Simulated Datasets Generator for Testing Data Analytics Methods

Serhii Toliupa, Anna Pylypenko, Oleh Tymchuk and Oleksii Kohut

Taras Shevchenko National University of Kyiv, 64/13 Volodymyrska St, Kyiv, 01601, Ukraine

Abstract

This article explores the role of benchmark datasets in testing data analysis tools. The use of synthetic data generators is motivated by the need for scaling the size of training datasets, filling data gaps, and safeguarding data confidentiality. Existing research in this field emphasizes the importance of applications in various domains, such as fraud detection, healthcare, and computer graphics. The efficacy of constructing Gaussian distributions using the Box-Muller transform is investigated, while limitations in generating extreme values are highlighted. The integration of specialized distributions is proposed to address this gap and enhance dataset variability for improved performance in data analytics methods. The article presents a synthetic data generator capable of producing datasets for effective evaluation of machine learning methods. Practical tests demonstrate the software's effectiveness in creating test datasets with controlled variations. Four different tests were conducted, each with three different variants: 1) normal distribution parameters, namely standard deviation, 2) class imbalance, 3) missing values, and 4) outliers. The generated datasets were used to conduct controlled tests of logistic regression. Performance evaluation of the logistic regression model employed metrics such as Precision, Accuracy, Type 1 Error, Type 2 Error, and F1-Score.

Keywords [1]

Benchmark datasets creation, synthetic dataset generator, analysis methods, extreme values.

1. Introduction

The modern world generates an enormous volume of data every day, ranging from sensor data to textual information. Data analytics tools require the availability of realistic and representative datasets for testing that would reflect contemporary challenges in data processing. The relevance of this topic is also driven by the development of artificial intelligence and machine learning: the high demand for the development of machine learning and artificial intelligence algorithms necessitates data for training and evaluating these algorithms. Creating realistic datasets becomes critically important for precise assessment and comparison of various methods. The complexity of this problem is associated with its interdisciplinary nature. Analytical tools and methods are used across various domains, including medicine, finance, biology, ecology, and others. The creation of benchmark datasets can also contribute to addressing ethical concerns related to data privacy and security by developing anonymized or synthetic data. So, the creation of benchmark datasets for testing data analytics tools is essential for advancing science and technology in this field and ensuring their practical utility across diverse domains.

There are a lot of published research results that provides comprehensive insights into the process of creating benchmark datasets and its significance. In [1], the authors introduce the concept of benchmark metrics in machine learning for scientific purposes and review existing approaches. They underscore that the selection of the most appropriate machine learning algorithm for scientific data analysis remains a significant challenge due to the potential applicability of machine learning frameworks and models, computer architectures. The results of the research by authors Krizhevsky, Sutskever, and Hinton [2] were the first to demonstrate the profound influence of datasets on the

Dynamical System Modeling and Stability Investigation (DSMSI-2023), December 19-21, 2023, Kyiv, Ukraine
EMAIL: tolupa@i.ua (A. 1); anna.pylypenko@knu.ua (A. 2); oleh.tymchuk@knu.ua (A. 3); oleksii_kohut@knu.ua (A. 4)
ORCID: 0000-0002-1919-9174 (A. 1); 0000-0002-6343-4469 (A. 2); 0000-0002-9046-8015 (A. 3); 0009-0004-2203-3927 (A. 4)

learning outcomes of deep neural networks. It can be instrumental in comprehending the significance of benchmarking. The article [3] by M. Fernandez-Delgado, E. Cernadas, S. Barro and D. Amorim is a significant contribution to the field of machine learning and data classification, providing practical insights into the use of classifiers in real-world scenarios. The article explores various classification algorithms and compares their performance across different benchmark datasets, which can serve as a valuable resource for discussing the effectiveness of various analytical tools.

Each data generator program uses a unique approach to data creation. The article [5] presents a data generator designed to fill in gaps that may exist in other programs. The developed system allows users to customize and create known statistical distributions to achieve the desired outcome. Additionally, it offers real-time data behavior visualization to analyze whether they possess the characteristics necessary for effective testing. In the articles [6, 7], the authors provide an overview of the design and architecture of the Information Discovery and Analysis Systems (IDAS) Data Set Generator (IDSG), which enables a fast and comprehensive evaluation of IDAS. IDSG generates data using statistical algorithms, rule-based algorithms, and semantic graphs that represent interdependencies between attributes. To illustrate this approach, an application for credit card transactions is used. Sran Popić et al. [8] provided a brief overview of various types of generators in terms of their architecture and anticipated usage, as well as listed their advantages and disadvantages. They also presented a review of the data generation algorithms used and best practices in various domains.

Researchers in this field has attempted to assess the utility of synthetic data generators using various evaluation metrics. However, it has been found that these metrics lead to conflicting conclusions, complicating the direct comparison of synthetic data generators. In their study, Fida Kamal Dankar and colleagues [9, 10] identified four criteria for evaluating masked data by categorizing available utility metrics into different categories based on the information they seek to preserve: attribute fidelity, two-dimensional parameter fidelity, population fidelity, and application fidelity. In the article [11], the authors have introduced several novel and efficient methods and multidimensional data structures that can enhance the decision-making process in various domains. They have examined online range aggregation, range selection, and weighted range median queries; for most of these, data structures and techniques are presented that can provide answers in near-polynomial time.

In practice, obtaining real data can be challenging due to confidentiality issues. Additionally, real data may not conform to specific characteristics required for evaluating new approaches under certain conditions. Given these constraints, the use of synthetic data becomes a viable alternative to supplement real data in various domains. For example, in the article [12], the authors described the process of generating synthetic data using the publicly available tool Benerator to mimic the distribution of aggregated statistical data obtained from the national population census. The generated datasets successfully replicated microdata containing records with social, economic, and demographic information. Forensics also requires testing digital information tools. Thomas Göbel et al. [13] introduced the concept of a structure called hystck for creating synthetic datasets based on the ground truth. This framework supports automated generation of synthetic network traffic and artifacts of operating systems and software by simulating human-computer interactions. To preserve confidentiality, banks are unwilling to share fraud statistics and datasets with the public. To overcome these limitations, Ikram Ul Haq et al. [14] introduced an innovative technique for generating uniformly distributed synthetic data (HCRUD) based on highly correlated rules. This technique allows the generation of synthetic datasets of any size, replicating the characteristics of restricted actual fraud data, thus supporting further research in fraud detection. Access to medical datasets is also complicated due to concerns about patient confidentiality. The development of synthetic datasets that are realistic enough for testing digital applications is considered as a potential alternative that allows their deployment. Theodoros Arvanitis et al. [15] have devised a method for generating synthetic data statistically equivalent to real clinical datasets and have demonstrated that the approach based on Generative Adversarial Networks aligns with this goal. Thus, the concept of creating realistic medical synthetic datasets has been successfully validated. However, data quality issues exist both in real and synthetic data, with the latter reflecting real-world problems and artifacts created by synthetic datasets. The intellectual analysis of synthetic healthcare data represents a novel field with its unique challenges. According to Alistair Bullward et al. [16], researchers should be aware of the risks associated with extrapolating results from synthetic data studies to real-world scenarios and should evaluate outcomes using analysts who can review the underlying data. Synthetic data is frequently utilized in computer

graphics, which is used for training computer vision models, as mentioned in [17, 18]. In many industrial computer vision tasks, deep learning methods such as convolutional neural networks have been successfully employed, as indicated by the works [19, 20]. In recent years, generative adversarial networks (GANs) have been effectively utilized for generating new realistic images and manipulating them, as noted in the research by I. H. Rather and S. Kumar [21].

