2024 Paris Session

10486
D1 Materials and emerging test techniques
PS1 Testing, monitoring and diagnostics

Synthetic Data Generation and Its Applications: Training a Machine Learning Dissolved Gas Analysis Time Series Predictor

Mauricio SOTO*
Hitachi Energy
USA
mauricio.soto@hitachienergy.com

SUMMARY

This study proposes a methodology for generating synthetic time series as applied to a compilation of real dissolved gas analysis data. We discuss the proposed data synthesis mechanism, the feature engineering, and data processing techniques applied to our data to transform it from a time series dataset to a tabular dataset with added features to enhance model training. We later train machine learning classifiers with the generated synthetic data and show how models trained with our technique achieve an accuracy of 98.03%, and regressors trained with our dataset show a 0.16 ppm root mean squared error when evaluated in a held-out dataset of dissolved gas analysis readings from real power transformers. Overall, this study shows a promising technique to generate synthetic transformer data, which would allow researchers in the power industry to have access to a vast body of data. This data can be used to further the creation of artificial intelligence techniques to improve the design, manufacturing, and monitoring of electric components in the power grid.

KEYWORDS

Machine Learning, Synthetic Data, Transformers, Dissolved Gas Analysis, Time Series

1. INTRODUCTION

Machine learning and overall artificial intelligence techniques have great potential in the transformer industry due to their ability to predict transformer behavior, optimize transformer design and overall increase productivity, reliability, and security of transformers.

However, one of the main problems that arises when attempting to apply machine learning techniques in problems suitable for the context, is the need for vast amounts of transformer data, which is typically scarce and not publicly available. Due to privacy, security, and competitiveness concerns, utilities typically refrain from sharing transformer data. Thus, the lack of operational and historical data prevents researchers in the transformer industry from creating machine learning models and analyzing data to create algorithms for the analysis of transformer behavior and their corresponding issues with monitoring and condition assessment, among others.

There have been several approaches proposed to obtain data from utilities including anonymizing data delivery through blockchain [1] or creating platforms for voluntary data submission in exchange for free data analysis. However, they have not yet been successful, given the risks considered by utilities.
We are thus proposing, implementing, and validating an approach to generate high-fidelity synthetic data for transformer functionality based on a combination of real transformer data, domain knowledge expertise, and statistical tools. This approach allows us to generate large amounts of high-fidelity data to analyze and create machine learning models from, without the need to obtain large amounts of real data.

2. INTRODUCTION

Synthetic data is defined as any production data applicable to a given situation that is not obtained by direct measurement [2]. Data generated by statistical tools based on real data, thus falls into this category. Synthetic data is typically created to meet specific needs or to be bound by certain conditions where real data is difficult to be obtained.

Synthetic data is commonly found in applications of physical modelling, such as music synthesizers or flight simulators. Generated data is also used in settings where real data is protected and therefore, using real data would compromise its confidentiality. This includes, for example, customer data protected by GDPR in Europe or HIPAA laws in the United States, as well as many other data privacy laws in different parts of the globe. The intent of creating synthetic data is to approximate how real data would behave without the need of obtaining or using real data. Synthetic data should maintain the structure of the original data, as shown in Figure 1, by creating new data based on the original data.

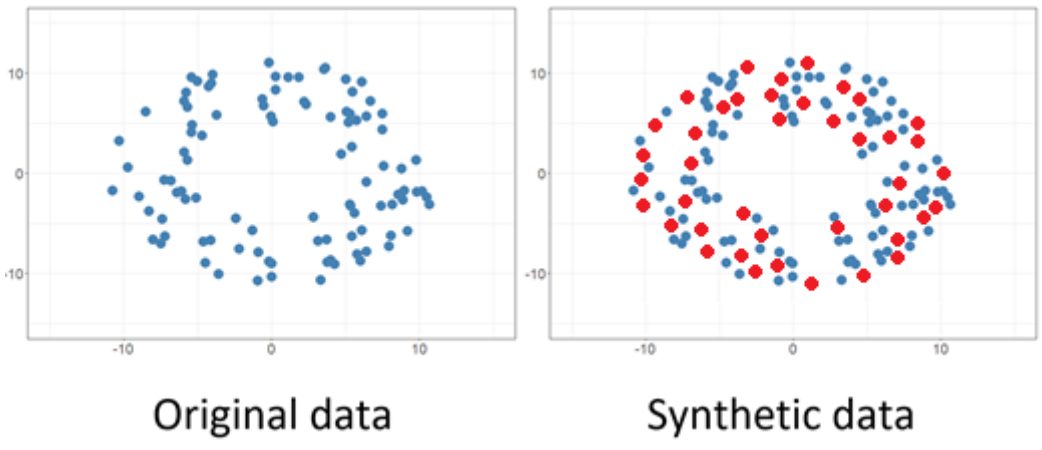


Figure 1. Original data vs. Synthetic data (in red)

Synthetic data can then be used as theoretical values or situations that could potentially occur in real life, and thus, the data can be analyzed to cover these cases. It is generated by increasing the number of samples which allows us to train ML algorithms, particularly in cases for which there are not enough data samples to obtain good accuracy. This helps to prepare for unexpected situations and propose solutions for cases that can but have not happened in the real world.

