UIMP
Universidad Internacional
Menéndez Pelayo

Máster Interuniversitario en Ciencia de Datos

ANÁLISIS DEL RENDIMIENTO DE MODELOS DE APRENDIZAJE AUTOMÁTICO SOBRE DATOS ANONIMIZADOS
(Analyzing the performance of machine learning models on anonymized data)

Trabajo de Fin de Master para acceder al
MÁSTER EN CIENCIA DE DATOS

Autora: Carmen Marcos Sánchez de la Blanca
Directora: Judith Sáinz-Pardo Díaz
Co-Director: Álvaro López García
Julio - 2023

Resumen

La gran cantidad de datos abiertos disponibles hace necesario el estudio y desarrollo de técnicas que garanticen la seguridad de dichos datos para su posterior tratamiento y análisis. En concreto, el estudio de las técnicas de anonimización se centra en el análisis de la distribución de los cuasi-identificadores y atributos sensibles en una base de datos. Existen muchas técnicas que pueden aplicarse, cada una de de ellas pueden evitar distintos tipos de ataques.

En este estudio se exploran tres técnicas de anonimización clásicas, su bases teóricas y diferentes tipos de ataques que previenen: *k-anonimato*, *l-diversidad* y *t-cercania*. Además, se utilzan diferentes herramientas para garantizar la fiabilidad de estas técnicas, que son aplicadas a diferentes niveles sobre dos conjuntos de datos en abierto, tras pre definir diferentes jerarquías sobre los cuasi-identificadores.

A continuación, se estudiará el rendimiento de una batería de modelos de Machine Learning aplicado en los datos anonimizados presentados anteriormente. Se generarán un amplio rango de resultados experimentales, variando la técnica de anonimización empleada, así como el nivel establecido.

Todo el código es desarrollado en Python, y distribuido mediante un repositorio de datos en abierto. Además, los datasets han sido anonimizados utilizando el Software ARX.

Palabras clave: anonimización, aprendizaje automático, análisis de rendimiento, privacidad, k-anonimato.

Abstract

The large amount of open data available makes it necessary to study and develop techniques that guarantee its security for processing and analysis. Specifically, the study of anonymization techniques focuses on analyzing the distribution of the quasi-identifiers and sensitive attributes in a database. There are numerous techniques that can be applied, each of which can prevent different types of attacks.

The present study explores three classical anonymity techniques, their theoretical basis and the kind of attacks they prevent: *k-anonymity*, *ℓ-diversity* and *t-closeness*. Specifically, different tools are used to ensure the reliability of these techniques which are applied at various levels on two open-access datasets, after pre-defining different hierarchies for the quasi-identifiers.

Next, the performance of a battery of machine learning models applied on the anonymized data is studied. A wide range of experimental results is carried out, varying the anonymization technique employed, as well as the level established.

All the code developed is written in Python and is distributed through an open source repository. In addition, the datasets were anonymized using the ARX Software.

Keywords: anonymization, machine learning, performance analysis, privacy, k-anonymity.

Acronyms

AB Adaptive Boosting.

ANN Artificial Neural Network.

AUC Area under the ROC Curve.

DL Deep Learning.

GB Gradient Boosting.

GDPR General Data Protection Regulation.

KNN k-nearest neighbors algorithm.

KRR Kernel Ridge Regression.

ML Machine Learning.

QI Quasi-identifier.

RF Random Forest.

ROC Curve Receiver operating characteristic Curve.

SA Sensible attribute.

SVM Support vector machine.

Contents

Chapter 1

Introduction

1.1 Motivation and objectives

Large amounts of data are generated daily that need to be processed for subsequent analysis using machine learning techniques. However, not all of this data can be treated in the same way. Some of them may contain sensitive information, so they need to be handled with different anonymization techniques according to the laws established in the corresponding territory.

The objective of this work, "*Analyzing the Performance of Machine Learning Models on Anonymized Data*", is to evaluate the effectiveness of anonymization techniques on different datasets containing sensitive information. By analyzing the performance of machine learning models on anonymized data, we aim to understand the trade-offs between privacy preservation and data utility for analysis.

This study aims to review three of the most used existing anonymization techniques, apply them to diverse data sources with sensitive information, create and assess machine learning models suitable for analyzing both the original and anonymized data and quantifying the impact of anonymization on model performance.

In particular, we will analyze how the results of the different machine learning models scale when applying the different techniques, also varying the selected parameters. A wide range of machine learning models will be studied, from the most classical ones such as linear regression, k-nn or svm, to ensamble methods such as random forest, gradient boosting, and neural networks among others.

In the following, the state of the art regarding both data anonymization and ML models and these comparative studies are presented.

1.2 State of the art

Nowadays, machine learning models are increasingly being used to analyze sensitive data, such as financial data, financial transactions and medical records among others [1]. How-

ever, the use of machine learning (ML) models on this kind of data raises several concert about data privacy and how the personal information is going to be used. Anonymization is a technique that is used to protect the privacy of the individuals and addressing these problems at the same time that still allows machine learning models to be successful while training with the modified data.

1.2.1 Data anonymization

Data anonymization is essential in processing sensitive data when analyzing given information. In fact, it is known as the process of removing or altering data in a way that protects the privacy of the individuals depicted in the data. This concept dates back to the early days of computing. In the 1960s, researchers began to develop various techniques to remove personal information from datasets to preserve the privacy of the individuals in the data [2].

When handling information from databases or datasets with sensitive information, the individual responsible for the anonymization process must identify the attributes and classify them based on their nature and sensitivity. We distinguish between identifiers, quasi-identifiers, sensitive attributes, and insensitive attributes. Moreover, during this process, some attributes may need to be stratified into hierarchies or even removed to achieve the desired level of anonymity.

The relevance of anonymization is self-evident in many areas where sensitive data are available and when machine learning (ML) models are potential tools for carrying out inference processes. However, it should be noted that an excessively strict level of anonymity may compromise the usefulness of the data for processing and inference [3]. This can occur either because a large number of records had to be eliminated to reach the desired level of anonymization or because the hierarchies applied to the quasi-identifiers dilute their initial information. It is necessary to achieve a balance between the level of data anonymization and the utility of the data for analysis.

However, it was not until the 1970s that one of the first regulations concerning data anonymization came into force. In the United States of America, the government enforced a law that made anonymization of datasets compulsory before their publication due to several privacy violations that had occurred in the past. From that date until the year 2000, new anonymization techniques were improved and developed to avoid different attacks such as inference, re-identification, or aggregation attacks (see “The cost of privacy” [4]).

There are several data anonymization techniques currently used, each with its own strengths and weaknesses. Some of the most common techniques include k-anonymization, l-diversity, and t-closeness, which we will discuss in this analysis.

In addition, regulatory compliance is another challenge to consider in these days. With the increasing focus on data protection and privacy, there are regulations, such as the General Data Protection Regulation (GDPR), that impose strict requirements on the handling of personal data. ML models trained on anonymized data must comply with these regula-

tions to avoid legal and ethical issues. GDPR is one of the main regulations is the General Data Protection Regulation. This regulation was established on May 24, 2016, and became enforceable on May 25, 2018.

The General Data Protection Regulation aims to ensure data protection and privacy for all inhabitants of the European Union and its Economic Area. Its main objectives include ensuring transparency in the use of personal data by companies and imposing stronger consent requirements for the use of our data [5].

To overcome these challenges, researchers and practitioners need to carefully evaluate the quality of anonymized data, implement robust anonymization techniques, and ensure compliance with relevant regulations. Only by addressing these challenges can ML models effectively utilize anonymized data while maintaining privacy and data integrity.

1.2.2 Machine Learning Models

Another key factor to consider in this study is the use of machine learning models to study the performance with both anonymized and non-anonymized data.

Machine learning and deep learning are a rapidly evolving fields, with new models or variations of existing ones being developed all the time. In this study, we will implement some of the most popular and widely used models, which have been studied in the master's program. These models include *Linear Regression*, *k-Nearest Neighbors*, *Support Vector Machines*, *Classification Trees*, *Random Forest*, *Adaptive Boosting*, *Gradient Boosting*, and *Artificial Neural Networks*. Additionally, we will use the Stacked Generalization Ensemble, which has not been covered in the master's program, to address a specific case that arose during the analysis.

The development of the theory associated with most of these algorithms and models dates back to the 1960s. However, due to the lack of infrastructure and computing power at that time, their implementation was not feasible. It was not until the early 2000s when the development of cloud computing allowed us to access computing resources on demand, making the implementation of these models much more accessible.

In terms of the application of these techniques in the financial and medical fields, we can confidently state that the future of these domains is closely intertwined with machine learning. In finance, models are currently used for a wide variety of tasks, including fraud detection, risk assessment, and many others. Similarly, in the healthcare industry, machine learning is being used for diagnosing various pathologies (such as cancer detection, glaucoma detection, and cardiac arrhythmia detection) [6] [7], as well as for prognosis and treatment planning [8] based on personalized medicine.

1.2.3 Performance analysis over ML models

The performance of machine learning (ML) models on anonymized data can be analyzed using various methods, including accuracy, the F1-score, recall, and precision among others. However, there are several challenges associated with the performance of ML models on anonymized data. These challenges include data quality problems, privacy concerns, and regulatory compliance.

One significant challenge is data quality. Previous works have demonstrated that data quality can have a substantial impact on the accuracy of models [4, 9, 3]. If the anonymized data is of poor quality or has been overly anonymized, the models may struggle to learn accurate representations of the underlying data. This issue can arise not only from over-anonymization but also from incomplete datasets, lack of data accuracy, or bias.

Therefore, in this study, we will analyze the performance of various machine learning models using two distinct datasets with different levels of anonymization. The datasets employed in this analysis exhibit notable differences, with one dataset containing a large number of records and the other dataset consisting of significantly fewer records. Further information about the uses cases will be given in Chapter 3.

1.3 Structure of the work

This study is structured into different chapters. The first chapter introduces the motivation behind the study and provides an overview of the current state of the art. Additionally, it covers fundamental concepts essential for understanding the work, including data anonymization and machine learning.

Chapter 2 delves into the foundations of classic anonymity tools. It presents key concepts, various anonymization techniques, and algorithms used in the field.

In Chapter 3, the datasets utilized in the study are presented, along with the software employed for the analysis. Furthermore, an introduction to the different machine learning models utilized to evaluate the performance of the various anonymization techniques is provided.

The fourth chapter presents the results obtained from the analysis, along with a detailed examination of each use case. It explores the data exploration process and delves into the intricacies of the anonymization procedure.

Finally, the study concludes with a summary of the results and suggests future avenues for expanding upon this work.

Chapter 2

Classic anonymity tools

2.1 Main concepts

As it has been stated in the state of the art,

Definition 2.1.1. Identifiers Are attributes of a database which uniquely identify individuals within a dataset or a database. Some example are the name and surname, personal ids or email addresses.

Definition 2.1.2. Quasi-identifiers Are attributes that have the potential to identify individuals indirectly when combined. They may not be unique identifiers on their own, but when combined with external attributes, they can reveal sensitive information about individuals. This can be done with different attacks, depending of the level of anonymity that we have achieved when out anonymized dataset don't meet several levels o anonymity of a certain technique.

Definition 2.1.3. Sensitive attributes Are attributes that contain personal or sensitive information about individuals that we want to protect from the unveiling. In other words, it is the focus of privacy concerns and need to be protected during data analysis. Examples of sensitive attributes include medical conditions, financial data, or any information that can lead to potential harm or discrimination if exposed.

Definition 2.1.4. Insensitive attributes Are attributes within a dataset that does not contain sensitive or personal identifiable information about an individual. These attributes are not associated with privacy concerns and typically include demographic information, timestamps, or other non-sensitive data.

Definition 2.1.5. Equivalence classes
Are groups of individuals in a dataset that share the same values for quasi-identifiers. These classes are formed by grouping rows of the database whose quasi-identifying attributes are indistinguishable from each other. Anonymization techniques aim to ensure that individuals within the same equivalence class cannot be distinguished from one another, thus preserving privacy.

In the dataset proposed in Example 2.1.1, we can create equivalence classes based on the combination of gender, age and blood type as presented in the following.

ID	Gender	Age	Blood Type	Disease
1	Male	20	O+	Hepatitis
2	Female	25	A-	Allergy
3	Male	30	B+	Hepatitis
4	Female	34	O+	Cold
5	Male	39	A+	Hepatitis

Table 2.1: Example table for the explanation of the different anonymity techniques (raw)

Example 2.1.1. In Table 2.2, we have the *gender*, *age*, *blood type* and *disease* of some individuals, as well as the *ID* of each of them.

Here we have a dataset of some patients of the same hospital that includes the attributes *ID*, *gender*, *age*, *blood type* and *disease*. The patient *ID* will be unique for each patient and the *disease* is a sensitive attribute which must not be known by an attacker. On the other hand, the rest of the attributes, all combined, could be used to identify the patients (QI).

2.2 Most Commonly Used Anonymization Techniques

As it has been stated before, the main idea behind anonymization is that it covers the techniques and the processes of modifying data, so it is more difficult than in the starting point to identify the people associated with the original information. This involves removing or altering identifiable attributes while maintaining the data's usefulness and analytical value. Next, we will discuss the most commonly used anonymization techniques and models that will be used throughout this dissertation.

In this work, we will focus on providing definitions and explanations specifically for the case of a single sensitive attribute (SA), as the datasets used in our analysis will contain only one attribute that is considered sensitive. However, it is worth noting that in real-world scenarios, datasets can have multiple sensitive attributes. In such cases, additional considerations and techniques need to be applied to protect the privacy of individuals effectively.

To address datasets with multiple sensitive attributes, one approach is to apply anonymization techniques independently to each sensitive attribute while considering the relationships and dependencies among them. This can involve treating each sensitive attribute separately, applying appropriate anonymization methods, and considering the potential interactions between the sensitive attributes during the anonymization process. In some cases, it may be necessary to prioritize the protection of certain sensitive attributes over others based on their level of sensitivity or the potential harm that can arise from their disclosure. This will be futher explored at the end of this chapter.

2.2.1 k-anonymity

k-anonymity is an anonymization technique that ensures that, given an anonymized dataset, each equivalence class is formed by k rows and, therefore, that each individual in the anonymized dataset is indistinguishable from at least $k-1$ other individuals with respect to the set of quasi-identifiers. This technique prevents linkage and re-identification attack [10].

While linkage attack involves combining anonymized data with external datasets or auxiliary information to re-identify individuals, re-identification does it “re-identifying" individuals in the anonymized dataset, linking them back to their original identities.

Example 2.2.1. Let's exemplify this technique using the dataset given in Table 2.2:

ID	Gender	Age	Blood Type	Disease
1	Male	20	O+	Hepatitis
2	Female	25	O+	Allergy
3	Male	30	B-	Hepatitis
4	Female	24	O+	Cold
5	Male	39	A-	Hepatitis
6	Male	27	O-	Cold

Table 2.2: Example table for the explanation of the different anonymity techniques (raw data).

