ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940www.ijsem.org

Vol 9, Issuse.2 June 2021

Tools for Information Visualization and Machine Learning Testing with Generated Datasets

DR. SUJIT KUMAR PANDA

.

ABSTRACT

Applications known as "data generators" create artificial datasets that can be used to test data analytics tools including machine learning algorithms and information visualization approaches. The method used to generate data varies depending on the data generator application. As a result, each one has functional shortcomings that prevent it from being used for certain purposes (e.g., lack of ways to create outliers and non-random noise). The data generator application described in this work intends to fill in any pertinent gaps left by previous programs and offers a flexible tool to let researchers rigorously test their methods in various contexts. The suggested approach enables users to define and combine well-known statistical distributions to get the required result, visualizing the data's behavior in real-time to determine whether it possesses the necessary properties.

INDEX TERMS: Data production system, creation of synthetic datasets, and creation of benchmark datasets.

INTRODUCTION

Using actual data is the best case scenario for evaluating machine learning algorithms and data visualization methods.

Nonetheless, getting the data might be a significant challenge, given that acquiring this information can take a while, cost money, or compromise your privacy. This situation forces researchers to repeatedly employ the same, well-established dataset. Researchers are either manually constructing synthetic datasets or developing tools to help this work in an attempt to overcome these difficulties.

Researchers utilize these programs to gain more command over the features of their data, allowing them to tailor datasets to tackle targeted issues such outlier identification, missing values, and noisy data. [1], [2].

Often referred to simply as "data generators," synthetic data applications are computer programs that alter the identifying characteristics of data using mathematical algorithms.

PROFESSOR, Mtech,Ph.D
Department of CSE
Gandhi Institute for Technology,Bhubaneswar

It was Shahzad Mumtaz, an associate editor at the journal, who oversaw the review process and gave final approval before the submission was published.

Several types of generators, including those for probability distributions, sets of categories, and so forth. Some generators let users can save a description of generators, which are often lightweight les, rather than specific data pieces, making it easy to share a blueprint of the created dataset with others.

There are several advantages to using a synthetic dataset generator, one of which is the ability to manipulate data features including patterns, trends, As more and more solutions are developed, it gets increasingly difficult to choose the most appropriate one. That's so tough due of the constraints that

The easiest way to judge the efficacy of a machine learning algorithm or a data visualization technique is to use real-world examples.

However, obtaining the information might prove to be quite difficult, considering the time, effort, and maybe even privacy risks involved in gathering such data. As a result, scientists have little choice but to keep using the same reliable data set. In an effort to get around these problems, researchers are either manually producing synthetic datasets or using technologies to aid in this job.

In order to address specific problems, such locating outliers, filling in missing values, and cleaning up noisy data, researchers use these tools to acquire more control over the characteristics of their data. [1], [2].

Synthetic data applications are computer programs that change the identifying properties of data using mathematical procedures, and are sometimes referred to as "data generators."

Journal assistant editor Shahzad Mumtaz handled the review process and provided final permission before publication.

Several sorts of generators, such as those for generating sets of categories, random numbers, and other distributions, and so on. Rather of saving individual data bits, some generators let users can record a description of generators, which are data type, format, outliers, dimensions, and missing values.

[3]. In order to thoroughly evaluate visualization strategies or machine learning algorithms in a controlled environment, data generators may manipulate data elements such that their properties address a specific issue [4, 5]. In certain generators, for instance, researchers may examine not only whether the method can withstand the presence of outliers, but also at what threshold of outlier percentage it can function well.

However, there is currently no software that can generate data perfectly.

typically lightweight les, making it simple to share a blueprint of the produced dataset with others.

The flexibility to change things like patterns, trends, data type, format, outliers, dimensions, and missing values is only one of the many benefits of employing a synthetic dataset generator.

[3]. Data generators may alter data items such that their attributes address a particular problem [4, 5] in order to conduct in-depth evaluations of visualization tactics or machine learning algorithms in a controlled setting. For example, in the case of particular generators, scientists may look at not only how effectively the approach handles outliers, but also at what percentage of outliers it continues to perform well.

Unfortunately, there is no ideal data-generating program available right now.

As more options become available, it becomes more challenging to choose on the best one. That's difficult because of severe limitations.

II. RELATED WORKS

Many fields in computing, including data visualization, data mining, software engineering, and artificial intelligence, recognize the value of generating synthetic datasets for testing purposes. Sran Popi et al. [7] conducted a literature review on the topic of synthetic data creation with an emphasis on application testing; they focused on the system designs and the intended use of the apps, and they outlined the benefits and drawbacks of the various methodologies they looked at. A failure-based application to produce synthetic data

units for testing software modules has been presented by Demillo and Offut [8]. There are genetic algorithm works in evolutionary computing [9], [10] that create data for software testing.

Multidimensional data may be generated using the methodology proposed by Albuquerque et al. [11]. By adjusting statistical distributions through a graphical user interface, the user may create a model that accurately represents the data of interest. However, [11] only deals with integer and oating numbers, therefore it doesn't cover the production of categorical data. And, [11] didn't

During the statistical distribution setup, if there is a method to get a preview of the data that shows how it could behave, please provide that information.

Wang et al. [12] have shown a program in which the user may hand-draw the data distribution. Because of this, the system builds the generators' data model from the user's sketches. A similar method was taken by Kwon et al. [13], who relied on design-based interactions to direct the development of a multi-dimensional data visualization depending on the expertise of the viewer.

To evaluate learning rules classication, Liu [14] developed a synthetic data generator. Using a method called decision tree algorithms, the work produces learning rules based on the characteristics given by the user in order to construct associations between these qualities. Similarly minded publications [15, 16], [17] have also surfaced in the literature suggesting data synthesizers for use in data mining applications. Since getting actual data might be prohibitively expensive or restricted by privacy concerns, these initiatives produce data for testing in data mining technologies. But there are occupations that create data for specific challenges, as [18] who published a paper on the data production system for healthcare applications, and these jobs often restrict the creation of new data to those specific domains.

The technique developed by Garca and Millán [19] to produce synthetic data is applicable to many different fields of study. The authors have illustrated both the benefits and drawbacks of their generator system and compared it to other applications now available. There is a no cost version of the program available.

