Use of GAN's to Generate Synthetic Data: Medical Data Case Study

Final Project – Deep Learning – 2019/2020

Group 13
Inês Diogo (M20190301), Lara Neves (M20190867), Susana Paço (M20190821)

Motivation

The application of Generative Adversarial Networks (GANs) to generate synthetic data has been considered a possibility since its inception in 2014, by Ian Goodfellow (Goodfellow et al., 2014).

In this work we will endeavor to create a simple GAN architecture that can be applied by the *Instituto de Medicina Molecular* to generate anonymized data for their projects. For the medical research area this approach brings several benefits, namely:

- Assuring patient privacy by creating synthetic data not directly related to any patient.
- Creation of more data in areas where the organic collecting and distribution of data is difficult, namely genomics or patient data.
- Enabling collaboration between medical researchers and engineering/data science teams without the violation of medical privacy laws.
- Allowing researchers to expand their datasets where money and other constraints can severely hinder this process and thus facilitate the use of Artificial Intelligence in more limited fields of medical research.

While our project should be easily generalized to all types of tabular data, our primary focus will be with patient data during the current COVID-19 pandemic, which both necessitates anonymization and the creation of more patient profile data, as such data is not easily accessible to the public despite the potential for valuable insight to be gathered with this data if it was widely available.

Due to the aforementioned difficulty in obtain patient records, the data used in this project comes from a Machine Learning Project, as it has similar characteristics to a standard dataset of pandemic patient data, allowing us to experiment with it and avoid any privacy issues. Our goal is to generate a new dataset based upon the original one and evaluate that it maintains its statistical consistency, even after being fully synthesized.

We aimed for a simplified architecture and the assurance that the data structure remains intact.

Introduction

This work explores the development of a simple GAN architecture that enables the creation of synthetic data from medical data. The data used is a generic medical type data from a Machine Learning classification project. The initial problem consisted of the creation of a simple survival prediction system

from a hypothetical pandemic and shares similar characteristic to the COVID-19 patient data, namely categorical data that can be encoded into dummy variables, in addition to discrete and continuous features. As such, the main goal is to determine which people die and which people survive this disease.

We attempt to demonstrate that it is possible to generate an entirely synthetic dataset from this data that can go through the same predictive modelling as the original dataset and attain similar results. In a real-life situation this synthetic dataset could have been widely distributed thus enabling collaboration between interdisciplinary groups and allowing for different and potentially innovative results.

Development

Our dataset, named “The Boolean Pandemic dataset” required some preprocessing to aid in the correct application of the GAN. This allowed us to standardize and convert all the dataset into numerical variables to make them easier to be digested by the GAN.

The original data contained identifying information about the patients, namely *First_Name*, *Family_Name*, *Patient_ID* and *Family_Case_ID*. This information was removed in a first anonymization phase, in section 3 of the jupyter notebook. Additionally, some feature engineering was added to preserve the information that existed in the previously mentioned features, while keeping the data private such as the variable *is_alone* representing a patient with no other relatives on the dataset.

Without standardizing the data, the distribution of the synthetic data is negatively impacted, consequently the Robust Scaler from the *sklearn* package was employed specifically because it is robust to outliers as shown in section 4 of the notebook.

Through plotting the data, we noticed that there is a class imbalance present in favour of *Deceased*, this can also be corrected by generating data with the GANs.

Finally, we created a Naïve Bayes (NB) classifier in section 6 of the notebook, a typical model used for this type of survival analysis which will allow to create a baseline as a measure of comparison between the original data and the synthetic data. If our synthetic data behaves similarly to original data in the NB classifier then there’s reasons to believe that it will be an adequate replacement of the original dataset as intended.

With the preprocessing done, we are now ready to implement the GANs, corresponding to the section 7 of the jupyter notebook.

We started by implementing a conditional GAN package called *CTGAN* (Xu et al., 2019) to be used as a baseline for comparison with the GAN and Conditional GAN that was developed and trained right after:

Our first step was to research current architectures in the GAN space for tabular data. We found articles by *Fabiana Clemente* (Clemente, 2020) and by Cody Nash (Nash, 2019). These productions seemed to solve a similar problem to ours and we used them as guidelines.

