A report submitted for
INFO2017 POSTGRADUATE PROJECT A
in partial fulfilment of the requirements for the degree of Masters of Data science

Western Sydney University

NLP for clinical notes

Author:
Bibek Chaudhary
Student ID:
2210843

Author:
Pramod KC
Student ID:
22085342

Supervisor
Dr Rosalind Wang Wang
School of Computing, Engineering and Mathematics Western Sydney University

Co-Supervisor
Dr. Jim Basilakis
School of Computing, Engineering and Mathematics Western Sydney University

November 3, 2024

Abstract

In this paper, the topic seeks to shed more light on the application of Generative Adversarial Networks (GANs) in generation of synthetic data with high granularity for de-discovery mainly the MIMIC-III Patient Database. Especially in preserving confidentiality of health care data, the existing de-identification methodologies may lead to distorting task-specific statistical properties and patient-identification risk. Consequently, it is clear that unlike other methods of generating synthetic data, such as traditional models or infinite monolithic databases, GANs create individuals for simulations within constraints without distorting the initial correlation in the available data. First, this work investigates different types of GAN architectures, such as CTGAN, TVAE, DPGAN, and PATE-GAN, and examine how they perform in terms of statistical soundness, privacy preservation and how well they are suited towards machine learning applications. Among them, CTGAN is ranked high in terms lack of alteration in the data structure and so it is most suitable for cases where need for data utility comes first. On the other hand, TVAE provided the highest creation of privacy shield as its synthetic samples do not reach close the real data mitigating the chance of reidentification. However, GANs such as DPGAN and PATE-GAN preclude such internal dynamics by ensuring privacy levels supported by the concept of noise addition, processes, and ratios required for the statistical analysis. From the results gathered, it also suggest that CTGAN is good for making prediction and analysis in terms of structures while TVAE is suggested for applications with very strict privacy. Moreover, this information shows how GANs can work when it comes to tailoring the sharing of sensitive information in the health sector.

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Dr. Rosalind Wang Wang, for her invaluable guidance, encouragement, and insightful feedback throughout this project. Her expertise and support have been instrumental in shaping our research and achieving our objectives.

We would also like to thank our co-supervisor, Dr. Jim Basilakis, for his valuable suggestions and constant support, which helped us refine our work and approach the project with a critical perspective.

We are grateful to Western Sydney University and the School of Computing, Engineering, and Mathematics for providing the resources and facilities necessary to complete this study. Additionally, we acknowledge the assistance provided by our friends, family, and peers who offered their support and encouragement.

Finally, we would like to extend our appreciation to everyone involved, directly or indirectly, in helping us complete this project. Thank you for your encouragement, patience, and understanding throughout this journey.

Table Of Contents

Contents

1 CHAPTER 1: INTRODUCTION

In today's world data is consider most valuable resource. And health sectors have collected huge amount of data which can be use for various purpose. For instance, to find the relation between disease and predict the possibility for disease based on patient medical history and so more. As there are lots of data at he same time it is hard to get data. Data are expensive and time consuming to get one. Health data are not publicly available because it contains personal in formation of the patient and there is risk of privacy breach. For this reason anonymisation of data is necessary. It is hard to anonymise data, it is not simple as removing patient name, date of birth and address. Attackers can make relation among disease history and find the identity. Therefore, generating synthetic data is more suitable for medical data. part by Bibek

Synthetic data are artificially created datasets with similar properties and patterns of the orig inal real-life dataset but do not contain any genuine information from it. With its ability to protect privacy and boost data availability, it is now extensively in applications related to data science and machine learning. Generating synthetic data can be challenging because while gen erating synthetic data utility and privacy of the data should be consider. Utility indicates how much useful, valuable the data to be analyzed and derived insights for decision making. The data remains usable for its original intended purpose, e.g., as input to statistical techniques, building machine learning models or conducting a specific predictive analysis. For an accurate and valid result in such projects, one requires data that carries high utility. Privacy means anonymizing the information in data which refers to individuals personal privacy. Which means making sure the data does not have any sort of personal or identifying information. This is especially true of personal identifiable information, health records or anything sensitive. Balancing utility and privacy can be tricky at times when considering measures, like anonymization or encryption to safeguard privacy may lead to a loss of data utility by hiding details necessary for analysis. On the side maintaining the highest level of usability could jeopardize privacy when an excessive amount of confidential data is kept or revealed in the information.

2 CHAPTER 2: LITERATURE REVIEW

2.1 History

Historically, de-identification techniques, such as anonymization and differential privacy, were the primary solution applied as far as the need for protection of Personally Identifiable Information (PII) is concerned. Among such are methods like data redaction and ensuring to achieve two conflicting aspects of synthesizing characters that have no meaning, making the de-identified information untraceable. part by Bibek

In more recent years, there has been a shift, and several authors have stated that even statistical disclosure control techniques, particularly âtraditional' methods such as random generalization, are not sufficient to assure the privacy of respondents. For instance, they have revealed that de-identified information can be easily re-identified. In their study, they were able to re-identify 99.98% of the American population with only 15 keystrokes [Rocher et al., 2019]. This was a significant revelation because it questions the existence and certainly the applicability of the current methods in the era of new data protection rules like the General Data Protection Regulation.

proposition Traditional methods like anonymization and differential privacy just aren't cutting it anymore when it comes to protecting sensitive personal data. Recent studies have shown that it's alarmingly easy to re-identify individuals even after these techniques are applied. So,

we're leaning toward a new approach: using *Generative Adversarial Networks (GANs)* [Goodfellow et al., 2014] to create synthetic data. GANs are a cutting-edge deep learning technology that can generate data strikingly similar to real datasets without exposing any actual personal information. While GANs have mostly been used to create realistic images, we believe they can be adapted to handle tabular data from databases, which is a big step forward. Yes, there are challengesâlike ensuring the synthetic data is useful for real-world applications and managing the computational demandsâbut we think these can be tackled with the right strategies. By embracing GANs, we can better balance data utility and privacy, comply with regulations like GDPR, and ultimately make data sharing safer and more effective for everyone involved.

2.2 Reason to choose GANs

In 2014, *Ian Goodfellow introduced GANs* [Goodfellow et al., 2014], a kind of machine learning part model that has gained popularity, particularly in the picture generating space. The generator by and discriminator neural networks essentially compete with one another to function. Whereas Bibek the discriminator seeks to distinguish between authentic and fraudulent data, the generator attempts to produce artificial data that appears authentic. This rivalry gradually makes the generator's outputs more realistic. Why, therefore, are GANs a smart choice for creating synthetic data and protecting data privacy? To begin with, GANs are capable of generating high-quality synthetic datasets that replicate the statistical characteristics of real data while excluding any personal information. This implies that there is no privacy risk when using these artificial datasets for analysis, machine learning, or sharing with outside parties. It's similar to having both things at once. Second, GANs have a wide range of applications. Researchers have modified GAN architectures to handle many data types, including tabular data from databases, despite its well-known usage in producing lifelike images of faces, cats, or even complete scenes. The purpose of models such as the Conditional Tabular GAN (CTGAN) is to produce synthetic tabular data while preserving the relationships and patterns present in the original dataset. Another significant benefit is that GANs assist firms in adhering to data privacy laws such as GDPR. The synthetic data is less likely to violate privacy rules because it doesn't contain actual personal information. In addition to lowering legal risks, this increases user and consumer trust in the face of growing concerns about the use of their data. Of course, there are difficulties in putting GANs into practice. Mode collapse, in which the generator generates a limited variety of outputs, is one problem that can make training GANs computationally demanding and occasionally challenging. Ongoing studies and developments, however, are constantly enhancing GAN training methods and effectiveness.

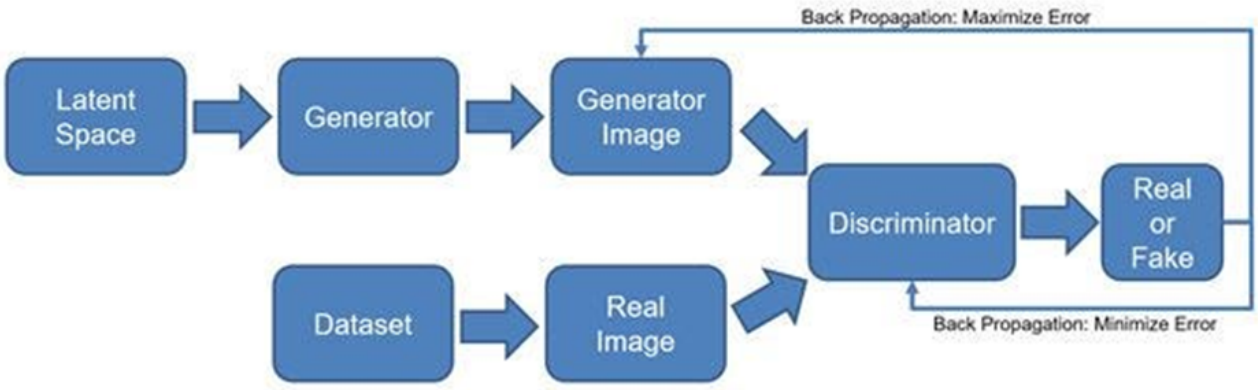


Figure 1: Architecture of GAN.

Algorithm:

- **Generator Initialisation:** The generator kicks things off by creating synthetic data based purely on random noise since it hasn't yet learned anything about the real data.
- **Discriminator Evaluation:**The discriminator, trained to tell real from fake, receives a mix of real data and the synthetic data generated by the generator.
- **Initial Discriminator Success:** At first, the generator's outputs are basically noise, so the discriminator has no trouble spotting them as fake.
- **Generator Adjustment:**Each time the discriminator flags something as fake, the generator gets feedback and tweaks its internal settings to make its next attempt a little closer to real.
- **Adversarial Improvement Loop:** Each time the discriminator flags With every cycle, the generator's outputs get more realistic, forcing the discriminator to improve its detection. This back-and-forth gradually pushes both networks to get better at their tasks.
- **Convergence:** After many rounds, the generator becomes so good at mimicking real data that the discriminator can no longer tell real from fake-it's left to guess with 50/50 accuracy.
- **GAN Achieves Realistic Results:** At this point, the GAN is fully trained, and the generator can produce realistic synthetic data. This synthetic data is now useful for tasks like data sharing and privacy-preserving analysis without revealing any actual information from the original data.

2.3 Existing Research

With the growing concerns about data privacy the field of synthetic data generation has gained significant interest. Researchers have been exploring various methods to create realistic synthetic datasets that can be used for analysis and modeling without exposing sensitive information. Some of the major approach is mentioned below. part by Bibek

2.3.1 TGAN(Tabular GAN)

Generative Adversarial Networks (GANs) are used in the TGAN process to create synthetic tabular data. The publication *"Synthesizing Tabular Data using Generative Adversarial Networks" by Xu and Veeramachaneni (2018)* [Xu and Veeramachaneni, 2018] fdescribes this method in depth. Python package called TGAN on PyPI python package is also avaliable to use. GANs are strong models that can create new samples that have the same distribution after learning the dataset's probability distribution. Because it can handle both continuous and categorical variables, TGAN specializes in tabular data with mixed variable types. Techniques TGAN generates data one column at a time using Long Short-Term Memory (LSTM) networks with attention techniques to address the complexity of tabular data.

The synthetic data should closely resemble the real data in terms of the relationships between variables (such as mutual information). Preparing Data Numerical variables are represented as multinomial distributions after being converted into values between -1 and 1. This reversible change improves the model's ability to learn numerical data. One-hot encoding is used to convert categorical variables. This approach should be improved, though, as it might not

fully capture the subtleties of categorical distributions. Managing Distributions in Multiple Modes Numerical variables in tabular datasets frequently have numerous peaks (multimodal distributions) rather than a simple distribution. Statistical Similarity: The associations between variables (such mutual information) in the synthetic data should be quite similar to those in the real data.

Getting Data Ready After being transformed into values between -1 and 1, numerical variables are represented as multinomial distributions. The model's capacity to learn numerical input is enhanced by this reversible modification. Categorical variables are converted via one-hot encoding. However, this method has to be refined because it may not adequately account for the nuances of categorical distributions. Controlling Distributions in Various Ways Instead of having a simple distribution, numerical variables in tabular datasets sometimes contain multiple peaks (multimodal distributions).

2.3.2 CTGAN(Conditional Tabular GAN)

Modeling intricate tabular data distributions and producing superior synthetic data are the goals of CTGAN. Several special issues that arise with tabular data are addressed by CTGAN, which is described in detail in the paper *"Modeling Tabular Data using Conditional GAN"* [Xu et al., 2019a]. Discrete and continuous variables are frequently included in tabular data. The proper transformations are applied by CTGAN, which uses tanh for continuous data and softmax for categorical data.Continuous variables frequently show numerous modes and hardly ever follow a Gaussian distribution in tabular data. Using methods like as variational Gaussian mixture models, CTGAN handles these complications by implementing mode-specific normalization.There is a significant imbalance in many category variables, with some categories predominating in the data. Minor categories are sufficiently represented during training thanks to CTGAN's use of a conditional generator and training-by-sampling techniques.

By normalizing data while maintaining the underlying distribution patterns, mode-specific normalization solves the problem of non-Gaussian and multimodal distributions.Conditional Generator Enhances the representation of minority classes by enabling the model to provide data conditioned on particular qualities. A resampling method that guarantees effective sampling of all categories during training, particularly those that are less frequent.

```
metadata = SingleTableMetadata()
metadata.detect_from_dataframe(data=train)

# Initialize the CTGAN synthesizer with the metadata
ctgan = CTGANSynthesizer(metadata)
# Fit the model on the original data
ctgan.fit(train)
# Generate synthetic data with the same number of rows and columns
train_ctgan = ctgan.sample(num_rows=len(train))
# Display the synthetic data
print(train_ctgan.head())


```

2.3.3 Differentially Private GAN (DPGAN)

The goal of DPGAN [Xie et al., 2018] is to create synthetic data while protecting the original dataset's privacy. Standard GANs are problematic when handling sensitive data, such as

medical records, because they may memorize and replicate specific data points from the training set. DPGAN targets the Wasserstein distance calculation during training by introducing precisely crafted noise into the gradients. By adding noise, the discriminator and generator are prevented from learning certain data points. DPGAN offers a mathematical guarantee of privacy by introducing noise into the gradients, making sure that the result is not substantially impacted by the inclusion or absence of a single data point.

The noise parameter ϵ controls the privacy-utility trade-off. Selecting an appropriate ϵ requires experimentation to balance data utility and privacy. The approach focuses on preserving privacy during training rather than post-processing the final model parameters, which often leads to better utility.

2.3.4 PATE-GAN

In order to ensure strict privacy when producing synthetic data, PATE-GAN incorporates the Private Aggregation of Teacher Ensembles (PATE) [Jordon et al., 2018] framework into GANs. Disjoint subsets of the data are used to train several discriminators, or teachers. Every educator casts a vote regarding the veracity of the data produced. learns to create data without having direct access to the raw data by imitating the teacher discriminators' consensus. To protect differential privacy, the votes cast by teacher discriminators are combined with additional noise. By doing this, the student generator is unable to deduce private information about specific data points. PATE-GAN guarantees that the generator cannot, even indirectly, recover the original data by dividing the discriminator into many teachers and introducing noise during aggregation. Ideal for industries like healthcare or finance where stringent privacy guarantees are necessary.

