VEGUETA

Vol 9 Issue 02,Aug 2021

ISSN NO: 1133-598X

Visualization Software for Data and Machine Learning Experiments with Mock Datasets

V Sudha[1,] Smd.Rahamathulla 2, N J Pramod Dinakar3, B Gouri4

[1,2,3,4] Asst. Professor,

Department of CSE, K.S.R.M College of Engineering(A), Kadapa

ABSTRACT:

Software referred to as "data generators" produce simulated data sets, which may be used to evaluate various data analytics tools, such as machine learning algorithms and various methods of data visualization. Different data generator programs utilize different strategies to produce data. Because of this, they can only be utilized for limited tasks due to various limitations (such an absence of tools to generate outliers and non-random noise, for example). This paper describes a data generator application with the goal of filling in any gaps left by earlier programs and providing a versatile tool to allow academics to thoroughly test their approaches in a variety of settings. The proposed method lets users specify and combine well-known statistical distributions to reach the desired outcome, and then see the data's behavior in real-time to check whether it satisfies the requirements.

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

INTRODUCTION

Machine learning algorithms and data visualization techniques are best evaluated when applied to real-world data.

However, this information may be difficult to get because of the time, money, or privacy risks associated with doing so. 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.

Researchers use these tools to take control of their data's characteristics, transforming their datasets into more refined studies of problems like outlier detection, missing values, and excessive noise. [1], [2].

Synthetic data applications, sometimes known as "data generators," are software programs that use mathematical techniques to change the distinguishing properties of data.

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

There are many different kinds of generators, such as those that produce random numbers, sets of categories, and so on. Instead of saving individual data bits, some generators let users can record a description of generators, which are often lightweight les, making it simple to share a blueprint of the produced dataset with others.

Synthetic dataset generators provide various benefits, one of which is the capacity to modify data characteristics such as patterns, trends, data type, format, outliers, dimensions, and missing values.

In order to rigorously test visualization tactics or machine learning algorithms in a simulated setting, data generators may alter data pieces such that their attributes address a particular problem [3, 4]. Researchers may test not only how well a given generator performs in the presence of outliers, but also at what proportion of outliers it performs optimally.

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.

Using real-world examples is the simplest approach to evaluate a machine learning algorithm or a data visualization tool.

However, it might be challenging to collect the information due to the time, effort, and sometimes even privacy hazards associated in acquiring such data. Since this data collection has shown to be accurate in the past, scientists have no option but to continue utilizing it. Researchers are either manually creating synthetic datasets or using technology to assist in this task in an attempt to circumvent these issues.

Researchers utilize these technologies to gain greater agency over the properties of their data in order to solve certain issues, such as finding outliers, filling in missing values, and cleaning up noisy data. [1], [2].

Synthetic data applications, sometimes known as "data generators," are computer programs that modify the identifying qualities of data by mathematical techniques.

Shahzad Mumtaz, an assistant editor of the journal, oversaw the review process and gave final approval.

Multiple types of generators, including ones that may produce random numbers, various distributions, sets of categories, and more. Some generators let users to record a description of generators; these descriptions are often lightweight strings that make it easy to share a blueprint of the generated dataset with others.

Using a synthetic dataset generator has various advantages, including the ability to modify data characteristics such as patterns, trends, data type, format, outliers, dimensions, and missing values.

It is possible for data generators to manipulate data items such that their properties address a specific issue [4, 5] in order to undertake controlled, in-depth assessments of visualization strategies or machine learning techniques. For instance, researchers may examine not just how well a technique handles outliers, but also for what fraction of outliers it continues to perform well, when considering a given generator.

Right present, there is no perfect software that can generate data.

The optimal choice might be increasingly difficult to make when more possibilities become accessible. Extreme constraints make that a challenge.

II. RELATED WORKS

The utility of creating synthetic datasets for testing reasons has been acknowledged by many areas of computing, including data visualization, data mining, software engineering, and artificial intelligence. Sran Popi et al. [7] reviewed the literature on synthetic data production, with a focus on application testing; they paid special attention to the system designs and the intended use of the applications, and they detailed the advantages and disadvantages of the different approaches they considered. Demillo and Offut [8] described a failure-based program to generate synthetic data units for testing software modules. Data for software testing may be generated by genetic algorithm works in evolutionary computing [9], [10].