The starting point of our implementation and the previous examples’, made use of the GAN-Sandbox (MIT, n.d.) using *Keras* library and *TensorFlow* backend. Note however that this implementation is set up for image processing and as such a significant level of adjustment was necessary to adapt this architecture to generating tabular data, as it lacks any special structure among the variables which would be required to

implement the type of convolutional transformations used between the layers of an image processing network.

To solve this, the convolutional networks was substituted by densely connected layers and thus, allowing the network to learn from tabular data.

Both the GAN and Conditional GAN (CGAN) used this type of architecture on their generator and discriminator networks. This structure allowed the testing of several architectures in both these networks. The activation function used was *relu* which seemed to be the most effective while the last layer of the discriminator network was, as expected, a *sigmoid* function. Each following layer increased proportionally in the generator and then decreased in the discriminator.

The differentiating factor of the CGAN in comparison with the vanilla GAN is the presence of the dataset labels, in this case *Deceased* as a binary label. This will be particularly advantageous for us as it will, in a one step implementation, generate synthetic data with the label already included. It also tends to perform better than GAN when it comes to maintaining the distributions between real and synthetic data as similar as possible.

Finally, several visualizations were created to validate the similitude of the synthetic data with the real data, to do so it was necessary to use a classifier that would allow for a fast and powerful classification, without much tuning, but with the possibility of establishing a stopping criteria to minimize loss and maximize accuracy. The algorithm *xgboost* (Chen & Guestrin, 2016) is a gradient-boosted decision tree algorithm and the one we will use.

Results

To evaluate the best possible GAN architecture created, a collection of tests was assembled.

First as mentioned previously, the baseline classification of the original data with a Naïve Bayes classifier was as following.

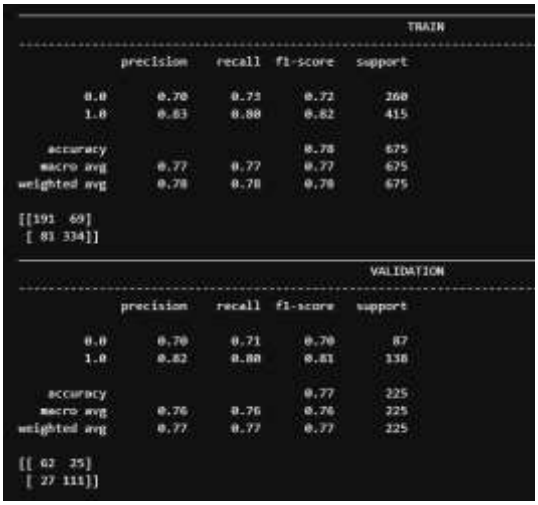

Figure 1 - Naïve Bayes Classifier results for the Original Data

The CTGAN baseline was used for 500 epochs to generate 900 rows of synthetic data with the following results on the NB classifier:

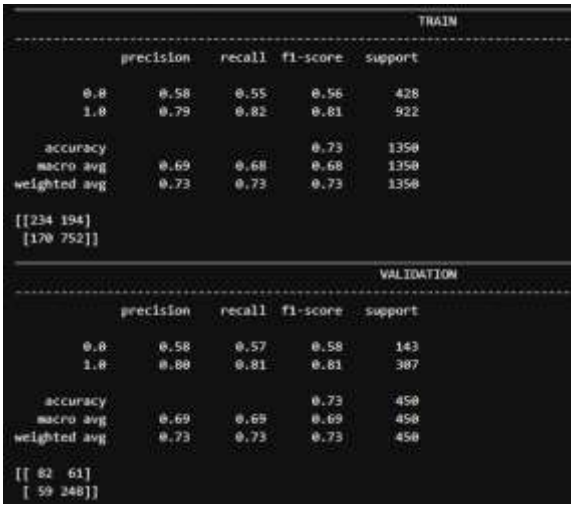

Figure 2 - CTGAN with 500 epochs generating 900 rows of data

These will be the values to beat with our GAN and CGAN architecture, which had a preliminary result of:

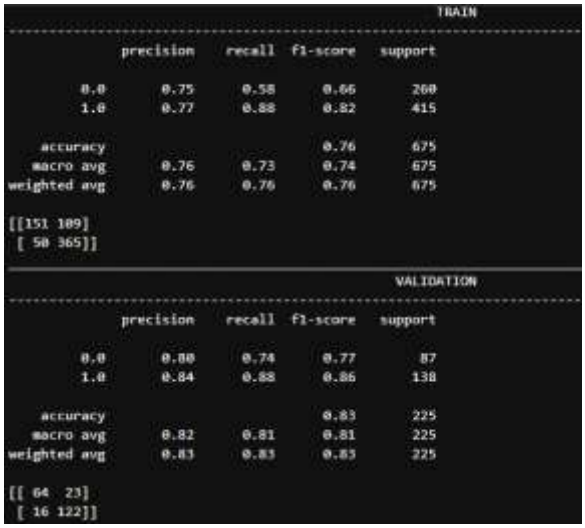

Figure 3 – CGAN with 500 epochs generating 900 rows of data

The results of the 900 rows of synthetic data generated by CGAN on the NB classifier are indeed very promising. However as seen in Figure 4, the distribution of the synthetic data is in no way similar to the real data. We will attempt to improve this by altering the parameters, namely the number of epochs, the learning rate, and the structure of the network.

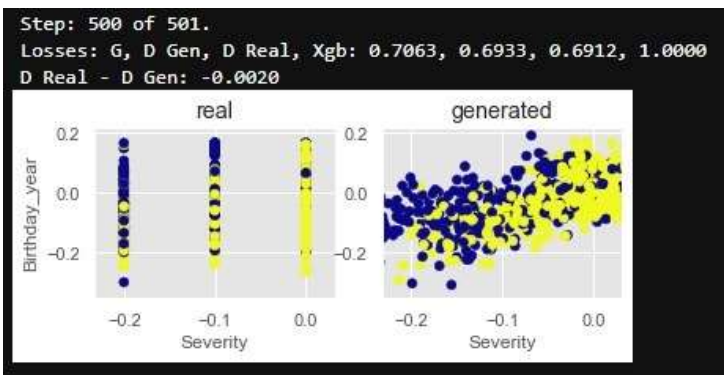

Figure 4 - Distribution of real vs synthetic data in CGAN with 500 epochs

Structure of The Network

Several sized structures on the generator and discriminator networks were attempt.

First a larger one with 8 layers on the generator and 6 layers on the discriminator.

