# Leveraging GANs to Generate Tabular Synthetic Data

[TOC]

# Executive Summary

## Project Definition

**What is the context?**

Advanced analytics is transforming all industries and is inherently data hungry. In health care data privacy rules detract data sharing for collaboration. Synthetic data, that retains the original characteristics and model compatible, can make data sharing easy and enable analytics for health care data.

**What is the need for change? Why now?**

Conventionally statistical methods have been used, but with limited success. Current deidentification techniques are not sufficient to mitigate re-identification risks. Emerging technologies in Deep Learning such as GAN are very promising to solve this problem.

**What ONE question, if answered, would solve the problem?**

How can you certify that the generated data is as similar and as useful as original data for the intended uses?

**What is the desired end use?**

- Derive insights
- Use for new product design
- Use for software quality improvement

# Introduction

Optum deals with sensitive healthcare data that has Personal identifiable Information (PII) of 100M+ people and it is expanding every day. The healthcare industry is particularly sensitive as Patient Identifiable Information data is strictly regulated by the Health Insurance Portability and Accountability Act (HIPPA) of 1996. Healthcare firms need to keep customer data secure while leveraging it to innovate research and drive growth in the firm. However, current data sharing practices (to ensure de-identification) have resulted in wait times for data access as long as 3 months. This has proved to be a hindrance to fast innovation at Optum. The need of the hour is to reduce the time for data access and enable innovation while protecting the information of patients. The key question to answer here is:

**"How can we safely and efficiently share healthcare data that is useful?"**

**Complication**

The key questions involve the inherent trade-off between safety and efficiency. With the inception of big data, efficiency in the data sharing process is of paramount importance. Availability and accessibility of data ensure rapid prototyping and lay down the path for quick innovation in the healthcare industry. Efficient data sharing also unlocks the full potential of analytics and data sciences through use cases like the diagnosis of cancer, predicting response for drug therapy, vaccine developments, drug discovery through bioinformatics. Apart from medical innovation, efficient data sharing helps to bridge the shortcomings in the healthcare system through salesforce effectiveness, managing supply chain and improve patient engagement. While efficient data sharing is crucial, the safety of patient's data can not be ignored. Existing regulations like HIPPA and recent privacy laws like the California Consumer Privacy Act are focused on maintaining the privacy of sensitive information. More advanced attacks are being organized by hackers and criminals aimed at accessing personal information. As per IBM's report on cost data breaches, the cost per record is ~$150. But the goodwill and trust lost by the companies, cannot be quantified So, the balance between data sharing and privacy is tricky.

# History of the Project

Existing de-identification techniques involve two main techniques 1) Anonymization Techniques 2) Differential Privacy. Almost every firm relies on these techniques to deal with sensitive information in PII data. These techniques have proven to be successful in the past and thus act as low hanging fruit for any organization.

1. Anonymization techniques: These techniques try to remove the columns which contain sensitive information. Methods include deleting columns, masking elements, quasi-identifiers, k-anonymity, l-diversity, and t-closeness.

2. Differential privacy: This is a perturbation technique which adds noise to columns which introduce randomness to data and thus maintain privacy. It is a mechanism to help to maximize the aggregate utility of databases ensuring high levels of privacy for the participants by striking a balance between utility and privacy.

However, these techniques are not cutting edge when it comes to maintaining privacy and data sharing. Rocher et al have proven that 99.98 percent of Americans (in a sample size of the population of Massachusetts) would be correctly re-identified in any dataset using as few as 15 demographic attributes. They conclude that "even heavily sampled anonymized datasets are unlikely to satisfy the modern standards for anonymization set forth by GDPR and seriously challenge the technical and legal adequacy of the de-identification release-and-forget model.

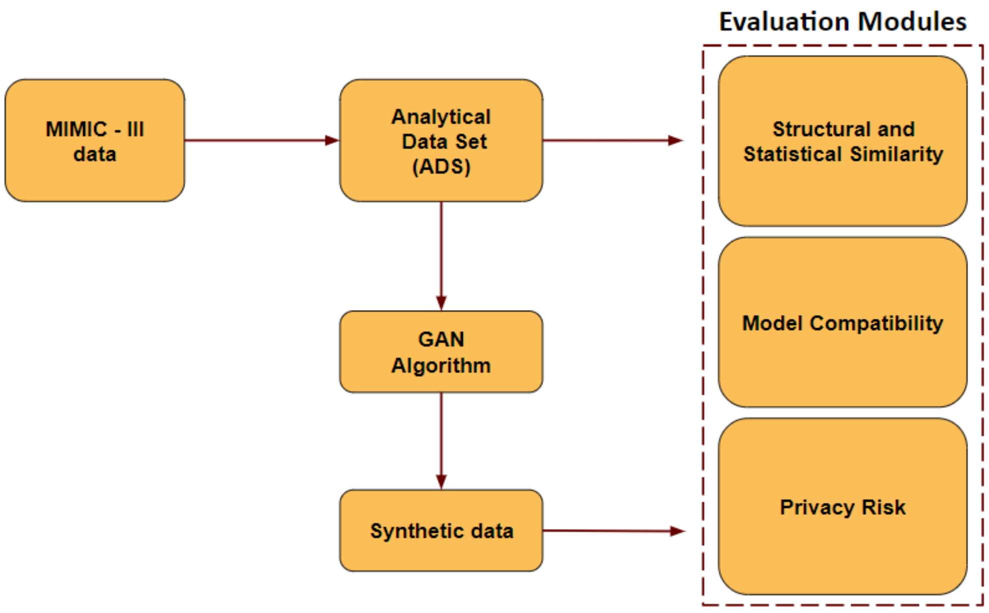
**Proposition**

Currently, the field of AI which is being given a lot of importance is Deep Learning. It addresses the critical aspect of data science in this age through universality theorem (identifying function form) and representation learning (correct features). Of late, generative modeling has seen a rise in popularity. In particular, a relatively recent model called Generative Adversarial Networks or GANs introduced by Ian Goodfellow et al. shows promise in producing realistic samples. While this is a state-of-the-art deep learning models to generate new synthetic data, there are few challenges which we need to overcome.

| Salient Features | Challenges |
|---|---|
| Neural Network is cutting edge algorithm in industry | Trained to solve one specific task, can it fit all use cases? |
| Generate image using CNN architecture | Can we generate table from relational databases? |
| Generate fake images of human faces that looks realistic | Would it balance the trade-off between maintaining utility and privacy of data |
| Requires high computational infrastructure like GPUs | How to implement GAN for big data? |

# Methodology

In order to validate the efficacy of GANs to serve our purpose, we propose a methodology for thorough evaluation of synthetic data generated by GANs.



## Synthetic Data Generation

This method is the state of the art in reducing the reidentification risk. As we observed earlier, Data anonymization if effective but reduces the utility, Differential privacy adds small noise but has very bad model compatibility. However, Synthetic data, can be tuned to add privacy without losing either the utility, neither exposing privacy of individual data points. As the data doesn't represent any real entity, the disclosure of sensitive private data is eliminated. If the information available in the released synthetic data matches with any real entity participated in the original data then it is purely a co-incidence which gives individuals plausible deniability

A synthetic dataset is a repository of data that is generated programmatically.

- It can be numerical, binary, or categorical (ordinal or non-ordinal),
- The **number of features and length of the dataset** should be arbitrary
- It should preferably be **random** and the user should be able to choose a wide variety of **statistical distribution** to base this data upon i.e. the underlying **random process can be precisely controlled and tuned**,
- If it is used for classification algorithms, then the **degree of class separation** should be controllable to make the learning problem easy or hard
- Random noise can be interjected in a controllable manner
- For a regression problem, a complex, **non-linear generative process** can be used for sourcing the data

## Statistical Similarity

The team has to make sure that the generated datasets are statistically similar to the original data to preserve its utility. The generated dataset should have minimal loss when compared to the original data. For both categorical and continuous value columns, the algorithms should be robust enough to not only preserve the multimodal distribution for individual columns, but also the joint distribution of the columns. the algorithm should detect intricate relationships between columns and preserve them in the generated synthetic data working equally well on balanced as well as imbalanced datasets.

We will be evaluating the datasets as follows:

**Descriptive Statistics**

- Central Tendencies (Mean, Median and Mode)
- Standard Deviation
- Skewness
- Kurtosis

- Unique Values

**Principle Component Analysis**

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

In simpler terms, PCA combines your input features in a specific way that you can drop the least important feature while still retaining the most valuable parts of all of the features. As an added benefit, each of the new features or components created after PCA are all independent of one another.

**t-Distributed Stochastic Neighbor Embedding (t-SNE)**

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. It is extensively applied in image processing, NLP, genomic data and speech processing. To keep things simple, here's a brief overview of working of t-SNE

- The algorithms starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space. The similarity of points is calculated as the conditional probability that a point A would choose point B as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at A.
- It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.
- To measure the minimization of the sum of difference of conditional probability t-SNE minimizes the sum of Kullback-Leibler divergence of overall data points using a gradient descent method.

**Note**: Kullback-Leibler divergence or KL divergence is is a measure of how one probability distribution diverges from a second, expected probability distribution.

In simpler terms, t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding.

In this way, t-SNE maps the multi-dimensional data to a lower dimensional space and attempts to find patterns in the data by identifying observed clusters based on similarity of data points with multiple features. However, after this process, the input features are no longer identifiable, and you cannot make any inference based only on the output of t-SNE. Hence it is mainly a data exploration and visualization technique.

**PCA vs t-SNE**

Although both PCA and t-SNE have their own advantages and disadvantages, some key differences between PCA and t-SNE can be noted as follows:

- t-SNE is computationally expensive and can take several hours on million-sample datasets where PCA will finish in seconds or minutes.
- PCA it is a mathematical technique, but t-SNE is a probabilistic one.
- Linear dimensionality reduction algorithms, like PCA, concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is essential that similar data points must be represented close together, which is something t-SNE does not PCA.
- Sometimes in t-SNE different runs with the same hyperparameters may produce different results hence multiple plots must be observed before making any assessment with t-SNE, while this is not the case with PCA.
- Since PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.

Below Diagram provides a more detailed approach to the methodology outlined above.

**Column-wise Evaluation:**

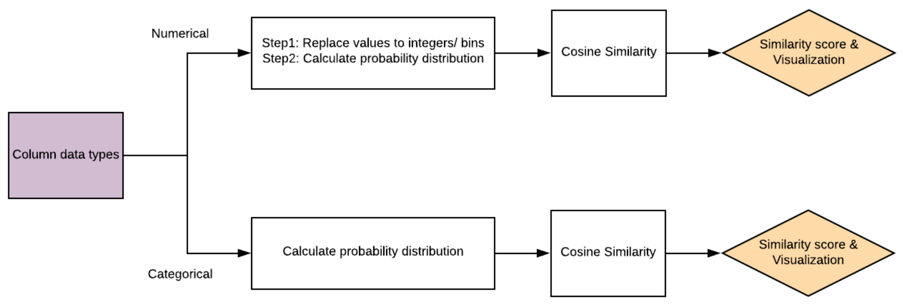
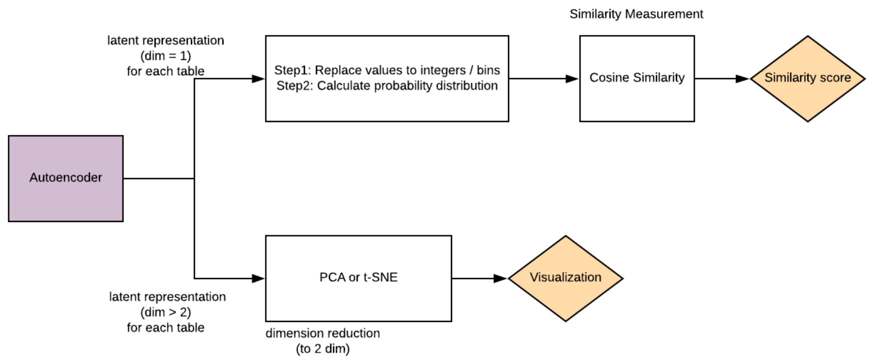

Numerical
Step1: Replace values to integers/ bins
Step2: Calculate probability distribution
Cosine Similarity
Similarity score & Visualization
Column data types
Calculate probability distribution
Cosine Similarity
Similarity score & Visualization
Categorical

**Table-wise Evaluation:**


Similarity Measurement
latent representation (dim = 1) for each table
Step1: Replace values to integers / bins
Step2: Calculate probability distribution
Cosine Similarity
Similarity score
Autoencoder
PCA or t-SNE
Visualization
latent representation (dim > 2) for each table
dimension reduction (to 2 dim)

## Model Compatibility

1. **Overview**

The synthetic data generated by GAN algorithms can be shared with internal business, academic researchers and third parties to tackle business problems. These stakeholders can then use machine learning algorithms on synthetic data to perform future predictions and create strategies as per their needs. In order to serve these needs, the synthetic data generated should have the same utlity as the original data and provide fairly similar prediction if not exactly same. Thus, it's crucial to evaluate if models generated using synthetic data are compatibile with original data? In this module, we will build different machine learning algorithms on 2 different use cases; a) Length of Stay b) Mortality prediction. The description of use cases can be found under **Use cases** section. These use cases are selected to evaluate both regression and classification models.

