Master 2 Apprentissage, Information et Contenu
Internship report

Medical applications of Generative Adversarial Networks

Author :
Adrien Pavão

Supervisor :
Isabelle Guyon

Contents

Abstract

In this report, I present how synthetic datasets, generated by machine learning models, can be used to replace sensitive medical data and therefore be able to conduct studies and teach with it. To evaluate the quality of the data generation methods, I needed metrics to compare distributions. Results show that it should be possible to use synthetic data to imitate the behaviour of real data while preserving the privacy. This was put into practice by my team by creating challenges with fresh fake data.

Keywords generative models, deep learning, metrics, privacy, challenges.

1 | Introduction

1.1 Background and motivations

Data science allowed great advancements in many application domains: image recognition, speech synthesis, credit risk analysis and many more. Medical research is a promising field for machine learning that could greatly benefit from these advancements. Medical record analysis is **human subject research**, which means that this scientific investigation involves human beings as research subjects. According to the MIT course *“Data or specimens only research”*[1], three important conditions must be met to conduct this type of research: **respect for persons**, **beneficence** and **justice**. Indeed, dealing with confidential information calls for utmost caution.

A solution to this problem could be to use machine learning to **generate** new data distributions similar to the original datasets, but without the possibility of a privacy leakage.

The goal of our project, *Medi-Chal*, is therefore to study generative models in order to create data **challenges** on medical problems with synthetic data, mainly for teaching purposes. Challenges are motivating, attract people and are known to have regularly led to think of new methods surpassing the state-of-the-art.

We mainly aim to tackle those problems:

- Evaluation of generative models by **comparing original and generated distributions**. Indeed, data generation naturally raises the non-trivial problem of comparing distributions. This is tricky and, so far, most comparison evaluations rely on human expertise such as automatic joke writing [1] or automatic musical composition [2]. We therefore want to find numerical metrics to measure the resemblance between distributions but also the respect of confidentiality.

- **Generate data**. We want the synthetic dataset to follow the same distribution as the original but without containing the exact same values. The generative model must be able to extrapolate values, have a kind of “creativity”, avoid the mode collapse (lack of diversity of samples) and avoid outliers or monsters (samples radically far from the original distribution).

We wrote programs that help to visualize and process data, compare distributions and generate new datasets easily. This report contains many example screen-shots from the programs and our work is available on GitHub[2].

[1] `https://www.nyu.edu/research/resources-and-support-offices/getting-started-withyourresearch/human-subjects-research/tutorial.html`

[2] `https://github.com/Didayolo/medi-chal`

1.2 Meet the Medi-Chal Team

This work has been done by a team from March to August 2018. The *Medi-Chal* team is led in supervision by Dr. Guyon, Professor at Université Paris-Sud in Paris and Dr. Bennett, Professor at Rensselaer Polytechnic Institute in New-York.

The team is composed of the following members:

- Isabelle Guyon, Professor at UPSud, Paris.
- Kristin Bennett, Professor at RPI, New-York.
- Andrew Yale, PhD student at RPI, New-York.
- Adrien Pavão, Master student at UPSud, Paris.
- Thomas Gerspacher, Master student at UGA, Grenoble.
- Saloni Dash, Bachelor student at Birla Tech, India.
- Ritik Dutta, Bachelor student at IIT Gandhinagar, India.

My role was to coordinate the *Medi-Chal* project and to supervise Ritik and Saloni's part-time internships. We had weekly teleconference meetings to discuss the project advancements, present white papers and assign work. We also organized two workshops at the Laboratoire de Recherche en Informatique (LRI) where we presented our research. Saloni and Ritik managed to be present in France for their presentations. Slides are available here[3][4].

In this report, I present the work of the team as well as my own contributions.

I have chosen to write this report in English to be consistent with the English working environment and to improve my expression skills.

[3] https://github.com/Didayolo/medi-chal/tree/master/documents/workshop_slides_1
[4] https://github.com/Didayolo/medi-chal/tree/master/documents/workshop_slides_2

2 | Data

2.1 Datasets

The medical data we work with are **patient records**: medications, hospital entrance and exit dates, health status and so on. The features are mainly binary values, count (integers) and **categorical variables**.

Categorical variables raise the issue of encoding in order to reduce them to numerical variables. We have evaluated the qualities and deficiencies of the different encoding methods we found in the literature. Indeed, smartly encoding categorical data could greatly improve our results and this type of data is highly represented in medical reports.

We systematically convert the data to the *AutoML*[1] format (cf. 2.2) which allows us to extract meta-information about the dataset. We had developed the software behind this format to make it easier to use and add new options such as automated data *pre*-processing.

We have always tested our methods on different datasets to check the generalization and robustness. Here are some examples of such datasets:

- **MIMIC**[2]. The main dataset of the project. MIMIC III is an openly available dataset of electronic health records of about $40,000$ critical care patients. This rich dataset lets us work on our methods with the entire group since everyone can get access to the data.
- **OptumLabs**[3]. Optum data is many orders of magnitude larger than the MIMIC data with over $200,000,000$ patients. This dataset is controlled and not publicly available so the access is much more restricted.
- **Classical datasets** of various size, variable types and application domains such as Iris, Mushrooms and many others.
- **Artificial** data. Data synthesis is useful to compare original and generated data since it is a well known environment (sort of "unit testing"). For instance, we could perfectly control the rules that created rows or the diversity between points.

  Note: Unlike data generation from the estimation of a statistical distribution, data synthesis here refers to the creation of data from a set of rules, for example pictures of predefined geometric shapes.
- **Semi-artificial** data: Modified real data.

Most of the data we used are stored in this repository[4]. We can't, of course, make it public because it contains protected datasets.

2.2 AutoML format

We worked on a data format called *AutoML*[1]. The format is defined by several files containing raw data, column names, train/test/validation split, target variable name(s), variable types (binary, categorical, numerical) and meta-information such as name, problem type (classification, regression), size, etc. Our program automatically computes the processing and data visualization tools presented below.

[1]`http://automl.chalearn.org/`
[2]`https://mimic.physionet.org/`
[3]`https://www.optumlabs.com/`
[4]`http://gitlab.com/didayolo/data`

2.3 Pre-processings

Normalization

Normalizing the data is done so that all the input variables have the same treatment in the model so its parameters are not scaled with respect to the units of the inputs. Furthermore, some machine learning algorithms require the input data to be normalized in order to converge to good results.

The program can hold the following normalizations:

- Standard normalization: $x_i = \frac{x_i - mean(x)}{std(x)}$
- Min-max normalization: $x_i = \frac{x_i - min(x)}{max(x)}$

and then apply the normalization with the same parameters to test set.

Missing values imputation

The program also handles several methods for missing values imputation:

- Discard row.
- Replace by:
  - Most frequent value.
  - Median.
  - Mean.

Categorical variable encodings

A categorical variable is a variable that can take one of a limited, and usually fixed, number of possible values assigning each individual to a particular category on the basis of some qualitative property. Medical records are full of it (e.g. skin color). The encoding procedure means to convert categories into numerical values.

Categorical variables raise some issues:

- It can be tricky to deal with **high cardinality** of categories.
- Algorithms mainly take **numerical** variables as input.
- A **smart encoding** is needed to make the learning easier because the machine does not know the semantic information hidden in the categories (e.g. "potato" is closer to "carrot" than "strawberry").

AutoML program includes multiple methods for categorical variable encoding:

- **Label encoding:** Categories are arbitrarily replaced by ordinal values (from 0 to the number of categories).
- **Frequency encoding:** Categories are replaced by their number of occurrences. It is quiet similar to label encoding but the values have more signification, in other words they contains information about the data.
- **One-hot encoding:** Each category value is turned into a binary vector where all columns are equal to 0 besides the category column equal to 1. Dimensionality is increased according to the number unique categories.
- **Rare version** of one-hot encoding.
  1. **If** *occurence* $<$ *average_occurence* $\times$ *rarity_coefficient*:
  2. **Then** the category is replaced by "RARE".
- **Feature hashing:** [3]
  1. Fix vector size.

2. For every categorical feature: use a hash function to map values to indices of the feature vector (and one-hot the indice).
3. Sum all vectors into one.

- **Target encoding:** A numerical variable is defined as the target (typically the class). Each category is replaced by its mean target value.
- **Likelihood encoding:** Target encoding on the first principal component of the continuous variables

I have conducted an experiment on Adult income dataset [5] to evaluate those encodings. This dataset contains 15 variables including 9 categorical variables.