Synthetic network data is sometimes generated by apps [20]. Examples include a proposal by Brodkorb et al. [21] to generate synthetic network data in which nodes are associated with geo-location. Thus, the user may experiment with the produced network by interacting with the displayed map, and then fine-tune the results.

To simulate water use in two cities, Konas et al. [22] developed a technique for creating synthetic data.

The developed technique logs the irregularity of a household's daily water use and verifies the accuracy of the resulting data using validation algorithms that compare a number of evaluation criteria on both actual and simulated data.

For the purpose of training and testing neural networks, Sun et al. [23] explain the use of a Gaussian matrix to simulate correlations between the weights of a network. They also created and used synthetic data alongside genuine data. Similar work has been done by Kang et al. [24], who used synthetic data to build tasks and put them through their paces as test cases for multi-tasking learning. In their presentation of a trained artificial neural network, Ma et al. [25] noted that the synthetic data allowed for a better analysis of the model's durability with respect to its initialization and the unpredictability of the data. Synthetic data production in recent works has been accomplished, however this data can't be used in another application since it was generated for a different issue.

Few works exist that can produce synthetic data without the requirement for programming, hence enabling the modification of data features and the creation of complicated patterns in data abilities, with the goal of evaluating instruments or machine learning algorithms, or viewing the data with a broad perspective.

III. PROPOSED APPLICATION

The primary objective of this paper is to provide a data generating app that can be used to help scientists evaluate data visualization and machine learning methods. The Software that lets you play around with statistical generators to get the simulated data you need. Therefore, researchers may easily replicate studies by exchanging synthetic datasets and their associated descriptive

metadata with one others. As seen in Figure 1, the tool's use ow chart.

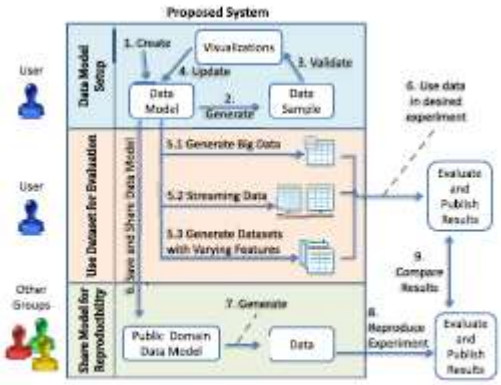


FIGURE 1. A typical use flow of the application.

The normal flow of the software includes three stages: installation, actual usage, and sharing. When a user generates a data model, they begin the iterative process of creating a synthetic dataset model (1). Specifying and composing generators to construct a description of data behavior (such as a correlation between specific dimensions or the existence of outliers) is the process of developing a data model. After making adjustments to the generators, the program generates a short sample dataset (2) to provide visual feedback on how the data is behaving graphically (3). (4).

It should be noted that the data created in the preview are not the same as those in the nal le; the preview's only function is to provide a short look at the behavior of the generators that the model will use to construct the nal dataset.

After the initial configuration is complete, users may either produce the data needed to test their apps or they can choose to publish the data model they created so that others can repeat their experiments. In addition to generating a dataset le that conforms to the model's generators, users can also generate massive amounts of data if necessary (5.1), feed the tested method or algorithm into a streaming of data generation (5.2), and generate a series of datasets that are nearly identical except for a few minor differences in their features (5.3). (5.3).

A user in this kind of data flow can decide to let other researchers use the model they created to recreate the first experiment (6).

Researchers that get this data model will have a greater chance of their own own data collection with distributions matching the ones used to generate them (7). The data points are similar because they exhibit the same properties and behavior (e.g., same correlations, probabilities, outliers), even when two datasets derived from the same model are not identical due to the inherent unpredictability of the creation process. This facilitates rapid comparison of findings by allowing researchers to quickly replicate experiments (8), even when using enormous synthetic datasets (9).

SYSTEM ARCHITECTURE OVERVIEW

In Figure 2 we see a high-level perspective of the app's framework:

The blue boxes show what kind of data was created, while the gray boxes reflect the key components. The you may get a brief model explanation, a le with control data points, or a visual representation of the data as output.

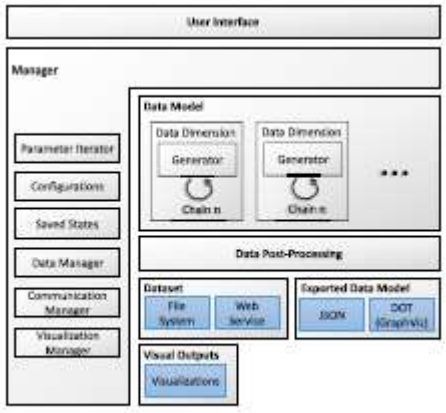


FIGURE 2. An overview of the application architecture.

Requests made through the GUI are sent to the management module, which is then responsible for processing them and passing them along to the appropriate submodules. This module cuts over many other areas.

Controls the flow of data across an application by acting as a go-between between the front end and the back end (the interface and the logic behind producing output, respectively). Each module is comprised of the following parts:

This is an Iterator for a Set of Parameters:

Configurations; Saved States; DMC; VMC; Data Manager; Data Manager; Communication Manager; Visualization Manager;

2) MODEL OF DATA

The data model is responsible for managing the dataset's data dimensions and the value generators. This thing we call a "data model" is just a representation of some kind, made up of details that define how data acts. In addition to the whole, final dataset, the data models may provide data samples, which are smaller datasets (by default). size (in this case, 100 rows) that all act in the same predefined way. Data samples provide for visual feedback since it is easy to see whether their properties meet the testing criteria. Using the same data model to generate many datasets (or data samples) yields data that is comparable (i.e., has the same behavior) but not identical (i.e., has different data values).

Sharing the data model that has been exported with other researchers makes it simpler to replicate an experiment, since the model is often smaller and more manageable than a big dataset.

3) MEASUREMENTS

Each dimension in the data model has its own set of generators. Data creation rules are stored in the dimensions. Each of the four dimensions includes the following fields: order number; title; data type; and generator chain. Dimensional data might be numerical, categorical, temporal, or mixed depending on the creation rules connected with it.