### Methodology

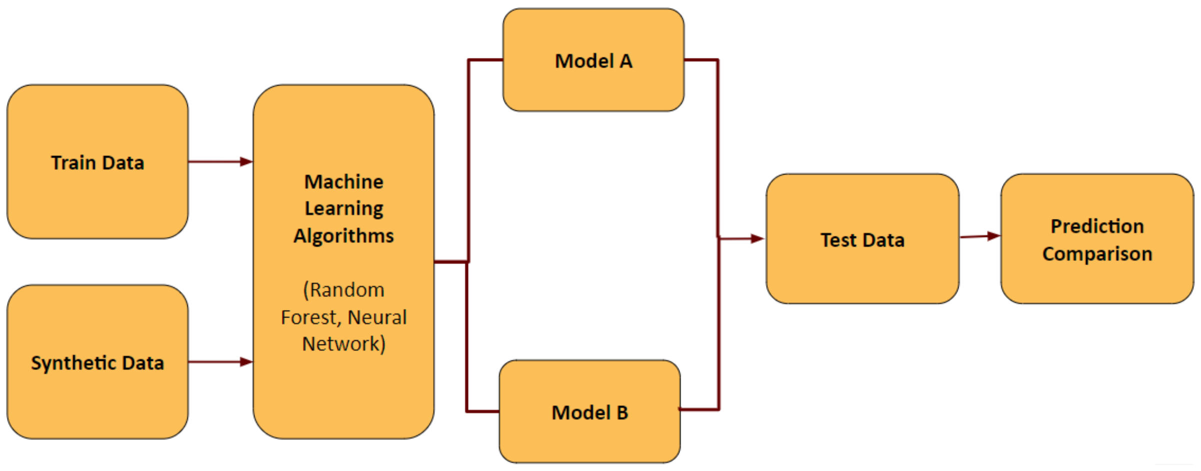
i. **One hot encoding (optional):** In case there are any categorical variables required for prediction, we first need to perform one hot encoding. This is done to make sure we don't miss out any categories in test dataset after splitting data into train and test. Example, in case of ethnicity column, there are multiple types like White, Hispanic, Asian, African etc. If we split data into train and test before performing one hot encoding, then there are chances that no Hispanic is present in test data and only present in train data. This can lead to problems while evaluating accuracy of models.
ii. **Split data into train and test:** The analytical data set generated as per the use cases is first split into 2 parts i.e. train and test in the ratio of 80:20. While splitting data set, stratified sampling is performed using the dependent / target variable, duration of stay in use case 1 (length of stay) and expiry flag in use case 2 (mortality prediction).
iii. **Generate Synthetic Data:** Train dataset is used as an input to GAN algorithms to generate synthetic data of same name of rows. We generated synthetic data using TGAN and CTGAN which was then used to build models.

iv. **Standarize variables (optional):** Numerical variables are present on different scales and some variables cna influence the prediction results more than others. Example, Age and Salary are both numeric variables however, range of salary is much bigger than range of age and thus can impact the prediction results. In order to account for this, we standardized all variables with mean 0 and standard deviation 1. This is same as z-score formula.

v. **Model building:** Using original train data and synthetic train data (generated by GAN algorithm). For use case 1 (Length of Stay), we used regression algorithms like Regression Tree, Random Forest, XGBoost, Support Vector Machine and K-Nearest Neighbor. For use case 2 (Mortality Prediction), we used classification algrotihms like Logistic Regression, XGBoost and Neural Network. Variety in use cases allows us to evaluate performance of synthetic data on various machine learning algorithms. In classification, due to imbalanced class of target variable, we used parameters within algorithms to assign appropriate weightage to each class.

vi. **Hyperparameter tuning:** 5-fold cross validation is performed along with grid search to perform hyperparameter tuning and selected best combination to improve the models. The metrics to evaluate can be different for different use case example, for mortality prediction, focus is on recall because we want to make sure we predict the patients who are going to die and take actions to save lives.

vii. **Prediction:** Finally, test data is used for prediction based on models generated and range of metrics were reported to compare performance. For classification, metrics like accuracy, precision, recall, f1-score and AUC-ROC. For regression, metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error.



Above image gives us a better idea of the overall process we are going to follow here.

- With the existing data Sources, the team will formulate analytical datasets common use cases with the data such as
    - Predicting length of stay in the ICU
    - Predicting Hospitality Mortality Rate
- These datasets will be run through various machine learning algorithms ranging from simple, ensemble to Neural networks to evaluate their performance on holdout data to compare their usefulness in real world usage

## Reidentification Risk

Along with Statistical similarity and Model compatibility, we need to ensure to minimize reidentification risk for the data points. Current anonymization techniques directly mask user demographics information to protect privacy, but bad actors can still join this data with other databases to identify individuals.

Original data has the best utility but as we mask more and more demographics information for privacy and regulatory reasons, the information provided by the dataset decreases. This in turn slows down research and development which could've potentially saved lives.

Sone of the current industry practices to prevent reidentification include:

**Data Anonymization**
Data anonymization means directly masking the important demographic and personally identifiable attributes from the dataset.

Few techniques include:

1. **Removal:** This process involves removing entire fields of data to reduce the risk of linking it to any source.

2. **Redaction:** This is used in many forms of government communication. It's a simple form of removing sensitive information – someone will print out a hard copy of a document and manually mark out sensitive or identifying information before passing it off to another party.

3. **Encryption:** For the most security, data anonymization isn't meant to be able to be reversed, but some people and organizations still use encryption as their means of anonymization. Encryption uses technology to render sensitive information as unreadable or unintelligible and can only be read after application of a decryption key, which must be kept separate from the encrypted files. Because there is a decryption key at all, however, there is a higher risk of potentially exposing or accessing sensitive information.

## Privacy Risk

Data breaches have been on the rise. United Healthcare / Optum deals with the PII (personally-identifiable information) of about 100M patients**. Hence there is a need to ensure that the data shared with the analyst working under does not jeopardize the privacy.

The solution to generate synthetic data has been gaining great traction, but it is still at a nascent stage in terms of research and deployment. A common problem is the idea of coming up with metrics that define the level of privacy achieved in a synthetic dataset, that are easily explainable.

### Common Options to reduce Privacy Risk

#### Reduce the number of columns by removing identifier columns:

Risks: Very common columns like SSN or Patient ID which are unique to a record need to be removed and these definitely reduce the Privacy Risk dramatically as they can be used to join with external tables and mine information. But in practice, these identifier information are not very useful in the use cases for Optum's research initiatives. The question then becomes - what other columns do we need to drop in order to reduce our Privacy Risk? There is no way to measure which columns make a dataset sensitive.

#### Reduce the number of rows:

Risks: We still do not have a quantifiable metric to say how much of Privacy Risk was reduced. The larger risk here is that we arbitrarily remove a fraction of the dataset, this might lead to the dataset becoming less useful. We need to remove rows in a way that retains the usefulness of data, while reducing the risk. Note that every single datapoint that is retained is 100% exposed and can be used by bad actors to mine information (by joining with external tables)

#### Synthesize new data that resembles original data

Risks: There is a potential that our synthetic data generation tool presents data that is still relatively close or 100% similar to our original data points if we try to maximize the utility of the dataset too much. We need a metric to still quantify the risk that we accept in these scenarios.

The goal of this whitepaper is to define how Optum can define the Privacy Risk in a synthetically created medical dataset. We would like to walk through a new metric - **'Privacy At Risk (PaR)'**, how it can be used to define risk in multiple situations, and how PaR values can be used to effectively assess situations and enable quicker data-sharing practices within Optum.

### Privacy At Risk (PaR) Metric

We are predominantly concerned about bad actors being able to join our datasets with the synthetic dataset with outside information to gain access to our PII information. Hence, if there is an opportunity to confuse the bad actor in a way they cannot link a synthetic datapoint to an original datapoint, that is the ideal scenario for us.

**The PaR Metric works** by leveraging this idea of confusion. How many data points in our set can be confused for with other people/other records? The higher the confusion, the lesser the chance of a person being re-identified. The focus is primarily on whether including a certain record increases the chance of exposing a person/record and the degree of the exposure.

The Synthetic Data Generation process by itself brings in noise into the data, in a way that maximizes utility while also minimizing the chances of data looking like the original data (hence increasing the confusion aspect)

But there is still a possibility of the synthetic data presenting exact clones of the original data points when overfitted while modelling. We need to be able to catch the situations

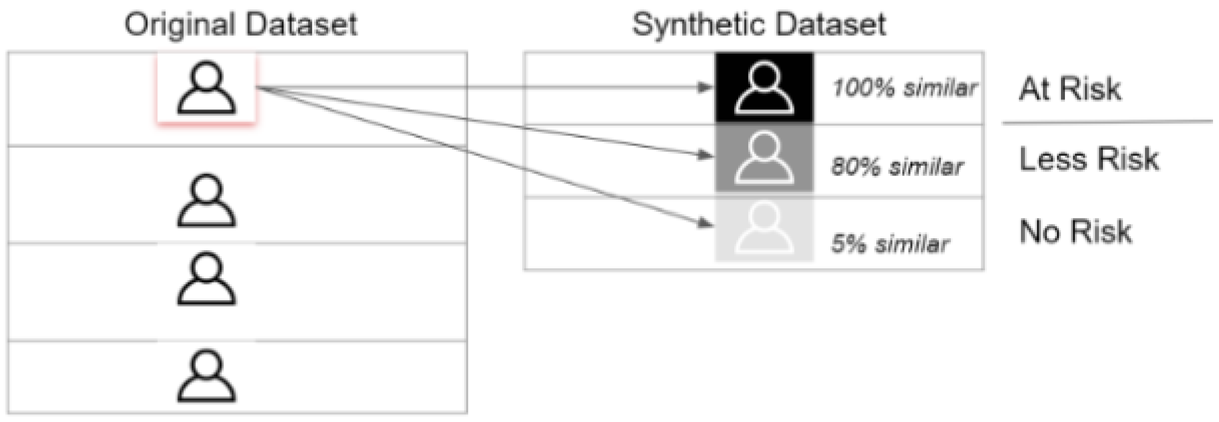
But what about situations where a synthetic datapoint is x% similar? (x: 0-99). The problem with this approach would mean we need to set thresholds pertaining to every situation. We overcome this problem by bringing the problem down to a relative scale within the dataset, by comparing External and Internal Similarity to assess the level of confusion as a binary variable.

## External Similarity¶

For every datapoint in the original dataset, we would like to see how similar data points in the synthetic dataset are using distance/similarity metrics. For our analysis, we primarily use Euclidean distance after one-hot encoding.