Therefore, the increasing availability and utilization of data analytics tools make the standardization of benchmark datasets an essential task for their further adoption across various fields. The challenge of developing more objective metrics and methods for assessing the utility of synthetic data remains unresolved. One of these problems is adequate tail-end modeling of probability density functions. Extreme values can significantly impact risks and outcomes in several sectors such as finance, insurance, climatology, engineering, among others. This article presents research in the field of creating synthetic data that aligns with real-world requirements and enables their effective use for testing various analytical tools. The primary focus is on comprehending and analyzing characteristics of this generator, especially in the tail areas of the Gaussian probability density function.

2. Mathematical methods

The generation of Gaussian distributions is foundational in various scientific and computational fields, playing a pivotal role in modeling natural phenomena and simulating random variables. To create a Gaussian distribution, it is suggested to use the Box-Muller transform [22]. It produces a pair of Gaussian random numbers using a pair of uniform numbers. The fundamental principle of the Box-Muller algorithm lies in its ability to generate pairs of independent standard Gaussian random variables from uniformly distributed random numbers. This method leverages the polar coordinate representation in a two-dimensional space to transform pairs of independent uniformly distributed variables into sets of normally distributed variables. By employing trigonometric functions and geometric interpretations, this algorithm constructs Gaussian values by utilizing the magnitude and angle derived from uniformly generated random variables. The algorithm to generate the Gaussian samples:

1. Generate sample using two distinct uniform random number generators, u_1 and u_2.
2. Apply the inverse cumulative distribution function (CDF) of the exponential distribution to u_1 ($\lambda = 1$):

$$r = \sqrt{-2ln(1-u_1)} = \sqrt{-2ln(u_1)}, \quad (1)$$

where r is the distance from origin for each sample.

For simplicity, u_1 is replaced by $1 - u_1$ since they are both uniform samples on $(0, 1)$.

3. Apply the inverse CDF of the uniform distribution on $(0, 2\pi)$ to u_2:

$$\theta = 2\pi u_2, \quad (2)$$

where θ is the angle of the sample.

4. Finally, determine the x and y using basic trigonometric calculations:

$$x = rcos(\theta), \quad y = r\,sin(\theta). \quad (3)$$

The Box-Muller algorithm generates two independent random numbers upon each execution. Each pair of generated numbers represents two independent random variables following a standard normal distribution. These variables possess a mean of 0 and a standard deviation of 1. The produced values can be utilized as required for various statistical or computational purposes. For instance, both numbers from each pair can be employed to generate pairs of normally distributed random variables. Alternatively, a single number from each pair might suffice if the need is for a singular normally distributed value. By combining both sets of generated values into a single dataset, a larger and potentially more diverse sample can be constructed, enriching the dataset used for testing machine learning models. This integration allows for a broader spectrum of data points, enhancing the robustness of the evaluation process and potentially fortifying the model's generalization capabilities.

The Box-Muller algorithm, while efficient in generating core values from the standard normal distribution, may necessitate additional methods to generate values from the tails of the distribution [23]. Extreme or outlier values that reside in the tails of the distribution are often critical for assessing rare events or evaluating the performance of algorithms. The generation of extreme values in random numbers can be achieved using specialized distributions. Some distributions, such as the exponential,

Weibull, Fréchet extreme value distribution, among others, directly model extreme values. The proposed algorithm incorporates the following steps to generate such values.

Continuation of the algorithm to generate the Gaussian samples:

5. Set the parameter c represents the shape parameter governing the tail behavior. It determines the shape and heaviness of the distribution's tails.

$c > 0$: indicates a distribution with bounded tails. It implies that the distribution's tails are bounded, and the probability of encountering extreme values decreases more rapidly than in a normal distribution.

$c = 0$: corresponds to the exponential distribution, where the tails are light, and the probability of extreme values decreases exponentially.

$c < 0$: indicates a distribution with heavier tails than the exponential distribution. This suggests that the probability of encountering extreme values decreases slower than in an exponential distribution.

6. Establish the probability density function of the distribution is given by the formula:

$$f(t;c) = \begin{cases} e^{-[1+ct]^{-1/c}} \cdot (1+ct)^{-1/c-1} & \text{for} \quad c \neq 0, \\ e^{-e^{-t}} \cdot e^{-t} & \text{for} \quad c = 0, \end{cases} \tag{4}$$

use the inverse transform method to generate extreme values:

$$t = \begin{cases} \frac{1}{c}(-ln(1-u)^{-c} - 1) \text{ for } c \neq 0, \\ -ln(-ln(1-u) \text{ for } c = 0, \end{cases} \tag{5}$$

where u is a random variable from a uniform distribution on the interval $(0, 1)$ and can be obtained as a fraction of random variables generated in step 1. Validate the generated extreme values to ensure they align with the expected tail behavior.

7. After obtaining the standard normal random variables and extreme values $z := concatenate(x, y, t)$, can move on to a normally distributed value with mathematical expectation μ and standard deviation σ:

$$\xi = \mu + \sigma z. \tag{6}$$

Depending on the practical task there might be a need to shift the distribution along the axis of values or change its scale. In such cases, it would be advantageous to employ the Generalized Extreme Value (GEV) distributions, which combine the Gumbel, Fréchet, and Weibull families, also known as type I, II, and III extreme value distributions [24]. These distributions offer flexibility in adjusting the positioning and scaling of the distribution to accommodate various scenarios and analyses.

3. Development of synthetic dataset generator

In the domain of data generation for machine learning research, a spectrum of tools and libraries, including MOSTLY.AI (https://mostly.ai/), Mockaroo (https://www.mockaroo.com/), and Scikit-learn (https://scikit-learn.org/stable/datasets/sample_generators.html), among others, is at researchers' disposal. These tools furnish a fundamental framework for the generation of synthetic datasets, enabling users to craft data with predetermined statistical distributions. Nevertheless, they exhibit inherent limitations. For instance, scikit-learn is primarily constrained by its capacity to generate synthetic datasets featuring a limited array of distributions and parameters, rendering it less suitable for the creation of intricate or verisimilar data representations. These tools primarily target numerical data and lack the specialized capabilities required for the synthesis of structured data types, such as textual information, categorical attributes, or time series. In the subsequent sections of this article, the utilization of synthetic data will be examined primarily in the context of testing machine learning methods, considering the noted shortcomings of existing tools.

