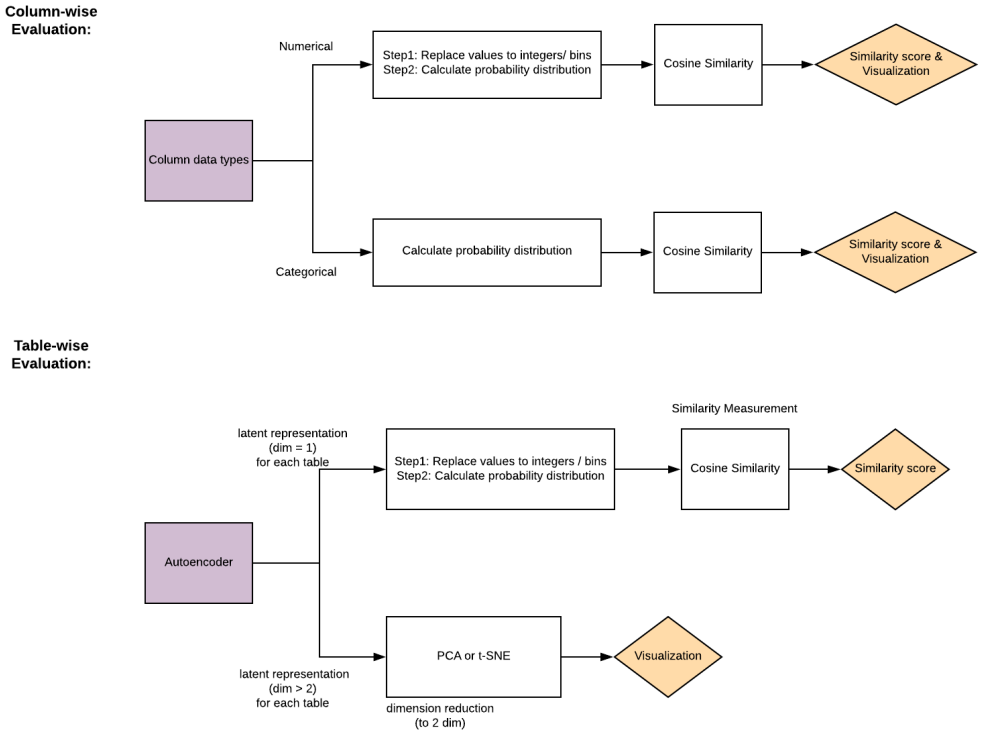
Statistical Similarity Methodology

To calculate the similarity between two tables, our methodology transfers the problem into calculating how different the synthetic data generated by GAN algorithm is from the original data. The smaller the difference between two data sets, the more similar they are with each other. In this methodology, the similarity between the synthetic data and original data will be evaluated in two perspectives: the corresponding columns and the relationships between all columns in each tables. The diagram of the metrics is shown below:



Overview of Datasets and results

Original datasets:

Data set	# of rows	# of columns	Column types	Column names
POC dataset	382 rows	7 columns	1 continuous ("los"), 6 categorical	'subject_id', 'insurance', 'ethnicity', 'gender', 'expire_flag', 'los', 'hospital_expire_flag'
MIMIC 3 data - Use case 1: Length of Stay Prediction	116354 rows	27 columns	1 continuous ("los"), 26 categorical	'final_id', 'ADMISSION_TYPE', 'INSURANCE', 'ETHNICITY', 'HOSPITAL_EXPIRE_FLAG', 'GENDER', 'EXPIRE_FLAG', 'LOS', 'certain conditions originating in the perinatal period', 'complications of pregnancy, childbirth, and the puerperium', 'congenital anomalies', 'diseases of the blood and blood-forming organs', 'diseases of the circulatory system', 'diseases of the digestive system', 'diseases of the genitourinary system', 'diseases of the musculoskeletal system and connective tissue', 'diseases of the nervous system', 'diseases of the respiratory system', 'diseases of the sense organs', 'diseases of the skin and subcutaneous tissue', 'endocrine, nutritional and metabolic diseases, and immunity disorders', 'external causes of injury and supplemental classification', 'infectious and parasitic diseases', 'injury and poisoning', 'mental disorders', 'neoplasms', 'symptoms, signs, and ill-defined conditions'
MIMIC 3 data - Use case 2: Mortality Prediction	58976 rows	19 columns	12 continuous, 7 categorical	SUBJECT_ID', 'HADM_ID', 'ADMISSION_TYPE', 'MARITAL_STATUS', 'ETHNICITY', 'HOSPITAL_EXPIRE_FLAG', 'GENDER', 'NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS', 'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS', 'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS', 'NUMNOTEVENTS'