Methods for producing multidimensional data have been suggested [11] by Albuquerque et al. The user may tailor the model to their needs by modifying statistical distributions through a graphical user interface. Categorical data creation is not addressed in [11] since it is limited to integer and oating numbers. Moreover, [11] didn't If there is a way to acquire a preview of the data that demonstrates how it may behave during the statistical distribution setup, please share that information.

Wang et al. [12] demonstrated a tool in which the user may manually design the distribution of the data. The system derives the data model for the generators from the user's drawings in this way. Kwon et al. [13] used a similar approach by using design-based interactions to guide the creation of a multi-dimensional data display that takes into account the knowledge of the viewer.

Liu [14] created a synthetic data generator to test learning rules classication. The work generates learning rules based on the attributes offered by the user using a technique called decision tree algorithms to establish connections between these features. Data synthesizers have been proposed in a number of articles [15, 16], [17] for use in data mining projects. Given the high cost of data collection and the need to protect individuals' privacy, these efforts generate data for use in data mining technology evaluations. The development of fresh data, however, is sometimes limited to certain fields by the nature of the activities that generate it, such as [18], who released a study on the data production system for healthcare applications.

The methodology for creating synthetic data proposed by Garca and Millán [19] may be used in a wide variety of research areas. The authors have shown the advantages and disadvantages of their generating system and compared it to existing software. The software also comes in a freemium edition.

Applications may sometimes produce synthetic network data [20]. Brodkorb et al. [21] propose a method for creating synthetic network data in which nodes include geo-location information. A map is presented, and the user may play about with the generated network by clicking and dragging.

Konas et al. [22] devised a method for synthesizing synthetic data to model water usage in two cities.

The created method keeps track of the unpredictable patterns in a family's daily water use and then validates the data's veracity using validation algorithms that evaluate several evaluative criteria using both real and simulated data.

In order to replicate correlations between a network's weights during training and testing, Sun et al. [23] describe how to employ a Gaussian matrix. They employed both real and fake data in their analysis. Kang et al. [24] utilized synthetic data to construct tasks and put them through their paces as test cases for multi-task learning, therefore the study is similar. In their discussion of the robustness of a trained artificial neural network in the face of initialization and unpredictable input, Ma et al. [25] emphasized the need of using synthetic data. Recent efforts have achieved synthetic data creation, however this data cannot be utilized in another application since it was developed for a distinct problem.

Few works exist that can generate synthetic data without the need for programming, allowing for the customization of data features and the generation of intricate patterns in data capacities, all with the end goal of evaluating instruments or machine learning algorithms, or gaining a holistic view of the data.

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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

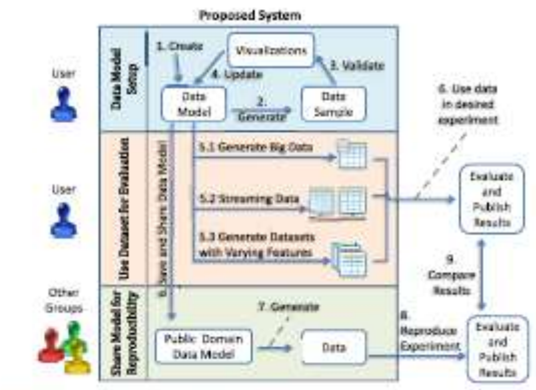


FIGURE 1. A typical use flow of the application.

Installation, real use, and sharing are the three steps of the software's usual workflow. A synthetic dataset model (1) is created via an iterative process that begins when a user develops a data model. The process of constructing a data model involves the specification and composition of generators to produce a description of data behavior (such a correlation between various dimensions or the occurrence of outliers). After tweaking the generators, the application creates a small sample dataset (2) to show the user the graphical behavior of the data (3). (4).