GENERATORS,

Values are created and changed by the generators. The Decorator pattern [26] enables a chain of generators to be constructed in a sequential fashion. Each parent and offspring generator is referenced by each other generator, allowing for bidirectional communication up and down the chain. The data produced by each generator mimics a cascading system in which the output of the parent generator affects the output of the child generator.

The schematic of the generator chain, including its inputs and outputs, is shown in Figure 3. During setup, the user specifies parameters and an operator () for each generator in the chain. The arguments that generators need to create values (such as the mean and standard deviation in Gaussian generators) are the parameters. The operator may be addition, subtraction, multiplication, division, or modulo, and it combines the results from two generators.

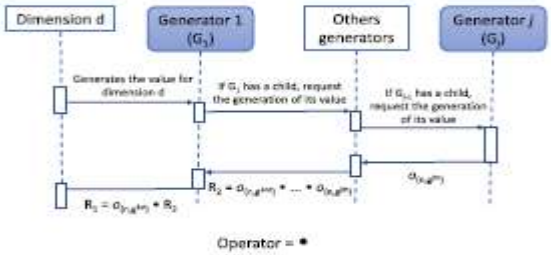


FIGURE 3. A general scheme of the generator chain for one dimension *d*.

Consider that a dataset DS is a set of n entries $DS = \{E_1, E_2, \ldots, E_n\}$, where each entry $E_{i|1 \leq i \leq n}$ is a set of m values $E_i = \{v_{(i,1)}, v_{(i,2)}, \ldots, v_{(i,m)}\}$, and each value $v_{(i,j)|1 \leq j \leq m}$ is related to a data dimension. Besides, each data dimension has a chain of generators $G_j = \{g_{(j,1)}, g_{(j,2)}, \ldots, g_{(j,\omega)}\}$, which is responsible to generate the values $\{v_{(1,j)}, v_{(2,j)}, \ldots, v_{(n,j)}\}$.

In order to create a value $v_{(i,j)}$, the generators recursively operate their results as follows:

$$r_k = g_{(j,k)} \cdot r_{k-1},$$
$$r_1 = g_{(j,1)}$$

Given that r1 is the first step in a recursive process and r! D v(i;j) is the output when the recursion reaches the final generator g(j;!), the first step is r1.

36 unique features are available in the current version of the app data generators, each of which has its own personality when it comes to producing numerical output, which may shift based on the output of its offspring. Accordingly, each successive generator serves as a building block from which a user may create their own distribution of data. Random, geometric, auxiliary, functional, and sequence generators are the five most common kinds.

a: THE RNGs generate each new value independently and arbitrarily according to some probability density or rule. For instance, the Uniform generator generates values between a minimum min and maximum max values with the same probability to any value in the range, while the Gaussian generator generates values based on a predened mean and standard deviation. Using the user-defined methods, random generators may be combined to produce novel distributions (for instance, a uniform distribution might be added to a Gaussian distribution to produce a new distribution).

As seen in Table 1, the program provides a number of different random number generators. Both real R and categorical C constants are acceptable for the

params. The probability density functions underlying the value creation process are seen in the output column.

In b, THE GEOMETRIC GENERATORS generate numbers based on geometrical building blocks. The user specifies parameters of forms in spaceR2, and the generator outputs data points in accordance with the pattern.

The output is not a value on, but rather an ordered pair, since geometric forms are specified on the space R2, which means that a single data dimension cannot describe the values (on1; on2).

This additional data dimension is required to produce the 2-dimensional data.

When a Geometric generator is assigned to a dimension's generator chain, it only returns the first element on1 of the ordered pair. The second element on2 of the pair may be generated with the help of a special generator called Get Extra; its behavior is described in more depth in the section on Accessory generators.

In Table 2 we see a catalog of Geometric generators. Constants (a1, a2, and a3) of real R type are used to define the parameters.

TABLE 1. The list of Random generators.

Name	Params.	Output
Uniform	$(min, max) \in \mathbb{R}$	
Gaussian	$(\mu, \sigma) \in \mathbb{R}$	
Cauchy	$(x_0, \gamma) \in \mathbb{R}$	
Poisson	$\lambda \in \mathbb{R}$	
Bernoulli	$p \in \mathbb{R}$	
Categorical	$a_{1,\ldots,z} \in C$	
Weighted Categorical	$a_{1,\ldots,z} \in C$, $b_{1,\ldots,z} \in \mathbb{R}$	

TABLE 2. The list of Geometric generators.

Name	Params.	Output
Stroke Quadratic Bézier	$a_{1,2,3} \in \mathbb{R}^2$	
Fill Quadratic Bézier	$a_{1,2,3} \in \mathbb{R}^2$	
Stroke Cubic Bézier	$a_{1,2,3,4} \in \mathbb{R}^2$	
Fill Cubic Bézier	$a_{1,2,3,4} \in \mathbb{R}^2$	

in how the forms act, like where the controls are. Each generator's output column depicts an illustration of the data point distribution inside the specified shape.

How to utilize the geometric generators is shown in Figure 4.

In order to generate a Cubic Bezier Stroke, the first element on1 of the pair is given the dimension D1. The second piece, on2, is obtained using a Get Extra Accessory and linked to dimension D2. The user has the option of making individual chains for each dimension, such as limiting noise to D2.

The ACCESSORY GENERATORS are the ones in charge of tweaking the results of the main generators. The Missing Value Accessory, for instance, is the component that shuffles and randomly modifies the input data.

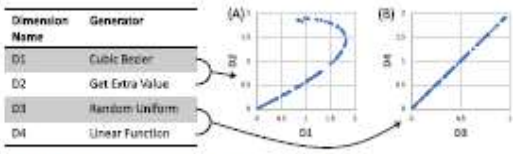


FIGURE 4. Using the Geometric generators and Get Extra Acessory.

missing values, with the user able to choose the proportion of values to be created by the child generator.

Deterministic results (MinMax, Linear, etc.) are possible at. Measurement Scale), or stochastic (e.g., Constant Noises, Missing Values).

Since generators may only generate a single value at a time, they can only return a single element from an n-tuple (a1; a2;::: ; an), in this case a1.

