in classification cases. The proposed method was evaluated by entering the data into several ML methods such as deep neural network regression (DNN-R), linear regression (ML-R), ransac regression (RC-R), K-nearest neighbour regression (KNN-R), decision tree regression (DT-R), random forest regression (RF-R), Ada boost regression (AD-R), Gradient boost regression (GB-R) and XG-boost regression (XGB-R). Based on the test results, when compared without the proposed method, an increase in accuracy was obtained on DNN-R, ML-R, RC-R, KNN-R, DT-R, RF-R, AB-R, GB-R, and XGB-R by 0.958%, 9.51%, 35.74%, 18.133%, 8.236%, 11.348, 9.47%, 5.093%, and 11.264% respectively.

Keywords: Electronic-nose, Data augmentation, Breath ketone level, Machine learning, Gas sensor.

1. Introduction

The use of Electronic-nose for breath analysis has recently been widely implemented in the health sector due to its advantage as a non-invasive method that does not require penetration into the patient's body. Thus, Electronic-nose is more practical and does not hurt the patient. One of the topics that have been extensively studied is the case study of diabetes and body metabolism. On the topic, several studies have implemented breath analysis for detecting diabetes [1, 2], blood sugar levels [3] and ketone levels [4, 5]. Research on the relationship between breath acetone values and blood ketones has been currently quite popular because it is not only used for detecting ketone levels in complications of diabetic ketoacidosis (DKA) but also widely used by ketogenic dieters.

The previous study conducted by [6] examined the relationship between the results of breath acetone measurements from gas sensors and blood ketone values. The data were collected from 35 patients with diabetes to detect patients at risk of developing diabetic ketoacidosis (DKA). The data of the patients were analyzed using the regression equation and correlation coefficient and yielded a correlation coefficient of 0.828. This value indicates a strong correlation between breath acetone and blood ketone levels. The other previous study by [7] also tested the relationship between breath acetone content and blood ketone values. This study collected data from 72 patients with type 1 diabetes mellitus and 9 patients with DKA. The results of the study showed a significant correlation between breath acetone and blood ketone levels. Based on these results, breath acetone levels can be used to categorize the blood ketone levels of the patients. The level categories

include the normal, high, and risky category. A strong correlation found in the previous studies [6, 7] has provided an opportunity to implement machine learning in detecting blood ketone levels using breath acetone as input.

From the literature search, the authors have not found the use of machine learning in detecting blood ketone levels using breath acetone from Electronic-nose as an input. Nevertheless, some studies were conducted in a similar area as the present studies by implementing machine learning with the input of breath acetone of electroninc-nose data to detect diabetes and Blood sugar levels. For example, recent studies by [1, 2] used the machine learning method to detect diabetic and non-diabetic patients using input data from Electronic-nose signals on the patient's breath. These studies produced a high accuracy, reaching 90% in the study [1] and 80% in the study [2]. The other studies conducted by [8, 9, 10, 11] used the machine learning method to predict blood sugar levels using Electronic-nose data from breath analysis. In the study [8] correlation coefficient was 0.996, while in the study [9] R^2Score was 0.081. Furthermore, in the study [10] the accuracy was 74.76%, and in the study [11] correlation coefficient was 0.6982. In general, these studies show that machine learning is effective to use for Electronic-nose signals.

Despite its promising utility, machine learning requires a large number of training data at the model training stage. The larger the number of training data, the higher the statistical power in pattern recognition. However, collecting data under certain conditions such as the Covid-19 pandemic condition is highly challenging because it is difficult to obtain a large number of data on patient breath. The limited number of data can reduce the total data in the machine learning training process which in turn may result in decreased accuracy of the prediction. Theoretically, the number of datasets can be expanded by augmenting synthetic data based on the patterns of the existing data. One of the most widely used data augmentation techniques is SMOTE.

SMOTE oversampling is typically used to deal with data imbalance issues in classification cases. Data on the minority class were oversampled using synthetic data generated by SMOTE. One of the strengths of SMOTE is its low complexity, which makes it a practical method to implement. The previous study conducted by [12] implemented SMOTE to Electronic-nose data to balance minority class in fruit disease detection. The use of synthetic data from SMOTE increased the accuracy by 3.98% and F1-Score by 0.042. These findings indicate the effectiveness of SMOTE to implement in Electronic-nose data. In this study, the use of SMOTE solely focused on balancing the number of data in the minority class and did not focus to increase the total amount of data. In addition, the use of SMOTE in the regression case study has not been explained in the literature.

Based on the background described above, this study proposes a machine learning-based breath ketone level detection method with a small number of datasets. The main contributions of the present studies are as follows:

(i) explaining the scenario of implementing SMOTE in Machine learning to predict based ketone level from Electronic-nose data using the regression, whereas SMOTE has generally been used for classification;

(ii) improving SMOTE data generation scenarios to increase the amount of training data in predicting ketone level from Electronic-nose data using the regression.

In general, the strategy to meet contribution (i) involved two main steps. Firstly, the numeric output data were converted into categorical form. Secondly, SMOTE was applied using both feature data and categorical output data. Meanwhile, contribution (ii) was fulfilled using two steps. The first step involved duplicating the training dataset, while the second step focused on balancing the number of datasets at each ketone level. To determine their performance, the data from the proposed method were tested using various machine learning methods.

The general sequences of the present study consisted of hardware explanation, dataset retrieval, raw data pre-processing, data preparation, data augmentation, machine learning testing, and evaluation.

Overall, the structure of this paper is organized as follows: Section 1 discusses the research background, section 2 explains the proposed method, section 3 presents the results of this study, and section 4 contains the conclusions of the research that was conducted.

2. Proposed method

This section describes the sequence of the proposed method in the present study. It consists of hardware explanation, data acquisition, data pre-processing, data preparation, machine learning testing, and evaluation. The sequence of the method can be seen in Fig. 1.

2.1 Hardware explanation

This study used a self-developed Electronic-nose device that consists of four Acetone sensitive