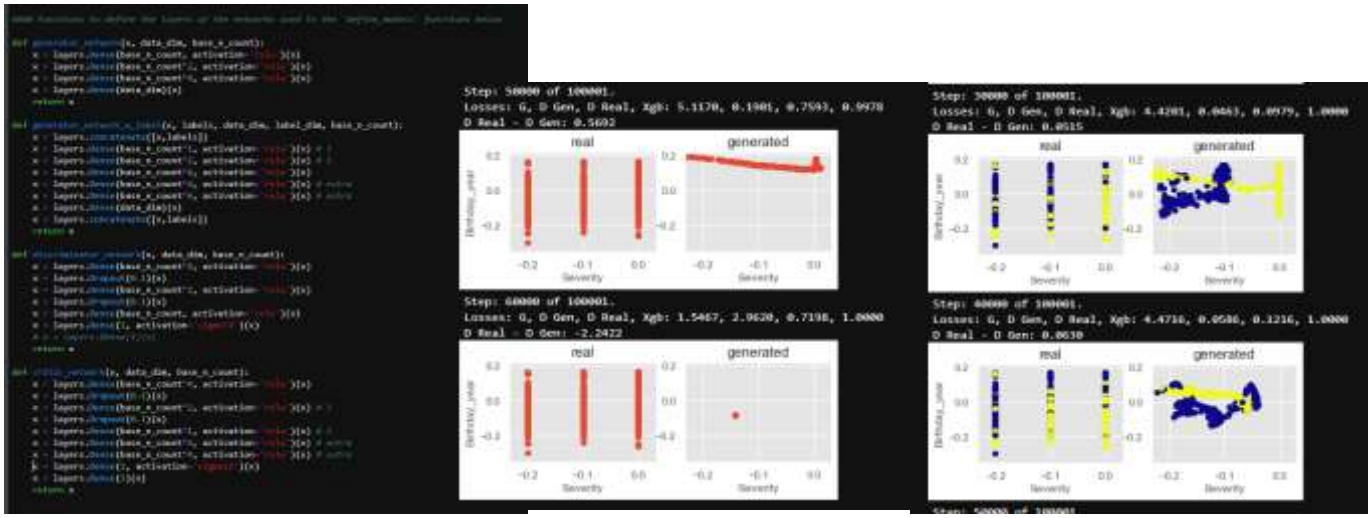

Figure 5 – GAN and CGAN with 100.000 epochs and a larger amount of layers. Learning rate of $5e^{-4}$.

A bigger network is not always the best answer. The plots demonstrate that after 50.000 steps of the GAN and 30.000 steps of the CGAN, both networks cease to learn, probably due to vanishing gradient.

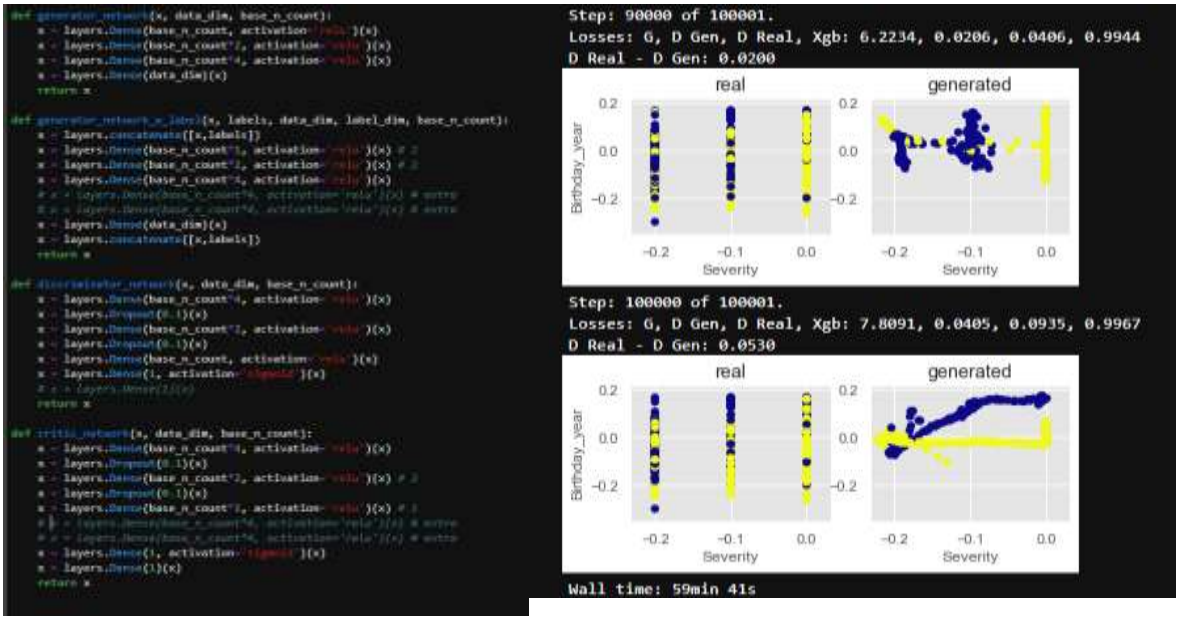

Figure 6 - CGAN with 100.000 epochs and a medium amount of layers. Learning rate of $5e^{-4}$.

A medium sized network was attempted as seen below, showing noticeably better behavior but still suffering from vanishing gradient.

A slightly smaller network, with 6 layers on the generator and 4 on the discriminator, show GAN distributions much closer to the real data distributions on two different steps.