In order to achieve *k-anonymity*, we can replace the combination of *age*, *gender*, and *blood type* for each patient with a generalization or suppression technique (will be explained in Chapter 3). In this way, we ensure that for each patient in the database, each combination of the QI is verified for, at least, $k-1$ other individuals in the database.

For instance, to achieve *2-anonymity* ($k=2$), we can generalize the *age* attribute into groups of 10 years each. In addition, we can also generalize the blood groups as follows:

ID	Gender	Age	Blood Type	Disease
1	Male	20-29	O	Hepatitis
2	Female	20-29	O	Allergy
3	Male	30-39	A/B	Hepatitis
4	Female	20-29	O	Cold
5	Male	30-39	A/B	Hepatitis
6	Male	20-29	O	Cold

Table 2.3: Example table verifying *k-anonymity* with $k=2$

After this anonymization,all equivalence classes contain at least 2 records, ensuring that each patient is indistinguishable from at least another patient in the database.

As exposed in the previous example, *k-anonymity* can be achieved by grouping records into equivalence classes, where each class has at least k records sharing the same values for certain attributes, known as quasi-identifiers. By ensuring that each individual is

indistinguishable within their equivalence class, anonymity is preserved, and the identification of specific individuals becomes difficult. This can be achieved using hierarchies, also known as taxonomy trees. A hierarchy is a structure that organizes the values of an attribute into different levels of generality. For example, for the attribute "studies" the hierarchy could have levels such as "undergraduate" "post-graduate," and "illiterate." By generalizing attributes to higher levels in the hierarchy, the utility of the data is preserved while specific details of individuals are concealed.

Some windly known algorithms that can be implemented to achieve *k-anonymity* are *mondrian*, *incognito* and *data fly* [11]:

- **Mondrian:** This algorithm achieves *k-anonymity* by creating partitions (Mondrian boxes) that contain a minimum of k records and have similar generalization levels for the quasi-identifiers. This prevents individual identification while maintaining the overall statistical properties of the data. When can divide this algorithm into four main steps:

  First of all, we order the quasi-identifier attributes based on their generalization hierarchy. Then we select an attribute from the ordering, and randomly choose a splitting point along its generalization hierarchy, partitioning the original dataset. We split the original mondrian box in two until each box contains at least k records or cannot be splitted further with our k level restrictions. The next step is to generalize the values of the quasi-identifiers to ensure the *k-anonymity*, so we choose one of the levels of the hierrarchy. Finally we repeat this process for each new box until all of them satisfy the *k-anonymity* requirement.

- **Incognito:** This algorithm achieves the set *k-anonymity* levels by using a generalization lattice. This means that, after doing the attribute generalization, groups must be created by assigning records that have similar generalized quasi-identifier values, ensuring that each group satisfies the *k-anonymity* requirement. After that, comes the anonymization step, were we generalize and/or suppress sensitive attribute values within each group. Finally, we evaluate the anonymized dataset to make sure it meets the *k-anonymity* level desired. This is done totally iterative until we meet the criteria. This algorithm produces an optimal solution [11].

- **Data fly:** This algorithm aims to achieve *k-anonymity* by generalizing and suppressing sensitive information in a private table. It is achieved following this steps:

  First of all, we create a frequency list. It must be a list of unique sequences of values from the quasi-identifier in the dataset, along with their frequencies and associated identifiers. For doing it, we iterate through the dataset, adding frequency entries to the list for each quasi-identifier. Therefore, we can now determine which sequences have a frequency below the desired *k-anonymity* threshold. If there are more sequences with counts below the threshold than allowed, generalize the data by collapsing attribute values with the most distinct values. Repeat the process until the count is within the threshold. All in all, we suppress the necessary number of tuples to comply with the allowed loss factor while preserving *k-anonymity*. We reconstruct the modified frequency list to create a generalized version of the original dataset.

2.2.2 ℓ-diversity

ℓ-diversity is another data anonymization technique used to protect data privacy. The main idea of this technique is that for each equivalence class in the dataset there are at least ℓ different values of that sensitive attribute. In this way we ensure that the sensitive attribute values of each group of individuals in the dataset are diverse enough.

Example 2.2.2. Coming back to check the example 2.2 introduced previously, we can only ensure that that dataset verifies ℓ-diversity with ℓ=1. However, note that if the disease of the patients 3 or 5 where different, ℓ-diversity with ℓ=2 would be verified. For instance, if the individual 3 had Allergy instead of Hepatitis. After this change, every equivalence class would have at least 2 different values for the sensible attribute (*disease*).

2.2.3 t-closeness

In this section we present *t-closeness*, which is another data anonymization technique used to protect data privacy, focusing both in the quasi-identifiers and the sensitive attributes. The goal of *t-closeness* is to ensure that every row in the dataset is not distinguishable based on the sensitive attributes of that row.

Mathematically, it is defined as the maximum absolute difference between the distribution of a sensitive attribute in an equivalence class [hipervínculo] in the anonymized dataset and the overall distribution of the sensitive attribute in the original dataset. This parameter (t) represents the maximum difference between these distributions. According with [10], in the case of categorical values the *equal distance* will be used, and the *Earth Mover's Distance (EMD)* would be used for numerical attributes.

One of the main advantages of *t-closeness* over *ℓ-diversity* is that it directly measures the degree of similarity between the distribution of the sensitive attribute in the anonymized dataset and the distribution in the original dataset. On the other hand, *ℓ-diversity* has some advantages over *t-closeness*, particularly in cases where the sensitive attribute has low diversity or where the dataset contains outliers or rare values. *ℓ-diversity* can provide stronger privacy guarantees in these scenarios by ensuring that each equivalence class contains a certain number of distinct sensitive attribute values, which makes it harder to identify individual records.

Example 2.2.3. This example shows the t value obtained for table 2.3, which is an anonymized version of table 2.2 using $k = 2$. Since the sensible attribute is a categorical attribute, we will use the equal distance to calculate the distance between the distribution of the SA in each equivalence class regarding the global distribution in the whole table as follow:

- Equivalence class 1: ID 1 and 6. The distance with the distribution of the global dataset is: 0.1667.
- Equivalence class 2: ID 2 and 4. The distance with the distribution of the global dataset is: 0.5.

- Equivalence class 3: ID 3 and 5. The distance with the distribution of the global dataset is: 0.5.

Then, the value of $t = max(0.1667, 0.5, 0.5) = 0.5$.

In this section, all the techniques have been presented to address the case where there is only one sensitive attribute in the database. However, they can also be applied to cases where there are multiple sensitive attributes. For instance, adding the sensible attribute *treatment*.

In this case, the techniques such as in l-diversity and t-closeness, would have to follow different approaches, as presented in [10]. The two following paradigms can be considered:

- Analyze each technique considering a single sensitive attribute and generalize the result [10]. For example, if we have two sensitive attributes and it holds true and it is verified ℓ_1-*diversity* for one of the sensitive attributes, y ℓ_2-*diversity* for the others, we assume that verifies ℓ^*-*diversity* with $\ell^* = min(\ell_1, \ell_2))$.
- Analyze each technique for each sensitive attribute, so that if there are n sensitive attributes $SA = \{S_1, \ldots, S_n\}$, he technique is analyzed for each SA. For the S_i consider the entire initial set of quasi-identifiers (QI), as well as $SA \setminus S_i$ (i.e., all sensitive attributes except the one being analyzed) as additional quasi-identifiers. This means that all the remaining sensitive attributes also become quasi-identifiers, as they may provide additional information if known [10].

Chapter 3

Methodology and development

3.1 Datasets analyzed

In order to analyze the scalability of different anonymization techniques for the use of anonymized data in machine learning and inference processes, this study will examine two openly available datasets. Specifically, in order to work with data that may contain sensitive user information and therefore require prior anonymization to reduce the risk of privacy attacks, one of the datasets under investigation is derived from the banking domain, while the other is from the medical one. These are two areas where privacy concerns are evident, as they deal with personal data that may reveal economic and/or medical information. The following sections present these datasets.

3.1.1 Use case 1: Diabetes Dataset

The dataset focused on the medical field contains personal and sensitive information of different patients. It was collected by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) in the early 1990s in the United States as they were trying to understand the causes of diabetes so that they can develop better treatments and prevention strategies. All the features were collected from patients who had been diagnosed with diabetes [12].

Those patients were asked about their age, sex, bmi, blood pressure, serum insulin level, serum glucose level, serum triceps skinfold thickness, serum hdl-cholesterol level, and serum total cholesterol level among other medical conditions. In addition to this features, we have the prediction variable that is whether or not the patient has diabetes. This set of data has data from 768 patients [12].

Prior to analyzing and assessing various models and anonymizing the datasets, we conducted an initial data exploration to understand the distribution of data. The following plot 3.1 displays the distribution of the selected features that will subsequently be considered during the evaluation of the machine learning models. As it can be seen in the plots of the distributions, we have a natural approach to normal distribution in the age feature, what really depict the population age in the society the medical study took place.

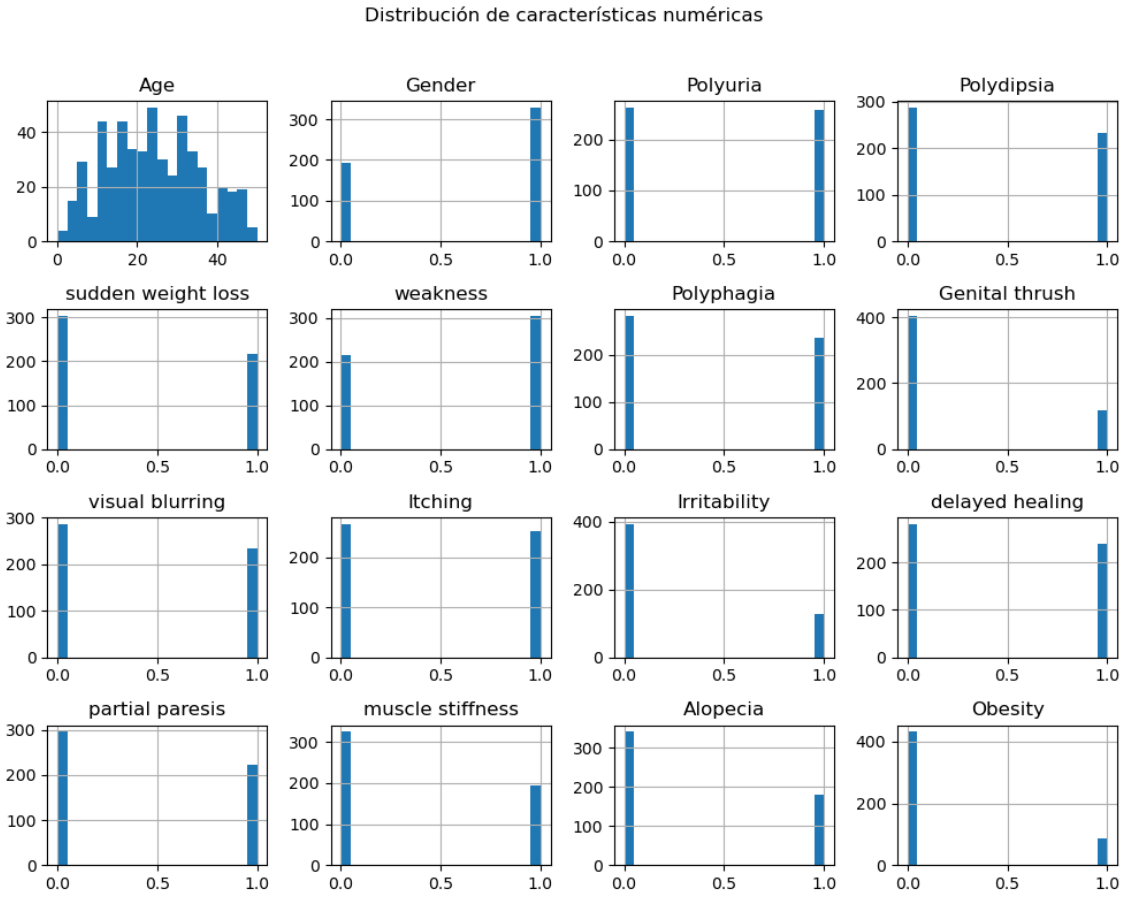


Figure 3.1: Barplots of main features distribution diabetes dataset

We can also see that Polyuria, medical condition where the patient urinates more than seven times per day, is really frequent in the population under study, with almost the 50%. In addition, obesity, irritability and genital thrust is not a very extended condition in the sample.

It's also interesting to analyze the correlation (see Figure 3.2) between the main features. In this way, we can start unveiling some pattern and discard some variables that does not look relevant at all.

In the matrix it is really notable some correlation between several features. For instance, Polyuria and Polydipsia have a positive correlation of 0.6. This makes sense, as Polyuria is the condition or urinate more than seven times per day and Polydipsia is the condition of a huge ingest of water and increased thirst. We can also see that muscle stiffness and visual blurring have a notable correlation, as well as the itching with delayed healing. On the other hand, we can see a strong negative correlation between alopecia and gender, polydipsia and gender, and partial paresis (weakness in extremities).

We also wanted to observe the distribution of the target class to inspect if the dataset could be imbalance. Therefore, the barplot given in Figure 3.3 is analyzed.