3.1. Data model

The specific capabilities of a synthetic data generator may vary from implementation. However, such software should have the following main features:

- defining the name, description, and additional information associated with the model;

- describing all dimensions of the model, including their data type, name, description, and most importantly, the algorithm for generating the value;
- export synthetic data rows to a file or database based on the created model.

Each data model must have at least one dimension. Each dimension should be defined by the following characteristics: dimension name; data type (integer or real number, category, string); expression/formula that defines how the data will be filled in; additional options (presence of blank values/outliers). Values in columns can be either independent expressions or calculated based on values in other columns (while avoiding cyclic dependencies, where, for instance, expression A relies on the value from column B, and expression B relies on the value from column A). The data model serves as an abstraction of a dataset, comprising specifications that characterize the behavior of the data [5]. Figure 1 depicts an example of rudimentary data model illustrating the characterization of specific individuals for the purpose of analyzing patient ages. There are three dimensions:

1) Name: an informative dimension, intended more for identifying the string than for analyzing the data. Such strings are not useful in machine learning, but if it were real data, there would be a privacy issue. Given that this string is generated in a random manner, there is no need for concern regarding the utilization of personal data belonging to individuals. Since this dimension is defined without modifiers, the data in this row will always be present.

2) Age: dimension determines the patient's age. Its expression defines a random value in a normal distribution with a mean of 14.0 and a variance of 3.0. There are no blank fields.

3) IsAdult: dimension determines whether the patient is an adult. This is the only dimension that uses another dimension in its calculations. If the patient is over 18, this field is set to 1, otherwise 0. There are no blank fields.

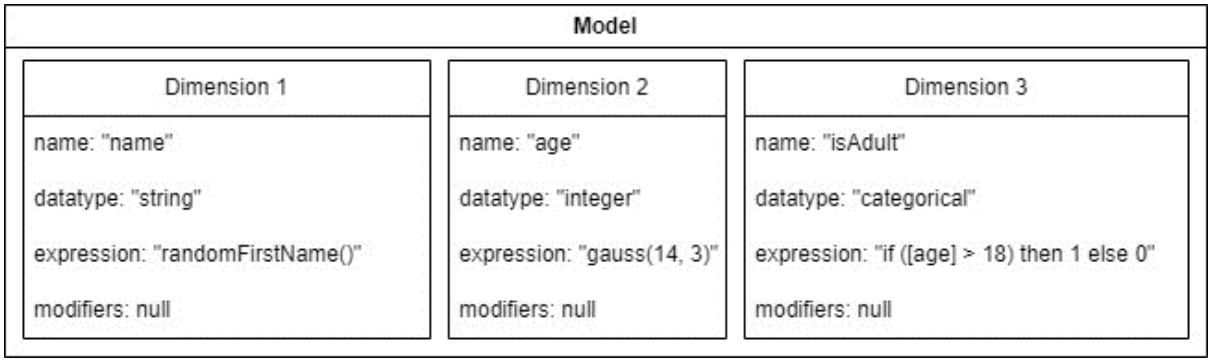

Figure 1: Example of data model

3.2. Modelling of the generator

Editing a data model includes editing general information about the model, such as the name, description, and availability to other users. Most importantly, in this mode, you can edit each dimension separately, with a real-time check for the correctness of the entered data. Dimensions can be added or deleted, but each model will always have at least 1 dimension with a line number. Each dimension must be given a unique name that will not match other dimensions within the model. All dimensions must be one of the defined data types (see Table1).

Table 1
Supported data types

Data type	Description	Examples
String	A sequence of symbols with variable length	"Oleksii", "Kohut", "test123"
Integer	$x \in \mathbb{Z}$	-2, -1, 0, 1, 2...
Real	$x \in \mathbb{R}$	0.5, 1.3e22, 3.14, -0.001
Category	Custom data type, choose one item from given list	(1, 2, 3), ("apple", "banana", "orange")
Boolean	Logical data type (example usage of category type)	(0, 1), ("true", "false")

The lists of operands and functions available for generation is described in Tables 2-4.

Table 2
The list of operand generators

Operand	Description	Usage example
+	Addition	x + y
-	Substraction	x – y
*	Multiplication	x * y
/	Division	x / y
^	Exponentiation	x ^ y
?	Conditional operator	(a = 0) ? x : y

Table 3
The list of random generators

Distribution	Arguments	Usage example
Uniform	$a, b \in \mathbb{R}$	Uniform (a, b)
Normal (Gaussian)	$\mu, \sigma \in \mathbb{R}$	Gauss (mu, sigma)
Cauchy	$x_0, \gamma \in \mathbb{R}$	Cauchy (x0, gamma)
Poisson	$\lambda \in \mathbb{R}$	Poisson (lambda)
Bernoulli	$p \in \mathbb{R}$	Bernoulli (p)
Categorical	$a_0, a_1 \dots a_z \in C$	Category ("a", "b", "c")
Weighted categorical	$a_0, a_1 \dots a_z \in C$, $p_0, p_1 \dots p_z \in \mathbb{R}$	Weighted_category (("a", 1.0), ("b", 2.0), ("c", 3.0))

Table 4
The list of function generators

Function	Arguments	Usage example
Modulo	x – value, $x \in \mathbb{R}$	abs(x); |x|
Sign	x – value, $x \in \mathbb{R}$	sign(x)
Sine	x – value, $x \in \mathbb{R}$	sin(x)
Cosine	x – value, $x \in \mathbb{R}$	cos(x)
Tangent	x – value, $x \in \mathbb{R}$	tan(x)
Exponent	x – value, $x \in \mathbb{R}$	exp(x)
Logarithm	x – value, p - base, $x, p \in \mathbb{R}$	log(x, p)
Natural logarithm	x – value, $x \in \mathbb{R}$	ln(x)
Square root	x – value, $x \in \mathbb{R}$	sqrt(x)
Cube root	x – value, $x \in \mathbb{R}$	cbrt(x)
Root of arbitrary base	x – value, p – base, $x, p \in \mathbb{R}$	nroot(x, p)

3.3. Usage scenarios: generating test datasets

To verify usefulness of such a software, practical tests were performed with the help of the program. A data model was specified in the generator and multiple tweaks were applied to it. The data set was created using the proposed algorithm to generate the Gaussian samples (1) – (6). The default model is described as follows:

- two classes;
- two dimensions: one dimension for the class, another for the value;
- value is generated depending on class with the help of ternary operator and normal distribution;
- no outliers, no blank values.