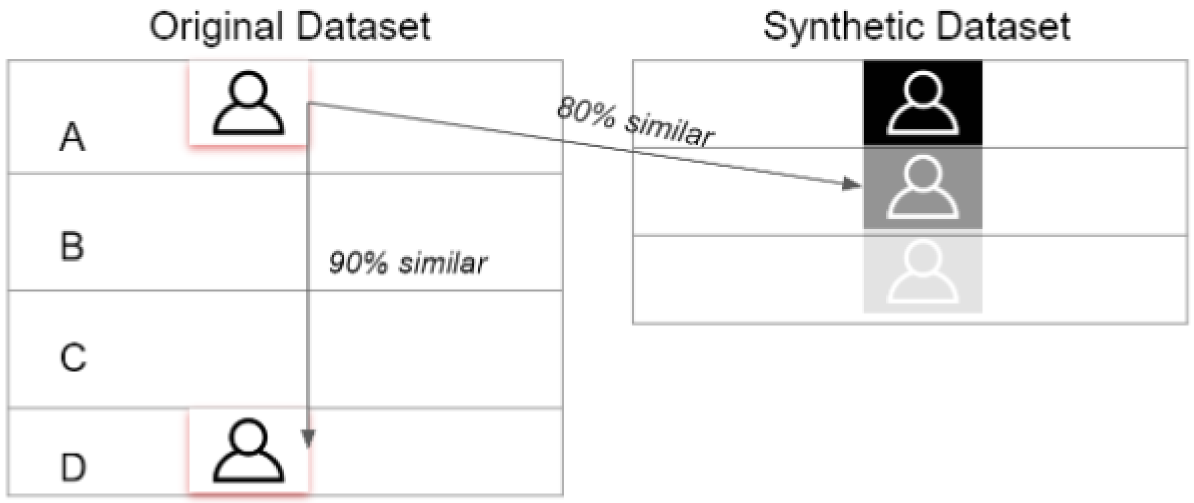
For every data point in the Original Dataset, we need to look at what records in the synthetic dataset are the most similar. This is the idea of 'External Similarity'.

Let's say for Person A in Original Dataset: Record 1 in synthetic dataset is 100% similar. This means that our synthetic data generation process has been practically useless for this user's privacy. The bad actor can still completely leverage the information in the synthetic data to find this user and mine information.



But we can observe Record 3 is only 5% similar, the chances of a bad actor using Record 3 to find Person A is extremely low.

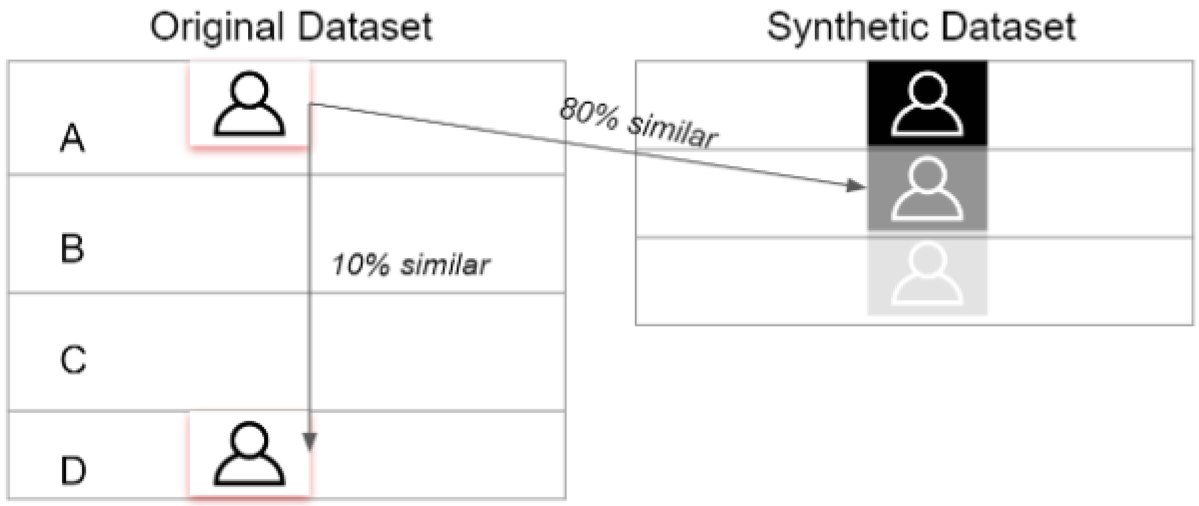
But let's now look at Record 3. We have an 80% similarity. Intuitively we may think this record is risky for us. But how do we draw a threshold to say what constitutes a risky data point and not? Is it really a risky data point, rather?



In this example you can see that although Person A is 80% similar with Record 2; We can also note that Person A is 90% similar to Person D. This is the idea of 'Internal Similarity'

What this essentially means is there is a higher chance of connecting Person A with Person B rather than connecting Person A with the synthetic Record 3.

Let's now look at the counter-scenario:



Now we can see the closest resemblance to A inside the Original Dataset is D again, but they are only 10% similar. So that means Person A is already a stand-out unique record. On top of this, our Synthetic Dataset Record 2 is now 80% similar. So when a bad actor gets access to Record 2, there is a high chance they can connect it with Person A and nobody else without any confusion. We need to avoid these scenarios.

*Summarizing,*

- Internal Similarity > External Similarity : Ideal Data Point for us; Bad actors cannot attribute a data point to a single person/original record.
- External Similarity > Internal Similarity: Risky Data Point for us; The record in synthetic dataset is so unique that the synthetic data record can only point to one user record and hence increases the privacy risk of that data point.
- Hence Privacy Risk = Number of Risky Points / (Number of Ideal Points + Number of Risky Points)

**PaR is a conservative metric**

A Privacy at Risk value of 6% does not mean 6% of the records are exposed to bad actors. It only means that there is a higher potential of a bad actor trying to interpret the synthetic information to a real record/person with 6% of the records. The inherent assumption here is that all the records in the original dataset are potentially vulnerable.

## Distance / Similarity Metrics

Determining which synthetic data records are similar to the original dataset

### 1. Personal Information Dataset: A dataset that consists of one row per person

In this case, one record constitutes a single real person (as in our examples) and one person has only one record in the table.

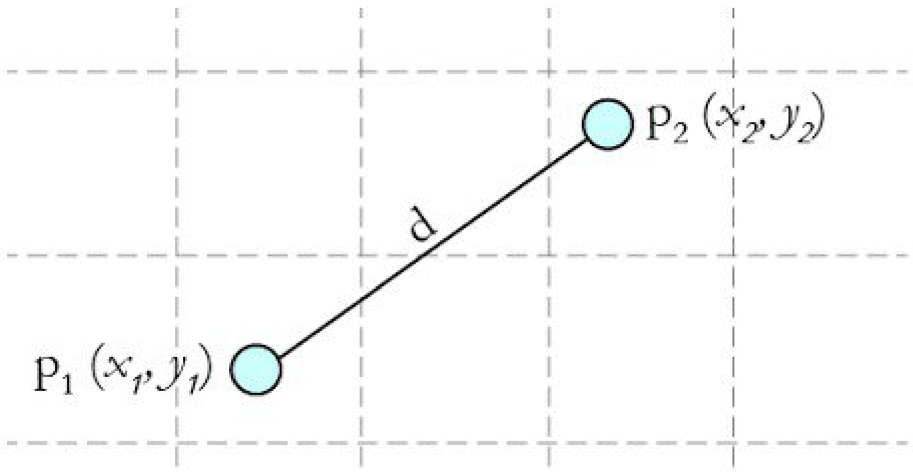
We use our synthetic data generation techniques to come up with an equal number of *'fake'* people for the synthetic dataset.

So the question becomes : Which of the fake people's information can be used to connect back to the original person and mine their information? Hence we need to find out how similar are original person information these fake people.

We compute similarity using Distance Metrics in this analysis.

There are a number of Distance Metrics to choose from - Euclidean, Manhattan, Gower Distance (which can handle categorical and continuous variables) and so on - but for our analysis, we prefer to use the simple Euclidean distance after one-hot encoding the data.

A lower value of Euclidean Distance means a higher degree of similarity and vice versa.



$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

We go about computing the distance between every single datapoint in the Original Dataset with every other point in the Synthetic Dataset.

These are all the **External Distances** we obtain as a matrix of size: Length of Original Dataset x Length of Synthetic Dataset

We also need to compute the Internal Distances so that we can compare the External Distance vs Internal Distance for every real Person in the dataset.

**Internal Distances** matrix of size: Length of Original Dataset x Length of Original Dataset

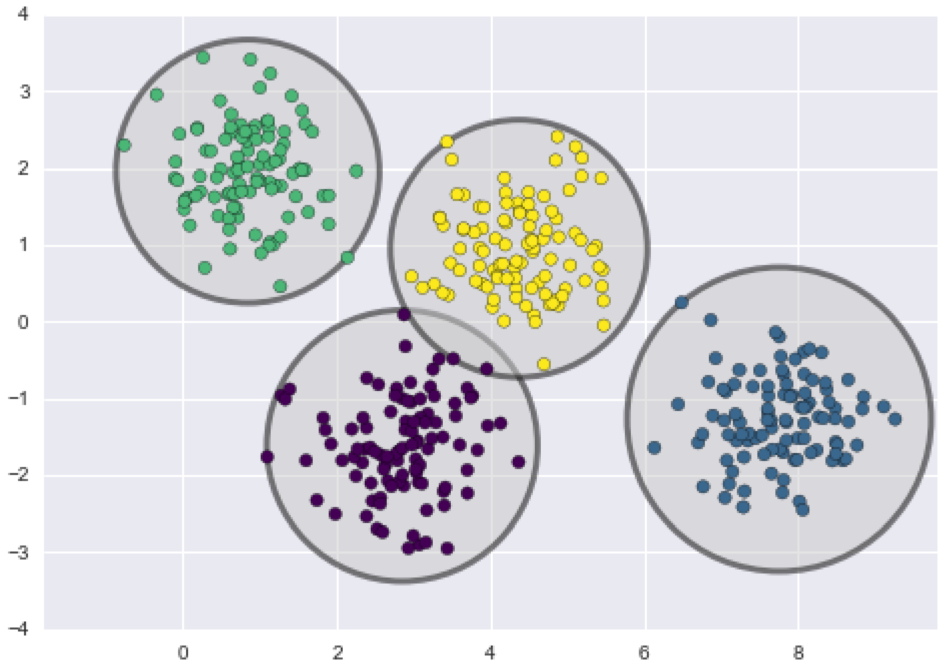
A note: Normalization is extremely important here

### Closest Person (Record) Matters here

We only wish to look at the closest/most similar records given a real person in the Original Dataset. The way bad actors can obtain information about this real person is by leveraging a very similar looking record and connect it to a real person's information outside.

But defining 'close' is subjective. A simple approach would be to use n=1 ie. only look at the closest person in the synthetic dataset (External Similarity), and the closest other person in the original dataset (Internal Similarity)

There's always an option to extend this into a radius of close people. By setting n=5, we can always look at the closest 5 people in Original and Synthetic datasets to achieve an idea of how close are records on average.



### Feature Sensitivity - Weighted Distance Metrics

Distance metrics, by default, may not account for domain knowledge about what defines similarity. Hence, a weighted distance metric might yield better results for a defined problem, given domain knowledge.

In our scenario, we are dealing with Mortality information of Patients in ICU Wards. We know that information like Gender, Ethnicity and Age can often be used to join with External Datasets and mine patient's personally identifiable information.

Hence, we might modify the Euclidean Distance to add a lower weight for features of Gender, Age and Ethnicity. When we add lower weights for highly sensitive fields, we report higher similarity and we will be better able to catch risky situations often.

$$d_{WE}(x, y) = \left( \sum_{k=1}^{p} w_k (x_k - y_k)^2 \right)^{\frac{1}{2}}$$

## 2. Multiple Records per User (Patient Journey Scenario)

This case will be dealt with differently with respect to Internal Similarity.

In this scenario, when we compare a single record with every other record in the same Original Dataset - we are not really answering the question of whether there is another person who really resembles this person/record. The problem here is each person might have multiple records. Hence when we compute Internal Similarity, we may end up finding 'similar people' always when those are all records belonging to the same person.