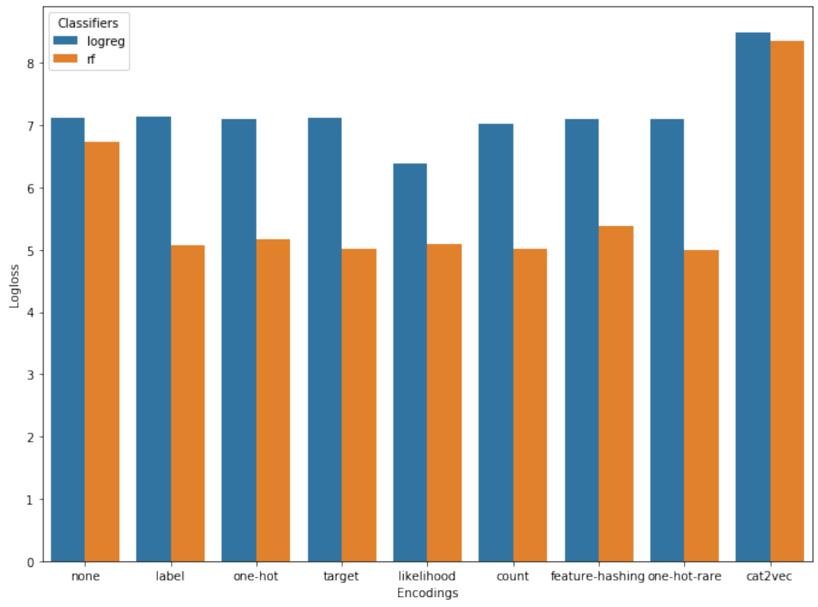


Figure 2.1: Classification accuracy scores obtained on Adult dataset by Random Forest and Logistic Regression for different categorical variable encoding methods.

The performance differences are not really representative in this dataset. However, the influence of the encoding choice is not null. It is a kind of proof-of-concept that encoding with information about data can improve results. On bigger datasets with more samples and more dimensions "smart" encodings could have greater influence on performance.

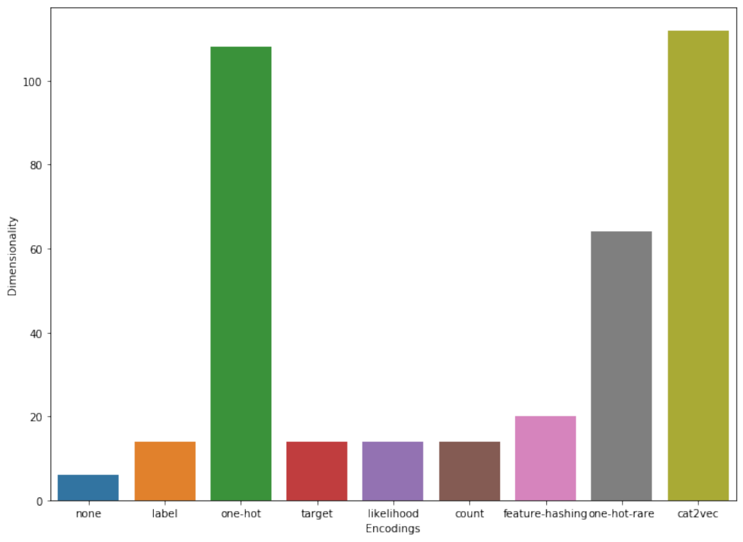


Figure 2.2: Number of features depending on encoding.

[5]https://archive.ics.uci.edu/ml/datasets/adult

Some interesting encodings relying on deep learning are not implemented in the program but are worth a look. I can cite *deep embedding* [4], *Cat2Vec* [5] and *autoencoders*. Typically by modeling dependencies between variables and compressing data in a lower dimensionality latent space, those encodings can give a wise representation of data and improve algorithm's training.

Synthetic Data Vault

Synthetic data vault is a method to build a system that automatically creates synthetic data to enable data science endeavors. It is presented in the following paper [6]. The authors have outlined methods for converting columns based on whether there are missing values and based on the type, whether it is numeric or categorical. Once the data is in a common format each **column is modeled by either a uniform or normal distribution** and produces a distribution and covariance that are stored.

The stored values make it possible to reverse the encoding (to decode) data. It restores categories and create realistic synthetic data.

Copula trick

The copula trick consists in making marginals uniform by **replacing variable by their rank**. It can be used either as *pre*-processing or *post*-processing. Copulas will be defined later in the *Generators* section (cf. 4.4).

2.4 Dataset characterization

To prepare analysis properly it is important to have a good understanding of data by identifying missing values, sparsity, variable types, dependencies between variables and other meta-features. In this section we are going to review various techniques for dataset characterization. The dataset characterizer is available as a notebook here[6].

Meta-features

We grouped meta-features as proposed in papers [7] [8].

- **Simple Measurements** such as the number of classes or the number of attributes are directly and easily accessible properties of the dataset which need almost no computation.
- **Statistical features:** The use of statistical analysis methods and tests [9, 10].
- **Discriminant Measurements:** Computing a discriminant analysis leading to various measurements.
- **Information-theoretic features:** Typically the use of entropy measures of the attributes and the class label [11].
- **Model-based features:** We create a model of the data, e.g. a decision tree, and use its properties, e.g. the width and height of the tree, as features (e.g. [12] used 17 properties of a decision tree).
- **Landmarking features:** We apply computationally fast classifiers (e.g. Naive Bayes) on the dataset [13, 14] and use the resulting performance as meta-features.
- **Time-based features** Contain time measures of several computations regarding the dataset, for instance the time for computing the other meta-features. [15] present 9 different time-measurements.

[6]`https://github.com/Didayolo/medi-chal/blob/master/notebooks/auto_ml/ID_notebook.ipynb`

Visualization

Some plots help to visualize data in a human-friendly way, such as univariate and bivariate distribution of variables. In the high-dimensionality case, we could let the user choose variables and by default plot the most relevant: the most important variables according to an algorithm or the most correlated pair of variables.

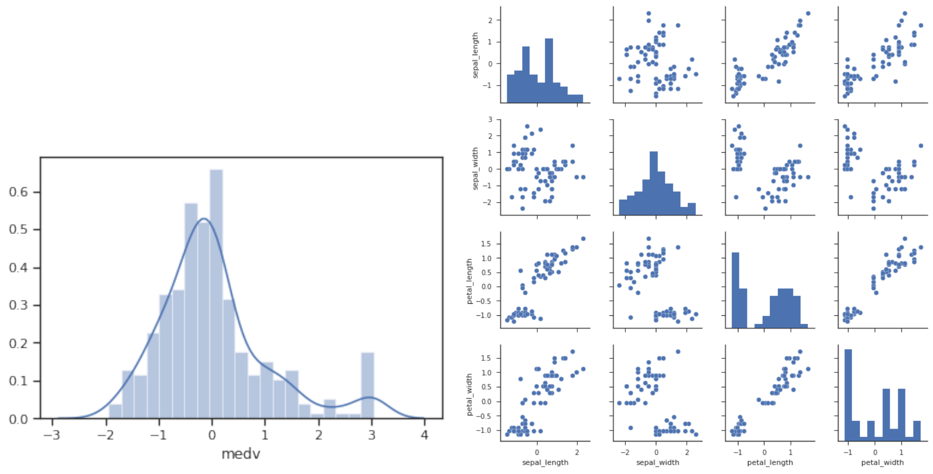


Figure 2.3: Examples univariate and bivariate distribution plots.

Visualization tools can also be features heat-map, correlation matrix, hierarchical clustering or dimensionality reduction to plot data in two dimensions: principal components analysis (PCA), linear discriminant analysis (LDA), t-distributed stochastic neighbor embedding (T-SNE).