2.3.5 G-PATE

Building on the PATE-GAN framework, G-PATE [Long et al., 2021] seeks to simplify the architecture and increase the effectiveness of privacy budget usage. connects instructor discriminators straight to the generator, eliminating the requirement for a student discriminator. presents a gradient aggregator that gathers and makes differentially private gradients from teacher discriminators before sending them to the generator. G-PATE guarantees that resources are used where they have the greatest influence on data creation by concentrating the privacy budget on the generator. Since the data does not need to be made publicly available or differentially private, the discriminator can be trained on actual data without privacy restrictions. The discriminator can learn more efficiently from the real data if it is not subject to as many privacy restrictions, which could result in higher-quality synthetic data

2.3.6 TVAE

The basic idea of the TVAE-GAN model is the generation of synthetic data at a high quality with the combined strengths of VAE and GAN. It differs from the basic GAN, which usually has mode collapse and does not provide diverse outputs, since it leverages the advantages of probabilistic modeling in VAE to catch the underlying data distribution effectively. During training, it combines reconstruction loss from VAE with adversarial loss of GAN, so as to let the model learn the global structure and local detail of the data. This dual approach strengthens the realistic generation of samples while providing a way to control the latent space to generate varied outputs. The framework allows synthesis of data generated for high-dimensional tasks in such a way that not only are the samples generated realistic, but also diverse; hence, overcoming some limitations found in traditional GAN architectures, like image generation.

```
    metadata = SingleTableMetadata()
metadata.detect_from_dataframe(data=train)
# Initialize the TVAE synthesizer with the metadata
tvae = TVAESynthesizer(metadata)
# Fit the model on the original data
tvae.fit(train)
# Generate synthetic data with the same number of rows and columns
train_tvae = tvae.sample(num_rows=len(train))
# Display the synthetic data
print(train_tvae.head())

```

3 CHAPTER 3: METHODOLOGY

3.1 MIMIC - III Dataset

The Medical Information Mart for Intensive Care Unit is known as MIMIC-III. Several tables in this open source database can be merged to provide a variety of use cases. Data is available via the link below, and access to it requires a few certificates, which can be earned by doing online trainings. An extensive clinical dataset of more than 40,000 patients admitted to intensive care units was created by the MIT Lab and is available as an open source dataset. part by Bibek

Source: https://mimic.physionet.org/gettingstarted/overview/

Overview of MIMIC-III data:

To get a further insight, various clinical table datasets need to be available, such as the Admissions, Patients, CPT events, Chart events, Diagnosis and others Involving over 45,000 patients that represent patient history and demographic details together with information about the status of the patient like gender, ethnicity and marital status Occupying rather more than 65,000 of ICU feedbacks in Beth Israel Deaconess Medical Centre in Boston, Massachusetts over the course of the decade from 2001 to 2012 years. The information in Issue would include personal information, such as the details of the patient such as those in demographics, diagnosis, laboratory tests, prescriptions and the like. Applications that could be performed include varied fields of study in different kinds of data such as health data (Data related to the prescriptions), financial data (Data related to the payer) and data about the people (Data relative to the patient) In this regard the project will concentrate on particular tables in the database, which will be a subset rather than all 26 tables. Use case, which are detailed in the section below, was produced by combining many tables from this database. Initial comprehension is essential to comprehending the variable names used in subsequent documentation, as these use cases will be utilized in modules such as statistical similarity and privacy risk.

3.2 Use Case: Mortality Prediction

Goal part by Bibek

Use case for MIMIC-III dataset can be prediction of mortality.This use case, inspired by a Kaggle kernel (reference below), focuses on predicting patient mortality based on the frequency of interactions between a patient and the hospital. Using counts of lab tests, prescriptions, and procedures as predictors, it also evaluates privacy risk through PII columns like GENDER, ETHNICITY, and MARITAL_STATUS. This is framed as a classification problem where the goal is to predict if a patient will expire during a single hospital admission.

Reference: Kaggle Kernel - Predict Hospital Mortality (MIMIC-III)

Methodology

Tables Used in the Analysis:

- **Patients:** Lists every unique patient in the database with details like SUBJECT_ID and GENDER.
- **Admissions:** Contains data on every hospitalization for each patient, with columns such as SUBJECT_ID, HADM_ID, HOSPITAL_EXPIRE_FLAG, MARITAL_STATUS, ETHNICITY, and ADMISSION_TYPE.
- **CallOut:** Records information on ICU discharge planning, including when patients were cleared for discharge and the actual discharge time, with columns like SUBJECT_ID, HADM_ID, and NUMCALLOUT (total count of callout events).
- **CPTEvents:** Captures procedures using Current Procedural Terminology (CPT) codes, with fields such as SUBJECT_ID, HADM_ID, and NUMCPTEVENTS (count of CPT events).
- **Diagnosis_ICD:** Includes hospital-assigned diagnoses coded by the ICD system, with columns like SUBJECT_ID, HADM_ID, and NUMDIAGNOSIS (count of diagnosis entries).
- **Inputevents_CV**: Tracks intake data for patients monitored via the Philips CareVue system in the ICU, including SUBJECT_ID, HADM_ID, and NUMINPUTEVENTS (count of input events).
- **Labevents**: Contains laboratory test results from both hospital and outpatient clinics, with columns such as SUBJECT_ID, HADM_ID, and NUMLABEVENTS (count of lab events).
- **Noteevents**: De-identified notes from medical staff, ECG reports, imaging summaries, and discharge notes, with SUBJEC_ID, HADM_ID, and NUMNOTEVENTS (count of note entries).
- **Outputevents**: Holds output information for ICU patients, with fields SUBJECT_ID, HADM_ID, and NUMOUTEVENTS (count of output events).
- **Prescriptions**: Includes medication orders, with columns such as SUBJECT_ID, HADM_ID, and NUMRX (count of prescription events).
- **Procedureevents_mv**: Tracks procedures for patients monitored in the ICU with iMDSoft MetaVision, including SUBJECT_ID, HADM_ID, and NUMPROCEVENTS (count of procedure events).
- **MICROBIOLOGYEVENTS**: Contains microbiology test results, with fields SUBJECT_ID, HADM_ID, and NUMMICROLABEVENTS (count of microbiology lab events).
- **Procedures_icd**: Lists patient procedures coded by ICD, with columns such as SUBJECT_ID, HADM_ID, and NUMPROC (count of procedures).
- **Transfers**: Details patient transfers within the hospital, including ICU admissions and discharges, with fields like SUBJECT_ID, HADM_ID, and NUMTRANSFERS (count of transfers).

Steps to Create the Analytical Dataset

- Load CSV Files: Import the comma-separated files downloaded from MIMIC-III.
- Roll Up Tables: For each table, aggregate data by SUBJECT_ID and HADM_ID to obtain counts of various events or interactions between the patient and hospital. This step provides a summary of events for each hospital admission.
- Merge Tables: Use SUBJECT_ID and HADM_ID as composite keys to join all tables, forming a consolidated dataset for analysis.

Code

- Import required libraries and read csv files
- Function to roll up tables
- Merge all tables
- Exploratory Analysis

Import required libraries

```
## Model Compatibility Module
from sklearn import tree, neighbors, datasets, linear_model, svm, naive_bayes
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.model_selection import cross_validate,GridSearchCV
from sklearn import metrics, linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction import DictVectorizer
from sklearn.utils.multiclass import unique_labels
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
import xgboost as xgb
import numpy as np
import warnings
from keras import models, regularizers, layers, optimizers, losses
import tensorflow as tf
warnings.filterwarnings('ignore')
```

Exploratory Analysis

```
    #One Hot encoding
mortality_pred = mimic_data.drop(columns=['subject_id','hadm_id','ethnicity',
'marital_status'])
mortality_pred = pd.get_dummies(mortality_pred,prefix=['admission_type',
'gender'])
mortality_pred.head()
```

hospital_expire_flag	NUMCALLOUT	NUMCPTEVENTS	NUMDIAGNOSIS	NUMOUTEVENTS	NUMRX
0	0.0	4.0	21	3.0	54.0
1	0.0	0.0	6	261.0	0.0
1	0.0	1.0	9	32.0	36.0
0	1.0	13.0	14	25.0	105.0
1	0.0	5.0	14	22.0	99.0

NUMPROCEVENTS	NUMMICROLABEVENTS	NUMPROC	NUMTRANSFERS	NUMINPUTEVENTS	NUMLABEVENTS
0.0	20.0	7.0	3	6.0	251
0.0	13.0	2.0	2	573.0	700
0.0	3.0	1.0	2	430.0	148
0.0	2.0	2.0	6	163.0	302
0.0	13.0	4.0 2	821.0	287	

NUMNOTEVENTS	admission_type_ELECTIVE	admission_type_EMERGENCY	admission_type_URGENT	gender_F	gender_M
0.0	False	True	False	True	False
0.0	False	True	False	True	False
0.0	False	True	False	True	False
0.0	False	True	False	True	False
0.0	False	True	False	False	True

```
    ## target variable hospital_expire_flag is not balanced
mortality_pred.groupby('hospital_expire_flag').size().plot.bar()
plt.show()
```

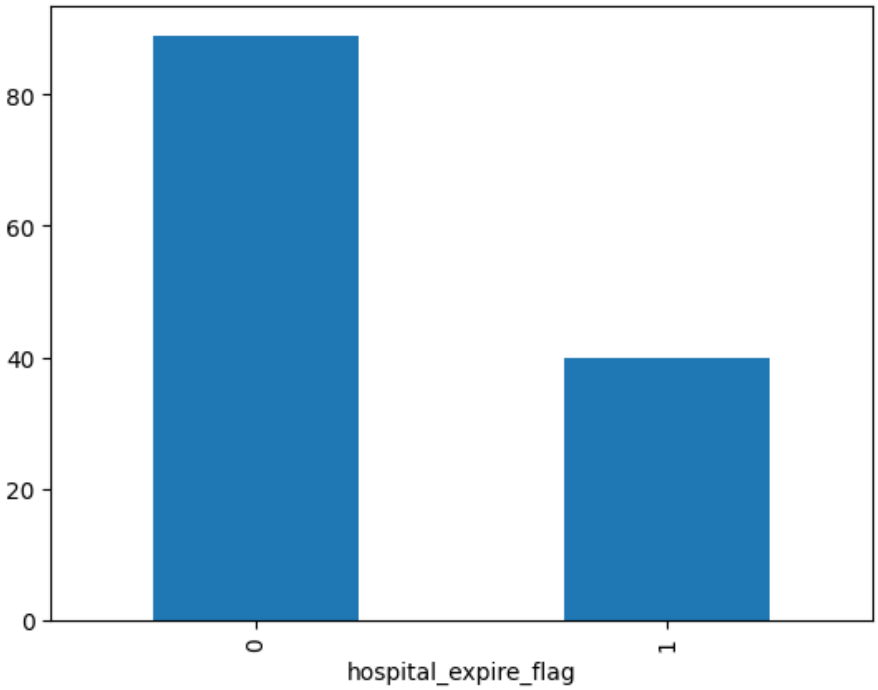


Figure 2: Bar graph of Hospital Expire.

```
    # Split data using target variable
train, test = train_test_split(mortality_pred, test_size=0.20,
random_state=42,stratify=mortality_pred['hospital_expire_flag'])

```

```
   metadata = SingleTableMetadata()
metadata.detect_from_dataframe(data=train)

# Initialize the CTGAN synthesizer with the metadata
ctgan = CTGANSynthesizer(metadata)
# Fit the model on the original data
ctgan.fit(train)
```

```
# Generate synthetic data with the same number of rows and columns
train_ctgan = ctgan.sample(num_rows=len(train))
# Display the synthetic data
print(train_ctgan.head())
```

hospital_expire_flag	NUMCALLOUT	NUMCPTEVENTS	NUMDIAGNOSIS	NUMOUTEVENTS	NUMRX
0	0.0	0.0	25	58.0	176.0
0	2.0	4.0	18	49.0	138.0
1	1.0	0.0	18	28.0	48.0
0	0.0	0.0	21	0.0	115.0
1	0.0	0.0	21	293.0	68.0

NUMPROCEVENTS	NUMMICROLABEVENTS	NUMPROC	NUMTRANSFERS	NUMINPUTEVENTS	NUMLABEVENTS
30.0	15.0	4.0	6	42.0	1721
37.0	25.0	6.0	3	30.0	777
4.0	16.0	1.0	5	265.0	54
59.0	32.0	3.0	9	282.0	568
3.0	4.0	4.0	5	0.0	233

NUMNOTEVENTS	admission_type_ELECTIVE	admission_type_EMERGENCY	admission_type_URGENT	gender_F	gender_M
0.0	False	True	False	False	True
0.0	False	True	False	False	False
0.0	False	True	False	True	False
0.0	False	True	False	False	True
0.0	False	True	False	True	False

```
  metadata = SingleTableMetadata()
metadata.detect_from_dataframe(data=train)
# Initialize the TVAE synthesizer with the metadata
tvae = TVAESynthesizer(metadata)
# Fit the model on the original data
tvae.fit(train)
# Generate synthetic data with the same number of rows and columns
train_tvae = tvae.sample(num_rows=len(train))
# Display the synthetic data
print(train_tvae.head())
```

hospital_expire_flag	NUMCALLOUT	NUMCPTEVENTS	NUMDIAGNOSIS	NUMOUTEVENTS	NUMRX
0	1.0	2.0	17	32.0	59.0
0	1.0	16.0	17	19.0	87.0
0	1.0	3.0	11	60.0	38.0
0	0.0	5.0	11	17.0	51.0
0	0.0	7.0	10	90.0	70.0

NUMPROCEVENTS	NUMMICROLABEVENTS	NUMPROC	NUMTRANSFERS	NUMINPUTEVENTS	NUMLABEVENTS
4.0	2.0	1.0	3	13.0	54
12.0	5.0	2.0	4	3.0	519
1.0	1.0	2.0	5	0.0	237
0.0	1.0	0.0	4	213.0	95
1.0	7.0	3.0	4	92.0	515

NUMNOTEVENTS	admission_type_ELECTIVE	admission_type_EMERGENCY	admission_type_URGENT	gender_F	gender_M
0.0	False	True	False	False	True
0.0	False	True	False	False	True
0.0	False	True	False	False	True
0.0	False	True	False	True	False
0.0	False	True	False	True	False

```
    ## Data Correction
train_ctgan = train_ctgan.round(0)
train_ctgan[train_ctgan < 0] = 0
```

```
#train_ctgan.head()

train_tvae = train_tvae.round(0)
train_tvae[train_tvae < 0] = 0
#train_tvae.head()
```

```
    import matplotlib.pyplot as plt
from sklearn.metrics import (
    accuracy_score, classification_report, confusion_matrix,
    roc_auc_score, roc_curve, precision_recall_curve, auc
)
import numpy as np

def evaluate_model(alg, x_train, y_train, x_test, y_test, class_weight=None):
    # Check if the algorithm supports class_weight and fit accordingly
    if class_weight and hasattr(alg, "fit"):
        alg.fit(x_train, y_train, class_weight=class_weight)
    else:
        alg.fit(x_train, y_train)

    # Predictions for training and testing
    y_train_pred = alg.predict(x_train)
    y_test_pred = alg.predict(x_test)

    # If model has predict_proba, calculate probabilities for ROC and PRC
    if hasattr(alg, "predict_proba"):
        y_test_proba = alg.predict_proba(x_test)[:, 1]
    else:
        # If predict_proba isn't available, set to None (e.g.,
        #for SVM without probability output)
        y_test_proba = None

    # Metrics
    train_accuracy = accuracy_score(y_train, y_train_pred)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    class_report = classification_report(y_test, y_test_pred)
    conf_matrix = confusion_matrix(y_test, y_test_pred)

    # Print main results
    print(f"Training Accuracy: {train_accuracy:.2f}")
    print(f"Testing Accuracy: {test_accuracy:.2f}")
    print("Classification Report:\n", class_report)
    print("Confusion Matrix:\n", conf_matrix)

    # Calculate AUC-ROC if probabilities are available
    if y_test_proba is not None:
        auc_roc = roc_auc_score(y_test, y_test_proba)
        print(f"AUC-ROC: {auc_roc:.2f}")

        # ROC Curve
```

```
        fpr, tpr, _ = roc_curve(y_test, y_test_proba)
        plt.figure(figsize=(10, 5))
        plt.plot(fpr, tpr, label=f"AUC = {auc_roc:.2f}")
        plt.plot([0, 1], [0, 1], linestyle='--')
        plt.xlabel("False Positive Rate")
        plt.ylabel("True Positive Rate")
        plt.title("ROC Curve")
        plt.legend()
        plt.show()

        # Precision-Recall Curve
        precision, recall, _ = precision_recall_curve(y_test, y_test_proba)
        auc_pr = auc(recall, precision)
        plt.figure(figsize=(10, 5))
        plt.plot(recall, precision, label=f"AUC = {auc_pr:.2f}")
        plt.xlabel("Recall")
        plt.ylabel("Precision")
        plt.title("Precision-Recall Curve")
        plt.legend()
        plt.show()
    else:
        print("AUC-ROC and Precision-Recall curves are not
        available for this model.")
```

```
    features =['NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS',
        'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS',
        'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS',
        'NUMNOTEVENTS', 'admission_type_ELECTIVE', 'admission_type_EMERGENCY',
        'admission_type_URGENT', 'gender_F',
        'gender_M']
x_train = train[features]
y_train = train['hospital_expire_flag']
x_test = test[features]
y_test = test['hospital_expire_flag']
```

```
    scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled=scaler.transform(x_train)
x_test_scaled=scaler.transform(x_test)
```