Similarity Results with GAN Algorithms Overview:

Dataset Type	GAN algorithms	Table Similarity
POC Data (382 rows originally)	t-GAN (400 rows)	
	ct-GAN (382 rows)	
	t-GAN (10,000 rows)	
	ct-GAN	
MIMIC 3 data - Use case 1: Length of Stay Prediction (116,354 rows originally)	t-GAN (116,400 rows)	
	ct-GAN (rows)	
	t-GAN (rows)	
	ct-GAN	
MIMIC 3 data - Use case 2: Mortality Prediction (rows originally)	t-GAN (rows)	
	ct-GAN (rows)	
	t-GAN (rows)	
	ct-GAN	

1. Column-wise Similarity Evaluation

Distribution Similarity - Cosine Similarity and KL-divergence

*Description and Rationale for **KL-divergence***

Quantifying the difference between two distributions can be referred to calculating the statistical distance between them. However this can be challenging as it can be difficult to interpret the measure. Instead, we can use a more common measurement, *Kullback-Leibler Divergence* (KL-divergence), to measure the distribution similarity between the pair-columns in both original table and synthetic table. KL-divergence quantifies how much information is lost when we compare the synthetic column distribution with original one. More specifically, the KL-divergence is just a slight modification of our formula for entropy and essentially, in our case, is the expectation of the log difference between the probability of data in the original column distribution(p) with the synthetic column distribution(q). The formula is below:

For the probability distribution of continuous variable:

(A **continuous random variable** is defined as one which takes an infinite number of possible values.)

$$KL(P||Q) = \int p(x) log(\frac{p(x)}{q(x)}) dx$$

For the probability distribution of discrete variable:

(A **discrete random variable** is defined as one which may take on only a countable number of distinct values and thus can be quantified.)

$$KL(P||Q) = \sum p_i(x) log(\frac{p_i(x)}{q_i(x)})$$

KL divergence is not symmetric, which means that how different two distributions compared to each other is usually not the same. In this case, KL divergence would be used to calculate how different the synthetic data generated is from the original data. The intuition for KL divergence score is that when the probability for an event from p is large, but the probability for the same event in q is small, then it would result in a large divergence.

KL divergence can be used to measure continuous or discrete probability distributions, but in the latter case the integral of the events is calculated in stead of the sum of the probabilities of the discrete events. It requires the input data to have a sum of 1, it would not be a proper probability distribution otherwise. Therefore, transformation step before calculating the KL divergence is required, which converts the original column-wise data to an array of probability values of different events.

For discrete columns:

Step 1: Convert original data to probability values

Step2: Calculate the Cosine Similarity and KL divergence (if applicable)

Different techniques required for discrete and continuous data:

Different from how it is for discrete data, it is impossible to generate a specific and accurate probability value for each data points in the sample of continuous data. The probability of continuous data is defined as the area under the pdf curve, so theoretically the probability of each data point is 0. In our use case, the 'length of Stay' is the only continuous variable because both day and the time during the day are considered in this variable. We will round the values of length of stay to make it all integers for better calculating the the similarity of probability distribution. The probability values will be ordered based on the number of length stayed, the number of length stayed that does not show up in the rounded data would be assigned 0 as its probability. For example, if there is no patient staying in the ICU around 6 days, the probability of the value "6" will be assigned as 0.

*Limitation of **KL-divergence**:*

Even though KL divergence is a good measurement for common cases, it is restricted to be used on two distributions that have the same length of probability distribution. In the case when the generated discrete column has less number of events from the original one, we need to make sure including all events into the probability distribution of the synthetic one by adding 0 as the probability of the omitted event. According to KL-divergence's formula, the probability sequence put in should not contain zero, it would output infinity other wise.

Considering the special property of continuous data, we propose using cosine similarity for calculating the probability list of the continuous data, in order to maintain the closest probability distribution towards the original data.