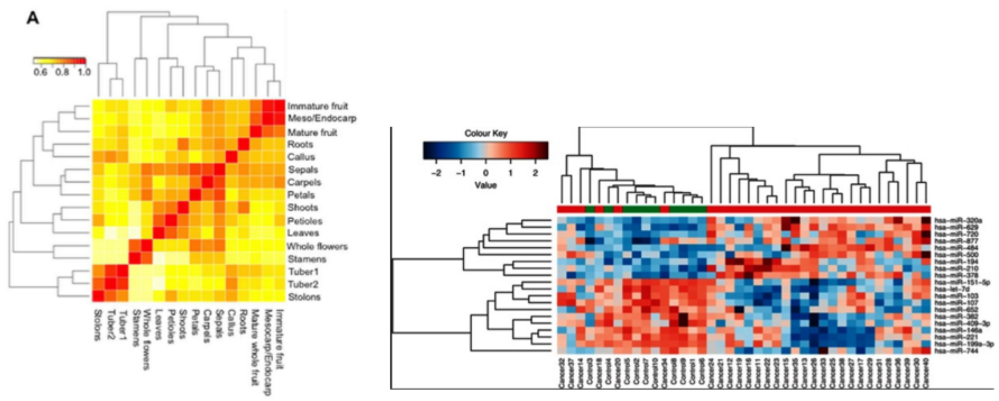


Figure 2.4: Examples of hierarchical clustering on correlation matrix (left) and heat-map (right)

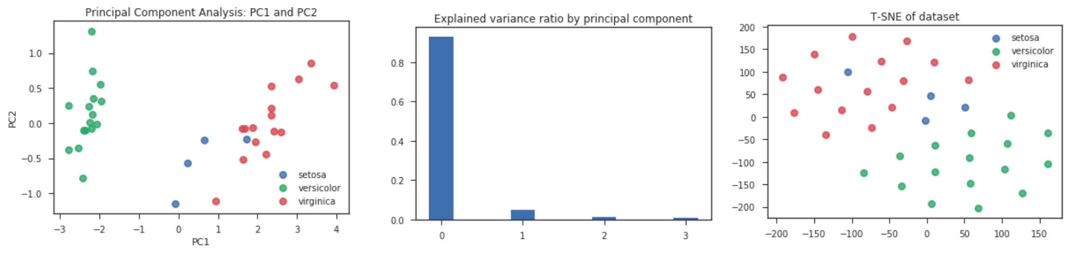


Figure 2.5: Examples of PCA (left) and its explanation ratio (center) and T-SNE (right)

3 | Datasets comparison

Datasets comparison is where we check if a synthetic data **resemble** the original data and if the **privacy** is preserved. Evaluating similarity between distributions is a tricky problem. There is no definitive score or metric that could do the job. However we have studied and implemented some metrics which could help in quality checks of the generated datasets. Using the results of all the metrics, one might get an approximate measure of the degree of similarity of one dataset to another one, relative to all the other datasets. The comparator is available as a notebook here[1].

We consider three level of utility:

1. **Educational:** If the purpose is simply educational then the resemblance may be rudimentary.
2. **Benchmark:** At this utility level, we want to have the ability to obtain the same results on given tasks. In other words, the relative performance of algorithms must be the same (e.g. Random Forest being better than Logistic Regression).
3. **Discovery:** The most demanding level is to be able to use the artificial data to make scientific discoveries. A possible test is to train a model to predict a disease on generated training data and test it on real data.

Note that Ritik, my intern, helped me a lot with the review of comparison metrics. There are various approaches:

Overall meta-features:

Simple distances between the descriptors of each dataset (e.g. number of samples, sparsity).

Individual variables

One-dimensional statistical test can be used to compare individual features together. A positive measure on two objects, which is null when the objects are equals, is called a **divergence**. If a divergence measure is symmetric and has the triangular equality property then it is a **distance** (or metric).

We can cite the Wasserstein distance, the Jensen Shannon divergence and distance, the Kullback-Leibler divergence or the mutual information score. For categorical variables there is also the chi-square test.

Principal component analysis

Principal component analysis (PCA) [16] provides an interesting data visualization. By taking the two first components we reduce the dimensionality in order to visualize data in two dimensions and have a good intuition of it.

[1]`https://github.com/Didayolo/medi-chal/blob/master/notebooks/auto_ml/comparison_notebook.ipynb`

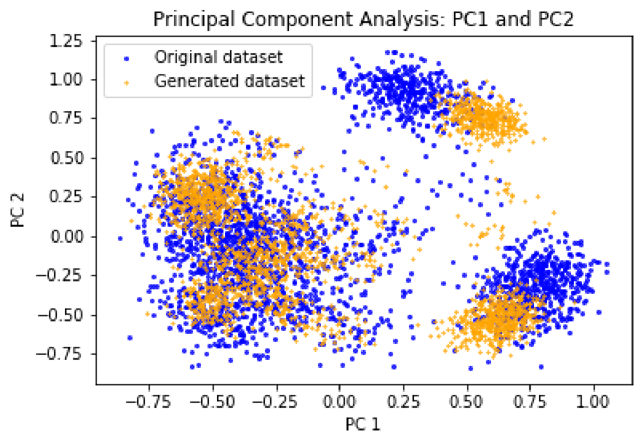


Figure 3.1: PCA plot of mushrooms dataset. Original in blue, generated with Random Forest imputations in orange.

As you can see, we overlay the scatter plot of the real data test set and generated data.

Minimum distance accumulation

The goal of this metric is to compute a value for resemblance as well as for respect of privacy regarding an arbitrary distance threshold. The computation goes like this:

1. For each point: compute the distance of its nearest neighbor from the other distribution.
2. Compute the following graph: a distance θ on x axis and the number of points with a minimum distance smaller than θ on y axis.
3. Define a threshold distance for privacy/resemblance. Under the threshold, we want points to be as less as possible similar to respect privacy. Over the threshold, we want points to be as more as possible similar to maximize resemblance. The metrics are the top left and bottom right **areas drawn by the curve and the threshold**.

Distances and accumulation are normalized to have more robust metrics. The areas are normalized with the threshold to be within 0 and 1.

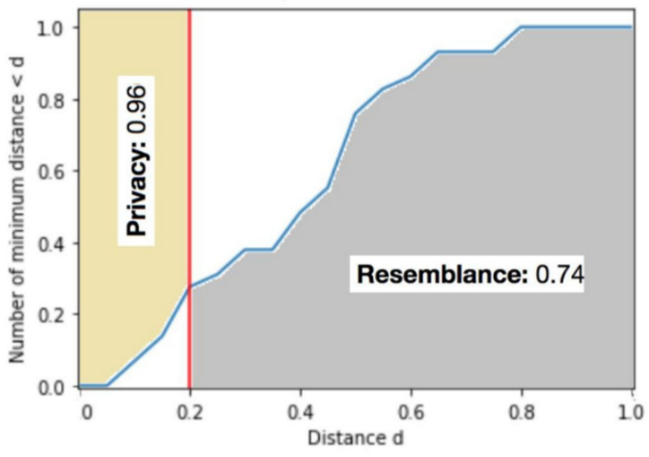


Figure 3.2: Minimum distance accumulation curve example with *threshold* = 0.2 and areas highlighted.

With this tool we can observe the privacy/resemblance trade-off. By adding noise on the generated data we increase privacy value while decreasing resemblance value. The arbitrary choice of the threshold has influence on the results. We can avoid this influence by computing the variation of the resemblance and privacy values with a varying threshold.

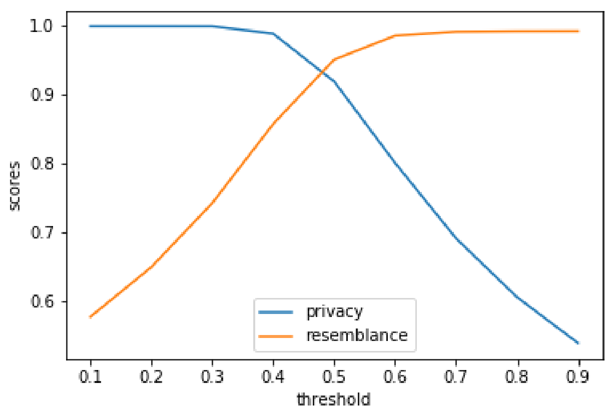


Figure 3.3: Illustration of privacy and resemblance values for different threshold.

Marginals plots

We compare marginal distributions of the datasets by computing a metric on each variable and comparing values for the two datasets. Some possible metrics:

- Mean,
- Minimum,
- Maximum,
- Standard deviation,
- Correlation with target.

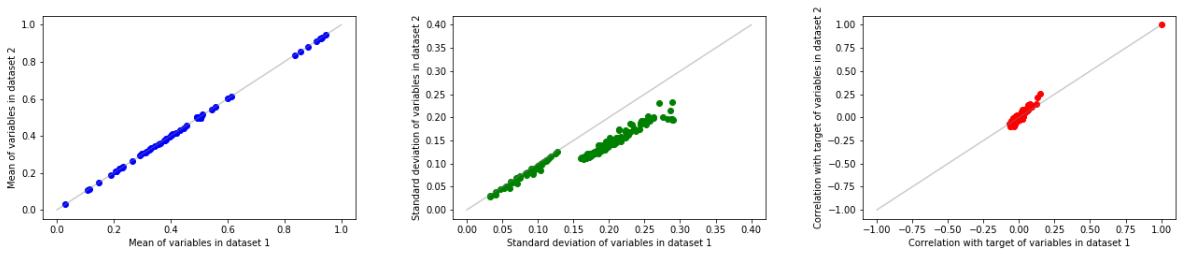


Figure 3.4: Marginals plots for MIMIC dataset generated by Random Forest imputations. Metrics are from left to right: mean, standard deviation, correlation with the class.

If the marginal distributions are similar then all the points appear on the grey line which represents the linear equation $y = x$. We can of course group them in the same plot for a more compact visualization.