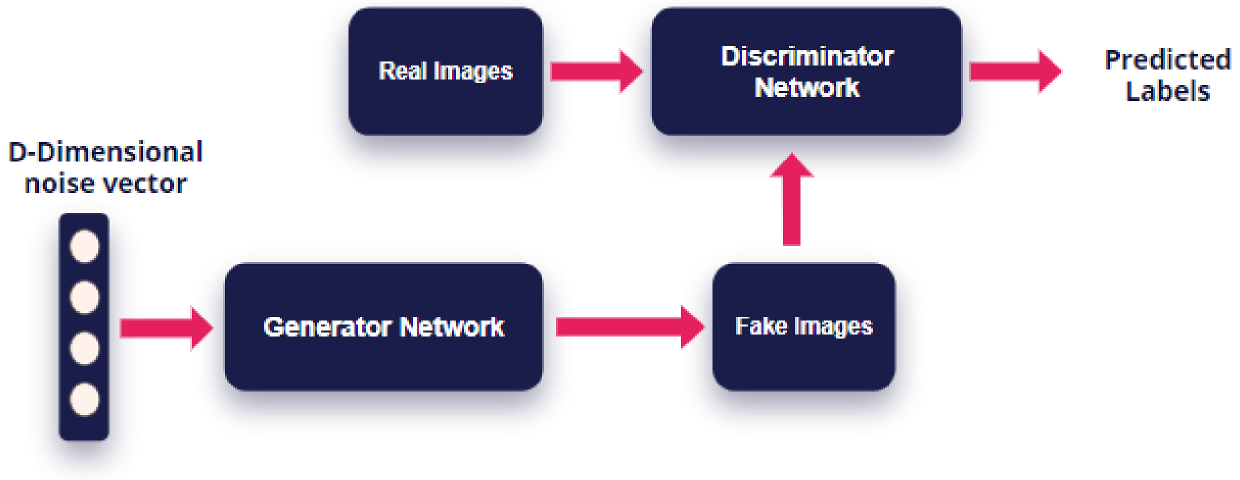
Hence the solution is to only look at records belonging to other users / patients in our use case. We use the Patient ID information to determine which records belong to which user. Hence:

***Internal Similarity is the distance of a given datapoint to every other datapoint belonging to a different user.***

# Why GANs?

## Introduction

A generative adversarial network (GAN) is a class of machine learning systems invented by **Ian Goodfellow** in 2014. GAN uses algorithmic architectures that use two neural networks, pitting one against the other (thus the “adversarial”) in order to generate new, synthetic instances of data that can pass for real data.



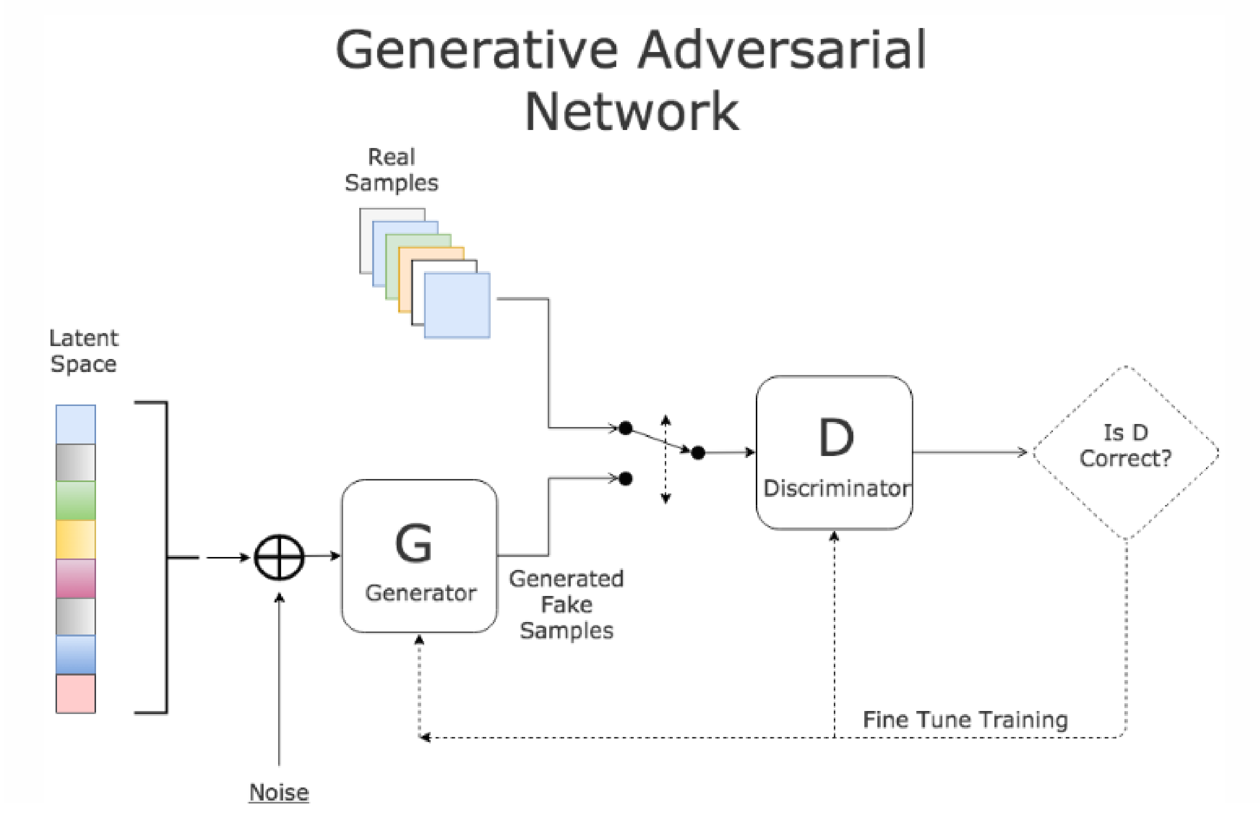
GANs consist of Two neural networks contest with each other in a game. Given a training set, this technique learns to generate new data with the same statistics as the training set. The two Neural Networks are named Generator and a Discriminator.

## GAN Working Overview

**Generator**
The generator is a neural network that models a transform function. It takes as input a simple random variable and must return, once trained, a random variable that follows the targeted distribution. The generator randomly feeds actual image and generated images to the Discriminator. The generator starts with Generating random noise and changes its outputs as per the Discriminator. If the Discriminator is successfully able to identify that generate input is fake, then then its weights are adjusted to reduce the error.

**Discriminator**
The Discriminators job is to determine if the data fed by the generator is real or fake. The discriminator is first trained on real data, so that it can identify it to acceptable accuracy. If the Discriminator is not trained properly, then it in turn will not be accurately able to identify fake images thus poorly training the Generator.

This is continued for multiple iterations till the discriminator can identify the real/fake images purely by chance only.



- **Algorithm:**
  Now lets see how GANs algorithm works internally.

- - The generator randomly feeds real data mixed with generated fake data for the discriminator

  - To begin, in first few iterations, the generator produces random noise which the discriminator is very good at detecting that the produced image is fake.
  - Every iteration, the discriminator catches a generated image as fake, the generator readjusts its weights to improve itself. much like the Gradient Descent algorithm
  - Over time, after multiple iterations, the generator becomes very good at producing images which can now fool the discriminator and pass as real ones.
  - Now, its discriminators turn to improve its detection algorithm by adjusting its network weights.
  - This game continues till a point where the discriminator is unable to distinguish a real image from fake and can only guess by chance.

# Existing Research in Synthetic Data Generation

## TGAN

This methodology has been created from the work provided in this paper:

Synthesizing Tabular Data using Generative Adversarial Networks

and this python package

https://pypi.org/project/tgan/

Generative adversarial networks (GANs) implicitly learn the probability distribution of a dataset and can draw samples from the distribution. Tabular GAN (TGAN) is a a generative adversarial network which can generate tabular databy learning distribution of the existing training datasets and can generate samples which are . Using the power of deep neural networks.

TGAN focuses on generating tabular data with mixed variable types (multinomial/discrete and continuous) and propose TGAN. To achieve this, we use LSTM with attention in order to generate data column by column. To asses, we first statistically evaluate the synthetic data generated by TGAN.

The paper also evaluates Machine learning models performance against traditional methods like modelling a multivariate probability or randomization based models.

## Data preparation

For a table containing discrete and continuous random variables, They follow some probability distribution. Each row in the table is a sample from this distribution, whihch is sampled independently and the algorithms learn a generative model such that samples generated from this model can satisfy two conditions:

- A Machine Learning model using the Synthetic table achieves similar accuracy on the test table
- Mutual information between an arbitrary pait of variables is similar

**Numerical Variables**

For the model to learn the data effectively, a reversible transformation is applied. The a numerical variables are converted into a scalar in the range (1, 1) and a multinomial distribution, and convert a discrete variable into a multinomial distribution.

Often, numerical variables in tabular datasets follows multimodal distribution. Gaussian Kernal density estimation is used to estimate these number of noes in the continuous variable. To sample values from these, a gaussian mixture model is used.

**Categorical Variables** - Improvement needed

categorical variables are directly converted to to one-hot-encoding representation and add noise to binary variables

In TGAN, the the discriminator D tries to distinguish whether the data is from the real distribution, while the generator G generates synthetic data and tries to fool the discriminator. the algorithm uses a Long Short Term Memory(LSTM) as generator and a Multi Layer Perceptron (MLP) as a discriminator.

## Implementation

```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import tensorflow as tf
from tgan.model import TGANModel
from tgan.data import load_demo_data
```

```
def tgan_run(data, cont_columns):
    tgan = TGANModel(continuous_columns)
    return tgan.fit(data)

def tgan_samples(model, num_samples):
    return tgan.sample(100000)
```

## Results

## CTGAN

CTGAN is a GAN-based method to model tabular data distribution and sample rows from the distribution. CTGAN implements mode-specific normalization to overcome the non-Gaussian and multimodal distribution (Section 4.2). We design a conditional generator and training-by-sampling to deal with the imbalanced discrete columns (Section 4.3). And we use fully-connected networks and several recent techniques to train a high-quality model.

**Several unique properties of tabular data challenge the design of a GAN model.**

- Mixed data types Real-world tabular data consists of mixed types. To simultaneously generate a mix of discrete and continuous columns, GANs must apply both softmax and tanh on the output.
- Non-Gaussian distributions: In images, pixels' values follow a Gaussian-like distribution, which can be normalized to [−1, 1] using a min-max transformation. A tanh function is usually employed in the last layer of a network to output a value in this range. Continuous values in tabular data are usually non-Gaussian where min-max transformation will lead to vanishing gradient problem.
- Multimodal distributions. We use kernel density estimation to estimate the number of modes in a column. We observe that 57/123 continuous columns in our 8 real-world datasets have multiple modes. Srivastava et al. [21] showed that vanilla GAN couldn't model all modes on a simple 2D dataset; thus it would also struggle in modeling the multimodal distribution of continuous columns.
- Learning from sparse one-hot-encoded vectors. When generating synthetic samples, a generative model is trained to generate a probability distribution over all categories using softmax, while the real data is represented in one-hot vector. This is problematic because a trivial discriminator can simply distinguish real and fake data by checking the distribution's sparseness instead of considering the overall realness of a row.
- Highly imbalanced categorical columns. In our datasets we noticed that 636/1048 of the categorical columns are highly imbalanced, in which the major category appears in more than 90% of the rows. This creates severe mode collapse. Missing a minor category only causes tiny changes to the data distribution that is hard to be detected by the discriminator. Imbalanced data also leads to insufficient training opportunities for minor classes.

