Pontificia Universidad Javeriana Cali
Engineering and Sciences Faculty
Computer and Systems Engineering
Undergraduate Thesis.

Machine Learning with Data Augmentation to Predict Glucantime Effectiveness Against Cutaneous Leishmaniasis

Juan José Hoyos Urcué

Thesis Director: Gloria Inés Álvarez .PhD
Thesis Co-director: Diego Luis Linares Ospina .PhD

November 19, 2021

Pontificia Universidad
JAVERIANA
Cali

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y computación.

Dr. Hernán Camilo Rocha Niño
Decano de la Facultad de Ingeniería

Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera Ingeniería de Sistemas.

Dra. Gloria Inés Álvarez Vargas
Directora del Trabajo

Dr. Diego Luis Linares Ospina
Codirector del Trabajo

Dra. María Constanza Pabón Burbano
Jurado 1

ING. Juan Pablo García Cifuentes
Jurado 2

Pontificia Universidad JAVERIANA Cali | Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería de Sistemas y Computación

Fecha: 21/02/2022

Autor: Juan José Hoyos Urcué.

Nombre del Proyecto de Grado:

Machine Learning With Data Augmentation To Predict Glucantime Effectiveness Against Cutaneous Leishmaniasis

Directores: Gloria Inés Álvarez y Diego Luis Linares.

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma de Directora del Proyecto de Grado

Firma de Co-Director del Proyecto de Grado

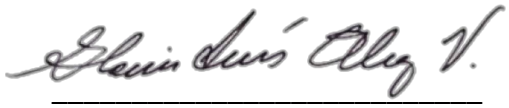
Santiago de Cali, 19 de noviembre del 2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria.
Director de Carrera de Ingeniería de Sistemas y Computación.

Cordial Saludo.

Por medio de la presente nos permitimos informarle que el estudiante de Ingeniería de Sistemas y Computación Juan José Hoyos Urcué (cod: 8935139) trabajó bajo nuestra dirección en el proyecto de grado titulado: "Machine Learning with Data Augmentation to Predict Glucantime Effectiveness Against Cutaneous Leishmaniasis", el cual hemos revisado detalladamente y consideramos apto para ser presentado.

Atentamente:



Dra. Gloria Inés Álvarez
Directora del Trabajo de Grado
Profesora del Departamento de Electrónica y Ciencias de la Computación.
Pontificia Universidad Javeriana Cali.

Dr. Diego Luis Linares Ospina
Codirector del Trabajo de Grado
Profesor del Departamento de Electrónica y Ciencias de la Computación.
Pontificia Universidad Javeriana Cali.

Santiago de Cali, 19 de noviembre del 2021.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria.
Director de Carrera de Ingeniería de Sistemas y Computación.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado: "Machine Learning with Data Augmentation to Predict Glucantime Effectiveness Against Cutaneous Leishmaniasis" con el fin de cumplir con los requisitos exigidos por la Universidad para posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe de que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del trabajo de grado.

Atentamente:

Juan José Hoyos Urcué.
Juan José Hoyos Urcué.
Código: 8935139

Acknowledgements

My special appreciation to my professors Gloria Inés Álvarez and Diego Luis Linares Ospina, who dedicated much of their valuable time guiding me and giving me feedback in a patient, cheerful, and especially willing way during the research and thesis development process.

Besides, I want to thank Dr. Maria Adelaida Gómez (Leishmaniasis Research Program Leader at CIDEIM), who shared exciting biology classes with me, instilled in me the necessary concepts to understand the technical and social context of this disease, highlighting her willingness and commitment with society by dedicating her entire life to solving orphan disease problems.

Likewise, I want to express gratitude to my colleagues and the academic staff of Javeriana Cali University for supporting me during my studies and training as a professional and a person.

Finally, I want to dedicate this thesis to my parents and my family. Their love and patience have supported me in achieving my goals and objectives; thank you for instilling in me the example of effort and courage to face problems and life.

Abstract

Facing data analysis problems on small data sets is a common problem in medical research; likewise, it is a problem that makes the application and success of classic machine learning algorithms very difficult. Many techniques have tackled the problem of a small data set, mainly for computer vision and image processing fields. However, for tabular data, short has been disseminated. In this degree project, the use of tabular data augmentation techniques is proposed to introduce synthetic instances quite similar to real instances, particularly in the context of a medical/social problem of predicting the effectiveness of Glucantime as a treatment against cutaneous Leishmaniasis. Experiments show that using these data augmentation algorithms enhances the characteristics of the initial data set and dramatically improves the performance of machine learning models.

Keywords: Machine Learning, Tabular Data Augmentation, Cutaneous Leishmaniasis, Infectious Disease, Synthetic Data, Small Dataset, K-Nearest Neighbors, Logistic Regression, Support Vector Machines, Neural Networks, Random Forest, SMOTE, Borderline SMOTE, Gaussian Mixture Model, ADASYN, Genetic Algorithm, Generative Adversarial Network.

Contents

List of Figures

List of Tables

Introduction

Leishmaniasis is an infectious disease transmitted by the bite of the Phlebotomine sandfly or female simulid mosquito. In Colombia, 95% of the annual cases are cutaneous *Leishmaniasis* [1]. The most widely used treatment to treat this disease is Glucantime, a highly toxic chemical substance that can generate severe adverse effects and even worse than the disease itself [2]. The most worrying thing is that it does not work in 100% of patients; that is why it arose the need to study the medical history and try to predict early whether or not this treatment will work in patients with this disease.

The research was developed with the International Center for Medical Training and Research (CIDEIM), which provided a demographic dataset for data analysis and machine learning experiments. However, this data set was too small, which made the conventional process of experiments with machine learning difficult. For this reason, it was purposed to use data augmentation techniques to take full advantage of the real data characteristics, generating more data to train the machine learning models and get better models performance.

All this allowed experimentally to build applicable models by health professionals, helping decide when to use Glucantime as a treatment for their patients suffering from *Leishmaniasis*, positively impacting their life quality and reducing the indifference and underestimation suffered by communities affected from it.

This report presents the problem description, problems to be solved based on the objectives, theoretical framework necessary for the project, experiments carried out in each phase to construct the models, and finally, the results and comparative analysis between machine learning and data augmentation techniques.

Chapter 1

Problem Description

1.1 Problem Statement

Disease research around the world has different motivations; on many occasions, researchers and governments are motivated by the amount of the population that is affected by a particular disease, which, for obvious reasons, generates a significant impact on society; others are motivated by the amount of money invested in treatments or cures for certain diseases. However, many diseases are not attractive in research because they affect a small part of the population that generally live in extreme poverty. Research in them is not viable because the economic return does not compensate for the investment that has to be made. Such is leishmaniasis, a disease that mainly affects poor people in Africa, Asia, and Latin America. This disease is associated with malnutrition, displacement, poor housing conditions, and a weak immune system.

Cutaneous leishmaniasis is the most common type of Leishmaniasis. It is a skin infection caused by a parasite transmitted through the bite of a sandfly. An average of 10,000 new Leishmaniasis cases are registered annually in Colombia, of which approximately 95% are cutaneous Leishmaniasis [1]. Cutaneous Leishmaniasis is generally treated with pentavalent antimonials such as Glucantime, which in many cases are effective in fighting the disease; however, there are other cases where they do not work, and it is not clear why. As a possible solution to this problem, the idea was born of applying machine learning algorithms that allow, based on patients' socioeconomic information, to predict when treatment with Glucantime will be helpful against cutaneous Leishmaniasis when it will not.

Machine learning can be understood as a subset of artificial intelligence, which contains a group of computational algorithms that are essentially in charge of automating experience-based learning; these algorithms are handy when predicting one or more events in which different variables interact. A very relevant characteristic of this type of algorithm is that they are not explicitly programmed to perform any task but are in charge of finding patterns or characteristics relevant to a data set that humans may not even consider naturally.

However, the International Center for Medical Training and Research (CIDEIM), the entity with which the research was carried out, provided a data set that has few records and many attributes, so applying classic machine learning algorithms is somewhat not very promising, as these require a considerable data set in order to learn more effectively. For this reason, the opportunity arises

to apply data augmentation techniques, which allow increasing the size of the data set with probabilistic distributions very similar to those of the real data, this in order to increase the raw material of the machine learning algorithms and consequently improve their ability to predict.

1.2 Objectives

1.2.1 General Objective

Predict the effectiveness of Glucantime as a treatment for cutaneous Leishmaniasis using classic machine learning algorithms and data augmentation techniques.

1.2.2 Specific Objectives

- Identify useful techniques to prepare the data set provided by CIDEIM.
- Experiment with data augmentation techniques to replace the limited existence of clinical data.
- Iteratively and experimentally define hyperparameters values for the machine learning models.
- Experiment Glucantime effectiveness prediction using k-nearest neighbors, logistic regression, support vector machines, neural networks, and random forest.
- Evaluate the performance of machine learning models using standard supervised learning assessment metrics (Precision, Recall, and F1score).
- Propose a graphical interface that allows entering the data of a new patient and, based on them, predict whether or not it is advisable to use Glucantime in the said patient to treat cutaneous Leishmaniasis.

1.3 Justification

Cutaneous Leishmaniasis is a disease neglected by governments because it affects a population with no political, social, or economic influence, a stark reality experienced by thousands of families worldwide, especially in Africa, Asia, and Latin America.

In Colombia, the most common Leishmaniasis species are *Leishmania braziliensis* and *Leishmania panamensis*, species that abound in scattered rural areas of the national territory, areas far from the city where health service is practically inaccessible. In addition, the high stigmatization of people who suffer from Leishmaniasis is worrying since the areas where this disease abound are the scene of armed conflict. Its habitants fear consulting a doctor when they begin to develop the

disease because they will most likely be labeled as guerrillas.

CIDEIM is an international medical research center that has been studying many of these neglected diseases, including cutaneous Leishmaniasis. During this process of around 40 years, it has been observed that treatment against cutaneous Leishmaniasis with Glucantime is not always successful, which has a negative impact on the life quality of patients, since Glucantime is highly toxic [2] and can leave irreparable consequences in people's bodies, totally unnecessary effects if the treatment does not eradicate Leishmaniasis. Therefore, the need arises to apply methods that allow recognizing the patterns of successful and unsuccessful treatments that allow predicting treatment behavior in new patients. The solution proposed and implemented in this research is about applying machine learning over sociodemographic data to provide a computational model to society and the medical and biological community that allows assisting in making decisions regarding when it is feasible or not to face this disease with Glucantime.

In addition, as a computational challenge, there is a small data set with many variables, which was covered by using data augmentation techniques to increase the learning capacity of each of the machine learning models.

Chapter 2

Reference Framework

2.1 Thematic Areas

- Computing Methodologies - Machine learning - Learning Paradigms - Supervised Learning
- Computing Methodologies - Data Augmentation - Tabular Data Augmentation

2.2 Theoretical Background

In this section, each of the stages used in this research work will be listed and described, including data augmentation and machine learning techniques used.

2.2.1 Dataset Exploration

The exploration of the data set is the development phase that allows establishing the first contact with the data set. Therefore, it is essential to know, understand and interpret the present characteristics within the dataset in detail. At this stage, it is necessary to ask questions that allow identifying decisions about what to do with each attribute to maximize its utility. [3, p. 4]

The activities that are usually carried out in the data exploration stage are:

- **Knowing Data Type**: It is done for each attribute in particular in order to know the values that it can take and understand how it is represented.
- **Measures of Central Tendency**: This activity helps to understand in which ranges of values the variables move and the expected behavior of each of them. The most common is to use statistical descriptions such as the mean, the median, and the mode.
- **Dispersion Measures**: They allow us to understand how far the data is from the central tendency, the most common is to use the variance or standard deviation.
- **Correlation Matrix**: It is a table containing the correlation coefficients between different variables simultaneously and assists in investigating the linear dependence between each pair of variables in a quantitative data set.
- **Identification of Missing Values**: It identifies the number of missing values both by rows and columns.