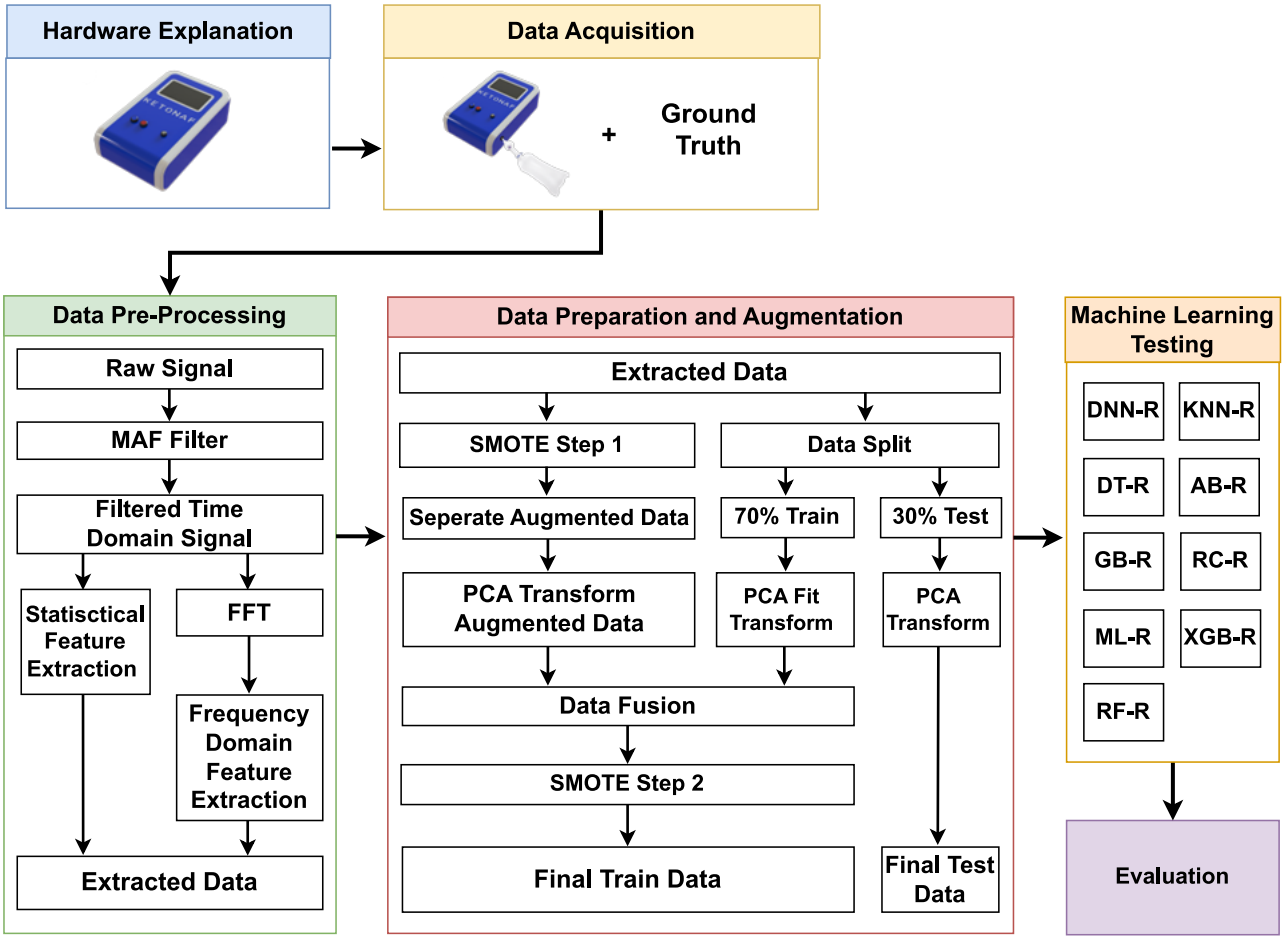

Figure. 1 Proposed method workflow

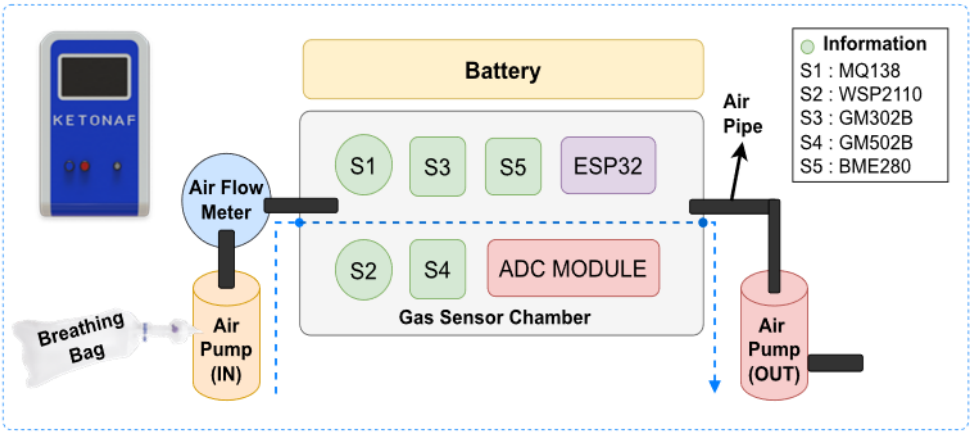

Figure. 2 Electronic-nose hardware configuration

gas sensors, namely WSP2110 [13], MQ138 [14], GM302B [15] and GM502B [15]. In addition, the device also has two pumps that serve as air suction into the chamber and remove air from the chamber. Furthermore, it also has HEPA filter connected to the intake air pipe. For collecting the patient's breath sample, a breathing bag is provided to collect the blown air.

The breathing bag is connected to a HEPA filter which is then sucked into the chamber and read by four gas sensors. The breathing bag is connected to a HEPA filter which is then sucked into the chamber and read by four gas sensors. The sensor is read by the Esp32 microcontroller which has been integrated with all components. An illustration of the hardware developed can be seen in Fig. 2.

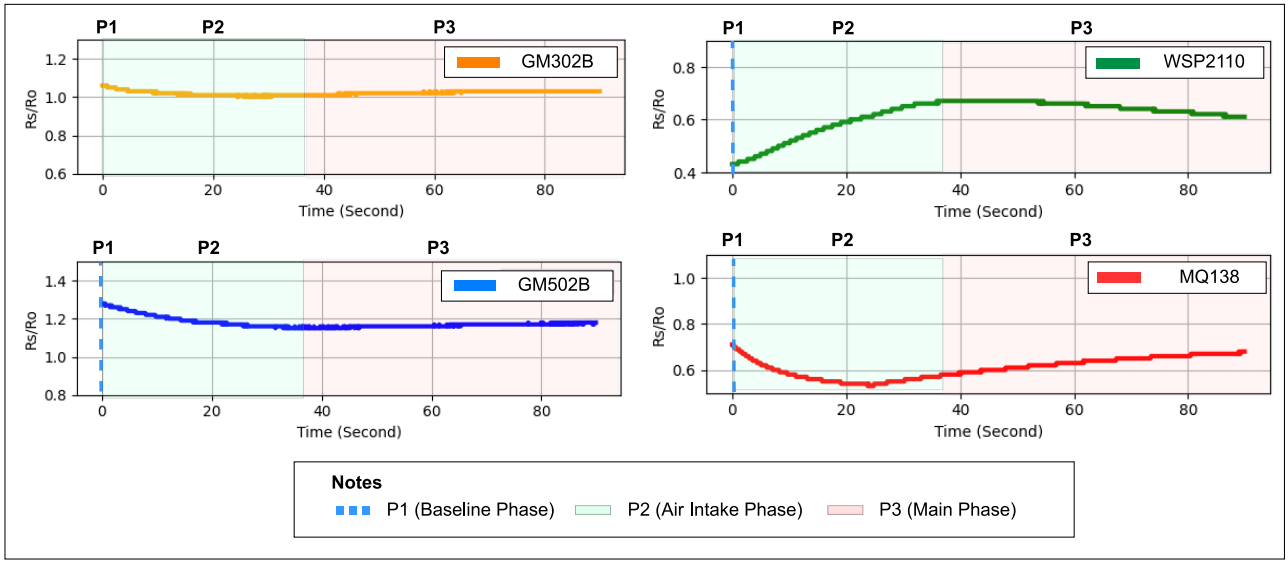

Figure. 3 Electronic-nose signal representation

The reading of the sensor value begins with reading the ADC (analog to digital converter) and subsequently calculating the *VRL* using Eq. (1). *Vref* is 5v, while *R* is 16. Subsequently, the *Rs* value can be calculated using Eq. (2). *Vc* is 5v (sensor input voltage) and *Rl* value is according to the datasheet reference for each sensor. The next step is to calibrate the sensor by finding the value of *Ro*. *Ro* itself is the value of *Rs* in fresh air condition. Furthermore, the *Rs/Ro* value was determined where this value was correlated with the acetone gas concentration value according to the sensor datasheet reference.

$$Vrl = ADC\ \frac{Vref}{(2^R - 1)} \tag{1}$$

$$Rs = \frac{Vc - Vrl}{Vrl} \times Rl \tag{2}$$

The gas sensor reading flow refers to research by [16] and sensor datasheets. In addition, the BME280 sensor produced temperature, humidity, and air pressure values. Meanwhile, the flow meter yielded total milliliters of intake air and air velocity values. Data collection lasted for 90 seconds at a sampling frequency of 10 Hz or 10 data per second. Thus, during a single-session data collection, each sensor produced 900 samples. The data was represented to a time domain signal which can be seen in Fig. 3. The signal consists of 3 phases, namely Phase 1 (P1): baseline value, Phase 2 (P2): when the air was sucked in, and Phase 3 (P3): the main signal under stable conditions.