It is important to note that the preview does not produce the same data as the final result, but rather only gives a quick glimpse into the behavior of the generators that will be used by the model to produce the final dataset.

Once setup is complete, users may generate test data for their applications or share their data model with the community so that others can replicate their findings. Users can generate a dataset le that follows the model's generators (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 features (5.3), among other things. (5.3).

A participant in such a data flow may choose to share their model with other researchers so that they may do the same experiment (6).

To a larger extent, if researchers get this data model, their own data sets will likely have distributions that are consistent with the ones used to produce them (7). When two datasets created from the same model are not identical owing to the inherent unpredictability of the development process, the data points are comparable because they display the same attributes and behavior (e.g., same correlations, probabilities, outliers). Researchers can readily duplicate experiments (8), even with massive synthetic datasets (9), allowing for faster comparison of results.

SYSTEM ARCHITECTURE OVERVIEW

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

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

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.

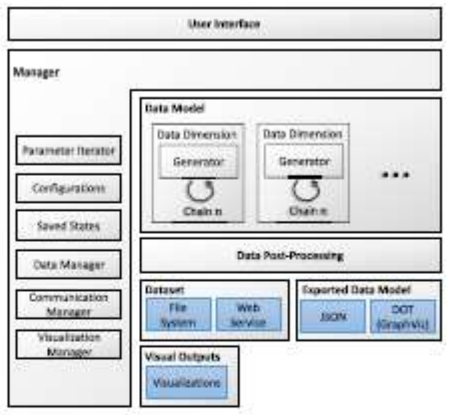


FIGURE 2. An overview of the application architecture.

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,

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

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.

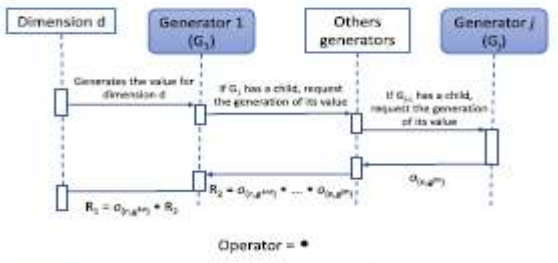

Dimension d
Generator 1 (G_1)
Others generators
Generator j (G_j)
Generates the value for dimension d
If G_1 has a child, request the generation of its value
If G_{j-1} has a child, request the generation of its value
$R_2 = o_{(r,g^{(m)})} * ... * o_{(n,g^{(m)})}$
$R_1 = o_{(r,g^{(m)})} * R_2$
Operator = *

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 $E_{i|1 \leqslant i \leqslant n}$ is a set of m values $E_i = \{v_{(i,1)}, v_{(i,2)}, \ldots, v_{(i,m)}\}$, and each value $v_{(i,j)|1 \leqslant j \leqslant m}$ 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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

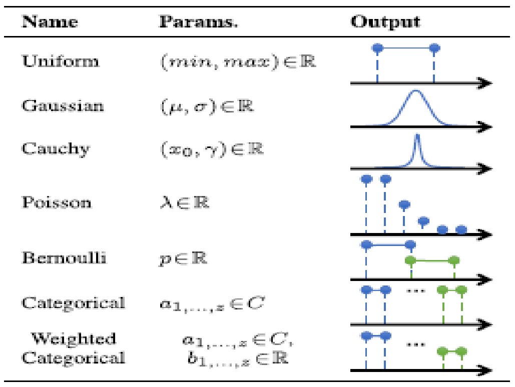
The output is not a value on, but rather an ordered pair, since geometric forms are specified on the space R2, which means that a single data dimension cannot describe the values (on1; on2).

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.