- **Identification of Target Variable Distribution**: It allows to know the class distribution in the target variable.
- **Features Selection**: The attributes set is analyzed to identify the characteristics of most significant relevance to the learning process. Therefore, the elimination or transformation of some attributes can be suggested.

This list of activities allows to methodically explore data before the augmentation or learning process, allowing to identify its viability and its most significant difficulties.

The analysis done in this stage can be summarized in developing a plan to follow in the pre-processing stage.

2.2.2 Pre-processing

In this stage, the plan built in the exploration stage is executed, including data normalization, the transformation of categorical data into numeric, and others. It helps to clean and standardize the data set.

2.2.3 Data Augmentation

These techniques are used primarily when the data set has too many attributes and too few records. Since the small number of records limits the learning process, the most widely used data augmentation methods on tabular sets are:

- **ADASYN**: "The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn." [4, p. 1]
- **SMOTE**: "To create the new synthetic minority class instances, SMOTE first selects a minority class instance $\boldsymbol{a}$ at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors $\boldsymbol{b}$ at random and connecting $\boldsymbol{a}$ and $\boldsymbol{b}$ to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances $\boldsymbol{a}$ and $\boldsymbol{b}$." [5, p. 47]
- **Borderline SMOTE**: This algorithm only oversample the minority class examples that are on the decision border. First, the examples of the decision boundary are discovered; then, synthetic examples are generated from them and added to the original set. [6]
- **Gaussian Mixture Model (GMM)**: The Expectation-Maximization algorithm is used to fit the GMM to the data set. GMM learns the representation of a multimodal data distribution as a combination of unimodal distributions. GMM fits the K Gaussian components to the data set by parameterizing each group's weight, mean, and covariance. After fitting the data

with multiple Gaussian distributions, the results can be used to group any new data points into one of the identified clusters. [7, p. 144]

- **Genetic Algorithm (GA)**: "Implementing GA for synthetic data generation involves random searching in a solution space of possible datasets based on optimization criteria (which are properties of the original dataset) expressed as a fitness function. (...) This means that we can explore different fitness function permutations to establish which are sufficient for a full set of analytical properties to emerge." [8, p. 144]

- **Generative Adversarial Networks (GANs)**: "GANs often comprise a generator and a discriminator that learn simultaneously. The generator tries to capture the potential distribution of real samples and generates new data samples. The discriminator is often a binary classifier, discriminating real samples from the generated samples as accurately as possible. Both the generator and the discriminator can adopt the structure of currently popular deep neural networks. The optimization process of GANs is a minimax game process, and the optimization goal is to reach Nash equilibrium, where the generator is considered to have captured the distribution of real samples."[9, p. 1]

2.2.4 Machine Learning

- **K-nearest neighbors (KNN)**: This algorithm has its foundation in finding the k nearest neighbors (calculating a measure of distance) of the sample to be classified, in such a way that each one of them contributes a vote for the class it belongs to, classifying the sample in the class that has the most votes. [10, p. 423]

- **Random Forest**: "Random Forest is a group of un-pruned classification or regression trees made from the random selection of samples of the training data. Random features are selected in the induction process. Prediction is made by aggregating (majority vote for classification or averaging for regression) the predictions of the ensemble."[11, p. 4]

- **Logistic Regression**: This classification algorithm seeks to estimate the probability that an instance belongs to a particular class. If the estimated probability is greater than or equal to 0.5, the instance is classified in class 1, otherwise in class 0.

  Like linear regression, the logistic regression model computes the weighted sum of the input characteristics, plus a bias term. However, instead of returning the direct result of that operation, it applies a logistic transformation that is essentially an S-shaped curve that returns a number between 0 and 1, which is defined as:

$$f(t) = \frac{1}{1 + e^{-t}} \tag{2.1}$$

  Furthermore, this is its plot:

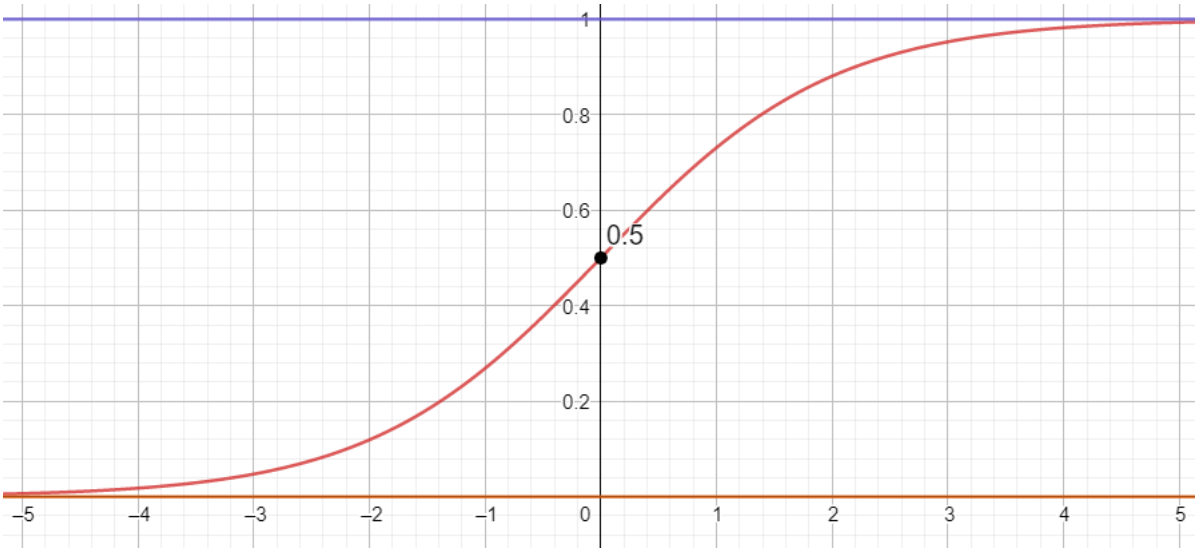


Figure 2.1: Plot of Logistic Output Transformation Function

[12, p. 178]

- **Support Vector Machines (SVM)**: This classification model builds a multidimensional optimal hyperplane to separate two classes and minimize error. It can be easily extended to complex instances that are not linearly separable by mapping the training samples to a larger dimension space where they possibly become linearly separable by using a kernel function [13, p. 256]

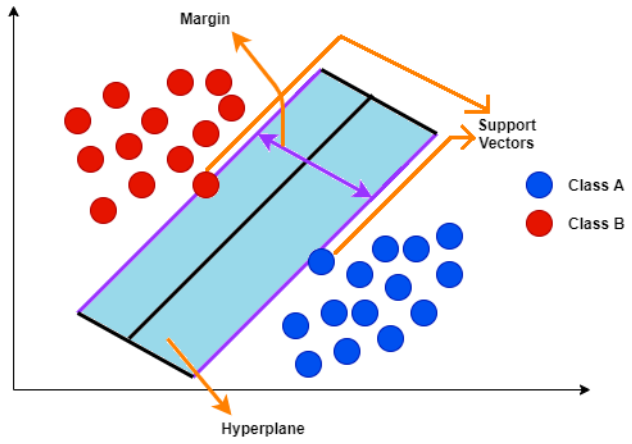


Figure 2.2: Support Vector Machines Explanation Plot

Minimizing the error means making the margin as grater as possible so that the decision bounder has the maximum distance.

- **Multilayer perceptron**: The multilayer perceptron has a more complex architecture than the simple perceptron. The network can contain many intermediate layers between the input and output layers, called hidden layers, which are extremely useful for modeling more complex relationships between input and output variables.

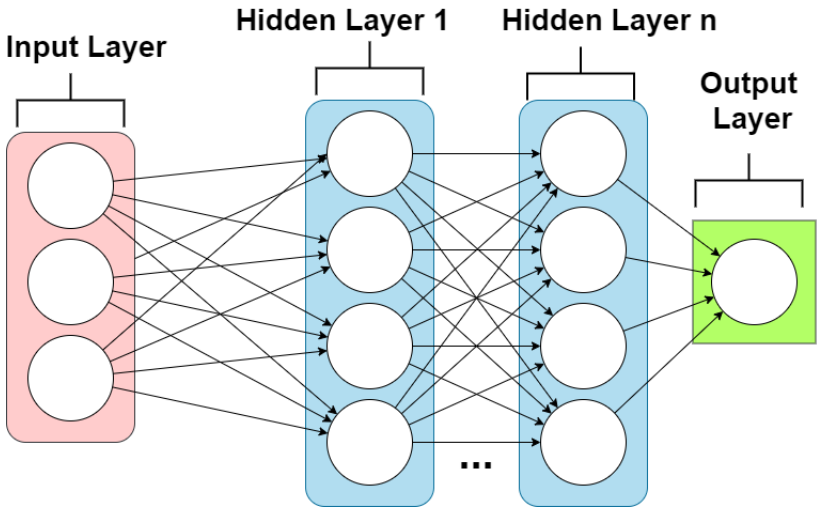


Figure 2.3: Binary Multilayer Perceptron Architecture

These layers perform computations that are transmitted from layer to layer until reaching the output layer. The use of an activation function is required, as in the simple perceptron. However, the most usual for a multilayer perceptron is to use activation functions other than the sign or linear function since this prevents layers from computing linear combinations of the input parameters.

The goal of the neural network learning algorithm is to find a set W that represents the weight of each of the edges of the network in such a way that the total sum of the squared errors is minimized. The backpropagation algorithm solves this optimization problem efficiently. [13, p. 246]

2.2.5 Evaluation Metrics

- **Confusion matrix**: The confusion matrix is used to summarize the number of samples that the classifier model correctly and incorrectly predicted.

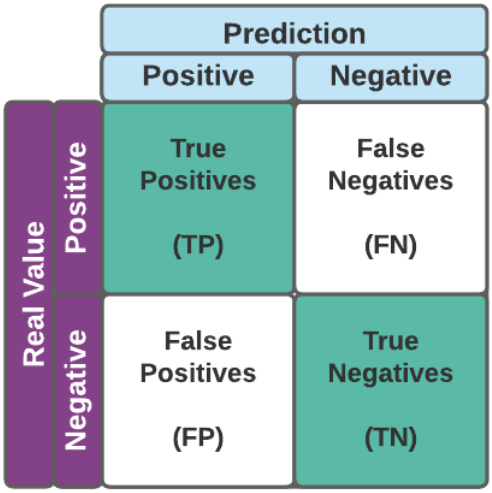


Figure 2.4: Confusion Matrix Heatmap

The most used metrics for supervised learning models emerge from this matrix, which will be listed below:

- **Precision**: This metric allows knowing the proportion between the actual positive samples and those that the classifier predicted as positive, and it is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

- **Sensitivity or Recall**: This metric allows knowing the proportion of samples that the classifier predicted as positive over those whose actual label was positive, and it is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

- **F1 Score**: This metric is a harmonic mean of Precision and Recall that allows us to see in a general way the model's performance, and it is defined as follows:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.4}$$

2.3 Related Work

- **Overcoming Small Data Limitations in Heart Disease Prediction by Using Surrogate Data**: This article explains the results of experimentation using synthetic data to model the prediction of heart disease. The best model (Neural Network) was trained with a set of synthetic data of 60,000 records whose evaluation metrics all converge around 96.7% with an approximate variability of 1%.