2.2 Data acquisition

The data collection was conducted at the naval central hospital of surabaya (RSPAL). Data was obtained by collecting breath samples from patients using the developed device and ground truth values. The ground truth uses the "Ketoscan" breath ketone meter that is already available on the market. "Ketoscan" can measure ketone values from level 0 to 12 based on breath acetone concentration in part per million (ppm). Level 0 is a condition where ketones are not detected, while level 12 is a condition of Ketoacidosis [17]. Data collection was performed in the outpatient unit of the hospital with a random sampling method. The results of data collection consisted of data from 100 patients with ketone levels varying from 1 to 9. Only a small amount of data can be collected because it was difficult to take breath data during the Covid-19 period. Many patients were afraid to take breath data.

2.3 Signal pre-processing

Signal pre-processing was conducted to process the raw signal data obtained in data acquisition. Pre-processing began with removing the noise in the signal that can lead to data bias and low data quality. Noise reduction was performed using the moving average filter (MAF).

MAF is a filter method that calculates the moving average based on a specified point value, denoted as M. The greater the M value, the smoother the results of the filter. MAF filters have been extensively used in signal processing due to their simple calculations [18]. MAF filter can be done using Eq. (3). In this equation, y is the output signal after filtering, while i is the index value of the data in each iteration. For example, in Eq. (4), a filter calculation was performed from the 0th index signal data with a point value of M = 3. After filtering, the

signal was converted into the frequency domain using the Fast Fourier Transform (FFT) in the subsequent step.

$$y[i] = \frac{1}{M} \sum_{i=0}^{M-1} x\,[i+j] \quad (3)$$

$$y[0] = \frac{x[0]+x[1]+x[2]}{M} \quad (4)$$

Conversion to the frequency domain aimed to expand signal variations to the input features [19]. This conversion process was conducted using Eq. (5). In this equation, Sf represents the signal of the frequency domain, N denotes the number of samples on the signal, Sn represents signals on the time domain and $e^{-\frac{i2\pi}{N}fn}$ is a constant of a signal [20].

$$Sf = \sum_{n=0}^{N-1} Sn.\, e^{-\frac{i2\pi}{N}fn} \quad (5)$$

From the previous process, a filtered signal with a time domain and frequency domain was obtained. Thus, the extraction process on these two kinds of signals can be conducted. Signals in the time domain were extracted with statistical parameters: average (Avg), standard deviation (Std), skewness, kurtosis, minimum value (Min) dan maximum value (Max). Meanwhile, signals in the frequency domain were extracted by calculating the average energy (AE) and maximum energy (ME). The average can be calculated using Eq. (6). In this equation, n entails the number of data points of the signal and i entails the index data. Meanwhile, xi represents the value of signal data on each index.

$$Avg = \frac{\sum_{i=0}^{n} xi}{n} \quad (6)$$

Std can be calculated using Eq. (7). In the equation, $\bar{x}$ entails the average value of the signal data.

$$Std = \frac{\sum_{i=1}^{n} (xi-\bar{x})^2}{n} \quad (7)$$

$Skewness$ can be calculated using Eq. (8). In this equation, σ is the value of standard deviation. Furthermore, $Kurtosis$ can be determined using Eq. (9).

$$Skewness = \frac{1}{n\sigma^3} \sum_{i=0}^{n} (x_i - \bar{x})^3 \quad (8)$$

$$Kurtosis = \frac{1}{n\sigma^4} \sum_{i=1}^{n} (x_i - \bar{x})^4 \quad (9)$$

AE can be calculated using Eq. (10). In this Equation $F(i)$ denotes the value signal on the frequency domain. Furthermore, the value of ME was calculated by determining the highest value of the signal of the frequency domain.

$$AE = \frac{\sum_{i=0}^{n-1} F(i)^2}{n} \quad (10)$$

After feature extraction was performed, the next stage was data preparation and augmentation processes.

2.4 Data preparation and augmentation

This Sub-section explains data preparation and Augmentation" which contains a major contribution to the present study. In the initial stages data augmentation process was performed using the synthetic minority oversampling (SMOTE).

SMOTE is a standard method for balancing the number of minority class data in the classification case. Data on the minority class is extended by generating synthetic data. The process of synthetic data generation is performed using the K-nearest neighbor principle. The standard SMOTE workflow consists of three main stages. To illustrate, there are 10 data in class A (Majority) and 5 data in class B (Minority). In the first stage, data in the minority class B were selected randomly as symbolized by (Xi). Subsequently, the difference between the data and K-nearest neighbors $(\hat{X}_k)$ is calculated. The nearest neighbor is determined by selecting the closest distance between (Xi) and each data in class B using the Euclidean distance. In the second step, the difference values are multiplied by a random number (δ) ranging from 0 to 1 [12]. In the third step, the results of the multiplication operation in stage 2 were added with (Xi) to produce a new synthetic data $xSynthetic$. The synthetic data in class B is added to increase the amount of data. The first to the third step is iterated for other sample data in class B until the required data is sufficient. Data is considered sufficient if the amount of data in the minority class is equal to the amount of data in the majority class. The calculation process in the first to third stages can be represented by Eq. (11).

$$xSynthetic = Xi + \left(\hat{X}_k - Xi\right) \times \delta \quad (11)$$

Data generation on the standard SMOTE is conducted in a classification case with the categorical output ($yCat$). $yCat$ is a class value that can be used as a boundaries in the data generation process.

Algorithm 1: Convert $yNum$ value to $yCat$

Input : Y value in numeric $yNum$
Output : Y value in categorical $yCat$
Step :
if ($yNum$ >= 0) and ($yNum$ <= 0.9) : **return** 0
if ($yNum$ >= 0.9) and ($yNum$ <= 1.9) : **return** 1
if ($yNum$ >= 2.0) and ($yNum$ <= 2.9) : **return** 2
if ($yNum$ >= 3.0) and ($yNum$ <= 3.9) : **return** 3
if ($yNum$ >= 4.0) and ($yNum$ <= 4.9) : **return** 4
if ($yNum$ >= 5.0) and ($yNum$ <= 5.9) : **return** 5
if ($yNum$ >= 6.0) and ($yNum$ <= 6.9) : **return** 6
if ($yNum$ >= 7.0) and ($yNum$ <= 7.9) : **return** 7
if ($yNum$ >= 8.0) and ($yNum$ <= 8.9) : **return** 8
if ($yNum$ >= 9.0) and ($yNum$ <= 9.9) : **return** 9
if ($yNum$ >= 10) and ($yNum$ <= 40.0): **return** 10
if ($yNum$ >= 41.0) and ($yNum$ <= 60): **return** 11
if ($yNum$ >= 60) : **return** 12

Meanwhile, this study used numeric output data ($yNum$). $yNum$ is the value of breath acetone concentration in ppm from the ground truth (Ketoscan). Thus, there should be a strategy to implement SMOTE. In this study, the SMOTE implementation strategy was carried out by converting $yNum$ to $yCat$. The conversion process is done by clustering $yNum$ based on the ketone level guide by Ketoscan [17]. The sequence of the conversion process is presented in Algorithm 1.

After the conversion process, a synthetic data generation process was performed to enhance the amount of training data. The process is called SMOTE Step-1. Before the oversampling process began, the $yNum$ data were temporarily inserted into the $xDataset$. Because the synthetic data of $yNum$ were required in the subsequent stage. Furthermore, SMOTE Step-1 was conducted using the same calculations as standard SMOTE using $xDataset$ and $yCat$. The SMOTE Step-1 process would equalize the data on minority $yCat$ by generating synthetic data. The synthetic data was subsequently separated from the original data and stored in the $xSynthetic$ dataframe. Meanwhile, $yNum$ in $xSynthetic$ was removed and stored in a new dataframe, namely $ySynthetic$. From this process, SMOTE Step-1 generated $xSynthetic$ and $ySynthetic$. The sequence of SMOTE Step-1 is presented in detail in Algorithm 2.