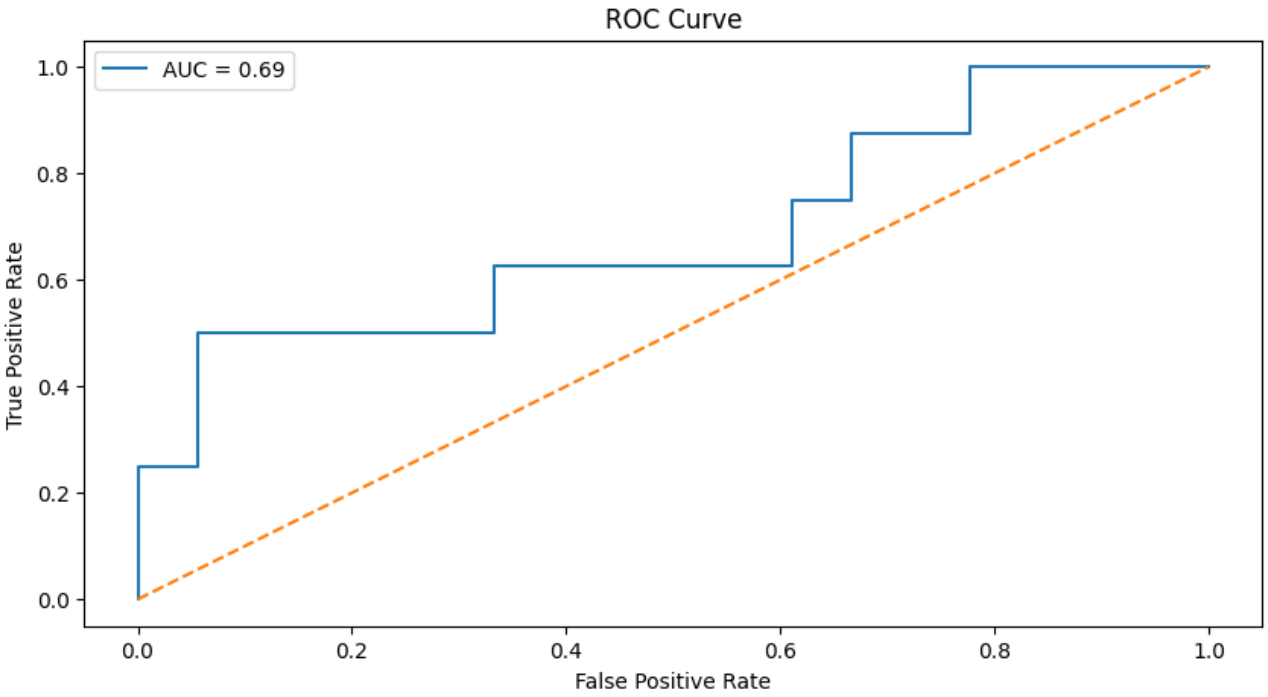
```
    param_test ={'C':[0.001, 0.01, 0.1, 1, 10, 100],
                 'penalty':('l1','l2')}
lr=linear_model.LogisticRegression(random_state=10,multi_class='ovr',
class_weight='balanced')
gsearch = GridSearchCV(estimator=lr,
                       param_grid=param_test,
                       scoring='roc_auc',
                       n_jobs=-1,
```

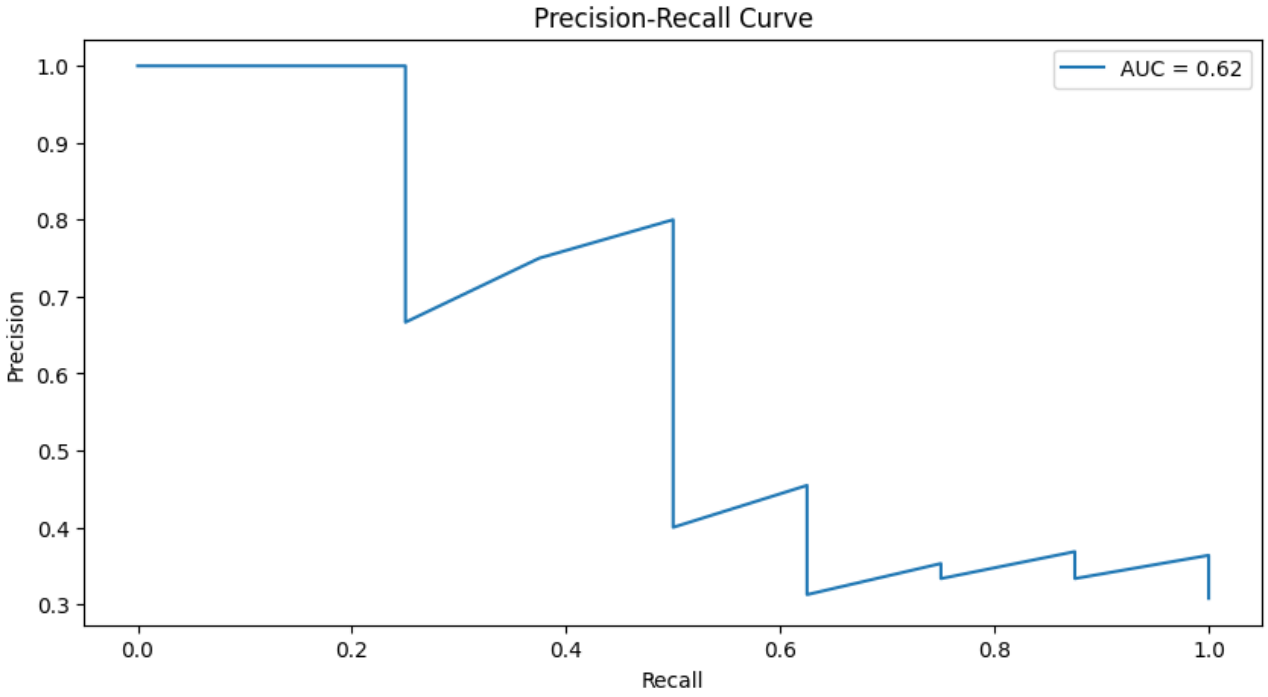
```
                          cv=5)
gsearch.fit(x_train_scaled,y_train)
print("Best Score:",gsearch.best_score_)
print("Best parameters:",gsearch.best_params_)
```

```
 Best Score: 0.6637641723356009
Best parameters: {'C': 0.001, 'penalty': 'l2'}
```

```
   # Ensure y_train and y_test are 1D arrays
y_train = y_train.ravel() if len(y_train.shape) > 1 else y_train
y_test = y_test.ravel() if len(y_test.shape) > 1 else y_test

# Now call evaluate_model with corrected y_train and y_test
logistic_regression = linear_model.LogisticRegression(C=0.001,penalty='l2',
                                                       multi_class='ovr',
                                                       class_weight='balanced')
print("Logistic Regression Model\n")
evaluate_model(logistic_regression, x_train_scaled, y_train,
x_test_scaled, y_test)
```

```
Logistic Regression Model

Training Accuracy: 0.74
Testing Accuracy: 0.69
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.78      0.78        18
           1       0.50      0.50      0.50         8

    accuracy                           0.69        26
   macro avg       0.64      0.64      0.64        26
weighted avg       0.69      0.69      0.69        26

Confusion Matrix:
 [[14  4]
 [ 4  4]]
AUC-ROC: 0.69
```

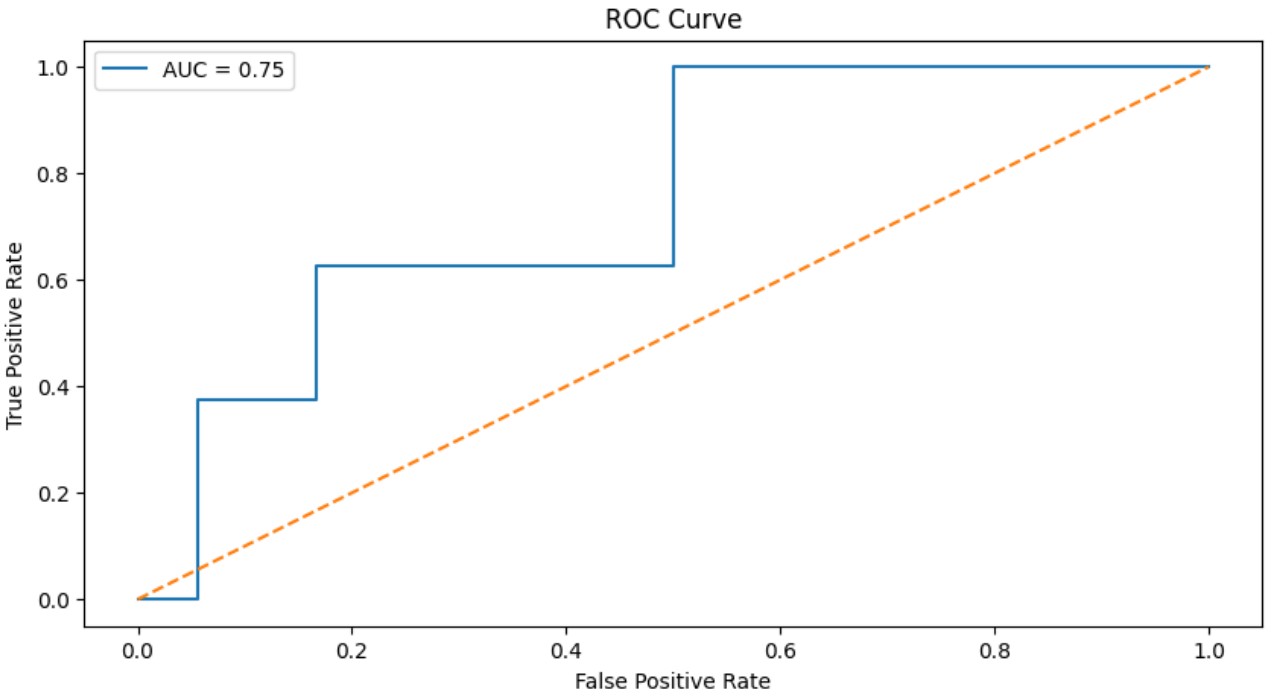

ROC Curve
AUC = 0.69
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

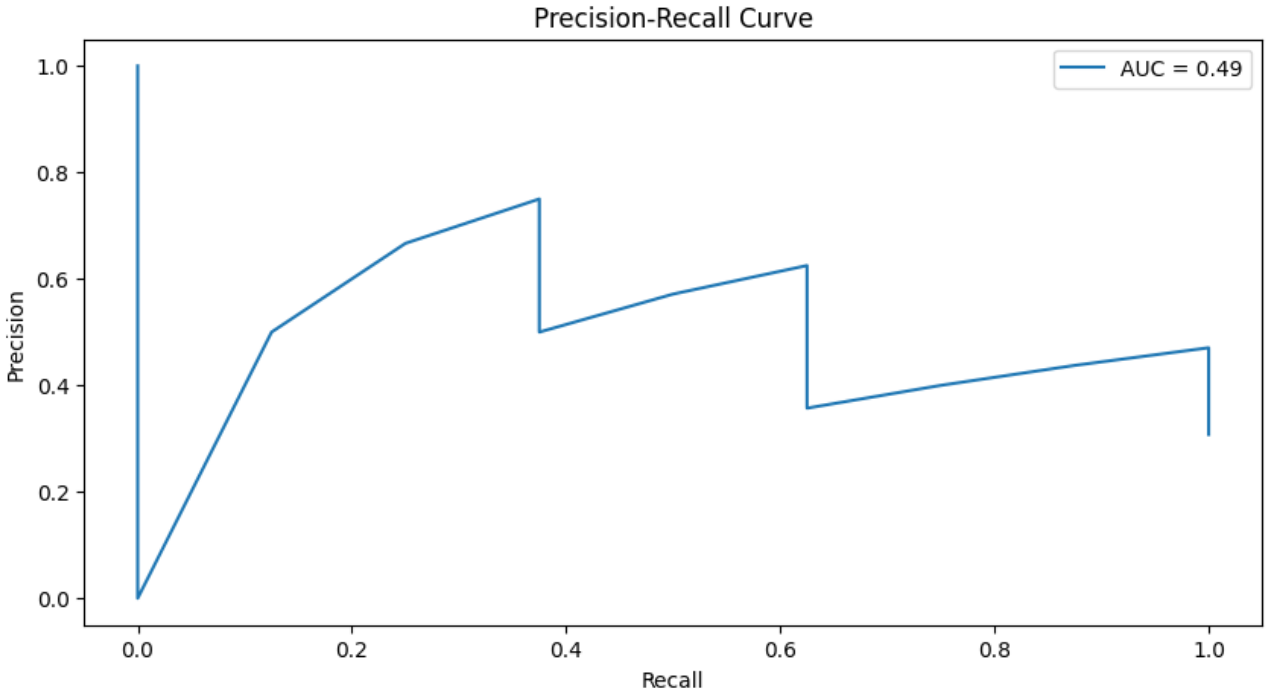

Precision-Recall Curve
AUC = 0.62
Precision
Recall
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

```
   from scikeras.wrappers import KerasClassifier
from tensorflow.keras import models, layers
import tensorflow as tf

# Define the neural network model
def nn_model():
    model = models.Sequential()
    model.add(layers.Dense(500, activation='relu', input_dim=17))
    # Updated to match the input shape
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(500, activation='relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(500, activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(500, activation='relu'))
    model.add(layers.Dropout(0.3))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=[tf.keras.metrics.AUC()])
    return model
```

```
   # Create the KerasClassifier wrapper with class weights
NN = KerasClassifier(model=nn_model, epochs=50, batch_size=200,
verbose=0, class_weight={0:1, 1:12})

# Now you can call the model_fit function (assuming it's defined as
#per your earlier code)
evaluate_model(NN, x_train_scaled, y_train, x_test_scaled, y_test)
```

```
Training Accuracy: 0.74
Testing Accuracy: 0.58
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.39      0.56        18
           1       0.42      1.00      0.59         8

    accuracy                           0.58        26
   macro avg       0.71      0.69      0.58        26
weighted avg       0.82      0.58      0.57        26

Confusion Matrix:
 [[ 7 11]
 [ 0  8]]
AUC-ROC: 0.75
```