  There, it was concluded that when not-so-complex classifiers are used, the result of using Synthetic data is very similar to that obtained with the original data. Also, they detail the advantage of producing high volumes of synthetic data to make it possible to use deep learning models, such as Neural Networks that can model more complex relationships and improve the prediction quality, in this case, over 16%. [14]

- **Efficient Treatment of Outliers and Class Imbalance for Diabetes Prediction**: This article explains experimentation results using synthetic data to model the prediction of diabetes using the SMOTE algorithm to balance the target variable distribution. The data set used had 500 positives and 268 negative samples. The best result was registered with the C4.5 algorithm, which reached an F1 score equal to 0.65, which means an improvement of 3% compared to the result obtained without balancing the target variable distribution. [15]

- **Neural Data Mining for Credit Card Fraud Detection**: This article explains experimentation results using synthetic data to model the sequence of credit card transactions using a confidentiality-based neural network. The data set generated had 5000 synthetic transactions (for every 100 genuine transactions, there is a fraudulent one). The best result was recorded with a ratio of true positives of 91.2% and false positives of 13.35%. [16]

- **Telecom Fraud Identification Based on ADASYN and Random Forest**: This article explains experimentation results using synthetic data to model the identification of fraudulent telecommunications using the ADASYN algorithm to balance the target variable distribution. The data set used had 30.718 records supplied by a Chinese operator, of which 30.000 were considered normal users and 718 fraudulent users. The best model (Random Forest) registered an F1score of 0.93, which increased 11% compared to the algorithm trained without balancing the target variable distribution. [17]

- **Bankruptcy Prediction Using Deep Learning Approach Based on Borderline SMOTE**: This article explains experimentation results using synthetic data to model the identification of companies going bankrupt using the Borderline SMOTE algorithm to balance the target variable distribution. The data set used was collected over five years with the distribution of 41.314 companies that went not bankrupt and 2.091 that did. Eight machine learning models were trained whose evaluation metrics increased 10% to 30% compared to their training without balancing the target variable distribution. [18]

Chapter 3

Dataset Description

3.1 Variables Included in the Study

CIDEIM provided a data set with fifty-eight different demographic variables collected during the research work. In the original structure of the data set, there were records of the patients in five different time instances; the first was the diagnosis, and the last was the end of the treatment. However, there were many variables for a machine learning experiment, especially if there were just twenty-nine registers. Therefore, it was necessary to find a strategy to reduce the number of variables. Given the nature of the problem, a solution that offered a real improvement was a solution able to determine from the first medical appointment if the patient would have a good response with the Glucantime treatment or not. Hence, the study focused on predicting whether Glucantime would work from the first medical appointment, i.e., a diagnosis based on the information gathered on the first patient's visit.

Excluding all the variables that contain information after the patient's first visit, the following variables remained:

Id	Variable Name	Type
1	gender	int64
2	ethnicity	int64
3	age	int64
4	infection_department	object
5	weight	float64
6	height	int64
7	evolution_time	int64
8	number_active_injuries	int64
9	ulcer_horizontal	float64
10	ulcer_vertical	float64
11	ulcer_area	float64
12	injury_horizontal	float64
13	injury_vertical	float64
14	injury_area	float64
15	prescribed_medication	int64
16	glucantime_dosage	float64
17	cure_or_fail	float64

Table 3.1: Initial Dataset Variables and Types

By choosing this approach, the base dataset shape was 17 features and 29 registers.

Subsequently, an explorative analysis of the data was carried out to identify its characteristics and develop a pre-processing plan. Said exploratory analysis was achieved with different activities, which will be listed below:

1. Mean, Standard Deviation, and Mode computation for Numerical Attributes:

Variable Name	Mean	Std	Mode
age	31.38	9.26	20.0
weight	72.97	13.27	58.6
height	166.21	9.78	174.0
evolution_time	6.86	5.01	4.0
number_active_injuries	2.34	1.78	1.0
ulcer_horizontal	20.63	19.37	0.0
ulcer_vertical	17.78	16.11	0.0
ulcer_area	1976.67	2816.72	0.0
injury_horizontal	31.21	25.2	2.0
injury_vertical	29.03	23.3	6.0
injury_area	4464.63	5583.49	12.6
glucantime_dosage	17.09	2.65	20.0

Table 3.2: Basic Statistics over Numerical Variables

According to the table, the variables have different value ranges, which is not ideal for training machine learning algorithms since predicting one direction can be preferred because some variables have more significant numbers than others [19]. Therefore, it was considered essential to normalize the data set.

2. Mode computation for Categorical Attributes:

Variable Name	Mode
gender	1
ethnicity	1
infection_department	Narino
prescribed_medication	1
cure_or_fail	0.0

Table 3.3: Mode Computation over Categorical Variables

From this table, it can be noted that the variable `infection_department` has not a numerical representation, which may not be support for all machine learning algorithms. That is why it was needed to transform it into a numerical representation.

3. Missing Values by Row:

Row Id	Number of Missing Values
9	1
14	1
24	1
25	1
26	1
27	1
28	1

Table 3.4: Missing Values by Row Id Computation

4. Missing Values by Column:

Column	Number of Missing Values
`glucantime_dosage`	2
`cure_or_fail`	5

Table 3.5: Missing Values by Column Computation

The tables presented above show that seven records have missing values in characteristics of high importance as the variable to be predicted and the glucantime dose. Given this lack of information, none of the seven records were helpful, and then they were deleted.

5. Trend of Demographic Variables:

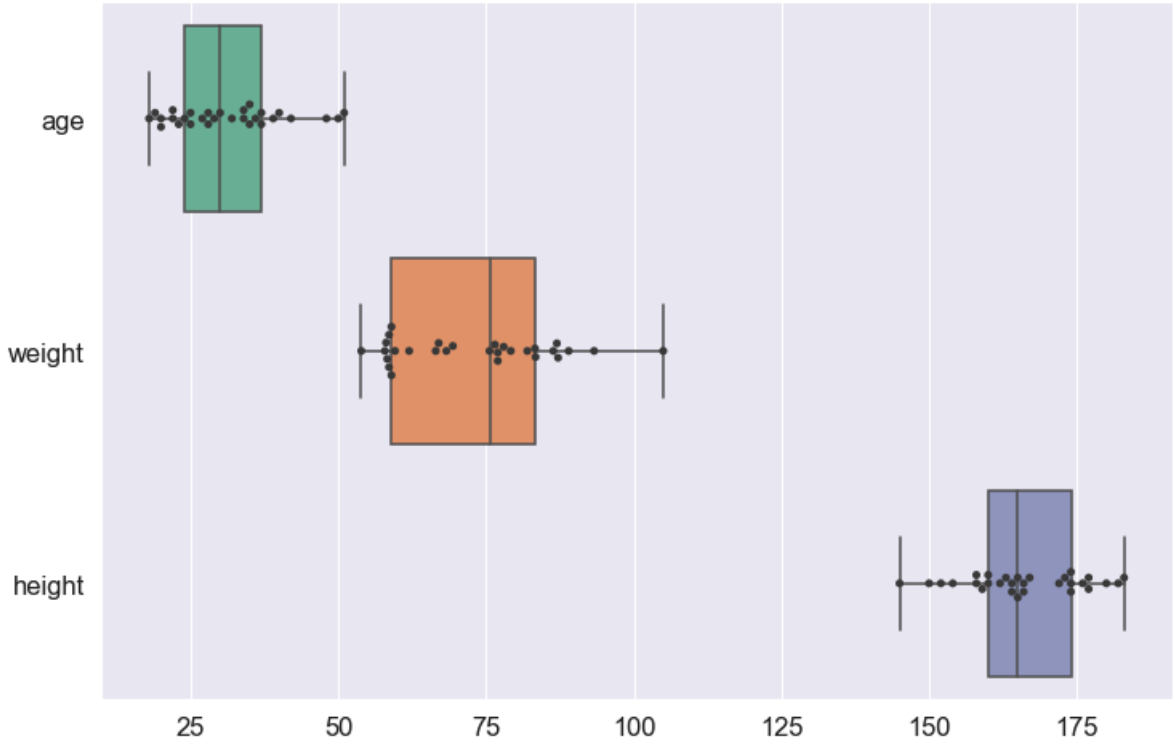


Figure 3.1: Demographic Variables Boxplot

This boxplot shows no presence of outliers in the demographic variables. Most of the data is really close to the median and further from the minimum and maximum.

The Figure 3.2 shows an individual boxplot for each one of the injury-associated variables. There are few outliers in some of them, but not as much as those inside the boxplot, adding that must of those variables will be deleted becouse of Correlation Matrix plot.