Once SMOTE-step 1 was completed, the data was then split into 70% training data ($xTrain$ and $yNumTrain$) and 30% testing data ($xTest$ and $yNumTest$). Standardization was subsequently performed to equalize the range of data for each feature. The standardization process was performed

Algorithm 2: SMOTE Step-1

Input : $xDataset$ (Dataframe contain extracted feature).
$yCat$ (Dataframe contain target data in categorical).
Output : $xSynthetic$ (Dataframe contain sintetic data feature).
$ySynthetic$ (Dataframe contain sintetic data target).
Step :
Add $yNum$ to $xScaled$ dataframe
$xSynthetic, ySynthetic \leftarrow$ **Perfome** SMOTE on ($xDataset, yCat$).
$ySynthetic \leftarrow$ **Replace** with $yNum$ from $xSynthetic$.

Drop $yNum$ from $xSynthetic$
Return $xSynthetic, ySynthetic$

using Eq (12). In this equation, $xScaled$ denotes standardized data, while μ is the Mean of all data and σ is the standard deviation. The standardization process is carried out on the $xTrain$, $xTest$, and $xSynthetic$, producing $xTrainScale, xTestScale,$ and $xSynScale$ standardized data.

$$xScaled = \frac{xFeature - \mu}{\sigma} \quad (12)$$

Because of the extremely large feature dimensions, a dimension reduction process was performed using the principal component analysis (PCA). PCA aims to reduce data by extracting the dominant pattern in each feature [20]. PCA has five main steps, namely standardization, calculation of the covariance matrix, calculation of Eigenvector and Eigenvalue, feature vectors, and recast data. PCA process is performed on standardized data with the results of $xTrainPca$, $xSynPca$, and $xTestPca$. The sequence from the data split process to PCA is presented in Algorithm 3. After the dimension was reduced, the data were split for validation using stratified K-fold cross validation (SKF). SKF is a technique of dividing the sample into folds by dividing the number of classes in the same proportion [21]. In this research SKF was configured with 4 folds. Fold 1 to 3 were used for the training data, while fold 4 was used for data validation. The validation process was iterated several times until each data had ever taken part in data testing and data validation. In this process, the $xTrainPca$ and $yNumTrain$ data frames were split using SKF and at each iteration generated training data and validation data, namely $xTrainFold, yTrainFold, xValFold,$

Algorithm 3: Split dataset and PCA reduction

Input :
$xDataset$ (dataframe contain extracted feature).
$yNum$ (dataframe contain target data in numeric).
$xSynthetic$ (dataframe contain synthetic data feature result of SMOTE Step-1).

Output :
$xTrainPca$ and $yNumTrain$ (Feature training data with dimension reduction and training target).
$xTestPca$ and $yNumTest$ (Feature testing data with dimension reduction and testing target).
$xTestPca$ (Syntetic data feature with dimension reduction).

Step 1 : Data Split and Standardization.
$xTrain, xTest, yNumTrain, yNumTest \leftarrow$ **Split** $(xDataset, yNum)$.

$xTrainScale, xTestScale, xSynScale \leftarrow$ **StandardScale**$(xTrain, xTest, xSynthetic)$.
Step 2 : Dimension Reduction.
$xTrainPca, xTestPca, xSynPca \leftarrow$ **PCA** $(xTrainScale, xTestScale, xSynScale)$.
Return..$xTrainPca, xTestPca, xSynPca, yNumTrain, yNumTest$.

and $yValFold$. To enlarge the number of training data, $xTrainFold$ and $yTrainFold$ were combined with synthetic data of ($xSynPca$ and $ySynthetic$).

The combination data was stored in the $xTrainCom$ and $yTrainCom$ data frames. However, this combination resulted in unbalanced data distribution at each ketone level became unbalanced. Thus, SMOTE Step-2 was implemented to balance data distribution at each ketone level balanced. Before conducting SMOTE Step-2, temporary $yTrainCom$ was added to the $xTrainCom$ data frame. Afterward, $yTrainCom$ was converted into categorical using Algorithm 1 to produce $yTrainComCat$. This process aimed to oversample numerical data as in SMOTE Step-1. SMOTE step-2 was performed using the steps described in Eq. (11). Using SMOTE step-2 the data would have a balanced proportion again at each ketone level and would be stored in the $xSmote2$ and $ySmote2$ data frame. The next step was to save $yTrainCom$ of $xSmote2$ into $yTrainFinal$ data frame and drop $yTrainCom$ which was stored in the $xSmote2$data frame. Then, $xSmote2$ became $xTrainFinal$. The sequence of the entire process from SKF to SMOTE step-2 was presented in Algorithm 4 and produced

Algorithm 4: Cross validation SMOTE step-2

Input :
$.xTrainPca, yNumTrain, xTestPca, yNumTest$
$.xSynPca, ySyntetic$.

Output :
$xTrainFinal$ and $yTrainFinal$ (Feature training data with dimension reduction and training target).
$xValidationFinal$ and $yValidationFinal$ (Feature testing data with dimension reduction and testing target).

Step 1 : Stratisfied *K-fold* Cross Validation (SKF) data split.

$xTrainFold, yTrainFold, xValFold, yValFold \leftarrow$ **SKF**$(xTrainPca, yNumTrain)$.

Step 2 : Data Combination.
$xTrainCom \leftarrow$ **Combine**$(xTrainFold, xSynPca)$
$yTrainCom \leftarrow$ **Combine**$(yTrainFold, ySynthetic)$.

Step 3 : SMOTE Step-2.
Add $yTrainCom$ to $xTrainCom$ dataframe.
$yTrainComCat \leftarrow$ **Convert** $yTrainCom$ to Categorical.
$xSmote2, ySmote2 \leftarrow$ **Perfome SMOTE**$(xTrainCom, yTrainComCat)$.
$yTrainFinal = yTrainCom$ from $xSmote2$
Drop $yTrainCom$ from $xSmote2$.
$xTrainFinal = Smote2$.
$xValidationFinal = xValFold$.
$yValidationFinal = yValFold$.
Return
$xTrainFinal, yTrainFinal, xValidationFinal, xTrainFinal$.

$xTrainFinal$, $yTrainFinal$, $xValidationFInal$, and $yValidationFinal$.

The use of Two-Step SMOTE made it practical to add and maintain a balanced training dataset. After completing data preparation and augmentation, machine learning testing is described in the next subsection.

2.5 Machine learning testing

In this study, machine learning testing was conducted by inputting a dataset to detect ketone levels using a regression technique. The machine learning testing was performed in three scenarios, namely the deep learning regression, standard machine learning regression, and boosting regression.

Deep neural network regression (DNN-R) was employed in the deep learning scenario. The DNN-R is a method developed from the neural network that

has more hidden layers [21]. Each hidden layer consists of a group of neurons responsible for processing information from the dataset. The number of neurons in each hidden layer is determined based on the dataset. Therefore, to gain optimal results, experiments were conducted to determine these parameters.

Standard machine learning uses five methods, namely standard multiple linear regression (ML-R), K- nearest neighbor regression (KNN-R), ransac regression (RC-R), decision tree regression (DT-R), and random forest regression (RF-R).

ML-R is a technique for predicting the outcome of the dependent variable from the input of more than one independent variable [22]. The ML-R model can be formulated using Eq. (13). In this equation, y is the dependent variable, $\beta 0$ is y-intercept of y variable when the other independent variable is 0, βi regression coefficient of the independent varible for index i , and Xi the value of independent variable for index i .

$$y = \beta 0 + \sum_{i=0}^{n} \beta i Xi + e \quad (13)$$

KNN-R is a regression method based on the KNN-C classification method. KNN-R makes predictions by calculating the distance of data input of testing data for each training dataset. Distance calculations are usually performed using the Euclidean distance [23]. The data on the k nearest neighbors are then selected. Afterward, the Mean of data on the k nearest neighbors is determined. The k value can be configured so that the best results are obtained.