ROC Curve
AUC = 0.75
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve
AUC = 0.49
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

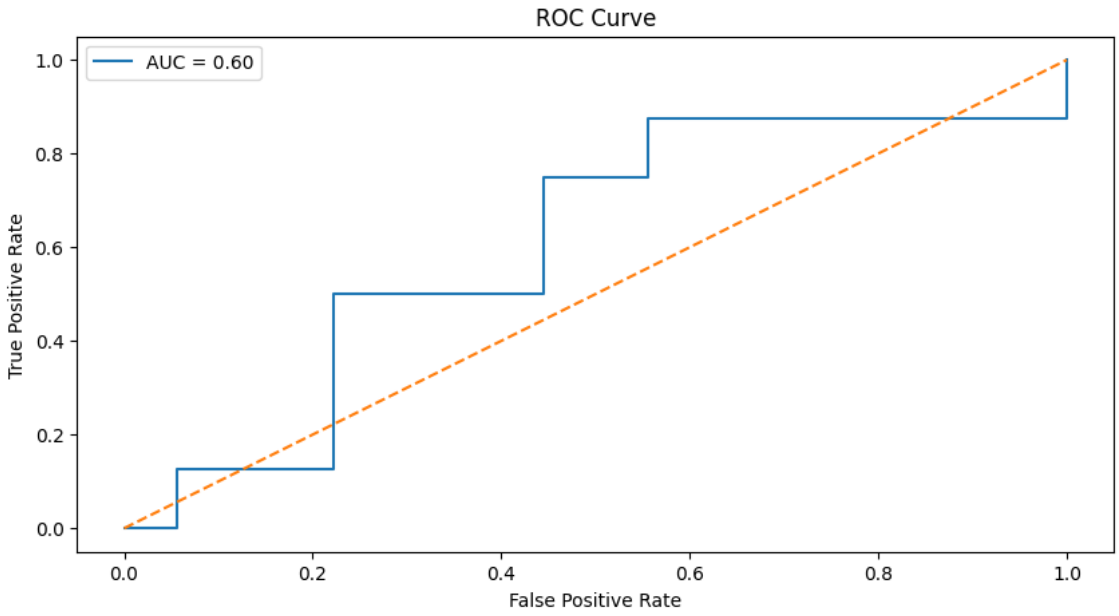
```
   ## ctgan model
features =['NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS',
       'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS',
       'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS',
       'NUMNOTEVENTS', 'admission_type_ELECTIVE', 'admission_type_EMERGENCY',
       'admission_type_URGENT', 'gender_F',
       'gender_M']
x_train = train_ctgan[features]
y_train = train_ctgan['hospital_expire_flag']
x_test = test[features]
y_test = test['hospital_expire_flag']
     scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled=scaler.transform(x_train)
x_test_scaled=scaler.transform(x_test)
param_test ={'C':[0.001, 0.01, 0.1, 1, 10, 100],
             'penalty':('l1','l2')}
lr=linear_model.LogisticRegression(random_state=10,multi_class='ovr',
class_weight='balanced')
gsearch = GridSearchCV(estimator=lr,
                       param_grid=param_test,
                       scoring='roc_auc',
                       n_jobs=-1,
                       cv=5)
gsearch.fit(x_train_scaled,y_train)
print("Best Score:",gsearch.best_score_)
print("Best parameters:",gsearch.best_params_)
```

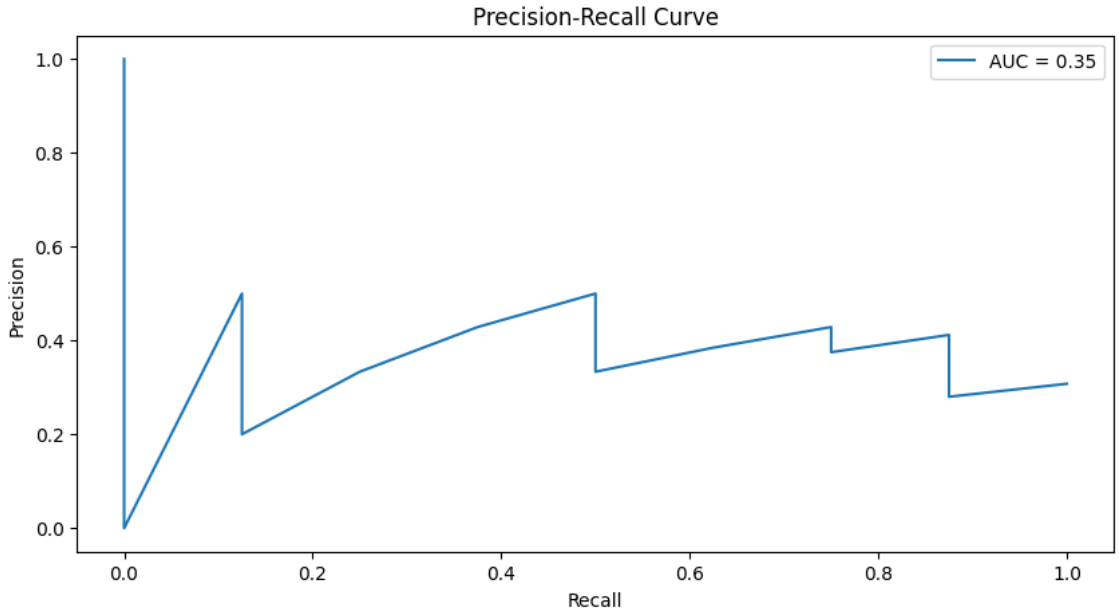
```
   Best Score: 0.6098116169544741
Best parameters: {'C': 0.001, 'penalty': 'l2'}
```

```
   # Ensure y_train and y_test are 1D arrays
y_train = y_train.ravel() if len(y_train.shape) > 1 else y_train
y_test = y_test.ravel() if len(y_test.shape) > 1 else y_test

# Now call evaluate_model with corrected y_train and y_test
logistic_regression = linear_model.LogisticRegression(C=0.001,penalty='l2',
                                                       multi_class='ovr',
                                                       class_weight='balanced')
print("Logistic Regression Model\n")
evaluate_model(logistic_regression, x_train_scaled, y_train,
x_test_scaled, y_test)
```

```
    Logistic Regression Model

Training Accuracy: 0.68
Testing Accuracy: 0.58
Classification Report:
```

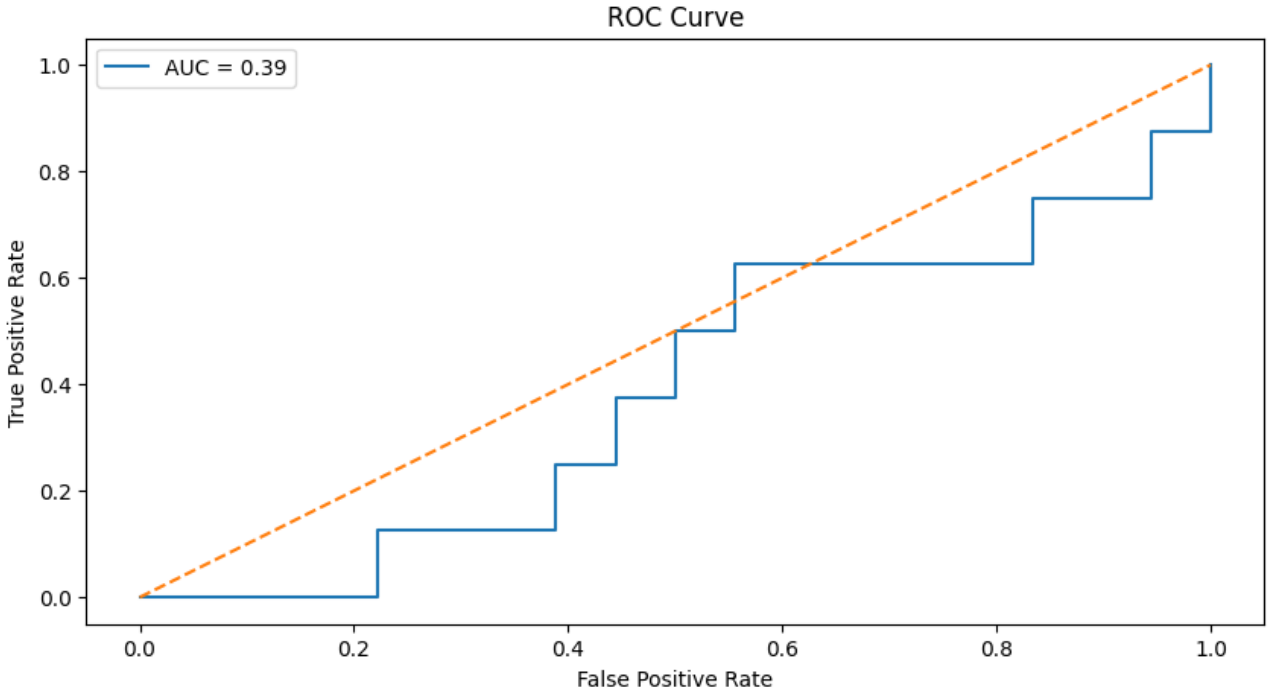
```
              precision    recall  f1-score   support

           0       0.73      0.61      0.67        18
           1       0.36      0.50      0.42         8

    accuracy                           0.58        26
   macro avg       0.55      0.56      0.54        26
weighted avg       0.62      0.58      0.59        26

Confusion Matrix:
 [[11  7]
 [ 4  4]]
AUC-ROC: 0.60
```

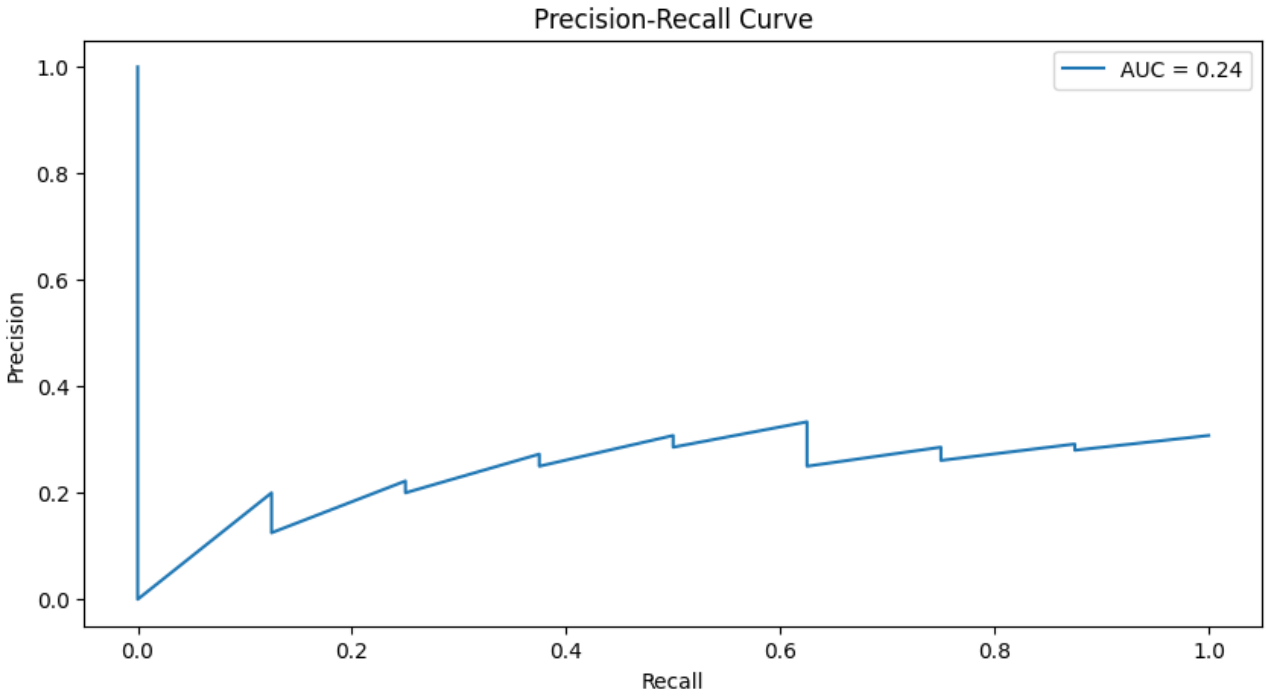




```
evaluate_model(NN, x_train_scaled, y_train, x_test_scaled, y_test)
```

```
   Training Accuracy: 0.74
Testing Accuracy: 0.31
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.17      0.25        18
           1       0.25      0.62      0.36         8

    accuracy                           0.31        26
   macro avg       0.38      0.40      0.30        26
weighted avg       0.42      0.31      0.28        26

Confusion Matrix:
 [[ 3 15]
 [ 3  5]]
AUC-ROC: 0.39
```





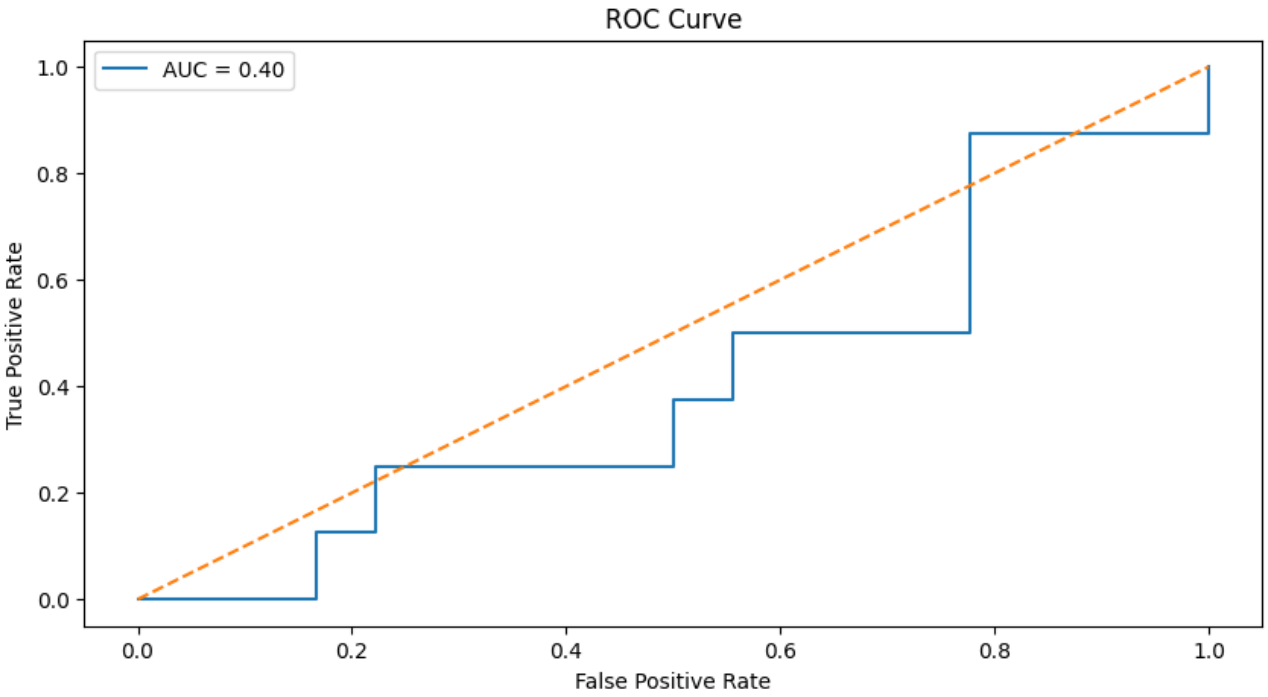

```
    ## tvae model
features =['NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS',
       'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS',
       'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS',
       'NUMNOTEVENTS', 'admission_type_ELECTIVE', 'admission_type_EMERGENCY',
       'admission_type_URGENT', 'gender_F',
       'gender_M']
x_train = train_tvae[features]
y_train = train_tvae['hospital_expire_flag']
x_test = test[features]
y_test = test['hospital_expire_flag']
scaler = StandardScaler()
scaler.fit(x_train)
x_train_scaled=scaler.transform(x_train)
x_test_scaled=scaler.transform(x_test)
param_test ={'C':[0.001, 0.01, 0.1, 1, 10, 100],
             'penalty':('l1','l2')}
lr=linear_model.LogisticRegression(random_state=10,multi_class='ovr',
class_weight='balanced')
gsearch = GridSearchCV(estimator=lr,
                       param_grid=param_test,
                       scoring='roc_auc',
                       n_jobs=-1,
                       cv=5)
gsearch.fit(x_train_scaled,y_train)
print("Best Score:",gsearch.best_score_)
print("Best parameters:",gsearch.best_params_)
```