6. Trend of Variables Associated with Injury:

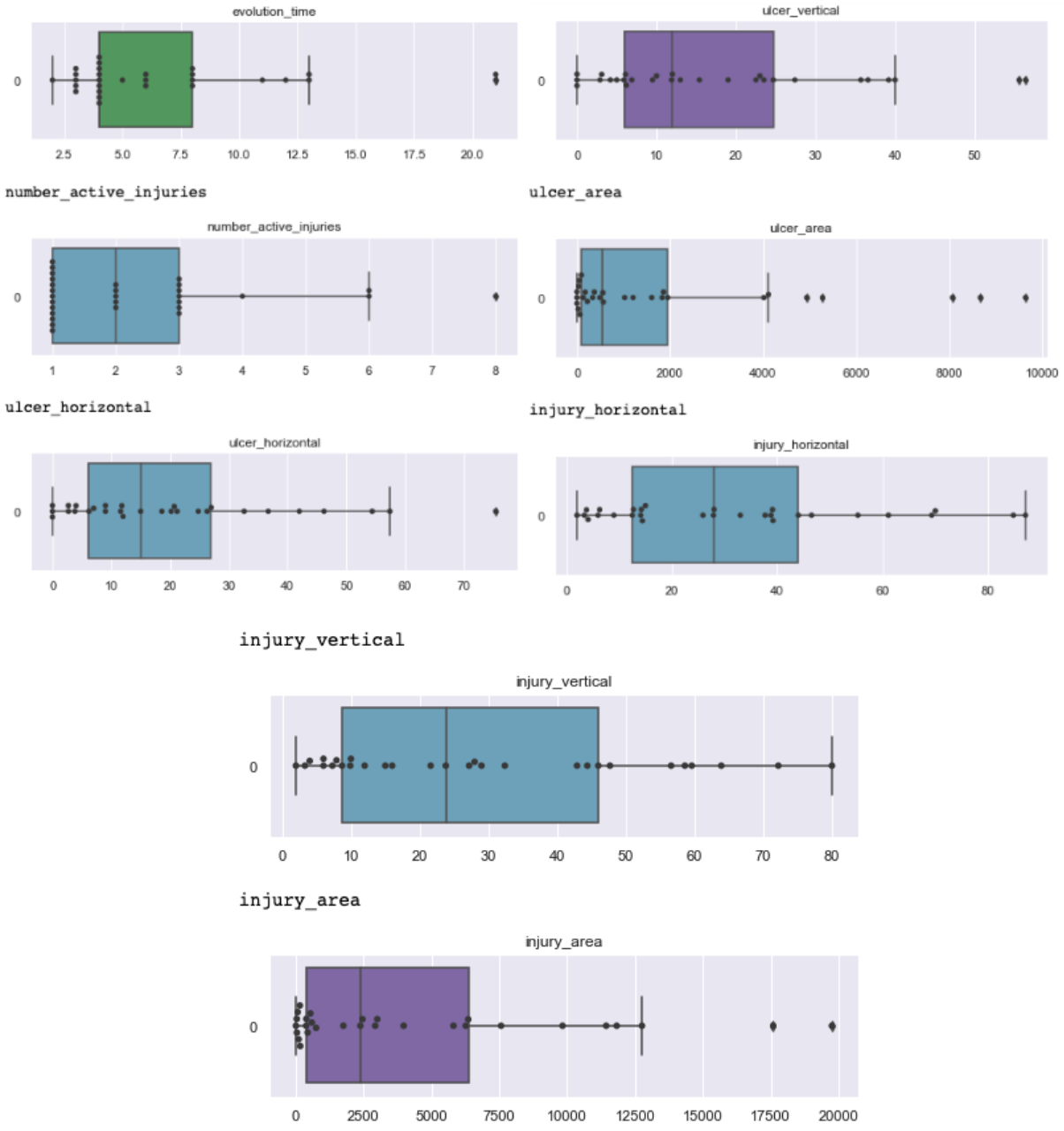


Figure 3.2: Injury Variables Boxplot

7. Correlation Matrix:

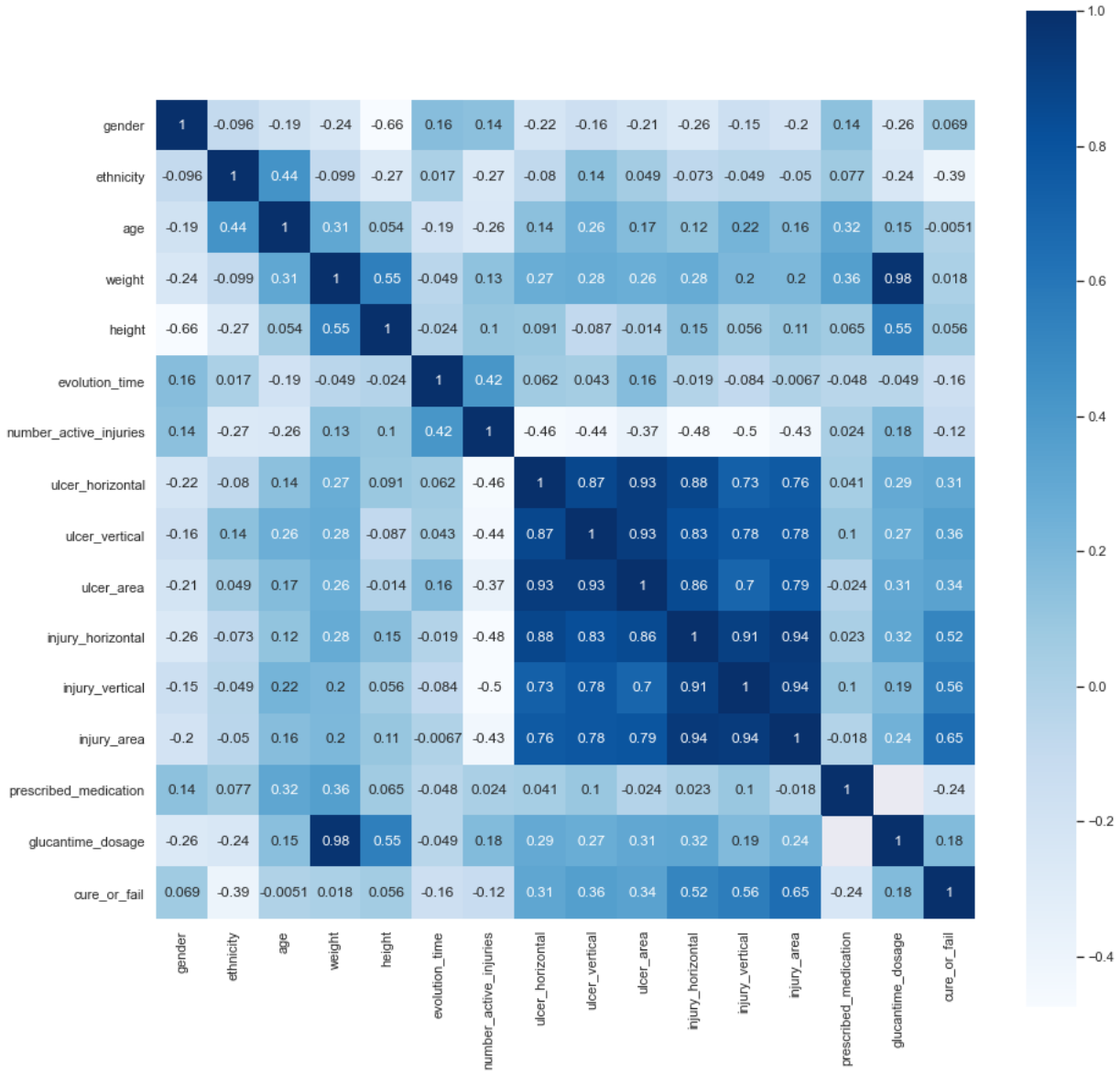


Figure 3.3: Correlation Matrix non-pre-processed

The information provided by figures 3.3 and 3.4 allowed us to know that there are variables with high correlation coefficients, i.e., a high linear dependence relationship. That implies an almost equal contribution of information by both variables. Therefore, it was essential to include in the pre-processing plan the elimination of one variable from each pair of variables with a correlation greater than 0.85 to avoid redundancy in data.

	VAR_1	VAR_2	CORR
0	weight	glucantime_dosage	0.976073
1	ulcer_horizontal	ulcer_vertical	0.870769
2	ulcer_horizontal	ulcer_area	0.929444
3	ulcer_horizontal	injury_horizontal	0.877656
4	ulcer_vertical	ulcer_area	0.928906
5	ulcer_area	injury_horizontal	0.855457
6	injury_horizontal	injury_vertical	0.907958
7	injury_horizontal	injury_area	0.944400
8	injury_vertical	injury_area	0.935635

Figure 3.4: Variables with Correlation greater than 0.85

8. Target Variable Distribution:

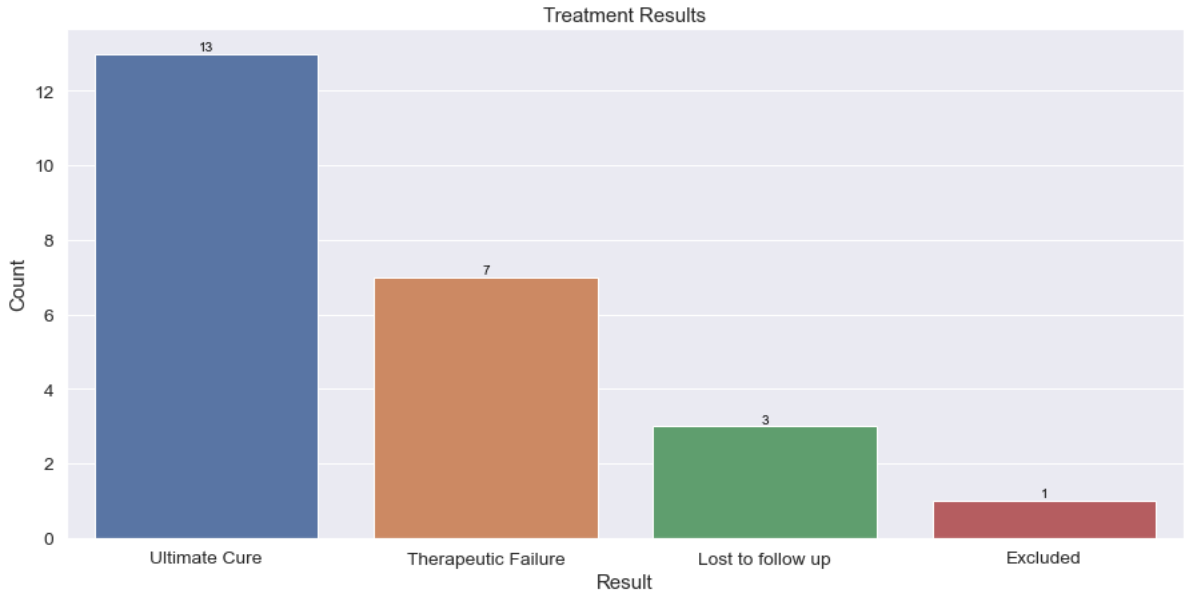


Figure 3.5: Target Variable Distribution non-pre-processed

The target distribution plot helped to identify different classes to those that were intended to be studied (cure or failure). For that reason, the elimination of patients whose target variable was neither cure nor failure was added to the pre-processing plan.

Summarizing all the information provided by each activity, the following **pre-processing plan** was established:

(a) Eliminate non-useful records based on the target variable (Nan, excluded patients, loss patients tracking and non-Glucantime patients treated).

(b) Transform `infection_department` into numerical scale.

(c) Eliminate one of the attributes for each pair of attributes that have a correlation coefficient greater than 0.85.

(d) Normalize the data set.

Chapter 4

Dataset Pre-processing

4.1 Dataset Pre-processing

According to step (a) of the pre-processing plan, the classes other than cure or failure present in the target variable were eliminated, obtaining the following class distribution:

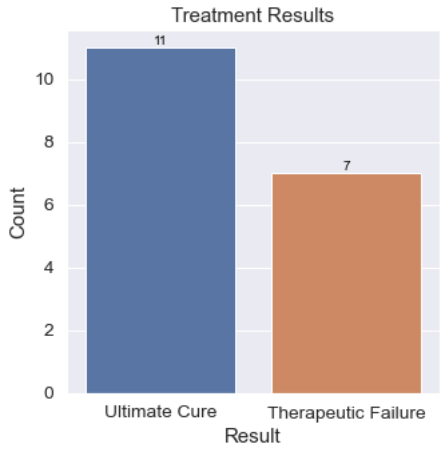


Figure 4.1: Target Variable Distribution pre-processed

Likewise, for step (b), the text representation was changed to numeric for the variable `infection_department`, as shown below:

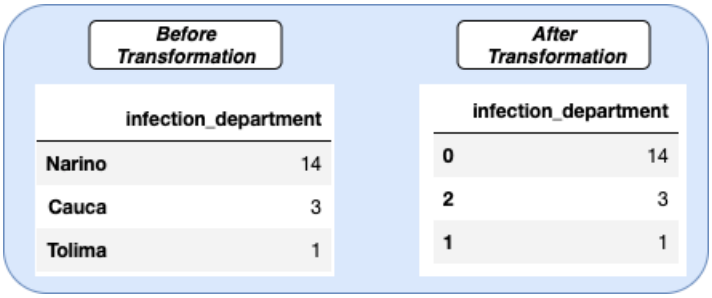


Figure 4.2: Variable infection_department transformation

For step (c), a variable was eliminated for each pair of variables that correlated greater than 0.85, obtaining the following correlation matrix:

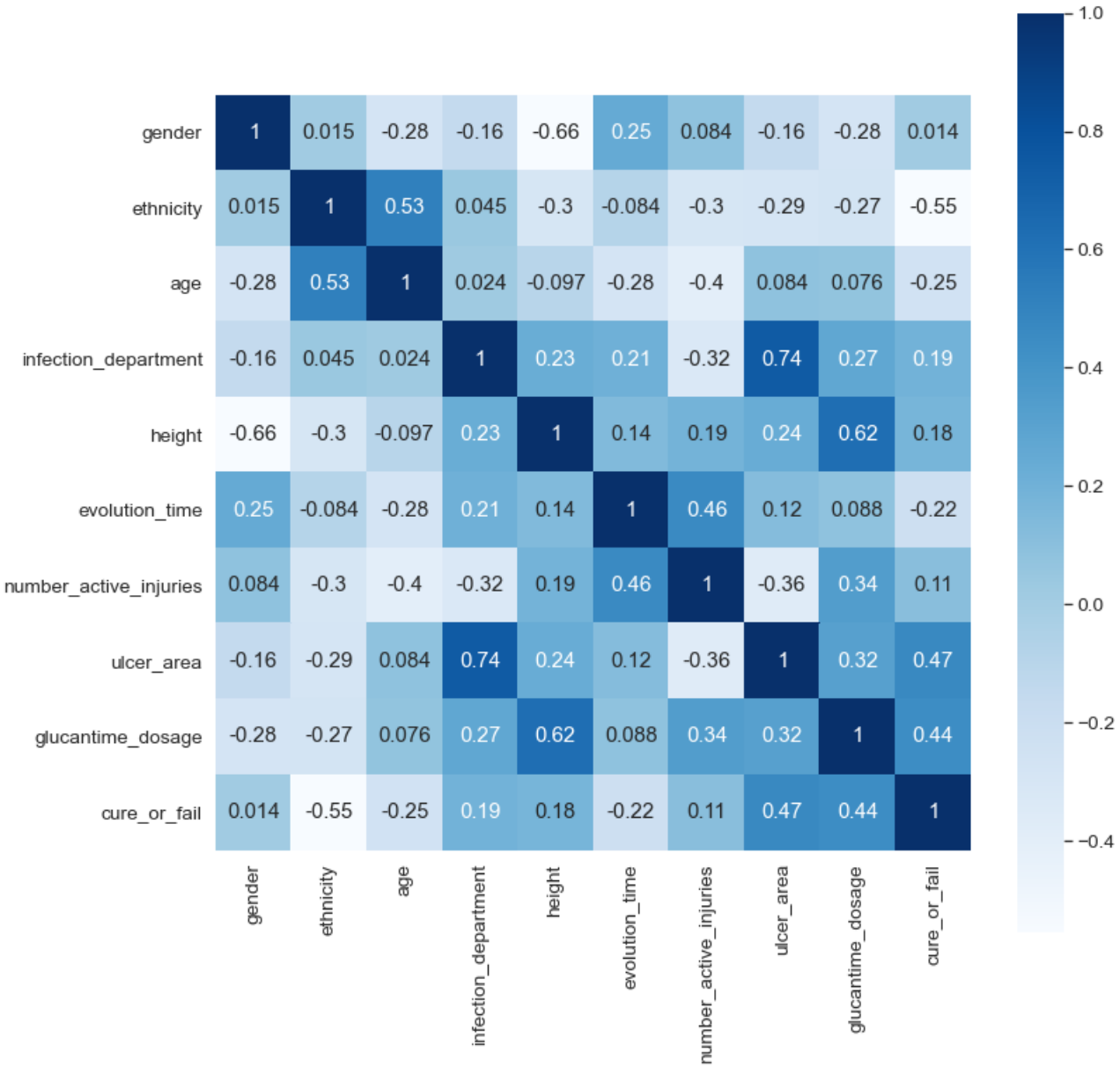


Figure 4.3: Correlation Matrix pre-processed

Finally, the normalization of the data was carried out, and a pre-processed data set with ten attributes and eighteen records was obtained, as shown in the following figure:

```
Columns:

gender
ethnicity
age
infection_department
height
evolution_time
number_active_injuries
ulcer_area
glucantime_dosage
cure_or_fail

The pre-processed dataset has 18 rows and 10 columns
```

Figure 4.4: Pre-processed Dataset Shape

Chapter 5

Data Augmentations Techniques

According to the shape specifications of the data set after pre-processing, there are eleven examples of patients who cure and seven patients who fail. The target variable has an imbalance; in addition, the number of examples per class is minimal. Therefore, there were essentially two good options, class balancing or data augmentation. Class balancing would mean increasing the minority class, but in a low proportion, due to the small amount of data from the majority class, which does not significantly change since the test and train ratio is almost the same. Therefore, the better option was to perform data augmentation in order to have the ability to generate high-quality data and keep the original data only for machine learning algorithms testing.

5.1 ADASYN

The ADASYN module available in the `over_sampling` module of the imblearn library for python was used to implement this data augmentation model [20]. Five patients with characteristics of a patient who cures and thirteen patients with features of a patient who fails were generated. The data set generated with this technique is used in the machine learning chapter, training the algorithms with only synthetic data and testing them with the real data.

5.2 SMOTE

The SMOTE module available in the `over_sampling` module of the imblearn library for python was used to implement this data augmentation model [20]. Fifteen patients with characteristics of a patient who cures and fifteen patients with features of a patient who fails were generated. The data set generated with this technique is used in the machine learning chapter, training the algorithms with only synthetic data and testing them with the real data.

5.3 Borderline SMOTE

The BorderlineSMOTE module available in the `over_sampling` module of the imblearn library for python was used to implement this data augmentation model [20]. Fifteen patients with characteristics of a patient who cures and fifteen patients with features of a patient who fails were generated. The data set generated with this technique is used in the machine learning chapter, training the algorithms with only synthetic data and testing them with the real data.

5.4 Gaussian Mixture Model

The GaussianMixture module available in the mixture module of the sklearn library for python was used to implement this data augmentation model [21]. Sixteen patients with characteristics of a patient who cures and fourteen patients with features of a patient who fails were generated.The data set generated with this technique is used in the machine learning chapter, training the algorithms with only synthetic data and testing them with the real data.

5.5 Genetic Algorithm

This algorithm learns evolutionarily according to the aptitude function, which verifies at all times that the individual generated in time t meets the criteria of eachone of the dataset original features.Below the algorithm configuration is listed:

Item	Explanation
Number of Generations	100
Individuals per Generation	5.000
Mutation Probability	0.05
Crossing Strategy	Given two Individuals ind1 = [a1, a2, a3, a4, a5, a6,..., an] ind2 = [b1, b2, b3, b4, b5, b6,..., bn] A pivot is randomly selected: pivot = 4 then new individuals are generated splitting originals by pivot: new1 = [a1, a2, a3, a4, b5, b6,.., bn] new2 = [b1, b2, b3, b4, a5, a6,...,an]

Table 5.1: Genetic Algorithm Settings and Strategies

Listing 5.1: Fitness Function - Pseudocode

```
def fitness(individual = [a1,a2,a3,...,an], min, max, average, std):
    # min => minimum of each original feature
    # max => maximum of each original feature
    # average => average of each original feature
    # std => standar deviation of each original feature

    score = 0
```

```
for each individual feature:
    if min <= ind_feature <= max:
        score = score + 5

        if average - std <= ind_feature <= average + std:
            score = score + 10


return score
```

The fitness function essentially rewards a synthetic individual with five points if (for a given characteristic) its value is between the minimum and maximum of the original characteristic. In addition, if the individual is in the range mean minus dispersion and mean plus dispersion, it is awarded an additional 10 points.

5.6 Generative Adversarial Network

A GAN was used to generate samples of patients who cure and another for patients who fail. These data were generated separately, and later they were mixed in the same dataset.

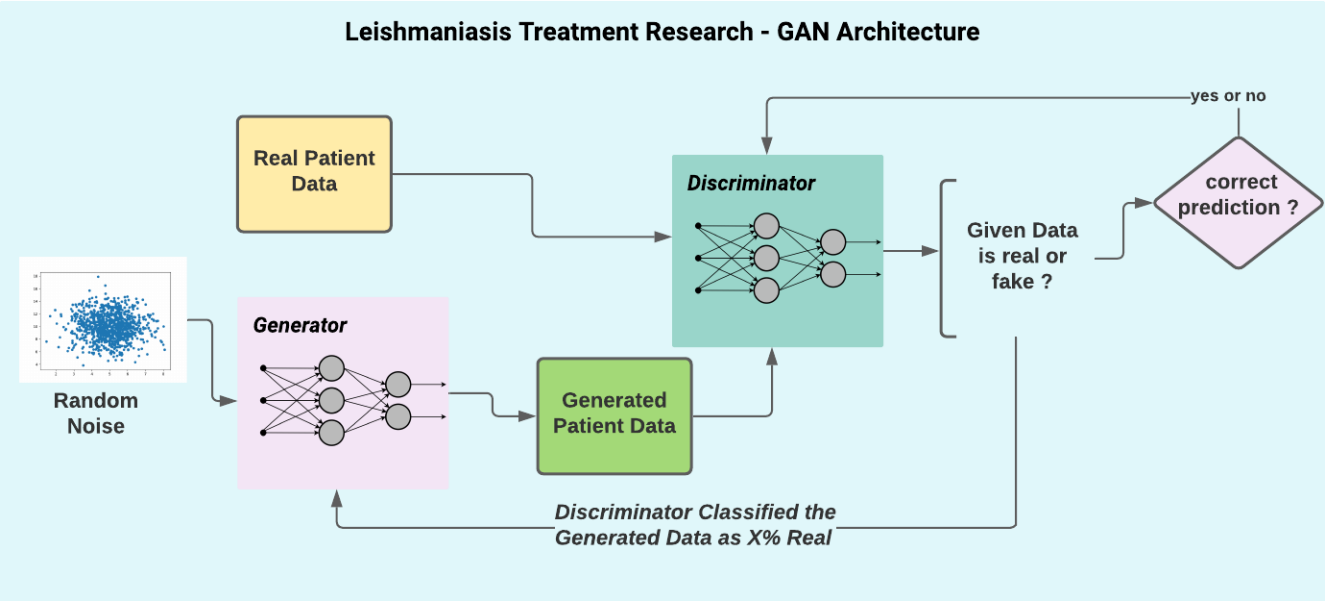


Figure 5.1: GAN Architecture

The GAN built has two neural networks, a generator that is the one that produces the synthetic data of the patients and another discriminator that is responsible for distinguishing whether the data is real or not. After a while, the generator can generate data closely to the real one that it is

challenging for the discriminator to distinguish.

This GAN was implemented with the base code elaborated in [22], whose purpose was also to generate synthetic data of patients in another context.

The dataset generated with this technique is used in the machine learning chapter, training the algorithms with only synthetic data and testing them with the real data.