When feeding data to the GAN algorithm, CTGAN samples so that all categories are correctly represented. Specifically, the goal is to resample efficiently in a way that all the categories from discrete attributes are sampled evenly (but not necessary uniformly) during the training process, and to recover the (not-resampled) real data distribution during test

These three things need to be incorporated:

- Modify the input for conditional vector creation
- The generated rows should preserve the condition
- The conditional generator should learn the real data conditional distribution

### Implementation

```
import pandas as pd
import tensorflow as tf

from ctgan import load_demo
from ctgan import CTGANSynthesizer
```

```
data = load_demo()
discrete_columns = ['workclass','education', 'marital-status', 'occupation', 'relationship', 'race', 'sex','native-coun
ctgan = CTGANSynthesizer()
ctgan.fit(data, discrete_columns)
```

### Results

## Differentially Private GAN (WIP)

Source: https://arxiv.org/pdf/1802.06739.pdf

One common issue in above proposed methodologies in GANs is that the density of the learned generative distribution could concentrate on the training data points, meaning that they can easily remember training samples due to the high model complexity of deep networks. This becomes a major concern when GANs are applied to private or sensitive data such as patient medical records, and the concentration of distribution may divulge critical patient information. Differentially Private GANs is achieved by adding carefully designed noise to gradients during the learning procedure.

DPGAN focuses on preserving the privacy during the training procedure instead of adding noise on the final parameters directly, which usually suffers from low utility. Noise is added to the gradient of the Wasserstein distance with respect to the training data.

**Note:** Wasserstein distance is a distance function defined between probability distributions on a given metric space

The algorithm guarantees that the parameters of discriminator and generator have differential privacy with respect to the sample training points. The algorithm inputs noise e in the generator parameters which enables this privacy, however one needs to perform a grid search over a large range of noise parameter **e** to get best results.

## PATE-GAN (WIP)

Source: https://arxiv.org/pdf/1906.09338.pdf

Generative Adversarial Networks (GAN) provide a powerful method for using real data to generate synthetic data but it does not provide any rigorous privacy guarantees. PATE GAN modifies the existing GAN algorithm in a way that does guarantee privacy

PATE GAN consists of two generator blocks called student block and teacher block on top of the existing generator block. With traditional privacy techniques, it is possible for the Generator to reconstruct the original data even after adding noise. PATE GAN prevents this by breaking down the generator into three stages. After the generator creates the data and adds noise, there is an ensemble block which factors in majority voting to create the input. After this there is a student block which aggregates the inputs from the teacher blocks and generates the final data.

The synthetic data is (differentially) private with respect to the original data DP-GAN: The key idea is that noise is added to the gradient of the discriminator during training to create differential privacy guarantees. Our method is similar in spirit; during training of the discriminator differentially private training data is used, which results in noisy gradients, however, we use the mechanism introduced in A noticeable difference is that the adversarial training is no longer symmetrical: the teachers are now being trained to improve their loss with respect to G but G is being trained to improve its loss with respect to the student S which in turn is being trained to improve its loss with respect to the teachers.

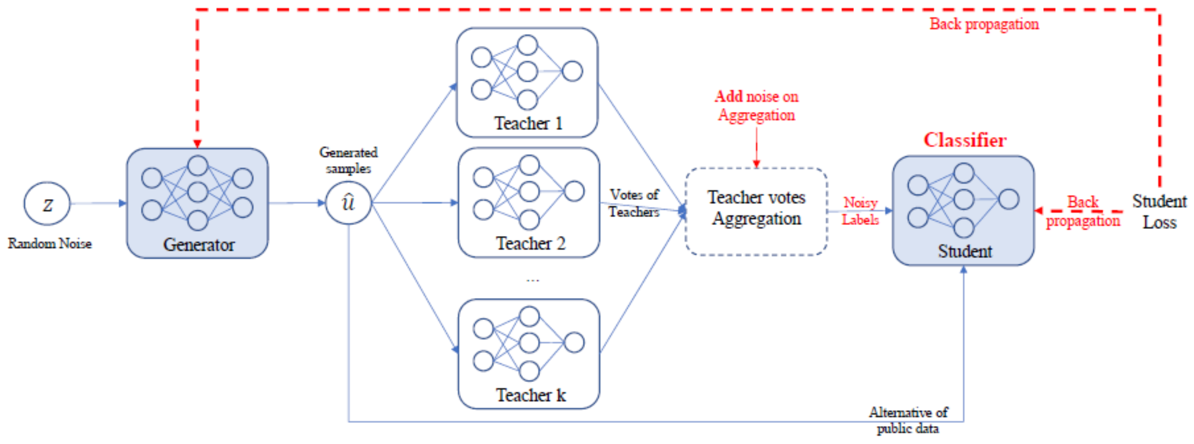


Figure 4: Block diagram of the training procedure for the student-discriminator and the generator. The student-discriminator is trained using noisy teacher-labelled generated samples (the noise provides the DP guarantees). The student is trained to minimize classification loss on this noisily labelled dataset, while the generator is trained to maximize the student loss. Note that the teachers are not updated during this step, only the student and the generator.

VARYING THE PRIVACY CONSTRAINT ($\epsilon$) IN TERMS OF AUPRC

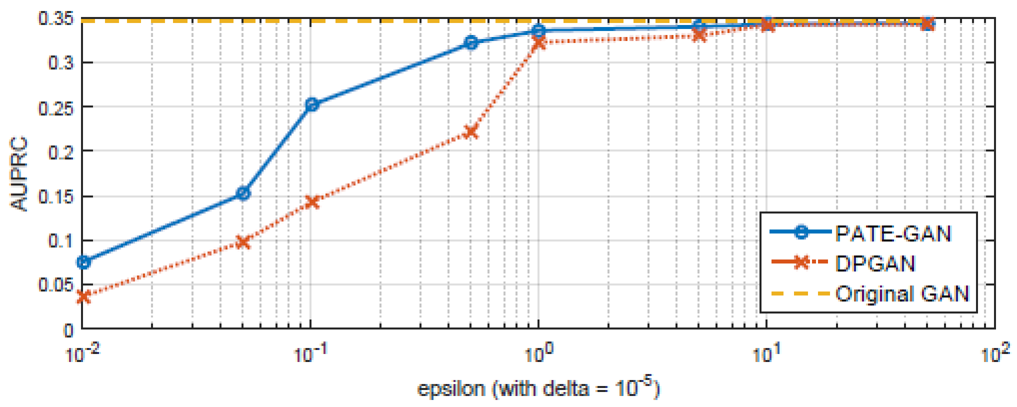


Figure 2: Average AUPRC performance across 12 different predictive models trained on the synthetic datas generated by PATE-GAN and DPGAN with various $\epsilon$ (with $\delta = 10^{-5}$) (Setting B).

## G-PATE (WIP)

Theoretically, the generator in GAN has the potential of generating an universal distribution, which is a superset of the real distribution, so it is not necessary for the student discriminator to be trained on real records. However, such a theoretical bound is loose. In practice, if a generator does generate enough samples from the universal distribution, there would be a convergence issue. On the other hand, when the generator does converge, it no longer covers the universal distribution, so the student generator may fail to learn the real distribution without seeing real records.

It is not necessary to ensure differential privacy for the discriminator in order to train a differentially private generator. As long as we ensure differential privacy on the information flow from the discriminator to the generator, it is sufficient to guarantee the privacy property for the generator. Therefore, instead of focusing on ensuring differential privacy for the whole GAN framework, we design a novel framework to guarantee that all information flowed from the discriminator to the generator satisfies differential privacy.

Compared to PATE-GAN, our approach has two advantages. First, we improve the use of privacy budget by applying it to the part of the model that actually needs to be released for data generation. Second, our discriminator can be trained on real data because itself does not need to satisfy differential privacy. The teacher discriminators do not need to be published, so they can be trained with non-private algorithms.

In addition, we design a gradient aggregator to collect information from teacher discriminators and combine them in a differentially private fashion.
Unlike PATE-GAN, G-PATE does not require any student discriminator. The teacher discriminators are directly connected to the student generator. The gradient aggregator sanitizes the information flow from the teacher discriminators to the student generator to ensure differential privacy The privacy property is achieved by sanitizing all information propagated from the discriminators to the generator.

# Use Cases

## MIMIC-III Dataset

We used MIMIC-III data for this project as it is an open source dataset and contains multiple tables which can be combined to prepare use cases. Data can be accesed on link provided below and in order to access data, few certifications are required which can be obtained by completing trainings on website. Medical Information Mart for Intensive Care Unit (MIMIC) III: Open source dataset developed by the MIT Lab comprising de-identified health data is a comprehensive clinical dataset of 40,000+ patients admitted in ICU. Few salient features of this databases include:

- **26 tables:** Comprehensive clinical dataset with different tables
- **40,000+ patients:** Data associated with multiple admissions of patients
- **65,000+ ICU admissions:** Admitted in ICUs at Beth Israel Deaconess Medical Centre, Boston, from 2001 to 2012
- **PII data:** Includes patient sensitive details like demographics, diagnosis, laboratory tests, medications etc.
- **Use cases:** Can be used for multiple analysis like clinical data (prescriptions), claims data (payer), members data (patient)

For this project, we focus on using specific tables from this database and create 2 different use cases which can be found in following section.

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

## Use Cases

### Use Case 1: Length of Stay

Predicting the length of stay in ICU using 4 out of 26 tables

Use case: Out of variety of possible use cases from MIMIC III dataset we focus on single use case to predict the number of days a patient stays in the ICU to further generate synthetic data. This usecase seemed important to us because its always a benefit for payer to predict the tendency of the length of stay of a patient. It helps in changing the premium charged by the payer according to the comparison of predictions and baseline (the defined no. of days covered by a particular plan of the patient). For this use case the model utilises the total number of diagnosis that occured for different disease category for each patient.

To build this use case we focus on primarily 4 tables.

1. **Patients:** Every unique patient in the database (defines SUBJECT_ID)
   **Columns like:** SUBJECT_ID, GENDER (count of rows: 46520 count of columns: 7)
2. **Admissions:** Every unique hospitalization for each patient in the database (defines HADM_ID)
   **Columns like:** SUBJECT_ID, HADM_ID, HOSPITAL_EXPIRE_FLAG, MARITAL_STATUS, ETHNICITY, ADMISSION_TYPE (count of rows: 58976 count of columns: 18 )
3. **ICUSTAYS:** Every unique ICU stay in the database (defines ICUSTAY_ID)
   **Columns like:** SUBJECT_ID, HADM_ID, LOS (count of rows: 61532 count of columns: 11 )
4. **Diagnosis_ICD:** Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system

**Columns like:** SUBJECT_ID, HADM_ID, NUMDIAGNOSIS (count of rows: 651047 count of columns: 5)

## Methodology

1. **Load csv files:** Read the comma seperated files donwloaded from link (https://mimic.physionet.org/gettingstarted/overview/)
2. **Merge tables:** Use 'SUBJECT_ID' to merge tables like ADMISSIONS, PATIENTS and ICU_STAYS and finally concatenate 'SUBJECT_ID' and 'HADM_ID' to form 'final_id' as composite key.
3. **Prepare diagnosis dataset:** DIAGNOSIS_ICD table is used to map the disease category type using the first three code digits if ICD-9 code. The mapping is used to convert the unique 6984 ICD-9 codes into 18 different disease area categories and finally concatenate 'SUBJECT_ID' and 'HADM_ID' to form 'final_id' as composite key..
4. **Pivot up diagnosis dataset:** After the mapping the disease categories using ICD-9 codes, the datset is pivoted up at the level of the 18 disease categories and the total count of diagnosis is being populated across 'final_id'
5. **Merge pivoted diagnosis datset to the main dataset:** Finally, the above generated dataset is then merged to the main dataset using the 'final_id' as the key.

Note: 6984 ICD-9 codes: The diagnosis dataset contains unique International Classification of Disease (ICD-9) codes
18 primary categories: We consider categories of conditions for the predictive modeling
Finally only the relevant columns required for the analysis are selected and we use the dataset for the synthetic data generation.
The final data has 116354 rows and 27 columns.