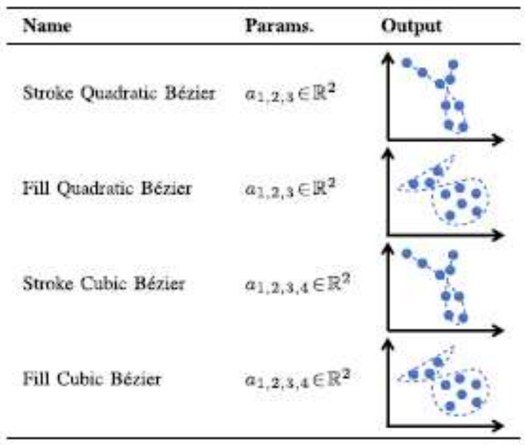
In Table 2 we see a catalog of Geometric generators. Constants (a1, a2, and a3) of real R type are used to define the parameters.

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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

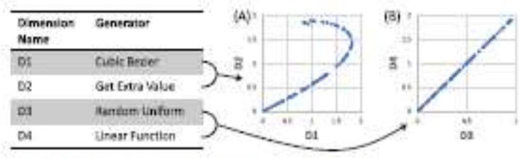


FIGURE 4. Using the Geometric generators and Get Extra Acessory.

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	$a_1 \in \mathbb{P}$, $a_2 \in \mathbb{R}$, $r(a)$	$P(o_n = i_n) = 1 - a_1$ and $P(o_n = i_n + a_2 r(a)) = a_1$
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	-	$o_n = \begin{cases} i_n & i_n \notin \{i_{n-1}, \ldots, i_1\} \\ ch(A) & i_n \in \{i_{n-1}, \ldots, i_1\} \end{cases}$
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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

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

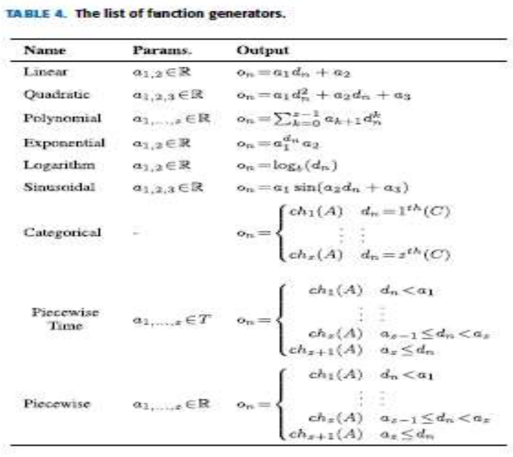


TABLE 4. The list of function generators.

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	$a_{1,\dots,z} \in \mathbb{R}$	$o_n = \sum_{k=0}^{z-1} a_{k+1} d_n^k$
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	$a_{1,\dots,z} \in T$	$o_n = \begin{cases} ch_1(A) & d_n < a_1 \\ \vdots & \vdots \\ ch_z(A) & a_{z-1} \le d_n < a_z \\ ch_{z+1}(A) & a_z \le d_n \end{cases}$
Piecewise	$a_{1,\dots,z} \in \mathbb{R}$	$o_n = \begin{cases} ch_1(A) & d_n < a_1 \\ \vdots & \vdots \\ ch_z(A) & a_{z-1} \le d_n < a_z \\ ch_{z+1}(A) & a_z \le d_n \end{cases}$

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.

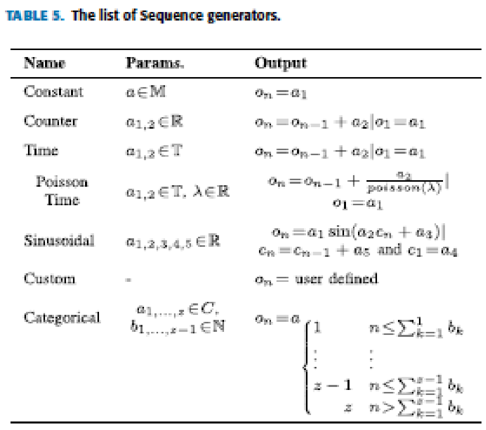


TABLE 5. The list of Sequence generators.

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

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.