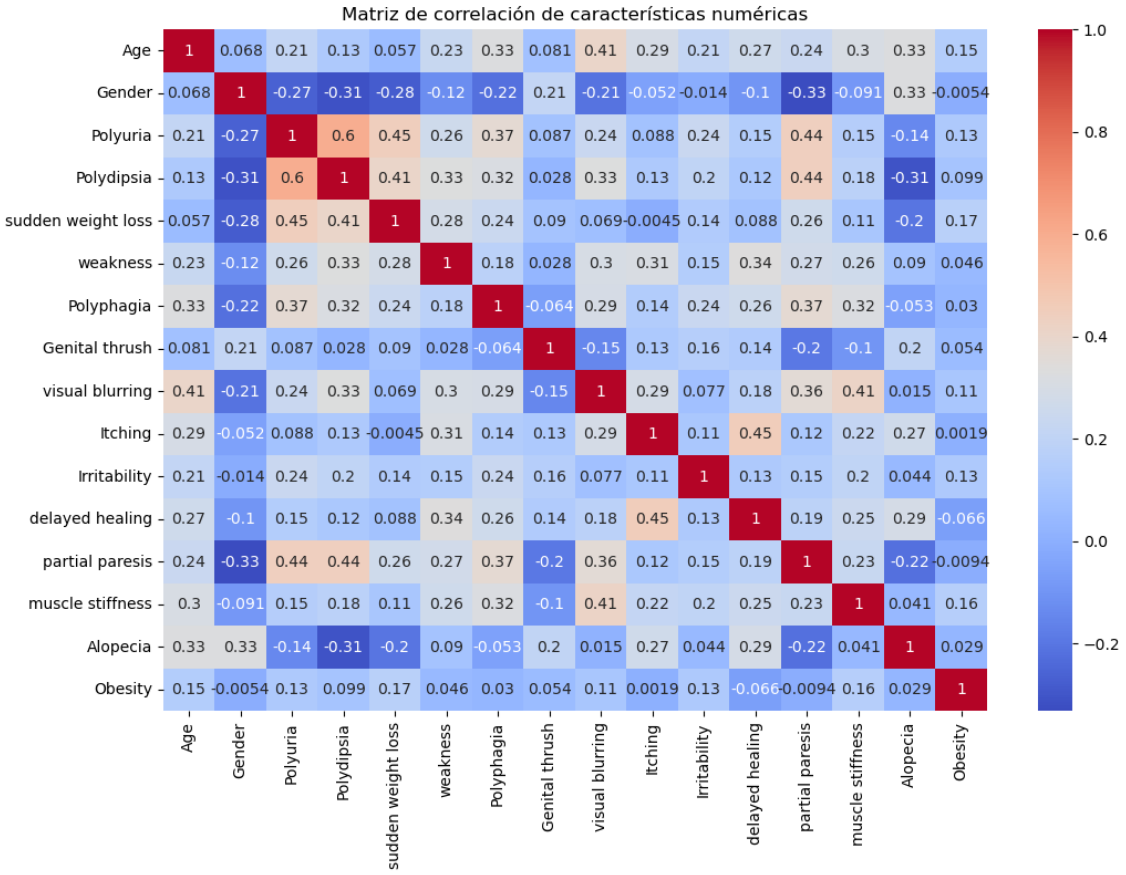


Figure 3.2: Correlation matrix of the main features diabetes dataset

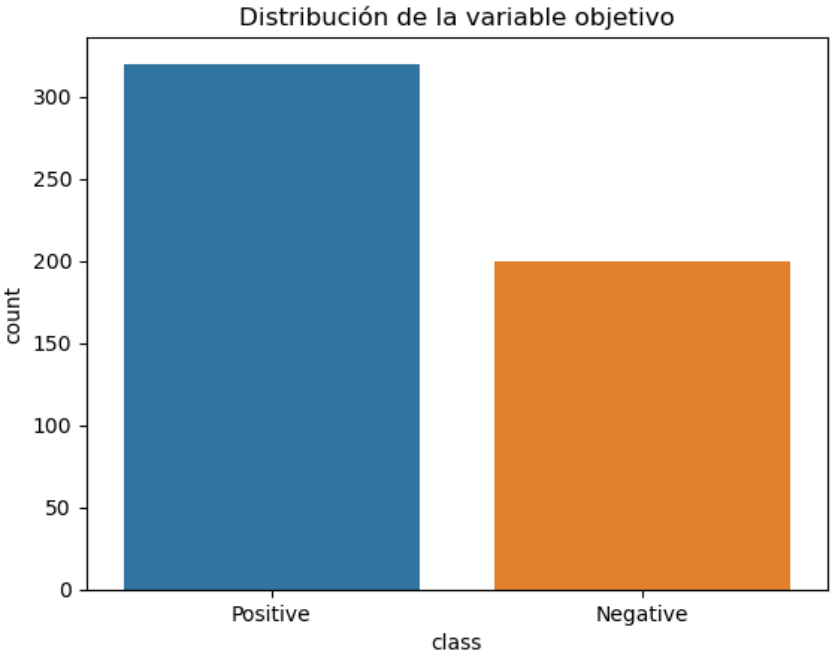


Figure 3.3: Target variable distribution in the diabetes dataset

In view of the graph 3.3, the dataset is not that imbalance, as we have a distribution of 60%-40%. This means that, it is highly unlikely the need of use oversampling or any other technique to solve the issue.

3.1.2 Use case 2: Bank Marketing Dataset

This dataset contains information about customers of a bank. All the data was collected by a Portuguese bank in 2008 through a phone survey. This survey asked customers about their age, job, marital status, education level, and whether or not they had defaulted on a loan. The bank also collected information about the economic conditions at the time, such as the employment variation rate, the consumer price index, and the consumer confidence index. The purpose of the bank was trying to predict whether or not customers would subscribe to a term deposit in their bank [13].

This set contains 23 different features such as age, job, marital status, education level, the date, the euribor of the month the data was acquired and many others. Then we have the objective variable, y, that is whether or not the customer opened a deposit in this bank. In addition, this dataset has more than 40000 samples of different people, what makes it a really complete sample [13].

We have to point out that not all features have been selected for the analysis as not of them are relevant, or cannot be included. For instance, the variable *"Duration"* is dropped as the attribute highly affects the outcome of the predictions. This happens because if the duration is zero or any other low value, it means that the call has been really short and no customers subscribed to the banking product. On the other hand, if the call was long enough, it usually means that the client did subscribe to the proposed product.

Before analysing and evaluating the different models and anonymising the datasets, wecarried out an initial scan of the dataset, trying to understand how the data was distributed in the target class and its distribution. The distribution of some of the features that will then be taken into account for the evaluation of the chosen models is shown in Figure 3.4

As seen in the graphs, the dataset is highly imbalanced. Therefore, before training the models, it is anticipated that an oversampling technique will be required to balance the dataset for training. The majority class is likely to bias the predictions of our model. However, it is essential to observe how this imbalance evolves as we anonymize the data. During the anonymization process, some records will be removed, which may help balance the distribution concerning the target variable.

Another interesting aspect to consider before delving deeper is how the age of customers influences their subscription to a bank deposit. To investigate this, a *violinplot* was created, as shown below in Figure 3.5.

According to the plot, almost all of the customers over 62 years old, aproximately, decided to subscribe to the bank deposit after the call. It also has to be mentioned that the percentage of people over 62 years old is low, being only around the 2%.

We also wanted to explore the correlation between different variables. Consequently, the correlation matrix given in Figure 3.6 has been created.
In addition to the graphs presented in this report, further analysis and data visualization was conducted on the raw initial data. This analysis can be found in the repository of the Master's Thesis. It can be observed in the notebook created, as well as in the report generated using the *Pandas* library, which served as a basis for making decisions regarding data cleaning and preprocessing.

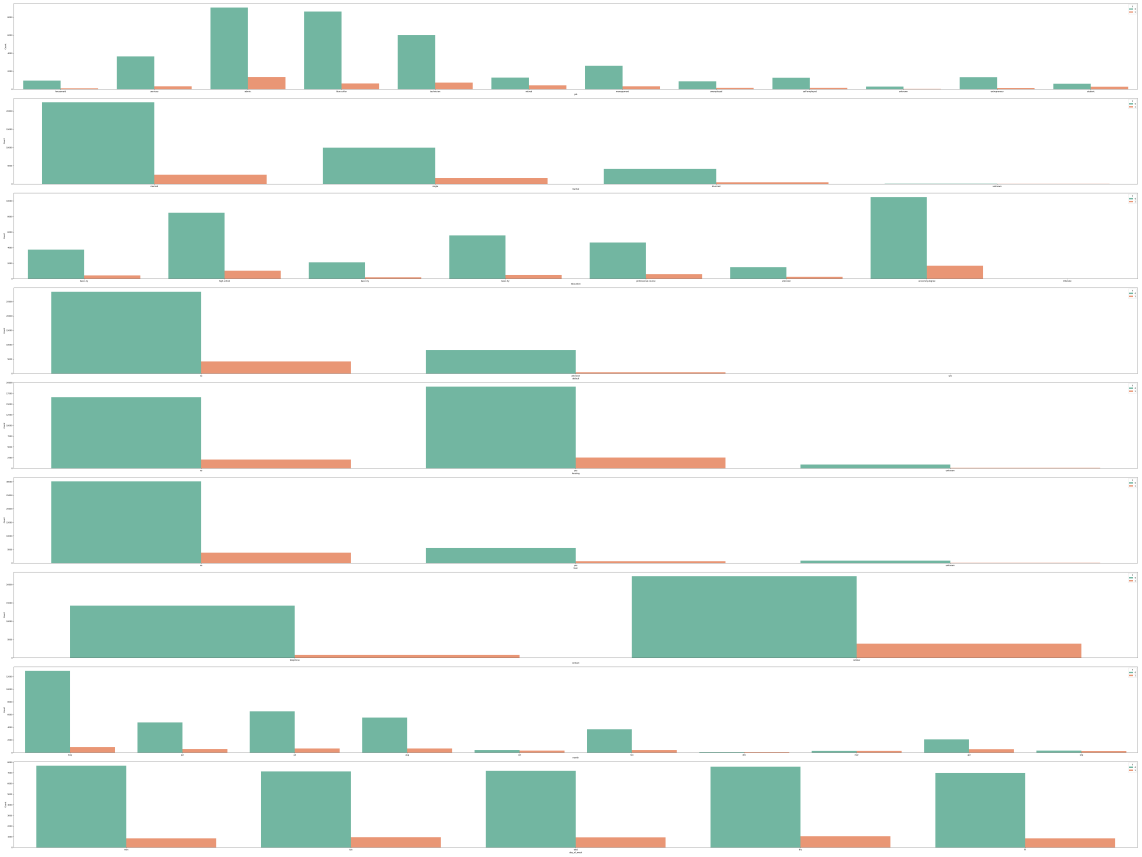


Figure 3.4: Bar plots of main features distribution bank dataset

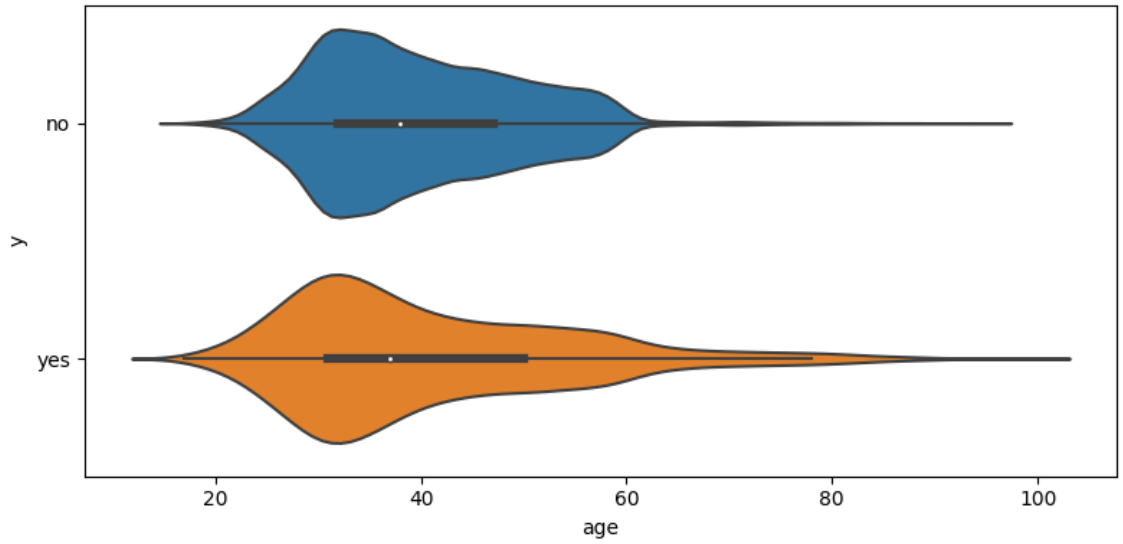


Figure 3.5: Violinplot showing whether people invest in the deposit or not based on age

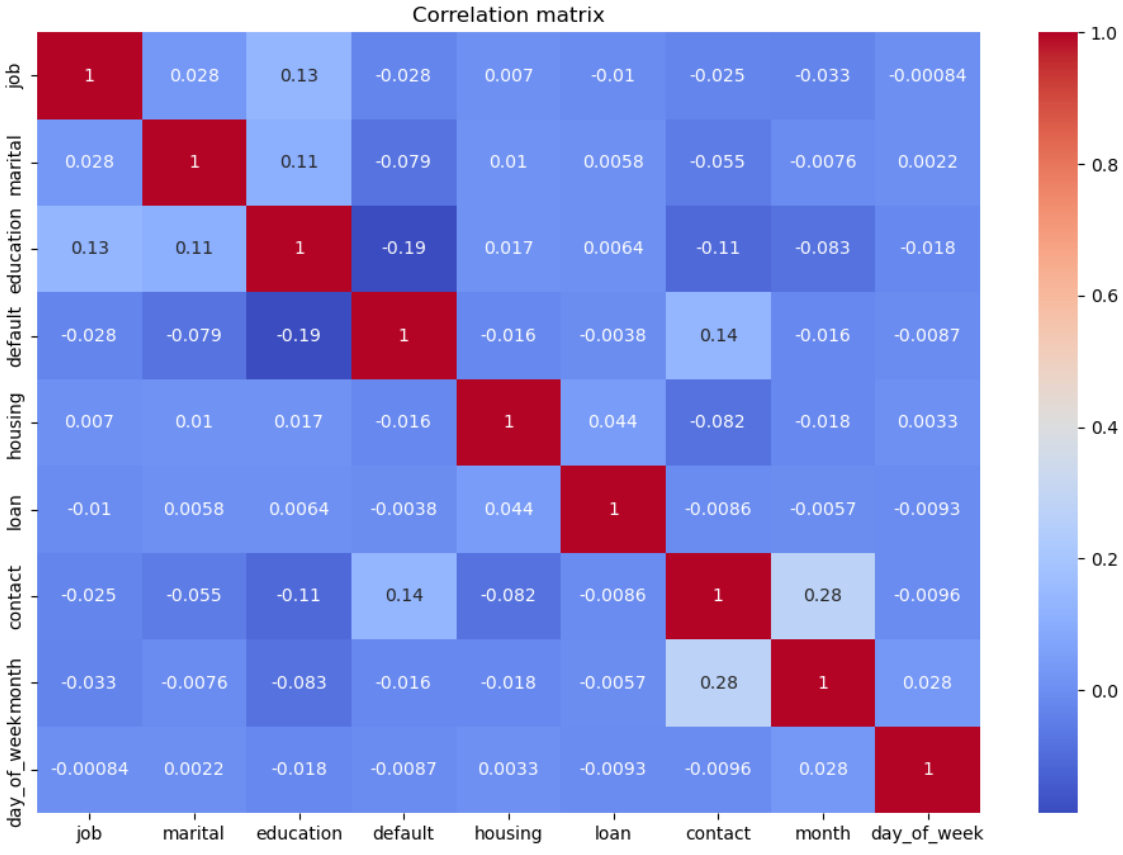


Figure 3.6: Correlation matrix of the main features of the bank dataset

3.2 Models under study

- **Logistic Regression:**

  It is a supervised learning algorithm for binary classification tasks. It is a statistical model that predicts the probability of an instance belonging to a particular class. It calculates the probability of an instance belonging to a particular class using the logistic function, which transforms the weighted sum of input features into a value between 0 and 1.

- **k-Nearest Neighbors:**

  It is a supervised learning algorithm used for classification and regression tasks. It is a non-parametric model, meaning it does not make any assumptions about the underlying data distribution. In k-NN, the prediction for a new instance is based on its proximity to the k nearest neighbors in the training dataset. The "k" refers to the number of neighbors to consider.