Performance on task (landmark)

For datasets with a supervised learning task, we can compare classification or regression scores. There are three experiences to compute:

1. Train on real data then test on real data,
2. Train on real data then test on synthetic data,
3. Train on synthetic data and test on synthetic data.

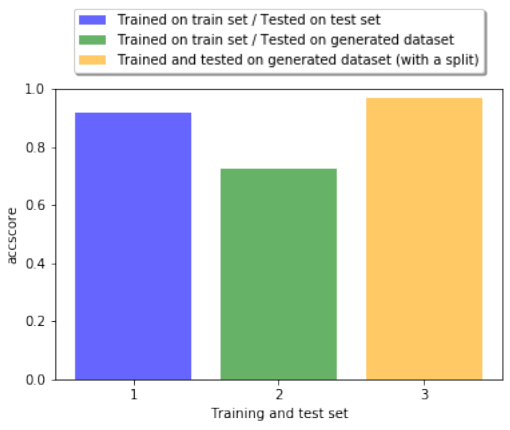


Figure 3.5: Accuracy scores for MIMIC dataset original and generated by kernel density estimation. In this example, the model is a Random Forest with 50 estimators.

Discriminant score

A **classifier** is trained to output the probability of a given sample being **real or fake**. The metric then consists in the performance of the classifier. If it can tell the source of samples, then the distributions are different. More precisly, the more sophisticated the classifier needs to be to succeed in separating the data, the more similar the datasets are. If the classifier can't separate the data, the datasets are either too similar or the classifier isn't good enough (not enough data, bad model choice, etc.).

This idea is close to the method used to train Generative Adversarial Networks (GANs), but we'll come back to that later.

```
                  precision    recall

 Original dataset      0.87      0.04
Generated dataset      0.68      1.00

      avg / total      0.74      0.68
```

Figure 3.6: Discriminant score for boston dataset generated by multivariate gaussian. The classifier is a multilayer perceptron with layers size [100, 200], ReLU activation function and a constant 0.001 learning rate.

The best possible score is 0.5 while the worst is 1.

Maximum Mean Discrepancy

Maximum Mean Discrepancy [17] (MMD) is a kernel-based approach aimed at measuring the distance between two probability distributions. This test consists in drawing samples from each of them, finding a smooth function whose value is large for points belonging to one of the classes, while its value is small for points belonging to the other class. The difference between the mean function values on the two samples is computed. If this value is large, the samples are likely from different distributions. The value should be zero if the two distributions are equal.

It can be used as a function to optimize in GAN training as in [18]. Gao Huang et al. in [19] recommend MMD as reliable metrics for evaluating GANs.

Autoencoder

A possible divergence metric could be to train an autoencoder on a dataset and evaluate its retrieval ability on the other dataset (and reciprocally). The intuition behind this is that similar data will be compressible in the same latent space.

Remark: we didn't implement this idea.

Privacy metrics

Privacy is one of our main concern. The primary goal of **data anonymization** in the context of data publishing is to ensure that no record in the data can exclusively identify an individual - this concept is called Unique Identity Disclosure (UID).

In general privacy models can be categorized as follows:

- **Generalization:** Replacement of a value for a more general one. For instance, a zip code of *12345* can be replaced by *123*** or a Profession of Actor can be replaced by Artist.
- **Suppression:** Removal of some attribute values to prevent information disclosure.
- **Anatomization:** Disassociate Quasi-Identifiers (QIDs) and sensitive attributes in two separate tables.
- **Perturbation:** Replacement of the original data for synthetic values with identical statistical information. This is the kind of privacy model involved when generating synthetic data using **GANs**. When generating records, we face a **trade-off** between resemblance and privacy.

Here is a review of **privacy level metrics**. This work is essentially the result of a review and synthesis carried out by my intern Saloni.

- **Confidence Level:** [20] One of the first proposed metrics to measure data privacy is the confidence level This metric is used in additive-noise-based randomization techniques, and measures how well the original values may be estimated from the randomized data. If an original value may be estimated to lie in an interval $[x_1, x_2]$ with c% confidence, then the amount of privacy of this interval is the at c% confidence. The problem with this metric is that it does not take into account the distribution of the original data, therefore making it possible to localize the original distribution in a smaller interval than $[x_1, x_2]$, with the same c% confidence.
- k**-anonymity:** [21, 22] If the identifiable attributes of any database record are indistinguishable from at least other k-1 records, then the dataset is said to be k-anonymous. In other words, with a k-anonymized dataset, an attacker could not identify a single record since other k-1 similar records exist. *k is selected a priori, hence offers control over privacy of data.*
- l**-diversity:** This model expands the k-anonymity model by requiring every equivalence class to abide by the l-diversity principle. An l-diverse equivalence class is a set of entries such that at least l "well-represented" values exist for the sensitive attributes. The sensitive attributes are "well-represented" if there are at least l distinct values in an equivalence class, what is known as distinct l-diversity. *l is selected a priori, hence offers control over privacy of data.*
- t**-closeness:** [23] In order to prevent attribute disclosure from the distribution skewness (skewness attacks), the authors presented the t-closeness privacy model. This model requires the distribution of the sensitive values in each equivalence class to be "close" to the corresponding distribution in the original table, where close is upper bounded by the threshold t. That is the distance between the distribution of a sensitive attribute in the original table and the distribution of the same attribute in any equivalence class is less or equal to t; this is known as the t-closeness principle.

  The t-closeness principle also has various instantiations depending on the distance function that is used to measure the closeness. The three most common functions are the variational distance, the Kullback-Leibler (KL) distance and the Earth Mover's distance (EMD). *t is selected a priori, hence offers control over privacy of data.*
- ϵ **differential privacy:** The notion of differential privacy [24] measure the difference on individual privacy disclosure between the presence and the absence of the individual's record. The ϵ-differential privacy model ensures that a single record does not considerably affect the outcome of the analysis over the dataset. In this sense, a person's privacy will not be affected by participating in the data collection since it will not make much difference in the final outcome. *ϵ is selected a priori, hence offers control over privacy of data.*
- **Average Conditional Entropy:** [25] It is based on the concept of information entropy. Given two random variables X and Z the average conditional privacy of X given Z is $H(X|Z) = 2^h(X|Z)$, where $h(X|Z)$ is the conditional differential entropy of X, defined as:

  $h(X|Z) = -\int f_{X,Z}(x,z) log_2 f_{X|Z=z}(x) dxdz$

  Where $f_X(.)$ and $f_Z(.)$ are the density functions of X and Z respectively.

- **Variance:** [26] In multiplicative noise randomization, privacy may be measured using the variance between the original and the perturbed data. Let x be a single original attribute value, and z the respective distorted value, $\frac{Var(x-z)}{Var(x)}$ expresses how closely one can estimate the original values, using the perturbed data.
- **Hidden Failure:** [27] is used to measure the balance between privacy and knowledge discovery. The hidden failure may be defined as the ratio between the sensitive patterns that were hidden with the privacy-preserving method, and the sensitive patterns found in the original data. More formally:

  $HF = \frac{\#R_P(D')}{\#R_P(D)}$

  Where HF is the hidden failure, D' and D are the sanitized dataset and the original dataset, respectively, and $\#R_P(.)$ number of sensitive patterns.

  If $HF = 0$, all sensitive patterns are successfully hidden, however, it is possible that more non-sensitive information will be lost in the way.

4 | Generators

Now, let's dive into different generative models for the actual creation of synthetic dataset. According to [6], synthetic data must meet two requirements:

- It must somewhat resemble the original data **statistically**, to ensure realism and keep problems engaging for data scientists.
- It must formally and **structurally** resemble the original data, so that any software written on top of it can be reused.

We studied the following generative models.

4.1 Multivariate Gaussian

Data are modeled by a *Gaussian Mixture Model* (GMM) probability distribution whose parameters are estimated. We can then generate random samples from the fitted gaussian distribution. We used its simplest version as a baseline: a single multivariate gaussian distribution.

4.2 Parzen Window Density Estimation

Parzen window density estimation, or kernel density estimation, is a non-parametric method for estimating continuous density function from the data.

The main idea is that you approximate the distribution by a mixture of continuous distributions K (using your notation ϕ), called kernels, that are centered at datapoints and have scale (bandwidth) equal to h:

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

With x_1, ..., x_i the data points. This is illustrated on the picture below, where normal distribution is used as kernel K and a bandwidth h is used to estimate distribution given the data points. The densities are kernels centered at x_i points. Notice that h is a relative parameter, its value is always chosen depending on your data and the same value of h may not give similar results for different datasets.