3. DATA PREPARATION

In this paper, we obtained a small collection of real dissolved gas analysis data from several transformers, which we then used to generate synthetic data, followed by an evaluation of a model trained using the generated synthetic data. These readings are collected with an online DGA sensor (i.e., M10) for an overall of ~18,500 readings per transformer. In this section, we will describe the process of data cleansing and feature engineering applied to our data to set up our experiment.

Two samples of our small dataset are shown in Figure 2 and Figure 3. Figure 2 shows a sample of a transformer with normal Acetylene values during a time span of four and a half months. Figure 3 shows a sample of a transformer with abnormal Acetylene values in the same time frame.

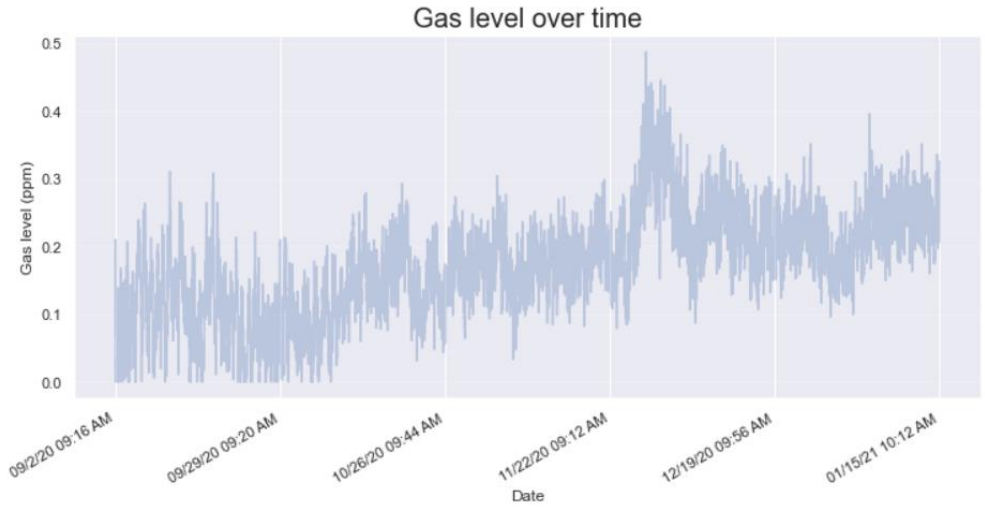


Figure 2. Transformer with normal Acetylene values

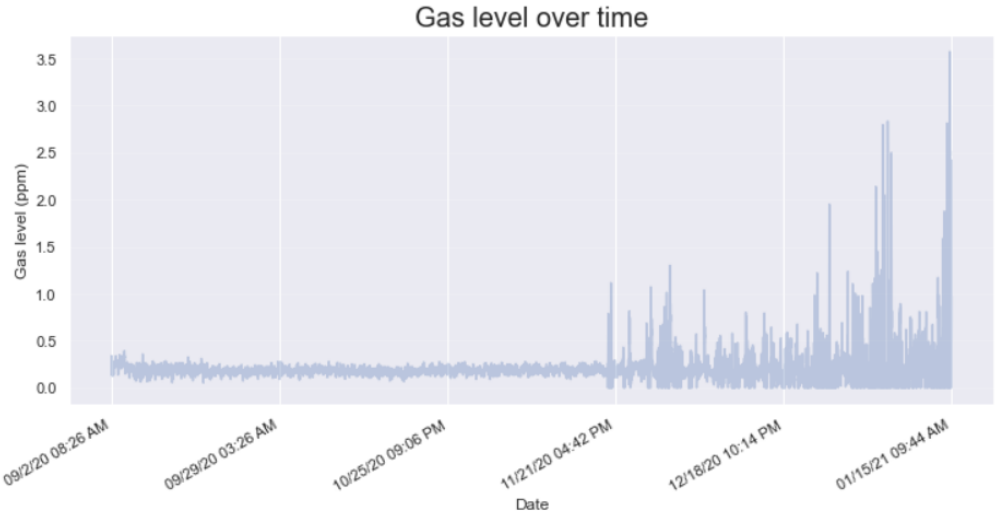


Figure 3. Transformer with abnormal Acetylene values

3.1. **Missing Data.** The first step towards data cleansing is the handling of missing data. In our dataset, there are portions of data where the sensor was not able to obtain a proper reading for the DGA values, thus, these values are stored as null values. We applied two different techniques based on the amount of missing data in each case.

3.1.1. **Interpolation:** When there is less than one hour worth of missing data, we applied linear interpolation as a way to maintain a natural flow of data, which is appropriate in small portions of data since DGA values do not have wide variability in short time spans because dissolved gas builds slowly over time. Figure 4 shows an example in our dataset of interpolation being used to fill small portions of missing data. The picture in the left shows a disconnect between two portions of data. The picture in the right shows, in red, the missing data being interpolated to provide a full flow of DGA values.