  For classification tasks, the class label of the majority of the k nearest neighbors determines the predicted class for the new instance. The distance metric, such as Euclidean distance or Manhattan distance, is typically used to measure proximity. For regression, the prediction is based on the average or weighted average of the target values of the k nearest neighbors. The weighted average assigns higher weights to closer neighbors, considering their distance from the new instance.It can be computationally expensive for large datasets, as it requires calculating distances between the new instance and all training instances. Efficient data structures like kd-trees or ball trees can be used to speed up the search process.

  Other parameter that affects the model's performanceis the k value. A small k can lead to overfitting and higher sensitivity to noise, while a large k can result in over-smoothing and loss of local patterns. The optimal value of k is typically determined through cross-validation or other model selection techniques. On the other hand, it is particularly useful when the decision boundary or relationship between features and the target variable is complex and nonlinear. It can handle both continuous and categorical features. However, it is sensitive to the choice of distance metric and the scale of features, so proper preprocessing and normalization of data are important.

- **Decision Trees:**

  It is a supervised learning models used for both classification and regression tasks (classification in this work). They represent a tree-like structure where each internal node represents a feature, each branch represents a possible feature value, and each leaf node represents a class label or a regression value.

  The construction of a decision tree involves recursively partitioning the data based on the values of different features. The goal is to create splits that maximize the separation between different classes or minimize the variance within each partition

for regression tasks.

During training, the decision tree algorithm selects the most informative feature at each step to split the data. The selected feature and its corresponding split point create two or more child nodes. This process is repeated recursively for each child node until a stopping criterion is met, such as reaching a maximum depth, minimum number of samples per leaf.

- **Suppport Vector Machine:**

  It is a powerful model that finds an optimal hyperplane to separate different classes or predict a continuous target variable. It aims to maximize the margin between the hyperplane and the nearest data points. SVM can handle high-dimensional data and is effective even when the number of features exceeds the number of instances.

  It has various hyperparameters that can be tuned to optimize the model's performance:

  The C hyperparameter in SVM controls the penalty for misclassifications. It determines the trade-off between achieving a wider margin and allowing some training instances to be misclassified. A smaller C value emphasizes a wider margin, prioritizing a simpler decision boundary at the expense of potential misclassifications. On the other hand, a larger C value places more emphasis on correctly classifying each training instance, resulting in a narrower margin and potentially leading to overfitting if the data is noisy or there are outliers.

  The kernel function in SVM allows for the modeling of non-linear decision boundaries by mapping the data into a higher-dimensional feature space. Commonly used kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid. The linear kernel is suitable for linearly separable data and applies the product between the input features and the support vectors. The polynomial kernel enables capturing polynomial relationships between features, with the degree parameter specifying the degree of the polynomial. On the other hand, the RBF kernel is effective for capturing complex, non-linear relationships and uses a Gaussian similarity measure with the gamma parameter. The sigmoid kernel is used for non-linear classification problems, although it is less frequently used compared to other kernel functions.

  The gamma hyperparameter influences the shape and smoothness of the decision boundary when using the RBF or sigmoid kernel. A higher gamma value results in more complex and tightly fitted decision boundaries. It makes the model more sensitive to individual data points, potentially leading to overfitting. On the other hand, a lower gamma value creates smoother decision boundaries, making the model less sensitive to individual points and reducing the risk of overfitting. However, very low gamma values might result in underfitting the data. The optimal gamma value depends on the specific dataset and the degree of complexity in the underlying data distribution.

- **Random Forest:**

  It is an ensemble learning method that combines multiple decision trees to make predictions. It is a supervised learning algorithm used for both classification and regression tasks. Each tree is trained on a random subset of the training data and a random subset of input features. This randomness introduces diversity among the individual trees, reducing the risk of overfitting and improving generalization performance. The fianl prediction is obtained by aggregating the predictions of all the individual trees. Random forests are robust against overfitting and tend to provide accurate predictions.

  During the training, each tree in the Random Forest is built independently. At each split, instead of considering all features, a random subset of features is considered, typically referred to as feature bagging or feature subsampling. This process ensures that each tree has a different perspective on the data, making them less likely to make the same mistakes.

  As in previous models, there are several hyperparameters than can be tunned in order to adjust the model to out case and dataset:

  The number of trees, often denoted as n_estimators, is an important parameter in Random Forest. Increasing the number of trees can improve the model's performance, as it allows for a more stable ensemble. However, adding too many trees can lead to diminishing returns and increase the computational cost. It is important to find an optimal balance between the number of trees and computational efficiency [14].

  The maximum depth of each decision tree controls the depth of the tree. A deeper tree can capture more complex relationships in the data but also increases the risk of overfitting. Other important hyperparameter is the minimum number of samples required to split an internal node. This parameter prevents further splits if the number of samples at a node falls below the specified threshold. Setting a higher value can prevent overfitting, but it may also lead to underfitting if the dataset is small or imbalanced.

- **Adaptive Boosting:**

  It is an ensemble learning algorithm that combines multiple weak learners to create a strong learner. It is effective in binary classification tasks. Each weak learner in the ensemble is trained on a subset of the training data. Initially, each instance is given equal weight, and the weak learner tries to classify the instances correctly. However, after each iteration, the weights of the misclassified instances are increased, directing the weak learners to focus more on these challenging instances.

  The number of estimators is a parameter that determines the number of weak learners to be included in the AdaBoost ensemble. We also have the learning rate parameter.

It controls the contribution of each weak learner to the ensemble. During hyperparameters tuning, different combinations of these values will be evaluated to determine the optimal configuration that maximizes the model's performance.

- **Gradient Boosting:**

  It is machine learning algorithm that combines the strengths of multiple weak predictive models to create a stronger and more accurate one. It is an ensemble method that iteratively, tries to optimize a cost function by adding new models to the ensemble, where each new model tries to correct the mistakes made by the previous models. The algorithm makes use of gradient descent to minimize the cost function.

  In this model, we also need to tune different hyperparameters. One of them is the learning rate. It determines the contribution of each tree to the final prediction. A lower learning rate means each tree has less influence and the algorithm will require more iterations to converge. A higher learning rate can lead to overfitting.

  Other important one is the number of trees selected. As this model builds an ensamble of decision trees, we have to set it to a value. That value will be the iterations to perform (as many as trees). A larger number of trees can increase the model complexity and improve accuracy, but it can also increase the computational cost and the risk of overfitting.

  Some other important features to tune in this model are the same as we have on any other decision tree based model. Such as the maximum depth of the trees, minimum number of samples required to split a node, minimum number of samples required at a leaf node, etc.

- **Artificial Neural Networks:**

  Artificial Neural Networks with dense layers, as the one used in this work, are widely used for various machine learning tasks, including classification, as this case, and regression. Dense layers are fully connected layers in which each neuron is connected to every neuron in the previous layer. However, like any other model, we have to tune several hyper-parameters.

  One of them is the number of layers, as it can have multiple hidden layer between the input and output visible layers. By adding more layers, we can increase the capacity of the network to learn but can lead to overfitting too.

  Then, once we know the layers, we have to set the neurons per layer, including the input and the output one. As the number of neurons per layer increases, the network can learn more complex representations, but it increases the computational costs.

  The activation function is applied to the output of each neuron in a layer and introduces non-linearity to the network. Common activation functions include ReLU (Rectified Linear Unit) and sigmoid. The choice of activation function depends on

the task and the nature of the data.

The learning rate is also really important, as it determines the step size at which the network updates its weights during training. A high learning rate may cause the network to converge quickly but risk overshooting the optimal weights. A low learning rate may require more training time but can improve convergence.

To end up with this model it is important to mention the impact of the batch size. During training, ANNs update their weights based on a subset of the training data, the batches. Its size determines the number of training samples used in each weight update. A smaller batch size introduces more noise but can lead to faster convergence. On the other hand, a larger batch size provides a more accurate estimate of the gradient but may require more memory.

- **Stacked Generalization Ensemble:**

  Stacked generalization is based on, as the name suggests, stacking the output of an individual estimator and using a classifier to calculate the final prediction. This allows us to get the most out of the individual estimators by chaining the output of one with the input of another.

  The hyper-parameters to tune this model depends on the nature of the individual estimator that we chain to create the Stacked Generalization Ensemble.

It should be pointed out that all of the models of both datasets that we are going to use, have been trained using the train-test split method to follow the correct way of training machine learning models. The percentage selected has been 75% for training and 25% for test, as we have done in the masters program.

3.3 Software used and code availability

3.3.1 ARX Data Anonymization Tool

ARX is an open-source anonymization software that uses different techniques to ensure that data privacy is preserved based on some inputs parameters, the anonymity algorithms selected and other several parameter such as the data suppression, generalization and data disruption levels. Once the dataset has been anonymized, we can inspect all the possible solutions that suit the parameters we gave as input. evaluate them and choose the one that fits better according to our criteria.

In this master's thesis, this software will be used to anonymize the datasets that have been introduced in section 3.1. Therefore, I will explain in more detail the basic functionalities of ARX and the ones I have used to carry out this work.

Data suppression

Data suppression is an anonymization technique that allows us to apply different anonymization techniques by removing specific data (e.g. rows) of the dataset to accomplish the desired anonymity level. It should be noted that using this technique may affect the quality of the dataset, hence it has to be applied with certain level of awareness.

We have to define a maximum rate of data suppression allowed when using ARX. When using the data anonymization tool, we can select in the menu the maximum percentage of data to suppress. However, even-though you indicate that it can remove the 100% of the data, the tool is intelligent enough to not to try to remove the data. It will try to eliminate as few records as possible. This option is very useful if we want to preserve an specific high percentage of registries.

Generalization

Generalization is an anonymization technique to reduce the details of data in order to preserve its privacy and anonymity. As we are going to work with sensible data, we need to use this technique. There are different modes of applying this principle based on how we modify the raw data:

- **Basic generalization:** We should replace specific values with broader values. For instance, if we have ages in our data set, we can create intervals instead of including the real age of the people. In this manner it is more difficult to identify the individuals that are present in our data.
- **Hierarchical generalization:** We should organize the data in a hierarchy. In this way, more specific data can be generalized into more generic categories. For instance, instead of saying that a person is married of divorced we could say that all these people has been married. In this way by generalizing this data that may not be relevant for our analysis, we can preserve the anonymity of these individuals.

3.3.2 Computing environment

Throughout this project, a wide range of Python libraries will be used both to process and visualize the data and to build machine and deep learning models. Some of this tools have been used during the master's degree in different subjects, such us *pandas*, *scikit-learn* or *tensorflow* among others. All the libraries used are summarized below:

Basic Python libraries

- **Pandas**: Pandas is a widely used open-source library in Python specifically designed for data manipulation, analysis, and exploration tasks. Developed to work with structured data, it offers a set of data structures and functions that facilitate the handling of datasets. Its primary data structures, namely Series and DataFrame, enable intuitive organization and manipulation of data, resembling tables commonly found in relational databases [15]. In this work it has been used for import data

from various file formats, preprocess and clean datasets, perform data filtering, aggregation, and transformation, as well as execute some analytical operations.

- **Numpy**: Numpy is also an open-source library for scientific computing in Python. It provides the fundamentals for numerical operations, handling of large multidimensional arrays, and a collection of mathematical functions. This library has optimized routines efficiently, making it an essential tool for handling large-scale numerical data and executing high-performance computations. Additionally, Numpy seamlessly integrates with other scientific libraries, enabling collaborative usage and enhancing productivity across various domains, including physics, engineering, machine learning, and data analysis. Its versatility, speed, and extensive functionality make Numpy an indispensable resource for researchers seeking to leverage numerical computing capabilities and develop robust scientific algorithms [16].

Machine learning libraries

- **Scikit-learn**: Scikit-learn is a open-source machine learning library in Python. It provides a suite of tools and algorithms for various aspects of machine learning, including classification, regression, clustering and model selection. Designed with a user-friendly and intuitive API, scikit-learn facilitates the implementation of machine learning workflows. It offers preprocessing techniques for data normalization, feature extraction, and handling missing values in order to have a good data quality. Scikit-learn also has an extensive collection of supervised and unsupervised learning algorithms, enabling users to train and evaluate models efficiently. These algorithms includes some popular techniques that have been used in this work, such as support vector machines, random forests, gradient boosting and adaptive boosting among others. The library incorporates consistent and rigorous evaluation metrics, cross-validation techniques, and hyperparameter optimization methods, allowing researchers to assess and fine-tune their models rigorously as the one used in this analysis, GridSearchCV. Furthermore, scikit-learn promotes its integration with other scientific and data analysis libraries, facilitating comprehensive data exploration and model deployment [17].

- **Tensorflow:** TensorFlow, as all the others libraries mentioned up to know, is an open-source machine learning framework developed by Google. It provides a ecosystem for building, training, and deploying machine learning models. TensorFlow's key strength lies in its ability to handle large-scale computations and deep neural networks effectively. These models can be efficiently executed across multiple devices, including CPUs, GPUs, and specialized accelerators. TensorFlow also provides a wide range of pre-built neural network layers, optimization algorithms, and evaluation metrics, enabling users to construct and train deep learning models with ease compared to many other libraries or frameworks.

  Keras, on the other hand, is a high-level neural networks API that serves as a user-friendly interface for TensorFlow. Keras simplifies the process of building neural networks by providing a simple syntax, enabling a quickly prototype and experimenting. It offers a modular and flexible design, allowing for easy customization

of network layers, loss functions, and optimizers. We could say that Keras is the friendly user API, while Tensorflow is the powerfull backend [18].

- **Shap**: Shap (SHapley Additive exPlanations) is a library that provides interpretability and explainability techniques for machine learning models. Shap uses Shapley values from cooperative game theory to attribute feature importance to individual predictors in a model's predictions. One of the key advantages of Shap is that it is compatibility with a wide range of machine learning models, including the ones we use in this work, such as tree-based models, neural networks and support vector machines. Furthermore, an important feature that Shap offers is the model-agnostic interpretability [19].

Visualization libraries

- **Matplotlib**: It is a library for creating visualizations and plots. It provides a set of functions for generating graphics, ranging from basic line plots to complex 3D visualizations. Matplotlib allows the creation of figures, customize various aspects of the plots, and integrate them seamlessly into scientific papers and presentations, with its intuitive syntax and good quality documentation [20].

- **Plotly**: It is an interactive library used for data visualization in Python. The main difference with the widely known library "Matplotlib" is that Plotly offers you the possibility to interact in real time with the chart generated by adding or removing information, using the hover property of the mouse to display additional information, etc. [21]

- **Seaborn**: This visualization library is based in Matplotlib. However, it offers more functionalities. With seaborn you can illustrate more advanced relationships and patters, such as heat maps, dispersion maps, etc. We can say that it is focused for more statistical purposes.