Chapter 6

Machine Learning

6.1 Hyperparameter Tuning

In this stage, the hyperparameter optimization was carried out using the Grid Search module of the Sklearn library available for python. A broad-spectrum was tested for each one of the machine learning models, as listed below :

6.1.1 Logistic Regression

Parameter	Description	Value(s)
penalty	Type of regularization term to use	['l1', 'l2', 'elasticnet']
C	Determines the strength of the regularization	[0.001, 0.009, 0.01, 0.09, 1, 5, 10, 25]
solver	Algorithm to use in the optimization problem	['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
tol	Tolerance for stopping criteria	[0.001, 0.0001, 1e-05]
fit intercept	Specifies if a constant (bias or intercept) should be added to the decision function	[True, False]
max iter	Maximum number of iterations taken for the solvers to converge	[50, 100, 150]

Table 6.1: Logistic Regression Hyperparameter Grid

6.1.2 K Nearest Neighbor

Parameter	Description	Value(s)
n neighbors	Number of neighbors to use	[2, 3, 4, 5, 6]
weights	Weight function used in prediction	['uniform', 'distance']
algorithm	Algorithm used to compute the nearest neighbors	['ball tree', 'kd tree', 'brute']

Table 6.2: K Nearest Neighbor Hyperparameter Grid

6.1.3 Random Forest

Parameter	Description	Value(s)
criterion	The function to measure the quality of a split	['gini', 'entropy']
max depth	The maximum depth of the tree	[1, 11]
min samples split	The minimum number of samples required to split an internal node	[2, 4, 6]
min samples leaf	The minimum number of samples required to be at a leaf node	[1, 2, 3, 4, 5]
max features	The number of features to consider when looking for the best split	['sqrt', 'log2']
max leaf nodes	Grow trees with max leaf nodes in best-first fashion	[1, 11, 21, 31, 41]

Table 6.3: Random Forest Hyperparameter Grid

6.1.4 Support Vector Machine

Parameter	Description	Value(s)
C	Determines the strength of the regularization	[1, 10, 100, 1000]
kernel	Specifies the kernel type to be used in the algorithm	['linear', 'poly', 'rbf', 'sigmoid']
gamma	Kernel coefficient for 'rbf', 'poly' and 'sigmoid'	[0.001, 0.0001]

Table 6.4: Support Vector Machine Hyperparameter Grid

6.1.5 Multilayer Perceptron

Parameter	Description	Value(s)
hidden layer sizes ,	MLP Layer Size (a,b,c) neuron with 3 layers a = units in layer 1 b = units in layer 2 c = units in layer 3	[(1, 1, 1), (3, 3, 3), (5, 5, 5), (7, 7, 7), (9, 9, 9), (11, 11, 11), (13, 13, 13), (15, 15, 15) (17, 17, 17), (19, 19, 19)]
solver	The solver for weight optimization	['sgd', 'adam']
learning rate	Learning rate schedule for weight updates	['constant', 'adaptive']
activation	Activation function for the hidden layer	['logistic', 'tanh', 'relu']
alpha	L2 penalty parameter	[0.0001, 0.05]

Table 6.5: Multilayer Perceptron Hyperparameter Grid

The hyperparameters optimization was performed using the GridSearch module over the data set generated by the SMOTE technique. Obtaining finally for the models the following best params:

6.1.6 Best Params

Model	Hyperparameters Values
Logistic Regresion	C: 5 fit intercept: True max iter: 50 penalty: l1 solver: liblinear tol: 0.001
K nearest neightbors	algorithm: ball tree n neighbors: 2 weights: distance
Random Forest	criterion: gini max depth: 31 max features: sqrt max leaf nodes: 11 min samples leaf: 3 min samples split: 6
Support Vector Machines	C: 10 gamma: 0.001 kernel: linear
Multi Layer Perceptron	activation: tanh alpha: 0.0001 hidden layer sizes: (9, 9, 9) learning rate: constant solver: adam

Table 6.6: Best Params Search Results

By using these optimized parameters were built all the machine learning models shown in the next section.

6.2 Train - Test - Split

For the training and testing of each machine learning algorithm, the same data sets were always used, training only with synthetic data and testing with real data.

The following table shows the proportion of training data (generated with each data augmentation algorithm) and test data (which is the total data from the original data set):

Data Augmentation Model	Train - Synthetic	Test - Real
ADASYN	Cure: 5 | Fail: 13	Cure: 11 | Fail: 7
SMOTE	Cure: 15 | Fail: 15	
Borderline SMOTE	Cure: 15 | Fail: 15	
Gaussian Mixture Model	Cure: 16 | Fail: 14	
Genetic Algorithm	Cure: 194 | Fail: 200	
Generative Adversarial Network	Cure: 30 | Fail: 30	

Table 6.7: Train - Test - Split Table

6.3 Machine Learning Experiments

6.3.1 Without Data Augmentation

As an initial reference experiment, the construction of machine learning models without data augmentation was carried out. Those models were trained and tested by partitioning the dataset in training (60%) and test (40%). Due to the limited amount of data available and especially to the notable difference between the minority class and the majority class, this experiment was done with this single partition, which made the execution of a cross-validation experiment impossible.

The results of this experiment are shown below:

Machine Learning Model	Precision	Recall	F1-Score
MLPClassifier	0.67	0.5	0.57
KNeighborsClassifier	1.0	0.25	0.4
LogisticRegression	0.0	0.0	0.0
RandomForestClassifier	0.0	0.0	0.0
SVC	0.0	0.0	0.0

Table 6.8: Machine Learning Results Without Data Augmentation

As shown above, the original data did not perform very well and was not enough for a representative machine learning experiment, so, there was a great opportunity to apply data augmentation and machine learning combination.

6.3.2 With Data Augmentation

All machine learning experiments presented below were done using synthetic data generated in the Data Augmentation Step.

6.3.2.1 K Nearest Neighbors

For this experiment, it was used the generated data with each one of the data augmentation models. Then, the k-nearest neighbor model was trained only with synthetic data and tested with the real data, achieving the following results:

Data Augmentation Model	Precision	Recall	F1-Score
ADASYN	0.78	1.0	0.88
Borderline SMOTE	0.78	1.0	0.88
SMOTE	0.86	0.86	0.86
Generative Adversarial Network	1.0	0.71	0.83
Genetic Algorithm	0.62	0.71	0.67
Gaussian Mixture Model	0.5	0.57	0.53

Table 6.9: K Nearest Neighbors - Results

A notable improvement was found in the KNeighborsClassifier model performance compared to the previously trained model without data augmentation, especially with the ADASYN and Borderline SMOTE data augmentation techniques, reaching an F1score of 0.88, which is more than double than the obtained without data augmentation (0.4)

6.3.2.2 Logistic Regression

For this experiment, it was used the generated data with each one of the data augmentation models. Then, the Logistic regression model was trained only with synthetic data and tested with the real data, achieving the following results:

Data Augmentation Model	Precision	Recall	F1-Score
ADASYN	0.78	1.0	0.88
SMOTE	0.78	1.0	0.88
Generative Adversarial Network	0.64	1.0	0.78
Borderline SMOTE	0.6	0.86	0.71
Genetic Algorithm	0.57	0.57	0.57
Gaussian Mixture Model	0.5	0.57	0.53

Table 6.10: Logistic Regression - Results

Logistic regression was not a successful model in the non-data augmentation experiment. It performed poorly, reporting an F1score of 0, which means that all predicted values were badly predicted. With data augmentation, things changed for a big, reaching an F1score of 0.88 with ADASYN and SMOTE techniques; it shows the value these techniques, combined with the logistic model, can apport.

Additionally, by analyzing the logistic regression coefficients, it can be observed which are the variables that influence with greater significance in the construction of the model, which is interesting when having good results because it could reference which variables are the most important for the problem that is being solved, in this case, the variables of highest significance are: ethnicity, time of evolution, glucantime dose, height and ulcer area 1. This can be seen in the following figure:

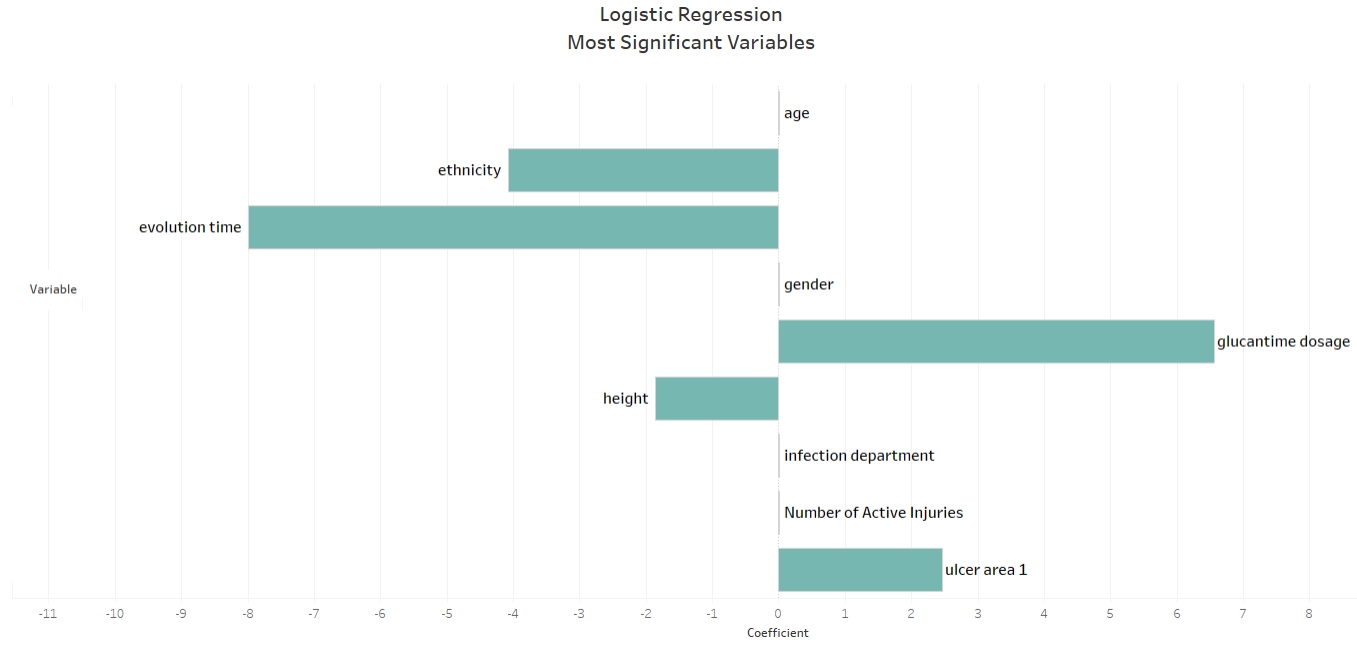


Figure 6.1: Most Significant Variables - Logistic Regression

6.3.2.3 Support Vector Machines

For this experiment, it was used the generated data with each one of the data augmentation models. Then, the Support Vector Machine model was trained only with synthetic data and tested with the real data, achieving the following results:

Data Augmentation Model	Precision	Recall	F1-Score
SMOTE	0.88	1.0	0.93
Generative Adversarial Network	0.88	1.0	0.93
Borderline SMOTE	0.78	1.0	0.88
ADASYN	0.7	1.0	0.82
Genetic Algorithm	0.57	0.57	0.57
Gaussian Mixture Model	0.43	0.43	0.43

Table 6.11: Support Vector Machine - Results

Support Vector Machines was not a successful model in the non-data augmentation experiment. It performed poorly, reporting an F1score of 0, which means that all predicted values were badly predicted. With data augmentation, things changed for a big, reaching an F1score of 0.93 with SMOTE and Generative Adversarial Network techniques; it shows the value these techniques, combined with the Support Vector Machines model, can apport.

6.3.2.4 Neural Networks

For this experiment, it was used the generated data with each one of the data augmentation models. Then, the Neural Networks model was trained only with synthetic data and tested with the real data, achieving the following results:

Data Augmentation Model	Precision	Recall	F1-Score
ADASYN	0.88	1.0	0.93
SMOTE	0.78	1.0	0.88
Borderline SMOTE	0.7	1.0	0.82
Generative Adversarial Network	0.71	0.71	0.71
Genetic Algorithm	0.57	0.57	0.57
Gaussian Mixture Model	0.43	0.43	0.43

Table 6.12: Neural Networks - Results

A notable improvement was found in the Multilayer Perceptron model performance compared to the previously trained model without data augmentation, especially with the ADASYN and SMOTE data augmentation techniques, reaching F1scores of 0.93 and 0.83 respectively, which is much better than the obtained without data augmentation (0.57)

6.3.2.5 Random Forest

For this experiment, it was used the generated data with each one of the data augmentation models. Then, the Random Forest model was trained only with synthetic data and tested with the real data, achieving the following results:

Data Augmentation Model	Precision	Recall	F1-Score
SMOTE	0.86	0.86	0.86
Borderline SMOTE	0.7	1.0	0.82
ADASYN	0.58	1.0	0.74
Generative Adversarial Network	0.71	0.71	0.71
Genetic Algorithm	0.8	0.57	0.67
Gaussian Mixture Model	0.44	0.57	0.5

Table 6.13: Random Forest - Results

Random Forest was not a successful model in the non-data augmentation experiment. It performed poorly, reporting an F1score of 0, which means that all predicted values were badly predicted. With data augmentation, things changed for a big, reaching an F1score of 0.86 with SMOTE technique; it shows the value these technique, combined with the Random Forest model, can apport.