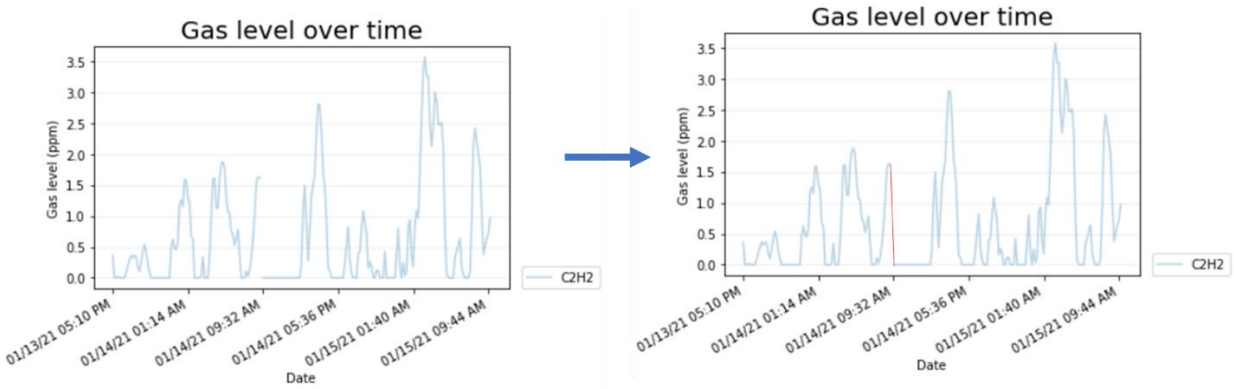


Figure 4. Interpolation in small portions of data

3.1.2. **Segregation:** In the portions where, missing data surpasses the one-hour time span, we remove the missing data from the timeline, and consider the two disconnected portions as two separate samples in the dataset.

Once the collection of real transformer data is curated and cleansed, we prepare to generate synthetic data based on the preprocessed real data.

4. SYNTHETIC DATA GENERATION

In this paper we propose a methodology for generating time series data inspired by SMOTE (i.e., Synthetic Minority Over-sampling Technique) [3]. SMOTE is an approach created with the intent of over sampling the minority class in imbalanced datasets as shown in Figure 5, typically to train classifier models. A dataset is imbalanced if the classification categories are not approximately equally represented. In the real world, it is common that datasets are composed predominantly of samples that happen often in the studied scenarios.

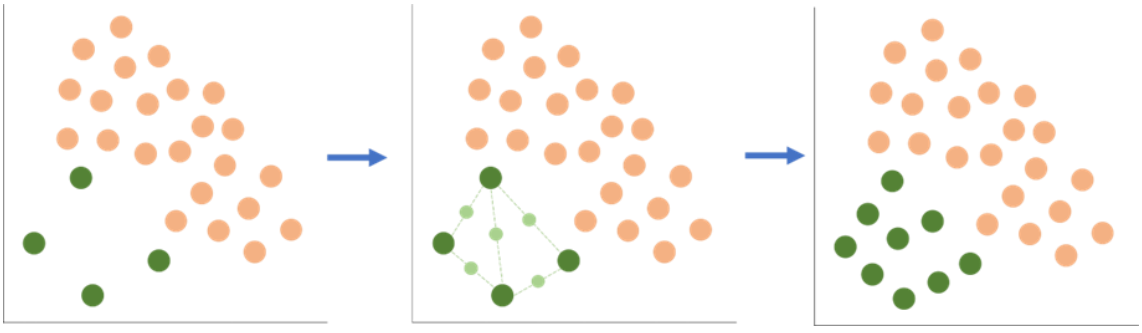


Figure 5. SMOTE process

It is also common, that the interesting cases being analyzed are the uncommon cases in the dataset. For example, authentic credit card transactions are very common in everyday life, and fraudulent credit card transactions are very rare. A bank interested in protecting its customers could then attempt to create a machine learning model that is able to identify when a credit card transaction is fraudulent or not. Similarly, with critical or abnormal levels of any dissolved gas in the oil of a power transformer when performing dissolved gas analysis, abnormal load, abnormal heat, etc. The abnormal samples outside of a typical range are typically severely outnumbered by the number of samples within a typical range.

Statistical tools like SMOTE are then useful to synthetically generate more cases of the minority (less common) class. The process of synthetically generating more samples of a minority class is known in the artificial intelligence field as *over-sampling*. Similarly, if there is a vast amount of data, the opposite is possible as well by removing samples of the majority class in a process called *under-sampling*.

In this paper, we created a methodology for generating synthetic timeline data inspired in the algorithm of SMOTE [3]. SMOTE uses k-nearest neighbors (i.e., KNN) to determine where the new samples will be generated in the feature space. In simple terms, to use SMOTE we define a '*k'* (number of neighbors per data point). Then, the algorithm iterates over each data point obtaining the *k* nearest neighbors using a measurement of distance and calculates a vector between the analyzed datapoint and the selected neighbors. Finally, it randomly selects a number between zero and one, and multiplies that number by the calculated vector. The product of that multiplication results in the coordinates for a new synthetically generated sample. Figure 5 shows the process of starting with the original dataset, then generating synthetic samples using SMOTE, and finally the resulting dataset. As with many other techniques, several variations of SMOTE have emerged since its conception.