- **Binder**: Binder is an open-source platform that enables developers to create interactive and reproducible computational environments for sharing and running code. It allows users to turn their code repositories, such as those on GitHub, into executable environments called "Binder repositories". These Binder repositories contain all the necessary dependencies, configurations, and code files required to recreate the computational environment.

  Binder leverages containerization technologies, such as Docker, to encapsulate the computational environment, ensuring that the code runs consistently across different platforms and systems. It supports multiple programming languages and frameworks, making it versatile for a wide range of disciplines and projects.

  This platform promotes open science and reproducibility by enabling researchers to share not only their code but also the entire computational environment, including software dependencies and package versions. This allows others to recreate and validate the results, fostering transparency and facilitating collaboration, as it is done with the data generated for this analysis.

Like any code, a license is required. In this case we selected Apache License 2.0. Apache License 2.0, as they state, it is a *"permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code"* [22].

Last but not least, it should be noted that for the bank dataset, a deployment on the AI4EOSC platform [23] has been used as the computational environment, so that 16 GB of RAM and 8 CPUs have been used, as this dataset had a larger number of records.

Chapter 4

Results and analysis

In this chapter we will analyze the results obtained after applying the battery of machine learning models on the previously mentioned and introduced datasets, as well as those same datasets anonymized in different ways with different parameters.

We will start with the dataset that has the least data, the one from the medical field, that of diabetes prediction. Then we will move on to the banking dataset.

4.1 Use case 1: Diabetes Dataset

The process followed in all the anonymizations of this dataset has been almost identical. After having performed the previous exploration introduced in Chapter 3, we moved on to data processing, where we performed data curation and cleaning. For the data transformations needed for the anonymization techniques, hierarchies have been created as it is going to be explained below.

4.1.1 Anonymization process

The performance of the machine learning models over the original dataset is going to be compared to the ones obtained with the dataset anonymized by the three techniques already explained: *k-anonymity*, ℓ*-diversity* and *t-closeness*. Specifically, this anonymization followed a process of transformations using the ARX Software:

First of all, we upload the original raw data into the software. After this step, we have to define the category of the attribute. In this use case, all the attributes except from the objective class have to be quasi-identifiers. The left attribute, depending the anonymization technique that is going to be implemented will change (only using ARX for technical reasons). If we just apply *k-anonymization*, the objective value has to be insensitive. Nevertheless, if we want to apply ℓ*-diversity* or *t-closeness* it has to be defined as sensitive (which its real nature). This has to be this way because ARX software [24] won't let you set an attribute as sensible without applying a protection technique. Consequently, we have to set it as s sensible attribute only when applying ℓ*-diversity* and *t-closeness*. This will not affect the process of *k-anonymization* as this technique only focuses on quasi-

identifiers.

The following stage is designing the hierarchies needed. Due to the fact that all the features except the age are binary, we just need to create the hierarchy for the age attribute as intervals. We created six different levels without counting the original value of that feature (consider as level 0). In the level 1 we have a five years interval. As we level up, we add multiply by 2 the interval in the previous level. Therefore we can establish the hierarchy such that:

- **Level 0:** The original value for the Age attribute given.
- **Level 1:** Five years interval, from 0 value to 100.
  E.g.: [0-5), [5-10), [10-15), ..., [85-90), [90-95), [95-100)
- **Level 2:** Ten years interval, from 0 value to 100.
  E.g.: [0-10), [10-20), [20-30), ..., [70-80), [80-90), [90-100)
- **Level 3:** Twenty years interval, from 0 value to 100.
  E.g.: [0-20), [20-40), [40-60), [60-80), [80-100)
- **Level 4:** Forty years interval, from 0 value to 100.
  E.g.: [0-40), [40-80), [80-120)
- **Level 5:** Eighty years interval, from 0 value to 100.
  E.g.: [0-80), [80-160)
- **Level 6:** Sixty years interval, from 0 value to 100. **Total transformation**
  E.g.: [0-60), [60-120)

We must take into account that the higher level of anonymization we choose, the more information we lose. Therefore, is important to prioritize a lower level to preserve more information from the original data.

We have applied this generalization hierarchy for the three techniques introduced. Nevertheless, the application of the ℓ-*diversity* technique with $\ell = 2$(the maximum allowed value as the SA is binary) has not been possible as the dataset resulted of the transformation just had nine records left. This means that the dataset didn't met the condition of having the same equivalence class for two different target attributes.

This is really harmful when we really need to anonymize, but really positive for the model, as it is clear that there are some equivalence classes in which no more than one prediction value can be interpreted from the same quasi-identifiers.

For this use case we will have four anonymized datasets to analyze, specifically using the following transformations: ***k=2, k=3, k=5, k=2 and t=0.5.***

Regarding the value of k, different test have been performed and values that are representative in terms of transformations and deleted records have been finally selected.

The value selected for the t parameter in the t-closeness has been selected according to the following criteria. First of all, we used the *pyCANON librar*y [10] to calculate the intrinsic value of **t** in the $k = 2$ anonymized dataset. This value was 0.56. Hence, the t value had to be somehow more restrictive than 0.56, but at the same time not extremely restrictive (low) if we want to have a large enough number of records. Then, we selected 0.5 for t value for *t-closeness*.

4.1.2 Data curation and cleaning

First of all, for data curation we defined a seed to be able to reproduce the results of the models generated. This value is going to be introduced as a parameter in all models to achieve the goal. After that, we load the dataset taking into consideration its original format. Next, we clean the dataset if needed by removing null records.

Another important step that has been performed is the categorization of the variables that were not already categorized by the "nature" of the attribute itself. In this way we only have to handle numbers, what makes it easier for the models training.

However, this is not enough, data also needs to be scaled. By scaling, we are refering to the process of normalizing or standardizing the features of the dataset. It is a preprocessing step that transforms the data to a specific range or distribution. The primary goal of scaling is to bring different features into a comparable level, so that no particular feature dominates the learning process or introduces bias.

It must be highlighted that scaling has to be carried out independently in train and test. In this way we avoid biases, test data won't be affected by the values of train. In addition, we manage to maintain the relative scale, i.e., all characteristics have a similar scale.

After this data curation, now we can proceed to train the proposed model over the train partition generated from the original dataset.

4.1.3 Machine learning models: results and discussion

In this subsection we will analyze and show the results of applying the machine learning models explained above on the diabetes dataset with different anonymization levels and techniques, as well as on the original dataset (raw).

In each case, the hyperparameters evaluated for the different models, as well as the final tuning and the hyperparameters selected as the best for each model will be indicated. Specifically, in this first used case cross validation has been applied with 5 folds. A table with the most relevant data will be shown and comments on them are going to be done. For the best performing models more plots are going to be shown (ROC curves and confusion matrix).

- **kNN:**
  - **Number of neighbors:** 3, 5, 7, 9, 11

  - **Weights:** Uniform, distance
  - **Algorithm:** Ball tree, kd tree, brute

- **DT:**
  - **Max depth:** 3, 5, 7, 9, None
  - **Min samples split:** 2, 5, 10
  - **Min samples leaf:** 1, 2, 4

- **RF:**
  - **Number of estimators:** 50, 100, 200
  - **Max depth:** None, 5, 10
  - **Min samples split:** 2, 5, 10
  - **Min samples leaf:** 1, 2, 4

- **SVM:**
  - **C:** 0.1, 0.25, 0.5, 0.75, 1, 2
  - **Kernel:** Linear, Poly, Rbf, Sigmoid
  - **Gamma:** Scale, Auto

- **AB:**
  - **Number of estimators:** 50, 100, 200
  - **Learning rate:** 0.1, 0.5, 1.0

- **GB:**
  - **Number of estimators:** 50, 100, 200
  - **Learning rate:** 0.1, 0.5, 1.0
  - **Max depth:** 3, 4, 5

- **ANN:**
  - **Hidden Units:** 64, 128, 256
  - **Epochs:** 10, 20, 30
  - **Batch Size:** 16, 32, 64

- **Stacked Generalization Ensemble:**
  - **Random Forest Estimators:** 100, 200, 300
  - **Knn number of neighbors:** 3, 5, 7
  - **Final Estimator C:** 0.1, 1, 10

In this dataset analysis cross validation has been performed with five folds. Cross validation is a technique to evaluate the results of machine learning models
Original dataset:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 5, Weight = distance, Algorithm = brute.
- **DT:** Max depth = 5, Min samples split 2 = , Min samples leaf = 4.
- **RF:** Number of estimators = 100, Max depth: = 10, Min samples split = 2, Min samples leaf = 1 .
- **SVM:** C = 0.5, Kernel =rbf , Gamma = scale .
- **AB:** Number of estimators = 100, Learning rate = 0.5.
- **GB:** Number of estimators = 100 , Learning rate = 1.0, Max. Depth = 5.
- **ANN:** Hidden Units = 25 , Epochs =10 , Batch Size = 16 .

Anonymized dataset with k = 2:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors =3 , Weights = uniform, Algorithm = Ball tree .
- **DT:** Max depth = 5 , Min samples split = 2 , Min samples leaf = 2.
- **RF:** Number of estimators = 100 , Max depth: = None , Min samples split = 10 , Min samples leaf = 1.
- **SVM:** C = 0.5, Kernel = Poly, Gamma = Scame .
- **AB:** Number of estimators = 50, Learning rate = 0.1.
- **GB:** Number of estimators = 200, Learning rate = 1, Max. Depth = 3.
- **ANN:** Hidden Units = 12 , Epochs = 30 , Batch Size = 16.

Anonymized dataset with k = 3:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = Ball tree.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 2.
- **RF:** Number of estimators = 50, Max depth: = None, Min samples split = 2, Min samples leaf = 1.
- **SVM:** C = 0.25, Kernel = rbf, Gamma = scale.

- **AB:** Number of estimators = 100, Learning rate = 0.1.
- **GB:** Number of estimators = 50, Learning rate = 0.1, Max. Depth = 5.
- **ANN:** Hidden Units = 25, Epochs = 30, Batch Size = 64.

Anonymized dataset with k = 2 and t = 0.5

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 3, Weights = Uniform, Algorithm = Ball tree.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 1.
- **RF:** Number of estimators = 50, Max depth: = None, Min samples split = 2, Min samples leaf = 1.
- **SVM:** C = 0.1, Kernel = Linear, Gamma = Scale .
- **AB:** Number of estimators = 50, Learning rate = 0.1.
- **GB:** Number of estimators = 50, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 64, Epochs = 10, Batch Size = 16.

Summary table:

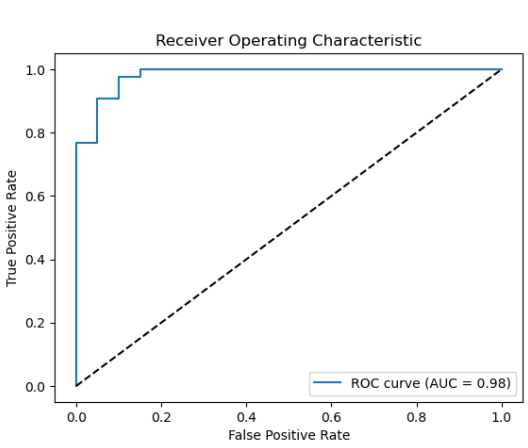
	Raw data		*k=2*		*k=3*		*k=2, t=0.5*	
ML model	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*
LR	0.92	0.96	0.81	0.92	1	1	0.95	0.94
kNN	0.86	0.97	0.93	0.96	0.84	0.98	0.95	0.47
DT	0.84	0.95	0.74	0.82	0.94	0.98	0.89	0.47
RF	0.95	0.98	0.81	0.92	0.94	1	0.89	0.47
SVM	0.92	0.96	0.74	0.84	0.94	1	0.95	0.94
AB	0.90	0.86	0.74	0.71	0.89	0.99	0.89	0.47
GB	0.93	0.99	0.74	0.87	0.89	0.99	0.89	0.47
ANN	0.94	0.98	0.74	0.91	1	1	0.95	0.94

Table 4.1: Accuracy and AUC obtained with the different ML models according with the anonymization technique applied

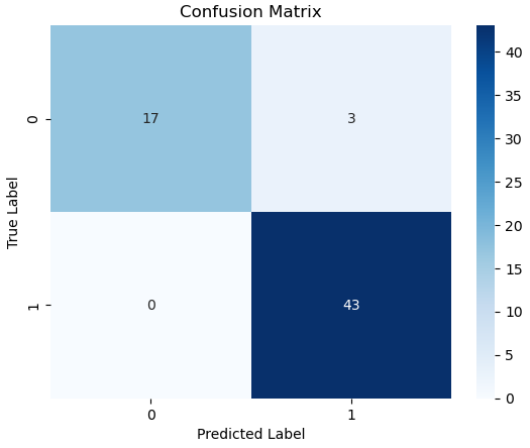
As can be seen in the table that summarizes the results obtained after training the models, there is a huge difference between the results obtained with the raw data and when applying *k-anonymity* with $k = 2$, compared to the rest. This is due to the fact that, after anonymizing the dataset, we have very few occurrences with $k = 3$ and with $k = 2$ together with $t = 0.5$. In these two cases, we have only twenty occurrences with unbalanced distribution in the target class.

However, between the raw data and when applying *k-anonymity* with $k = 2$ we can see a noticeable difference in the performance of the models. There is a decrease in accuracy in almost all the models studied. Another fact that we can extract from this table is the loss of performance in the AUC measure when we become stricter in the anonymization levels.

As for the models, the best performing is obtained with the Random Forest model for raw data and Adaptive Boosting for raw and $k = 2$. Knowing that those two are the best models, in the following we proceed to show the ROC curves and the confusion matrix.

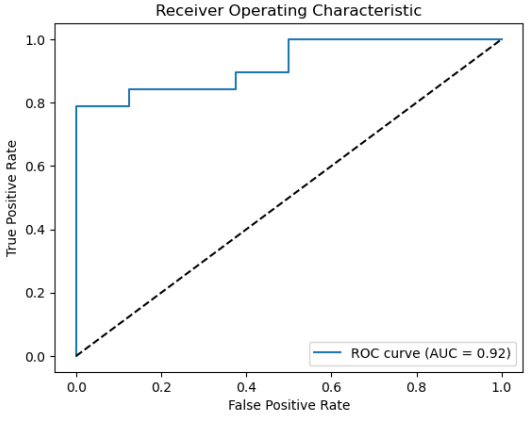


(a) ROC Curve.

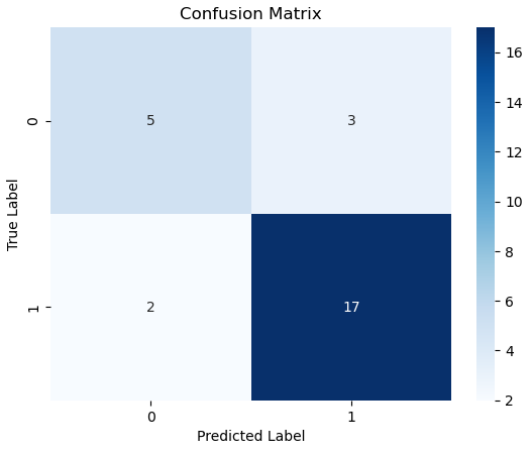


(b) Confusion matrix.