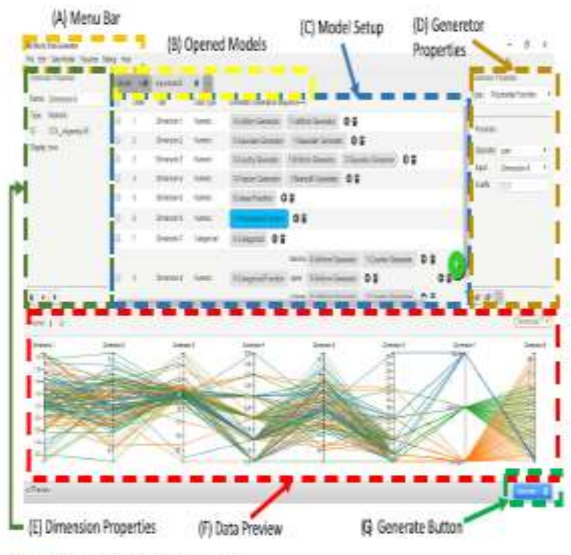


FIGURE 5 The main graphical interface of the application

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.

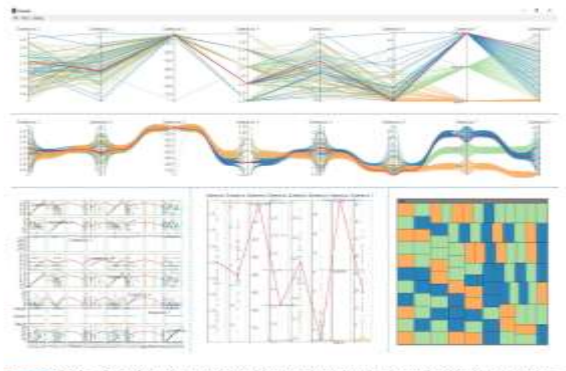


FIGURE 6. The visualization window shows sample data of the model, the red lines show the coordinated brushing between them.

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.

VEGUETA Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

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.

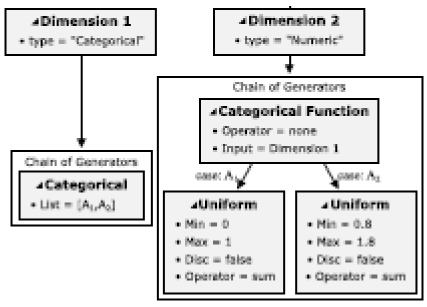
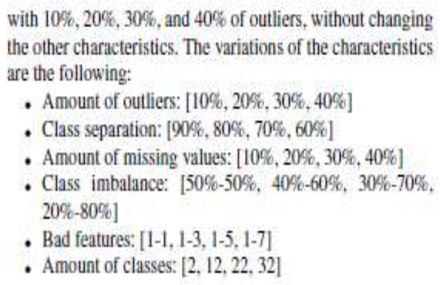


FIGURE 7. The generators that builds the default dataset.


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

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

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.

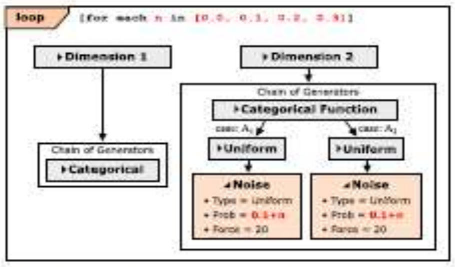


FIGURE 8. Adding noise generators to the uniform distributions creates outliers.

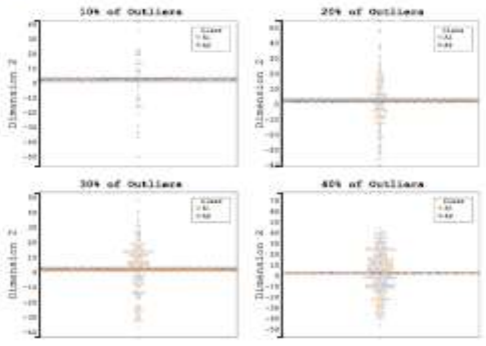


FIGURE 9. The accuracy of the models with varying amount of outliers.