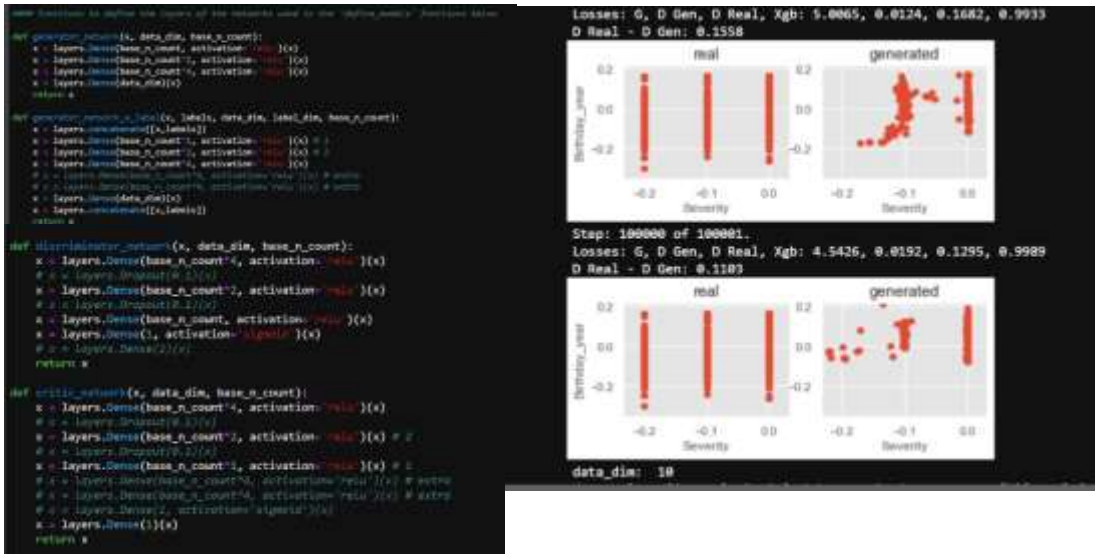

Figure 7 - GAN with 100.000 epochs and a smaller number of layers. Learning rate of $5e^{-4}$.

Testing The Learning Rate

During the development, several learning rates were tested namely $5e^{-2}, 5e^{-3}, 5e^{-4}, 5e^{-5}$. Consistently, the learning rate of $5e^{-5}$ allowed for better performance, both in the distributions of the data and in the final NB classifier score. Though this distinction was made clear only in very large epoch runs.

Number of epochs

In order to determine the optimal number of epochs, namely the one corresponding to the lowest combined loss and corresponding to the highest accuracy between the real and the synthetic test set, we ran both GANs with an unusually high number of epochs, set at 150.000, learning rate equal to $5e^{-5}$ and using the best architecture that we discovered, corresponding to Figure 7.

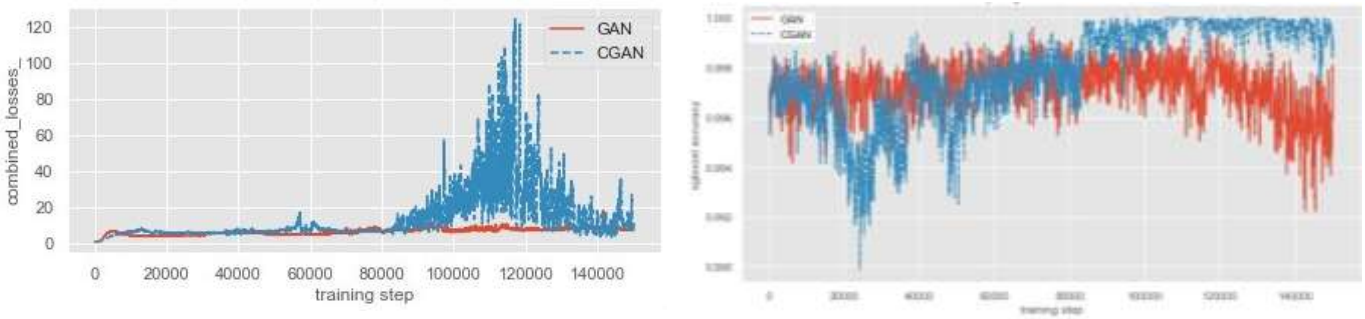

Figure 8 - Combined losses and accuracy vs number of epochs, for 150.000. Learning rate of $5e^{-5}$.