The dataset was specified on the data model tab. For each test a CSV file with 1000 entries was generated. Then, with the help of Python, graphs for this data were drawn. This provides a visual clue about how the change affects the data. A total of four different tests were performed, each with three different variations.

3.3.1. Changing parameters of normal distribution

In this test, the standard deviation of normal distribution for both classes were changed. The values chosen are 5.0, 3.0, 1.0. The expected result of such change is that values would become less dispersed across the axis. The result of the software for this case is presented in Figures 2-3.

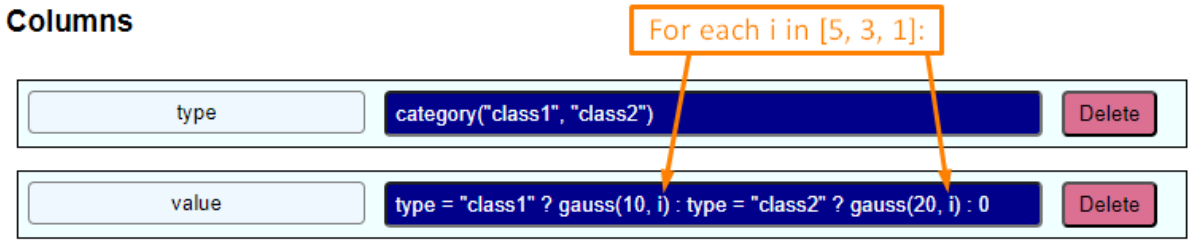

Figure 2: Data model options with standard deviation changes

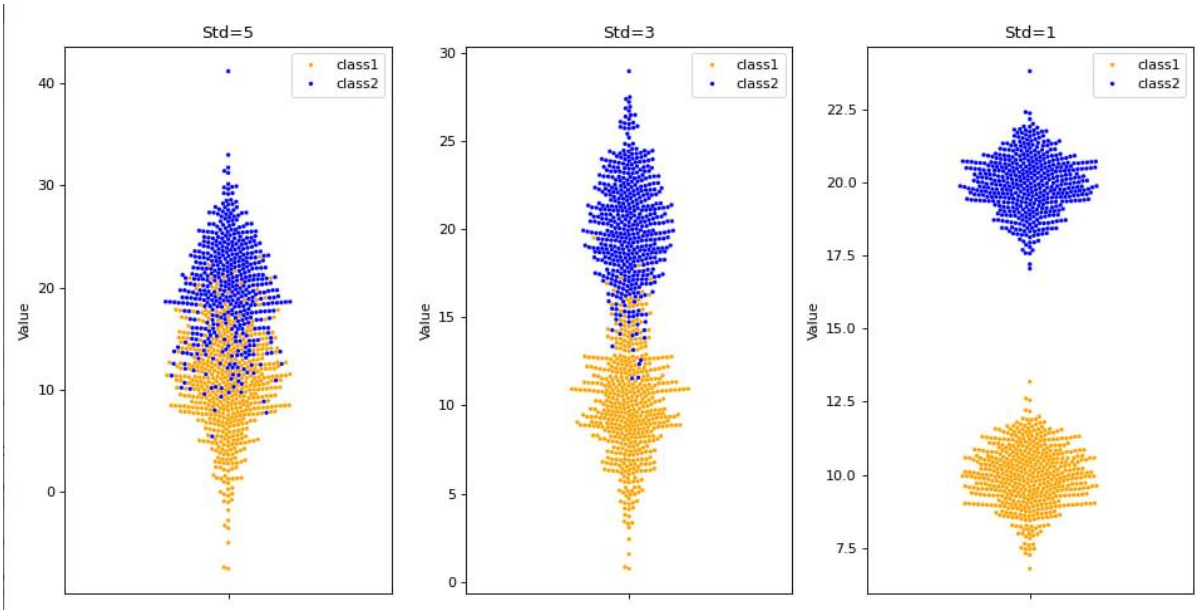

Figure 3: Resulting data graphs with standard deviation changes (original.csv, std3.csv, std1.csv)

For the current test, a swarm plot was used. The color represents the class of the item, and its position on the Y axis represents the value. With each next plot the values are getting more packed around central value, confirming that standard deviation is indeed changing.

3.3.2. Changing class distribution

Here, a weighted category was introduced in place of default category. For class 1, the probability of appearance will decrease each time, and for class 2 it will increase. The rate of change is 10%. So, the first distribution of classes will be 50-50%, then 40-60%, then 30-70% (Figures 4-5).

As can be seen on the Figure 5, the amount of class2 entries in getting higher with each next picture, confirming that this feature works.