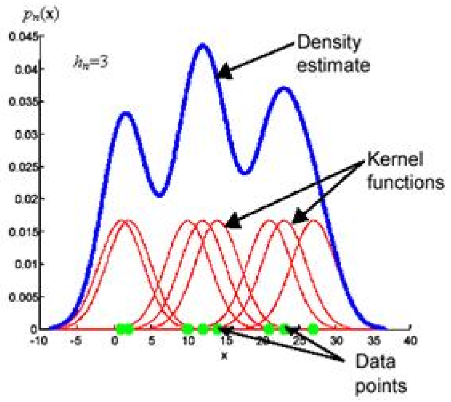


Figure 4.1: Parzen window density estimation scheme

4.3 Random Forest Imputations

A *Random Forest* is a homogeneous ensemble machine learning method. Ensemble methods are a kind of meta-algorithms that combine several techniques in order to decrease bias, variance and improve predictions [28].

In random forests, each tree in the ensemble is built from a sample drawn with replacement from the training set. In addition, instead of using all the features, a random subset of features is selected, further randomizing the tree. As a result, the bias of the forest increases slightly, but due to the averaging of less correlated trees, its variance decreases, resulting in an overall better model.

Suppose we want to **impute** missing values in Y using $X_1, \ldots, X_p$. Let y^{obs} and y^{mis} denote the observed and missing values in Y, and let x^{obs} denote the predictor values corresponding to y^{obs}. The principle is as follows:

- Grow a forest for predicting Y from $X_1, \ldots, X_p$, using y^{obs} and x^{obs},
- For a subject who is missing Y, find their terminal node based on their values of $X_1, \ldots, X_p$,
- Impute the missing Y using a random sample from the observed Ys in the given terminal node [29].

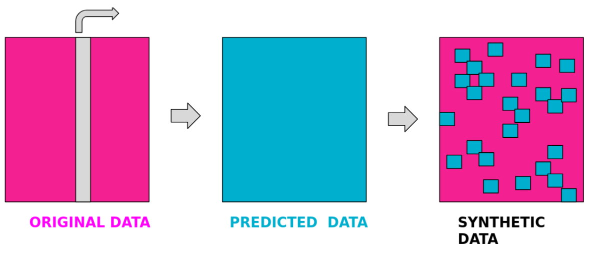


Figure 4.2: Random Forest Imputations generator scheme

Instead of copying the whole original table - and then randomly replace values - we can sample original data with replacement to generate n examples. Note that if the replacement probability $p = 1$, as Random Forest classifier and regressor are deterministic models, a same sample will each time be imputed in the same way. It is therefore impossible to draw more points than the size of the original dataset if $p = 1$.

Random Forest Imputations is a good idea for a baseline method because:

- It does not extrapolate values: it will only predict values within the range of original data,
- It computes fast,
- We can choose with the p parameter the degree of resemblance with the original data; $p = 0$ for exact same dataset, $p = 1$ for only new values.

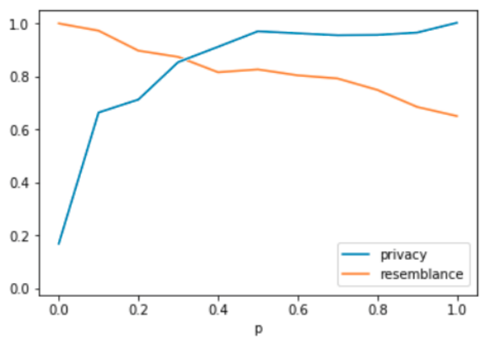


Figure 4.3: Influence of p parameter on resemblance and privacy (MDA metric with a fix threshold, cf. section 3) on Iris dataset.

4.4 Copulas

Copulas are tools for modeling **dependencies** of several random variables. The basic idea is to apply polynomial transformations on normally distributed random variables.

Steps involved in generating data are:

1. Converting the marginals to uniform distributions,
2. Calculating the inverse gaussian Cumulative Distribution Function (CDF) using the uniform distributions,
3. Performing Kernel Density Estimation (KDE) to approximate the probability density function,
4. Sampling from the KDE.

4.5 Generative Adversarial Networks

Generative Adversarial Networks [30] (GAN) is a framework for generative network training for both semi-supervised and unsupervised learning. It consists of two models:

- A **discriminator D** estimates the probability of a given sample coming from the real dataset. It works as a critic and is optimized to tell the fake samples from the real ones.
- A **generator G** outputs synthetic samples given a noise variable input z brings in potential output diversity. It is trained to capture the real data distribution so that its generative samples can be as real as possible, or in other words, can trick the discriminator to offer a high probability.

The whole idea of GANs is to solve the minimax objective between the two networks:

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)} \ln D(x) + E_{x \sim p_z(x)} \ln(1 - D(G(z))) \tag{4.1}$$

Where G and D represent the networks and can be seen as players.

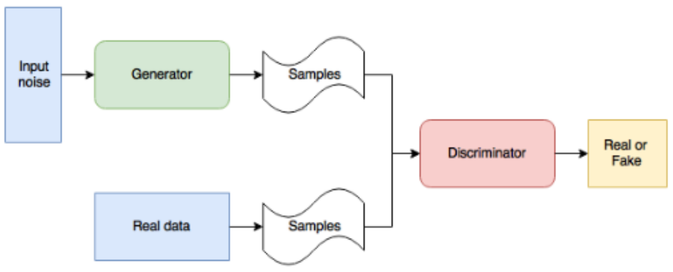


Figure 4.4: GAN Architecture

The game plays out in two scenarios.

1. Training examples x are randomly sampled from the training set and used as input for the discriminator D.
2. The discriminator receives input $G(z)$, a fake sample created by the generator.

In the first scenario, the aim of the discriminator is to output the probability that its input is real rather than fake. The discriminator strives to make $D(x)$ approach 1.

In the second scenario, both discriminator and generator participate. The discriminator strives to make $D(G(z))$ approach 0 while the generator strives to make the same quantity approach 1.

The training process consists in training simultaneously both gradient descent. Unfortunately, there are multiple problems that arise with GANs such as low dimensional supports, vanishing gradients and mode collapse.

Wasserstein GAN

Wasserstein GAN [31] (WGAN) is a derived version of the original GAN model that tries to overcome some of its shortcomings. The main idea behind WGAN is to **replace the loss function by a metric called Wasserstein distance** (or Earth mover's distance) which is a smooth measure more helpful for a stable learning process using gradient descents.

The mini-max objective can thus be rewritten the following way:

$$\min_{G}\max_{D\in\mathcal{D}} V(G,D) = E_{x\sim p_{data}(x)}D(x) - E_{x\sim p_z(x)}D(G(z)) \tag{4.2}$$

Where $\mathcal{D}$ is the set 1-Lipschitz functions.

In order to ensure the lipshitz constraint, weights need to be clipped between boundaries. Unfortunately, this adds an hyperparameter that leads to optimization difficulties.

Structural Agnostic Model

The Structural Agnostic Model [32] (SAM) leverages **multiples conditional GANs** [33] to infer a **causal graph** out of the data. Each variable is generated using all other true variables and using penalized coefficients in front of the variables and a constraint to recover a directed acyclic graph, the parent variables of each node are selected. Based on the trained GANs, and the obtained graph, building a hierarchical generative model allows for successful generation of data.

Differentially Private GAN

Differentially Private GAN [34] (DPGAN) is a derived version of **WGAN** that tries to introduce the notion of **privacy** in the generative process which is not handled by the GAN models.

One common issue 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.

The main idea begind DPGAN is to ensure the notion of Differential Privacy [35] which is aims to maximize accuracy while generating and to minimize the chance of identifying a patient after generation. This is translated by adding noise to gradients during the learning procedure and clipping the norm of the gradient.

MedGAN

medGAN [36] was introduced to generate high-dimensional, multi-label discrete variables which is commonly the case with electronic health records. It learns distributions of discrete features with the help of an **autoencoder** and an adversarial framework. This version of GAN introduces a couple of changes over the original GAN model such as combining an autoencoder, implementing a method called mini-batch averaging to cope with situations of mode collapse which makes it more suitable to generate medical data.

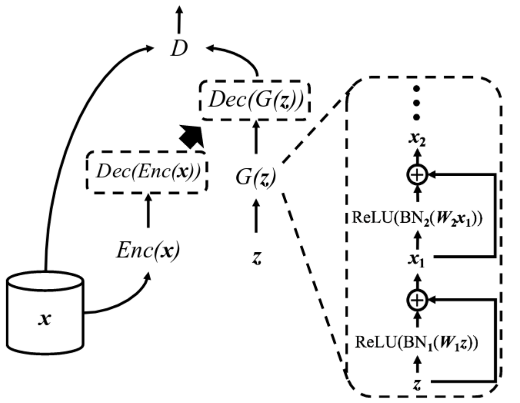


Figure 4.5: medGAN scheme

Evolutionary GAN

Evolutionary GAN [37] (E-GAN) is a GAN trained by **evolution strategy** optimization. The input of the black-box optimizer is the generator network weights and the objective function is the discriminator's feedback.