*Description and Rationale for **Cosine Similarity***

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The formula referenced from wiki page is shown below, where A and B in our case is the array of frequency probabilities of each unique value in the synthetic column and original column respectively:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

For continuous columns:

Step 1: Transform numerical data into values of bins

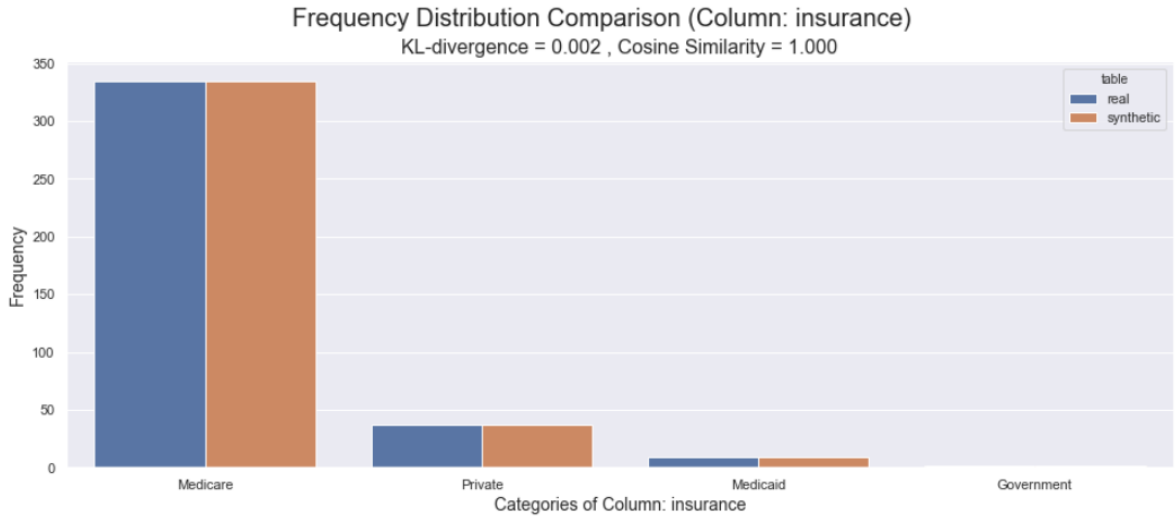
Step 2: Convert original data to probability values

Step3: Calculate the Cosine Similarity

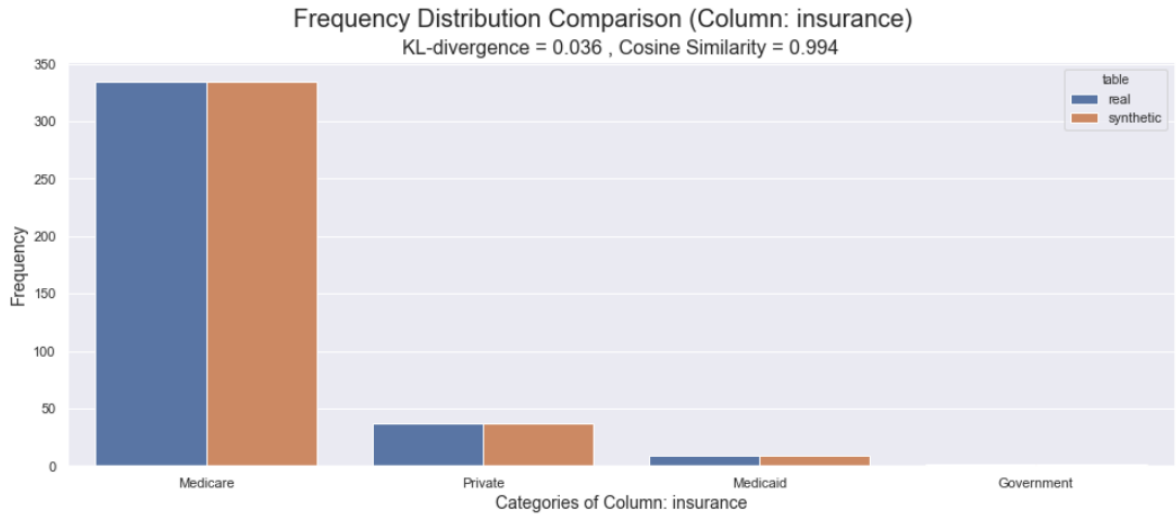
Demo:

1. **Discrete** column demo

KL divergence and frequency distribution of "insurance" pair-columns on **POC dataset** generated by **tGAN**:

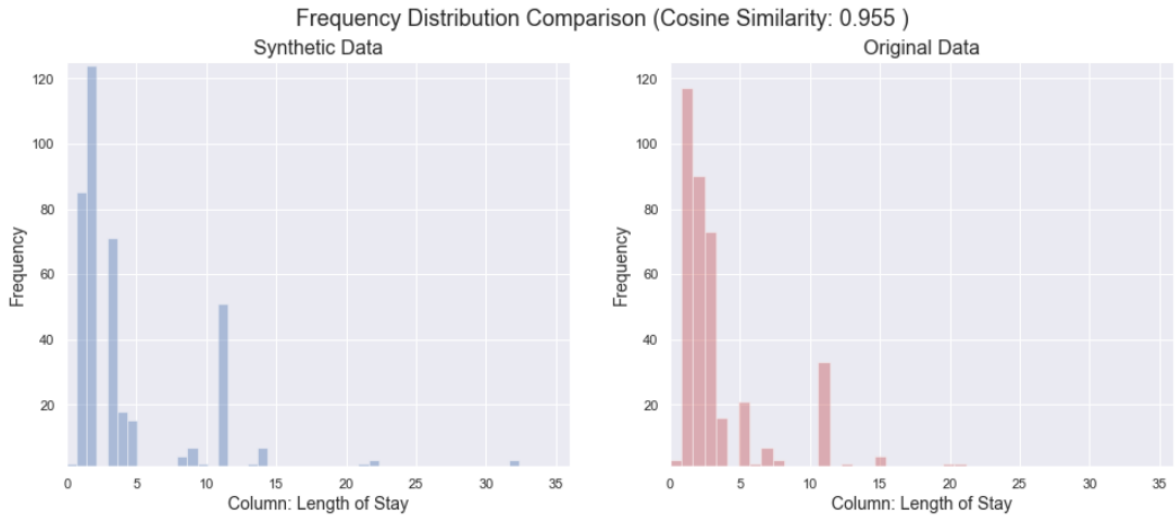


KL divergence and frequency distribution of "insurance" pair-columns on **POC dataset** generated by **ctGAN**:

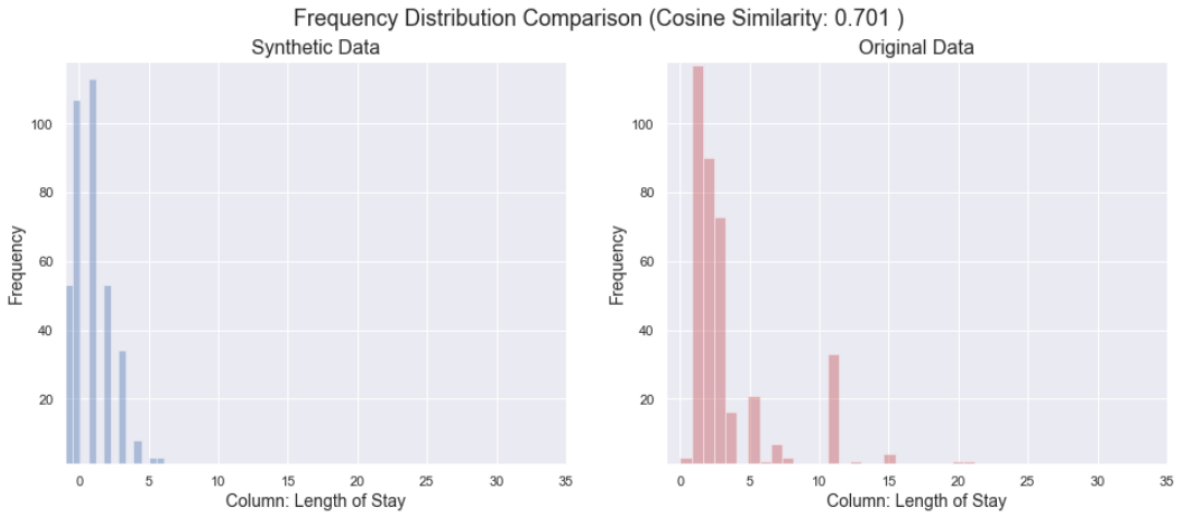


2. **Continuous** column demo

KL-divergence score and distributions of 'Length of Stay' pair-columns on dataset generated by **tGAN**:



KL-divergence score and distributions of 'Length of Stay' pair-columns on dataset generated by **ctGAN**:



Conclusion:

Regardless whether the column is discrete or continuous, every column generated by tGAN and ctGAN is highly similar statistics-wise to the original one. More specifically, tGAN outperforms ctGAN in terms of its column-wise similarity by showing a higher value of Cosine similarity score.