Figure 4.1: ROC curve and confusion matrix of the use of the Random Forest model with the raw dataset.



(a) ROC Curve.



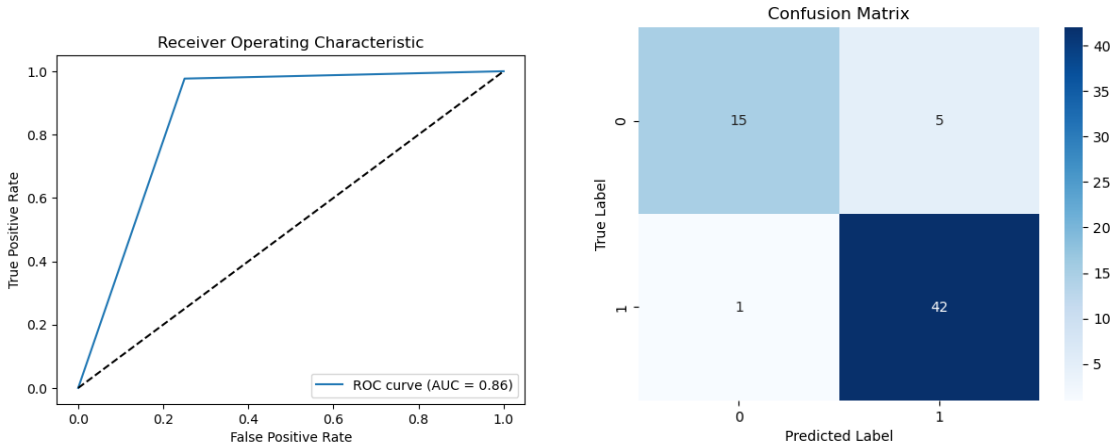
(b) Confusion matrix.

Figure 4.2: ROC curve and confusion matrix of the use of the Random Forest model with the *2-anonymous* dataset.

Comparing both graphs (Figures 4.1 4.5), it can be seen that by anonymizing the dataset by checking $k = 2$, the ROC curve devalues, as well as its AUC. However, as we are

making use of the model that best fits our data, we do not lose as much performance.

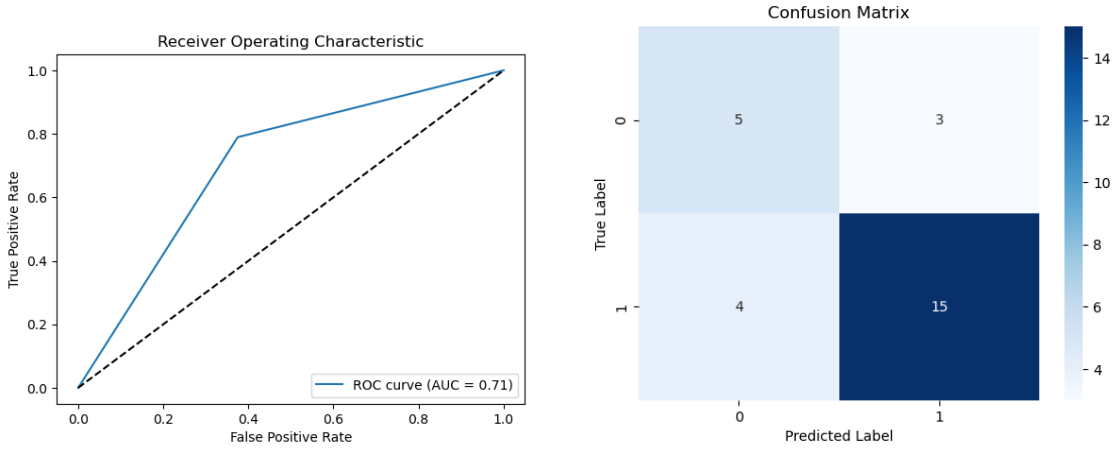
In the following graphs, AdaBoosting plots are going to be shown to illustrate how a model that does not also fit our data greatly devalues the performance of the model by anonymising. We go from having a 90% accuracy to a 74%. In addition to having a 15% lower AUC.



(a) ROC Curve.

(b) Confusion matrix.

Figure 4.3: ROC curve and confusion matrix of the use of the Adaptative Boosting model with the raw dataset.



(a) ROC Curve.

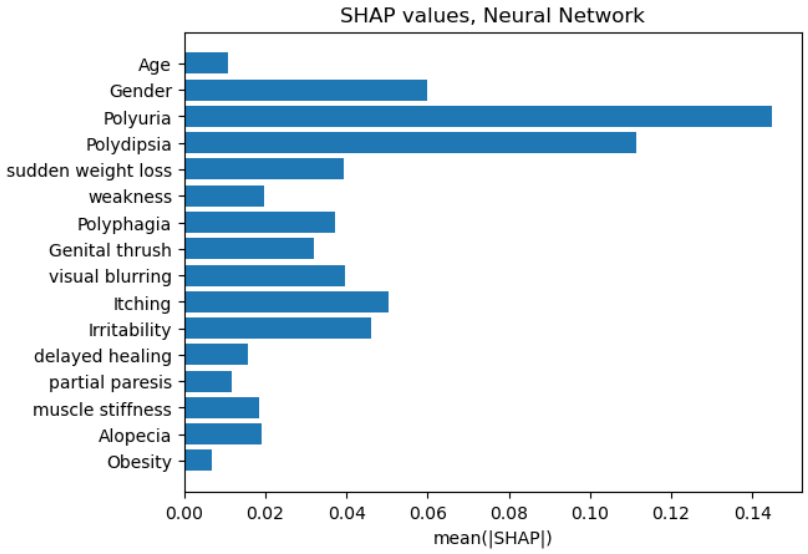
(b) Confusion matrix.

Figure 4.4: ROC curve and confusion matrix of the use of the Adaptative Boosting model with the k=2 anonymized dataset.

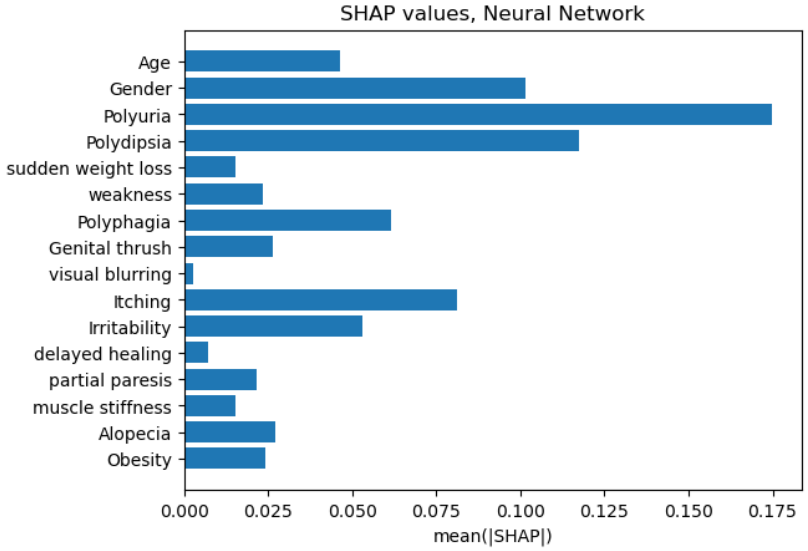
From what is shown in the plots, the devaluation of the model can come from the low number of records present in the dataset. However, it happened the same with the Random Forest approach and it could retain more performance. The main difference between the Random Forest model for $k = 2$ and the Adaptative Boosting for $k = 2$ comes from

the bad diagnosis of two records. They were classied as 0's in the AdaBoost model, when its true value was one. That small change in the predictions can really make a huge impact given the small amount of records present in our dataset after anonymizing.

We also tried to understand a bit better how the models put importance in different features. That's why the following shap plots are presented, using the SHAP values and the librery presented in the previous chapter in the computing environment section:



(a) Raw dataset.

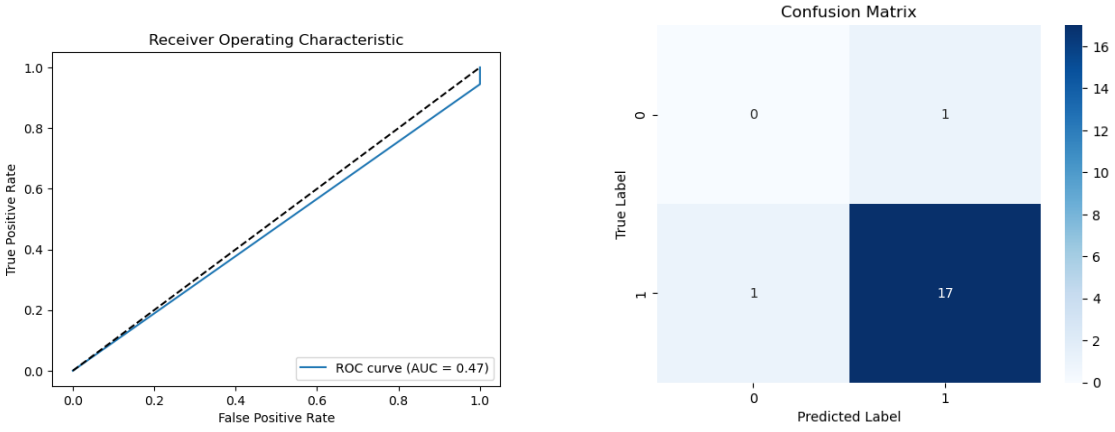


(b) k-anonymized with $k = 2$ dataset.

Figure 4.5: Shap values plot for different features in the Neural Network model for the raw dataset versus the anonimized with $k = 2$ one.

These graphs show the importance of the variables that come into play in the model once it has been trained. It can be seen that the most important features are the same in both cases (Polyuria, Polydipsia and Gender). Nevertheless, there are some big changes in the importance of Visual Blurring and Delayed Healing. This change may come from a elimination of relevant records that were taken into account in the raw dataset whilst they don't exist no more in the k-anonymized with $k = 2$ dataset.

To finish the analysis of this dataset, let's will present the ROC and AUC plot of the dataset with anonymized with $k = 2$ and $t = 0.5$. We are doing it for showing the great loss due to the reduction of records because of the anonymization. The model presented now is the Gradient Boosting, as it also presented a good performance in almost all the datasets studied.



(a) ROC Curve.

(b) Confusion Matrix.

Figure 4.6: Gradient boosting model for $k = 2$ and $t = 0.5$.

As we just have only one value for the 0 class, having only one miss-prediction in that class can make our AUC at least to a 0.5 value. And that is exactly what is happening in this case.

In light of all of the above, we believe that with small datasets like this one is very difficult to have a good classification if you anonymize the dataset in a very restrictive way, as could be intuitively expected. However, with a value of $k = 2$ for *k-anonymity*, although the performance of the models is devalued, the trade off between privacy and classification capability is sufficiently interesting to consider applying this technique, in order to have additional privacy.

4.2 Use case 2: Bank Dataset

The approach for analyze this dataset has been a bit different from the previous use case. This is because we have a dataset with much more records than the other, and quite unbalanced with respect to the target class.

Being a large dataset, lower k values to anonymize the datasets will not have as much impact on the data and we will have to use a higher k to see that devaluation in the models. On the other hand, having an unbalanced dataset may make us think that our accuracy is very good, but in reality our confusion matrix and AUC is quite low.

Therefore, we will have to be cautious and apply different techniques to correct it and improve our classifications.

4.2.1 Anonymization process

As it was explained in the previous use case, the performance of the machine learning models over the original dataset is going to be compared to the ones obtained with the dataset anonymized by the three techniques already explained: *k-anonymity*, ℓ*-diversity* and *t-closeness*.
The hierarchies created for this dataset have been more than in the previous case as the features are not binary in its vast majority. Hierarchies have been applied to the following quasi-identifiers: *Age*, *Job*, *Marital status* and *education*. Below we list the levels of anonymization by hierarchy for each of the quasi identifiers.

- **Age:**
  - **Level 0:** The original value for the Age attribute given.
  - **Level 1:** Five years interval, from 0 value to 100.
    E.g.: [0-5), [5-10), [10-15), ..., [85-90), [90-95), [95-100)
  - **Level 2:** Ten years interval, from 0 value to 100.
    E.g.: [0-10), [10-20), [20-30), ..., [70-80), [80-90), [90-100)
  - **Level 3:** Twenty years interval, from 0 value to 100.
    E.g.: [0-20), [20-40), [40-60), [60-80), [80-100)
  - **Level 4:** Forty years interval, from 0 value to 100.
    E.g.: [0-40), [40-80), [80-120)
  - **Level 5:** Eighty years interval, from 0 value to 100.
    E.g.: [0-80), [80-160)
  - **Level 6:** Sixty years interval, from 0 value to 100. **Total transformation**
    E.g.: [0-60), [60-120)
- **Job:**
  - **Level 0:** The original values for the Job attribute given. It could be any of these: student, unknown, administrative, blue-collar, entrepreneur, management, self-employed, services, technician, housemaid, unemployed and retired.

  - **Level 1:** We have created some classes to group together some of the original jobs possible values: administrative, blue-collar, entrepreneur, management, self-employed, services and technician have been group together into a single class, *Employed*. In addition, housemaid and unemployed have been group together into the unemployed class. Although we believe that housemaid should go into the Employed class, taking into consideration that this is a banking dataset, as it is an unpaid job, we have included it into the hierarchy of unemployed people.
  - **Level 2:** Total transformation. It will totally delete the feature and replaced by an asterisk.

- **Marital status:**
  - **Level 0:** The original values for the Marital status given: single, divorced, married and unknown.
  - **Level 1:** We have created a class to group together some of the original marital status possible values: divorced and single will group together into “not married" class.
  - **Level 2:** Total transformation. It will totally delete the feature and replaced by an asterisk.

- **Education:**
  - **Level 0:** The original values for the Education feature given: basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree and unknown.
  - **Level 1:** We have created a class to group together some of the original education values possible values: basic.4y, basic.6y and basic.9y will group together into “basic" class.
  - **Level 2:** Total transformation. It will totally delete the feature and replaced by an asterisk.

Like we stated previously, we must consider that the higher level of anonymization we choose, the more information we lose. For this use case we will have five anonymized datasets to analyze, specifically using the following transformations: ***k=2, k=5, k=10, k=10 and ℓ=2, and k=10 and t=0.2.***

Regarding the value of k, different test have been performed and values that are representative in terms of transformations and deleted records have been finally selected.

The value selected for the t parameter in the *t-closeness* has been selected according to the following criteria:
First of all, we used the *pyCANON librar*y [10] to calculate the intrinsic value of *t* in the $k = 2$ anonymized dataset. This value was 0.604. Hence, the t value had to be somehow more restrictive than 0.604, but at the same time not extremely restrictive (low) if we want to have a large enough number of records.