With the use of a Get Extra Accessory, you may extract a single item from the tuple that is returned, giving you access to all of its children.

Therefore, n1 additional dimensions with a Get Extra Accessory in each must be constructed so that all values of an n-tuple generator may be accessed.

The characteristics and outputs of each Accessory generator are listed in Table 3. Parameters (a1, a2, and a3) are constants of many possible types (real R, probability P, and natural N). The noise's probability distribution, denoted by r(), is an extra factor to consider when dealing with random noise (e.g., Gaussian or uniform).

To get a fresh value, the accessory will call the child ch(A) again if the value it received does not meet the limitations it was designed for (for example, if Range Filter gets a value outside the range).

TABLE 3. The list of Accessory generators.

Name	Params.	Output
Missing Value	$a \in \mathbb{P}$	$P(o_n = i_n) = 1 - a$ and $P(o_n = \varnothing) = a$
Range Filter	$a_{1,2} \in \mathbb{R}$	$o_n = \begin{cases} i_n & a_1 \geq i_n \geq a_2 \\ ch(A) & a_1 < i_n < a_2 \end{cases}$
Linear Scale	$a_{1,2,3,4} \in \mathbb{R}$	$o_n = \frac{i_n - a_1}{a_2 - a_1}(a_4 - a_3) + a_3$
MinMax	$a_{1,2} \in \mathbb{R}$	$o_n = \begin{cases} i_n & a_1 < i_n < a_2 \\ a_1 & i_n \leq a_1 \\ a_2 & i_n \geq a_2 \end{cases}$
Random Noise	$a_1 \in \mathbb{P}$, $a_2 \in \mathbb{R}$, $r(\alpha)$	$P(o_n = i_n) = 1 - a_1$ and $P(o_n = i_n + a_2 r(\alpha)) = a_1$
Constant Noise	$a_1 \in \mathbb{P}$, $a_2 \in \mathbb{R}$	$P(o_n = i_n) = 1 - a_1$ and $P(o_n = i_n + a_2) = a_1$
Low Pass Filter	$a \in \mathbb{R}$	$o_n = \frac{i_n - i_{n-1}}{a} + i_n$
No Repeat	-	$o_n = \begin{cases} i_n & i_n \notin \{i_{n-1}, ..., i_1\} \\ ch(A) & i_n \in \{i_{n-1}, ..., i_1\} \end{cases}$
Get Extra	$a \in \mathbb{N}$	$o_n = a^{th}(i_n)$

To create associated dimensions, d: THE FUNCTION GENERATORS convert the values produced in one dimension into another.

A Function generator requires the user to provide the

To make related dimensions, the d: THE FUNCTION GENERATORS transfer the values generated in one dimension onto another.

In order for a Function generator to work, the user must input

TABLE 4. The list of function generators.

Name	Params.	Output
Linear	$a_{1,2} \in \mathbb{R}$	$o_n = a_1 d_n + a_2$
Quadratic	$a_{1,2,3} \in \mathbb{R}$	$o_n = a_1 d_n^2 + a_2 d_n + a_3$
Polynomial	$a_{1,...,z} \in \mathbb{R}$	$o_n = \sum_{k=0}^{z-1} a_{k+1} d_n^k$
Exponential	$a_{1,2} \in \mathbb{R}$	$o_n = a_1^{d_n} a_2$
Logarithm	$a_{1,2} \in \mathbb{R}$	$o_n = \log_b(d_n)$
Sinusoidal	$a_{1,2,3} \in \mathbb{R}$	$o_n = a_1 \sin(a_2 d_n + a_3)$
Categorical	-	$o_n = \begin{cases} ch_1(A) & d_n = 1^{th}(C) \\ \vdots & \vdots \\ ch_z(A) & d_n = z^{th}(C) \end{cases}$
Piecewise Time	$a_{1,...,z} \in T$	$o_n = \begin{cases} ch_1(A) & d_n < a_1 \\ \vdots & \vdots \\ ch_z(A) & a_{z-1} \leq d_n < a_z \\ ch_{z+1}(A) & a_z \leq d_n \end{cases}$
Piecewise	$a_{1,...,z} \in \mathbb{R}$	$o_n = \begin{cases} ch_1(A) & d_n < a_1 \\ \vdots & \vdots \\ ch_z(A) & a_{z-1} \leq d_n < a_z \\ ch_{z+1}(A) & a_z \leq d_n \end{cases}$

There is a switch-case function between the Categorical, Piecewise Time, and Piecewise generators, with a unique set of generators for each of the three possible outcomes. Given that z is the total,

The Function generator ramies the chain into z offspring, denoted by ch1, ch2,..., chz, in the switch-case. The value dn of another dimension is utilized to determine which children are used to create the output on.

e: THE SEQUENCE GENERATORS produce numbers by following an algorithm that takes as inputs the parameters (a1; a2;::: ; az), the data index (n), and the preceding number (on1). The sequences might be arithmetic, geometric, or recursive, and they can have properties like being rising or decreasing, having convergent values, or being constrained to a finite set of values.

TABLE 5. The list of Sequence generators.

Name	Params.	Output
Constant	$a \in \mathbb{M}$	$o_n = a_1$
Counter	$a_{1,2} \in \mathbb{R}$	$o_n = o_{n-1} + a_2 \mid o_1 = a_1$
Time	$a_{1,2} \in \mathbb{T}$	$o_n = o_{n-1} + a_2 \mid o_1 = a_1$
Poisson Time	$a_{1,2} \in \mathbb{T}$, $\lambda \in \mathbb{R}$	$o_n = o_{n-1} + \frac{a_2}{poisson(\lambda)} \mid o_1 = a_1$
Sinusoidal	$a_{1,2,3,4,5} \in \mathbb{R}$	$o_n = a_1 \sin(a_2 c_n + a_3) \mid c_n = c_{n-1} + a_5$ and $c_1 = a_4$
Custom	-	$o_n =$ user defined
Categorical	$a_{1,...,z} \in C$, $b_{1,...,z-1} \in \mathbb{N}$	$o_n = a \begin{cases} 1 & n \leq \sum_{k=1}^{1} b_k \\ \vdots & \vdots \\ z-1 & n \leq \sum_{k=1}^{z-1} b_k \\ z & n > \sum_{k=1}^{z-1} b_k \end{cases}$

The application's current roster of Sequence generators is shown in Table 5. The parameters may be of any of many different types: mixed M, real R, temporal T, categorical C, or natural N.