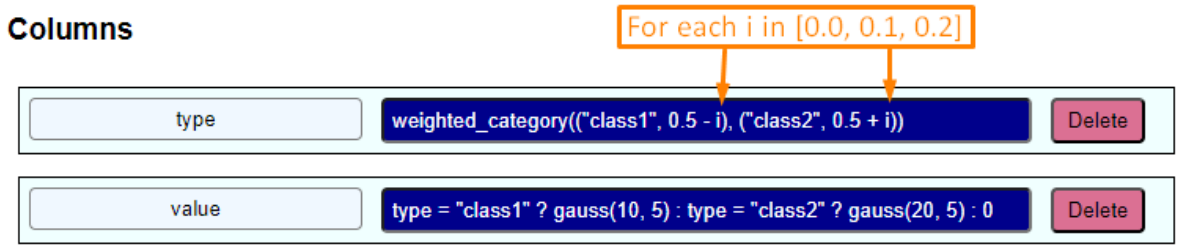

Figure 4: Data model options with class distribution changes

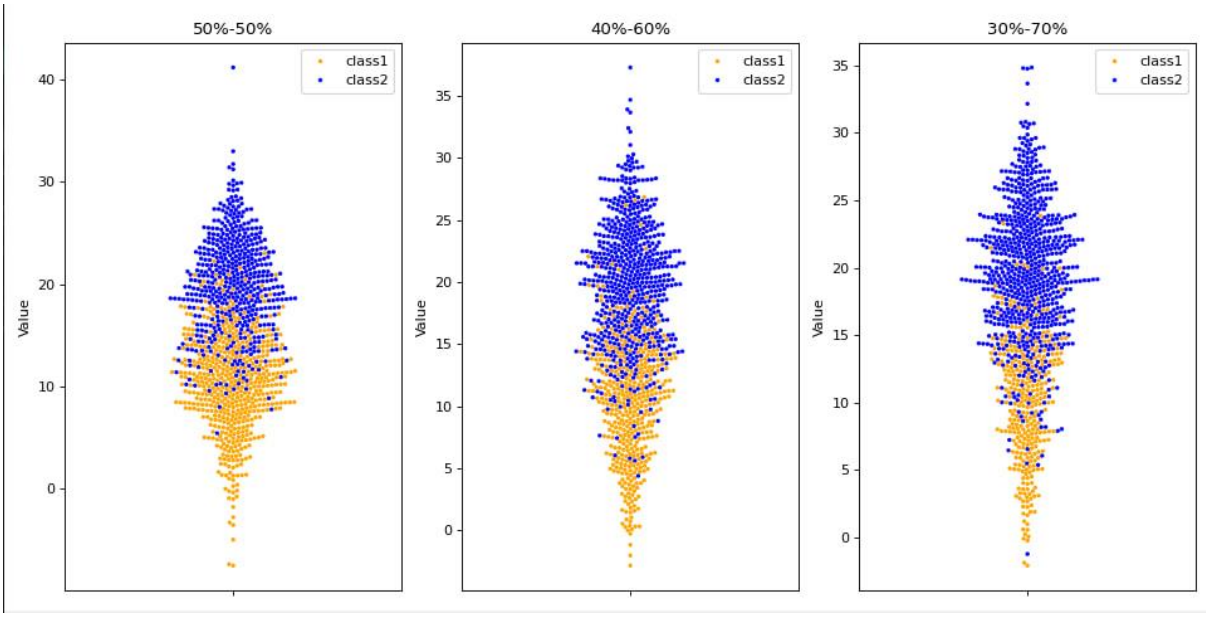

Figure 5: Resulting data graphs with class distribution changes (original.csv, 40-60.csv, 30-70.csv)

3.3.3. Adding outliers

To use outliers, an additional dimension was added. A random value is calculated with the help of uniform distribution, and if it's less than certain threshold (which equals probability of such event), then the value would be multiplied by 5 (Figures 6-7). Otherwise, the expected value will be placed.

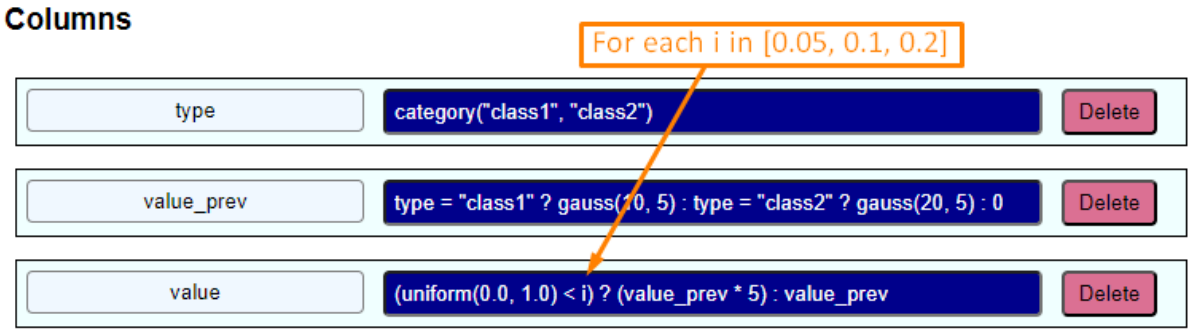

Figure 6: Data model options with added outliers

It's clear that with the increasing probability of outliers appearing, number of outliers will be bigger. Also, it's worth noting that class2 outliers reach higher values than class1 because of bigger base value. This creates additional separation between class1 and class2 in the higher values.

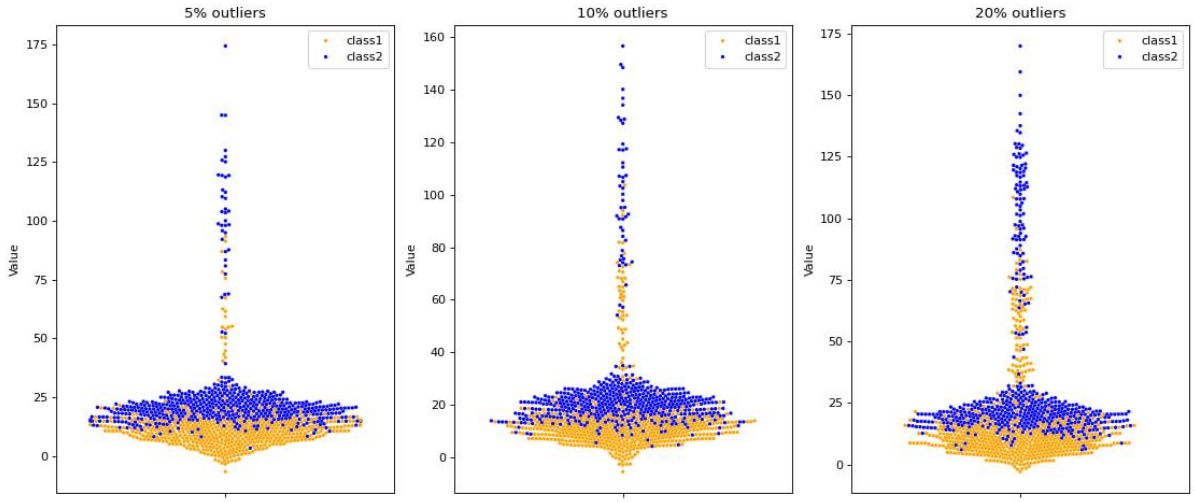

Figure 7: Resulting data graphs with outliers (outliers5.csv, outliers10.csv, outliers20.csv)

3.3.4. Setting up missing values

It's possible to make missing values in the dataset in a similar manner to outliers, with the help of an additional dimension. The only change is that missing() function is used instead of multiplication (Figures 8-9).

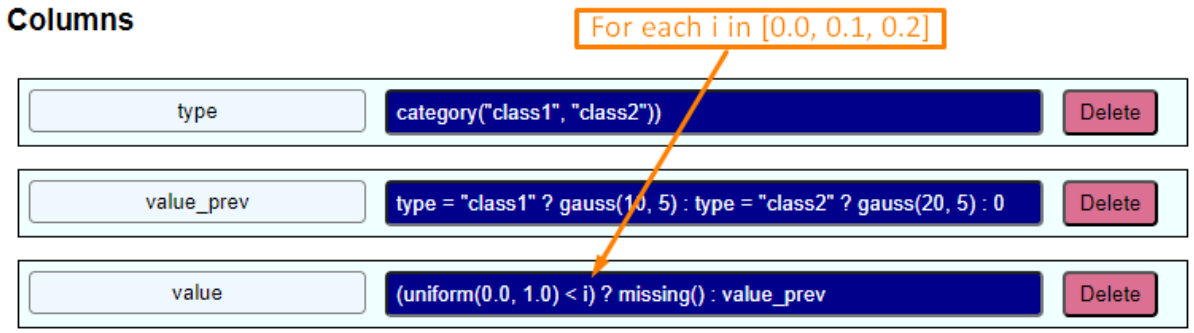

Figure 8: Data model options with blank values

For this test, an event plot was used instead of swarm plot. Colored lines represent objects generated (from 1 to 1000) with corresponding class. Each black line represents a missing value.