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

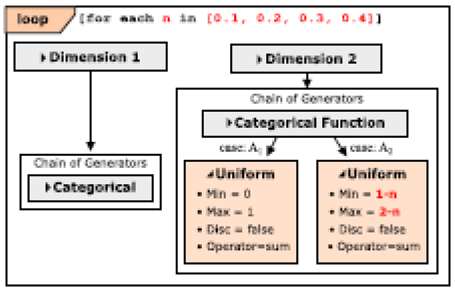


FIGURE 10. Sliding the interval of the uniform distribution of one class increases or decreases the class separation.

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.

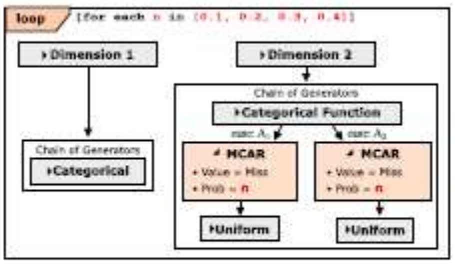


FIGURE 12. Adding an MCAR accessory before the uniform distributions create missing values randomly according to a probability.

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.

8th Annual Volume, 2020, 82925

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

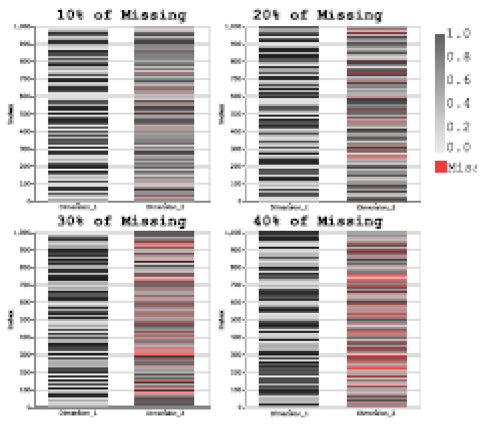


FIGURE 13. The amounts of missing value by its probability.

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).

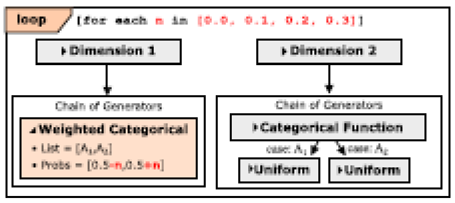


FIGURE 14. Changing the categorical generator to weighted categorical allows the generation of class-imbalanced datasets.

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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

The excellent feature shows a striking visual difference, and

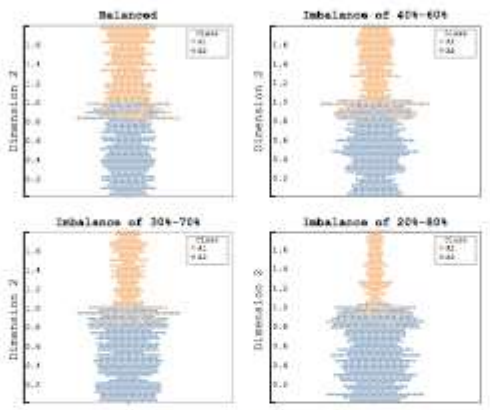


FIGURE 15. Imbalance of classes on beeswarm plots.

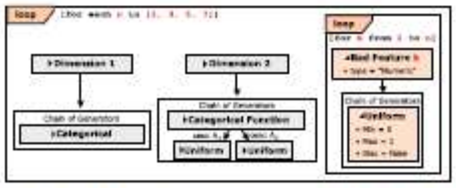


FIGURE 16. Adding dimensions without categorical functions creates bad

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.

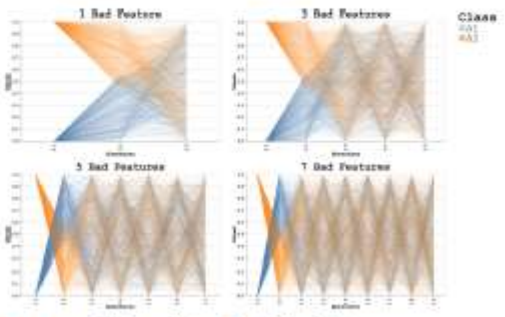


FIGURE 17. Parallel Coordinates showing bad features.