The Poisson distribution parameter is also needed by the Poisson Time Sequence Generator.

When the first value in a series, o1, depends on the value of an earlier step in the sequence, that earlier step must have been created. The Sinusoidal generator's starting value, c1, is determined by past angles, cn, rather than by outputs.

Custom sequence logic is defined by the user through a textual rule that specifies the values of each on based on arithmetic operations (such as addition, subtraction, multiplication, and division), the preceding value x D on1, and the data index n. To generate a counter sequence where each value is equal to its index, the user need just provide "n" as the text rule.

5) OUTPUT DATA

When the data model is complete, the user may choose between many different export methods. data export, data model export, web service data streaming, and data export specification.

Users may initiate the generation process to store data points into the le system after the data model is prepared to build the nal dataset. Another option is for the system to create data in real time through a Web Service, where it would be generated and supplied in response to requests made via URLs.

The system may produce a JSON (JavaScript Object Notation) representation of the model if the user simply wishes to export the model and not the whole dataset. All of the model's generators, operators, and parameters are recorded in a compact hierarchical JSON le that can be imported into the system at a later time to restore the data model.

It is also possible to export the data model using a DOT file.

le, which can then be imported into GraphViz [27] to create a model diagram that is understandable to humans.

User interface (B)

The GUI's seven most notable features are shown in Figure 5 below. A menu bar, B tabs for currently open models, C a setup window, D a panel for configuring the generator and dimensions, D a preview of the data, and F a button to generate the model (G).

In Figure 5 (A), we see the application's menu bar, which includes the options File > Edit > Data Model > Visualize > Help.

New Model, New Dimension, Open Model, Save Model, Save Model As, and Import Dataset may all be found under the File menu. In the Edit menu, you may choose between Undo and Redo. Data samples from the current model may be seen in a number of different visualization formats, all of which are accessible through the Visualize menu: bar chart, histogram, scatterplot matrix, beeswarm plot, treemap, sunburst, parallel coordinates, and bundled parallel coordinates [28, 29]. You may rename, delete, or export your model from the Model menu. Toggle Web Service, Open Web Service, Toggle Web Service, and Copy Web Service URI are all examples of DOT Files.

Tabs of open models are shown in Figure 5 (B). Every tab has its own setup panel (C) with data model specifications such dimension titles and types, generator chains, and useful buttons like lter,

add generator, remove generator, and delete dimension. The C button, located in the panel's lower right corner, allows users to give their models an extra dimension. It's also feasible to rearrange and reposition generators. Dimensions may be ltered out by the user, causing them to be removed from both the preview and the final dataset. When a user clicks on a generator, details about that generator, as well as the dimension (E) it belongs to, are presented (D).

The operator and settings for the generator are set in the generator properties panel (D). The Data Preview panel (F) refreshes a parallel coordinates representation of the data samples after any modifications to the model, providing instantaneous visual feedback on data behavior.

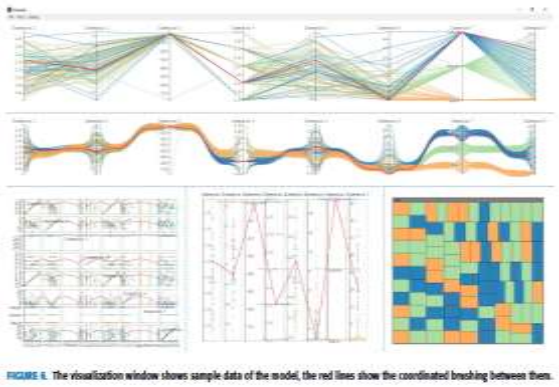
Selecting the blue "generate" (G) button in the lower-right corner of the window will launch a dialogue where you may specify the name, path, and length of your final dataset.

When you click the settings cog, a box opens up where you may adjust the Parameter Iterator's settings, which in turn produce a series of datasets with variable parameters.

Data samples from the model may be seen using the system's built-in visualization analysis tool (Menu > Visualize), in addition to the Preview Panel. This function is crucial because it allows the user to see in real time whether the data model is producing accurate results.

Users may pick and select the visualizations they want to work with in the visualization window, which can be a separate window from the primary one. As can be seen in Figure 6, the visualization window has a flexible layout, enabling the user to expand each visualization by moving the dotted line and even divide an area to add additional ones. Data objects from multiple views may be more easily related because to the consistent use of color, filtering, and selection across representations.

The user may also utilize several monitors to see many windows simultaneously.



IV. USAGE SCENARIOS - GENERATING DATASETS FOR

MACHINE LEARNING CLASSIFICATION

As an example of how to construct a test dataset for machine learning with variance in certain data attributes, consider the following situation.

Therefore, the proposed instrument will produce data sets.

Adjusting the number of classes, the number of outliers, the distance between classes, the percentage of missing data, the distribution of undesirable features, and the number of classes [31, 37].

Variations are specified on top of a standard dataset that contains the following features:

One thousand entries

0 exceptional cases

There are no blanks in this data.

One class and one important feature; no irrelevant characteristics are included.

80% Distinction between socioeconomic groups

Separate Tracks

Absence of Economic Disparity

Figure 7 depicts the procedure by which the system creates this predefined data collection. When mapping the class dimension (Dimension 1) to the feature dimension (Dimension 2), a Categorical Function acts as a decision point (Dimension 2). Parameters for the Uniform generator embedded in the Dimension 2 chains are class A1's Min D = 0

and Max D = 1:2, and class A2's Min D = 1 and Max D = 2. With these settings, there will be overlap of 20% in the feature dimension, resulting in only 80% class separation.

Therefore, six distinct datasets were produced, one for each of the six attributes in the default dataset. Systematically, the system produced four datasets for each dataset type, each with a tiny variation in the target attribute. The system generated datasets, for instance, to adjust the impact of the total number of outliers.