3.4. Software description

3.4.1. Module description

Software implementation of this synthetic dataset generator (lexData) consists of the three main parts: tokenizer, parser, and calculator. The input of this system is text expression describing formula for the dimension value. Figure 10 depicts how the text input can be processed into result:

In this simple example, there are multiple steps. In the first step, the input string is parsed into multiple tokens. A token is an atomic part of the mathematical formula. A single plus sign is a token, but a whole number or variable name is also a token, since we can't just divide the number into digits. After that, the sequence of tokens is handled by the parser, which builds a complete function. After passing concrete values for parameters (which may be other dimensions) specified in the function, the final value can be calculated and returned as the result. However, there are many more details such as verifying if referenced function exists or if there is no circular dependency. The output of calculation is just a single number or category, depending on the expression specified.

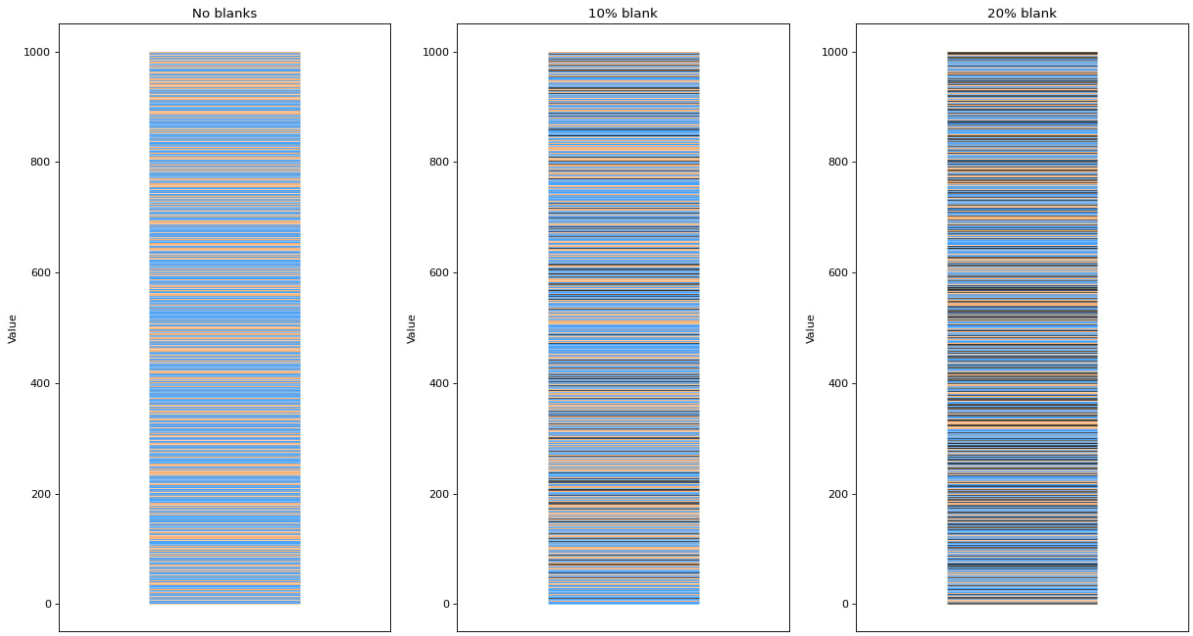

Figure 9: Resulting data graphs with blank values (original.csv, blank10.csv, blank20.csv)

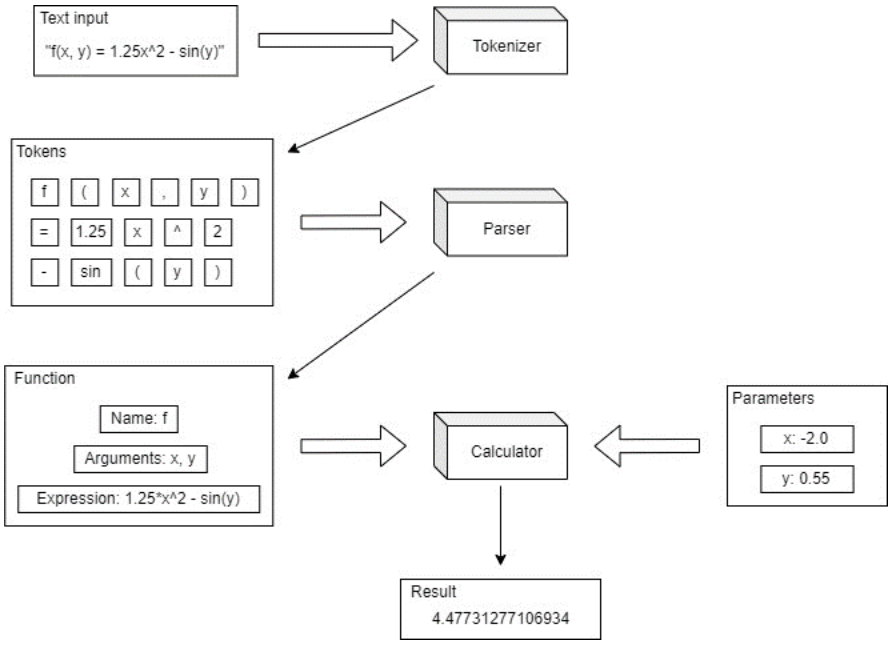

Figure 10: Module Interactions: Inputs and Output

3.4.2. Constraints and decisions

The .NET Framework and programming language C# were used by the designer of generator software. The generator core is implemented on top of custom-created library for parsing and evaluating math expressions, called lexCalculator. To integrate it with this software, it was modified to support such functions:

- generating random distributions;
- string data types;
- logical expressions;
- complex data types and lists.

With the help of NUnit, a unit testing was performed on this software. Most of the possible test cases were checked and tests provided more than 90% of code coverage.

The system was tested on the following system specifications:

- Windows 10 Pro;
- Intel(R) Core(TM) i5-8350U CPU;
- 16GB RAM;
- NVMe SK hynix 256GB SSD.

With such specifications, it was found out how the generator performed on the different datasets. The following table describes how many rows were generated per second on average for the specified dataset. Each dimension is just a simple switch between classes with normal distribution calculation:

As can be seen in Table 5, the number of classes doesn't affect performance of generation too much, except for the parsing stage, where more classes mean there are more possibilities to handle. As for the dimensions, these directly affect the performance, but that also depends on the expressions specified for these dimensions.