Limitations:

2. Table-wise Similarity Evaluation

2.1 Dimension Reduction Metric

Dimensionality reduction techniques are proposed to compress high-dimensional table in a way that generates a lower dimensional representation of original table, which further enables similarity score calculation and visualization of the relationships among columns in a table.

2.1.1 Auto-encoder

Description and Rationale for Autoencoder

Autoencoder is considered as a data compression algorithm and has long been thought to be a potential avenue for solving the problem of unsupervised learning, i.e. the learning of useful representations without the need for labels. Stated from Keras team about Autoencoder, autoencoders are not true unsupervised learning technique (which would imply a different learning process together), they are self-supervised technique, a specific instance of supervised learning where the targets are exactly the input data or generated from the input data.

In our case, the aim of an autoencoder is to learn a representation (embedding) of the relationships among multiple features in our table by setting the input and target data the same. Because Neural Networks can slowly approximate any function that maps inputs to outputs through an iterative optimization process called training. The embedding space has fewer dimensions (columns) than the feature data in a way that captures some latent structure of the feature data set.

Step1: Train an Autoencoder

Autoencoder can be broken in to 2 parts:

- **Encoder**: this part of the network compresses the input into an assigned number of vectors, which in our case for tabular data is the number of columns. The space represented by these fewer number of vectors is often called the *latent-space* or *bottleneck*. These compressed data that represent the original input are together called an “embedding” of the input.
- **Decoder**: this part of the network tries to reconstruct the input using only the embedding of the input. When the decoder is able to reconstruct the input exactly as it was fed to the encoder, you can say that the encoder is able to produce the best embeddings for the input with which the decoder is able to reconstruct well!

In order to make sure both synthetic table and original table are transformed exactly in the same process, we will train the auto-encoder using the original dataset and then predict on the synthetic dataset using the model built.

Step2: Extracting Embeddings from Autoencoder Network

After training the Autoencoder, extract the embedding for an example from the network. Extract the embedding by using the feature data as input, and read the outputs of the encoder layer. The embedding should contain information about reproducing the original input data, but stored in a compact way.

For the purpose of generating a similarity score, we would assign the dimension of embedding to be 1 so that we can use Cosine similarity or Euclidean distance to calculate the similarity. However, for visualization, we can choose either using autoencoder to compress both tables separately to a lower dimensional (but more than 2) embedding and then use PCA to further

compress the data to 2 dimension or using autoencoder to compress both tables directly to 2 dimensional embeddings. In this document, we will demonstrate later with the former method for the purpose of metric diversity.

Step3: Measuring Similarity from Embeddings

Next we would quantify the similarity for pairs of examples by using their embedding vectors.

We now have embeddings for the pair of tables. A similarity measure takes these embeddings and returns a number measuring their similarity. Remember that embeddings are simply vectors of numbers. To find the similarity between two vectors A=[a1,a2,...,an] and B=[b1,b2,...,bn], we can use Cosine Similarity that we mentioned before. It is a better measurement than Euclidean distance because it considers the angle of the vectors by putting them on a high dimensional space for comparison. And it outputs a score that ranges from 0 to 1 where 0 means that two vectors are oriented completely differently and 1 means that two vectors are oriented identically. This makes the comparison of the performances between different GAN algorithms easier.

Because there would be synthetic tables that have a different number of rows than the original table, we would consider the distribution as how we do for column-wise distribution evaluation instead of the pure values of the embeddings. We would take similar steps for

Demo:

Cosine similarity score using 5-dim latent representation from autoencoder using on dataset generated by **ctGAN**: 0.963.

Limitations:

1) The challenge applying Autoencoder with tabular data is the fact that each column represents its unique type of distribution. In other words, data types such as categories, ID numbers, ranks, binary values are all smashed into one sample table.

2) an autoencoder assume that all features (columns) in the table determines the similarity at the same degree, which means that this isn't the optimal choice when certain features could be more important than others in determining similarity.

2.1.2 PCA and t-SNE

For visualization, PCA or t-SNE can be used to generate a lower dimensional data and then visualize them to compare.