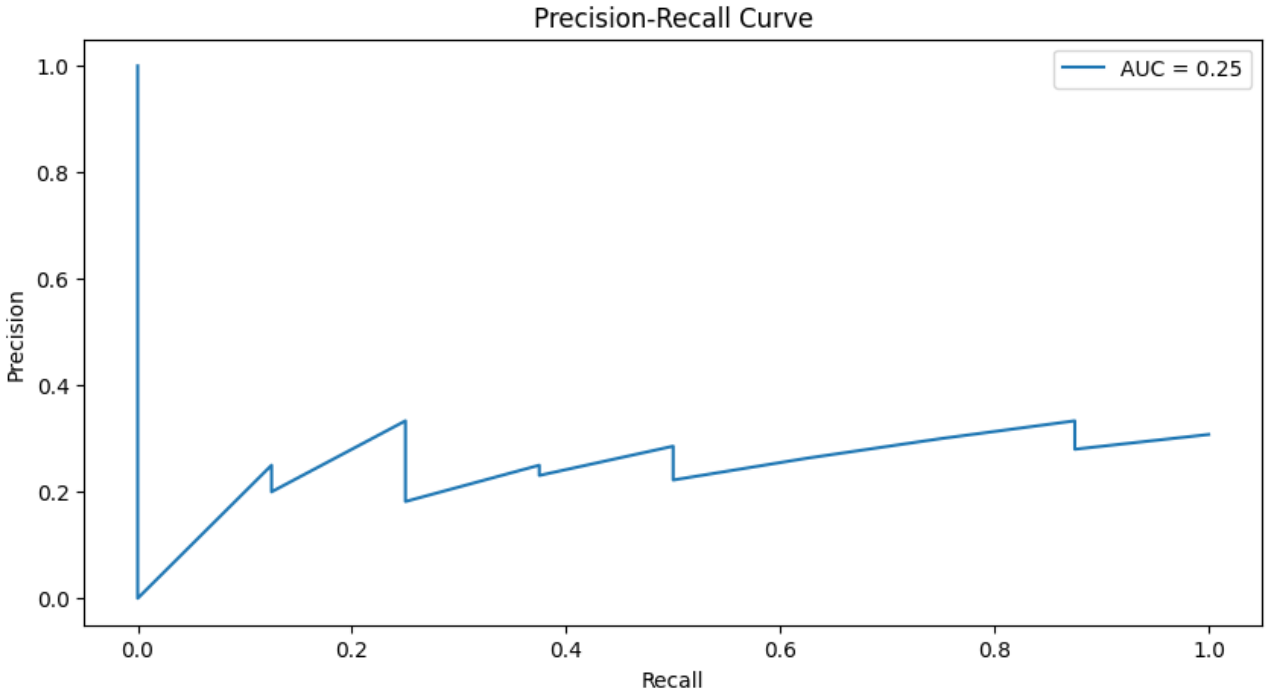
```
    Best Score: 0.5814814814814815
```

```
Best parameters: {'C': 1, 'penalty': 'l2'}
```

```
    # Ensure y_train and y_test are 1D arrays
y_train = y_train.ravel() if len(y_train.shape) > 1 else y_train
y_test = y_test.ravel() if len(y_test.shape) > 1 else y_test

# Now call evaluate_model with corrected y_train and y_test
logistic_regression = linear_model.LogisticRegression(C=1,penalty='l2',
                                                       multi_class='ovr',
                                                       class_weight='balanced')
print("Logistic Regression Model\n")
evaluate_model(logistic_regression, x_train_scaled, y_train,
x_test_scaled, y_test)
```

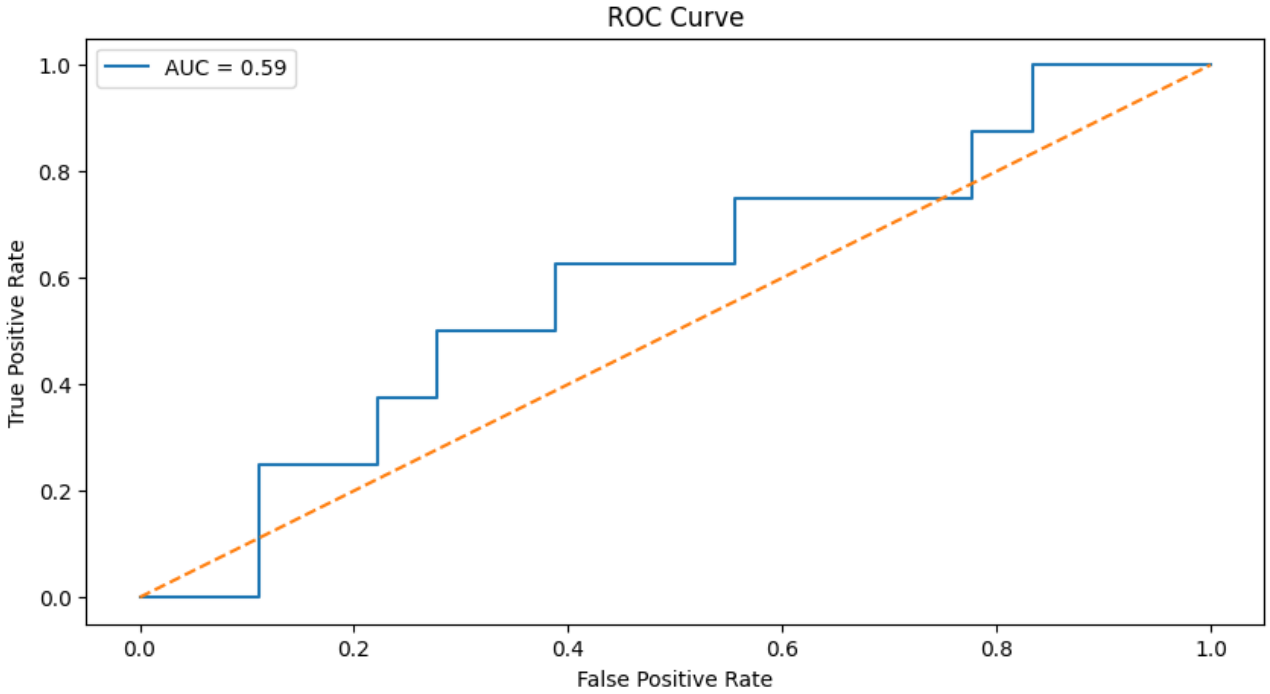
```
   Logistic Regression Model

Training Accuracy: 0.83
Testing Accuracy: 0.46
Classification Report:
               precision    recall  f1-score   support

           0       0.67      0.44      0.53        18
           1       0.29      0.50      0.36         8

    accuracy                           0.46        26
   macro avg       0.48      0.47      0.45        26
weighted avg       0.55      0.46      0.48        26

Confusion Matrix:
 [[ 8 10]
 [ 4  4]]
AUC-ROC: 0.40
```

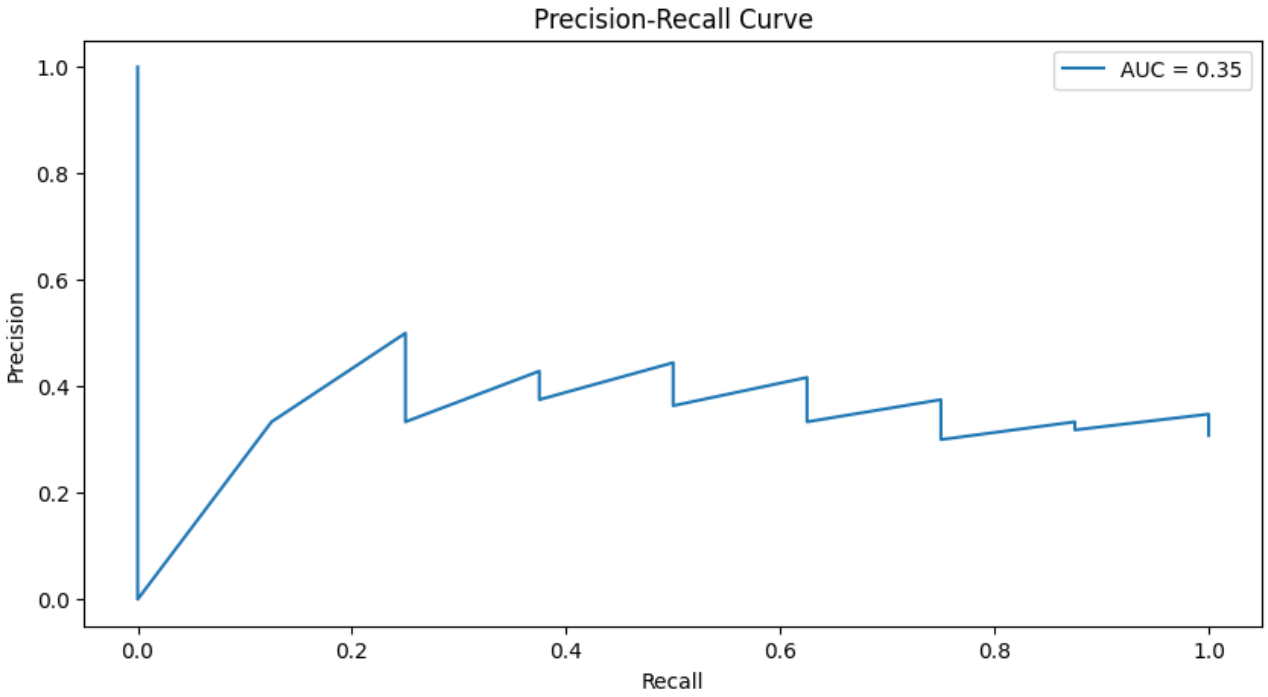

ROC Curve
AUC = 0.40
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve
AUC = 0.25
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

```
evaluate_model(NN, x_train_scaled, y_train, x_test_scaled, y_test)
```

```
    Training Accuracy: 0.96
Testing Accuracy: 0.62
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.89      0.76        18
           1       0.00      0.00      0.00         8

    accuracy                           0.62        26
   macro avg       0.33      0.44      0.38        26
weighted avg       0.46      0.62      0.53        26

Confusion Matrix:
 [[16  2]
 [ 8  0]]
AUC-ROC: 0.59
```





Conclusion

In the mortality prediction use case, CTGAN and TVAE were compared as synthetic data generation techniques to balance data utility and privacy. CTGAN generally performed better in capturing the structural properties of the original dataset, showing closer alignment with the patterns and distributions observed in the actual data. This alignment suggests that CTGAN is more effective for generating realistic synthetic data that preserves the relationships among features, which is essential for accurate mortality prediction models. On the other hand, TVAE, while generating diverse data points, displayed slightly more dispersion in capturing the original data's structure, which may make it less suitable when high fidelity to the original dataset is required. However, TVAE showed a higher Privacy at Risk (PaR) score, indicating better privacy protection as its synthetic data points were further from the original data points on average, thus reducing the risk of re-identification. In summary, CTGAN provides better utility for predictive modeling, whereas TVAE offers stronger privacy guarantees, making it a preferable choice for applications where privacy is prioritized over data utility.

3.3 Synthetic Data Generation

Synthetic data generation stands at the forefront of techniques aimed at reducing the risk of re-identifying individuals in datasets. Previously, we observed that while data anonymization can be effective, it often comes at the cost of reduced utility. Differential privacy introduces small amounts of noise to the data but can significantly impair model compatibility. Synthetic data offers a compelling alternative, it can be tailored to enhance privacy without sacrificing the usefulness of the data or exposing individual data points. Sensitive personal information is essentially prevented from being disclosed because synthetic data doesn't match actual entities. Such events are entirely coincidental, giving those persons plausible deniability, even if the synthetic data happens to resemble real people from the original dataset.

part by Bibek

A synthetic dataset is essentially a collection of data generated through programmed algorithms. This data can take various forms:

- They may include binary indicators, categorical variables, or numerical values, regardless

of whether they are ordinal (having a natural order) or non-ordinal.

- The number of features (variables) and the size of the dataset are both arbitrary and can be altered to meet specific needs.
- The degree of separation between classes can be altered when used to classification algorithms. The learning problem's degree of difficulty can be changed thanks to this control.
- Complex and non-linear generative methods can be used to generate the data for regression issues.
- The data should ideally be random so that consumers can choose from a variety of statistical distributions. This implies that it is possible to carefully control and fine-tune the underlying random processes.
- It is possible to imitate real-world data defects or further improve privacy by carefully adding random noise.

We used Google collab, so we required Tensorflow python package. To install latest version of Tensorflow

```
pip install tensorflow
```

3.4 Scalability Tests

statistical similarity analysis between the original dataset and synthetic datasets will provide a measure of the strength of replication of statistical properties, patterns, and relationships present in the original data by synthetic data. The intention is to make sure the synthetic data is a close approximation to real data in order not to disclose sensitive information directly. In evaluating the statistical similarity between the original and synthetic datasets, we employed several methodologies, each providing insights into different aspects of data similarity.

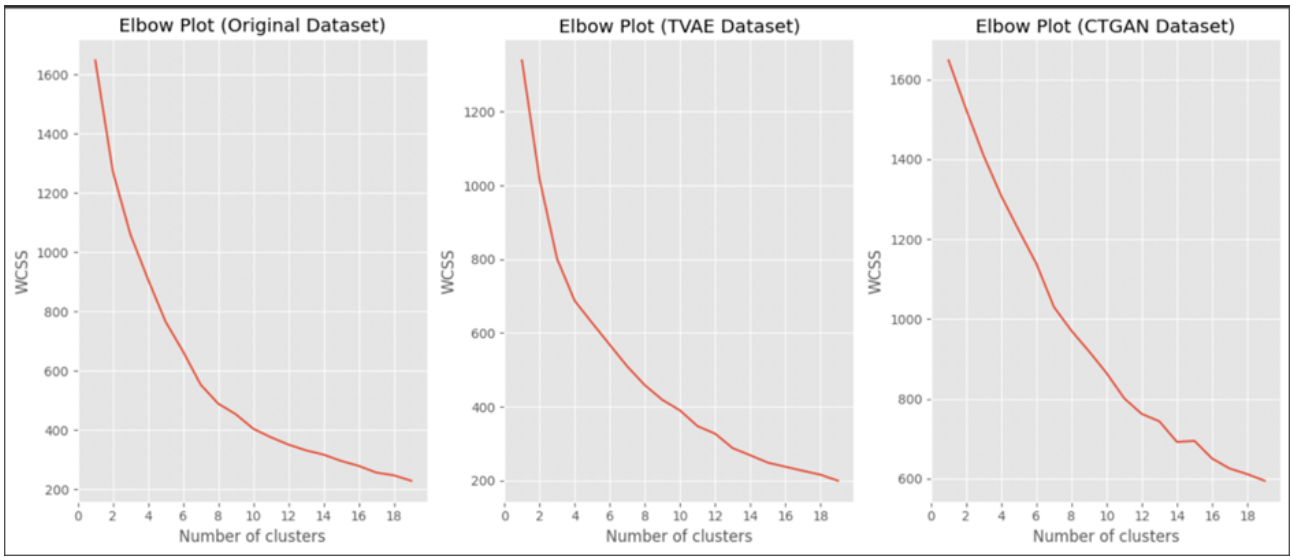
part by Pramod

3.4.1 K-Means Clustering

K-means clustering is a method that comes from signal processing and uses the k-means algorithm. It aims to group a set of n observations into k clusters. Each observation is placed in the cluster where the average (or center) of the cluster is closest to it, making that cluster its representative. [Vidhya, 2019] The K-means clustering was applied to verify the structural similarity through the analysis of cluster patterns in view of various counts of clusters that allowed us to find whether the synthetic datasets reproduced the natural groupings found in the original data. Comparing Descriptive Statistics: Basic summary statistics of different features were compared like mean, standard deviation, and skewness, which described that synthetic data maintained the central tendencies and variability of the original dataset.