RC-R is developed from standard linear regression which excludes outlier data. In the first step, RC-R began with selecting a random data set and model fitting the data using standard linear regression. The second step is calculating residual error using Eq. (14) from the results of the model fitting. Data with a residual error value lower than the threshold was seen as normal data (inlier). These first and second steps were iterated as needed. In the fourth step, the model with the highest number of inliers was saved [24].

$$ResidualError = Actual - Predicted \quad (14)$$

DT-R is a regression method in which model development is conducted using a tree structure that has root, branches, and leaf nodes [25]. The decision tree is completed by selecting the appropriate attribute with attribute selection measures (ASM). ASM can be done by calculating the information gain (Ig) of each feature in the dataset using Eq. (15). The feature with the greatest information gain will be the root node. The process will be repeated until all the features in the dataset are in the decision tree [26] . The concept is the same as a decision tree classification, but the regression has a numerical output value .

$$Ig = n - [WeightedAverage * Entropy(Each\ Input)] \quad (15)$$

RF-R is a regression method that employs ensemble learning techniques. The ensemble process was conducted by combining the prediction results in several decision trees. The final prediction results were obtained from the average prediction results of all decision trees [27].

The boosting regression involved three methods, namely, adaptive boosting regression (AB-R), gradient boost regression (GB-R), and extreme gradient boost regression (XGB-R). AB-R is a regression method that implements a set of weak learners such as a simple decision tree which is added sequentially at each iteration. The number of iterations was determined as needed [28]. At each iteration, the data was re-weighted based on the error from the previous weak learner. The final result was obtained from a combination of prediction results by all weak learners.

GB-R is a regression method similar to AB-R. This method combined a group of decision tree-based weak learners. The difference lies in the process in each iteration. In the first iteration of GB-R, a prediction process was conducted with the decision tree using the input *x* feature dataset and the *y* output. Subsequently, the value of the residual error of the process was used as the y output value in the next decision tree [29]. The process was iterated according to the specified iteration limit. XGB-R is a method developed from GB-R by applying regularization techniques.

In each machine learning method explained above, the hyperparameter was determined using a random tuning method. This tuning process is evaluated using the SKF technique using $xTrainFinal, yTrainFinalxValidationFinal,$ and $yValidationFinal$ data. The evaluation result was measured using the average matrix: accuracy (Acc), R^2 Score (R2) dan Mean Square Error (MSE). Accuracy was determined using Eq. (16), R^2 Score was calculated using Eq. (17), and MSE was determined using Eq. (18). On these equations, y

Algorithm 5: Hyperparameter tuning

Input : $xTrainFinal, yTrainFinal,$ $xValidationFinal\ dan\ yValidationFinal$

Output : Best Model For Each ML Method
Step 1 :
Random Hyperparameter based on ML method
SKF Iteration :
Model.fitting($xTrainFinal, yTrainFinal$)
$Accuracy, R^2Score, MSE \leftarrow$ **Evaluation** ($xValidationFinal$ dan $yValidationFinal$)
Calculate Average($Accuracy, R^2Score, MSE$)
Save model with best evaluation results
Step 2 : Repeat Step 1 for each ML Method (Except ML-R and RC-R).
Return Best model

Algorithm 6 : Balance testing

Input : $xTestPca, yTest$
Output : $balanceAccuracy, balance R^2Score$ $balanceMSE$
Step 1 :
For 1 **to** 10 :
$Accuracy, R^2Score, MSE \leftarrow$Model.**predict** **(**$xTestPca$).
balance$Accuracy, balance R^2Score, balMSE \leftarrow$ Calculate Average ($Accuracy, R^2Score, MSE$)

Return
balance$Accuracy, balance R^2Score, balMSE$

denotes the output value of the original data, $yPred$ was the output of prediction, and n is the number of data. The sequence of the tuning process is presented in Algorithm 5.

$$Acc = 100\% - \left(\sum_{i=0}^{n}\left[\frac{y-yPred}{yPred}\right] x\ 100\%\right) \quad (16)$$

$$R^2\ Score\ = \left(1 - \frac{\sum_{i=0}^{n}(y-yPred)^2}{\sum_{i=0}^{n}\left(y-\frac{\sum_{i=1}^{n}y}{n}\right)^2}\right) x\ 100 \quad (17)$$

$$MSE\ = \frac{1}{n}\ \sum_{i=0}^{n}(y\ - yPred)^2 \quad (18)$$

After obtaining the best model, $xTestPca$ and $yTest$ data were tested using each machine learning model with 10 fitting iterations for each machine learning model. Each iteration was evaluated for matrix accuracy, R^2 score and mean square error (MSE). The average value of the results of each iteration were determined to find balance accuracy, balance R^2 Score and balance MSE. The sequence of data testing can be seen in Algorithm 6.

Table 1 Extracted feature representation

avgMQ138	stdMQ138	...	totalml
0.0124	0.0054	...	67
...	...	...	...
0.0034	0.0044	...	66

The results of each machine learning method were compared with input data, without synthetic data (E1), with standard SMOTE (E2) and proposed two-step SMOTE (E3). SMOTE (E2) was conducted by converting the output y to a categorical form, and then the SMOTE process was conducted only to balance the number of training datasets based on the previous research [12] The results are described completely in Section 3.

3. Results and discussion

This section presents the results of this study that consists of the results of pre-processing, data augmentation, and the machine learning test.

3.1 Results of signal pre-processing

The pre-processing process was conducted in three steps namely: MAF Filter, FFT and feature extraction. The first stage of the MAF Filter was conducted to reduce signal fluctuation. The experiment was conducted by configuring the M value with several values. Fig. 4 presents the example of a signal to represent the experimental results. The figure shows the WSP2110 signal with a configuration of M values: 2, 4, 6, 8, 10 and 12. The results showed that the signal began to look smoother in the M configuration with a value of 10. Thus M = 10 was selected as filter configuration for all gas sensor data. Because if it is too smooth, it can reduce the information contained in the signal [18]. After obtaining a smooth signal, the FFT process was conducted to obtain a signal in the frequency domain. The conversion results to the frequency domain are presented in Fig. 5. The third stage is feature extraction on the time domain of the signal from the filter and the frequency domain of the signal from the FFT. The extraction resulted in 51 features in the each gas sensor signal (statistical, freq, baseline "P1", average "P2" and max "P2") and 7 additional features on the environmental sensor (average temperature, average humidity, average air pressure, min air pressure, max air pressure, and total ml). A