4.1. Proposed Methodology

We created a methodology to generate synthetic time series datasets based on the previously described process. A broad simplification of our technique is described in the pseudocode shown below in Algorithm 1.

```
1.  mult = 1 # fixed multiplier
2.  procedure Synthetic_Data_Generation():
3.      time_series = get_original_dataset()
4.      new_ts = [] # new synthetic time series list
5.      for each p in time_series:
6.          diff = (p.next - p)
7.          rand = random_number() # range [0,1]
8.          new_p = p + (diff * rand * mult)
9.          new_p = apply_filters(new_p)
10.         new_ts.add(new_p)
11.     return new_ts
```

Algorithm 1. Proposed synthetic data generation methodology

The basis of our data synthesis algorithm is to iterate over each data point in the time series, calculate a vector with the difference between the next data point and the current datapoint. Then multiply that vector by a random number (between zero and one), and by a configurable multiplier. Finally, we apply domain specific filters and add the newly generated datapoint to the new time series. These filters vary depending on the specific analyzed data, for example, in the context of DGA, we do not allow synthetically generated values below 0 ppm because negative particles per million are not defined in this context.

In our algorithm, we use *k*=1 and, since order is fundamental in time series datasets. Our algorithm takes the next datapoint as the nearest neighbor, as opposed to the calculated nearest neighbor used in SMOTE.

4.2. Configurable Multiplier

For our methodology we have also added the multiplication by a fixed multiplier which provides more flexibility to the data generation process. This multiplier describes how close will the generated dataset be to the original dataset.

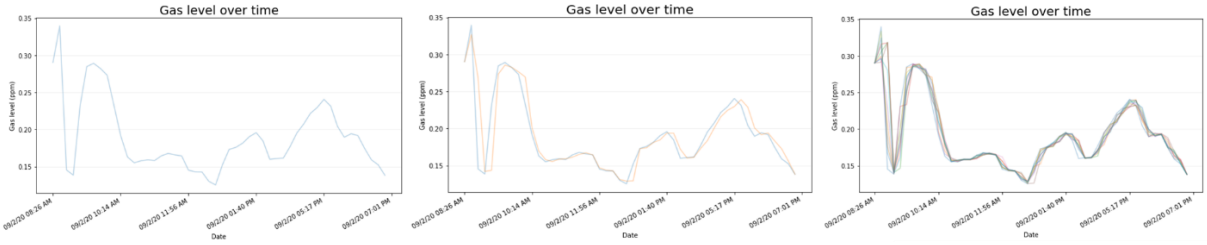


Figure 6. Left: Original time series; Center: Original time series plus one synthetic time series; Right: Original time series plus ten synthetic time series

Figure 6 shows the previously described technique in action. The graph on the left, describes the original time series dataset in blue. The graph in the center, shows the original time series in blue as well as a synthetically generated dataset in orange. The graph on the right, shows 10 synthetic time series together with the original. All of which vary around the original time series. In these examples the multiplier is set to one.

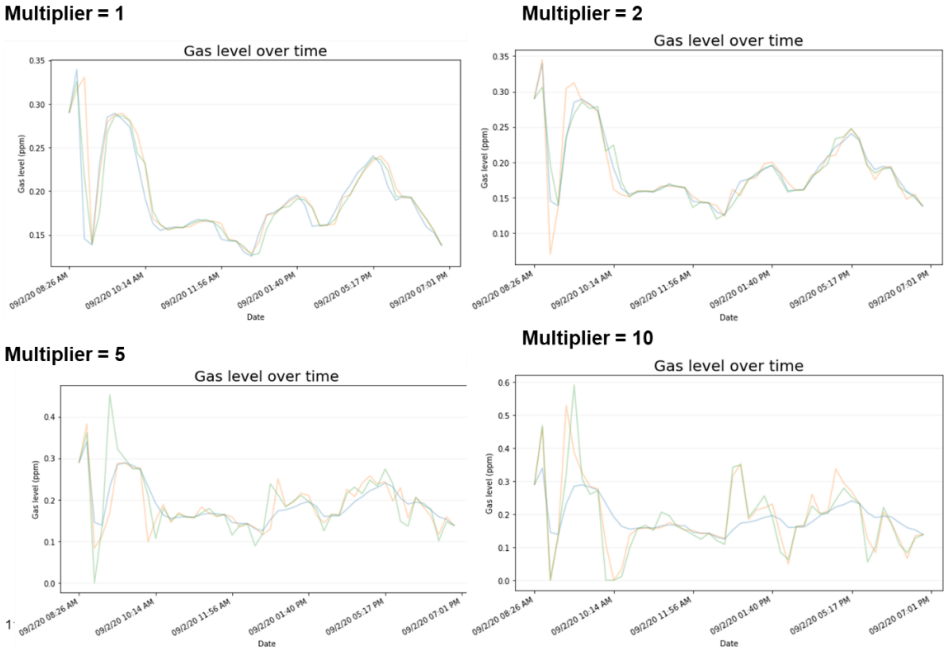


Figure 7: Several configurable multiplier values

Similarly, Figure 7 shows how different configurable multiplier values will allow for more flexibility regarding how far from the original dataset are created the synthetic time series. Overall, this methodology allows to generate high-fidelity synthetic time series data based on real time series readings in a flexible way. This methodology can also be repeated as many times as needed to generate more data.