Table 2
Performance testing of the software

Dataset description	Rows/s
2 classes, 2 dimensions	16534 rows
2 classes, 4 dimensions	10233 rows
2 classes, 8 dimensions	7610 rows
3 classes, 2 dimensions	15669 rows

4. Discussion

Generators of synthetic datasets represent a potent tool for conducting controlled experiments and investigating the performance of machine learning methods in various scenarios. They facilitate an enhanced understanding of the capabilities and limitations of these methods. For instance, in this study, the discussed synthetic datasets with known characteristics and distributions were utilized for the purpose of conducting controlled tests of logistic regression, as illustrated in Table 6. These benchmark datasets can encompass diverse data complexities, such as class imbalance, missing values, and outliers, thereby aiding in assessing how effectively logistic regression operates under different conditions and whether additional tuning is necessary. To assess the effectiveness of the logistic regression model, the following metrics were employed:

1. *Precision* is the ability of the classifier not to label as positive a sample that is negative. In table 6 this metric provides an assessment of the model's overall precision, considering the weights of each class based on their distribution in the dataset. This allows for accounting for class imbalance, where one class may have significantly more instances than others.

2. *Accuracy.* This metric indicates the accuracy of a classification model, measuring the overall percentage of correct predictions (both positive and negative) out of the total number of examples in dataset:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i), \quad (7)$$

where $\hat{y}_i$ is the predicted value of the i-th sample, y_i is the corresponding true value, $1(x)$ is the indicator function.

3. *Type 1 Error*. This metric indicates the precision of the model for the class denoted as "class1" or the positive class. Type 1 Error measures the percentage of correct positive predictions made by the model among all positive predictions:

$$Type\ 1\ Error = \frac{tp}{tp + fp}, \quad (8)$$

where tp (true positive) is correct result, fp (false positive) is unexpected result.

4. *Type 2 Error*. This metric typically represents the proportion of false negative predictions made by a classification model, specifically in the context of binary classification tasks. Type 2 Error quantifies the rate at which the model incorrectly predicts negative outcomes, providing insights into its ability to avoid missing positive cases:

$$Type\ 2\ Error = 1\ -\ \frac{tp}{tp + fn}, \tag{9}$$

where tp (true positive) is correct result, fn (false negative) is missing result.

5. *F1-Score*. This metric is a combination of Precision and Recall and is used to assess the balance between these two metrics. It is particularly useful in situations where there is class imbalance (different numbers of instances for different classes) because it considers both Precision and Recall for each class and calculates their weighted harmonic mean:

$$F1\_Score = 2 * \frac{P(y, \hat{y}) \times R(y, \hat{y})}{P(y, \hat{y}) + R(y, \hat{y})}, \tag{10}$$

where $P(y, \hat{y})$ is precision, $R(y, \hat{y})$ is recall.

Leveraging synthetic dataset generators enables rapid iteration and testing of various hypotheses and model parameters without the need to wait for real data.

Table 6
Classification comparison

Dataset	Precision	Accuracy	Type 1 Error	Type 2 Error	F1-Score
original.csv	0.860633	0.860000	0.844660	0.876289	0.859972
std1.csv	1.000000	1.000000	1.000000	1.000000	1.000000
std3.csv	0.955073	0.955000	0.961905	0.947368	0.955012
40-60.csv	0.865830	0.865000	0.876712	0.858268	0.863560
30-70.csv	0.864803	0.865000	0.862745	0.865772	0.860449
blank10.csv	0.856354	0.856354	0.858696	0.853933	0.856354
blank20.csv	0.885321	0.885350	0.884058	0.886364	0.885190
outliers5.csv	0.865167	0.865000	0.870968	0.859813	0.864888
outliers10.csv	0.860000	0.860000	0.871560	0.846154	0.860000
outliers20.csv	0.868215	0.865000	0.831776	0.903226	0.864848

5. Conclusion

The article explores the construction of Gaussian distributions using the Box-Muller transform, a method relying on uniform random numbers to generate pairs of independent Gaussian variables. However, while efficient for core Gaussian values, the Box-Muller algorithm may fall short in generating extreme or outlier values crucial for evaluating rare events. To address this limitation, the article proposes incorporating specialized distributions to generate extreme values in tails. By combining these extreme values with standard normal random variables, a broader dataset can be formed, enriching evaluations and bolstering machine learning models.

This study also introduces a synthetic data generator designed for evaluating data visualization methods and machine learning systems. The application is highly adaptable, providing users with the capability to create and store models, generate artificial data, and even explore models created by other users. This enhanced accessibility promotes collaborative learning and testing of machine learning models on data. This article also demonstrates the practical utility of the application in the realm of assessing machine learning algorithms. It illustrates the process of generating various datasets, enabling precise control over typical challenges encountered in machine learning tasks. The visual representations created for each scenario provide compelling evidence of the tool's reliability in validating diverse situations.

In future research endeavors, the exploration of employing machine learning techniques to enhance the realism of synthetic data by introducing common noise patterns observed in real-world data, while

preserving the fundamental distribution, can be considered. Additionally, another avenue of investigation involves the generation of synthetic data encompassing categorical, time-series, or mixed data types. This would enable the utilization of the generated synthetic data within the context of the Computing with Words Model [25, 26] and other fuzzy set models.

In conclusion, the creation of benchmark datasets for testing data analytics tools represents a crucial step in the advancement of data science and machine learning research, offering a standardized means of evaluating and comparing the performance of various analytical methodologies. These benchmark datasets not only facilitate fair and rigorous assessment of data analytics tools but also open avenues for future research in the refinement of synthetic data generation techniques and the development of more comprehensive and realistic benchmark datasets.

6. References

[1] J. Thiyagalingam, M. Shankar, G. Fox, and T. Hey, "Scientific machine learning benchmarks," *Nature Reviews Physics*, vol. 4, no. 6, pp. 413–420, Apr. 2022, doi: https://doi.org/10.1038/s42254-022-00441-7.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2012, doi: https://doi.org/10.1145/3065386.

[3] Fernández-DelgadoManuel, CernadasEva, BarroSenén, and AmorimDinani, "Do we need hundreds of classifiers to solve real world classification problems," Journal of Machine Learning Research, Jan. 2014, doi: https://doi.org/10.5555/2627435.2697065.

[4] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha, "Secure and Robust Machine Learning for Healthcare: A Survey," IEEE Reviews in Biomedical Engineering, vol. 14, pp. 156–180, 2021, doi: https://doi.org/10.1109/rbme.2020.3013489.

[5] P. Mendonca, S. Brito, C. Gustavo, R. Santos, and T. Araujo, "Synthetic Datasets Generator for Testing Information Visualization and Machine Learning Techniques and Tools," IEEE Access, vol. 8, pp. 82917–82928, Jan. 2020, doi: https://doi.org/10.1109/access.2020.2991949.