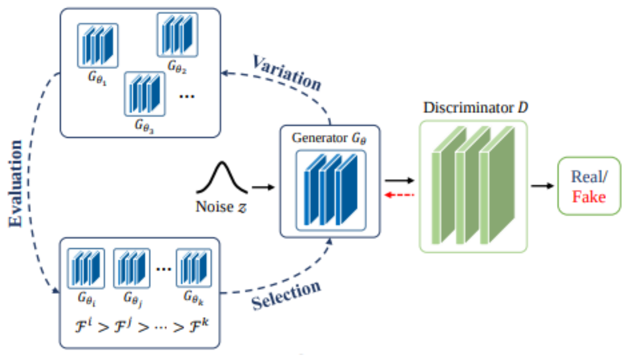


Figure 4.6: E-GAN

The use of 0-order optimization could allow us to generate directly integers values because it doesn't require differentiable activation functions. However, despite of the highly parallelisable nature of this algorithm, its convergence time increases with the number of dimensions.

Remark: We didn't implement this generator.

Deep convolutional GAN

DCGAN [38] is one of the popular and successful network design for GAN. It mainly composes of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the downsampling and the upsampling.

Remark: We didn't implement this generator.

Causal Generative Neural Network

Causal Generative Neural Network [39] (CGNN) learns **Functional Causal Models** (FCM) from observational data. CGNNs leverage conditional independencies and distributional asymmetries to discover bivariate and multivariate causal structures. CGNNs make no assumption regarding the lack of co-founders, and learn a differentiable generative model of the data by using backpropagation. Extensive experiments show their good performances comparatively to the state-of-the art in observational causal discovery on both simulated and real data, with respect to cause-effect inference, v-structure identification, and multivariate causal discovery.

Remark: We didn't implement this generator.

5 | Comparative study of generators

I conducted a systematic scientific study to properly compare specific generative models.

5.1 Study description

The goal of this comparative study is to fairly compare the different generators. I applied various comparison metrics on original and generated datasets, with systematic settings. The implementation is available here[1].

Datasets

Name	Size	Variable number	Variable type	Problem type
Iris	150	4	Numerical	Multi-class classification
Mushrooms	6499	23	Categorical	Binary classification
Boston housing	506	14	Mixed, mostly numerical	Regression
Adult	48842	15	Mixed	Binary classification
MIMIC (Medical applications)	26927	343	Mixed, lots of binary	Binary classification

Table 5.1: Datasets description

Each dataset has a train and a test set. The train set represent 70% of the entire dataset.

Data is available on GitLab on a private repository[2].

Processing

The *pre*-processing and *post*-processing were done using the SDV procedure (cf. section 2.3). The target variable is then decoded (to restore categorical class) using the stored parameters.

Generative models

Models are trained on the real data **train set**.

Number of samples generated for each generator:

- On Iris, Mushrooms, Boston: *10,000 samples.*
- On MIMIC: *20,000 samples.*
- On Adult: *30,000 samples.*

Except for *Random Forest imputation* model which gives synthetic datasets the same size as the original ones. It is not interesting to sample more points with this algorithm if the proportion of replacement parameter p is equal to 1, because under this condition the model is deterministic and can't generate more samples than there were on the original dataset.

[1]`https://github.com/Didayolo/medi-chal/blob/master/notebooks/systematic_study_pipeline.ipynb`
[2]`http://gitlab.com/didayolo/data`

As generators I used:

- **Multivariate gaussian** as a baseline.
  - *Mixture component number* = 1.
- **Parzen window** kernel density estimation.
  - *Bandwidth* = 0.2,
  - Gaussian kernel,
  - Euclidean metric.
- **Random Forest imputation**
  - *Replacement probability* = 1.
  - *Random Forest number of estimators* = 10.
- **Wasserstein GAN**.
  - *Epochs* = $200,000$,
  - *Batch size* = $5,000$,
  - Feed-forward architecture.
    - Generator: [100, (2$\times$ *output_dim*), (1.5$\times$ *output_dim*), *output_dim*]
    - Discriminator: [*output_dim*, 64, 128, 256, 1]

  Stable GAN training is an open research problem. A simple GAN training consists in using the DCGAN Architecture with the available hyperparameters as described in the paper [38]. GAN Training is highly volatile and using different hyperparameters will easily lead to mode collapse or vanishing gradients. The easier route with GANs is to use the Wasserstein GAN. Those are quite stable with arbitrary architectures. However, I suggest to use the hyperparameter proposed in this paper [40] because the training collapsed for other hyperparameters.

Metrics

All metrics are computed on real data **test set** and generated data (except classification/regression performance). For more accurate definitions, please refer to chapter 3.

- **Principal component analysis** (PCA). We overlay the two PCA plots.
- **Minimum distance accumulation** (MDA). The first plot shows the MDA curve while the second shows resemblance and privacy values for various threshold.
- **Classification/regression performance**. The scoring metric is **accuracy** for classification and **r2** for regression. r^2 best possible score is 1 while bad scores can be arbitrarily negative.
- **Discriminant score**: A classifier is trained to output the probability of a given sample being real or fake. The model trained is a MultiLayer Perceptron with layers size [100, 200], ReLU activation function and a constant 0.001 learning rate.

For PCA, MDA and discriminant score datasets are re-sampled at the same size.

5.2 Results

I only show here the results on the dataset that interests us the most: the MIMIC dataset. For others, please refer to the study report[3].

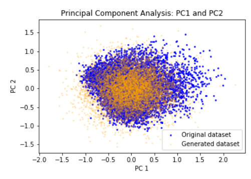
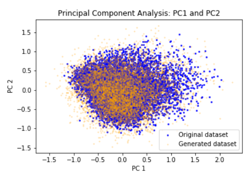
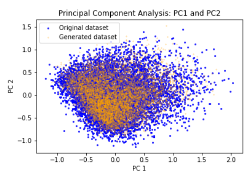
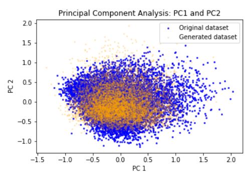
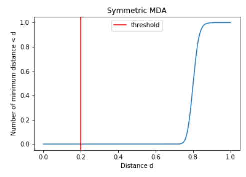
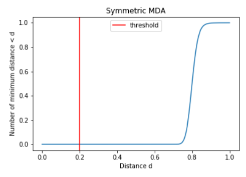
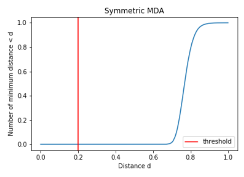
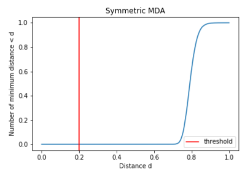
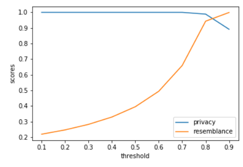
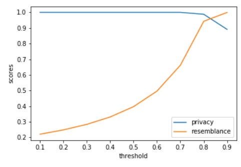
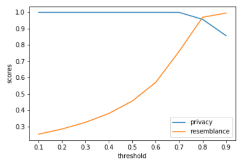
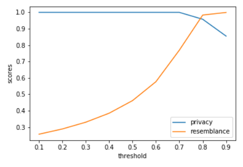
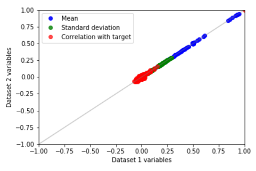
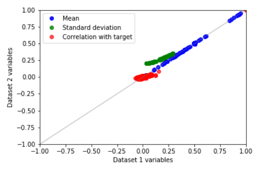
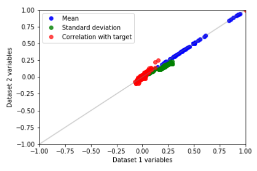
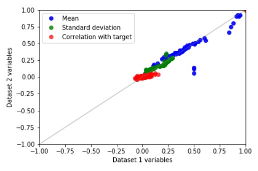
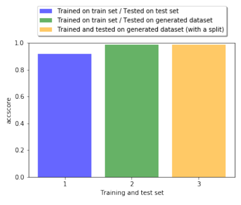
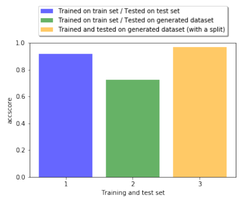
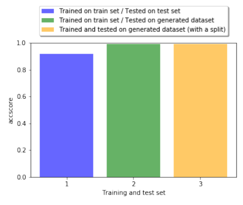
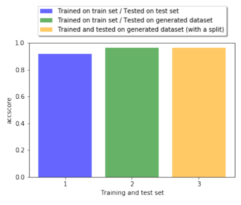
Generator	Multivariate Gaussian	Parzen Windows	Random Forest	Wasserstein GAN
PCA plot				
MDA plot				
Threshold variation				
Marginals				
Task score				
Discriminant score	precision recall Original dataset 0.56 0.33 Generated dataset 0.54 0.75 avg / total 0.55 0.54	precision recall Original dataset 0.81 0.66 Generated dataset 0.83 0.92 avg / total 0.82 0.82	precision recall Original dataset 0.55 0.35 Generated dataset 0.55 0.74 avg / total 0.55 0.55	precision recall Original dataset 0.77 0.22 Generated dataset 0.78 0.98 avg / total 0.78 0.78

Table 5.2: **MIMIC results**

Results show that *Random Forest imputation* works well but it amounts to add noise to the data. *Wasserstein GAN* have a huge potential but is hard to optimize. We also observed that the task score and discriminant metrics are simple but efficient way to compute discrepancy between complex distributions.