## Final data includes

- **Level of data:** Each instance in the final data set is unique admission for each patient and is defined by concatination of 'SUBJECT_ID' and 'HADM_ID' to form 'final_id'

- **Target Variables:** 'LOS' (length of stay) is used as target variable

- **Predictor variables:** 18 columns of different diagnosis category are used as predictor varibales.
  These 18 categories are:
  "certain conditions originating in the perinatal period"
  "complications of pregnancy, childbirth, and the puerperium",
  "congenital anomalies",
  "diseases of the blood and blood-forming organs",
  "diseases of the circulatory system",
  "diseases of the digestive system",
  "diseases of the genitourinary system",
  "diseases of the musculoskeletal system and connective tissue",
  "diseases of the nervous system",
  "diseases of the respiratory system",
  "diseases of the sense organs",
  "diseases of the skin and subcutaneous tissue",
  "endocrine, nutritional and metabolic diseases, and immunity disorders",
  "external causes of injury and supplemental classification",
  "infectious and parasitic diseases",
  "injury and poisoning",
  "mental disorders",
  "neoplasms" and
  "symptoms, signs, and ill-defined conditions".

- **Other descriptive variables:**

  "ADMISSION_TYPE", "INSURANCE", "ETHNICITY", "HOSPITAL_EXPIRE_FLAG", "GENDER" and "EXPIRE_FLAG"

**Code (data wrangling performed in R)**

1. Import required libraries and read csv files
2. Function for data preparation

### 1. Import required libraries and read csv files

```
install.packages('reshape')
library(reshape2)
library(dplyr)
```

### 2. Function for data preparation

```
# Note this code was written in R
# This code uses the filenames of admission, patients, icustays and diagnosis dataset as the input to the function usec


usecase_1 <- function(admissions,patients,icustays,diagnosis)
{

    #################### Loading datasets ##############################################

    adm <-read.csv(admissions)
    pat <- read.csv(patients)
    icu <- read.csv(icustays)
    diagn <- read.csv(diagnosis)
    map <- read.csv("mapping.csv")

    #################### Selecting specifc columns #####################################

    adm <- adm[,2:ncol(adm)]
    pat <- pat[,2:ncol(pat)]
    icu <- icu[,2:ncol(icu)]
    pres <- pres[,2:ncol(pres)]

    #################### Merging various tables ########################################

    merged_1<- merge(x=adm,y=pat,by.x = c("SUBJECT_ID"), by.y=c("SUBJECT_ID"))
    merged_2<- merge(x=merged_1,y=icu,by.x = c("SUBJECT_ID"), by.y=c("SUBJECT_ID"))

    #################### Data wrangling on Diagnosis dataset ###########################

    diagnosis_unique <- data.frame(unique(diagn$ICD9_CODE)
    names(diagnosis_unique)[1] <-"icd9"
    merged_diag<- merge(x=diagnosis_unique,y=map,by.x = c("icd9"), by.y=c("icd"))
    final_map <- merged_diag[,c(1,4)]
    diagn_final <- merge(x=final_map,y=diagn,by.x = c("icd9"), by.y=c("ICD9_CODE"))
    diagn_final$final_id <-paste(diagn_final$SUBJECT_ID,diagn_final$HADM_ID, sep="_")
    diagn_final$count <-1
    diagn_final <- diagn_final[,c(2,7,8)]

    #################### Pivoting up Diagnosis dataset #################################

    step1<- dcast(data = diagn_final, formula = final_id ~ icd_chp, fun.aggregate = sum, value.var = "count")
    merged_2$final_id <-paste(merged_2$SUBJECT_ID,merged_2$HADM_ID.x, sep="_")

    #################### Merging pivotted up diagnosis dataset with main dataset #######

    merged_3<- merge(x=merged_2,y=step1,by.x = c("final_id"), by.y=c("final_id"))
    merged_3 <- merged_3[,c(1,7,10,14,18,20,25,35,36:54)]
    merged_3 <- merged_3%>%filter(LOS!="NA")
}

# Finally write the database to be used as an input for various GAN algorithms
write.csv(merged_3,"full_database_los_v1.csv")
```

## Use Case 2: Mortality Prediction

Another use case which we implemented using MIMIC-III dataset is for mortality prediction. This use case is inspired by Kaggle kernel (reference below) where one can predict mortality just by the number of interactions betweeen patient and hospital as predictors through count of lab tests, prescriptions, and procedures. It can be used to evaluate privacy risk with the help of PII columns like GENDER, ETHNICITY, MARITAL_STATUS and also serves as classification problem where we have to predict if patient will expire or not for a single hospital admission.

Reference: https://www.kaggle.com/drscarlat/predict-hospital-mortality-mimic3

**Tables Used**

1. Patients - Every unique patient in the database (defines SUBJECT_ID)
   Columns like: SUBJECT_ID, GENDER
2. Admissions - Every unique hospitalization for each patient in the database (defines HADM_ID)
   Columns Like: SUBJECT_ID, HADM_ID, HOSPITAL_EXPIRE_FLAG, MARITAL_STATUS, ETHNICITY, ADMISSION_TYPE
3. CallOut - Information regarding when a patient was cleared for ICU discharge and when the patient was actually discharged
   Columns Like: SUBJECT_ID, HADM_ID, NUMCALLOUT (count of rows)
4. CPTEvents - Procedures recorded as Current Procedural Terminology (CPT) codes
   Columns Like: SUBJECT_ID, HADM_ID, NUMCPTEVENTS (count of rows)
5. Diagnosis_ICD - Hospital assigned diagnoses, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
   Columns Like: SUBJECT_ID, HADM_ID, NUMDIAGNOSIS (count of rows)
6. Inputevents_CV - Intake for patients monitored using the Philips CareVue system while in the ICU
   Columns Like: SUBJECT_ID, HADM_ID, NUMINPUTEVENTS (count of rows)
7. Labevents - Laboratory measurements for patients both within the hospital and in out patient clinics
   Columns Like: SUBJECT_ID, HADM_ID, NUMLABEVENTS (count of rows)
8. Noteevents - Deidentified notes, including nursing and physician notes, ECG reports, imaging reports, and discharge summaries.
   Columns Like: SUBJECT_ID, HADM_ID, NUMNOTEVENTS (count of rows)
9. Outputevents - Output information for patients while in the ICU
   Columns Like: SUBJECT_ID, HADM_ID, NUMOUTEVENTS (count of rows)
10. Prescriptions - Medications ordered, and not necessarily administered, for a given patient
    Columns Like: SUBJECT_ID, HADM_ID, NUMRX (count of rows)
11. Procedureevents_mv - Patient procedures for the subset of patients who were monitored in the ICU using the iMDSoft MetaVision system.
    Columns Like: SUBJECT_ID, HADM_ID, NUMPROCEVENTS (count of rows)
12. MICROBIOLOGYEVENTS - Microbiology measurements and sensitivities from the hospital database
    Columns Like: SUBJECT_ID, HADM_ID, NUMMICROLABEVENTS (count of rows)
13. Procedures_icd - Patient procedures, coded using the International Statistical Classification of Diseases and Related Health Problems (ICD) system
    Columns Like: SUBJECT_ID, HADM_ID, NUMPROC (count of rows)
14. Transfers - Patient movement from bed to bed within the hospital, including ICU admission and discharge
    Columns Like: SUBJECT_ID, HADM_ID, NUMTRANSFERS (count of rows)

**Final data includes**

- **Level of data:** Each instance in the final data set is one admission and is defined by 'SUBJECT_ID', 'HADM_ID'
- **Target Variables:** 'HOSPITAL_EXPIRY_FLAG' is used as target variable
- Predictor variables: 'ADMISSION_TYPE', 'MARITAL_STATUS', 'ETHNICITY', 'HOSPITAL_EXPIRE_FLAG', 'GENDER', 'NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS', 'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS', 'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS', 'NUMNOTEVENTS'

**Methodology**

1. **Load csv files:** Read the comma separated files downloaded from link (https://mimic.physionet.org/gettingstarted/overview/)
2. **Roll up tables:** We need count of various events or interactions between patients and hospital. In order to do this, group by or aggregate the tables at 'SUBJECT_ID' and 'HADM_ID' level and take count of number of rows for each. This will give total count of events related to single hospital admission.
3. **Merge tables:** Use 'SUBJECT_ID' and 'HADM_ID' as composite key to merge all tables together and create final analytical data set.

**Code**

1. Import required libraries and read csv files
2. Function to roll up tables
3. Merge all tables
4. Exploratory Analysis

**Import required libraries and read csv files**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

```
patients = pd.read_csv("data/patients.csv")
admissions = pd.read_csv("data/admissions.csv")
callout = pd.read_csv("data/callout.csv")
cptevents = pd.read_csv("data/cptevents.csv")
diagnosis = pd.read_csv("data/diagnoses_icd.csv")
outputevents = pd.read_csv("data/outputevents.csv")
rx = pd.read_csv("data/prescriptions.csv")
procevents = pd.read_csv("data/procedureevents_mv.csv")
microlabevents = pd.read_csv("data/microbiologyevents.csv")
proc = pd.read_csv("data/procedures_icd.csv")
transfers = pd.read_csv("data/transfers.csv")
inputevents = pd.read_csv("data/inputevents_cv.csv")
labevents = pd.read_csv("data/labevents.csv")
noteevents = pd.read_csv("data/noteevents.csv")
```

**Function to roll up tables**

```
def rollup_sub_adm(df,col):
    df=df.groupby(['SUBJECT_ID','HADM_ID']).agg({'ROW_ID':'count'})
    df.reset_index(inplace=True)
    df.columns=['SUBJECT_ID','HADM_ID',col]
    print(col,":",df.shape)
    return df
```

```
callout=rollup_sub_adm(callout,'NUMCALLOUT')
cptevents=rollup_sub_adm(cptevents,'NUMCPTEVENTS')
diagnosis=rollup_sub_adm(diagnosis,'NUMDIAGNOSIS')
outputevents=rollup_sub_adm(outputevents,'NUMOUTEVENTS')
rx=rollup_sub_adm(rx,'NUMRX')
procevents=rollup_sub_adm(procevents,'NUMPROCEVENTS')
microlabevents=rollup_sub_adm(microlabevents,'NUMMICROLABEVENTS')
proc=rollup_sub_adm(proc,'NUMPROC')
transfers=rollup_sub_adm(transfers,'NUMTRANSFERS')
inputevents=rollup_sub_adm(inputevents,'NUMINPUTEVENTS')
labevents=rollup_sub_adm(labevents,'NUMLABEVENTS')
noteevents=rollup_sub_adm(noteevents,'NUMNOTEVENTS')
```

```
NUMCALLOUT : (28732, 3)
NUMCPTEVENTS : (44148, 3)
NUMDIAGNOSIS : (58976, 3)
NUMOUTEVENTS : (52008, 3)
NUMRX : (50216, 3)
NUMPROCEVENTS : (21894, 3)
NUMMICROLABEVENTS : (48740, 3)
```

```
NUMPROC : (52243, 3)
NUMTRANSFERS : (58976, 3)
NUMINPUTEVENTS : (31970, 3)
NUMLABEVENTS : (58151, 3)
NUMNOTEVENTS : (58361, 3)
```

### Merge all tables

```
mortality=admissions[['SUBJECT_ID','HADM_ID','ADMISSION_TYPE','MARITAL_STATUS','ETHNICITY','HOSPITAL_EXPIRE_FLAG']]
mortality.loc[pd.isnull(mortality['MARITAL_STATUS']),'MARITAL_STATUS'] ='UNKNOWN (DEFAULT)'
mortality = mortality.merge(patients[['SUBJECT_ID','GENDER']],how='left',on='SUBJECT_ID')
mortality = mortality.merge(callout,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(cptevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(diagnosis,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(outputevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(rx,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(procevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(microlabevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(proc,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(transfers,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(inputevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(labevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(noteevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.fillna(0)
```

```
#Exporing data
mortality.to_csv('mortality_full_data.csv',index=False)
```