K-means clustering was performed and accompanied with the Elbow Plot analysis on the original and synthesized data generated through TVAE and CTGAN for structural similarity between these sets of data. This helps to identify how well the synthetic datasets retain similar clustering characteristics compared to the original data. First, the datasets were preprocessed by selecting relevant columns from both the original and synthetic datasets. Further, each individual dataset was standardized using the StandardScaler to make sure the features had a mean of zero and a standard deviation of one. This scaling is necessary because K-means clustering is sensitive to the scale of a dataset. Features with large numerical ranges may end up dominating in the clustering process, thus distorting the result. Then K-means clustering was

performed on each data set for a number of clusters from 1 to 19. For each number of clusters, the Within-Cluster Sum of Squares - also called inertia - was calculated. This represents how compact the clusters are. WCSS is defined as a sum of squared distances between each point in a cluster and the centroid of the cluster. The lower the value of WCSS, the denser and hence more coherent the cluster. To do this, we recorded the WCSS for every dataset over a range of values of Cluster Count to create WCSS sequences for each of the original, TVAE, and CTGAN datasets. Next, we plot the WCSS values against the number of clusters in an Elbow Plot. In such a plot, an "elbow"-a point beyond which the rate of WCSS reduction radically slows-indicates an optimal number of clusters. The same position of elbows and overall patterns in the WCSS values between original and synthetic data would imply that synthetic data retains the clustering structure present in the original data. It provides a high-level comparison of the structural organization for each dataset, showing how this technique informs us whether the synthetic data conveys the tendencies of groupings and spatial distributions seen in the original dataset.



Therefore, the clustering features of TVAE and CTGAN synthetic datasets look similar to that of the original dataset as evidenced from the shape and locations of the elbow points in the Elbow Plots. Such similarity in clustering pattern suggests that both the synthetic datasets capture the structural properties of the original data well and hence can act as good surrogates for the original dataset when applications depend on clustering structures.

3.4.2 Kolmogorov-Smirnov (KS) Test

The Kolmogorov-Smirnov Goodness of Fit Test (K-S test) compares your data with a known distribution and lets you know if they have the same distribution. Although the test is non-parametric, it doesn't assume any particular underlying distribution â it is commonly used as a test for normality to see if your data is normally distributed.[Glen, 2023]

The Kolmogorov-Smirnov (KS) Test was applied to continuous features to statistically compare their distributions, with a low KS statistic indicating a close match in distributions between the original and synthetic datasets. The KS test was performed on columnwise testing of similarity between the original dataset and two synthetic datasets, one from CTGAN and the other from TVAE. In principle, this test provides a statistical measure of similarity by comparing the CDFs of each feature within the two datasets, namely, the original and synthetic ones. For a given characteristic, each KS test computes a KS statistic-a statistic of maximum difference

between CDFs of the original and synthetic data-along with the p-value indicating whether this difference is statistically significant.

It can be noticed that, for each feature in the datasets, a loop has been done in the code above, where the KS test is applied in order to compute the statistic and p-value computation for each of them. Using prints in order to display the result, we can notice how much the distribution of each feature in the original and synthetic dataset is similar or deviates from each other. This function was applied on the CTGAN and TVAE datasets separately, thus allowing for the comparison of each synthetic dataset to the original data in detail.

```
CTGAN Dataset - KS Test Results
feature_0: KS Statistic = 0.4078, p-value = 0.0000
feature_1: KS Statistic = 0.5825, p-value = 0.0000
feature_2: KS Statistic = 0.1359, p-value = 0.2983
feature_3: KS Statistic = 0.3883, p-value = 0.0000
feature_4: KS Statistic = 0.1068, p-value = 0.6020
feature_5: KS Statistic = 0.2913, p-value = 0.0003
feature_6: KS Statistic = 0.1456, p-value = 0.2254
feature_7: KS Statistic = 0.2039, p-value = 0.0274
feature_8: KS Statistic = 0.1942, p-value = 0.0409
feature_9: KS Statistic = 0.5049, p-value = 0.0000
feature_10: KS Statistic = 0.3398, p-value = 0.0000
feature_11: KS Statistic = 0.0000, p-value = 1.0000
feature_12: KS Statistic = 0.9029, p-value = 0.0000
feature_13: KS Statistic = 0.9029, p-value = 0.0000
feature_14: KS Statistic = 0.9417, p-value = 0.0000
feature_15: KS Statistic = 0.5146, p-value = 0.0000
feature_16: KS Statistic = 0.5146, p-value = 0.0000
```

This output of the KS test thus is rather varied. Some features in particular have a high KS statistic with a very low p-value. This indeed shows that the distribution difference for those features is statistically significant. For example, in the CTGAN dataset, features like feature_0, feature_1, and feature_3 show big differences from the original data, as reflected by their high KS statistics and low p-values. By contrast, some features have very high p-values, such as Feature 11 for both datasets, indicating that these features do not show any significant difference in distribution between the original and synthetic datasets. These results suggest that some features are well-represented in the synthetic data, while others differ significantly in their distributions. The KS test is an informative one, but it has numerous limitations when applied within this context. Because the KS test is sensitive to sample size, for large datasets, even small differences between distributions can lead to very significant p-values. On top of that, the KS test considers one feature at a time without any indication of dependencies or correlations between features either, which might also be an important clue toward its multivariate structure. Besides this, the KS test is mainly designed for continuous data; hence, it may not be appropriate for categorical or binary features. Other tests, such as chi-square tests, exist for such data types. In general, the KS test provides a helpful feature-by-feature analysis of distributional similarity between the original and the synthetic datasets. It turns out that,

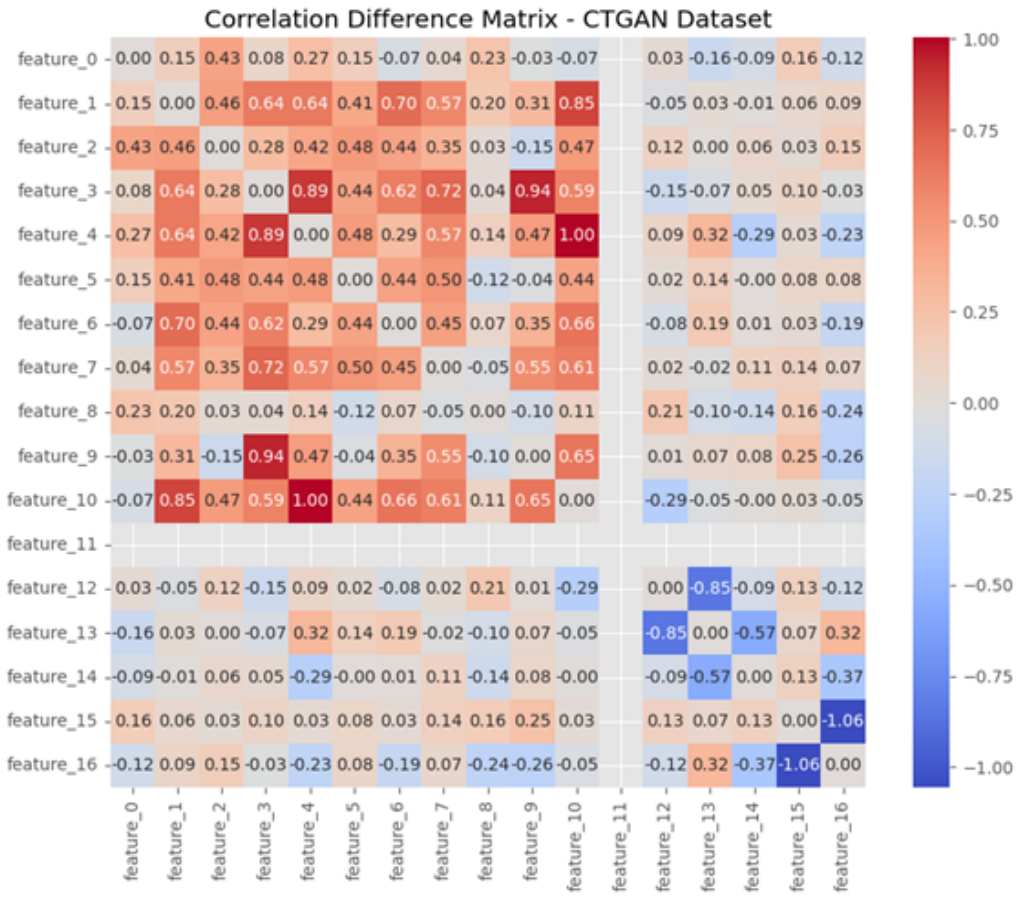
```
TVAE Dataset - KS Test Results
feature_0: KS Statistic = 0.5049, p-value = 0.0000
feature_1: KS Statistic = 0.2427, p-value = 0.0045
feature_2: KS Statistic = 0.1068, p-value = 0.6020
feature_3: KS Statistic = 0.1942, p-value = 0.0409
feature_4: KS Statistic = 0.1942, p-value = 0.0409
feature_5: KS Statistic = 0.4369, p-value = 0.0000
feature_6: KS Statistic = 0.2816, p-value = 0.0005
feature_7: KS Statistic = 0.2136, p-value = 0.0180
feature_8: KS Statistic = 0.2718, p-value = 0.0009
feature_9: KS Statistic = 0.6019, p-value = 0.0000
feature_10: KS Statistic = 0.2718, p-value = 0.0009
feature_11: KS Statistic = 0.0000, p-value = 1.0000
feature_12: KS Statistic = 0.9417, p-value = 0.0000
feature_13: KS Statistic = 0.9223, p-value = 0.0000
feature_14: KS Statistic = 0.9806, p-value = 0.0000
feature_15: KS Statistic = 0.5146, p-value = 0.0000
feature_16: KS Statistic = 0.5146, p-value = 0.0000
```

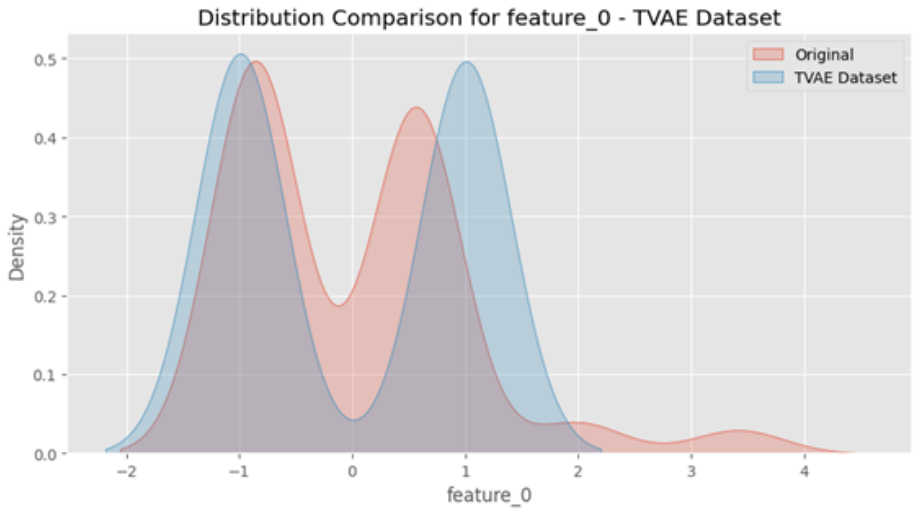
according to the following results, there are, in fact, a number of features in both the CTGAN and TVAE datasets whose distributions are significantly different from the original data, which thus suggests that the synthesized datasets may fail to capture the distributional characteristics of all features. However, assessing similarity with a KS test alone is narrow because it focuses only on individual feature distributions without accounting for feature relationships or multivariate structure. As such, although informative, the KS test should be supplemented by other analyses, such as comparisons of correlations or visualization techniques, in order to obtain a full picture regarding the fidelity of the synthetic data to the original dataset.

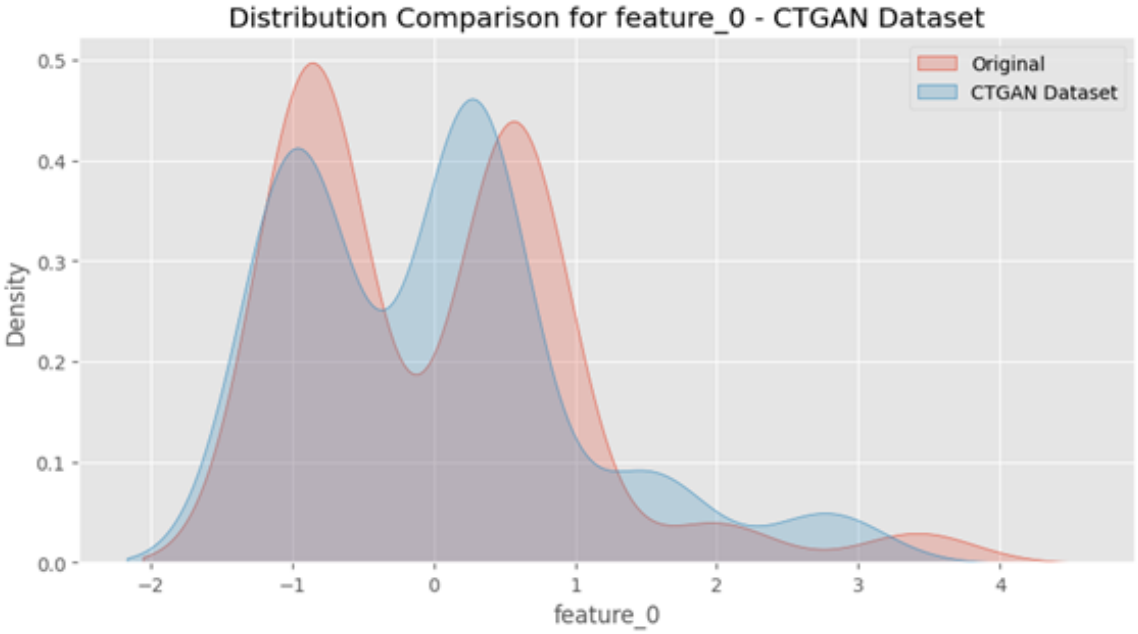
3.4.3 Statistical Analysis

For the comparison of the statistical similarities between the original dataset and the synthetic dataset created by CTGAN and TVAE, we used Descriptive Statistics Comparison, Distribution Plots, and Pairwise Correlation Comparison. Each of these methods provides a different aspect of similarity through which we can evaluate whether the synthetic data is effectively able to replicate the characteristics in the original dataset. Descriptive statistics comparison involves the investigation of some of the basic statistics-mean, median, standard deviation, and skewness-for each feature in the two sets of data. By comparing means, medians, and standard deviations between the original and synthetic data sets, we can verify if the synthetic data carries the central tendencies, variability, and shape regarding the distribution of features. Similar statistics of the synthetic data give evidence that the key statistical properties of the source dataset have been well captured, while large gaps denote deviations which may influence the usefulness of the synthetic data in question. Distribution Plots provide a clear graphical overview of the feature distributions. We have plotted Kernel Density Estimates (KDE) for selected features to visually compare the distribution of each feature present in the original and synthetic datasets. This allows us to visually inspect the shape, peaks, and spread of each

distribution, which helps us understand how well the synthetic datasets replicate the actual patterns in the data. From the comparison plot for feature_0 in the CTGAN dataset with feature_2 in the TVAE dataset, the similarities and differences are so striking that we are able to see precisely which features each synthetic dataset captures best. To verify whether the relations among the features were preserved, the Pairwise Correlation Comparison was done. We computed the two synthetic datasets' correlation matrices and plotted the Correlation Difference Matrix w.r.t. the original dataset. This matrix highlights the differences of the pairwise correlations: warmer colors reflect positive differences, and cooler colors reflect negative differences. A close match in the correlation structures shows that dependence and any relationship between the features are retained, which is important in many real applications that count on these relationships. Conclusion Overall, these methods allowed us to conduct a comprehensive analysis of similarity. Descriptive statistics and distribution plots enabled us to check the distributions of individual features, while pairwise correlation comparison told us about the retention of feature relationships. Whereas for some features in synthetic datasets, correspondence with the original data was close, for other features, there were discrepancies either in distribution or in correlation. Taken together, these approaches apply for the assessment of similarity of the original and synthetic data, but their capability to capture complex structures and dependencies of the data might be extended by other analyses such as multivariate tests.