4.3. Synthetic Data Validation

To validate if our synthetically generated dataset behaves in a realistic manner in comparison to how a real transformer would behave, we use Inverted Kolmogorov-Smirnov D Statistic (K-S statistic) to

compare a real dataset to our synthetically generated datasets. Inverted Kolmogorov-Smirnov D Statistic is a nonparametric test of the equality of continuous, one-dimensional probability distributions or samples [4]. In this statistic, the closer the value is to one, the more similar the two samples are.

Table 1 shows the results when using two different configurable multipliers (i.e., one, and ten) and their respective K-S statistic above. As shown in the table, when we set the configurable multiplier to '1' we get a synthetic time series that behaves very similar to real-life transformer DGA readings as opposed to setting the multiplier to '10'.

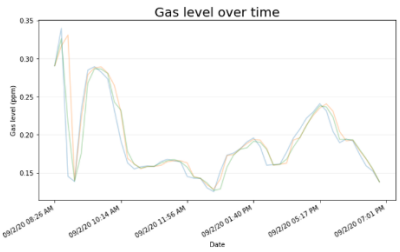
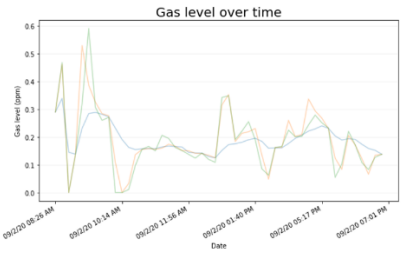
Configurable Multiplier	Multiplier = 1	Multiplier = 10
Inverted Kolmogorov-Smirnov D statistic	0.9948	0.8816
Graphical Representation		

Table 1. Inverted Kolmogorov-Smirnov D statistic results using different configurable multipliers

5. TRAINING MACHINE LEARNING MODEL FROM SYNTHETIC DATA

One of the main advantages of creating synthetic data, is opening the possibility to generating machine learning models that can accurately predict future values, among other possible uses. In the context of DGA, it would be beneficial to predict future values of DGA based on historical values so that, if abnormally high values are predicted in the future, the utility that owns the transformer can preemptively investigate the reason why it will produce these gases, as opposed to reacting after the gases have already been generated.

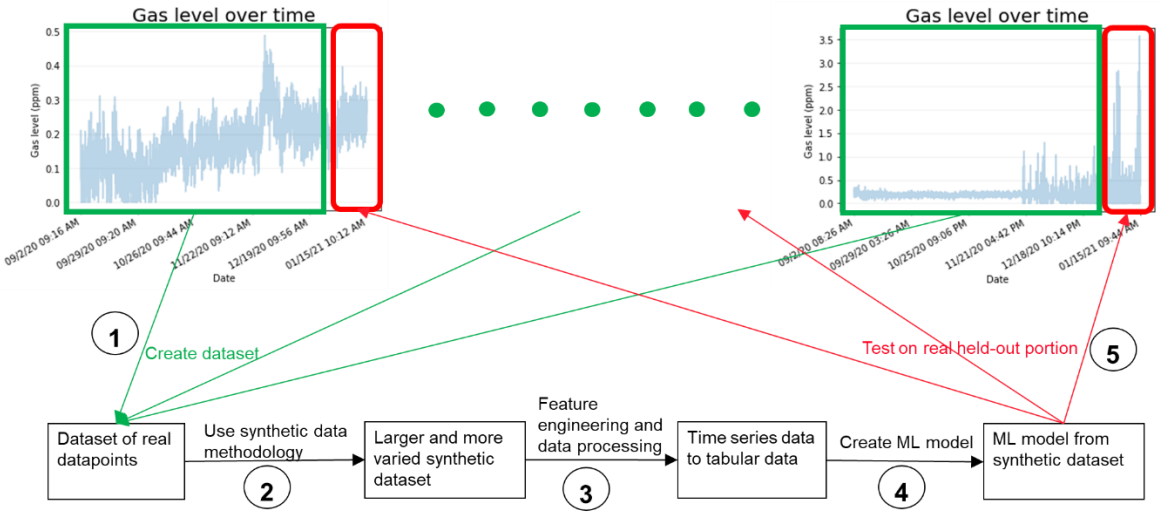


Figure 8. Experiment methodology

To evaluate the usefulness of using synthetic data in the generation of machine learning models that can predict real data, we followed the experiment methodology described in Figure 8. For each real dataset, we took the first 90% of the time series and used these datapoints to create a dataset of real datapoints as shown in step 1 (i.e., white circled '1' in Figure 8). Once we have the dataset of real data, we used

our synthetic data methodology (Section 4) to create a body of synthetic data, which we will later use to train a machine learning algorithm (step 2). In step 3, we performed feature engineering (Section 5.1) in the synthetic data collection and class value transformations (Section 5.2). We then use this tabular dataset to train several machine learning models to compare amongst themselves (step 4). Finally, we tested the trained models to check how good was their prediction in the held out 10% of real data (step 5). Using this 90/10 data separation methodology, allowed us to avoid data leakage of the training data to indirectly contaminate the model training. The result is the models properly predicting the held-out portion without any preconceived notion of how the held-out portion should behave.