### Exploratory Analysis

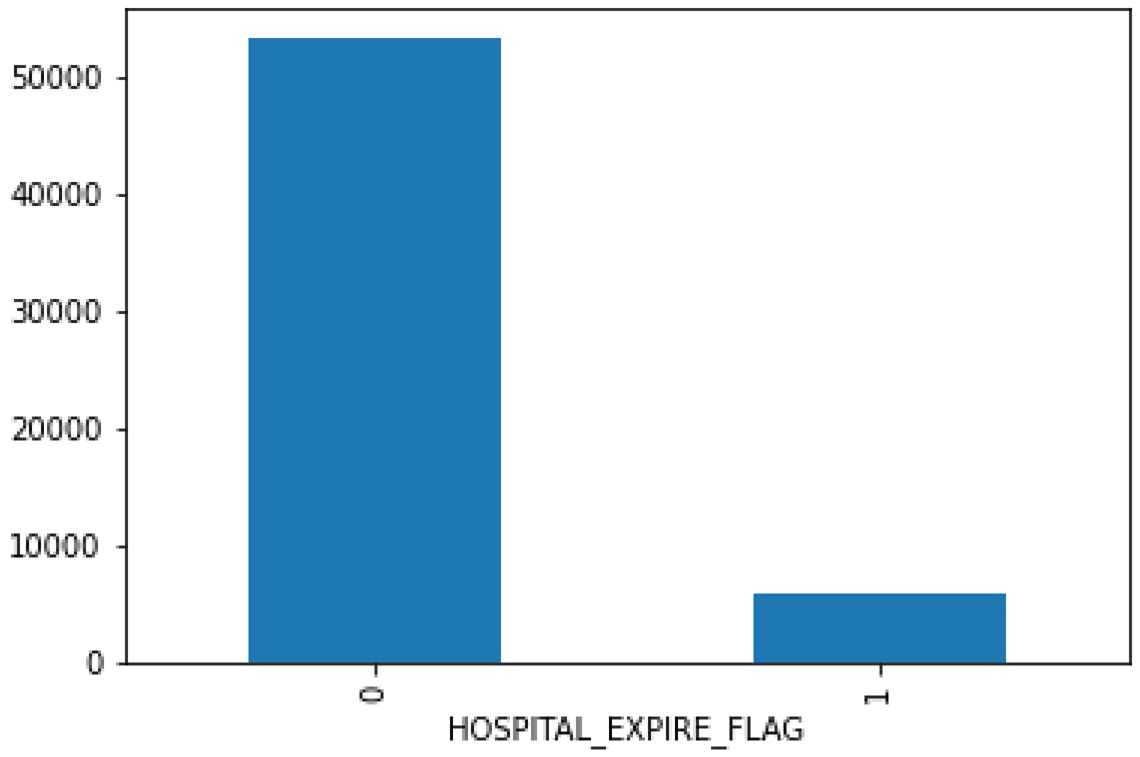
```
mortality.shape
```

```
(58976, 19)
```

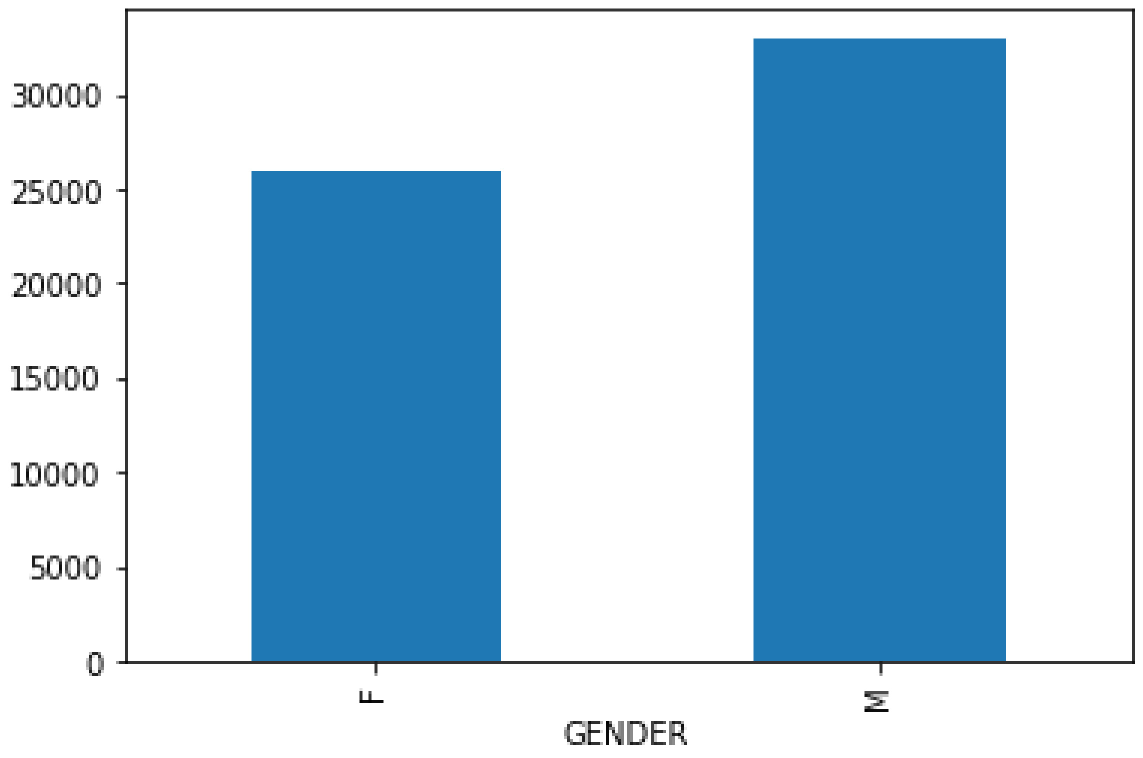
```
mortality.columns
```

```
Index(['SUBJECT_ID', 'HADM_ID', 'ADMISSION_TYPE', 'MARITAL_STATUS',
       'ETHNICITY', 'HOSPITAL_EXPIRE_FLAG', 'GENDER', 'NUMCALLOUT',
       'NUMCPTEVENTS', 'NUMDIAGNOSIS', 'NUMOUTEVENTS', 'NUMRX',
       'NUMPROCEVENTS', 'NUMMICROLABEVENTS', 'NUMPROC', 'NUMTRANSFERS',
       'NUMINPUTEVENTS', 'NUMLABEVENTS', 'NUMNOTEVENTS'],
      dtype='object')
```

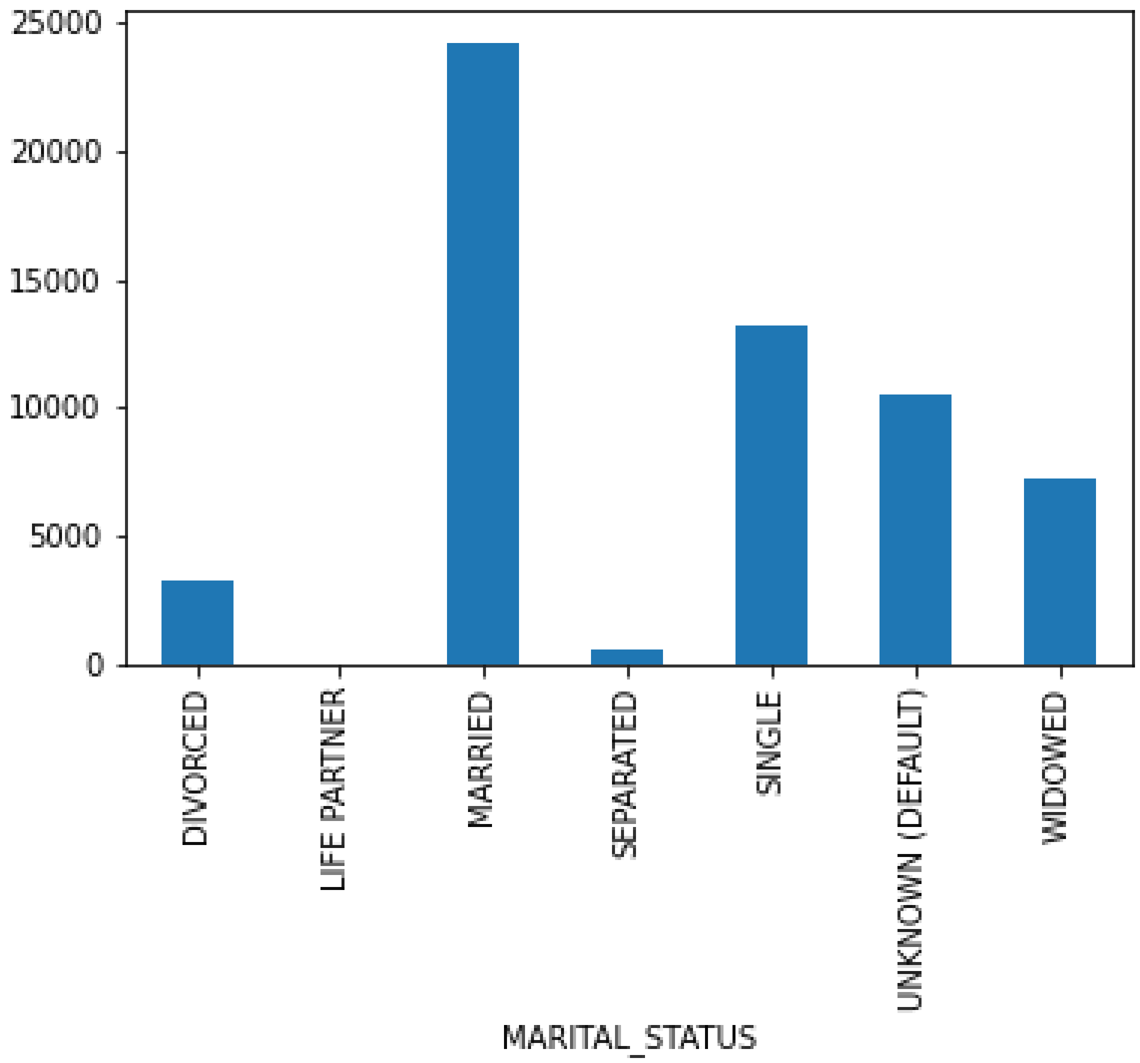
```
mortality.groupby('HOSPITAL_EXPIRE_FLAG').size().plot.bar()
plt.show()
```



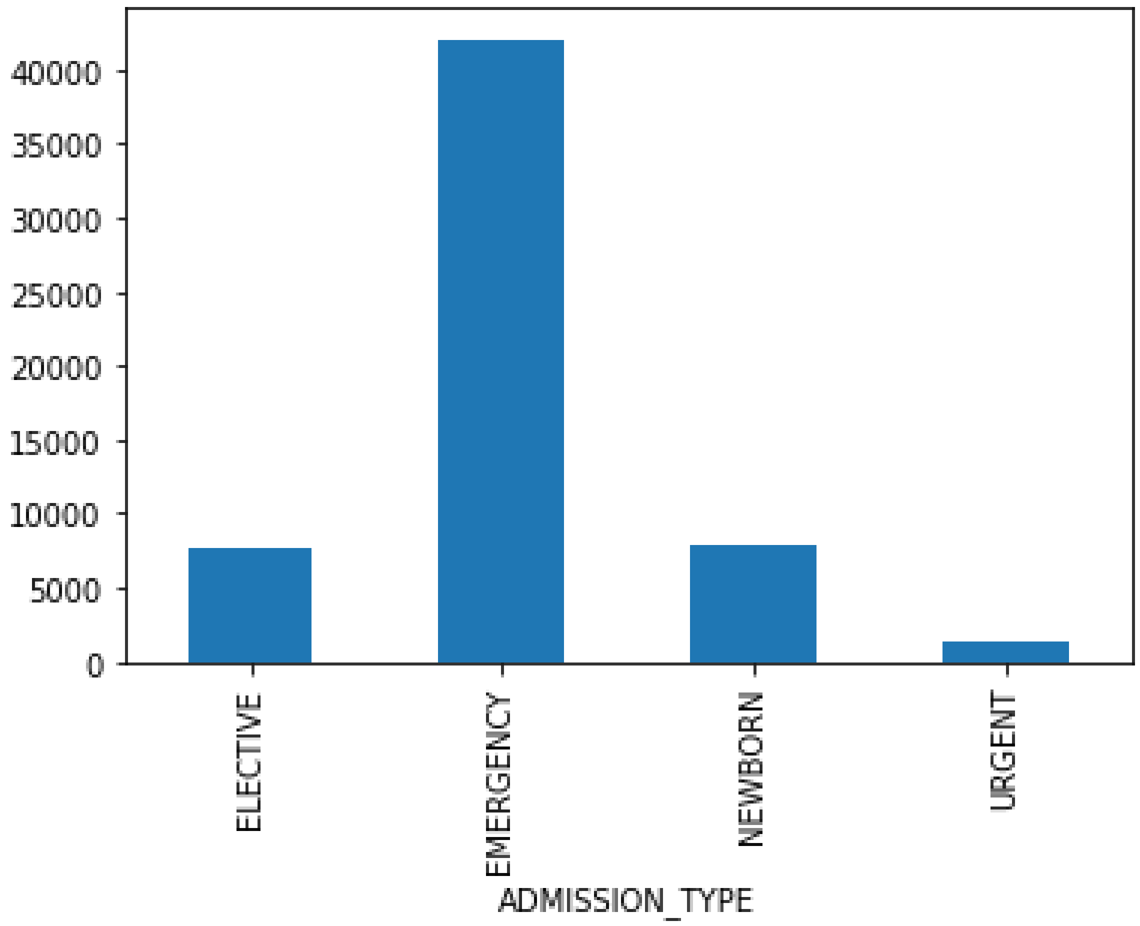
```
mortality.groupby('GENDER').size().plot.bar()
plt.show()
```