Distribution Comparison for feature_0 - TVAE Dataset
Original
TVAE Dataset
Density
feature_0
0.0
0.1
0.2
0.3
0.4
0.5
−2
−1
0
1
2
3
4


Distribution Comparison for feature_0 - CTGAN Dataset
Original
CTGAN Dataset
Density
feature_0
0.0
0.1
0.2
0.3
0.4
0.5
−2
−1
0
1
2
3
4

3.4.4 Autoencoder

The autoencoder similarity test shall be used to ascertain the strength of capture of the underlying structure in the original dataset by the synthetic datasets CTGAN and TVAE. An autoencoder is a neural network architecture that learns through an encoder-decoder process how to embed data in a compressed form. Here, an autoencoder is considered to learn the most informative patterns in the real data and later use the same trained encoder on synthetic datasets. By comparing these embeddings, we can identify the structural similarity between the synthetic and source datasets in a compact low-dimensional space.

Steps

- The Autoencoder Architecture: An autoencoder was defined, consisting of an encoder that compresses the input data to a lower-dimensional representation with 2D embedding for visualization and a decoder that reconstructs the data back to its original dimensions. The bottleneck structure in the encoder captures the most important patterns in data with noise reduction and redundancy.

- Training the Autoencoder: The autoencoder was trained on the original dataset alone. In training it on original data alone, the encoder learns to embed that captures true underlying structure of the original dataset. This step is important for measuring how well synthetic datasets align to the latent patterns of original data.

- Embeddings Extraction: For both the original dataset and each synthetic dataset, we utilized the encoder in extracting embeddings after training. Each dataset will now have an extracted embedding within this same reduced dimensional space and hence is comparable w.r.t. their structural properties.

- Cosine Similarity Calculation: To give a quantitative estimate of the similarity between the original and synthetic embeddings, we computed the cosine similarity between the original and CTGAN embeddings and between the original and TVAE embeddings. The cosine similarity score ranges between 0 and 1; a score of 1 means identical orientation in the latent space, while 0 indicates no alignment. This metric gives the direct comparison of how similar in structure the datasets are.

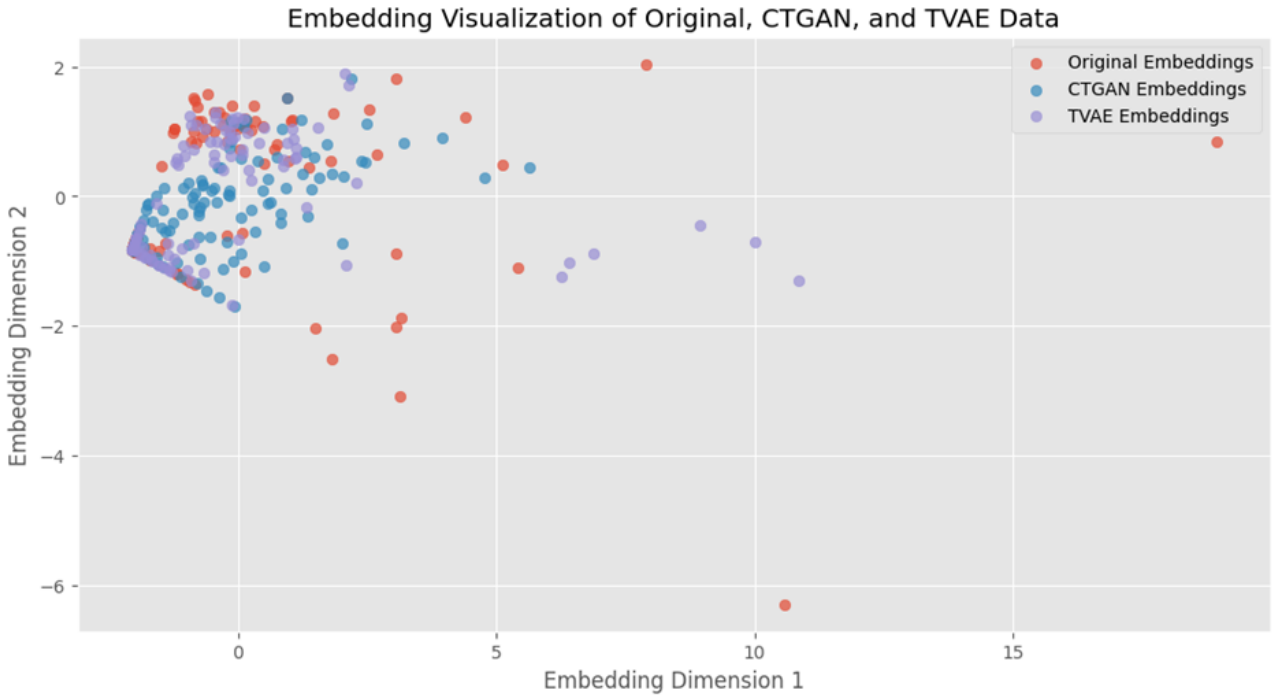
- Visualization: Finally, we projected the embeddings into 2D using PCA and plotted the original, CTGAN, and TVAE embeddings on the same graph for visualization purposes. This chart allows us to visualize the amount of overlap and alignment from the three datasets and further corroborates the cosine similarity scores above.

Why This Methodology? Because of the complex pattern capturing dependency in a compact form, an autoencoder methodology works best for comparing datasets. Other approaches, like feature-by-feature comparison, consider the overall structure and dependencies within the data. Focusing on a latent representation would therefore provide insight into whether the synthetic datasets preserve the high-level structure of the original data. This is especially helpful when the data has intricate dependencies or when comparisons of the distribution of individual features may miss more subtle similarities in the overall structure of the data.

3.4.5 Conclusion

The results are that the Cosine Similarity Score between the original and CTGAN embeddings is 0.7131, while for the TVAE embeddings, it is 0.6081. This reflects that, in the latent space,

CTGAN is more aligned to the original dataset than TVAE. Moreover, as seen from the embedding visualization, the CTGAN embedding is more closely clustered with the embeddings of the original dataset, while the TVAE embedding demonstrates a more dispersed pattern, which points at a weaker alignment. In general, from both cosine similarity scores and embedding visualization, one can tell that CTGAN captures structural features of the original dataset more adequately than TVAE. This method serves as a dependable answer to our question about similarity because it not only considers individual feature relationships but also holistic structure within data, confirming CTGAN as superior in terms of preservation of the underlying patterns of the original dataset.



3.5 Privacy Risk Module

With the increased cases of data breaches, healthcare organizations such as Optum and United Healthcare are at an elevated risk due to vast data on PII kept on patients. This has presented a major challenge, especially for those analysts who require the data for research, yet it takes considerable time to go through the access and authorization policies put in place for data protection. Synthetic data could be a future direction given that the data represents real data sets but actual PII is not exposed. It is required to check the privacy risks of synthetic data and other alternatives and compare those alternatives so businesses understand the implications of each with regards to privacy, while making informed choices

part by Pramod

Objectiv

Hurdles in data access and authorization have made the privacy risk the main barrier to productivity. In this regard, it is very important to define privacy risk in quantitative terms and understand its implications. However, privacy risk can be measured with a large amount of complexity and has specific challenges in various already existing and emerging techniques of measurement. Generally speaking, reliable metrics of privacy risk need to be developed for assurance that data access solutions like synthetic data balance utility with privacy. Hurdles in data access and authorization have made the privacy risk the main barrier to productivity.

In this regard, it is very important to define privacy risk in quantitative terms and understand its implications. However, privacy risk can be measured with a large amount of complexity and has specific challenges in various already existing and emerging techniques of measurement. Generally speaking, reliable metrics of privacy risk need to be developed for assurance that data access solutions like synthetic data balance utility with privacy.

Techniques for mitigating privacy risk: Current state and future direction

Removing identifier columns: The simplest method to reduce privacy risk is to remove the identifier columns. Identifiers, such as SSNs or Patient IDs, are unique and can link a dataset with external data sources that may unmask PII. This simple step reduces a great deal of the privacy risk because these identifiers are usually not needed in analyses. That is a good start but raises an obvious question: besides those most obvious identifiers, what other columns can be dropped to further reduce privacy risk? Currently, no standardized metric can be looked upon to decide which columns contribute most towards data sensitivity.

Another technique reduces the number of rows in a dataset. While this may decrease some risk of exposure, no quantitative measure can be assessed for the amount of privacy risk eliminated because of the removal of rows. Furthermore, random removal of rows may also destroy the utility of a dataset and affect its uses in the analysis. Every retained data point is theoretically fully exposed, and thus vulnerable to all manners of misuse, particularly when joined with external datasets.

Synthetic data generation is one methodology which may provide a trade-off for maintaining dataset utility with possible privacy protection. The idea is to generate synthetic data such that it more closely resembles the original data. However, there is a problem due to the generation of synthetic data being very similar to the actual data. Probably some real data points get replicated, thus failing in protection regarding privacy. There is a need to quantify the privacy risk in these cases so that balancing between data utility and privacy can be performed.

Measuring privacy is needed. Synthetic data generation has become one of the most popular solutions to ensure privacy; however, the art of developing such algorithms is still in its early days of development and study. A clear, understandable metric is missing that would be able to quantify the amount of privacy achieved in synthetic datasets. What's really needed is a metric that ensures synthetic data can't be easily linked to the original data points, protecting PII even if there's a malicious actor trying to match synthetic data against external sources. Such a metric would ideally quantify how well synthetic data confuses any such attempt at external re-identification.

Privacy Risk Matrics

Our approach would centre on defining and establishing a new metric, called Privacy At Risk, to effectively determine the privacy risk in synthetic medical datasets. PaR will allow Optum to systematically assess the privacy risks associated with synthetic data by providing a clear metric with which to assess privacy across scenarios. It may inform decisions around data-sharing practices in the development of more rapid and secure access by analysts at Optum, with the protection against possible breaches in privacy. The paper will explain in detail how PaR will be developed and applied, indicating that it provides a differential, effective, quantified measure of privacy risk able to guide Optum with confidence on the adoption of synthetic data solutions.

Below is a procedure for PaR calculation using the Nearest Neighbor Distance approach for quantifying privacy risk in synthetic data. The intuition of the method described herein is to estimate the similarity of synthetic data points to original data points by calculating the mean distance between every synthetic data point and its closest original data point. It assumes that synthetic data points, which are too close to original data points, could result in a higher

chance of privacy leakage since an attacker might link synthetic and original records to infer sensitive information.

3.5.1 K-Nearest Neighbor Distance

- Data Preparation: Both the original and synthetic datasets will be standardized with StandardScaler, which scales features to have approximately zero mean and unit variance. This would ensure that all of the features are on the same scale, which is crucial for methods based on distances, since this difference in scale could distort the distance measurements and therefore provide erroneous results. Calculation of
- Nearest Neighbor: For each synthetic data point, the nearest original data point is computed by the NearestNeighbors algorithm by the use of the Euclidean distance. In this way, nearest neighbor distances for all synthetic points are calculated, which gives some measure of closeness of each synthetic to an actual record. The average of these nearest neighbor distances is calculated to represent the Privacy at Risk score. A lower mean distance reflects that on average, synthetic data points are close to original data points and hence reflects a higher risk in terms of privacy because the synthetic data closely resembles the original. A higher mean distance infers that synthetic and original data are further apart, hence a lower risk in terms of privacy.
- Comparison: A comparison of the PaR scores for the CTGAN and TVAE datasets is done. The dataset with the higher average distance (PaR) has more protection in terms of privacy because its synthetic data points remain far from original data points due to lesser chances of re-identification.

Merits and Demerits of the Method

This nearest neighbor-based approach to measuring privacy has a number of very desirable properties: it leads to an intuitive, simple metric in its own right, directly quantifying the similarity of synthetic and original data points. For at least moderately sized datasets, the computational efficiency is very good, and the technique applies to any form of numerical data. It also allows easy comparisons across multiple synthetic datasets due to the single numeric value representing privacy risk.

Yet this approach has several limitations: for every synthetic record, it considers only the nearest original point, which can miss crucial relationships in high-dimensional data. The approach assumes that greater distance always implies greater privacy, which is not a nuanced view of privacy risk alone, especially when the structural nature of the distribution of synthetic data differs from that of the original data. This approach is also sensitive to the choice of a distance metric, which may not be appropriate for every dataset or every kind of data.

```
    ## Privacy Risk Using Nearest Neighbor Distance
import numpy as np
import pandas as pd
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
original_scaled = scaler.fit_transform(mortality_pred)
ctgan_scaled = scaler.transform(train_ctgan)
tvae_scaled = scaler.transform(train_tvae)
def calculate_privacy_risk(original, synthetic, label):
    # Use NearestNeighbors to find the closest original point
```

```
    #for each synthetic point
    nbrs = NearestNeighbors(n_neighbors=1, algorithm='ball_tree').fit(original)
    distances, _ = nbrs.kneighbors(synthetic)

    # Calculate mean distance (Privacy at Risk)
    mean_distance = np.mean(distances)
    print(f"Privacy Risk (PaR) for {label}: {mean_distance:.4f}")
    return mean_distance

# Calculate privacy risk for CTGAN and TVAE synthetic datasets
privacy_risk_ctgan = calculate_privacy_risk(original_scaled,
ctgan_scaled, "CTGAN")
privacy_risk_tvae = calculate_privacy_risk(original_scaled,
tvae_scaled, "TVAE")

# Determine which synthetic dataset has a lower privacy risk
if privacy_risk_ctgan > privacy_risk_tvae:
    print("TVAE has greater privacy protection based on PaR
    (higher mean distance).")
else:
    print("CTGAN has greater privacy protection based on PaR
    (higher mean distance).")
```

```
    Privacy Risk (PaR) for CTGAN: 3.5287
Privacy Risk (PaR) for TVAE: 1.0139
TVAE has greater privacy protection based on PaR (higher mean distance).
```

Final Conclusion

This PaR method based on the nearest neighbor gives a realistic tool to the practitioner who wants to explore the privacy risk of synthetic data by measuring how similar synthetic data points are to original data points. The method, thus concluded that the TVAE synthetic data was more privacy-protective than CTGAN synthetic data based on the fact that the average distance between TVAE points and original data points was higher. This highly valued method is very useful for quick and interpretable privacy assessments; however, it is best applied in conjunction with other privacy metrics, such as Membership Inference Attacks or checks for k-Anonymity, for a far more comprehensive assessment.