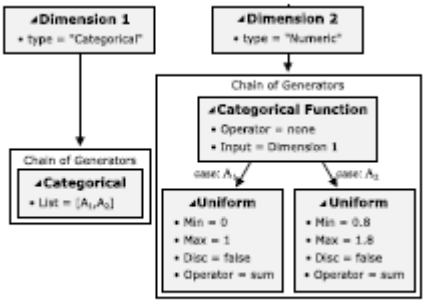


FIGURE 7. The generators that builds the default dataset.

with 10%, 20%, 30%, and 40% of outliers, without changing the other characteristics. The variations of the characteristics are the following:

- Amount of outliers: [10%, 20%, 30%, 40%]
- Class separation: [90%, 80%, 70%, 60%]
- Amount of missing values: [10%, 20%, 30%, 40%]
- Class imbalance: [50%-50%, 40%-60%, 30%-70%, 20%-80%]
- Bad features: [1-1, 1-3, 1-5, 1-7]
- Amount of classes: [2, 12, 22, 32]

AMOUNT OF OUTLIERS

The percentage of data points that are outliers is represented by the Amount of Outliers. Figure 8 demonstrates how to utilize the Noise Generator to generate the extreme values. There was some conjuring involved in making the noise generator work to modify the initial value by adding a uniformly distributed Gaussian noise with a mean of zero and a standard deviation of one, multiplied by 20 (the "force parameter"). The noise frequency ranged from ten percent to forty percent.

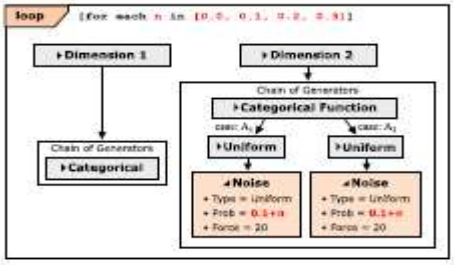


FIGURE 8. Adding noise generators to the uniform distributions creates outliers.

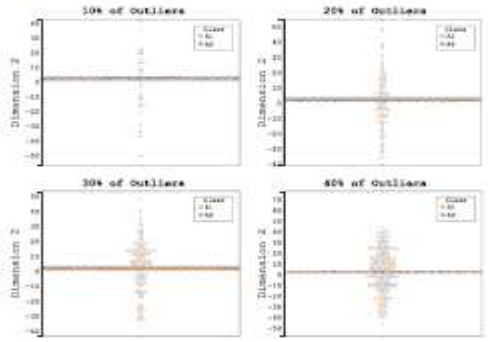


FIGURE 9. The accuracy of the models with varying amount of outliers.

In Figure 9, we see examples of these created datasets with varied numbers of outliers. Most of the information is between 0 and 1.8, with a few outliers at each end of the scale.

In Figure 8, found in the box labeled "Noise," we observe the relationship between the 'Prob' parameter and the number of outliers increasing from 10% to 40% graphically shown.

CLASS SEPARATION

The degree to which the distributions of two or more classes overlap is measured by the Class Separation feature. Parameters (Min and Max) of the Uniform Distribution Function are shown in Figure 10.

Altering the parameters of the generators allows for the introduction of a crossover in the distributions produced. For instance, in order to establish social stratification

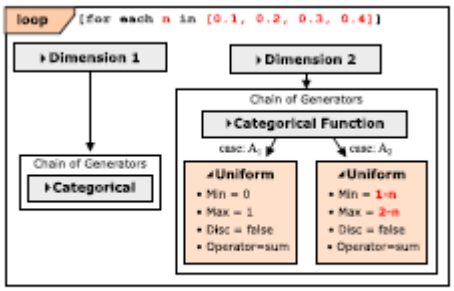


FIGURE 10. Sliding the interval of the uniform distribution of one class increases or decreases the class separation.

of 60%, 40% of the dimension range should be shared by the Uniform Generators (e.g., C1: $Min = 0$ and $Max = 1.4$; C2: $Min = 0.6$ and $Max = 2$; the interval [0.6, 1.4] is shared by both generators).

Distinction between the groups is seen in Figure 11.

The Histogram displays the total value for each category in the dataset, with a distinct break at the 96th percentile (0:96).

separation. From that point on, the intermixing of the various categories' distributions will rise. This data might be used to examine the hypothesis that as class overlap grows, so does classifier accuracy

AMOUNT OF MISSING VALUES

The percentage of blank cells in a dataset represents the number of missing values. This feature is generated via the MCAR (Missing Completely at Random) generator, as seen in Figure 12.

Using the probability specified by the parameter "Prob," this generator takes the output of another generator (here, a Uniform generator) and replaces it with a missing value.

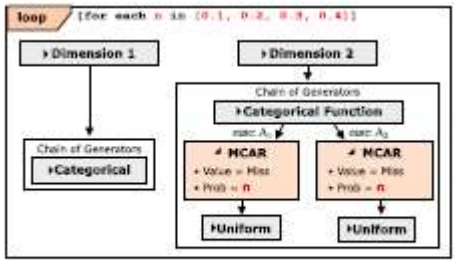


FIGURE 12. Adding an MCAR accessory before the uniform distributions create missing values randomly according to a probability.

Datasets with blank cells are shown in Figure 13. Dimension 2's red hue represents a 10% to 40% shift in values, and the red color itself maps the missing values.

grows as the class stays the same, which is what we want to see when we test out various classication methods with missing data. To keep the visuals from becoming too busy, we reduced the red's opacity to 0.2.

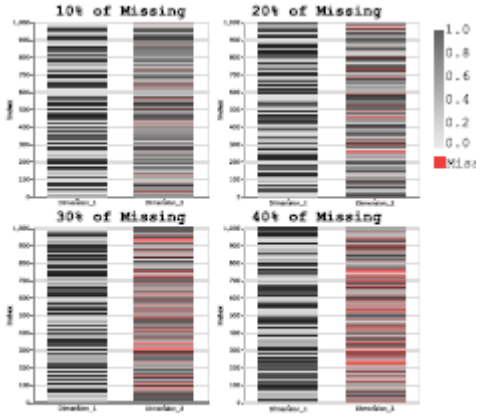


FIGURE 13. The amounts of missing value by its probability.

The created datasets might be used to assess a classifier's ability to cope with missing values in features. It's possible that other sources of missing values exist.