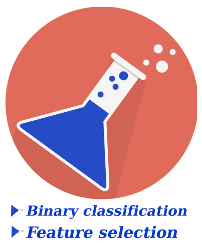
[3]`https://github.com/Didayolo/medi-chal/blob/master/documents/systematic_study_generative_model.pdf`

6 Challenges

The goal of the project was to use synthetic data in challenges, for educational and research purposes. Data challenges are **competitions** where participants try to obtain the best score on specific task(s) on one or more dataset(s). This is a good way to learn new techniques and it sometimes pushes back the limits of the state-of-the-art methods. Our challenges do not have end dates so you can give it a try by yourself; they are quiet fun to do. We have created three challenges so far:

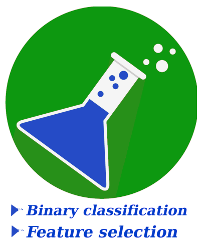
6.1 Chems

Version 1: Binary Classification



The goal of this challenge[1] is to create a model to predict ready biodegradation of chemicals by using molecular descriptors. A data set of with molecular descriptors and biodegradation experimental values of various chemicals were collected. Participants have to develop a computational model to predict the ready biodegradability of molecules.

Version 2: Feature Selection



This challenge[2] is based on the previous *Chems* challenge. We added a **feature selection** problem. That means adding many **fake features**, typically 3 times the number of real features. A fake feature consists in a random permutation of a real feature. By doing this, we create variables with equal properties such as variance and mean but totally uncorrelated and meaningless regarding the other variables. Users will have to classify each variable as fake or not and receive a score computed as for a classification problem.

X1	X2	X3	fake1	fake2	fake3	fake4	fake5
3	4	5	5	4	3	5	6
5	6	7	1	2	5	3	4
1	2	3	3	6	7	1	2

Table 6.1: Adding fake features to data to create a feature selection problem (illustration of concept)

[1]https://competitions.codalab.org/competitions/18751
[2]https://competitions.codalab.org/competitions/20006

6.2 To be or not to be? Mortality Prediction Challenge

To be, or not to be?
Mortality Prediction Challenge

The main question of this challenge[3] is: How to predict the survival of a patient given his or her medical record? More specifically, participants have to predict whether or not the patients died during their stay at the hospital.

The task is then a **binary classification** evaluated with **balanced accuracy** scoring metric.

The *pre*-processing and *post*-processing were done using the SDV procedure (cf. section 2.3). The synthetic dataset were generated with a Wassterstein GAN trained with the same parameters as in the previous comparative study of generators (cf. chapter 5).

Simply encode data and run basic models such as Random Forest leads to bad results with balanced accuracy scores near 0.50. It is probably due to the imbalanced classes distribution.

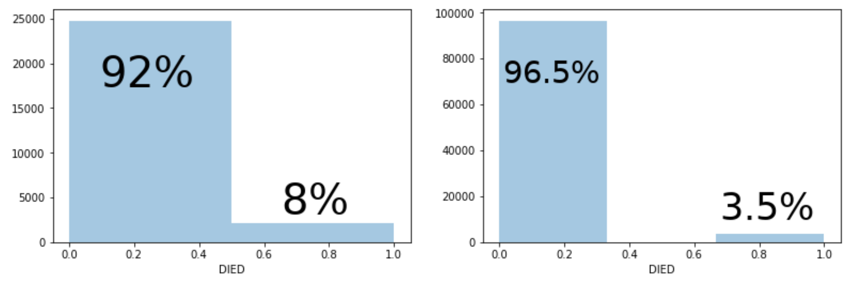


Figure 6.1: Classes distribution in original dataset (left) and synthetic dataset (right)

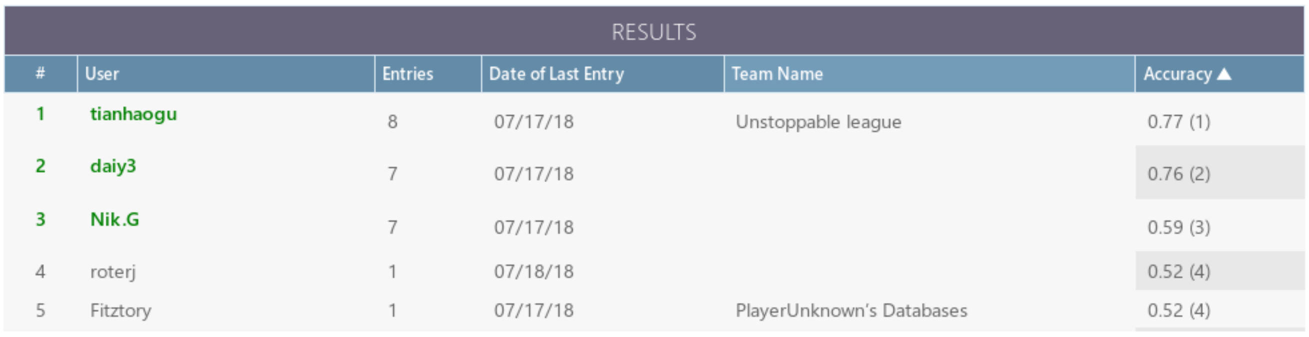
We can counter this effect by **oversampling**: we sample data with replacement with more probability on the rarest class. Here are the classification performances on original MIMIC dataset, on the synthetic one and the performances when trained on the synthetic and test on the original dataset. Results are cross-validated by doing 20 different train/test splits with a 70% train set size.

Model	Train on original Test on original	Train on synthetic Test on synthetic	Train on original Test on synthetic
LogReg	0.76	0.76	0.77
GradBoost 150	0.80	0.77	0.71
RF 100	0.50	0.50	0.50
MLP [100]	0.79	0.77	0.76
MLP [100, 100]	0.80	0.77	0.77

Table 6.2: Balanced accuracy for various models after oversampling

For some reason the Random Forest classifier returned “NOT DIED” for every data point despite its 100 estimators. We can see that the oversampling worked well and the synthetic and original data behaviours are close.

[3]`https://competitions.codalab.org/competitions/19365`



RESULTS

#	User	Entries	Date of Last Entry	Team Name	Accuracy ▲
1	tianhaogu	8	07/17/18	Unstoppable league	0.77 (1)
2	daiy3	7	07/17/18		0.76 (2)
3	Nik.G	7	07/17/18		0.59 (3)
4	roterj	1	07/18/18		0.52 (4)
5	Fitztory	1	07/17/18	PlayerUnknown's Databases	0.52 (4)

Figure 6.2: Leaderboard top 5 scores

The leaderboard shows a wide range of possible score which is interesting for a challenge. The best score can probably be improved.

7 | Conclusion and perspectives

To go further

Categorical data encoding There are interesting encodings (mentioned in section 2.3) relying on deep learning to try. We should test them on a data generation task instead of classification.

Causality To go further we could take into account the causality relationships between variables for generation with CGNN [39] or SAM [32] as defined in *GAN* section (cf. 4.5). Causality could also be used as a **resemblance metric** by comparing the **causal inference results**. The question is then: do we see the same causality relationships between variables in both datasets?

Survival Analysis Challenge We discussed a possible challenge on a survival analysis task. The objective of this challenge would not only be to assess participant's performance on a specific task but also to have an educational point of view and open a field of study to machine learning experts. Survival analysis is widely used by statisticians interested in medical studies. It is used to analyze data where the **outcome variable is the time *t* until an event *e* occurs.**

In this challenge, we want to apply survival analysis techniques, teach them to participants and let them find better solutions using machine learning algorithms or not.

Conclusion

The creation of synthetic datasets is a complex and interesting problem full of possibilities. GANs can show awesome results but are for now really hard to train. Applying the techniques directly by creating challenges was motivating and fun to do. I have learned a lot during this internship, not only in the domain of Data Science but also in organization and management thanks to my leadership experience. Now, I can dive more easily into scientific papers, I have improved my English expression skills and I strongly want to pursue my study of generative models.