6. DATA PREPROCESSING AND FEATURE ENGINEERING

Once we have generated a synthetic data collection from the real transformer data, we need to transform the synthetic data into tabular form to train the algorithms. With this goal, we used a *rolling window* [5] of last 50 readings. Figure 9 shows a simple example of how to transform a time series data into tabular form using rolling window, where each row takes the following *X* consecutive datapoints, and the *class value* (i.e., the value to be predicted) is the last value of the window. In the example of Figure 9, row 1 would be composed of three datapoints, the first two will be features of the machine learning model, and the third datapoint will be the value to predict/learn. We aim for the model to learn the patterns in the data of when the readings increase or decrease based on historical data shapes.

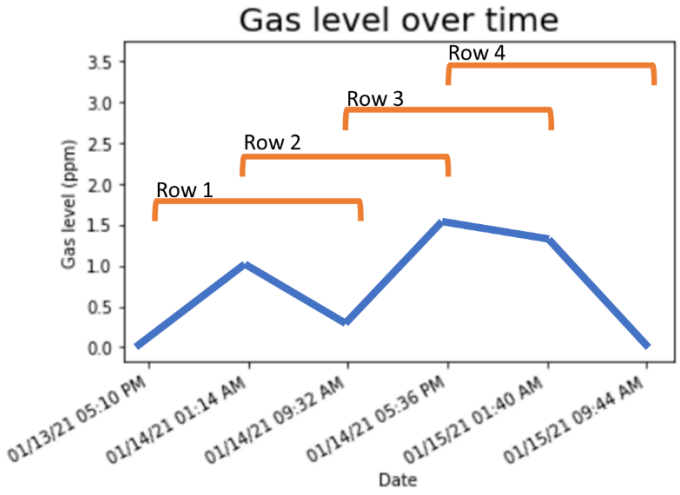


Figure 9. Example of rolling window algorithm

Besides the aforementioned features, we also included important information calculated in each row with the purpose of creating a richer feature space, as follows:

- Trend related measurements:
  - Slope
  - Intercept
  - R-value
  - P-value
  - Standard error
  - Standard deviation
  - R squared value
- Number of outliers in the window
- Abnormal value in the window

6.1. Class Value Transformations

To train our machine learning models we evaluated two families of models: Classification models and Regression models.

6.1.1. **Classification models:** Classification models refers to the machine learning algorithms whose outputs is in the form of a class. Thus, we needed to find a way to transform continuous DGA values into classes. With this goal, we binarized the DGA readings by using Table 1 from IEEE C57.104-2019 (shown as Table 2 in this document), which indicates the 90th percentile of DGA values from a large body of transformer data. If the DGA reading in our dataset is below the 90th percentile, we classified it as a "Normal" reading, and if it is above the 90th percentile, we classified it as an "Abnormal" reading. For example, if we have real transformer data from a transformer that is over 30 years old and whose oxygen to nitrogen ratio is below 0.2, and we are binarizing the Methane readings, we would convert the class value of each row to be "Typical" if the Methane ppm level is below 110, and "Abnormal" if it is above 110 ppm as shown in Table 2.

Table 1—90th percentile gas concentrations as a function of O_2/N_2 ratio and age in µL/L (ppm)

		O_2/N_2 Ratio ≤ 0.2				O_2/N_2 Ratio > 0.2			
		Transformer Age in Years				Transformer Age in Years			
		Unknown	1 – 9	10 – 30	>30	Unknown	1 – 9	10 – 30	>30
Gas	Hydrogen (H_2)	80	75		100	40	40		
	Methane (CH_4)	90	45	90	110	20	20		
	Ethane (C_2H_6)	90	30	90	150	15	15		
	Ethylene (C_2H_4)	50	20	50	90	50	25	60	
	Acetylene (C_2H_2)	1	1			2	2		
	Carbon monoxide (CO)	900	900			500	500		
	Carbon dioxide (CO_2)	9000	5000	10000		5000	3500	5500	
NOTE—During the data analysis, it was determined that voltage class, MVA, and volume of mineral oil in the unit did not contribute in significant way to the determination of values provided in Table 1.									

Table 2. from IEEE C57.104-2019 used to classify DGA readings into Normal and Abnormal values

6.1.2. **Regression models:** Regression models refers to the machine learning algorithms whose output is a continuous value. Therefore, since DGA readings are given by the sensor in a continuous spectrum, we did not modify the class value in this portion of the study.

Following the creation of the tabular data, we were then able to train two sets of machine learning models: Regression and Classification. The results of this evaluation are described in the next section.

7. MODEL PERFORMANCE

Following the previous description of the experiment conducted, we will present the results in two sections following the two families of machine learning models evaluated: Classification models and Regression models.

7.1. Classification Models

We trained a sample of six machine learning classification models as follows:

- Gaussian Naïve Bayes
- Linear Discriminant Analysis
- Decision Tree Classifier
- K-Nearest Neighbors Classifier
- Logistic Regression
- Support Vector Machine

In the evaluation of classification models, we found a problem very common to this type of datasets, which is the problem of unbalanced data. This happens when we are trying to predict behavior that is uncommon (in our example, having a lot more "normal" readings than "abnormal" readings in our DGA dataset). To tackle this problem, we applied under sampling [6]. In simple terms, under sampling is a data manipulation technique were we randomly remove samples of the majority class with the intention

of having an equal number of samples for both classes. If we do not perform this redistribution, there is a risk that the majority class overpowers the minority class, and even if the models perform poorly, they would appear to be high-performing because of this imbalance.