Methodological frameworks, such as the MAR, that are provided (Missing at Random).

The imputation techniques might be put to the test with this data as well.

CLASS IMBALANCE

Class Imbalance measures how evenly distributed the data is amongst the different categories. Because of the disparity in class, there are a number of hypotheses that might be explored using new methods in classication [38].

Using the percentage of occurrence in the dataset as its weight (or probability), the Weighted Categorical generator provides this feature (see Figure 14).

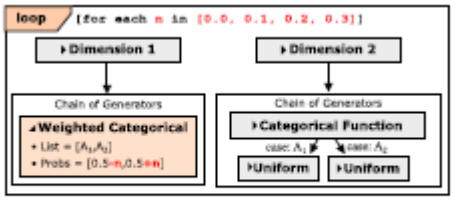


FIGURE 14. Changing the categorical generator to weighted categorical allows the generation of class-imbalanced datasets.

On beeswarm plots, as shown in Figure 15, there is an obvious gender and age disparity.

When everything is in equilibrium, the two distributions have the same thickness on the plot, but as the imbalance begins, they begin to diverge.

To evolve into something new. The thickness of A1 is substantially lowered while the thickness of A2 is increased in the end (20%-80%)

BAD FEATURES

The quantity of characteristics that have no connection to the class (i.e., are bad) is what is meant by the term "Bad Features." so that it might be included in the canon of classic literature. Adding dimensions using Uniform Generators is sufficient (see Figure 16) since the dimensions are not connected.

The flaws of a Parallel Coordinates are shown in Figure 17.

The excellent feature shows a striking visual difference, and

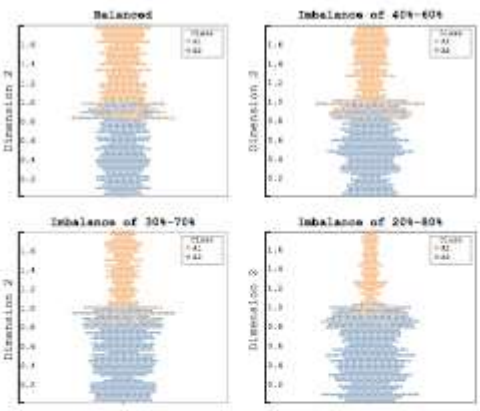


FIGURE 15. Imbalance of classes on beeswarm plots.

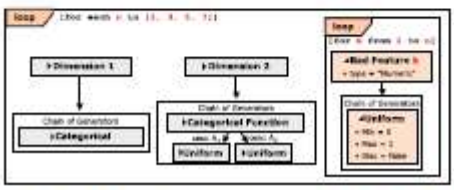


FIGURE 16. Adding dimensions without categorical functions creates bad

Inconsistent and uninteresting designs are used to illustrate the poor quality of the features. A classifier's ability to distinguish between useful and non-useful characteristics might be evaluated using such data sets.

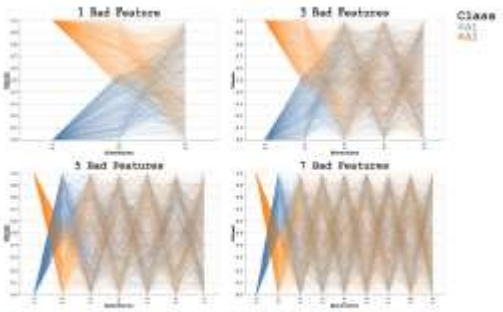


FIGURE 17. Parallel Coordinates showing bad features.

AMOUNT OF CLASSES

The quantity of classes is the total number of class dimensions. You can see how the Categorical Generator takes in a wide variety of categories in Figure 18.

Beeswarm plot class counts are shown binned in Figure 19. Each group is represented by a different hue, and their relative placement is also indicated. Class plots become quite tiny beyond 22 classes, yet data distribution shows clear distinction between groups. These data sets might be used to see whether classifiers perform well under high-volume classification.

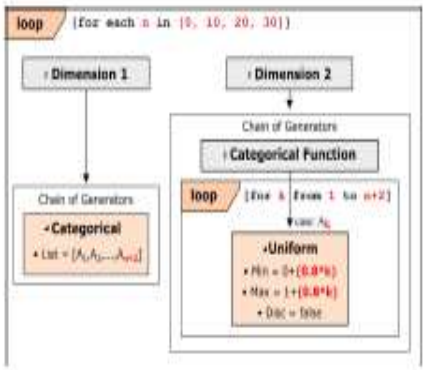


FIGURE 18. Adding categories to the Categorical Generator increases the number of classes.

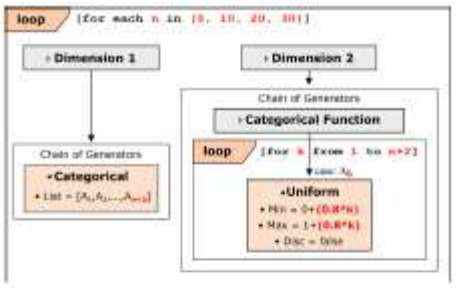


FIGURE 18. Adding categories to the Categorical Generator increases the number of classes.

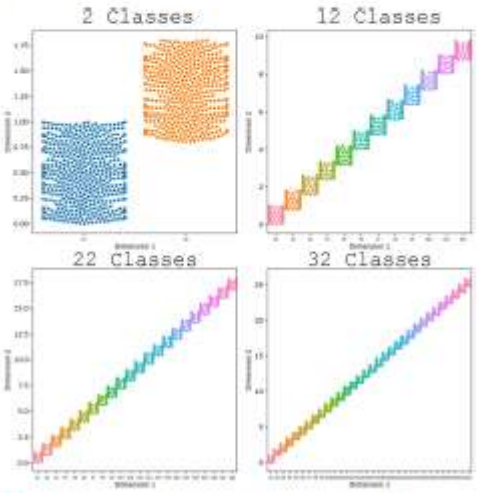


FIGURE 19. The accuracy of the models with varying amount of classes.

classes, also test if the accuracy of classi_ers remains balanced between classes.