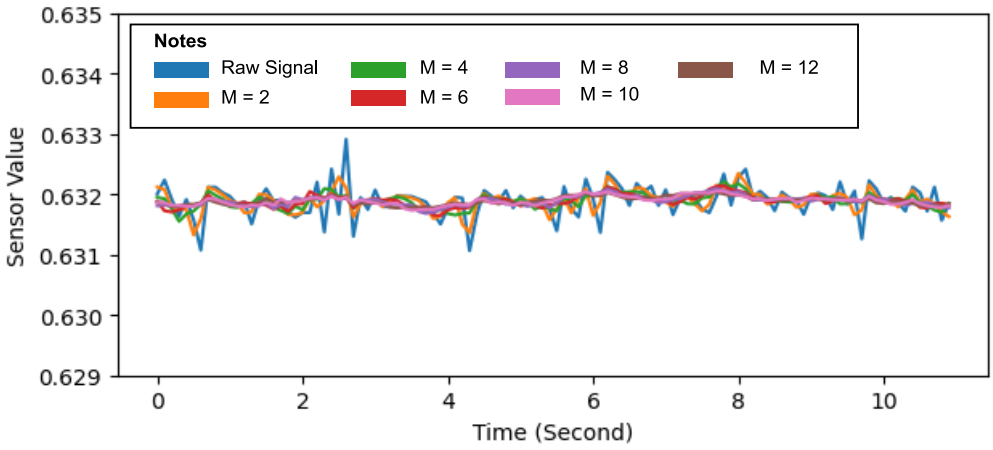

Figure. 4 MAF filter with different configuration

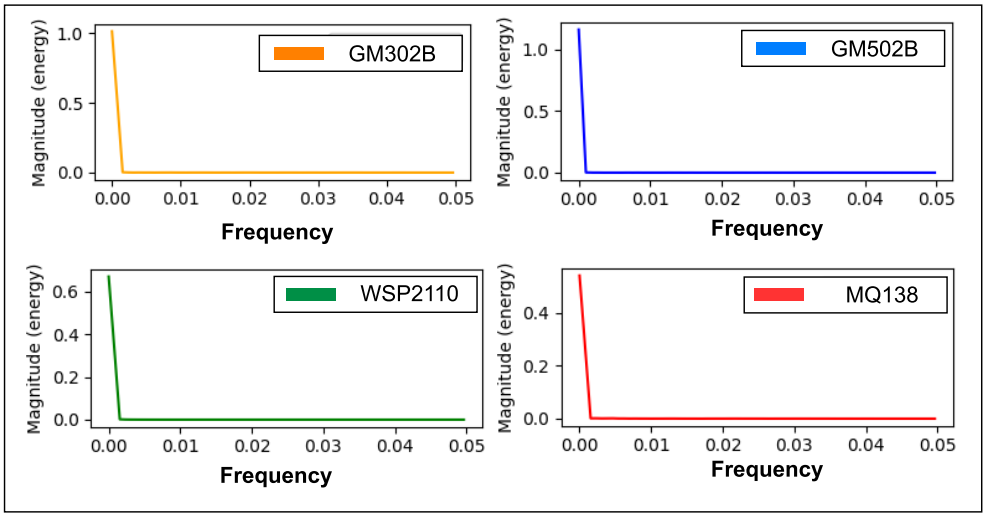

Figure. 5 Frequency domain signal representation

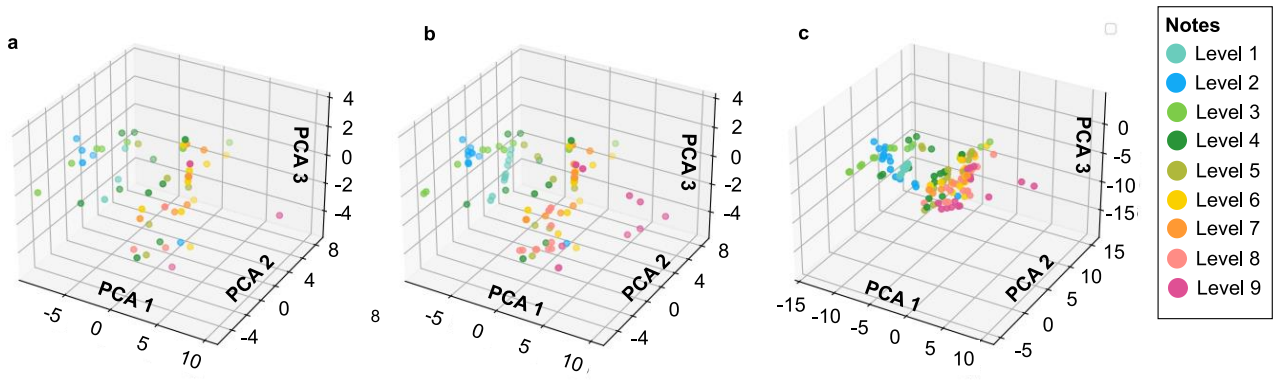

Figure. 6 Data distribution based on PCA component on: (a) E1, (b) E2, and (c) E3

representation of feature extraction can be seen in Table 1.

3.2 Results of data preparation and augmentation

The first stage involved 70% training and 30% testing data. Thus, the total number of training data is 70 data. The second stage is the SKF cross-validation stage with a 4 fold configuration. The training data obtained for each fold were 52 training folds and 18 testing folds. In the third stage, combination training-fold data was carried out with synthetic data from the results of SMOTE step-1, and a total of 114 data was obtained. The fourth stage was conducted using

Table 2 Comparison of the amount of data

Data Code	Number Of Training Fold Data
E1	52
E2	90
E3	135

Table 3 Comparison of data distribution at each ketone level

Level	E1	E2	E3
1	5.769 %	11.111 %	11.111 %
2	9.615 %	11.111 %	11.111 %
3	13.461 %	11.111 %	11.111 %
4	19.230 %	11.111 %	11.111 %
5	11.538 %	11.111 %	11.111 %
6	17.307 %	11.111 %	11.111 %
7	11.538 %	11.111 %	11.111 %
8	5.769 %	11.111 %	11.111 %
9	5.769 %	11.111 %	11.111

SMOTE step-2 and obtained a total of 135 data. In Table 2, the proposed method (E3) has a larger amount of training data compared to without synthetic data (E1) which consisted of 52 data and 90 standard SMOTE (E2) data. In experiment E2, oversampling was conducted using Standard SMOTE as in the previous study [12]. However, the *y* data were converted into categorical as in the SMOTE step 1 process. From the percentage of data balance in Table 3, the data without oversampling had an unbalanced distribution percentage at each ketone level. Furthermore, the standard SMOTE method produced a balance distribution of 11.111% on each ketone level. Then, with Two-Step SMOTE, the data distribution became balanced with a value of 11,111% at each ketone level. The distribution of E1, E2, and E3 data in PCA components can be seen in Fig. 6. Based on the results discussed, the E3 method has the largest amount of training data and still has a balanced distribution of data.

3.3 Results of machine learning testing

Machine learning testing was conducted by comparing the use of the E1, E2, and E3 datasets with 9 machine learning methods. In each dataset, the hyperparameter tuning process was performed for each machine learning method using a random tuning technique combined with SKF cross-validation (Algorithm 5). The process was performed using the same technique. However, the hyperparameters were adjusted according to the type of method. The best hyperparameters for each machine learning method

Table 4 Best hyperparameter for each dataset

Method	E1	E2	E3
DNN-R	{n1 : 282; n2 :37; n3:29; dr : 0.341; I2_r:0.001}	{n1:171; n2: 68; n3:163; dr:0.171; I2_R:0.001}	{n1:180; n2:80: n3 :111; dr:0.26; l2_r:0.001}
KNN-R	{k:6}	{k:13}	{k:4}
DT-R	{md :3}	{md: 222}	{md: 81}
RF-R	{md : 68; n_est :4}	{md :55; n_est :6}	{md : 63; n_est :12}
AB-R	{n_est: 3}	{n_est:3}	{n_est :24}
GB-R	{md :6; n_est :70}	{md :6; n_est:146}	{md :47; n_est :114}
XGB-R	{md :4; n_est :4;}	{md:298; n_est : 224}	{md:101; n_est : 148}