Additionally, by analyzing the Random Forest most significant variables, it was found that the variables of highest significance are: glucantime dose, ulcer area 1 and evolution time. This can be seen in the following figure:

Figure 6.2: Most Significant Variables - Random Forest

Chapter 7

Comparative Analysis

7.1 Between Data Augmentation Techniques

In order to show the results more compactly, the following chart was created, which allows breaking down the results of the five machine learning models grouped by each data augmentation technique. In this way, it is possible to observe with an easy glance the best results according to each data augmentation technique.

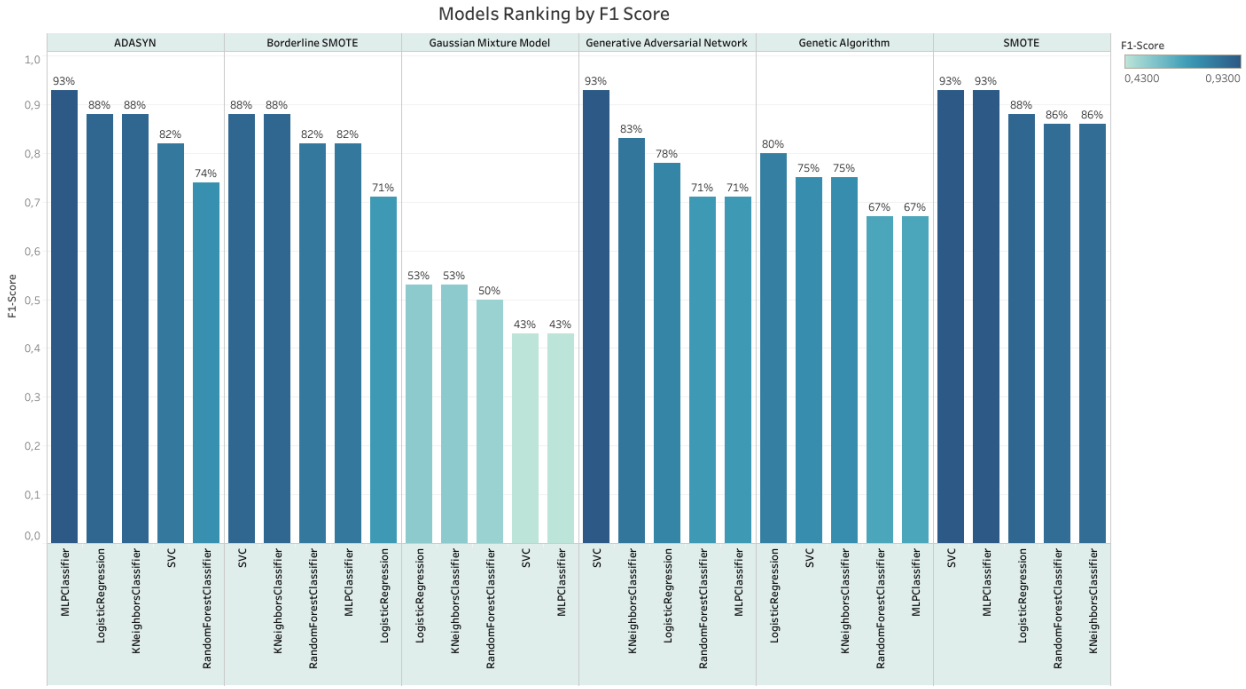


Figure 7.1: Comparative Results Plot

- The SMOTE algorithm in this particular problem shows excellent stability. It behaves similarly with each machine learning algorithm, keeping the results of the F1-score between 86% and 93% in each of the experiments carried out.

- With the Gaussian Mixture Model algorithm, strong stability was also achieved but in a medium score. The metrics reached with the dataset augmented by this algorithm were the lowest, supported by an F1-score that does not exceed 53%.

- The ADASYN algorithm was also able to generate a data set that yielded excellent results but with a little less stability than SMOTE, keeping the F1-score between 74% and 93%.

- As expected, the variant of SMOTE, Borderline SMOTE, also had good results. However, it was not as excellent as the original version, evidenced in F1 scores between 71% and 88%. Then, there are few class instances over the decision boundary, which prevent this particular problem from further enhancing results by using this data augmentation algorithm.

- The Generative Adversarial Network was the data augmentation algorithm that surprised the most with its results, mainly because it is an algorithm that has conventionally been used only within computer vision context. Now it was possible to adapt it to this particular tabular data set, finding that the generated data set combined with machine learning algorithms can keep an F1-score between 71% and 93%, results as good as those of ADASYN and SMOTE.

- The Genetic Algorithm has a medium F1 score compared to the other models of data augmentation; it remains between 67% and 80%, which can be supported by how complex it is to write aptitude functions on data that humans do not understand entirely, especially to describe a particular behavior or patterns over the patients' dataset.

In a complementary way, the results presented in the previous chart can be discerned in a tabular way below:

Machine Learning - Data Augmentation Model	Precision	Recall	F1-Score
ADASYN			
MLPClassifier	0.88	1	**0.93**
LogisticRegression	0.78	1	0.88
KNeighborsClassifier	0.78	1	0.88
SVC	0.7	1	0.82
RandomForestClassifier	0.58	1	0.74
SMOTE			
SVC	0.88	1.0	**0.93**
MLPClassifier	0.88	1.0	0.93
LogisticRegression	0.78	1.0	0.88
KNeighborsClassifier	0.86	0.86	0.86
RandomForestClassifier	0.86	0.86	0.86
Borderline SMOTE			
KNeighborsClassifier	0.78	1.0	**0.88**
SVC	0.78	1.0	0.88
RandomForestClassifier	0.7	1.0	0.82
MLPClassifier	0.7	1.0	0.82
LogisticRegression	0.6	0.86	0.71
Gaussian Mixture Model			
LogisticRegression	0.5	0.57	**0.53**
KNeighborsClassifier	0.5	0.57	0.53
RandomForestClassifier	0.44	0.57	0.5
SVC	0.43	0.43	0.43
MLPClassifier	0.43	0.43	0.43
Genetic Algorithm			
LogisticRegression	0.75	0.86	**0.8**
KNeighborsClassifier	0.67	0.86	0.75
SVC	0.67	0.86	0.75
RandomForestClassifier	0.62	0.71	0.67
MLPClassifier	0.62	0.71	0.67
Generative Adversarial Network			
SVC	0.88	1.0	**0.93**
KNeighborsClassifier	1.0	0.71	0.83
LogisticRegression	0.64	1.0	0.78
RandomForestClassifier	0.71	0.71	0.71
MLPClassifier	0.71	0.71	0.71

Thanks to the previous shown comparative diagrams, we found that several models had an excellent performance, which leads to very satisfactory conclusions for the study, and principally because of the best-models results, which are presented below:

7.2 Best Models

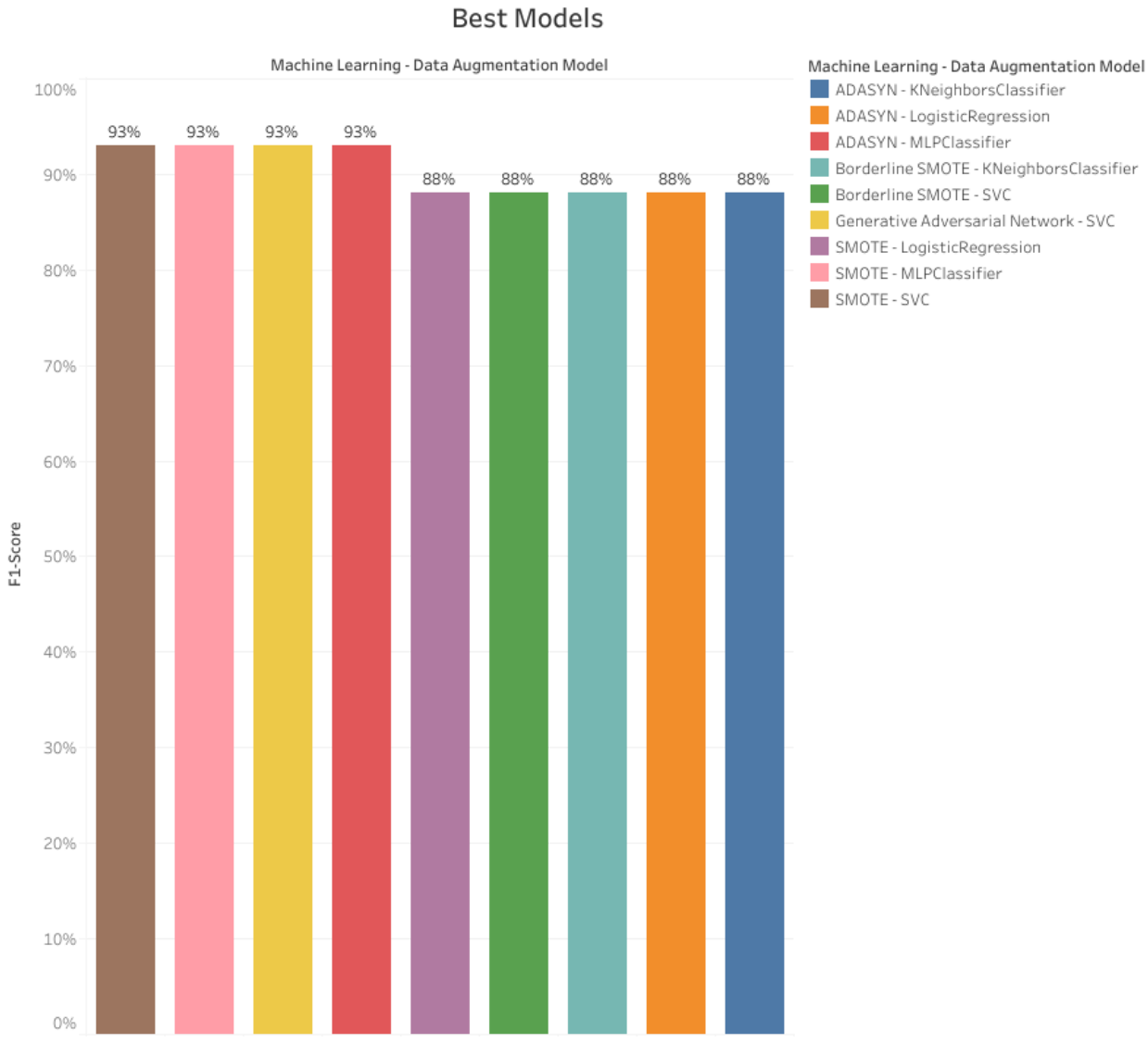


Figure 7.2: Best Models

Chapter 8

Additional Experiment

Concluding the investigation, specifically in the last month, CIDEIM provided us with a new data set. To take advantage of this new data arrival, we decided to use the same code from the experiments shown above with minor modifications to adapt it. As they are essentially identical experiments, the explanation of this new experiment will be very brief focused on showing the results.

The new dataset has 247 records; 189 belong to cured patients and 58 to fail-treatment patients.

The first thing that was done was to extract the variables equivalent to those used in the first experiment from the new dataset. Later, the same names used in experiment one were put to the variables extracted from the new data set to make compatible scripts already developed; then, the data augmentation was carried out, and finally, the training and testing of the machine learning algorithms.