3.5.2 Distance Density

The method of Distance Density presented here is used to assess the privacy risk of the synthetic datasets, here CTGAN and TVAE, by evaluating the closeness between the synthetic and original data. This is based on the idea of calculating the density of distances between points in synthetic data and their nearest neighbors in the original dataset. That would mean if synthetic points are closely packed, lying near the original points, their synthetic version will highly resemble them, and the likelihood of re-identification is much higher. Whereas in the case where synthetic data points are relatively farther away from the actual data points, the density will be low, meaning the privacy risk too.

Why This Method is Used

This method is followed to avail an intuitive and quantitative measure of privacy risk. This

will help us understand whether synthetic data points are too similar to real data by calculating how densely synthetic data points are distributed around original data points. The more similar the synthetic data points, the greater the risk that a malicious actor could use the synthetic dataset to make sensitive inferences about individuals in the original dataset. Thus, distance density will serve as a proxy in the measurement of privacy risk through proximity.

- Standardization of Data: First, the original data, CTGAN, and TVAE are standardized using StandardScaler. Standardization ensures all features are on the same scale; this is a requirement for distance-based metrics to make sense and not be biased by the different feature scales.
- Nearest Neighbor Computation: For every synthetic data point, we calculate its distance to the closest neighbour in the original set. Employing the NearestNeighbors algorithm, for each synthetic data point we identify its closest original data point.
- Density Calculation: The distances are then averaged to arrive at the mean distance. The density becomes the inverse of this mean distance, i.e., density = 1 mean_distance. This inverse is a simple measure of density, showing how close the synthetic points are from the original points. A high value in the density means that the synthetic data points lie closer on average to the original data points, hence the higher risk of privacy.
- Distance Distribution Plot: A histogram of the distribution is plotted with a kernel density estimate overlay. This can be used to visually inspect the spread of the distances between the synthetic and original points. The tighter the spread is around smaller distances, the more similar they are, and hence, the higher the privacy risk. Conversely, the wider the spread around larger distances, the less similar they are, and hence, the lower the privacy risk.
- Comparison and Interpretation: In this case, the density scores of both CTGAN and TVAE are compared. The one with a lower score should be taken as the synthetic dataset with lower privacy risk since its points are further away from the original data, and hence re-identification is unlikely.

Merits of the Distance Density Method

- The proposed method is an easy-to-understand metric for estimation of privacy risk, whose interpretation is straightforward:lower the density, the lower the privacy risk. It gives a clear idea of how much the synthetic data is close to the original data due to the direct measurement of the distances between synthetic and original data for privacy evaluation.
- Visual Analysis: The distance distribution plot gives an idea of the synthetic points clustered around the original points. It helps intuitively understand the privacy risk.

Limitation

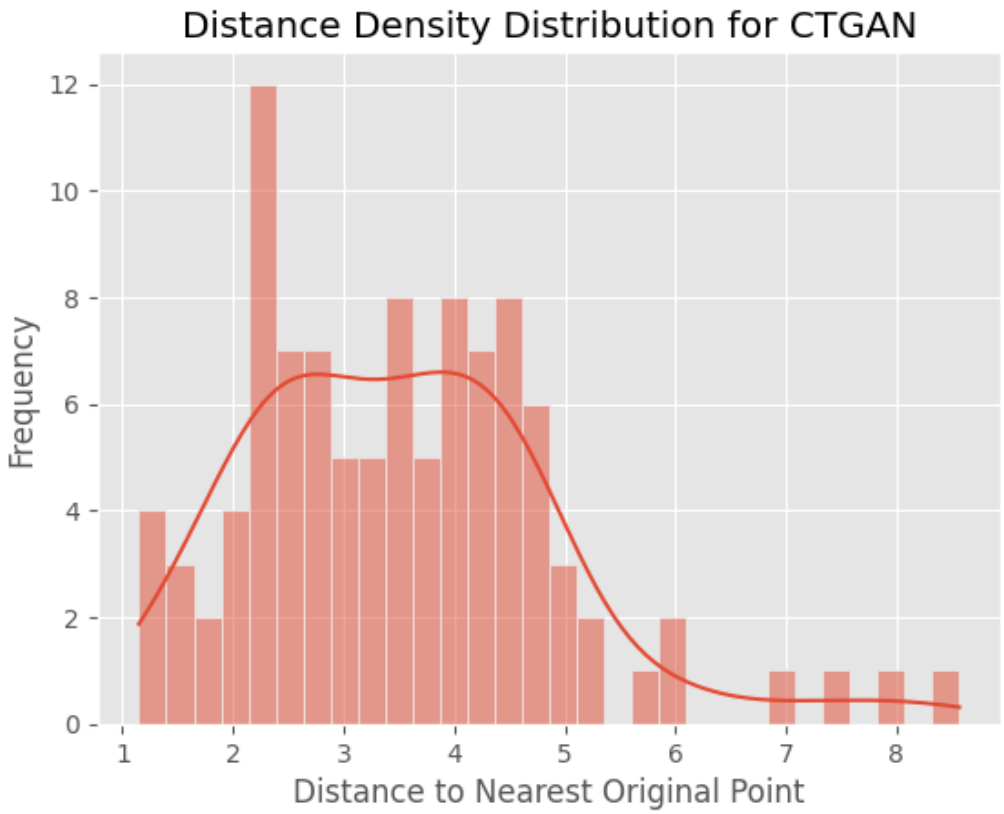
- Assumption of Privacy Risk: This method assumes that distance is a good proxy for privacy risk, while in fact the latter might be more complex than proximity issues only.
- Sensitive to Outliers: This may be prone to outliers or highly similar synthetic points that do not represent overall privacy risk.

- No Absolute Privacy Standard: Although it provides a relative measure of the privacy risk between datasets, it does not give an absolute measure on the protection of privacy, since the notion of what distance is considered "safe" depends on the context.
- Limited to Numerical Data: The technique works best for continuous, numerical data. For categorical data, other privacy metrics might be better suited.

```
    import numpy as np
import pandas as pd
from sklearn.neighbors import NearestNeighbors
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import matplotlib.pyplot as plt

# Standardize the datasets
scaler = StandardScaler()
original_scaled = scaler.fit_transform(mortality_pred)
ctgan_scaled = scaler.transform(train_ctgan)
tvae_scaled = scaler.transform(train_tvae)

def calculate_distance_density(original, synthetic, label):
    # Find the nearest neighbor distance for each synthetic
    #point to original points
    nbrs = NearestNeighbors(n_neighbors=1, algorithm='ball_tree').fit(original)
    distances, _ = nbrs.kneighbors(synthetic)

    # Flatten the distances and calculate density metrics
    distances = distances.flatten()
    mean_distance = np.mean(distances)
    density = 1 / mean_distance  # a simple density measure

    print(f"Distance Density for {label}: {density:.4f}")

    # Plot the distribution of distances
    sns.histplot(distances, bins=30, kde=True)
    plt.title(f"Distance Density Distribution for {label}")
    plt.xlabel("Distance to Nearest Original Point")
    plt.ylabel("Frequency")
    plt.show()

    return density

# Calculate distance density for CTGAN and TVAE synthetic datasets
density_ctgan = calculate_distance_density(original_scaled,
ctgan_scaled, "CTGAN")
density_tvae = calculate_distance_density(original_scaled,
tvae_scaled, "TVAE")

# Determine which synthetic dataset has a lower privacy risk
if density_ctgan < density_tvae:
```

```
    print("CTGAN has a lower density (lower privacy risk).")
else:
    print("TVAE has a lower density (lower privacy risk).")
```

```
Distance Density for CTGAN: 0.2834
Distance Density for TVAE: 0.9863
```

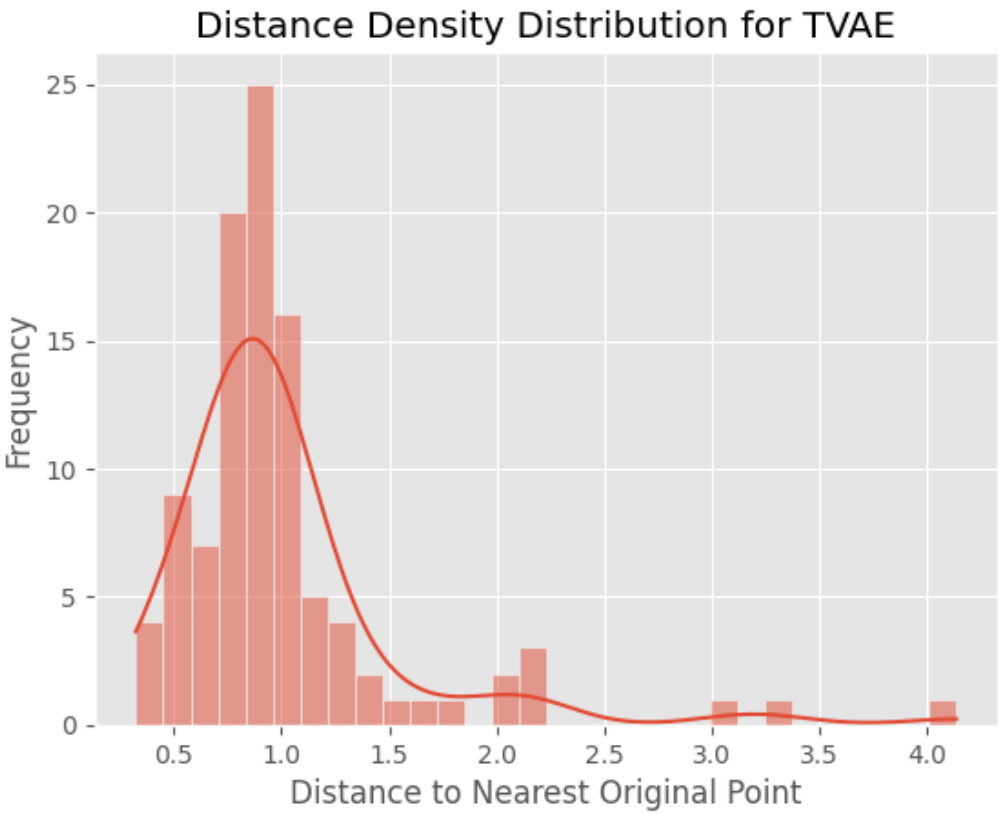


Final Conclusion

The results give the calculated distance density scores between CTGAN and TVAE. Based on the output: This means that if CTGAN has higher density, the generated synthetic data are closer to the original points, implying a higher risk concerning privacy. TVAMP is, in this case, better in the aspect of privacy as it has a lower density score, hence its synthetic data points being generally farther from the original data points, reducing the re-identification risk. In a word, the approach of Distance Density offers a more pragmatic measure of privacy risk since it provides the closeness between the synthetic data points and those in the original data. In this regard, TVAE has a lower score with respect to density and can be considered safer compared to CTGAN. However, Distance Density cannot be relied upon for complete judgment of privacy, and it is highly recommended to employ it along with other measures of privacy.

3.5.3 Bayesian Privacy Metrics

Think of Bayesian privacy metrics as tools that help us understand how much information about a person can be "guessed" or inferred based on some data that's been shared. Here, the main idea is that an "adversary" (someone trying to learn private information) has some knowledge before seeing the data and then uses what they see to make better guesses. This kind of privacy metric is based on Bayesian inference, which is all about updating beliefs based on new information.



Workflow :

- Updating Beliefs: Suppose an adversary has some initial guess about someone's sensitive data (like their income or health condition). When they see new data, they update this guess, becoming more (or less) certain. Bayesian metrics look at how much more confident they become after seeing the data.
- Measuring the Privacy Risk: These metrics give us a sense of "inference risk," or how likely it is that an adversary can figure out someone's private information based on what they know and what they observe. For instance, Bayesian privacy metrics can be used to measure the possibility that private information could still be deduced if a business distributes anonymised data.
- Mutual Information & Error Rates: Some particular metrics include examining predicted inference error, which calculates how effectively an adversary can forecast sensitive details after viewing the data, or mutual information, which illustrates how much uncertainty is decreased by the data.

These metrics are particularly useful when there's some known background knowledge about people's data, like in recommendation systems or healthcare, where personal information is sensitive but sometimes necessary for the system to function well.

3.5.4 syntactic Privacy Measures

Syntactic privacy measures are a bit different. Instead of focusing on what someone could infer about you, they're more about the structure of the data itself, and how it's altered or organized to hide individual details. These measures don't rely on assumptions about what an adversary

knows, instead they enforce rules to make sure certain patterns or combinations in the data can't easily identify someone.

Here are a few popular syntactic privacy measures:

- k-Anonymity: This rule makes sure that each person in a dataset "blends in" with at least k-1 other people. So if k=5, each individual's data would look identical to at least four other people in key respects, like age or zip code, making it harder to pick anyone out uniquely.

- *l*-Diversity: *l*-diversity takes k-anonymity a step further. It ensures that within each group of similar people, the sensitive attribute (like a medical diagnosis or income level) varies across at least l values. This means that even if someone knows which group you're in, they can't be sure of your specific sensitive information.

- t-Closeness: This one's all about the distribution of sensitive information. With t-closeness, the distribution of sensitive values within any group is close to the distribution in the overall dataset. This prevents someone from inferring details based on subtle patterns or skews in the data.

Syntactic privacy measures are commonly used in data anonymization, especially when making datasets available for research or public use. They're relatively straightforward to implement and can work well when we need to protect against specific types of re-identification risks. However, they can be less effective when an adversary has a lot of additional information.

4 Conclusion

This work emphasizes the benefits of utilizing GANs for generating synthetic medical data, leveraging a balance between data utility and privacy. Among the models tested, CTGAN works best for applications requiring high statistical fidelity, such as predictive modeling and statistical analyses, because of its ability to closely replicate the structural patterns in original datasets. In contrast, TVAE offers better privacy security with larger Privacy at Risk values, which makes it more acceptable for applications where privacy protection plays a crucial role. Application of these GAN models allows the sharing and analytics of data in a compliant and secure manner, thereby advancing the utility of synthetic data in sensitive domains such as healthcare. The present exploration supports the utility of GANs as a useful tool to handle privacy and utility trade-offs in the generation of synthetic data.

5 Reference

References

Stephanie Glen. Kolmogorov-smirnov test: Definition, step by step examples, 2023. URL https://www.statisticshowto.com/kolmogorov-smirnov-test/. Accessed: 2024-11-03.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

Yunhui Long, Boxin Wang, Zhuolin Yang, Bhavya Kailkhura, Aston Zhang, Carl Gunter, and Bo Li. G-pate: Scalable differentially private data generator via private aggregation of teacher discriminators. *Advances in Neural Information Processing Systems*, 34:2965–2977, 2021.

Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):1–9, 2019.

Analytics Vidhya. Comprehensive guide to k-means clustering, 2019. URL https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/. Accessed: 2024-11-03.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019a.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan, 2019b. URL https://arxiv.org/abs/1907.00503.