Table 5 Average evaluation for best hyperparameter

Method	E1	E2	E3
DNN-R	Acc : 73.204 R2 : 50.606 MSE : 2.481	Acc : 71.880 R2 : 47.294 MSE : 2.643	Acc : 80.023 R2 : 70.799 MSE : 1.481
KNN-R	Acc : 65.167 R2 : 47.337 MSE : 2.64	Acc : 65.809 R2 : 43.167 MSE : 2.845	Acc : 73.070 R2 : 51.362 MSE : 2.424
DT-R	Acc : 65.176 R2 : 16.856 MSE : 4.169	Acc : 68.9 R2 : 24.576 MSE : 3.73	Acc : 75.773 R2 : 48.914 MSE :2.539
RF-R	Acc : 68.523 R2 : 38.034 MSE : 3.099	Acc : 70.060 R2 : 40. 097 MSE : 2.985	Acc : 76.229 R2 : 61.547 MSE :1.926
AB-R	Acc : 68.876 R2 : 34.377 MSE : 3.275	Acc : 68.766 R2 : 38.956 MSE : 3.037	Acc :73.358 R2 : 53.586 MSE :2.319
GB-R	Acc : 65.859 R2 : 14.585 MSE : 4.272	Acc : 68.965 R2 : 31.352 MSE : 3.43	Acc : 77.848 R2 : 46.504 MSE : 2.680
XGB-R	Acc :64.583 R2 : 72.748 MSE : 1.363	Acc : 68.634 R2 : 22.962 MSE : 3.829	Acc : 78.857 R2 : 61.857 MSE :1.891

with E1, E2 , and E3 data are presented in Table 4. From the Table 4 n1 : the number of neuron in hidden layer 1, n2 : the number of neuron in hidden layer 2, n3 : the number of neuron in hidden layer 1, dr : the dropout value, I2_r : I2 regularization value, k : number of nearest neighbors, *md* : showed the maximum depth of the tree, and *n_est* shows the number of weak learners created. The value of evaluation for each of these parameters is presented in Table 5.

By using the best model, data testing was iterated 10 times. The test results for the evaluation balance value of each machine learning method are presented in Table 6. The best results were obtained

Table 6 Comparison of prediction results on testing data

Method	E1	E2	E3
DNN-R	Acc : 76.327 R2 : 59.981 MSE : 2.058	Acc : 73.482 R2 : 58.123 MSE : 2.154	Acc : 77.285 R2 : 73.241 MSE : 1.376
ML-R	Acc : 68.717 R2 : 59.621 MSE : 2.077	Acc : 72.277 R2 : 60.899 MSE : 2.011	Acc : 78.227 R2 : 73.115 MSE : 1.382
RC-R	Acc : 37.929 R2 : 0.441 MSE : 5.121	Acc : 66.734 R2 : 29.179 MSE : 3.193	Acc : 73.669 R2 : 60.922 MSE : 2.01
KNN-R	Acc : 61.345 R2 : 55.397 MSE : 2.294	Acc : 70.149 R2 : 63.449 MSE : 2.806	Acc : 79.478 R2 : 70.048 MSE :1.540
DT-R	Acc : 68.26 R2 : 42.325 MSE : 2.966	Acc : 69.083 R2 : 38.159 MSE : 3.181	Acc : 76.496 R2 : 50.776 MSE : 2.532
RF-R	Acc : 69.385 R2 : 57.589 MSE : 2.181	Acc : 73.796 R2 : 54.391 MSE : 2.346	Acc : 80.733 R2 : 71.346 MSE : 1.473
AB-R	Acc : 68.918 R2 : 46.153 MSE : 2.769	Acc : 68.934 R2 : 40.559 MSE : 3.057	Acc :78.397 R2 : 70.997 MSE :1.491
GB-R	Acc : 72.25 R2 : 42.845 MSE : 2.94	Acc : 70.521 R2 : 34.544 MSE : 3.367	Acc :77.344 R2 : 52.295 MSE :2.453
XGB-R	Acc : 71.278 R2 : 25.141 MSE : 3.850	Acc : 70.073 R2 : 27.217 MSE : 3.743	Acc : 82.543 R2 : 67.078 MSE : 1.603

in the proposed method E3 by implementing two-step SMOTE. Compared to E2, E3 improved the accuracy in DNN-R, ML-R, RC-R, KNN-R, DT-R, RF-R, AB-R, GB-R, and XGB-R by 3.803%, 5.95%, 6.935%, 9.329%, 7.413%, 6.937%%, 9.463%, 6.823% and 12.47% respectively. Furthermore, compared to E1, E3 improved the accuracy (Acc) by 0.958%, 9.51%, 35.74%, 18.133%, 8.236%, 11.348, 9.47%, 5.093%, and 11.264% respectively. The R^2Score (R2) in E3 increased compared to E2 by 15.118, 12.216, 31.743, 6.599, 12.616, 16.955, 30.438, 17.751 and 39.861 respectively. Similarly, compared to E1, the R^2Score of E3 increased by 13.26, 13.494, 60.481, 14.651, 8.451, 13.757, 24.844, 9.45, and 41.937 respectively. Additionally, the MSE decreased in E3 compared to DNN-R, ML-R, RC-R, KNN-R, DT-R, RF-R, AB-R, GB-R, and XGB-R by 0.778, 0.629, 1.183, 1.266, 0.649, 0.873, 1.566, 0.914, and 2.14 respectively. Meanwhile, compared to E1, MSE improved by 0.628, 0.695, 3.111, 0.754, 0.434, 0.708, 1.278, 0.487, and 2.247 respectively.

Overall, the E3 dataset with Two-Step SMOTE showed significant enhancements across multiple performance metrics compared to E1 and E2 [12].

4. Conclusion and future research

Based on the results of the present research, the proposed method can apply the SMOTE technique to the regression case by clustering the target data values into numeric to increase the total number of training data. Furthermore, the proposed method enhances the standard SMOTE technique with Two-Step SMOTE. Compared to the result without oversampling, the proposed method can enhance the balance value of accuracy, R^2Score, and MSE across all machine learning methods. Significant results were obtained in the RC-R, KNN-R, RF-R and XGB-R. Compared to the standard SMOTE, the proposed method can also enhance the balance value of accuracy, R^2 Score, and MSE across all machine learning methods. Significant results were obtained in the XGB-R, AB-R, and KNN-R

In particular, the DNN-R method using two-step SMOTE data produced the best prediction results that stable in all aspect. with an accuracy balance of 77.285%, R^2 Score of 73.241, and MSE of 1.376. These findings indicate feasibility of the proposed method in improving the accuracy of the dataset in the Electronic-nose ketone level detection with limited amounts of data.

Future studies can develop the method using other data augmentation techniques that utilize deep learning approaches such as Generative Adversarial Networks (GAN). The use of this method may generate higher-quality synthetic data. The better-quality data will eventually increase the accuracy of the prediction process.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

This study can be completed because of the following research contribution: Conceptualization, Dhiza Wahyu Firmansyah and Riyanarto Sarno; methodology, Dhiza Wahyu Firmansyah and Riyanarto Sarno; software, Dhiza Wahyu Firmansyah; validation Dhiza Wahyu Firmansyah; writing—original draft preparation, Dhiza Wahyu Firmansyah; writing—review and editing, Dhiza Wahyu Firmansyah; supervision, Riyanarto Sarno; Proposes problem ideas, Riyanarto Sarno.

Acknowledgments

This work was supported in part by Asian Development Bank under Higher Education for Technology Innovation (ADB HETI) Project; the

Indonesian Ministry of Education and Culture under Penelitian Terapan Unggulan Perguruan Tinggi (PTUPT) Program; and Institut Teknologi Sepuluh Nopember (ITS) under project scheme of the Publication Writing and Intellectual Property Right (IPR) Incentive Penulisan Publikasi dan Hak Kekayaan Intelektual (PPHKI)

References

[1] Hariyanto, R. Sarno, and D. R. Wijaya, "Detection of diabetes from gas analysis of human breath using e-Nose", In: *Proc. of 2017 11th International Conference on Information & Communication Technology and System (ICTS)*, IEEE, pp. 241–246, 2017.

[2] R. Sarno, S. I. Sabila, D. R. Wijaya, and Hariyanto, "Electronic Nose for Detecting Multilevel Diabetes using Optimized Deep Neural Network", *IAENG*, Vol. 28, No. 1, 2020.

[3] A. Rydosz, "A Negative Correlation Between Blood Glucose and Acetone Measured in Healthy and Type 1 Diabetes Mellitus Patient Breath", *J Diabetes Sci Technol*, Vol. 9, No. 4, pp. 881–884, 2015.