7.1.1. Cross Validation

To validate which model is best to select to predict our held-out data, we first compare the performance of the classification models within the training data through a methodology known as 10-fold-cross-validation. In this validation technique, we segregate the training data into ten non-overlapping sections, train a model with nine of them and predict the data in the tenth fold. We then rotate the validation fold ten times and finally average the results from the ten predictions. This step is performed using the *training* portion of the dataset only. Figure 10 shows the results from the cross validation of the six models being tested.

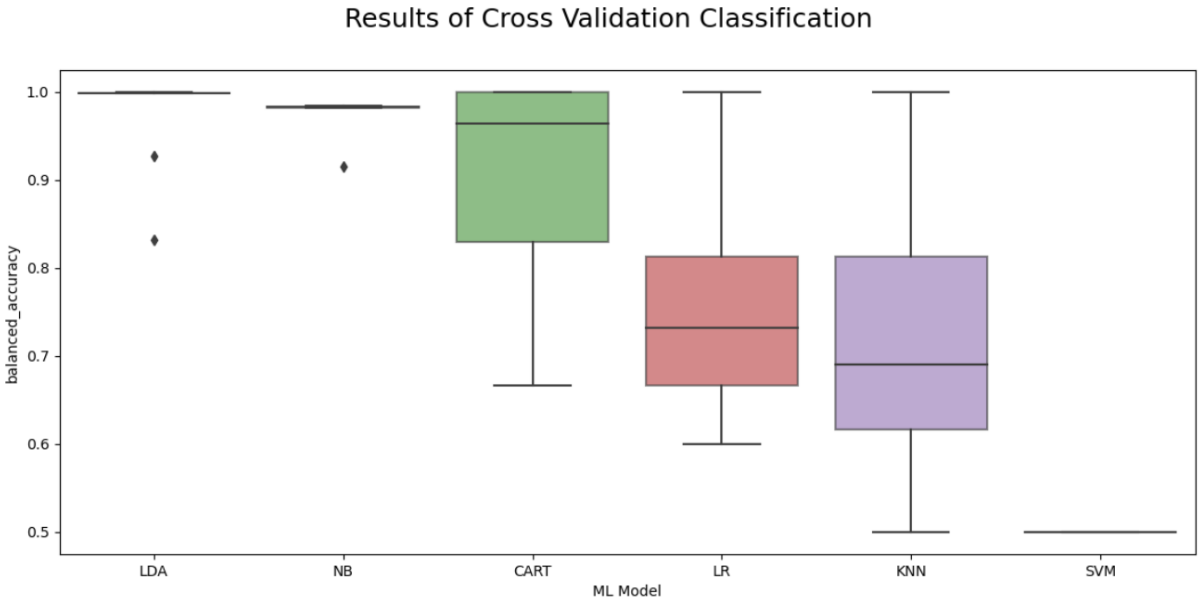


Figure 10. Cross validation of classification models

As seen in Figure 10, Linear Discriminant Analysis is the highest performing model when evaluated within the training dataset. It is important to understand that this is not a definitive answer as to which model should be used for future experiments since the performance of models varies depending on the data fed to them, data preparation, data latency, among others. Given that Linear Discriminant Analysis was the best performing model in this dataset, we proceed to use this model to predict and evaluate its performance in the held-out real data.

7.1.2. Held-out Results

We trained a Linear Discriminant Analysis model using all the synthetic data available and used that model to predict the held-out real dataset. Our results are as follows:

Accuracy: 98.03%

Accuracy describes the overall hit rate of our model. It answers the question "How many times did our model correctly predict the future DGA value?". However, given that this is an imbalanced dataset, it is important to give an equal weight to both the minority and majority classes (i.e., equal importance to predicting "normal" and "abnormal" readings). Thus, we also report the balanced accuracy:

Balanced accuracy: 93.97%

The *balanced accuracy* is defined as the average of recall obtained on each class, which gives a better perspective of how all classes are predicted. It is an accuracy measurement that considers the number of measurements of each class equally. Other important measurements to better understand the details of our model's performance are the following:

Normal Readings: Precision: 99.75% Recall: 98.22%
Abnormal Readings: Precision: 54.24% Recall: 89.72%

Precision answers the question "From the readings that our model said were from this class, how many were actually from this class?". In our case, normal readings have a great precision, but abnormal readings do not. Meaning that, our model still reports ~45% of false positives for the abnormal readings. Further work is required to improve this measurement.

Recall answers the question "From the readings that were actually from this class, how many did our model say were from this class?". We overall obtained good results for both normal and abnormal readings with a 98.22% and 89.72% respectively. Overall, these are very positive results with the caveat of the precision of abnormal values being relatively low, which can be improved by further refining the model training process.