At first, we selected 0.5 as t value. However, the transformations and deletions of records were almost insignificant. In other words, the value of t could be further decreased, resulting in a dataset that provides greater guarantees. Therefore, after testing different values, we set the value of t = 0.2, finding a balance between suppression of values and high performance of the future model.

4.2.2 Treatment of imbalanced data

When observing the ROC curves, the AUC values and the confusion matrices we noticed that, despite having a high accuracy, there is a problem with our models. The data we have are unbalanced, since we have much more data from the "No" class than from the "Yes" class. This makes that our models do not learn well due to the unbalance.

To fix it, we have different possible solutions:

- **Penalize the majority class:** Since we learn more with it, by putting a penalty we could make the other class to be taken more into account. However, not all models allow us to make this modification. The logistic regression and the Random Forest implemented does allow us to do it by means of the parameter "class_weight='balanced'". Even so, these models are not the best performing, so it is not an effective measure for the whole analysis

- **Try more complex ensemble methods** that work better for these types of unbalanced datasets. One of the possible implementations is to make use of the "StackingClassifier".

- **Make use of subsampling and oversampling:** That is, if we remove records from the majority class, or synthetically add records from the minority class, we can try to reduce the unbalance and make it more balanced, thus improving the results of our models.

- **Analyze if there are variables that are not relevant in the dataset for most of our models and eliminate them:** By doing this, we can eliminate variables that always take similar values that are causing this unbalance and that do not contribute information to our models.

In this case we choose to apply three of them. In the cases that the "class_weight='balanced'" is possible to implement, it has been implemented. In addition, we have also repeated the whole process of the model training for all datasets with the oversampling approach. Finally a new model that is really suitable for this cases, Stacked Generalization Ensemble [25] has been implemented.

4.2.3 Machine learning models: results and discussion

The same hyperparameters grid search as in the previous case has been applied, but in this case applying cross validation with three fold (as there a larger number of data).

Original dataset:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 9 , Weights = Uniform, Algorithm = kd tree.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 2.
- **RF:** Number of estimators = 100, Max depth: = 10, Min samples split = 2, Min samples leaf = 2.
- **SVM:** C = 2, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 200, Learning rate = 1.0.
- **GB:** Number of estimators = 50, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 25, Epochs = 10, Batch Size = 16.

Anonymized dataset with k = 2:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = kd tree.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 1.
- **RF:** Number of estimators = 100, Max depth: = 10, Min samples split = 10, Min samples leaf = 4.
- **SVM:** C = 2, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 50, Learning rate = 1.0.
- **GB:** Number of estimators = 100, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 128, Epochs = 10, Batch Size = 16.

Anonymized dataset with k = 5:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = brute.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 1.
- **RF:** Number of estimators = 200, Max depth: = None, Min samples split = 2, Min samples leaf = 4.
- **SVM:** C = 2, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 200, Learning rate = 1.0.

- **GB:** Number of estimators = 50, Learning rate = 0.1, Max. Depth = 4.
- **ANN:** Hidden Units = 25, Epochs = 10, Batch Size = 32.

Anonymized dataset with k = 10:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = kd tree.
- **DT:** Max depth = 5, Min samples split = 10, Min samples leaf = 2.
- **RF:** Number of estimators = 200, Max depth: = 10, Min samples split = 10, Min samples leaf = 4.
- **SVM:** C = 0.5, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 200, Learning rate = 0.5.
- **GB:** Number of estimators = 50, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 25, Epochs = 10, Batch Size = 16.

Anonymized dataset with k = 10 and l = 2:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = brute.
- **DT:** Max depth = 3, Min samples split = 2, Min samples leaf = 1.
- **RF:** Number of estimators = 100, Max depth: = 10, Min samples split = 2, Min samples leaf = 4.
- **SVM:** C = 0.5, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 200, Learning rate = 0.5.
- **GB:** Number of estimators = 100, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 64, Epochs = 10, Batch Size = 32.

Anonymized dataset with k = 10 and t = 0.2:

The selected configuration for this dataset regarding the Grid Search was:

- **kNN:** Number of neighbors = 11 , Weights = Uniform, Algorithm = ball tree.
- **DT:** Max depth = 5, Min samples split = 10, Min samples leaf = 2.
- **RF:** Number of estimators = 200, Max depth: = 10, Min samples split = 2, Min samples leaf = 2.

- **SVM:** C = 2, Kernel = rbf, Gamma = scale.
- **AB:** Number of estimators = 50, Learning rate = 1.0.
- **GB:** Number of estimators = 100, Learning rate = 0.1, Max. Depth = 3.
- **ANN:** Hidden Units = 256, Epochs = 10, Batch Size = 16.

Summary table:

When analyzing the results obtained, we saw that the confusion matrix was not as expected in all of our models: there was a class that did not predict correctly due to an imbalance problem in the output class.

Therefore, the same analysis has been performed by applying the different techniques explained in Section 4.2.2:

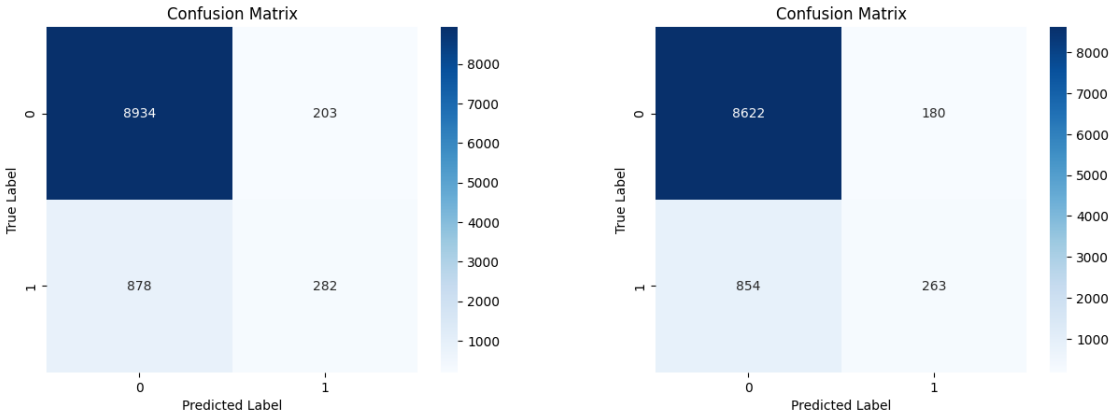
	Raw data		*k=2*		*k=5*		*k=10*		*k=10, ℓ=2*		*k=10, t=0.2*	
ML model	*Accuracy*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*
LR	0.89	0.77	0.89	0.78	0.89	0.77	0.89	0.77	0.89	0.76	0.90	0.75
kNN	0.89	0.72	0.89	0.73	0.89	0.74	0.89	0.73	0.89	0.72	0.90	0.71
DT	0.89	0.74	0.89	0.74	0.89	0.76	0.89	0.79	0.89	0.76	0.90	0.75
RF	0.90	0.79	0.90	0.80	0.89	0.78	0.90	0.80	0.90	0.79	0.90	0.78
SVM	0.89	0.66	0.89	0.66	0.89	0.67	0.89	0.64	0.89	0.63	0.90	0.65
AB	0.85	0.62	0.86	0.56	0.87	0.60	0.87	0.62	0.87	0.60	0.88	0.58
GB	0.89	0.79	0.89	0.80	0.89	0.80	0.89	0.79	0.89	0.78	0.90	0.78
ANN	0.89	0.78	0.89	0.78	0.89	0.78	0.89	0.79	0.89	0.77	0.90	0.76

Table 4.2: Accuracy and AUC obtained with the different ML models according with the anonymization technique applied

	Raw data		*k=2*		*k=5*		*k=10*		*k=10, $ell = 2$*		*k=10, t=0.2*	
ML model	*Accuracy*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*	*Acc*	*AUC*
LR	0.78	0.77	0.79	0.78	0.78	0.77	0.79	0.77	0.77	0.76	0.77	0.75
kNN	0.81	0.68	0.83	0.68	0.84	0.67	0.86	0.68	0.86	0.66	0.87	0.66
DT	0.85	0.64	0.87	0.66	0.87	0.66	0.87	0.66	0.87	0.62	0.87	0.62
RF	0.88	0.76	0.88	0.76	0.88	0.76	0.88	0.75	0.88	0.75	0.88	0.74
GB	0.88	0.75	0.88	0.75	0.88	0.75	0.88	0.76	0.88	0.74	0.89	0.73
ANN	0.88	0.77	0.88	0.77	0.89	0.78	0.89	0.78	0.89	0.76	0.90	0.75
SGE	0.86	0.72	0.86	0.72	0.86	0.73	0.86	0.73	0.86	0.73	0.87	0.72

Table 4.3: Accuracy and AUC obtained with the different ML models according with the anonymization technique applied using **Oversampling.**

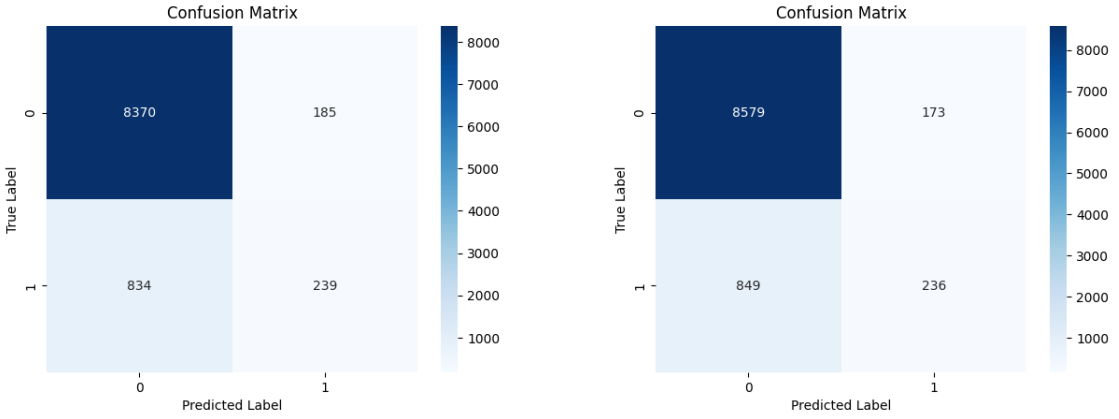
As it can be seen in both tables, the accuracy in the original application of the data models, has a good accuracy value. However, it is a bit tricky, as the model did not perform as expected in any of those cases. We realize when we visualized the followings confusion matrices. Let's show the ones that work better, random forest. If there is any interest in visualizing any other machine learning model that has been implemented, they are at your disposition in the official Github repository of this work. This analysis has been performed for the nine different models already explained.



(a) Confusion Matrix raw data.

(b) Confusion Matrix k=2.

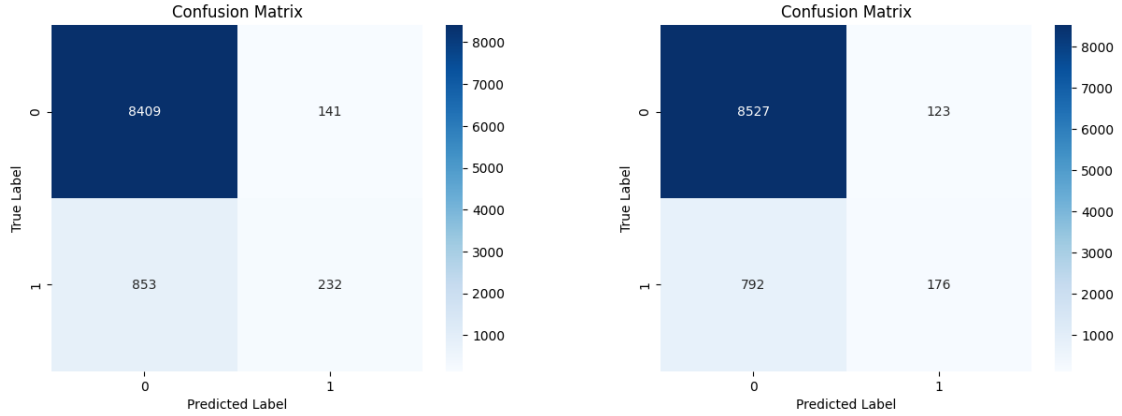
Figure 4.7: Confusion matrices for random forest (raw data and k=2).



(a) Confusion Matrix k=5.

(b) Confusion Matrix k =10

Figure 4.8: Confusion matrices for random forest (k=5 and k=10)

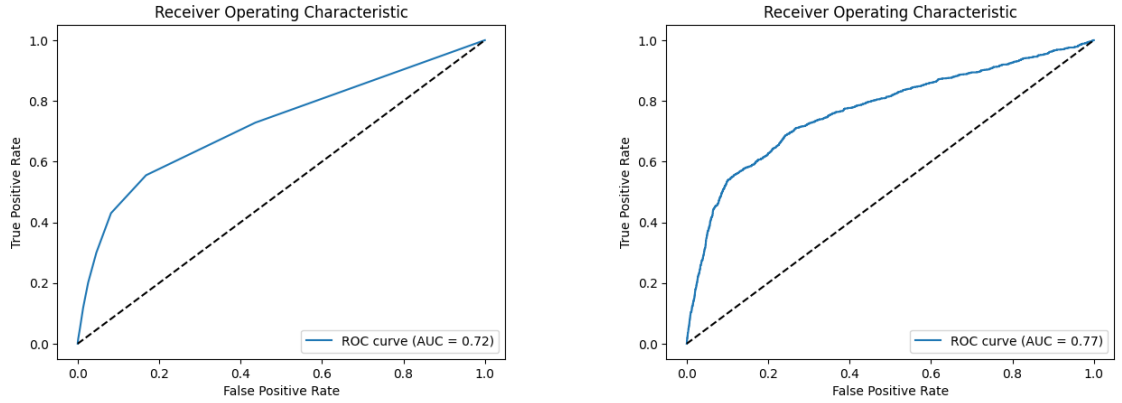


(a) Confusion Matrix k=10, ℓ=2.

(b) Confusion Matrix k =10, t=0.2.

Figure 4.9: Confusion matrices for random forest (k=10, t=0.2and k =10, ℓ=2)

As we can see in the images, all of them rank very poorly in the minority class, despite having a great accuracy. Therefore, the solutions proposed were implemented, achieving the performance shown in table 4.3. Based on that data, we also generated the ROC curves and correspondent confusion matrices. In some of the models, a huge improvement was made compared to the original value. For example, in the Logistic Regression:



(a) ROC Curve Logistic Regression without Oversampling.

(b) ROC curve Logistic Regression with Oversampling.

Figure 4.10: ROC Curve Logistic Regression with and without oversampling (raw).

It has been shown a substantial improvement of the performance of Logistic Regression model when applying oversampling with the SMOTE function. We have use this model, as it is the one with the most significant improvement, specially with the simplicity of this model. This can clearly be shown in the confusion matrices shown below where the number of registers of the minority class previously classified wrongly, has been reduced to more than half. Point out that for Logistic regression we also implemented the penalization in the majority class as already explained.