V. CONCLUSION

In this study, a synthetic data generator was introduced to test data visualization tools and machine learning algorithms. The program is very adaptable and provides the flexibility to build unique generation proles from a variety of distribution primitives, including but not limited to uniform and normal distributions. The included tools, functions, sequences, and geometric generators make it possible to create extremely individualized data sets.

The ability to import and export the descriptive model le is a big boon to the repeatability of scientific studies.

To facilitate integration of the system and the data it generates with other applications, the software provides a web service for receiving and processing data in a continuous stream.

In addition, this article demonstrates how the program may be put to work in the context of testing various machine learning approaches.

This demonstrated the feasibility of generating novel datasets, giving the user command over recurrent issues in machine learning projects. The built-in graphics for each case study demonstrate the tool's reliability in verifying circumstance that arises.

The process of separating generators from actual data is a potential direction for future research. An idiom may be developed from the modeling language that defines the composition of a chain of generators, which can then be used to the study of the behavior of actual data. If this barrier could be overcome, it would be possible to construct synthetic data from genuine ones by altering the characteristics that drive their distributions. Machine learning methods may also be used to improve the realism of synthetic data generation by include noises that occur naturally in actual data without significantly altering the distribution underlying the data.

In addition, the authors plan to incorporate: a constant seed to allow for the generation of a unique dataset, new visualizations for data validation, and new interaction possibilities, such as zoom in/out, lter, and re-ordering; more types of generators; and new ways to display generators to facilitate the understanding of the model's design. In addition, future work may include a visual or quantitative comparison component for verification or comparison. In addition, the system will make use of real-world data to model similar synthetic data, which the user may then compare to the original.

REFERENCES

[1] B. S. Santos and P. Dias, ``Evaluation in visualization: Some issues and best practices," *Vis. Data Anal.*, vol. 9017, Feb. 2013, Art. no. 90170O. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding aspx?doi=10.1117/12.2038259

[2] B. S. Santos, ``Evaluating visualization techniques and tools: What are the main issues?" in *Proc.Workshop Beyond Time Errors Novel Eval. Methods Vis. (BELIV)*, 2008, pp. 1_2.

[3] R. Redpath and B. Srinivasan, ``Criteria for a comparative study of visualization techniques in data mining," in *Intelligent Systems Design and Applications*. Berlin, Germany: Springer, 2003, pp. 609_620.

[4] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, ``Empirical studies in information visualization: Seven scenarios," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 9, pp. 1520_1536, Sep. 2012.

[5] S. Liu, W. Cui, Y. Wu, and M. Liu, ``A survey on information visualization: Recent advances and challenges," *Vis. Comput.*, vol. 30, no. 12, pp. 1373_1393, Dec. 2014.

[6] Y. P. dos Santos Brito, C. G. R. dos Santos, S. de Paula Mendonca, T. D. Araujo, A. A. de Freitas, and B. S. Meiguins, ``A prototype application to generate synthetic datasets for information visualization evaluations," in *Proc. 22nd Int. Conf. Inf. Vis. (IV)*, Jul. 2018, pp. 153_158.

[7] S. Popi¢, B. Pavkovi¢, I. Veliki¢, and N. Tesli¢, ``Data generators: A short survey of techniques and use cases with focus on testing," in *Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Sep. 2019, pp. 189_194.

[8] R. A. DeMilli and A. J. Offutt, ``Constraint-based automatic test data generation," *IEEE Trans. Softw. Eng.*, vol. 17, no. 9, pp. 900_910, Sep. 1991.

[9] M. Mann, O. P. Sangwan, P. Tomar, and S. Singh, ``Automatic goaloriented test data generation using a genetic algorithm and simulated

annealing," in *Proc. 6th Int. Conf.-Cloud Syst. Big Data Eng. (Con_uence)*, Jan. 2016, pp. 83_87.

[10] S. Rani and B. Suri, ``An approach for test data generation based on genetic algorithm and delete mutation operators," in *Proc. 2nd Int. Conf. Adv. Comput. Commun. Eng. (ICACCE)*, May 2015, pp. 714_718.

[11] G. Albuquerque, T. Lowe, and M. Magnor, ``Synthetic generation of highdimensional datasets," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2317_2324, Dec. 2011.

[12] B.Wang, P. Ruchikachorn, and K. Mueller, ``SketchPadN-D: WYDIWYG sculpting and editing in high-dimensional space," *IEEE Trans. Vis. Com-put. Graphics*, vol. 19, no. 12, pp. 2060_2069, Dec. 2013.

[13] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert, ``AxiSketcher: Interactive nonlinear axis mapping of visualizations through user drawings," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 221_230, Jan. 2017.

[14] R. Liu, B. Fang, Y. Y. Tang, and P. P. K. Chan, ``Synthetic data generator

for classi_cation rules learning," in *Proc. 7th Int. Conf. Cloud Comput. Big Data (CCBD)*, Nov. 2016, pp. 357_361.

[15] P. J. Lin, B. Samadi, A. Cipolone, D. R. Jeske, S. Cox, C. Rendón, D. Holt, and R. Xiao, ``Development of a synthetic data set generator for building and testing information discovery systems," in *Proc. 3rd Int. Conf. Inf. Technol., New Gener. (ITNG)*, 2006, pp. 707_712.

[16] D. R. Jeske, P. J. Lin, C. Rendón, R. Xiao, and B. Samadi, ``Synthetic data generation capabilties for testing data mining tools," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2007, pp. 1_6.

[17] M. Pasinato, C. E. Mello, M.-A. Aufaure, and G. Zimbrão, ``Generating synthetic data for context-aware recommender systems," in *Proc. 1st BRICS Countries Congr. Comput. Intell. (BRICS-CCI)*, Sep. 2013, pp. 563_567.

[18] J. Dahmen and D. Cook, ``SynSys: A synthetic data generation system for healthcare applications," *Sensors*, vol. 19, no. 5, p. 1181, 2019.

[19] D. García and M. Millán, ``A prototype of synthetic data generator," in *Proc. 6th Colombian Comput. Congr. (CCC)*, May 2011, pp. 1_6.

[20] *Graph Generation With Prescribed Feature Constraints*, Soc. Ind. Appl. Math., Philadelphia, PA, USA, Apr. 2009.