7.2. Regression Models

For regression models we followed a similar experimental approach with the main difference that our model will be predicting the exact DGA value, as opposed to "normal" and "abnormal" values used in the classification models. We trained ten different samples of regression models as follows:

- Linear Regression
- Lasso
- Ridge
- Elastic Net
- Decision Tree Regressor
- Ada Boost Regressor
- Gradient Boosting Regressor
- Bagging Regressor
- Rando Forest Regressor
- XGB Regressor

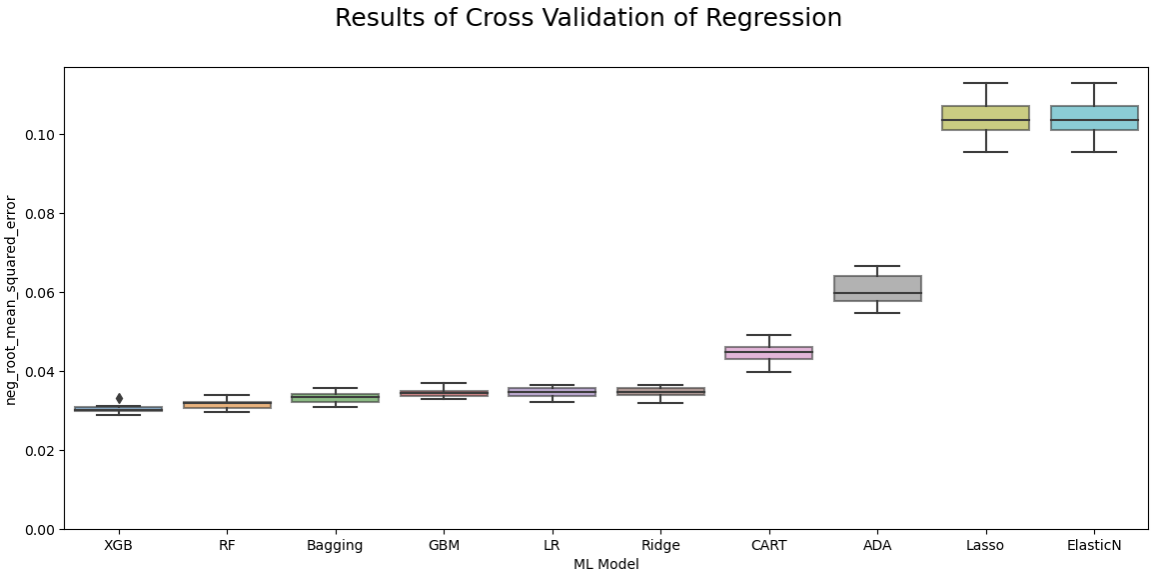


Figure 11. Cross validation of regression models

7.2.1. Cross Validation

As explained in Section 6.1.1, we used 10-fold-cross-validation to validate the performance of the regression models within the training data. Figure 11 shows the results of the cross-validation process where the y-axis shows the root mean squared error (RMSE). The RMSE describes the average error between the predicted and the actual value.

According to our results, XGBoost Regressor seems to be the best, although many other regressors perform very similarly. Thus, we will use XGBoost Regressor to predict the held-out real data.

7.2.2. Held-out Results

We trained an XGBoost Regressor model using all the synthetic data available and predicted the held-out real dataset. Our results are as follows:

Root mean squared error (RMSE): 0.16

This indicates that our model had an average error of 0.16 ppm when predicting Acetylene values (e.g., if the actual Acetylene value was 1.5 ppm, our model could have predicted 1.66 ppm or 1.34 ppm). Overall, we consider this to be very good performance for an initial validation of a model that can be further improved in future iterations. These experiments were performed with Acetylene readings since Acetylene is the most important gas to detect high-priority problems. However, the same data processing, feature engineering, data synthesis and experiment design can be used for all other gases.

8. CONCLUSIONS

This study proposes a methodology for generating synthetic time series data applied to dissolved gas analysis in power transformers. The generated synthetic data is later used to train machine learning models which can predict dissolved gas analysis. We show how our model trained with synthetic data can predict Acetylene values with an accuracy of 98.03% and with a 0.16 ppm root mean squared error. Overall, this study shows a promising technique to generate synthetic transformer data, which would allow the researchers in the power industry to have access to a vast body of data. This data can be used to further the creation of artificial intelligence techniques to improve the design, manufacturing, and monitoring of electric components in the power grid.

BIBLIOGRAPHY

1. Cheim, L.; Tehini, L.; Pearce, S., “Artificial intelligence/machine learning driven assessment system for a community of electrical equipment users”, US Patent #1,262,739B2, March 2022
2. Mullins, Craig S. (February 5, 2009). "What is Production Data?". NEON Enterprise Software, Inc. Archived from the original on 2009-07-21.
3. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique", Journal of Artificial Intelligence Research, 2002.
4. Weiss, M. “Modification of the Kolmogorov-Smirnov Statistic for Use with Correlated Data”, Theory and Method, 1976, pp 872-875
5. Zivot, E. and Wang, J. "Rolling Analysis of Time Series", Modeling Financial Time Series with S-Plus, 2003, pp 299–346
6. XY Liu, J Wu, ZH Zhou, “Exploratory undersampling for class-imbalance learning”, IEEE Transactions on Systems, Man, and Cybernetics, 2009, Volume 39, Issue 2