Principal Component Analysis (PCA) is a linear feature extraction technique. It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. It does so by calculating the eigenvectors from the covariance matrix. The eigenvectors that correspond to the largest eigenvalues (the principal components) are used to reconstruct a significant fraction of the variance of the original data.

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. However, t-SNE in Scikit learn has a cost function that is not convex according to Scikit learn documentation, which makes two visualizations of two tables generated by the same transformation function not comparable. Because they use different initializations for the transformation, we would get different results.

Therefore, in this document, we would use PCA as techniques for visualization comparison. t-SNE is still a great technique recommended for visualizing high-dimensional data because it is a probabilistic technique compared to PCA that is a mathematical one. Linear dimensionality reduction algorithms, like PCA, concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is essential that similar data points must be represented close together, which is something t-SNE does not PCA.

For visualization:

Step 1: Train autoencoder for a 5-dim embedding using original table, and then predict on synthetic table.

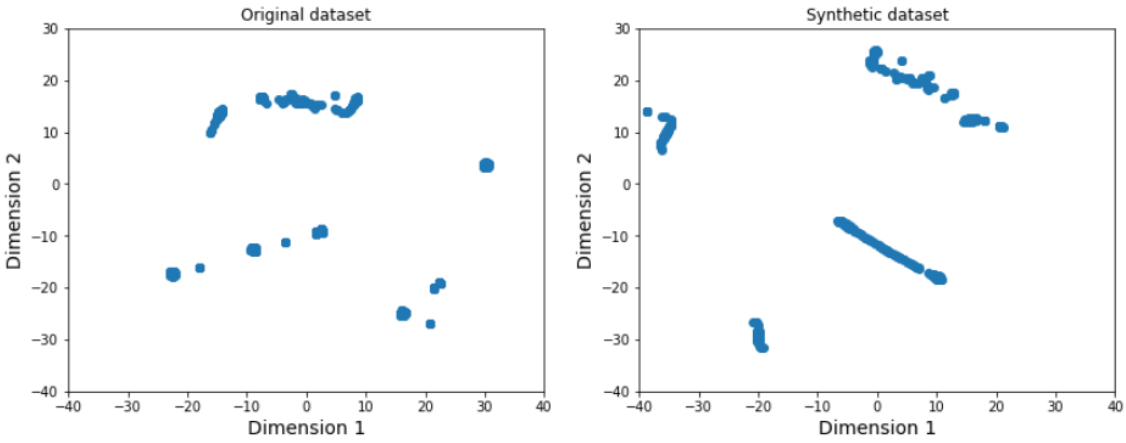
Step 2: Extract the embeddings of original table and synthetic table respectively

Step 3: Train a PCA transformation using the original embedding for reducing the dimension to 2, and use the same transformation for the synthetic embedding.

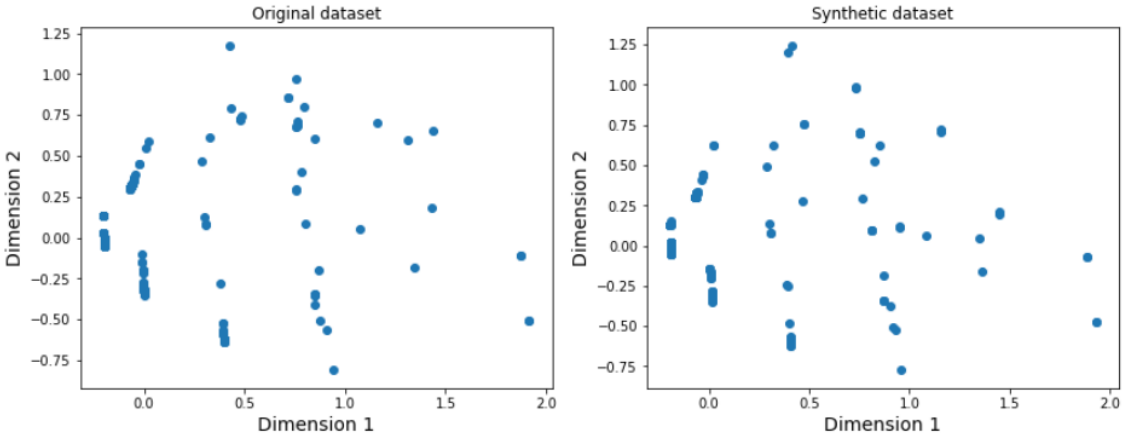
Step 4: Visualize the results of PCA with two scatter plots