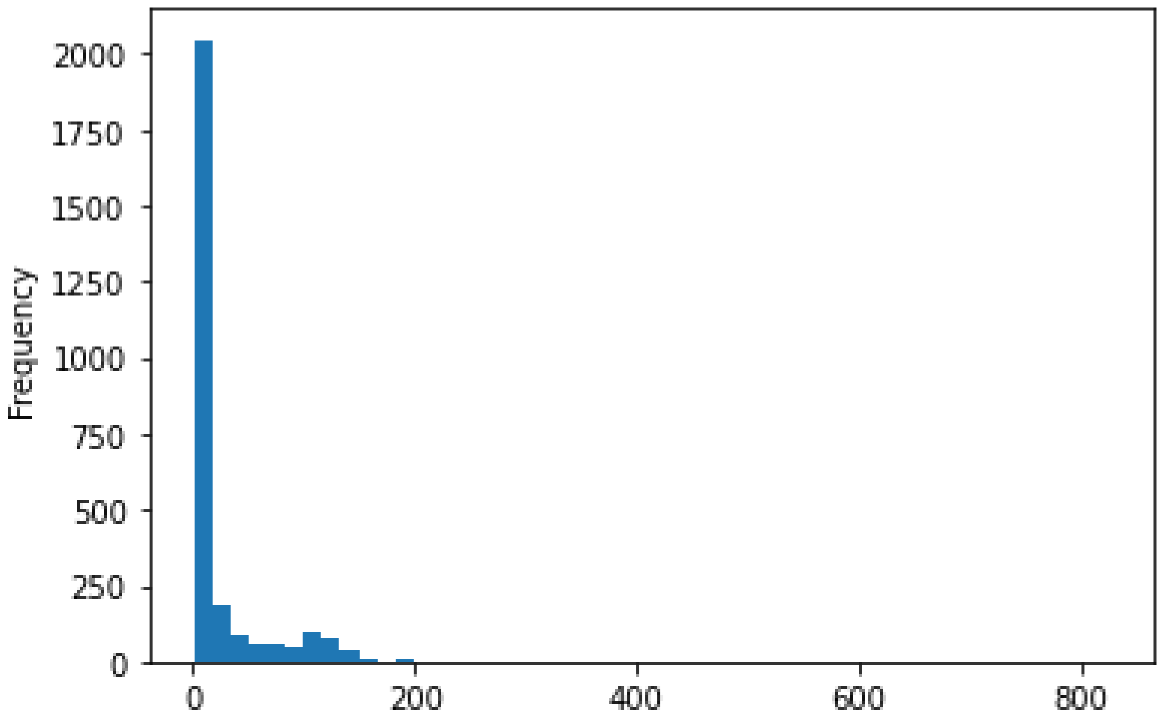
```
mortality.groupby('MARITAL_STATUS').size().plot.bar()
plt.show()
```



```
mortality.groupby('ADMISSION_TYPE').size().plot.bar()
plt.show()
```



```
mortality.groupby('NUMLABEVENTS').size().plot.hist(bins=50)
plt.show()
```



```
mortality.dtypes
```

```
SUBJECT_ID                 int64
HADM_ID                    int64
ADMISSION_TYPE            object
MARITAL_STATUS            object
ETHNICITY                 object
HOSPITAL_EXPIRE_FLAG       int64
GENDER                    object
NUMCALLOUT               float64
NUMCPTEVENTS             float64
NUMDIAGNOSIS               int64
NUMOUTEVENTS             float64
NUMRX                    float64
NUMPROCEVENTS            float64
NUMMICROLABEVENTS        float64
NUMPROC                  float64
NUMTRANSFERS               int64
NUMINPUTEVENTS           float64
NUMLABEVENTS             float64
NUMNOTEVENTS             float64
dtype: object
```

## Use Case 2: Mortality Prediction

Another use case which we implemented using MIMIC-III dataset is for mortality prediction. This use case is inspired by Kaggle kernel (reference below) where one can predict mortality just by the number of interactions between patient and hospital as predictors through count of lab tests, prescriptions, and procedures. It can be used to evaluate privacy risk with the help of PII columns like GENDER, ETHNICITY, MARITAL_STATUS and also serves as classification problem where we have to predict if patient will expire or not for a single hospital admission.

Reference: https://www.kaggle.com/drscarlat/predict-hospital-mortality-mimic3

**Tables Used**

1. Patients - SUBJECT_ID, GENDER
2. Admissions - SUBJECT_ID, HADM_ID, HOSPITAL_EXPIRE_FLAG, MARITAL_STATUS, ETHNICITY, ADMISSION_TYPE
3. CallOut - SUBJECT_ID, HADM_ID, NUMCALLOUT (count of rows)
4. CPTEvents - SUBJECT_ID, HADM_ID, NUMCPTEVENTS (count of rows)
5. Diagnosis_ICD - SUBJECT_ID, HADM_ID, NUMDIAGNOSIS (count of rows)
6. Inputevents_CV - SUBJECT_ID, HADM_ID, NUMINPUTEVENTS (count of rows)
7. Labevents - SUBJECT_ID, HADM_ID, NUMLABEVENTS (count of rows)
8. Noteevents - SUBJECT_ID, HADM_ID, NUMNOTEVENTS (count of rows)
9. Outputevents - SUBJECT_ID, HADM_ID, NUMOUTEVENTS (count of rows)
10. Prescriptions - SUBJECT_ID, HADM_ID, NUMRX (count of rows)
11. Procedureevents_mv - SUBJECT_ID, HADM_ID, NUMPROCEVENTS (count of rows)
12. microbilogyevents - SUBJECT_ID, HADM_ID, NUMMICROLABEVENTS (count of rows)
13. Procedures_icd - SUBJECT_ID, HADM_ID, NUMPROC (count of rows)
14. Transfers - SUBJECT_ID, HADM_ID, NUMTRANSFERS (count of rows)

**Final data includes**

- **Level of data:** Each instance in the final data set is one admission and is defined by 'SUBJECT_ID', 'HADM_ID'
- **Target Variables:** 'HOSPITAL_EXPIRY_FLAG' is used as target variable
- **Predictor variables:** 'ADMISSION_TYPE', 'MARITAL_STATUS', 'ETHNICITY', 'HOSPITAL_EXPIRE_FLAG', 'GENDER', 'NUMCALLOUT', 'NUMCPTEVENTS', 'NUMDIAGNOSIS', 'NUMOUTEVENTS', 'NUMRX', 'NUMPROCEVENTS', 'NUMMICROLABEVENTS', 'NUMPROC', 'NUMTRANSFERS', 'NUMINPUTEVENTS', 'NUMLABEVENTS', 'NUMNOTEVENTS'

**Methodology**

1. **Load csv files:** Read the comma seperated files donwloaded from link (https://mimic.physionet.org/gettingstarted/overview/)
2. **Roll up tables:** We need count of various events or iteractions between patients and hospital. In order to do this, group by or aggregate the tables at 'SUBJECT_ID' and 'HADM_ID' level and take count of number of rows for each. This will give total count fo events related to single hospital admission.
3. **Merge tables:** Use 'SUBJECT_ID' and 'HADM_ID' as composite key to merge all tables together and create final analytical data set.

**Code**

1. Import required libraries and read csv files
2. Function to roll up tables
3. Merge all tables
4. Exploratory Analysis

**1. Import required libraries and read csv files**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

```
patients = pd.read_csv("data/patients.csv")
admissions = pd.read_csv("data/admissions.csv")
callout = pd.read_csv("data/callout.csv")
cptevents = pd.read_csv("data/cptevents.csv")
diagnosis = pd.read_csv("data/diagnoses_icd.csv")
outputevents = pd.read_csv("data/outputevents.csv")
rx = pd.read_csv("data/prescriptions.csv")
procevents = pd.read_csv("data/procedureevents_mv.csv")
microlabevents = pd.read_csv("data/microbiologyevents.csv")
proc = pd.read_csv("data/procedures_icd.csv")
transfers = pd.read_csv("data/transfers.csv")
inputevents = pd.read_csv("data/inputevents_cv.csv")
labevents = pd.read_csv("data/labevents.csv")
noteevents = pd.read_csv("data/noteevents.csv")
```

**2. Function to roll up tables**

```
def rollup_sub_adm(df,col):
    df=df.groupby(['SUBJECT_ID','HADM_ID']).agg({'ROW_ID':'count'})
    df.reset_index(inplace=True)
    df.columns=['SUBJECT_ID','HADM_ID',col]
    print(col,":",df.shape)
    return df
```

```
callout=rollup_sub_adm(callout,'NUMCALLOUT')
cptevents=rollup_sub_adm(cptevents,'NUMCPTEVENTS')
diagnosis=rollup_sub_adm(diagnosis,'NUMDIAGNOSIS')
outputevents=rollup_sub_adm(outputevents,'NUMOUTEVENTS')
rx=rollup_sub_adm(rx,'NUMRX')
procevents=rollup_sub_adm(procevents,'NUMPROCEVENTS')
microlabevents=rollup_sub_adm(microlabevents,'NUMMICROLABEVENTS')
proc=rollup_sub_adm(proc,'NUMPROC')
transfers=rollup_sub_adm(transfers,'NUMTRANSFERS')
inputevents=rollup_sub_adm(inputevents,'NUMINPUTEVENTS')
```

```
labevents=rollup_sub_adm(labevents,'NUMLABEVENTS')
noteevents=rollup_sub_adm(noteevents,'NUMNOTEVENTS')
```

```
NUMCALLOUT : (28732, 3)
NUMCPTEVENTS : (44148, 3)
NUMDIAGNOSIS : (58976, 3)
NUMOUTEVENTS : (52008, 3)
NUMRX : (50216, 3)
NUMPROCEVENTS : (21894, 3)
NUMMICROLABEVENTS : (48740, 3)
NUMPROC : (52243, 3)
NUMTRANSFERS : (58976, 3)
NUMINPUTEVENTS : (31970, 3)
NUMLABEVENTS : (58151, 3)
NUMNOTEVENTS : (58361, 3)
```

### 3. Merge all tables

```
mortality=admissions[['SUBJECT_ID','HADM_ID','ADMISSION_TYPE','MARITAL_STATUS','ETHNICITY','HOSPITAL_EXPIRE_FLAG']]
mortality.loc[pd.isnull(mortality['MARITAL_STATUS']),'MARITAL_STATUS'] ='UNKNOWN (DEFAULT)'
mortality = mortality.merge(patients[['SUBJECT_ID','GENDER']],how='left',on='SUBJECT_ID')
mortality = mortality.merge(callout,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(cptevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(diagnosis,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(outputevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(rx,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(procevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(microlabevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(proc,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(transfers,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(inputevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(labevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.merge(noteevents,how='left',on=['SUBJECT_ID','HADM_ID'])
mortality = mortality.fillna(0)
```