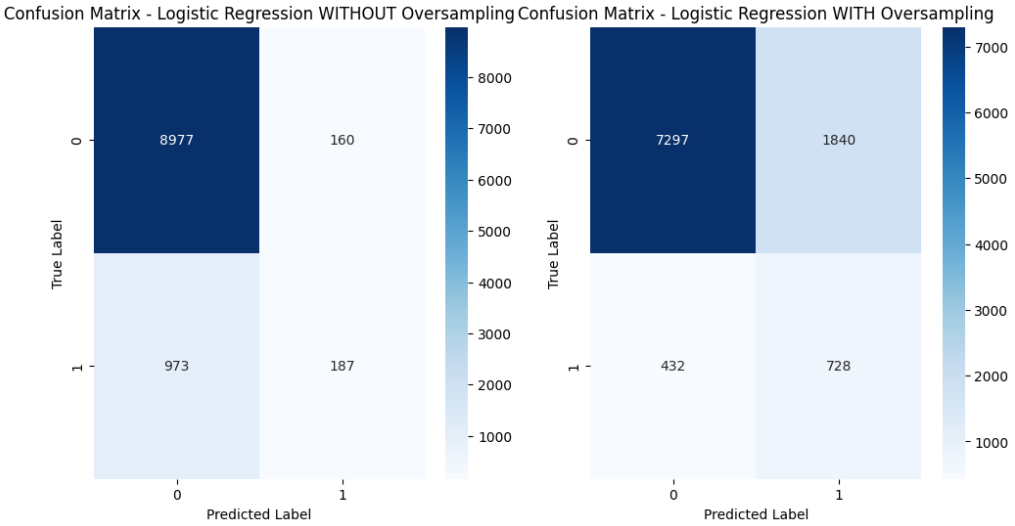


Figure 4.11: Confusion Matrix Logistic Regression with and without oversampling (raw).

The next step is showing the confusion matrices for the Random Forest with and without oversampling, as it is one of the models which performed better in both metrics under study. In this case it is shown for the dataset (it is important to mention that this analysis has been developed for all models previously mentioned and the anonymity levels already established. It can be found in the GitHub repository).

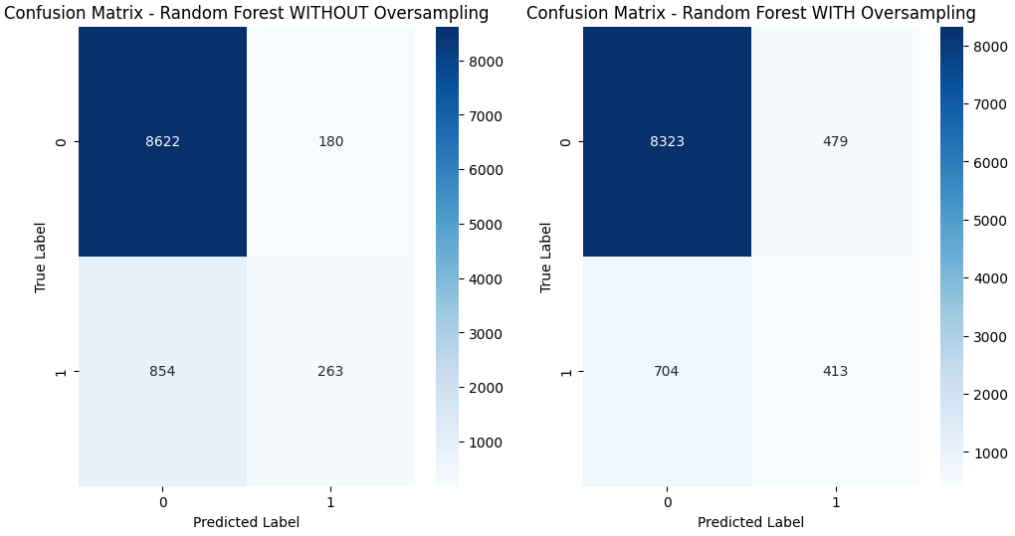


Figure 4.12: Confusion Matrix Random Forest with and without oversampling $(k = 2)$.

With this approach the minority class improves really slightly. The result was a better balance in terms of majority and minority classes than was obtained with Logistic Regression. It is noteworthy that in this case the results are shown again with $k = 2$, obtaining a very similar performance to the previous case (especially with respect to the majority class).

Finally, we are going to present the results obtained with Stacking Classifier, as this model has been specially included for compensate the imbalance problem as it usually performs really good in this cases. According to many studies, there are ”systematic ways to address the imbalanced data classification problem by applying the rule based ensemble learning techniques like bagging, boosting, voting and stacking to build model" [26], being one of them the Stacking Classifier.

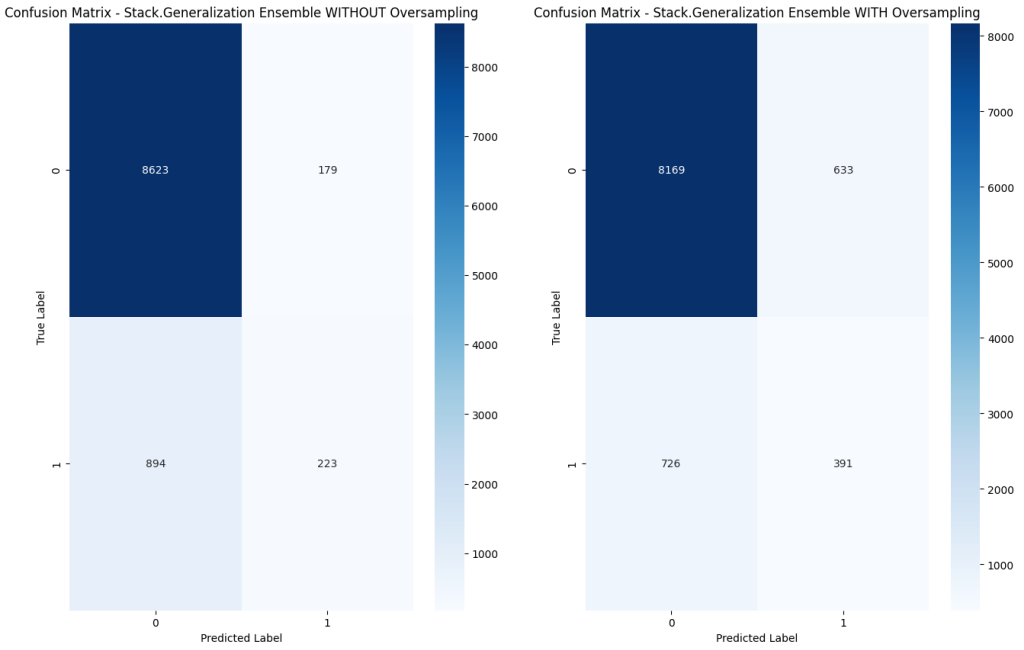


Figure 4.13: Confusion Matrix Stacking Classifier with and without oversampling ($k = 2$).

In datasets with class imbalance, standard classifiers tend to be biased towards the majority class due to its larger representation. The Stacking Classifier mitigates this bias by considering predictions from multiple classifiers, including those that specialize in capturing patterns from the minority class. The ensemble approach allows for a more balanced decision-making process, reducing the impact of the majority class dominance and improving overall performance.

It is important to note that while the Stacking Classifier generally performs well in datasets with class imbalance, its effectiveness may vary depending on factors such as the choice of base classifiers, the quality and relevance of features, and the specific characteristics of the dataset.

In this particular case, it doesn’t perform as bad as most of the original ones without oversamplig. However, it could have been better, regarding the confusion matrix, as it is classifying incorrectly almost the 80% of the minority class. As we have said some sentences ago, it usually performs better, but really depends on the data and the classifiers selected.

Chapter 5

Conclusions and future work

In light of all the above, anonymization techniques have a significant impact on the quality of the data, devaluing the performance models that are applied to them. It is important to mention that, although this is true in general terms, each dataset and its conditions can modify this impact, for better or worse.

In this work we have dealt with two datasets with particularities that are very often encountered when working with open and real-life data. In the first case, the fact of not having enough records has conditioned the models applied to anonymized datasets in a more restrictive way, as many records had to be deleted. This lack of data generates biases in the models, leading to very poor performance for high k values, or when introducing ℓ-*diversity* and *t-closeness* techniques. It is true that with values of $k = 2$, a good performance can be observed in most of the models given, if we take into account that we are anonymizing data in order to preserve the necessary levels of privacy in the domains of these datasets. In this case, the model that has performed the best when anonymizing has been random forest and the Logistic Regression, being better than the popular Neural Networks.

On the other hand, we also have the case of the banking dataset, where we have encountered problems with almost no occurrences of customers who would take out the deposit. This balancing problem can occur in many other use cases that require data anonymization, such as the number of cancer patients, which, fortunately, is usually a low percentage compared to the general population. In these cases, special techniques have to be applied, such as those shown and mentioned throughout this work. The main problem is that, despite having a high accuracy, and even AUC, the false positives or false negatives, proportionally to the class they belong to, are very high. Since we are dealing with sensitive data, these false predictions can have a very high cost in the lives of patients (for example, being diagnosed as not having a disease, when in fact they do). Therefore, these cases have to be analyzed in detail, being a possible line of continuation of this work: the management and anonymization of medical data when we have unbalanced data.

Another conclusion that can be drawn is that some models perform better than others depending on the nature of the data and/or the anonymization technique to which the data has been subjected. The study has shown that random forest has adapted very well to

the banking dataset, and when adding the oversampling technique to try to mitigate the unbalance, logistic regression has obtained the greatest improvement despite being one of the most basic models studied.

In addition to the study of the ML models, an exploratory analysis has been carried out in relation to the importance of each of the variables in the models in the diabetes dataset. It is important to stress that data is everything, and if we do not understand the data we are dealing with from the outset, their nature and how they may behave when certain transformations are applied to them, it is very likely that nothing will come out of it. Shap was implemented in the first dataset, as it is a dataset that is easier to handle than the second one. Using the Python library *shap*, we have been able to visualize which features are more important in the models, and how it varies at the same time as the dataset gets anonymized into a more restrictive way.

As for future work, we are interested in studying more complex datasets, especially in the medical field, since anonymization leads to a large reduction in the number of data. In the case of *k-anonymity*, as 3 types of algorithms have been presented, it is also interesting to analyze how the results scale when applying *k-anonymity* when using the different algorithms, either Mondrian, Data fly or Incognito. Other future consideration, could be expanding the analysis of this work, as we had a good battery of data science models applied to the datasets, but it is very difficult to deeply analyze what happens in all of them, as it exceeds the hours and extension of this project. That is why we have tried to exemplify with just a small amount of all the plots and models generated for this work, the analysis we were making.

All in all, anonymization is a fundamental technique nowadays in several sectors, like the financial and health one. We need to balance the performance with the anonymity levels that we want to establish to ensure that we meet the required legal criteria. This balance has to be subordinated to the models that are evaluated, as, like we have seen in this work, the model really affects how the metrics perform, and scale as the anonymization levels increases.

Bibliography

[1] J. A. Sidey-Gibbons and C. J. Sidey-Gibbons, "Machine learning in medicine: a practical introduction," *BMC medical research methodology*, vol. 19, pp. 1–18, 2019.

[2] N. Sartor, "A brief history of data anonymization," Sep 2019.

[3] J. Sáinz-Pardo Díaz and Á. López García, "Comparison of machine learning models applied on anonymized data with different techniques," *ArXiv*, vol. abs/2305.07415, 2023.

[4] J. Brickell and V. Shmatikov, "The cost of privacy: Destruction of data-mining utility in anonymized data publishing," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, (New York, NY, USA), p. 70–78, Association for Computing Machinery, 2008.

[5] P. Voigt and A. von dem Bussche, "The eu general data protection regulation (gdpr)," 2017.

[6] S. J. Kim, K. J. Cho, and S. Oh, "Development of machine learning models for diagnosis of glaucoma," *PLoS ONE*, vol. 12, 2017.

[7] A. K. Gárate-Escamila, A. H. el Hassani, and E. Andres, "Classification models for heart disease prediction using feature selection and pca," *Informatics in Medicine Unlocked*, vol. 19, p. 100330, 2020.

[8] R. C. Kessler, "The potential of predictive analytics to provide clinical decision support in depression treatment planning," *Current Opinion in Psychiatry*, vol. 31, p. 32–39, 2018.

[9] H. Wimmer and L. Powell, "A comparison of the effects of k-anonymity on machine learning algorithms," in *Proceedings of the Conference for Information Systems Applied Research ISSN*, vol. 2167, p. 1508, 2014.

[10] J. Sáinz-Pardo Díaz and Á. López García, "A python library to check the level of anonymity of a dataset," *Scientific Data*, vol. 9, no. 1, p. 785, 2022.

[11] V. Ayala-Rivera, P. McDonagh, T. Cerqueus, and L. Murphy, "A systematic comparison and evaluation of k-anonymization algorithms for practitioners," *Trans. Data Priv.*, vol. 7, pp. 337–370, 2014.

[12] "Diabetes dataset." https://archive.ics.uci.edu/dataset/34/diabetes.

[13] "Bank marketing dataset." https://archive.ics.uci.edu/dataset/222/bank+marketing.

[14] P. Probst and A.-L. Boulesteix, "To tune or not to tune the number of trees in random forest," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6673–6690, 2017.

[15] The pandas development team, "pandas-dev/pandas: Pandas," feb 2020.

[16] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computitheng in science & engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[17] P. Fabian, "Scikit-learn: Machine learning in python," *Journal of machine learning research 12*, p. 2825, 2011.

[18] "Tensorflow: Api documentation." https://www.tensorflow.org/api_docs.

[19] M. Z. DeMaere, F. M. Lauro, T. Thomas, S. Yau, and R. Cavicchioli, "Simple high-throughput annotation pipeline (shap)," *Bioinformatics*, vol. 27 17, 2011.

[20] P. E. Barrett, J. D. Hunter, T. Miller, and J. chung Hsu, "matplotlib – a portable python plotting package," 2006.

[21] P. T. Inc, "Collaborative data science, montreal, qc: Plotly technologies inc," 2015.

[22] A. Sinclair, "Licence profile: Apache license, version 2.0," *International Free and Open Source Software Law Review*, vol. 2, pp. 107–114, 2010.

[23] "Artificial intelligence for the european open science cloud (ai4eosc)." https://ai4eosc.eu/.

[24] F. Prasser, J. Eicher, H. Spengler, R. Bild, and K. A. Kuhn, "Flexible data anonymization using arx—current status and challenges ahead," *Software: Practice and Experience*, vol. 50, pp. 1277 – 1304, 2020.

[25] "Scikit learn: Ensemble methods." https://scikit-learn.org/stable/modules/ensemble.html.

[26] S. P. Potharaju and S. Marriboyina, "Ensembled rule based classification algorithms for predicting imbalanced kidney disease data," *Journal of Engineering Science and Technology Review*, vol. 9, pp. 201–207, 01 2017.