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.

Vol 9 Issue 02,Aug 2021
ISSN NO: 1133-598X

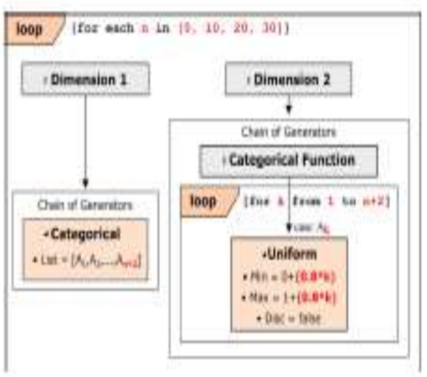


FIGURE 18. Adding categories to the Categorical Generator increases the number of classes.

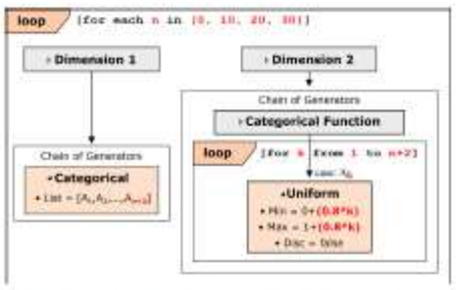


FIGURE 18. Adding categories to the Categorical Generator increases the number of classes.

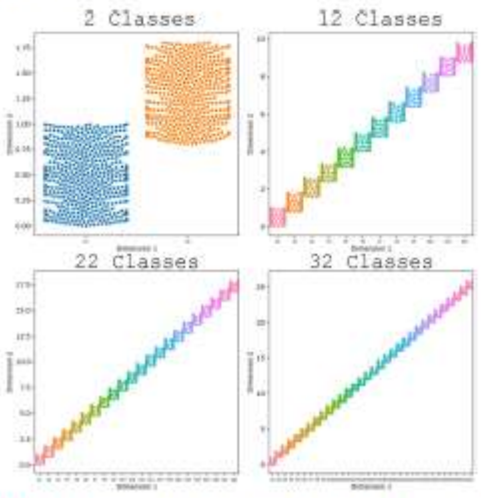


FIGURE 19. The accuracy of the models with varying amount of classes.

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

V. CONCLUSION

To evaluate data visualization tools and machine learning techniques, this research devised a synthetic data generator. The program's versatility allows users to construct their own generation proles from a wide range of distribution primitives, not only the standard uniform and normal distributions. With the help of the provided utilities, functions, sequences, and geometric generators, one may construct highly customized data sets.

The descriptive model le's import/export capabilities greatly improve the reliability of scientific research.

The program includes a web service for receiving and processing data in a continuous stream, allowing the system and the data it creates to be easily integrated with other applications.

This article also shows how the software may be put to use when putting different machine learning strategies to the test.

It proved that new datasets could be generated, providing the user more agency over recurring problems in ML initiatives. The built-in visuals for each case study show how effective the tool is in verifying unexpected situations.

Research might go in the direction of figuring out how to distinguish fake data from real data. The modeling language used to specify the construction of a series of generators may provide the basis for the development of an idiom that can be used to the analysis of real-world data's behavior. If this roadblock could be removed, it would be conceivable to create artificial data from real ones by manipulating the factors that determine their distributions. Incorporating noises that occur naturally in actual data without materially affecting the distribution underlying the data is another way in which machine learning approaches may be utilized to enhance the realism of synthetic data production.

The authors also plan to add more types of generators and new ways to display generators to facilitate the understanding of the model's design; 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. The inclusion of a quantitative or visual comparison component for verification purposes is also possible in the continuation of this line of study. As an added bonus, the system will use user-supplied input to construct synthetic data that closely resembles the input.

REFERENCES

"Evaluation in visualization: Some difficulties and recommended practices," Vis. Data Anal., vol. 9017, Feb. 2013, Art. no. 90170O, by B. S. Santos and P. Dias. [Online]. Proceeding: aspx?doi=10.1117/12.2038259 is available online at http://proceedings.spiedigitallibrary.org.