Looking at the graphs we see that the range of steps that maximizes the accuracy and minimizes the combined losses is between 80.000 and 100.000 epochs.

Final Results

Finally, we can show the final results with the optimal parameters:

- Number of Epochs of 85.010
- Learning Rate of $5e^{-5}$
- Architecture corresponding to Figure 7

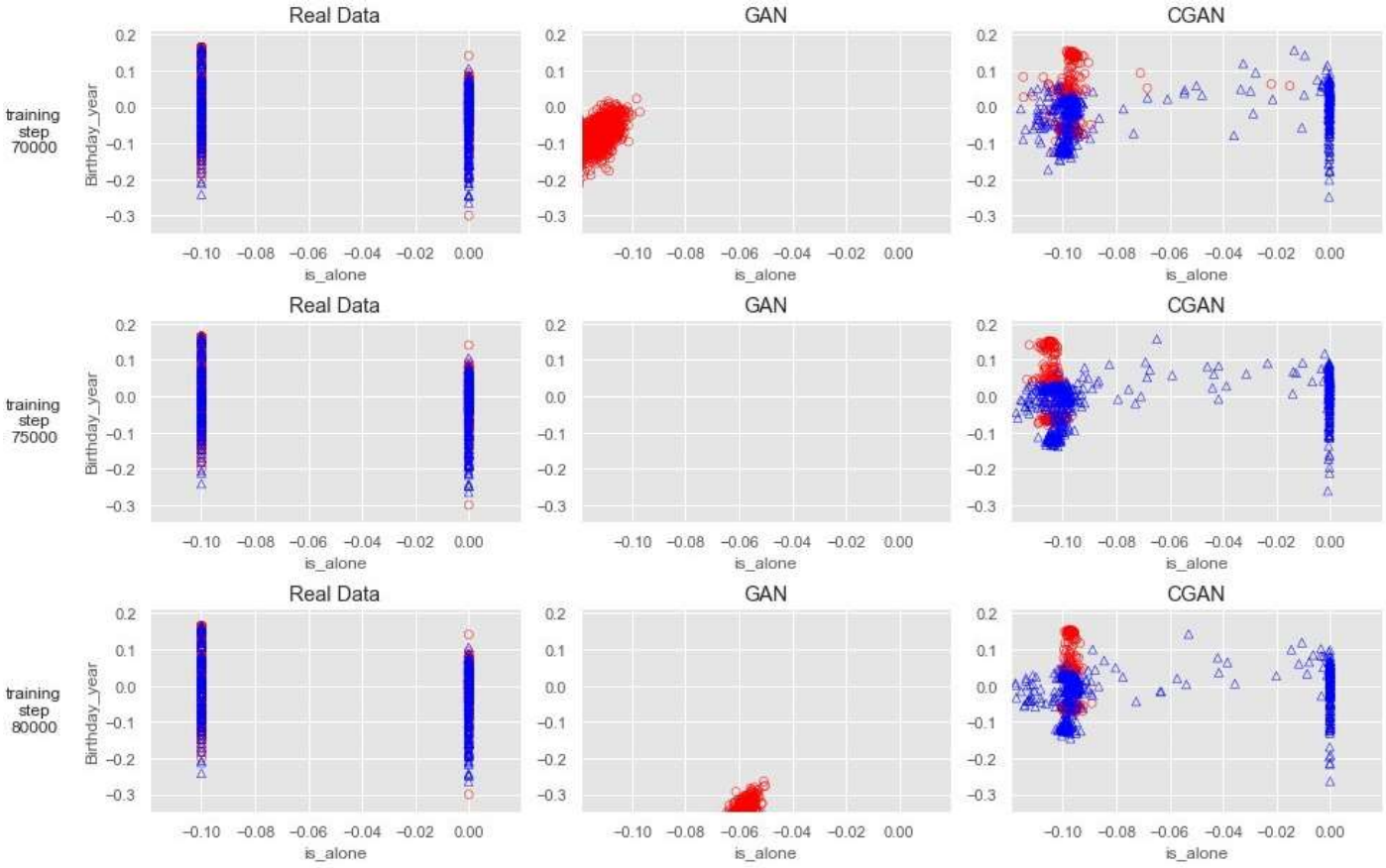

Figure 9 - Distribution of Birthday_Year and is_alone variables of the real data, the GAN and the CGAN generated data