[4] V. Saasa, M. Beukes, Y. Lemmer, and B. Mwakikunga, "Blood Ketone Bodies and Breath Acetone Analysis and Their Correlations in Type 2 Diabetes Mellitus", *Diagnostics*, Vol. 9, No. 4, p. 224, 2019.

[5] H. K. Akturk, J. S. Bergeon, L. Pyle, E. Fivekiller, S. Garg, and E. Cobry, "Accuracy of a breath ketone analyzer to detect ketosis in adults and children with type 1 diabetes", *J. Diabetes Complications*, Vol. 35, No. 11, p. 108030, 2021.

[6] S. Tsunemi, Y. Nakamura, K. Yokota, T. Nakagawa, H. Tsukiyama, Y. Kubo, T. Oyanagi, A. Takemoto, Y. Nagai, Y. Tanaka, and M. Sone, "Correlation between blood ketones and exhaled acetone measured with a semiconducting gas sensor", *J. Breath Res*, Vol. 16, No. 4, p. 046004, 2022.

[7] G. Hancock, S. Sharma, M. Galpin, D. Lunn, C. Megson, R. Peverall, G. Richmond, G. A. D. Ritchie, and K. R. Owen, "The correlation between breath acetone and blood betahydroxybutyrate in individuals with type 1 diabetes", *J. Breath Res*, Vol. 15, No. 1, p. 017101, 2021.

[8] A. Thati, A. Biswas, S. R. Chowdhury, and T. K. Sau, "Breath Acetone-Based Non-Invasive Detection of Blood Glucose Levels", *International Journal on Smart Sensing and Intelligent Systems*, Vol. 8, No. 2, pp. 1244–1260, 2015.

[9] D. Guo, D. Zhang, L. Zhang, and G. Lu, "Non-invasive blood glucose monitoring for diabetics by means of breath signal analysis", *Sens. Actuators B. Chem.*, Vol. 173, pp. 106–113, 2012.

[10] H. M. Saraoglu, A. O. Selvi, M. A. Ebeoglu, and C. Tasaltin, "Electronic Nose System Based on Quartz Crystal Microbalance Sensor for Blood Glucose and HbA1c Levels From Exhaled Breath Odor", *IEEE Sens. J.*, Vol. 13, No. 11, pp. 4229–4235, 2013.

[11] K. Yan and D. Zhang, "Blood glucose prediction by breath analysis system with feature selection and model fusion", In: *Proc. of 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6406–6409, 2014.

[12] H. Wei and Y. Gu, "A Machine Learning Method for the Detection of Brown Core in the Chinese Pear Variety Huangguan Using a MOS-Based E-Nose", *Sensors*, Vol. 20, No. 16, p. 4499, 2020.

[13] J. Tong, C. Song, T. Tong, X. Zong, Z. Liu, S. Wang, L. Tan, Y. Li, and Z. Chang, "Design and Optimization of Electronic Nose Sensor Array for Real-Time and Rapid Detection of Vehicle Exhaust Pollutants", *Chemosensors*, Vol. 10, No. 12, p. 496, 2022.

[14] T. I. Nasution, R. Asrosa, and I. Nainggolan, "Application of MQ-138 Semiconductor Sensor for Breath Acetone Detection", *J. Phys. Conf Ser.*, Vol. 1116, p. 032023, 2018.

[15] B. Liu, Y. Zhou, H. Fu, P. Fu, and L. Feng, "Lightweight Self-Detection and Self-Calibration Strategy for MEMS Gas Sensor Arrays", *Sensors*, Vol. 22, No. 12, p. 4315, 2022.

[16] D. R. Wijaya, R. Sarno, and E. Zulaika, "Gas concentration analysis of resistive gas sensor array", In: *Proc. of 2016 International Symposium on Electronics and Smart Devices (ISESD)*, IEEE, pp. 337–342, 2016.

[17] ketoscanmini.co.uk, "KETOSCAN Level Guide",*WebPage*.https://ketoscanmini.co.uk/wp-content/uploads/2019/09/KETOSCAN-Level-Guide.pdf (Accessed Aug. 07, 2022).

[18] T. C. Amri, R. Sarno, R. Abdillah, F. A. Haq, A. F. Septiyanto, and D. Sunaryono, "Filter Validation for Detecting Outliers of Photoplethysmograph Data", In: *Proc. of 2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 012–017, 2023.

[19] R. Zhi, L. Zhao, and D. Zhang, "A Framework for the Multi-Level Fusion of Electronic Nose

and Electronic Tongue for Tea Quality Assessment", *Sensors*, Vol. 17, No. 5, p. 1007, 2017.

[20] Malikhah, R. Sarno, S. Inoue, M. S. H. Ardani, D. P. Purbawa, S. I. Sabilla, K. R. Sungkono, C. Fatichah, D. Sunaryono, A. Bakhtiar, Libriansyah, C. R. S. Prakoeswa, D. Tinduh, and Y. Hernaningsih, "Detection of Infectious Respiratory Disease Through Sweat From Axillary Using an E-Nose With Stacked Deep Neural Network", *IEEE Access*, Vol. 10, pp. 51285–51298, 2022.

[21] M. Malikhah, R. Sarno, S. Inoue, M. S. H. Ardani, D. P. Purbawa, S. I. Sabilla, K. R. Sungkono, C. Fatichah, D. Sunaryono, A. Bakhtiar, Libriansyah, C. R. Prakoeswa, D. Tinduh, and Y. Hernaningsih, "A New Approach for Detection of Viral Respiratory Infections Using E-nose Through Sweat from Armpit with Fully Connected Deep Network", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 2, pp. 394–404, 2022, doi: 10.22266/ijies2022.0430.36.

[22] E. Moretti, M. G. Proietti, and E. Stamponi, "A multiple Linear Regression Model to predict indoor temperature trend in historic buildings for book conservation: the case study of "Sala del Dottorato" in Palazzo Murena, Italy", *J. Phys. Conf. Ser.*, Vol. 2069, No. 1, p. 012142, 2021.

[23] A. Sumayli, "Development of advanced machine learning models for optimization of methyl ester biofuel production from papaya oil: Gaussian process regression (GPR), multilayer perceptron (MLP), and K-nearest neighbor (KNN) regression models", *Arabian Journal of Chemistry*, Vol. 16, No. 7, p. 104833, 2023.

[24] O. Kaspi, A. Yosipof, and H. Senderowitz, "RANdom SAmple Consensus (RANSAC) algorithm for material-informatics: application to photovoltaic solar cells", *J. Cheminform*, Vol. 9, No. 1, p. 34, 2017.

[25] K. M. Hindrayani, T. M. Fahrudin, R. P. Aji, and E. M. Safitri, "Indonesian Stock Price Prediction including Covid19 Era Using Decision Tree Regression", In: *Proc. of 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 344–347, 2020.

[26] J. S. Kushwah, A. Kumar, S. Patel, R. Soni, A. Gawande, and S. Gupta, "Comparative study of regressor and classifier with decision tree using modern tools", *Mater Today Proc*, Vol. 56, pp. 3571–3576, 2022.

[27] L. Breiman, "Random Forests. Machine Learning", *Mach Learn*, Vol. 45, No. 1, pp. 5–32, 2001.

[28] S. B. Koduri, L. Gunisetti, C. R. Ramesh, K. V. Mutyalu, and D. Ganesh, "Prediction of crop production using adaboost regression method", *J. Phys. Conf. Ser.*, Vol. 1228, p. 012005, 2019.

[29] D. A. Otchere, T. O. A. Ganat, J. O. Ojero, B. N. T. Otoo, and M. Y. Taki, "Application of gradient boosting regression model for the evaluation of feature selection techniques in improving reservoir characterisation predictions", *J. Pet. Sci. Eng.*, Vol. 208, p. 109244, 2022.