"Evaluating visualization approaches and tools: What are the key issues?" by B. S. Santos in Proc.Workshop Beyond Time Errors Novel Eval. Methods Vis. (BELIV), 2008, pp. 1_2.

"Criteria for a comparative study of visualization approaches in data mining," in Intelligent Systems Design and Applications, by R. Redpath and B. Srinivasan (Reference 3). Springer, 2003, vol. 2, no. 2, pp. 609–620, Berlin, Germany.

"Empirical studies in information visualization: Seven scenarios," IEEE Trans. Vis. Comput. Graphics, volume 18, issue 9, pages 1520–1536, September 2012. [4] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale.

"A survey on information visualization: Recent advancements and difficulties," S. Liu, W. Cui, Y. Wu, and M. Liu, 2016. Vis. Comput., Volume 30 Issue 12 (December 2014), Pages 1373–1393.

"A prototype application to create synthetic datasets for information visualization assessments," by 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, was published in the proceedings of the 22nd International Conference on Information Visualization (IV) in July 2018 (pp. 153–158).

"Data generators: A brief assessment of approaches and application cases with emphasis on testing," in Proc. IEEE 9th Int. Conf. Consum. Electron. (ICCE-Berlin), September 2019, pages 189_194. [7] S. Popi, B. Pavkovi, I. Veliki, and N. Tesli.

"Constraint-based automated test data generation," by R. A. DeMilli and A. J. Offutt [8] September 1991 issue of IEEE Transactions on Software Engineering, pages 900–910.

Automatic goal-oriented test data creation using a genetic algorithm and simulated annealing," M. Mann, O. P. Sangwan, P. Tomar, and S. Singh, 2016.

annealing," in Proceedings of the Sixth International Conference on Cloud Computing and Big Data Engineering (Con_uence), January 2016, pages 83–87.

"An strategy for test data creation based on genetic algorithm and delete mutation operators," by S. Rani and B. Suri, published in Proceedings of the 2015 International Conference on Advances in Computing and Communications Engineering (ICACCE), May 2015, pages 714–718, is a good example.

"Synthetic production of highdimensional datasets," IEEE Trans. Vis. Comput. Graphics, vol. 17, no. 12, pp. 2317_2324, Dec. 2011; G. Albuquerque, T. Lowe, and M. Magnor.

"SketchPadN-D: WYDIWYG sculpting and editing in high-dimensional space," by B.Wang, P. Ruchikachorn, and K. Mueller. December 2013 issue of IEEE Transactions on Visualization and Computer Graphics, pages 2060–2069.

"AxiSketcher: Interactive nonlinear axis mapping of visualizations via user drawings," IEEE Trans. Vis. Comput. Graphics, volume 23, issue 1, pages 221–230, January 2017. [13] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert.

"Synthetic data generator", by R. Liu, B. Fang, Y. Y. Tang, and P. P. K. Chan

for classification rule learning," Proceedings of the Seventh International Conference on Cloud Computing and Big Data, CCBD 2016, pages 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 creating and testing information discovery systems," in Proc. 3rd Int. Conf. Inf. Technol., New Gener. (ITNG), 2006, pp. 707_712.

"Synthetic data production capabilties for evaluating data mining techniques," in Proc. IEEE Mil. Commun. Conf. (MILCOM), October 2007, pp. 1_6; D. R. Jeske, P. J. Lin, C. Rendón, R. Xiao, and B. Samadi.

"Generating synthetic data for context-aware recommender systems," in Proc. 1st BRICS Countries Congr. Comput. Intell. (BRICS-CCI), September 2013, pages 563_567; M. Pasinato, C. E. Mello, M.-A. Aufaure, and G. Zimbro.

"SynSys: A Synthetic Data Generation System for Healthcare Applications," by J. Dahmen and D. Cook, published in Sensors, volume 19, issue 5, page 1181, 2019.

"A prototype of synthetic data generator," in Proceedings of the Sixth Colombian Computing Conference (CCC), May 2011, pages 1–6. [19] D. Garca and M. Millán.

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