[6] D. R. Jeske et al., "Generation of synthetic data sets for evaluating the accuracy of knowledge discovery systems," Knowledge Discovery and Data Mining, Aug. 2005, doi: https://doi.org/10.1145/1081870.1081969.

[7] P. J. Lin et al., "Development of a Synthetic Data Set Generator for Building and Testing Information Discovery Systems," IEEE Xplore, Apr. 01, 2006. https://ieeexplore.ieee.org/abstract/document/1611688.

[8] Popic, S., Pavkovic, B., Velikic, I., & Teslic, N. (2019). Data generators: a short survey of techniques and use cases with focus on testing. 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin). https://doi.org/10.1109/ICCE-BERLIN47944.2019.8966202 .

[9] F. K. Dankar and M. Ibrahim, "Fake It Till You Make It: Guidelines for Effective Synthetic Data Generation," Applied Sciences, vol. 11, no. 5, p. 2158, Feb. 2021, doi: https://doi.org/10.3390/app11052158.

[10] F. K. Dankar, M. K. Ibrahim, and L. Ismail, "A Multi-Dimensional Evaluation of Synthetic Data Generators," IEEE Access, vol. 10, pp. 11147–11158, 2022, doi: https://doi.org/10.1109/access.2022.3144765.

[11] Madalina Andreica, Mugurel Ionut Andreica, Nicolae Cataniciu. Multidimensional Data Structures and Techniques for Efficient Decision Making. Proceedings of the 10th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE) (ISBN: 978-960-474-063-5 / ISSN: 1790-5109), Mar 2009, Prague, Czech Republic. pp.249-254. ⟨hal-00467676⟩

[12] V. Ayala-Rivera, P. McDonagh, T. Cerqueus, and L. Murphy, "Synthetic Data Generation using Benerator Tool," arXiv (Cornell University), Oct. 2013. URL: https://www.researchgate.net/publication/258125711_Synthetic_Data_Generation_using_Benerator_Tool

[13] T. W. Göbel, T. Schäfer, Julien Hachenberger, J. Türr, and H. Baier, “A Novel Approach for Generating Synthetic Datasets for Digital Forensics,” pp. 73–93, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-56223-6_5.
[14] Ul Haq, Ikram & Gondal, Iqbal & Vamplew, Peter & Layton, Robert. (2016). Generating Synthetic Datasets for Experimental Validation of Fraud Detection. Fourteenth Australasian Data Mining Conference, Canberra, Australia. Conferences in Research and Practice in Information Technology, Vol. 170. URL: https://www.researchgate.net/publication/316878436_Generating_Synthetic_Datasets_for_Experimental_Validation_of_Fraud_Detection.
[15] T. N. Arvanitis, S. White, S. Harrison, R. Chaplin, and G. Despotou, “A method for machine learning generation of realistic synthetic datasets for validating healthcare applications,” Health Informatics Journal, vol. 28, no. 2, p. 146045822210770, Jan. 2022, doi: https://doi.org/10.1177/14604582221077000.
[16] Bullward, A., Aljebreen, A., Coles, A., McInerney, C., Johnson, O. (2023). Research Paper: Process Mining and Synthetic Health Data: Reflections and Lessons Learnt. In: Montali, M., Senderovich, A., Weidlich, M. (eds) Process Mining Workshops. ICPM 2022, vol 468. Springer, Cham. https://doi.org/10.1007/978-3-031-27815-0_25.
[17] Gräßler, I., Hieb, M., Roesmann, D., Unverzagt, M. (2023). Creating Synthetic Training Data for Machine Vision Quality Gates. In: Lohweg, V. (eds) Bildverarbeitung in der Automation. Technologien für die intelligente Automation, vol 17. Springer Vieweg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-66769-9_7.
[18] A. Y. Barrera-Animas and J. M. Davila Delgado, “Generating real-world-like labelled synthetic datasets for construction site applications,” Automation in Construction, vol. 151, p. 104850, Jul. 2023, doi: https://doi.org/10.1016/j.autcon.2023.104850.
[19] C. Manettas, N. Nikolakis, and K. Alexopoulos, “Synthetic datasets for Deep Learning in computer-vision assisted tasks in manufacturing,” Procedia CIRP, vol. 103, pp. 237–242, 2021, doi: https://doi.org/10.1016/j.procir.2021.10.038.
[20] Holst, D., Schoepflin, D., Schüppstuhl, T. (2023). Generation of Synthetic AI Training Data for Robotic Grasp-Candidate Identification and Evaluation in Intralogistics Bin-Picking Scenarios. In: Kim, KY., Monplaisir, L., Rickli, J. (eds) Flexible Automation and Intelligent Manufacturing: The Human-Data-Technology Nexus . FAIM 2022. Lecture Notes in Mechanical Engineering. Springer, Cham. https://doi.org/10.1007/978-3-031-18326-3_28.
[21] Rather, I.H., Kumar, S. Generative adversarial network based synthetic data training model for lightweight convolutional neural networks. *Multimed Tools Appl* (2023). https://doi.org/10.1007/s11042-023-15747-6 .
[22] G.E.P. Box and M.E. Muller, “A Note on the Generation of Random Normal Deviates,” The Annals of Mathematical Statistics, vol. 29, no. 2, pp. 610–611, Jun. 1958, doi: https://doi.org/10.1214/aoms/1177706645.
[23] D.B. Thomas, W. Luk, P.H.W. Leong, and J.D. Villasenor, “Gaussian random number generators,” ACM Computing Surveys, vol. 39, no. 4, p. 11, Nov. 2007, doi: https://doi.org/10.1145/1287620.1287622.
[24] A. Albashir, Mohd, K. Ibrahim, and Noratiqah Mohd Ariff, “Extreme Value Distributions: An Overview of Estimation and Simulation,” Journal of Probability and Statistics, vol. 2022, pp. 1–17, Oct. 2022, doi: https://doi.org/10.1155/2022/5449751.
[25] O. Tymchuk, A. Pylypenko, and M. Iepik, ‘Forecasting of Categorical Time Series Using Computing with Words Model’, in Selected Papers of the IX International Scientific Conference ‘Information Technology and Implementation’ (IT&I-2022), Workshop Proceedings, Kyiv, Ukraine, November 30 - December 02, 2022, vol. 3384, pp. 151–159. URL: https://ceur-ws.org/Vol-3384/Short_2.pdf.
[26] N. Kiktev, V. Osypenko, N. Shkurpela and A. Balaniuk, "Input Data Clustering for the Efficient Operation of Renewable Energy Sources in a Distributed Information System," *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, Zbarazh, Ukraine, 2020, pp. 9-12, doi: 10.1109/CSIT49958.2020.9321940