Demo:

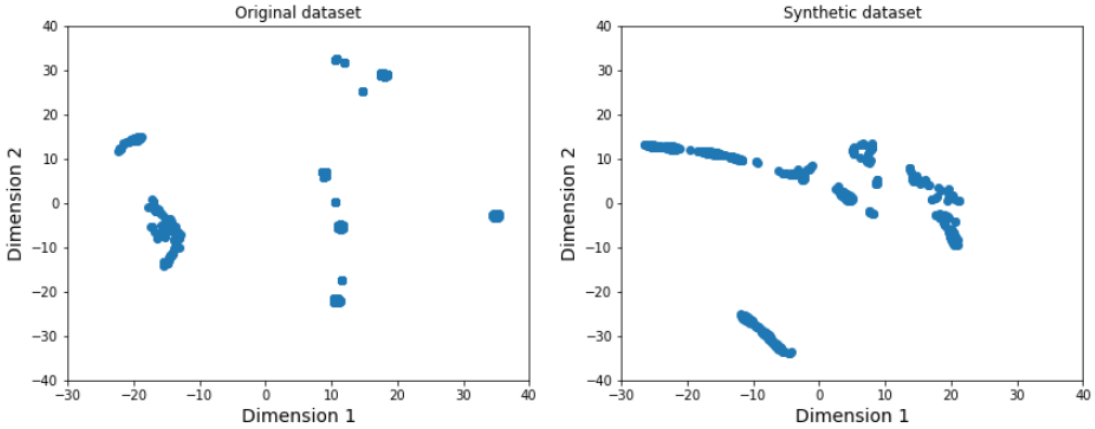
t-SNE visual result using 5-dim latent representation from autoencoder on **POC dataset** generated by **tGAN**:



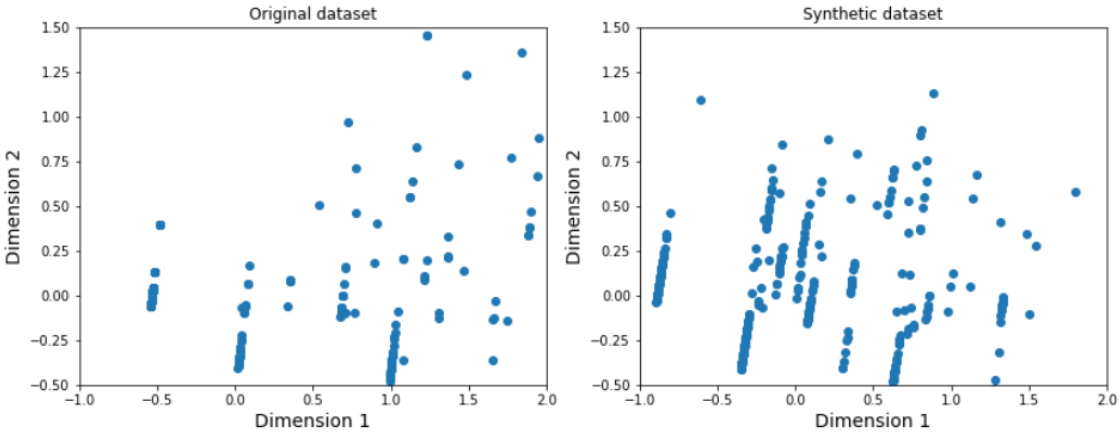
PCA visual result using 5-dim latent representation from autoencoder on **POC dataset** generated by **tGAN**:



t-SNE visual result using 5-dim latent representation from autoencoder on **POC dataset** generated by **ctGAN**:



PCA visual result using 5-dim latent representation from autoencoder on **POC dataset** generated by **ctGAN**:



2.2 Clustering Similarity Metric (WIP)

Gower distance + clustering + mutual information calculation

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html

Step1: Calculate Gower Distance Matrix

Step 2: Define the number of Clusters (Elbow Curve)

The k-means clustering algorithm requires the number of clusters (k) to be assigned manually. The elbow method is the most common method to validate the number of clusters. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k, and for each value of k calculate the sum of squared errors (SSE). Then plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k. So our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.

Step 3: Clustering Modeling

Step 4: Evaluating Clustering result (mutual information)