As we can see, while GAN does not perform well, CGAN has a very similar distribution to the real data for these two variables. All the other variable combinations also show reasonably good distributions though this specific variable combination is the most evident.

TRAIN

	precision	recall	f1-score	support
0.0	0.98	0.97	0.98	260
1.0	0.98	0.99	0.99	415
accuracy			0.98	675
macro avg	0.98	0.98	0.98	675
weighted avg	0.98	0.98	0.98	675

```
[[252   8]
 [  4 411]]
```

VALIDATION

	precision	recall	f1-score	support
0.0	0.96	0.93	0.95	87
1.0	0.96	0.98	0.97	138
accuracy			0.96	225
macro avg	0.96	0.95	0.96	225
weighted avg	0.96	0.96	0.96	225

```
[[ 81   6]
 [  3 135]]
```

Figure 10 – Naïve Bayes Classifier of 900 rows generated with the CGAN using the final parameters.

Figure 10 shows excellent results on the NB classifier, a clear improvement from the CTGAN baseline (Figure 2) and even better evaluation scores when in comparison with the NB classifying the real data (Figure 1). Which shows that the initial imbalance from original data might have been mitigated. However, further refinement is necessary to assure that there's no overfitting.

Conclusions

During this project we arrived at the following conclusion:

- Out of the box alternatives such as the CTGAN package, are not always the best in terms of performance, primarily due to lack of flexibility in choosing parameters and altering the architecture in accordance to the specificities of the dataset. In particular, we see that the CTGAN can crash with large numbers of epochs, being noticeably slower than the architecture we developed;
- GANs usually demand a small learning rate, however tests are mandatory as it can easily turn into the network not learning at all;
- GANs do not necessarily need to be very large to be effective, in fact with smaller data, they may suffer from the vanishing gradient problem if they are too big;
- While synthetic data may not look exactly like the original dataset, where for example the discrete variables may turn into continuous one, if their distribution is maintained they are capable of fulfilling their role as a substitute for real data;
- GANs can be used to improve classification problems either by generating an entirely new set of data or by balancing an imbalanced dataset and preventing severe cases of bias;
- GANS are a useful tool in an area were privacy and the lack of data are serious issues to be dealt with;

Possible future steps for this project would be:

- Continue to test the final results' CGAN in the epochs range of 65.000 and 85.000 in order to find the "sweet spot" that gives the best result without overfitting;
- Implement alternatives such as Wasserstein GAN and DRAGAN;
- Testing the architecture of the GAN on real medical data;
- Improve the conversion tools in order for synthetic data to be more similar "visually" to the original data so it can be used as a teaching tool;
- Improving the code and interface to make it more user friendly for medical researchers.

Bibliography

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-Augu*, 785–794. https://doi.org/10.1145/2939672.2939785

Clemente, F. (n.d.). *Generating synthetic tabular data with GANs — Part 2*. Retrieved June 10, 2020, from https://medium.com/ydata-ai/generating-synthetic-tabular-data-with-gans-part-2-a0aba150539

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Nets*. http://www.github.com/goodfeli/adversarial

Nash, C. (n.d.). *Using Generative Adversarial Networks to Create Data from Noise | Toptal*. Retrieved June 10, 2020, from https://www.toptal.com/machine-learning/generative-adversarial-networks

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling Tabular data using Conditional GAN*. http://arxiv.org/abs/1907.00503

MIT. (n.d.). *GAN-Sandbox*. (Waya.ai) Retrieved 06 10, 2020, from https://awesomeopensource.com/project/mjdietzx/GAN-Sandbox