The results of the machine learning algorithms without data augmentation are shown below:

Machine Learning Model	**Precision**	**Recall**	**F1-Score**
MLPClassifier	0.67	0.24	0.35
KNeighborsClassifier	0.52	0.44	0.48
LogisticRegression	0.0	0.0	0.0
RandomForestClassifier	0.88	0.6	0.71
SVC	0.9	0.24	0.39

Table 8.1: Machine Learning Results Without Data Augmentation

These algorithms significantly improved their performance compared to experiment one without data augmentation. There is even a prediction with an F1 score of 0.71 without data augmentation, which is an excellent indicator and has huge potential to improve by using data augmentation techniques.

The data augmentation was needed to modify in order to generate larger samples compared to the first experiment due to the statistical diversity provided by the new data set, which made it more difficult to extract specific characteristics of the patients; for this, it was necessary to train with a greater amount of data machine learning algorithms, as shown below:

Data Augmentation Model	**Train - Synthetic**	**Test - Real**
ADASYN	Cure: 374 | Fail: 371	Cure: 189 | Fail: 58
SMOTE	Cure: 376 | Fail: 378	
Borderline SMOTE	Cure: 376 | Fail: 378	
Gaussian Mixture Model	Cure: 194 | Fail: 562	
Genetic Algorithm	Cure: 378 | Fail: 378	
Generative Adversarial Network	Cure: 378 | Fail: 378	

Table 8.2: Train - Test - Split Table - Dataset-2

The results of the machine learning algorithms trained with the newly generated data and tested with real data are shown in the following table and figure:

Machine Learning - Data Augmentation Model	Precision	Recall	F1-Score
ADASYN			
KNeighborsClassifier	0.93	0.98	**0.96**
RandomForestClassifier	0.81	0.83	0.82
MLPClassifier	0.49	0.78	0.6
LogisticRegression	0.36	0.62	0.46
SVC	0.29	0.55	0.38
SMOTE			
KNeighborsClassifier	0.86	0.98	**0.92**
RandomForestClassifier	0.73	0.91	0.81
MLPClassifier	0.69	0.91	0.79
SVC	0.4	0.72	0.52
LogisticRegression	0.38	0.6	0.46
Borderline SMOTE			
KNeighborsClassifier	0.81	0.95	**0.87**
RandomForestClassifier	0.75	0.78	0.76
MLPClassifier	0.51	0.83	0.63
LogisticRegression	0.33	0.71	0.45
SVC	0.3	0.88	0.45
Gaussian Mixture Model			
LogisticRegression	0.18	0.57	**0.28**
KNeighborsClassifier	0.18	0.57	0.28
RandomForestClassifier	0.18	0.57	0.28
SVC	0.18	0.57	0.28
MLPClassifier	0.18	0.57	0.28
Genetic Algorithm			
RandomForestClassifier	0.33	0.71	**0.45**
KNeighborsClassifier	0.32	0.72	0.44
MLPClassifier	0.26	0.84	0.4
SVC	0.27	0.72	0.39
LogisticRegression	0.26	0.83	0.39
Generative Adversarial Network			
KNeighborsClassifier	0.39	0.52	**0.45**
LogisticRegression	0.33	0.64	0.44
MLPClassifier	0.33	0.55	0.41
SVC	0.31	0.55	0.4
RandomForestClassifier	0.31	0.48	0.38

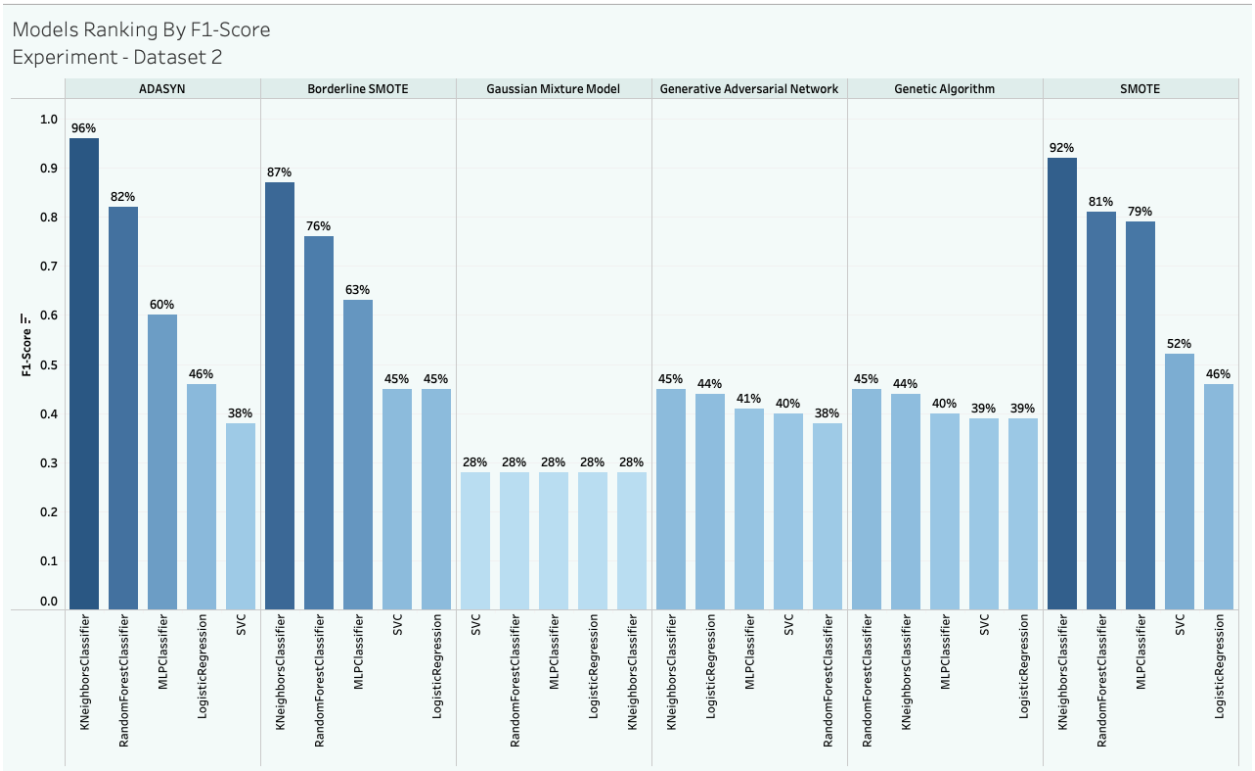


Figure 8.1: Comparative Results Plot - Dataset-2

Chapter 9

Conclusions

- The effectiveness of Glucantime as a treatment for Cutaneous Leishmaniasis was predicted using data augmentation and machine learning algorithms with F1-scores higher than 90%.
- The combination of data augmentation models and machine learning algorithms became very powerful given the problem conditions. The learning process was strengthened by extracting significant characteristics from the original data to generate new data and training the machine learning models.
- In problems where data lack is a common factor, such as the medical field, generating more data with data augmentation techniques is highly useful. This is based on the improvement of the machine learning algorithms trained with augmented data compared to those trained only with real data, showing an improvement in the performance of up to 25%, which adds much value and can significantly impact the patient's health and life.
- Despite the difficulties that arose due to the lack of data to implement a prediction model, it was possible to follow a successful route that made it possible to take advantage of the data of each real patient. This route allowed to improve the predictions, as they were awful due to the little training that could be done without the data augmentation. It also helped to solve the impossibility of cross-validating because of minimum resulting partitions. For this reason, the route mentioned above was strategically developed so that the machine learning algorithms were trained only with synthetic data and were tested only with real data.

9.1 Future Work

There are future works that would be interesting to carry out, which are listed below:

- Include genetic patient's behavior variables into the demographic corpus since this can contribute even more information of each one of the individuals to the learning models, allowing to execute current techniques and explore others that may get better results.
- If possible, it would be interesting to collect data from a greater number of patients. This would allow learning more about the patterns in the general population and their behavior with Glucantime as a treatment for Cutaneous Leishmaniasis, further generalizing the prediction models and possibly reducing data augmentation.

- For more advanced work, it is possible to think about involving variables that may significantly analyze the health patient state. For example, work with images of people's skin lesions, variables such as body temperature, blood type, and always keeping the proportion between the number of variables and observations so that the prediction keeps feasible to carry out.

Bibliography

[1] P. Zambrano, "Vigilancia y Análisis Del Riesgo En Salud Pública Protocolo De Vigilancia En Salud Pública Leishmaniasis," *World health organization*, vol. 02, p. 4, 2017.

[2] A. Masmoudi, N. Maalej, M. Mseddi, A. Souissi, H. Turki, S. Boudaya, S. Bouassida, and A. Zahaf, "Glucantime® par voie parentérale: Bénéfice versus toxicité," *Medecine et Maladies Infectieuses*, vol. 35, no. 1, pp. 42–45, 2005.

[3] A. C. Muller and S. Guido, *Introduction to Machine Learning with Python and Scikit-Learn*. 2015.

[4] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning," no. 3, pp. 1322–1328, 2008.

[5] T. Ryan Hoens and N. V. Chawla, "Imbalanced datasets: From sampling to classifiers," *Imbalanced Learning: Foundations, Algorithms, and Applications*, pp. 43–59, 2013.

[6] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Lecture Notes in Computer Science*, vol. 3644, no. PART I, pp. 878–887, 2005.

[7] S. Misra, H. Li, and J. He, *Robust geomechanical characterization by analyzing the performance of shallow-learning regression methods using unsupervised clustering methods*. Elsevier Inc., 2020.

[8] Y. Chen, M. Elliot, and J. Sakshaug, "A genetic algorithm approach to synthetic data production," *ACM International Conference Proceeding Series*, vol. 29-30-Augu, pp. 0–3, 2016.

[9] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F. Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[10] S. Agarwal, *Data mining: Data mining concepts and techniques*. 2014.

[11] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random Forests and Decision Trees," *International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 272–278, 2012.

[12] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media.

[13] Tan Steinbach Kumar, *Introduction to Data Mining (New International Editon)*. No. September, 2013.

[14] A. Sabay, L. Harris, V. Bejugama, K. Jaceldo-Siegl, and K. Jaceldo-Siegl DrPH, "Overcoming Small Data Limitations in Heart Disease Prediction by Using Surrogate Data," *SMU Data Science Review*, vol. 1, no. 3, p. 12, 2018.

[15] N. Nnamoko and I. Korkontzelos, "Efficient treatment of outliers and class imbalance for diabetes prediction," *Artificial Intelligence in Medicine*, vol. 104, no. February, p. 101815, 2020.

[16] T. Guo and G. Y. Li, "Neural data mining for credit card fraud detection," *Proceedings of the 7th International Conference on Machine Learning and Cybernetics, ICMLC*, vol. 7, no. July, pp. 3630–3634, 2008.

[17] X. L. H. S. Chao Lu, Shaofu Lin, "Telecom Fraud Identification Based on ADASYN and Random Forest," pp. 447–452, 2020.

[18] S. Smiti and M. Soui, "Bankruptcy Prediction Using Deep Learning Approach Based on Borderline SMOTE," *Information Systems Frontiers*, vol. 22, no. 5, pp. 1067–1083, 2020.

[19] Mahbubul Alam, "Data normalization in machine learning," 2020.

[20] G. Lemaˆıtre Lemaˆıtre, F. Nogueira, and C. K. Aridas char, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," tech. rep., 2017.

[21] F. Pedregosa FABIANPEDREGOSA, V. Michel, O. Grisel OLIVIERGRISEL, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot andÉdouardand, a. Duchesnay, and F. Duchesnay EDOUARDDUCHESNAY, "Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot," tech. rep., 2011.

[22] Vivek Maskara, "Generating Tabular Synthetic Data Using GANs," 2020.