Acknowledgements

I would like to thank Isabelle Guyon who, with optimism and wise advice, allowed me to do this internship in the best conditions. She had a great patience and gave me detailed feedback on my work and reports. By making me the team coordinator, she showed she trusted me to organize project and supervise the interns.

Thanks also to all the *Medi-Chal* Team members who have done an amazing job during this project and to Kristin Bennett who gave us directions and used our challenges in her courses.

I am grateful as well to the TAU Team at INRIA Saclay for the warm welcome, all the interesting discussions I had there and the great working atmosphere.

Bibliography

[1] He Ren and Quan Yang. "Neural Joke Generation" (). URL: https://web.stanford.edu/class/cs224n/reports/2760332.pdf.

[2] Huanru Henry Mao, Taylor Shin, and Garrison W. Cottrell. "DeepJ: Style-Specific Music Generation" (). URL: https://web.stanford.edu/class/cs224n/reports/2760332.pdf.

[3] Kilian Q. Weinberger et al. "Feature Hashing for Large Scale Multitask Learning". *CoRR* abs/0902.2206 (2009). arXiv: 0902.2206. URL: http://arxiv.org/abs/0902.2206.

[4] Cheng Guo and Felix Berkhahn. "Entity Embeddings of Categorical Variables". *CoRR* abs/1604.06737 (2016). arXiv: 1604.06737. URL: http://arxiv.org/abs/1604.06737.

[5] Tianyao Chen Ying Wen Jun Wang and Weinan Zhang. "Cat2Vec: Learning Distributed Representation of Multi-field Categorical Data". 2016. URL: http://openreview.net/pdf?id=HyNxRZ9xg.

[6] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. "The Synthetic data vault". *DSAA* (). URL: http://ieeexplore.ieee.org/document/7796926/.

[7] Matthias Reif. "A comprehensive dataset for evaluating approaches of various meta-learning task" (). URL: https://www.researchgate.net/publication/230625668_A_Comprehensive_Dataset_for_Evaluating_Approaches_of_various_Meta-Learning_Tasks.

[8] Guido Lindner and Rudi Studer. "AST: Support for Algorithm Selection with a CBR Approach" (). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.3556&rep=rep1&type=pdf.

[9] Robert Engels and C. Theusinger. "Using a Data Metric for Preprocessing Advice for Data Mining Applications". *ECAI*. 1998, pp. 430–434.

[10] So Young Sohn. "Meta Analysis of Classification Algorithms for Pattern Recognition". *IEEE Trans. Pattern Anal. Mach. Intell.* 21.11 (1999), pp. 1137–1144. DOI: 10.1109/34.809107. URL: https://doi.org/10.1109/34.809107.

[11] Emilio Corchado, Ajith Abraham, and Witold Pedrycz, eds. *Hybrid Artificial Intelligence Systems, Third International Workshop, HAIS 2008, Burgos, Spain, September 24-26, 2008. Proceedings.* Vol. 5271. Lecture Notes in Computer Science. Springer, 2008. ISBN: 978-3-540-87655-7. DOI: 10.1007/978-3-540-87656-4. URL: https://doi.org/10.1007/978-3-540-87656-4.

[12] Steffen Lange, Ken Satoh, and Carl H. Smith, eds. *Discovery Science, 5th International Conference, DS 2002, Lübeck, Germany, November 24-26, 2002, Proceedings.* Vol. 2534. Lecture Notes in Computer Science. Springer, 2002. ISBN: 3-540-00188-3. DOI: 10.1007/3-540-36182-0. URL: https://doi.org/10.1007/3-540-36182-0.

[13] Pat Langley, ed. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000.* Morgan Kaufmann, 2000. ISBN: 1-55860-707-2.

[14] Djamel A. Zighed, Henryk Jan Komorowski, and Jan M. Zytkow, eds. *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings.* Vol. 1910. Lecture Notes in Computer Science. Springer, 2000. ISBN: 3-540-41066-X. DOI: 10.1007/3-540-45372-5. URL: https://doi.org/10.1007/3-540-45372-5.

[15] Joscha Bach and Stefan Edelkamp, eds. *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings.* Vol. 7006. Lecture Notes in Computer Science. Springer, 2011. ISBN: 978-3-642-24454-4. DOI: 10.1007/978-3-642-24455-1. URL: https://doi.org/10.1007/978-3-642-24455-1.

[16] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis" (). URL: https://www.sciencedirect.com/science/article/abs/pii/0169743987800849.

[17] Arthur Gretton et al. “A Kernel Two-Sample Test”. *Journal of Machine Learning Research* 13 (2012), pp. 723–773. URL: http://dl.acm.org/citation.cfm?id=2188410.

[18] Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. “Traning generative neural networks via Maximum Mean Discrepancy optimization” (). URL: https://arxiv.org/pdf/1505.03906.pdf.

[19] Gao Huang et al. *An empirical study on evaluation metrics of generative adversarial networks.* 2018. URL: https://openreview.net/forum?id=Sy1f0e-R-.

[20] R. Agrawal and R. Srikant. “Privacy-preserving data mining”. *ACM SIGMOD Rec* 29.2 (2000), pp. 439–450.

[21] P. Samarati and L. Sweeney. “Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression”. *IEEE Symp. Res. Secur. Privacy* (1998), pp. 384–393.

[22] P. Samarati and L. Sweeney. “Generalizing data to provide anonymity when disclosing information”. *Proc. PODS* (1998), p. 188.

[23] N. Li, T. Li, and S. Venkatasubramanian. “t-closeness: Privacy beyond k-anonymity and l-diversity”. *Proc. IEEE 23rd Int. Conf. Data Eng. (ICDE)* (2007), pp. 106–115.

[24] C. Dwork. “Differential privacy”. *Automata, Languages and Program- ming* 4052 (2006), pp. 1–12.

[25] D. Agrawal and C. C. Aggarwal. “On the design and quantification of pri- vacy preserving data mining algorithms”. *Proc. 20th ACM SIGMOD- SIGACT-SIGART Symp. Principles Database Sys* (2001), pp. 247–255.

[26] S. R. M. Oliveira and O. R. Zaıane. “Privacy preserving clustering by data transformation”. *J. Inf. Data Manag* 1.1 (2010), p. 37.

[27] D. Lin E. Bertino and W. Jiang. “A survey of quantification of privacy preserving data mining algorithms”. *Privacy-Preserving Data Mining* (2008), pp. 183–205.

[28] Vadim Smolyakov. “Ensemble Learning to Improve Machine Learning Results” (2017). URL: https://blog.statsbot.co/ensemble-learning-d1dcd548e936.

[29] Jonathan Bartlett. “Methodology for multiple imputation for missing data in electronic health record data” (2014). URL: http://thestatsgeek.com/wp-content/uploads/2014/09/RandomForestImpBiometricsC
pdf.

[30] Ian J. Goodfellow et al. “Generative Adversarial Networks”. *CoRR* abs/1406.2661 (2014). arXiv: 1406.2661. URL: http://arxiv.org/abs/1406.2661.

[31] Alex Irpan. “Read-through: Wasserstein GAN” (2017). URL: https://www.alexirpan.com/2017/02/22/wasserstein-gan.html.

[32] D. Kalainathan et al. “SAM: Structural Agnostic Model, Causal Discovery and Penalized Adversarial Learning”. *ArXiv e-prints* (Mar. 2018). arXiv: 1803.04929 [stat.ML].

[33] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. *arXiv preprint arXiv:1411.1784* (2014).

[34] Liyang Xie et al. “Differentially Private Generative Adversarial Network”. *CoRR* abs/1802.06739 (2018). arXiv: 1802.06739. URL: http://arxiv.org/abs/1802.06739.

[35] Cynthia Dwork. “Differential Privacy: A Survey of Results”. TAMC’08 (2008), pp. 1–19. URL: http://dl.acm.org/citation.cfm?id=1791834.1791836.

[36] Edward Choi et al. “Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks”. *CoRR* abs/1703.06490 (2017). arXiv: 1703.06490. URL: http://arxiv.org/abs/1703.06490.

[37] Chaoyue Wang et al. “Evolutionary Generative Adversarial Networks”. *CoRR* abs/1803.00657 (2018). arXiv: 1803.00657. URL: https://arxiv.org/abs/1803.00657.

[38] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. *CoRR* abs/1511.06434 (2015). arXiv: 1511.06434. URL: http://arxiv.org/abs/1511.06434.

[39] Olivier Goudet et al. “Causal Generative Neural Networks” (). URL: https://arxiv.org/abs/1711.08936.

[40] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. *CoRR* abs/1704.00028 (2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.